

9609 296-0
1208 copy
4

This Sheet Has Been Replaced



1

This invention is in the field of digital telecommunications, and is more particularly directed to monitoring and redundancy schemes therein.

5

Background of the Invention

As is well known in the art, significant advances have
10 been made in telecommunications systems over recent years,
particularly in the rate at which information can be
communicated. Modern digital telecommunications systems
and communication media provide very high bandwidth, such
as the 44.736 Mbps data rate provided by the DS-3 data
15 frame standard. Furthermore, conventional fiber optic
cable and systems can provide even higher bandwidth and
data rates by time-division multiplexing of up to twelve
DS-3 lines, providing bandwidth of up to 536.8 Mbps.

20 These extremely high bandwidths now available in
digital telecommunications systems have enabled the
communication of large volumes of data at high speeds.
Since voice channels require very little bandwidth (on the
order of 4 kbps each), a large number of voice channels may
25 now be communicated over a single communication line by way
of time division multiplexing. The available bandwidth now
also enables the communication of large blocks of digital
data from computer-to-computer, as well as digital data
representative of other media such as video displays.

30

Unlike voice transmissions, however, in which some
amount of errored signals can be readily tolerated without
garbling of the message, the successful transmission of
digital data among computers requires high reliability and
35 high quality transmission. Accordingly, conventional
digital cross-connects now provide "performance monitoring"

THIS SHEET HAS BEEN
REPLACED



(commonly referred to as "PM"), by which the error rate of received digital data is monitored by way of cyclic redundancy check (CRC) and other conventional coding techniques. Such performance monitoring is used to ensure the desired grade of service desired by those telecommunications customers paying premium tariffs for high quality and low error rate communications.

Conventional telecommunications systems also generally provide some amount of redundancy so that failure of a telecommunications line or network element does not result in the loss of the communicated message. Conventional telecommunications systems with performance monitoring have implemented certain alarm conditions by which a human operator is alerted to events such as "loss of signal" and to error rates exceeding various thresholds. In response, the operator can manually switch to a redundant line to again enable communication of the digital data in the system. Of course, the procedure of generating an alarm condition and the manual switching of input/output ports to other lines cannot be effected quickly.

By way of further background, conventional fiber optic terminals (commonly referred to as FOTs) have implemented 1:1 redundancy for the fiber optic lines in a system, with some amount of automatic switching. According to this 1:1 redundancy scheme, the overhead portion of the bandwidth is monitored to determine if a loss-of-signal ("LOS") or alarm indication signal ("AIS") condition is being received. In these FOT 1:1 redundancy schemes, upon receipt of an LOS or AIS signal, the FOT will automatically switch its transmission to the other of the two fiber optic lines, enabling transmission of the data despite the failure of the first fiber optic line.



Figure 1 illustrates the bandwidth in a conventional 1:1 redundant fiber optic lines 1E, 1W, as received by a conventional fiber optic terminal (FOT) 3. Each of fiber optic lines 1E, 1W in this example are of the OC-12 type, and as such bidirectionally communicate twelve DS-3 lines in time-domain multiplexed fashion. The DS-3 paths occupy much of the available bandwidth, as shown in Figure 1. According to conventional standards, a portion of the remainder of the bandwidth communicated by fiber optic lines 1E, 1W is reserved for line data, such as framing and synchronization signals, frame identifying signals, and also signals such as LOS and AIS which indicate a line failure somewhere in the system. This line data portion of the bandwidth is also referred to as overhead, as it does not carry any traffic data. Fiber optic terminal 3 is operable to multiplex and de-multiplex the data to and from one of fiber optic cables 1E, 1W, for communication with twelve individual DS-3 lines.

According to the conventional 1:1 redundancy scheme, conventional FOT 3 monitors the line data and switches communication between fiber optic lines 1E, 1W upon receipt of a signal indicating a line failure. However, many conditions other than the LOS and AIS conditions are unacceptable to the telecommunication customer that is demanding high quality communication, particularly where computer data is to be transmitted and received. These other conditions may not be of such degree as to cause loss of an entire line (i.e., all channels or "paths" being communicated along the fiber optic line), but may be due to a hardware failure for a single path within the line. Examples of these other conditions include loss-of-frame ("LOF"), and "soft" errors in which a particular path has experienced a high rate of errored seconds. The mere monitoring of line data by conventional FOT 3 provides no visibility into these other path-related conditions, and

This Sheet Has Been Replaced



4

as such the redundancy scheme of Figure 1 is unable to provide the high level of protection desired by certain customers. Conventional FOT 3 thus only provides line protection, and as such requires doubling of the available
5 fiber optic capacity from that required for traffic only, to accomplish such line protection. As such, the utilization of system fiber optic cable is relatively low in such situations.

10 It is therefore an object of the invention to provide a method and system for providing facility protection for a digital telecommunications system, which monitors path data and provides automatic switching on both hard and soft error conditions.

15

It is a further object of the present invention to provide such a method and system to provide such protection where the switching time between the primary and secondary lines for a failing path is extremely short, such as on the
20 order of 50 msec or less.

It is a further object of the present invention to provide such a method and system which may be readily implemented into existing cross-connect installations.

25

It is a further object of the present invention to provide such a method and system in which the possibility of blocking is much reduced.

30 It is a further object of the present invention to provide such a method and system in a distributed manner in a digital cross-connect.

It is a further object of the present invention to
35 provide such a method and system in which the soft error rate thresholds may be modified.

THIS SHEET HAS BEEN
REPLACED

It is a further object of the present invention to provide such a method and system in which the utilization of fiber optic cables may be increased dramatically.

5

It is a further object of the present invention to provide such a method and system in which transmission media of different types (e.g., microwave and light wave) may provide protection for one another.

10

Other objects and advantages of the present invention will become apparent to those of ordinary skill in the art having reference to the following specification together with the drawings.

15

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED

Brief Description of the Drawings

Figure 1 is an illustration of the bandwidth, line and path data communicated in a conventional fiber optic terminal.

Figure 2 is an electrical diagram, in block form, of a telecommunications system in which the preferred embodiments of the invention may be used.

Figure 3 is an electrical diagram, in block form, of a digital cross-connect system in which the preferred embodiments of the invention may be implemented.

Figures 4a and 4b are matrix diagrams of a digital cross-connect, illustrating the operation of a first preferred embodiment of the invention.

Figure 5 is a flow chart illustrating the operation of the first preferred embodiment of the invention.

Figures 6a through 6c are matrix diagrams of a digital cross-connect, illustrating the operation of a second preferred embodiment of the invention.

Figure 7 is a flow chart illustrating the operation of the second preferred embodiment of the invention.

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED

Summary of the Invention

The invention may be implemented into a digital telecommunications system to provide facility protection by way of distributed fault sensing and facility switching at the digital cross-connects. According to the present invention, each line has three ports (head, east and west) assigned thereto, with the east and west ports on the system side to be protected, and providing 1:1 redundancy. AIS, LOF and degraded signal conditions are monitored on received paths at the east and west ports (i.e., facilities), and the facility is blind switched to the alternate redundant (east or west) port upon receipt of such conditions. The degraded signal thresholds are preferably different from those that cause measured events in conventional performance monitoring. According to a first embodiment of the invention, the east and west ports for a facility are on the same logical group in the cross-connect, allowing the bridging to be done in the transmit end (3rd stage) in a non-blocking manner. According to a second embodiment of the invention, the logical group limitation is overcome by moving fault monitoring to the transmit side of the head-end port, in accordance with an algorithm that minimizes blocking.

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED

Detailed Description of the Preferred Embodiments

Referring first to Figure 2, an example of a digital telecommunications system into which the preferred embodiments of the invention are implemented will be described. It is to be understood that the system of Figure 2 is presented by way of example only, and that the present invention may be incorporated into other system configurations, and provide the same level and importance of benefits and advantages as provided in the system of Figure 2.

The system of Figure 2 is a long-distance telecommunications system, in which digital cross-connects 2 and 4 bidirectionally communicate with one another over a distance that may be on the order of hundreds of miles. Digital cross-connects 2, 4 are of conventional construction, and are preferably 1633 SX digital cross-connects manufactured and sold by Alcatel Network Systems, Inc. While Figure 2 illustrates single ports in each of cross-connects 2, 4 in redundant communication with one another, it is of course well understood that each of cross-connects 2, 4 support many ports, for example up to on the order of 2048 ports. The ports served by cross-connects 2, 4 may be in communication with one another in the same manner as that shown in Figure 2, or may communicate with other lines to other cross-connects in the overall telecommunications system within which cross-connects 2, 4 are installed.

In the exemplary system of Figure 2, digital cross-connects 2, 4 communicate with one another along redundant routes in a bidirectional manner. A first bidirectional route (EAST) is between I/O port 6E of cross-connect 2 and I/O port 8E of cross-connect 4, and includes DS-3 lines 17



connected to each of ports 6E, 8E. The EAST route also includes fiber optic terminals (FOTs) 10, having fiber optic cable 13 connected therebetween; repeater 11 is provided within fiber optic cable 13 in the conventional manner. FOTs 10 may be of conventional construction, such as described hereinabove relative to Figure 1, except that the present invention allows elimination of the 1:1 redundancy line protection function. As is well known, modern fiber optic cable is able to communicate data at a capacity far exceeding a single DS-3 line; accordingly, multiple DS-3 lines 17 may be received by each FOT 10, for communication along fiber optic cable 13 in time-division multiplexed fashion.

The second route (WEST) in the system of Figure 2 is between ports 6W and 8W of cross-connects 2, 4, respectively. In this example, the WEST route includes microwave terminals 14 that receive DS-3 lines from cross-connects 2, 4, with microwave transmission line 15 connected therebetween. As such, the present invention may be used with transmission media of different types, so that microwave transmission can provide redundancy to fiber optic cabling; of course, the redundant routes may both be fiber optic, microwave or any other conventional medium.

The exemplary system of Figure 2 is an "open" system, in that communication is not limited to traffic only between cross-connects 2, 4. For example, DS-3 lines 17' may be received by FOTs 10 and microwave terminals 14 from digital cross-connects or other sources in the system, as shown in Figure 2. In addition, other conventional network elements, such as add/drop boxes and the like, may of course also be resident in the system of Figure 2.

The ports of digital cross-connects 2, 4 on the opposing sides from the EAST and WEST routes are connected



to local systems, in the conventional manner; these ports are referred to in the art, and hereinbelow, as the "head" ports. Such connection may be by way of DS-3 lines or other conventional communication channels.

As illustrated in Figure 2, transmission is effected by each of cross-connects 2, 4 from its head port to both the EAST and WEST ports, effecting a "head-end bridge" as is well known in the art. Each of cross-connects 2, 4 receive communication from only one of its EAST and WEST ports at any given time, however. In the example of Figure 2, communication is received by cross-connects 2, 4 from its EAST ports 6E, 8E, respectively; the WEST route is the secondary route in this example. The present invention performs blind receive-end switching within cross-connects 2, 4 in the event of a monitored hard or soft path fault or error condition, as will be described in further detail hereinbelow.

Referring now to Figure 3, the construction of cross-connect 2 according to the preferred embodiments of the invention will now be described (cross-connect 4 is preferably similarly constructed). As noted above, the preferred construction of cross-connect 2 is in such a manner that the computing and switching processing capability is widely distributed, as in the 1633 SX cross-connect system manufactured and sold by Alcatel Network Systems, Inc. The construction of cross-connect 2 shown in Figure 3 thus corresponds to the 1633 SX cross-connect system.

Cross-connect 2 includes administrative processing unit (APU) 12 which is the central data processor within the system. APU 12 is directly connected to conventional storage and input/output functions such as random access memory 21, disk storage 23, tape drive 25, and user



interface 19. APU 12 is also connected to bus BUS, which may include one or more information paths within the system. The function of APU 12 is to administer the operation of cross-connect 2 at a relatively high level, and to provide user control and visibility relative to the switching operations provided by cross-connect 2.

Cross-connect 2 further includes many distributed processors connected to bus BUS, illustrated in Figure 3 as interface processing units (IPUs) 16, 18. Each of IPUs 16, 18 is a relatively complex data processing unit, including a conventional programmable microprocessor, or alternatively custom logic, of adequate complexity and performance to perform the functions described hereinbelow. Each of IPUs 16 are for controlling input and output functions relative to the bidirectional DS-3 ports, and thus control the operation of the head, east and west ports of cross-connect 2 when installed into a system as illustrated in Figure 2. To accomplish this function, each of IPUs 16 is associated with and bidirectionally coupled to one of first/third matrix stage units 20. In this example, each of first/third stages 20 includes a portion of the switching matrix (as will be described hereinbelow), and the appropriate transceiver circuitry to provide four DS-3 ports.

Cross-connect 2, for these embodiments of the invention, serves as a digital telecommunications switch in a manner realized according to a modified version of the well-known Clos matrix. A preferred version of the modified Clos matrix useful in cross-connect 2 is described in copending application S.N. 882,920, filed May 14, 1992 in the name of E.K. Bowden, entitled "Methods and System for Immediately Connecting and Reswitching Digital Cross-connect Networks", assigned to Alcatel Network Systems, Inc. and incorporated herein by this reference. According



course, the present invention is also capable of protecting against line failures.

The operation of cross-connects 2, 4 in providing the facility protection will now be described in detail relative to the preferred embodiments of the invention. It should be realized that a system including a single one of cross-connects 2, 4 having the facility protection schemes described hereinbelow, with conventional cross-connects, switching systems and the like in the remainder of the network, may also take advantage of the present invention. As such, a customer desiring the high speed facility protection provided by the present invention may implement the same on his receiving cross-connect, without regard to the type of switches that are elsewhere in the system, so long as each facility to be protected is also received at a redundant port.

Zero Blocking Fast Facility Protection

Referring to Figures 4a, 4b, and 5, the operation of digital cross-connect 2 in providing facility protection according to a first embodiment of the invention will now be described in detail. The method is preferably implemented in a computer program according to which APU 12 and IPUs 16, 18 operate. It is contemplated that one of ordinary skill in the art will be able to readily implement the method described hereinbelow into a digital cross-connect without undue experimentation, based on the following description.

Figure 4a illustrates, in matrix form, the switch routing in cross-connect 2 with the EAST route as primary for the example of a single bidirectional path. On the transmission side, the path is received from the local side



of cross-connect 2 by input head port H_1 of first/third stage 20₀ (serving as the first stage of the Clos matrix for data from local to system), and is communicated through center stage 22₁ to first/third stage 20₂. First/third stage 20₂ serves as the third stage of the Clos matrix for data transmitted from local to system, and in this embodiment of the invention broadcasts to both east and west output ports E_0 and W_0 for transmission of the path associated with input head port H_1 . According to this embodiment of the invention, both east and west output ports E_0 , W_0 for a given path are controlled by the same first/third matrix stage 20₂ (and, in this example, by the same ASIC), in order to avoid a blocking condition because of the broadcast.

Accordingly, the path data received from the local side at input head port H_1 is switched to, and broadcast from, both east and west output ports E_0 , W_0 at all times. This third-stage broadcast enables the blind receive and switching used in the facility protection scheme according to the present invention.

On the receive side for this path, first/third stage 20₁ receives a facility from the system at active east input port E_1 (the WEST route being the secondary route in the system at this time). The facility is routed via center stage 22₂ back to first/third stage 20₁, and presented to output head port H_0 . It should be noted that, in this example, output head port H_0 is on the same first/third stage 20₁ as east and west input ports E_1 , W_1 , as each first/third stage 20 can serve multiple ports (whether head, east or west ports); however, according to this embodiment of the invention, the output head port H_0 may be on a different first/third stage 20 from that of the east and west input ports E_1 , W_1 with which it is associated. According to this embodiment of the invention, first/third



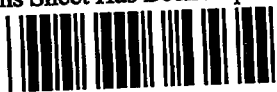
stage 20₁ is operable to switch the facility from being received from east input port E₁ to being received from west input port W₁, upon the occurrence of a hard or soft failure event as will be described hereinbelow.

Of course, while Figure 4a illustrates a single bidirectional path through cross-connect 2, similar switching and connection is made through cross-connect 2 for each of the active paths and ports that it is serving. In the example of the 1633 SX cross-connect, up to 2048 ports and paths are handled in parallel.

Figure 5 is a flow chart illustrating the operation of the facility protection monitoring and switching functions performed by cross-connect 2 according to this first embodiment of the invention. The functions illustrated in Figure 5 and described hereinbelow are performed by each IPU 16 for facilities received at its ports from the system, i.e., for data received at the east and west input ports E₁, W₁ of its associated first/third stage 20. Accordingly, each IPU 16 is able to monitor the facilities received at the input ports in its associated first/third stage 20, and to enable the switching necessary to provide protection.

According to this embodiment of the invention, both hard and soft errors or failure conditions may enable protection switching to the secondary redundant input port. Referring to Figure 5, the monitoring for hard errors is performed beginning with process 24, in which each of the active ports E₁, W₁ associated with an IPU 16 is scanned to determine if a hard error signal or condition has been received. The scanning performed by process 24 is performed of the portion of first/third stage 20₁ (in the case of input ports E₁, W₁) at which the signal is received.

This Sheet Has Been Replaced



16

In this example of the invention, the hard error conditions upon which switching may be enabled are as follows:

AIS : Alarm indication signal

LOS : Loss of signal

LOF : Loss of frame

LOS and LOF signals are well known path and line data signals by which loss of a facility is detected. As is well known in the art, the AIS signal is generated by a network element upon its detecting of an alarm condition from an "upstream" network element; this causes only a single alarm signal to be generated to facilitate corrective action, as the location of the fault can be more readily detected than if each network element passed along an alarm signal. Decision 25 determines if any one of the hard error signals is detected by the scan of process 24; if so, control passes to decision 31 to effect facility protection switching as will be described below. If no hard error is detected in decision 25, wait state 26 is entered for the particular path, until the designated period (e.g., 20 msec) elapses, at which time the scan of process 24 is initiated again. The frequency at which the scan of the active ports is performed depends primarily on the switching time to be guaranteed in the event of a hard fault. For example, if facility protection is to be effected in a short time, the scan period must be kept sufficiently short so that the sum of the scan period plus the time required to switch is less than the specified switching time. In this example, where the switching time specification is 50 msec (maximum), a scan period of 20 msec allows up to 30 msec to effect facility protection in response to a hard failure.

Soft error conditions are also monitored by each IPU 16 at its associated active input ports in process 26, but

THIS SHEET HAS BEEN
REPLACED



at a less frequent rate. The soft error conditions, as will be described hereinbelow, relate to measured errored seconds over a period of time, and as such the same urgency is not present for detection of a soft error threshold crossing as in the case of a hard error state. The table hereinbelow provides an example of the soft error thresholds upon which facility protection switching may be enabled:

10 errored seconds (ES) in 15 minutes; or
864 ES in 1 hour; or
4 severely errored seconds (SES) in 24 hours

where an ES is defined as any second within which a coding violation is detected, and where an SES is defined as any second in which more than 44 coding violations are detected (i.e., an error rate of greater than 10^{-6} errors per second).

It should be noted that the soft error thresholds checked in process 28 according to this embodiment of the invention differ in type and in degree from conventional performance monitoring thresholds. Firstly, the threshold error rates at which facility protection switching is to be initiated will generally differ from those of conventional performance monitoring (by which the grade of service is defined), and will generally be at more stringent levels so that facility switching may be used to avoid degradation in the grade of service for the facility. In addition, the threshold error rates that initiate facility protection switching are administrable, allowing the system operator to increase or decrease the sensitivity of facility protection switching by adjusting the threshold values via APU 12. Secondly, as hard error conditions (LOS, LOF, AIS) are monitored for facility protection separately from soft error threshold crossings, the threshold definitions in the



above table are purely for errored seconds due to coding violations, and do not include errored seconds due to the presence of LOS or AIS conditions (which are included in the conventional errored seconds definitions).

Referring back to Figure 5, process 28 and decision 29 are performed by IPU 16 at each of its active input ports in first/third stage 20 (such as east input port E_1) to determine if one of the soft error threshold conditions have been exceeded. The monitoring point is at the input port to ensure that only error conditions in the facility, and not errors due to the matrix, are used to effect the protection switching. If no soft error threshold condition has been exceeded, wait state 34 is entered until the desired period (e.g., one second) elapses before the next threshold crossing detection operation is performed.

If a soft error threshold crossing is detected in decision 29, the method according to this embodiment of the invention performs additional decisions to ensure that if the facility is switched, the result will not be worse than that causing the switch. IPU 16 first determines, in decision 30, if the "mate" port (i.e., the opposite east or west port from that checked in process 28) has recently crossed any soft error thresholds during the time it was most recently active. This determination is performed by IPU 16 based on information from its constant monitoring of the mate port which continues even for inactive ports, and thus, according to this embodiment of the invention, APU 12 need not intervene in the switching process. If not, control passes to decision 31 to effect the switching. If thresholds have been crossed for the mate port, IPU 16 analyzes the history of the mate port to determine if its grade of service is higher than that checked; if so, switching is again enabled. If the grade of service at the mate port is not better, control passes back to wait state

This Sheet Has Been Replaced



19

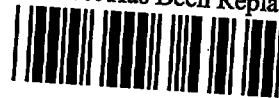
34; no facility switching is effected in this event, as the switch to the mate port would not improve the error rate of the facility.

It should be noted that the threshold crossing check for mate ports as performed by decisions 30, 32 is not performed if a hard failure (LOS, LOF, AIS) is detected. This is because an errored facility is preferred over a hard failed channel; as such, the error rate status of the mate channel is immaterial in the event of a hard failure, as the switch is to be made regardless of the error rate.

Upon entry to the switching portion of the method, whether due to a hard error (from decision 25) or a soft error rate threshold crossing (from decisions 29, 30, 32), IPU 16 first determines, in decision 31, whether the automatic switching function is enabled. If not, the system returns to the monitoring state (via process 40), in which the scan and checks of processes 24, 28 are again periodically performed. To avoid oscillation, it is preferred that each IPU 16 monitor the switching frequency for each facility it receives, and disable automatic switching for a facility upon detecting excessive switching frequency (e.g., greater than four switches in ten minutes). Re-enabling of the automatic switching may be effected either manually or automatically at the end of a defined period (e.g., at the end of a 24 hour period).

As shown in Figures 2 and 4a, according to this embodiment of the invention, each facility is broadcast on both the EAST and WEST ports at all times. Accordingly, at this stage of the switching process (when enabled), decision 33 is performed by which the incoming facility on the mate port is checked by IPU 16 to ensure that the facility received thereat is operable; if not, the system returns to the monitoring state via process 40. The check

THIS SHEET HAS BEEN
REPLACED



performed by decision 33 on the mate port (in the example of Figure 4a, port W_1 of first/third stage 20₁) is accomplished by monitoring the facility received thereat over a period of time, such as on the order of 10 msec, to ensure that no hard failure is present. If the mate port incoming facility is in service, process 36 is then performed to effect a blind receive-end switch.

Referring next to Figure 4b, the matrix condition of cross-connect 2 according to this example is illustrated after the blind switch of process 36. Figure 4b illustrates that first/third stage 20₁ is now receiving the facility on west input port W_1 from the system, and routing the facility through center stage 22₂ to head output port H_0 of first/third stage 20₁. Accordingly, referring back to Figure 2, cross-connect 2 is now receiving the facility from cross-connect 4 via the WEST (microwave) route, which is the secondary route to the EAST (fiber optic) route.

Upon completion of the switching of process 36, IPU 16 notifies APU 12 of the switch (process 38), so that the switching event is visible to the system operator via the supervisory functions and interface of APU 12. In addition to notification of the switching event and the ports affected, IPU 16 may also communicate such information as the cause of the switch (i.e., hard or soft error). The system then returns to the monitor state via process 40, and the process of Figure 5 is repeated periodically for the facility now being received via the WEST route.

It is well known in the art that facility failures are often due to line cuts, such as occur if a fiber optic cable is cut during building construction and other activities. Such a line cut will, of course, appear to cross-connects 2, 4 as a LOS or AIS condition (depending upon the location of the cut) for all of those facilities

This Sheet Has Been Replaced



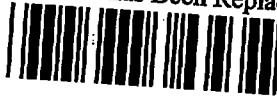
21

communicated by the cut line. The switching of each facility carried by the cut line may be effected in an individual manner by way of the process of Figure 5. However, once a line cut is detected, switching time may be greatly reduced by avoiding much of the decision time, and especially the mate port check of decision 33 which requires a 10 msec monitoring period.

The multiple switching of those facilities associated with a cut line, is preferably performed upon the IPU 16 detecting the first signal failure resulting from a cut line completing the 10 msec mate port monitoring period described above. Upon completion of this monitoring step, this IPU 16 checks all of the ports under its control to determine which, if any, are active east or west ports for a facility. IPU 16 will then execute the process of Figure 5 to check for hard failures for each of its ports which is an active east or west port, prior to relinquishing control, and without the 20 msec scan wait time (process 46) between examination of the incoming facilities at these ports. The overall switching time of the multiple facilities affected by a line failure, such as a line cut, is thus greatly reduced.

It is of course evident from the foregoing description that the facility protection switching scheme of this embodiment of the invention is unidirectional, as it performs a receive-end switch upon detection of a failure event. As noted above relative to line cuts, many facility failures are due to events that affect transmission in both directions (the obvious example being a line cut). Those other cross-connects in the system (such as cross-connect 4) that have the facility protection capability described hereinabove will simultaneously be performing the same monitoring and facility protection switching as that of cross-connect 2. Accordingly, the unidirectional facility

THIS SHEET HAS BEEN
REPLACED



protection switching performed according to this embodiment of the invention will be performed in a bidirectional manner (i.e., by both ends of the line switching unidirectionally) if both cross-connects 2, 4 are configured to do so.

It is also evident that this embodiment of the invention provides facility protection switching in a "blind" manner, in that no communication with the other end of the system is required in order to effect the switch. This blind switching thus provides fast and reliable facility protection. In addition, the distributed nature of the monitoring and switching functions, as performed locally by IPU 16 rather than manually responsive to a central processor, allows for facility protection to be performed in a very rapid manner, such as on the order of less than 50 msec from the time of a hard error fault. As such, the present invention provides a high degree of protection for the customer, particularly those demanding a high grade of service. Finally, as the protection scheme is performed on a facility level, rather than a line level, facility protection in the event of excessive error rates is available by way of the present invention.

Furthermore, the embodiment of the invention described hereinabove relative to Figures 4a, 4b and 5 provides facility protection switching in a manner that is guaranteed to be non-blocking, since the broadcast to the redundant ports is performed in the third matrix stage (see first/third stage 20_a of Figure 4a), and since the receive-end switch is performed in the first stage. This requires that the east and west input ports for a given path must be physically within the same input/output "bay", i.e., controlled by the same IPU 16; similarly, the east and west output ports for a given path must also be physically located within the same input/output bay. This

This Sheet Has Been Replaced



23

construction is not only non-blocking, but also is easily implementable; in this example, with the east and west input ports E_i and W_i in the same ASIC, a single write operation by the associated IPU 16 effects the receive-end switch.

Where the system includes fiber optic lines, the present invention also replaces the 1:1 line protection of conventional FOTs (see Figure 1 above). The redundant fiber optic line is thus available to carry traffic, or alternatively need not be implemented at all. In addition, non-fiber optic lines may now be used as redundant secondary routes for fiber optic lines. Fiber optic utilization is thus greatly improved by way of the present invention, when implemented in fiber optic transmission networks.

Flexible Port Assignment Fast Facility Protection

A second preferred embodiment of the present invention will now be described, in which the constraint on assignment of the east and west ports present in the first preferred embodiment of the invention is overcome. This second embodiment of the invention adds additional benefits to those provided by the first embodiment, especially in enabling implementation of the facility protection switching scheme into existing systems that do not have the east and west ports for a path assigned to the same input/output bay. This embodiment of the invention is additionally advantageous as it allows for flexibility in assignment and rearrangement of ports in newly installed systems.

As in the case of the first preferred embodiment, the method according to the second preferred embodiment of the

THIS SHEET HAS BEEN
REPLACED



invention is preferably implemented in a computer program according to which APU 12 and IPUs 16, 18 operate. It is contemplated that one of ordinary skill in the art will be able to readily implement the method described hereinbelow into a digital cross-connect without undue experimentation, based on the following description.

Figure 6a is a matrix diagram illustrating the state of cross-connect 2 for the example of a single path at the beginning of the method according to this second embodiment of the invention. As noted above, according to this embodiment of the invention, the east and west input ports E_1 , W_1 for this single path need not be assigned to the same first/third stage 20, and thus need not be physically located near each other or in the same input/output "bay" as in the first embodiment of the invention described hereinabove; similarly, the east and west output ports E_0 , W_0 similarly are not constrained to the same first/third stage 20. Accordingly, the present invention is especially useful as it may be implemented in existing installations of digital cross-connects.

As illustrated in Figure 6a, the input transmission from the local side of cross-connect 2 for this path is received at head input port H_1 . In this example, east output port E_0 is on a different first/third stage 20 from west output port W_0 (20, and 20_n, respectively). As such, the broadcast of the transmission from head input port H_1 to east output port E_0 and west output port W_0 cannot necessarily be made at the third stage of the matrix, but instead may have to be made at one of the center stages 22, or in the worst case (as shown in Figure 6a) at the first matrix stage 20, at the location of head input port H_1 .

It is well known in the art that broadcast transmission is preferably made at the latest possible



stage of the Clos matrix for the desired output ports, in order to minimize the possibility of a blocking condition for subsequently assigned ports. It is therefore preferred that the broadcast of the transmission from head input port H_i to east and west output ports E_o , W_o be done in a manner which minimizes the frequency of first or second matrix stage broadcasts. Such a method would first determine if a third stage broadcast is available, and would make a second stage broadcast only if no third stage broadcast would be sufficient; only if neither a third nor a second stage broadcast would be sufficient would a first stage broadcast be used.

It should also be noted, however, that the east and west routes are assigned and reserved as the system is set up according to this embodiment of the invention. As a result, the only time at which blocking could be encountered is at such time as new ports are added and their cross-connect matrix route assigned. In the event that implementation of a new port results in a blocking situation, the matrix routes can be readily reallocated during system configuration to remove the blocking condition.

Furthermore, the probability of blocking is relatively low even for relatively large numbers of ports in the cross-connect. By way of computer modeling, it has been calculated that the blocking probability in a 2048 port cross-connect, having 2n center stages (as described in the above-cited copending application S.N. 882,920, filed May 14, 1992 in the name of E. K. Bowden, entitled "Methods and System for Immediately Connecting and Reswitching Digital Cross-connect Networks", assigned to Alcatel Network Systems, Inc. and incorporated herein by reference) is essentially zero where 70% or fewer of the available ports are utilized and where 33% of the east/west port pairs are

This Sheet Has Been Replaced



26

not assigned within the same four-port logical group (i.e., to the same first/third stage 20), even where 75% of the utilized ports are implemented with the facility protection described herein. With 90% port utilization and 75% of the utilized ports having facility protection implemented, the blocking probability has been calculated to be less than 0.004%. Accordingly, this embodiment of the invention will provide great flexibility of implementation, while only adding an extremely small likelihood of blocking even for large and busy cross-connects.

On the receive side of the bidirectional path in the example of Figure 6a, the initially active route is the EAST route. East input port E_1 is implemented in first/third stage 20₁, and its transmission is routed through center stage 22₁ and first/third stage 20₃ to head output port H_0 . As before, head output port H_0 communicates the transmission to the local side of cross-connect 2.

In this example, the secondary inactive west input port W_0 is implemented at first/third stage 20₂, and thus in a different first/third stage 20 than that of east input port E_0 . West input port W_1 is connected via center stage 22₂ also to first/third stage 20₃, but since west input port W_1 is the secondary port at this time, no connection is made within the third matrix stage in first/third stage 20₃.

As noted above, each of first/third stages 20 may support multiple ports, for example, four ports. In this example, one of the spare ports in first/third stage 20₃ serves as protection output port P_0 . Protection port P_0 is not necessarily able to communicate traffic, but for purposes of this embodiment of the invention is able to be monitored by IPU 16, in the manner to be described hereinbelow.

THIS SHEET HAS BEEN
REPLACED

This Sheet Has Been Replaced



27

Also according to this embodiment of the invention, APU 12 includes a status register 42 for each of east and west input ports E_i , W_i . These status registers 42 contain certain information gathered by the appropriate IPUs 16 for the associated ports, including whether the most recent use of the port resulted in a hard error condition or if the port is available for successful receipt of a facility.

As shown in Figure 7, the method according to this embodiment of the invention includes multiple scanning and checking operations to detect hard and soft error conditions in the received facility. In contrast to the first preferred embodiment of the invention, the monitoring and switching function of this embodiment of the invention is performed at the third stage at which head output port H_0 is implemented, as this is the only first/third stage 20 within which it is guaranteed that traffic from both east and west input ports E_i , W_i will meet.

For the example of the single path in the condition shown in Figure 6a, process 44 is performed by which IPU 16 associated with first/third stage 20, scans incoming traffic at head output port H_0 for hard error conditions, namely the AIS signal noted hereinabove (since any incoming LOF or LOS signal is converted to AIS prior to communication in the matrix), and also any high-speed loss-of-frame condition caused by the matrix itself. Decision 45 is then performed to determine if a hard error condition has been detected; if not, wait state 46 is entered and control remains there until the desired scan period (e.g., 20 msec) elapses, at which time the scan of process 44 is repeated at the head output port H_0 .

If a hard error condition is detected in process 44 and decision 45, decision 47 is performed in which status register 42 assigned to the inactive port for the path (in

THIS SHEET HAS BEEN
REPLACED



this example, the SW₁ register 42 for west input port W₁) is interrogated to determine whether the inactive port is in an error state. If the status register 42 indicates an error state for the inactive port, the facility protection switch will be fruitless, and accordingly control passes back to the monitor condition (process 60). If the status register 42 interrogated in decision 47 indicates that the inactive port is not in an error condition, control passes to process 50 to make the facility switch.

Process 50 effects the facility switch at the head output port H₀ end of the path, which in this case is performed by first/third stage 20₃. As in the first embodiment of the invention, the switch is effected in this example of cross-connect 2 by the associated IPU 16 performing a write operation to the ASIC within which first/third stage 20₃ is realized. After completion of the head-end switch of process 50, the condition of the matrix is in the state illustrated in Figure 6b, with traffic from west input port W₁ routed via first/third stage 20₂ through center stage 22₂ to head output port H₀ via first/third stage 20₃. Traffic from the east input port E₁ is disconnected from head output port H₀ at first/third stage 20₃.

After the head-end switch is effected by process 50 responsive to the hard error condition, decision 57 passes control to decision 59 in which the signal at head output port H₀ is checked, in this example, to ensure that the facility now being received at west input port W₁ is valid. If not, process 62 is performed which reconnects head output port H₀ to the prior active input port E₁, with control passing back to the monitored state. If the facility received at the previously inactive port (e.g., west input port W₁) is valid, decision 59 passes control to process 58 in which the event of the switch is reported to



APU 12, together with information identifying the path for which the switch was made and also the reason for the facility protection switch. Included in the reporting of process 58 is the updating of status register 42 for the previously active port, indicating that the most recent use of the port had an errored condition; in addition, the status register for the newly active port is also updated to indicate that it is in a good condition. Control then passes back into the monitor state (process 60), for the newly active port.

Referring still to Figure 7, the monitoring and facility protection switch for the case of soft error rate threshold crossings will now be described. As in the case of the first embodiment of the invention described hereinabove, the soft error rate monitoring is performed in an interleaved fashion with the hard error scanning, but at a reduced frequency. Accordingly, over a given period of time, each path will be monitored for both hard error events and also for soft error rate excesses.

According to this embodiment of the invention, soft error threshold crossings are monitored at the first stage; in this example, first/third stage 20₁ is monitored for coding violations received at east input port E_1 , and first/third stage 20₂ is monitored for coding violations at west input port W_1 . IPU's 16 associated with these ports thus determine whether a soft error rate threshold is crossed, on an ongoing basis. The soft error rate thresholds checked are preferably similar to those discussed above relative to the first embodiment of the invention, namely errored seconds and severely errored seconds due purely to coding violations (i.e., not including seconds having LOF, LOS or AIS conditions) over varying periods of time; as in the first embodiment described above, the particular thresholds may be

This Sheet Has Been Replaced



30

administrated to those values desired for the grade of service.

The monitoring of soft error threshold crossings at the receive end (first matrix stage) is used to periodically update status registers 42 both for active and also inactive ports. As will be noted below, this information is used in the facility switching for error rate reasons.

In the event that a soft error threshold is crossed for a facility, its IPU 16 inserts a flag to this effect in the overhead data for the facility, prior to transmitting the facility along to its center stage 22. Process 48 checks head output port H_0 to determine if the overhead bits in the facility it receives indicates that a soft error rate threshold has been crossed from the active input port (in this example, east input port E_1). Decision 49 is then performed based on the results of the check of process 48. If no soft error threshold has been exceeded for the path, control passes to wait state 52 for the remainder of the check period (e.g., 400 msec). If a soft error threshold has been exceeded, decision 53 is performed by which the status register 42 of the inactive port is interrogated, similarly as in decision 47 for hard errors. In addition, decision 53 also interrogates the status register 42 to determine if a soft error threshold has recently been exceeded for the inactive port, to prevent switching to a port which also has an active threshold crossing; if either is the case, no switching is performed and control passes back to the monitor state of process 60.

If the inactive port is acceptable (no hard errors and no soft error threshold crossings), IPU 16 associated with first/third stage 20, performs a head-end switch to connect the inactive port to its protection output port P_0 , in

THIS SHEET HAS BEEN
REPLACED



process 54. The result of this switch is illustrated in Figure 6c, showing that traffic from the east input port E_i is still routed to head output port H_o , while traffic from the west input port W_i is routed to the protection output port P_o . IPU 16 associated with first/third stage 20, then examines the traffic at protection port P_o , by checking the overhead data received thereat to determine if a threshold crossing is now being reported by west input port W_i . Of course, the presence of a hard error condition at this facility will also be detected in process 54, such as the LOS, LOF, AIS conditions.

In decision 55, the results of the monitoring of process 54 is determined. If the inactive port currently coupled to protection output port P_o does not have an adequately clear signal, such as a hard error condition or a threshold crossing, no switching is effected, and control passes back to the monitor state (process 60).

If the inactive port currently connected to protection output port P_o is clear, control then passes to process 50 by which the third stage head-end switch is made, connecting the previously inactive port to the head output port H_o . In the example of Figures 6a through 6c, the head-end switch is effected by IPU 16 causing first/third stage 20, to connect west input port W_i to head output port H_o . The result of the head-end switch of process 50 due to crossing of a soft error rate threshold is illustrated in Figure 6b. Protection output port P_o is also then freed for use in connection with any other head output ports served by first/third stage 20.

Since the facility protection switch of process 50 was, in this case, due to excessive soft error rate (and since decision 55 has already determined if a threshold crossing from the new facility has occurred), decision 57



passes control directly to process 58 to report the head-end switch to APU 12, and to update the status register 42 for the previously active port (indicating a soft error threshold was exceeded) and for the newly active port (indicating a good condition). Control then passes back to the monitored state, via process 60.

As discussed above, the facility protection switching provided by this embodiment of the invention is unidirectional. Accordingly, in the event that a bidirectional fault (such as a cut line) occurs, the other cross-connect in the system will similarly respond, if it contains the same facility protection capability. In addition, it is of course contemplated that the monitoring and facility protection provided in this embodiment of the invention may be implemented for many of the ports serviced by a particular cross-connect, especially those for which a high grade of service is desired, such as for data communications.

As in the case of the first embodiment described hereinabove, this embodiment of the invention also provides the significant advantages of facility protection with fast switching times in the event of a hard failure, such switching times being on the order of 50 msec or less. In addition, the protection afforded by this embodiment of the invention is implemented not only for hard line failures, but also for hard failures affecting only a single facility, and also for excessive error rates exceeding a given threshold. In fiber optic networks, 1:1 line protection may be eliminated and the utilization of fiber optic lines for traffic improved. This facility protection is provided in a unidirectional manner, and uses blind switching which does not require handshaking communication with other cross-connects in the system, and thus is relatively simple and extremely fast.

In addition to the advantages provided by the first embodiment of the invention, this embodiment of the invention also allows for a high degree of flexibility in the assignment of ports within a cross-connect, as the east and west ports for a given path may each be physically located anywhere within the cross-connect. Accordingly, this embodiment of the invention is particularly well-suited for implementation into existing installations, as no rearrangement of port assignments is necessary.

While the invention has been described herein relative to its preferred embodiments, it is of course contemplated that modifications of, and alternatives to, these embodiments, such modifications and alternatives obtaining the advantages and benefits of this invention, will be apparent to those of ordinary skill in the art having reference to this specification and its drawings. It is contemplated that such modifications and alternatives are within the scope of this invention as subsequently claimed herein.

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED

This Sheet Has Been Replaced



claims in 92AD029 with reference numerals:

1. A method of facility protection by a cross-connect (2) in telecommunications system, in which a facility is received by the cross-connect (2), at one of first and second input ports (E_I , W_I) from first and second redundant network lines (EAST, WEST), and communicated to a head output port (H_O), comprising the steps of:
monitoring (24, 28, 44, 48) the received facility at said first input port (E_I) to detect an errored condition; and
responsive to detecting an errored condition, connecting (46, 50) the head output port (H_O) to said second input port (W_O) to receive said facility thereat.
2. The method of claim 1, wherein the monitoring step comprises:
detecting (28, 48) coding violations in the received facility over a selected period of time, said errored condition including the number of detected coding violations exceeding a limit.
3. The method of claim 2, wherein said monitoring step further comprises:
detecting (24, 44) a hard error signal in the received facility.
4. The method of claim 1, wherein said cross-connect (2) further comprises first and second output ports (E_O , W_O) coupled to the first and second redundant network lines (EAST, WEST), and coupled to receive digital signals from a head input port (H_I);
further comprising:
broadcasting said digital signals received at the head input port (H_I) at both of said first and second output ports (E_O , W_O) onto the first and second network lines (EAST, WEST).
5. The method of claim 4, wherein said cross-connect (2) is configured as a switching matrix of the Clos type;
and wherein said broadcasting step is performed from the third stage (20₃) of said switching matrix.
6. The method of claim 1, wherein said cross-connect (2) is configured as a switching matrix of the Clos type;
and wherein said broadcasting step is performed from the first stage (20₁) of said switching matrix.

THIS SHEET HAS BEEN
REPLACED

This Sheet Has Been Replaced



claims in 92AD029 with numerals
 draft of June 16, 1992
 page 3

wherein said input/output processing unit (16, 18) comprises a first input/output processor (16₁) coupled to a first stage (20₁) of said cross-connect matrix (20, 22) associated with said east and west input ports (E₁, W₁), said first input/output processor (16₁) for connecting the facility received at an active one (E₁) of said east and west input ports to a center stage (22₁) of the cross-connect matrix, for monitoring the received facility to detect an error condition, and for switching to connect the previously inactive one (W₁) of said east and west input ports to the center stage (22₁) of the cross-connect matrix responsive to detecting the error condition.

11. The system of claim 10, wherein said input/output processing unit (16, 18) comprises:

a second input/output processor (18₁) coupled to the center stage (22₁) of the cross-connect matrix to which the first stage (20₁) is coupled by said first input/output processor (16₁).

12. The system of claim 10, wherein said cross-connect matrix is of the Clos matrix type, having first (20), center (22) and third (20) stages;

wherein said input/output processing unit (16, 18) comprises:

a plurality of input/output processors (16, 18); and

a plurality of first/third stage circuits (20), each coupled to an associated one of said plurality of input/output processors (16);

wherein said east input port (E₁) corresponds to a first one of said first/third stage circuits (20₁), controlled by a first of said plurality of input/output processors (16₁);

wherein said west input port (W₁) corresponds to a second one of said first/third stage circuits (20₂), controlled by a second of said plurality of input/output processors (16₂);

and wherein said first and second ones (20₁, 20₂) of said first/third stage circuits are coupled to a third one of said first/third stage circuits (20₃) corresponding to said head output port (H₀), controlled by a third of said plurality of input/output processors (16₃).

13. The system of claim 12, wherein said first of said input/output processors (16₁) controls the monitoring of the received facility at said first (20₁) of said plurality of first/third stage circuits;



THIS SHEET HAS BEEN
 REPLACED



claims in 92AD029 with numerals
 of June 16, 1992
 ge 4



and wherein said third of said plurality of said input/output processors (20₃) controls the switching to the previously inactive one of said east and west ports (W₁) at said third of said first/third stage circuits (20₃).

14. The system of claim 13, wherein said third of said first/third stage circuits (20₃) further comprises a protection input port (P₀);

and wherein, responsive to detecting an error condition, said third of first/third stage circuits (20₃) switches the previously active one of said east and west ports (W₁) to said protection input port (P₀) for monitoring.

15. A digital telecommunications system, comprising:
 first and second transmission lines (EAST, WEST);
 a first cross-connect (2), comprising:

a plurality of ports, including first and second input ports (E₁, W₁) for receiving a first facility from said first and second transmission lines (EAST, WEST), respectively, and including a head output port (H₀) for presenting the first facility;

a central processing unit (12), coupled to a bus (BUS);

a cross-connect matrix (20, 22), coupled to said bus (BUS);

an input/output processing unit (16, 18) coupled to said bus (BUS) and to said first and second input ports (E₁, W₁), for connecting the facility received at one of said first and second input ports (E₁) to said cross-connect matrix (20, 22), for monitoring the first facility to detect an error condition, and for switching to connect the other of said first and second input ports (W₁) to said cross-connect matrix (20, 22) responsive to detecting the error condition.

16. A method of providing line protection for a telecommunications system including a fiber optic line (13) in such manner as to eliminate the need for providing a redundant fiber optic line, comprising:

providing a redundant route (15) relative to said fiber optic line (13);

broadcasting a facility over the fiber optic line (13) and said redundant route (15) to a cross-connect (2), said cross-connect (2) having first and second ports (6E, 6W) coupled to said fiber optic line (13) and said redundant route (15), respectively, and said cross-connect (2) having a head port (H₀) for presenting the received facility, said head port (H₀) coupled via a cross-connect matrix (20, 22) to said first port (6E or E₁);

monitoring (24, 28, 44, 48) the received facility at said first port (E₁) to detect an errored condition; and

responsive to detecting an errored condition, connecting (50) the head port (H₀) to said second port (W₁) to receive said facility thereat.

claims in 92AD029 with numerals
draft of June 16, 1992
page 7

Abstract of the Disclosure

A digital communications system with fast facility protection is disclosed. The system includes at least one digital cross-connect (2) which is in bidirectional communication with east and west routes (EAST, WEST) of the network via east and west transmit ports (E_o , W_o) and east and west receive ports (E_r , W_r). The cross-connect (2) broadcasts a facility over both the east and west routes, and receives the return facility over one of the routes. The cross-connect (2) monitors the error status of the received facility (E_r) for hard error conditions and excessive error rates, by way of a distributed processor (16) in communication with the port (E_r). In the event of an error, the cross-connect blind switches the facility to the inactive port (W_r) by way of a receive-end switch, according to a first embodiment. According to another embodiment, a head-end switch is performed, thus allowing for improved flexibility for port assignment in the cross-connect (2).

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED

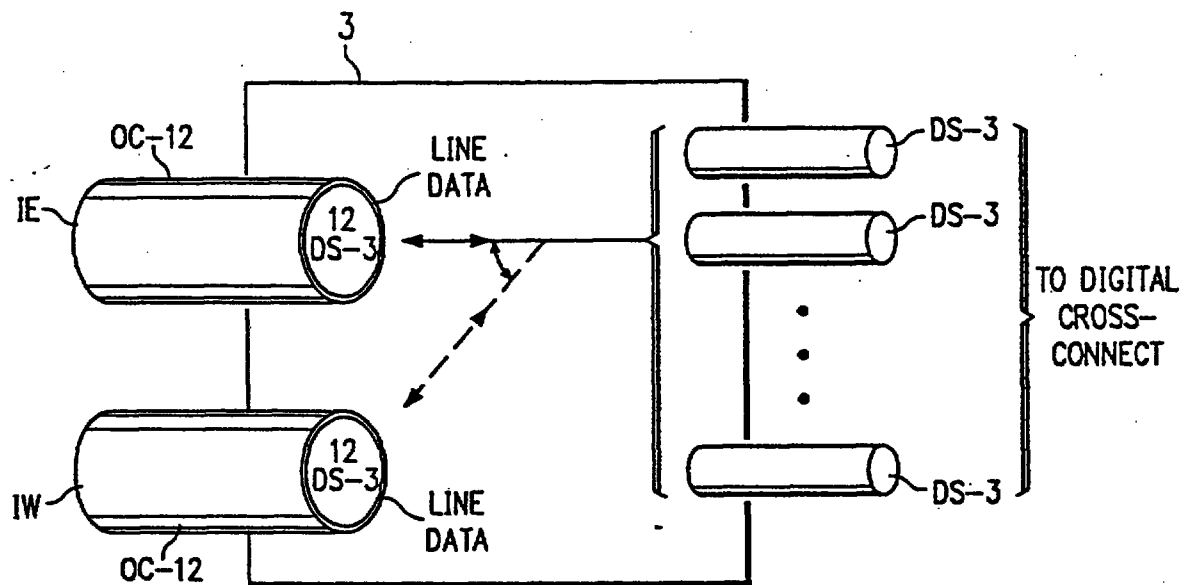


FIG. 1
(PRIOR ART)

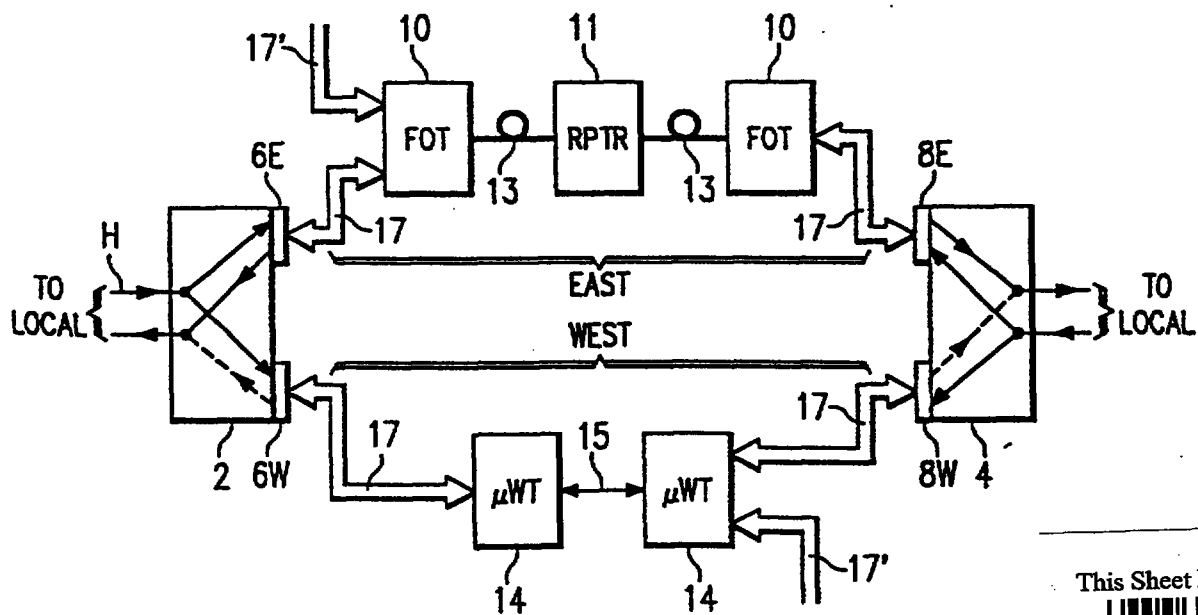


FIG. 2

This Sheet Has Been Replaced



THIS SHEET HAS BEEN
REPLACED



FIG. 3

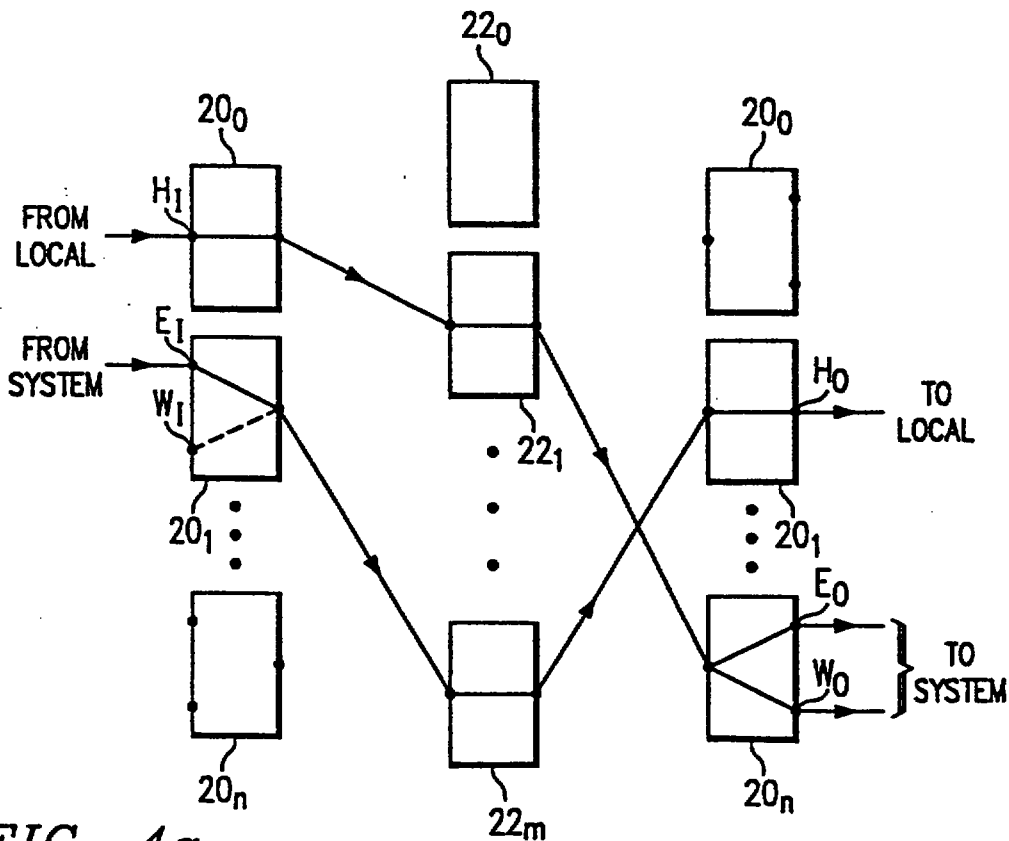
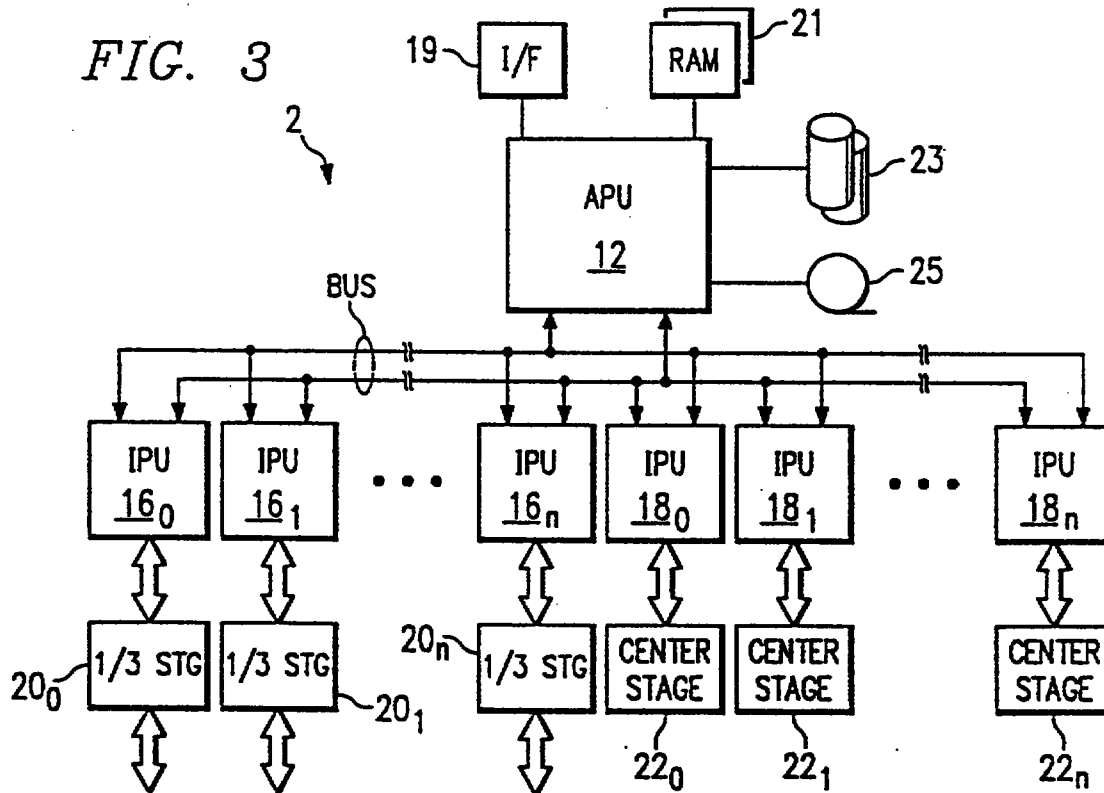


FIG. 4a

3/6

This Sheet Has Been Replaced



92AD028
C.Douglas Richardson.
ALC502/4-5

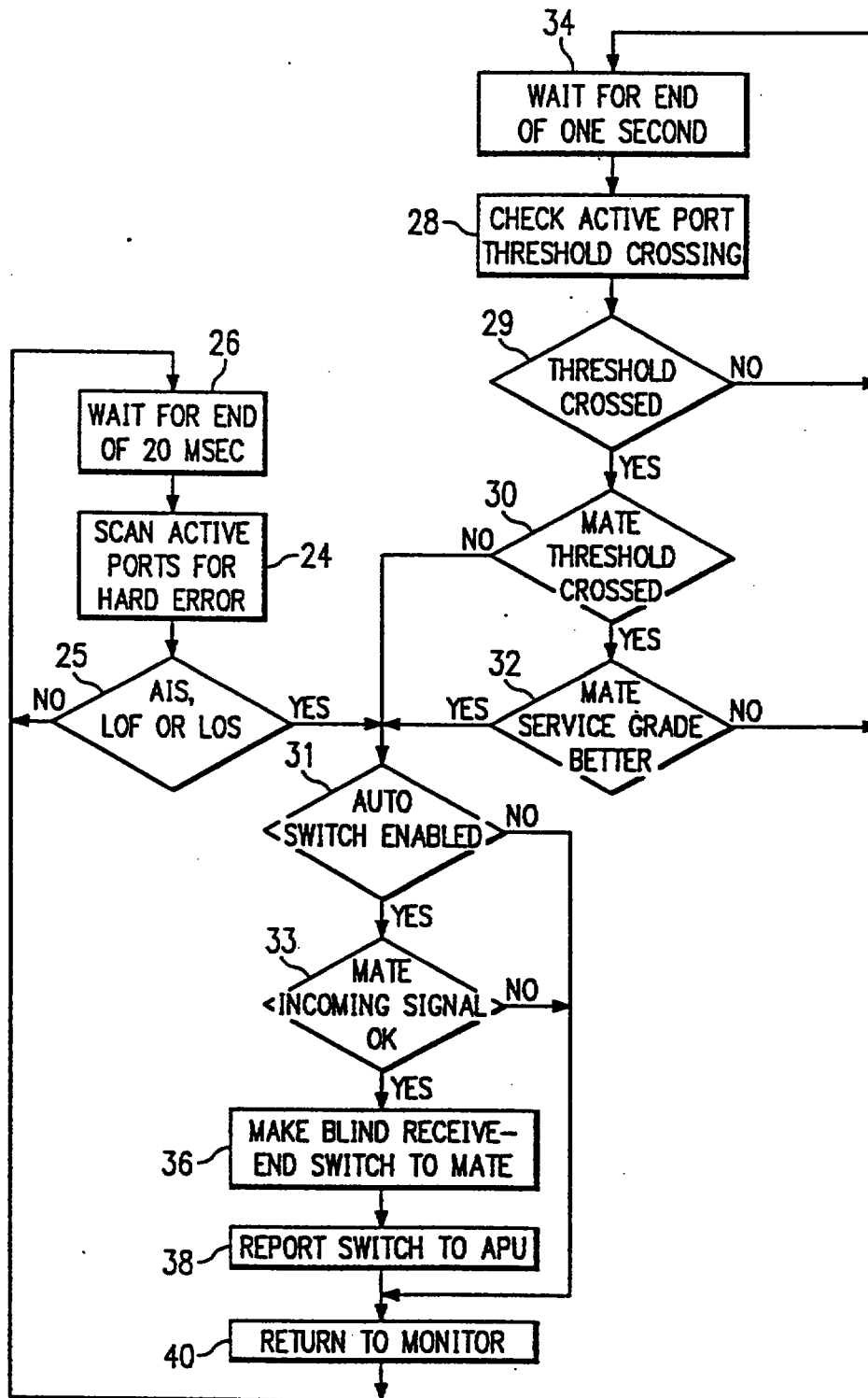
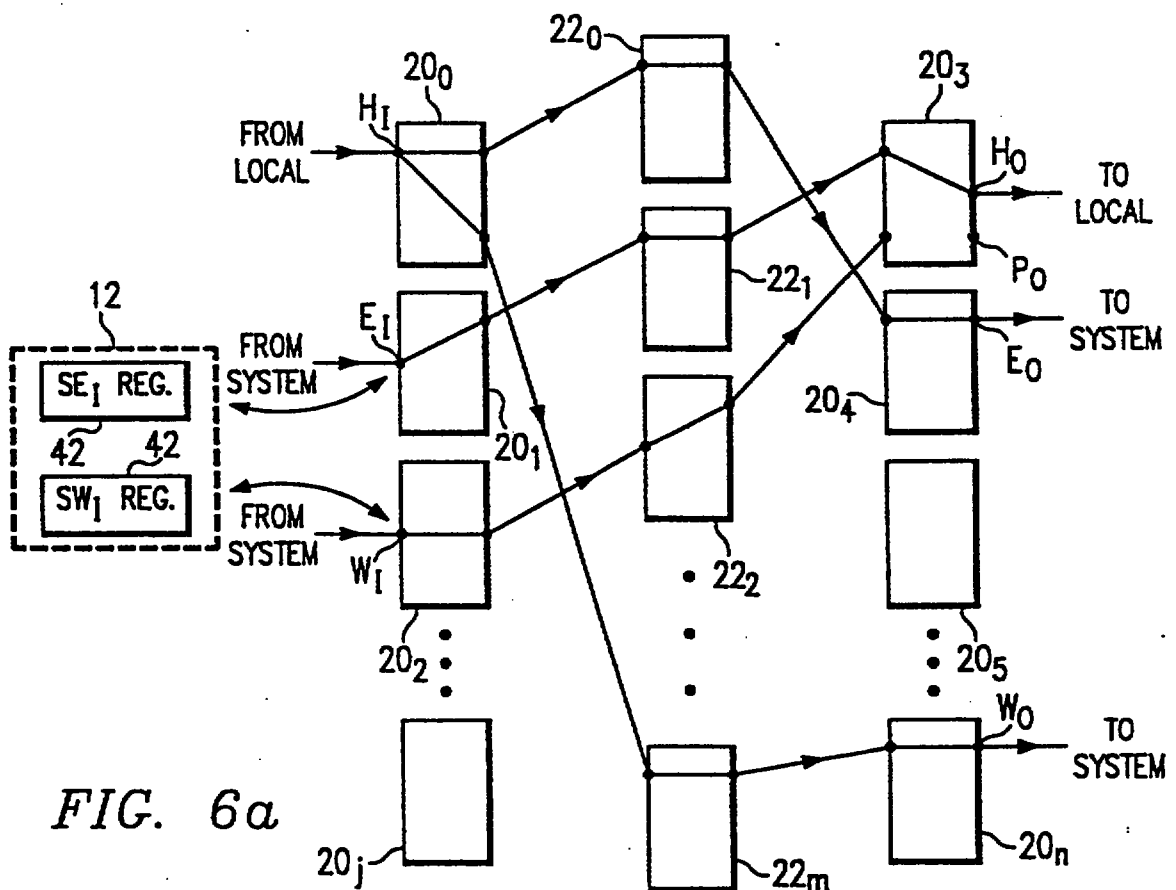
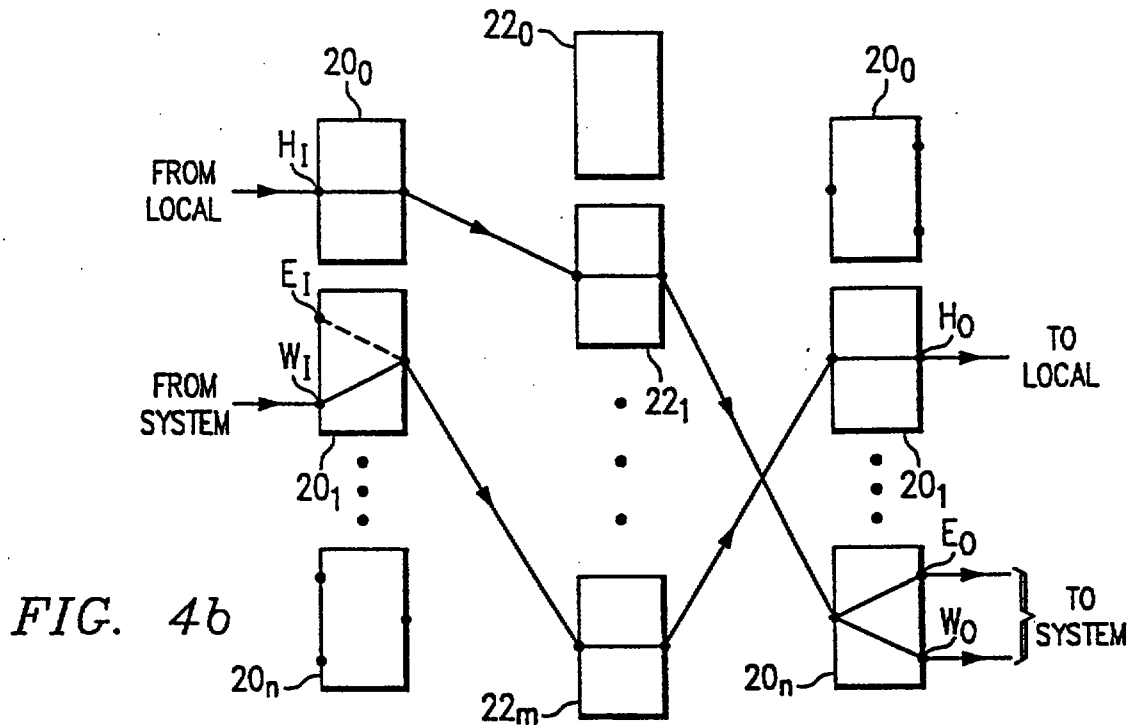


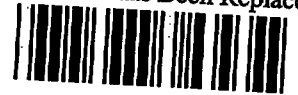
FIG. 5

THIS SHEET HAS BEEN
REPLACED

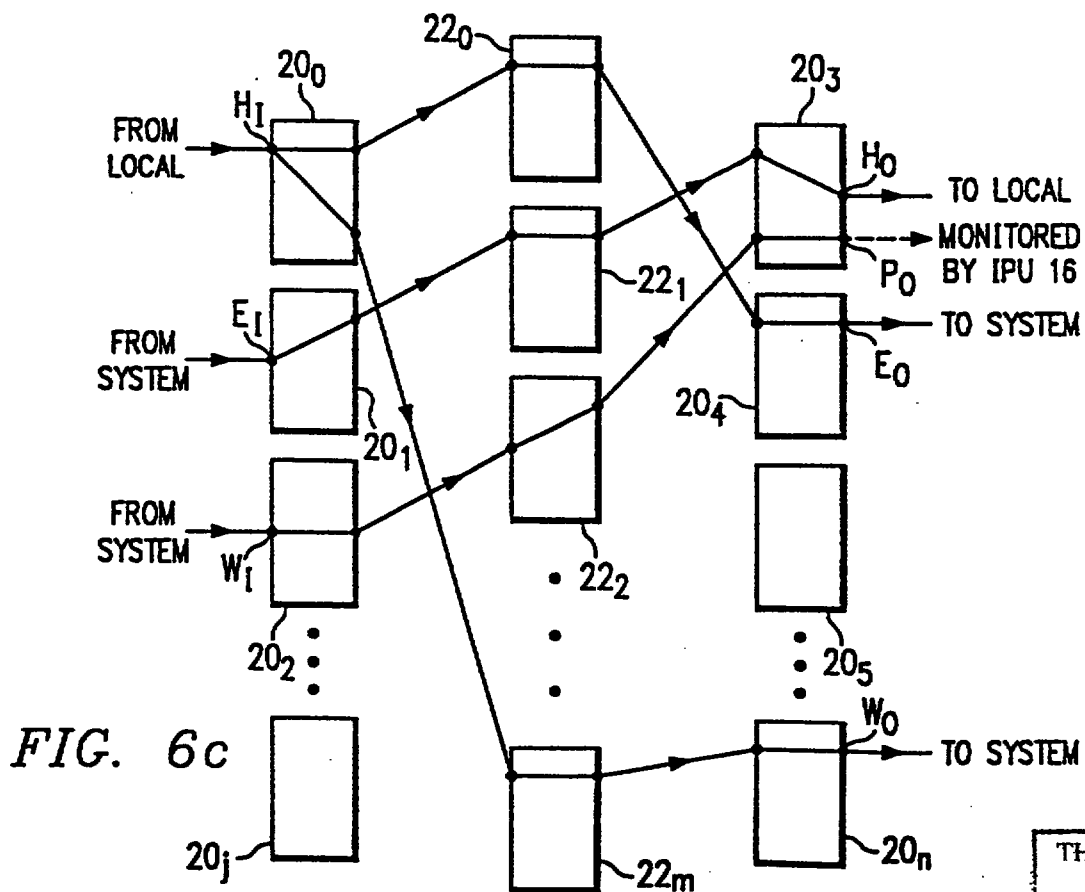
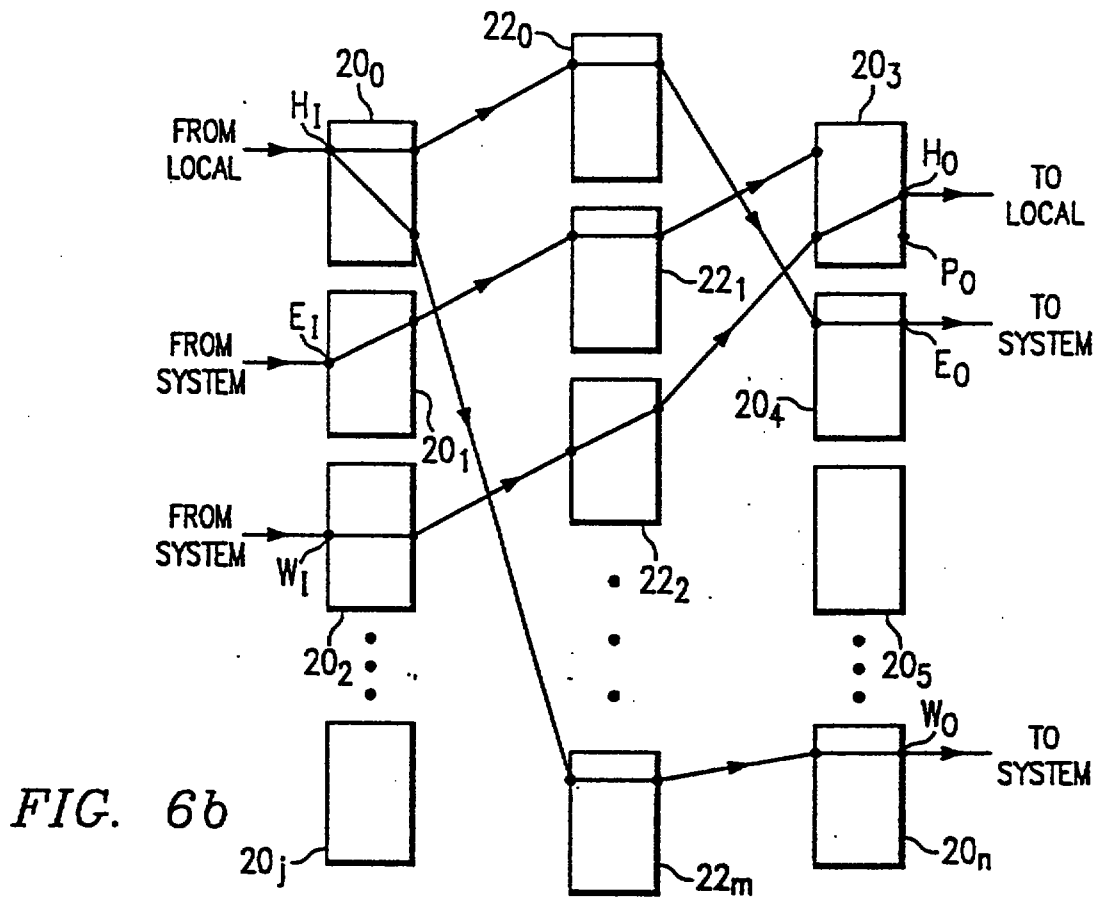


92AD028
C.Douglas Richardson.
ALC502/4-5





92AD028
C. Douglas Richardson.
ALC502/4-5



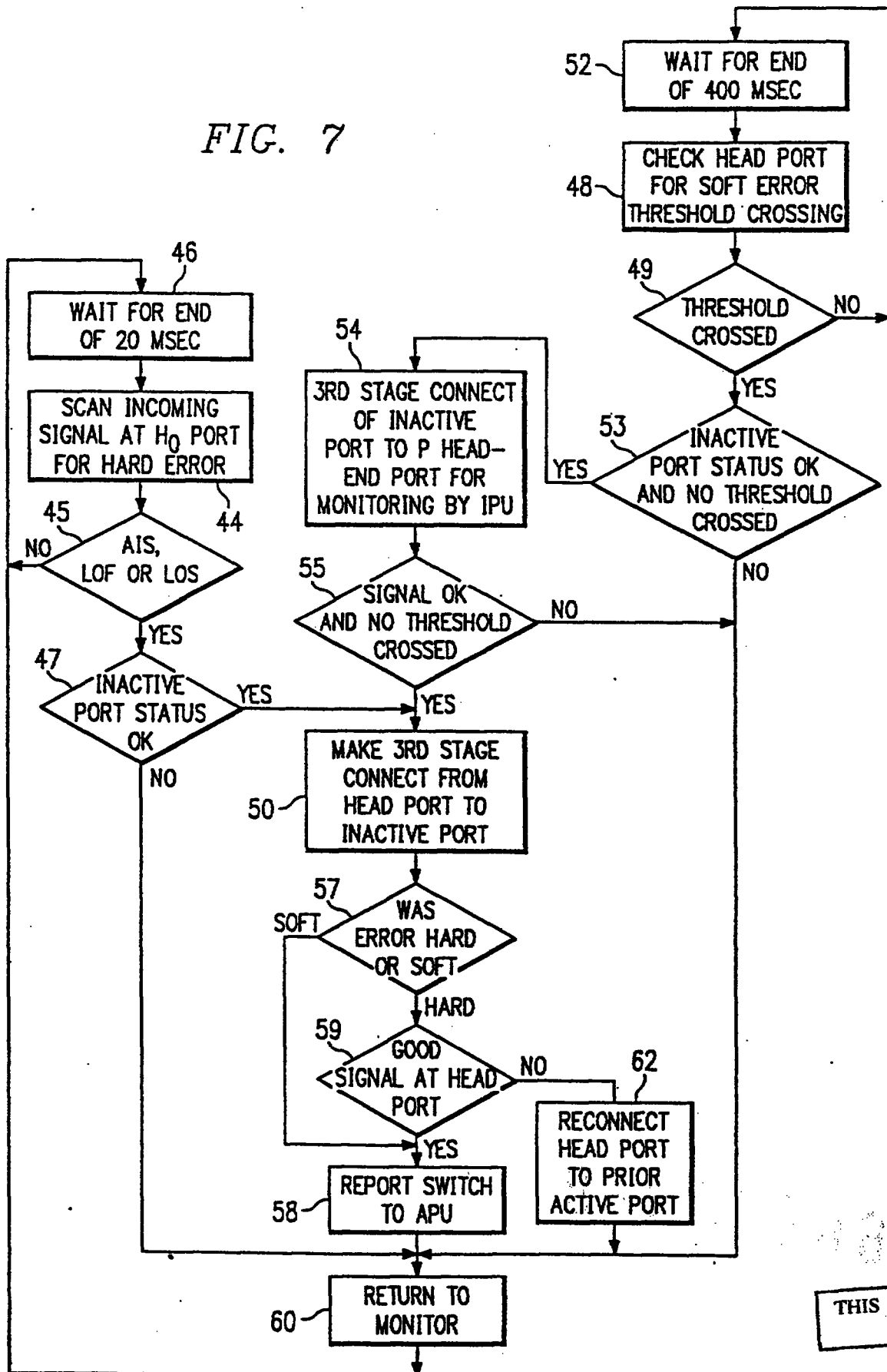
6/6

This Sheet Has Been Replaced



92AD028
C. Douglas Richardson.
ALC502/4-5

FIG. 7



THIS SHEET HAS BEEN REPLACED

24 APR 1997

- 1 -

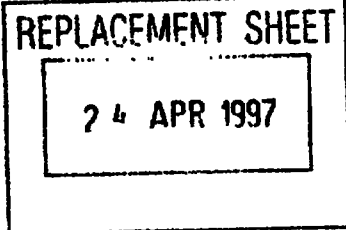
METHOD AND SYSTEM FOR SELECTING AN OPTIMAL
REARRANGEMENT SEQUENCE FOR A CROSS-CONNECT
COMMUNICATIONS MATRIX

NOTICE: COPYRIGHT (©) 1992 Alcatel Network Systems, Inc.

A PORTION OF THE DISCLOSURE OF THIS PATENT DOCUMENT
CONTAINS MATERIAL WHICH IS SUBJECT TO COPYRIGHT
PROTECTION. THE COPYRIGHT OWNER HAS NO OBJECTION TO THE
FACSIMILE REPRODUCTION BY ANYONE OF THE PATENT DOCUMENTS
OR THE PATENT DISCLOSURE AS IT APPEARS IN THE PATENT AND
TRADEMARK OFFICE FILE OF RECORDS, BUT OTHERWISE RESERVES
ALL COPYRIGHT RIGHTS WHATSOEVER.

TECHNICAL FIELD OF THE INVENTION

The present invention is concerned generally with
electronics and more specifically with communications
systems. Even more specifically, the invention is
concerned with a method and system for selecting an
optimal rearrangement sequence for connecting input gates
through a cross-connect communications matrix to output
gates.



- 2 -

BACKGROUND OF THE INVENTION

Digital cross-connect communications network devices, such as the 1631 SX manufactured by Alcatel Network Systems, Inc., are designed to make connections between input gates on input switches and output gates on output switches. Known designs for such cross-connect devices make use of matrices to connect the input gates to the output gates. To limit the physical space and manufacturing costs of the matrices, designers seek to minimize the number of cross-connects in the matrix. This objective becomes more complicated as market demands for communications services increase. In particular, as network devices address larger markets, the devices must service more and more input and output signals. These factors exacerbate the space and cost limitation problems even more for communications matrix devices.

One matrix configuration that uses a minimal number of matrix cross-connects for a given number of input gates and output gates and that provides a potentially optimal solution is known as a "rearrangeable" matrix. For the rearrangeable matrix, the number of center stage switches must equal or exceed the number of input gates on each input switch of the matrix. In a rearrangeable matrix, there exists a set of conditions such that, although the device does not use all input gates and all output gates, an attempt to use an idle input gate and an output gate is prohibited because existing connections block the signal flow through the matrix. This may happen, for example, if existing connections already occupy at least one link in every possible path between the input and output gates in question.

In a rearrangeable network, it is always possible to unblock a flow path from an idle input gate to an idle output gate by moving existing connections in the network. The term "rearrangeable," therefore, describes the property that for a given state of a network and any

24 APR 1997

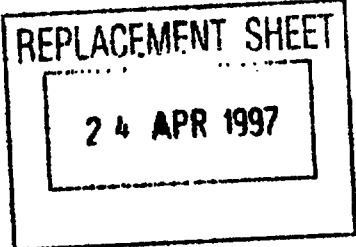
- 3 -

given idle pair of input and output gates, the existing connections of the matrix may be reassigned to new paths, if necessary, to connect the idle pair.

Existing methods and systems for connecting inputs to outputs in rearrangeable matrices generally use a standard rearrangement technique that determines which cross-connects of the matrix to rearrange to permit a signal to flow. N.C. Paull in "Reswitching of Connection Networks," The Bell System Technical Journal, May, 1962, pp. 833-856, describes this known method for unblocking a rearrangeable matrix. This method (hereinafter referred to as Paull's Method) suffers from a major limitation. Paull's Method requires breaking some of the cross-connects and making some other of the cross-connects to rearrange matrix. This procedure takes time and results in undesirable service delays or interruptions during matrix rearrangement.

It is an object of the present invention, therefore, to provide a method and system that permits immediately connecting idle input gates to idle output gates and, if a rearrangeably blocked condition occurs, rearrange the matrix after the immediate connection. An important aspect of this process is to rearrange the matrix using a minimal number of rearrangements of existing connections. The present invention, therefore, achieves this object with a minimal amount of additional circuitry and avoids the service delays and interruptions of known rearrangement methods and systems.

It is also an object of the present invention to provide a method and system for determining the minimum number of rearrangements for a rearrangeably blocked communications matrix by representing the communications matrix by a square matrix having a first dimension representing the input stage of the communications matrix and a second dimension representing the output stage of the communications matrix so that, in the square matrix,



- 4 -

cells represent intersections of the first dimension with the second dimension and indicate possible center stage connections between the input stage and the output stage of the matrix, and wherein the square matrix identifies a blocked center stage switch. A pair of center stage switches, one of which is not associated with the same output stage switch as the blocked center stage switch, and the other of which is not associated with the same input stage switch as the blocked center stage switch, are identified and the number of rearrangement steps necessary for a first rearrangement sequence through the communications matrix from the input stage to the output stage is determined first using one of the center stage switches of the pair and the number for a second rearrangement sequence through the communications matrix from the input stage to the output stage using the other center stage switch is determined. The method performs the rearrangement sequences in incremental parallel steps and ends upon first determining the shorter of the two rearrangement sequences that the pair of center stage switches yields.

BRIEF DESCRIPTION OF THE DRAWINGS

Other objects and advantages are apparent and best understood by reference to the following description of illustrative embodiments when read in conjunction with the accompanying drawings, wherein:

FIGURE 1, shows five-stage connecting network (TSSST) with embedded three-stage space matrix (SSS);

FIGURE 2 shows in more detail the three-stage connecting network that may be embedded logically in the five-stage TSSST network of FIGURE 1;

FIGURE 3 is a three-stage connecting network generalized to the $N(m,n,r)$ case;

FIGURE 4 introduces notation for understanding the preferred embodiment;

FIGURE 5 provides a flow chart of the method of the preferred embodiment;

FIGURES 6 through 14 illustrate an example of the sequence generating portion of the preferred embodiments; and

FIGURES 15 through 34 illustrate a simplified space-domain example of the preferred embodiment;

24 APR 1997

- 6 -

DETAILED DESCRIPTION OF THE INVENTION

In FIGURE 1, a large matrix configuration is shown that provides for connection and test access for up to 1,024 DS3 ports (or equivalently, 32,678 DS1 ports) for the 1631 SX device. This is a five-stage, Time-Space-Space-Space-Time (TSSST)-division multiplexed connecting matrix network 50. Matrix configuration 50 includes time stage 52 that through connections 54 connects to three center space stage (SSS) designated generally by reference numeral 56. Three center space stage SSS 56 connects through output line 58 to output time stage 60. With this configuration, the network 50 includes a time-domain of time stages 52 and 60 and a space domain of three center space stage 56. FIGURE 1 illustrates logically the relationship between time stages 52 and 60 and the center space stages 56. Input time stage 52 uses 128 time slots per input 62. The time slots may be represented by the connections 64 between I/O shelf inputs 62 to each time slot 66 of matrix 56. Center space stage 56 includes 128 time slots 66, each of which time slot 66 includes a $N(17,16,16)$ matrix. The number 17 of time slot matrix 66 represents the number of center stage switches; the first number 16 represents the number of inputs to each input stage switch, and the second 16 the number of input stage switches. (By symmetry, there are 16 outputs from each output stage switch and 16 output stage switches.) Output time stage 60 uses 128 time slot connections such as connection 68 for each matrix output 70 to an I/O shelf.

In FIGURE 2, center space stage 56 is represented mathematically as a three-stage connecting network cube of matrices 72 for each of which the symbol $N(17,16,16)$ fully describes each of the matrices. Thus, for each $N(17,16,16)$ matrix 72 there are in the example of FIGURE 2, 16 input switches such as input switch 74, for which each receives 16 input gates such as input gate 76. The

24 APR 1997

- 7 -

total number of input gates 76 make up the I/O shelf-to-time slot connections 64 of FIGURE 1 for each time slot. From each input switch 74 go input switch-center switch connections 78 to the center switches such as center switch 80. In the preferred embodiment, each $N(17,16,16)$ matrix 72 includes 17 center stage switches 80. From each center stage switch 80 go 16 output switch connections 82 to output switches such as output switch 84. Each output switch 84 provides 16 output gates such as output gate 86 that connect to I/O shelves such as I/O shelf 70 of FIGURE 1. The total number of output gates 86 make up the time slot-to-I/O shelf connections 68 of FIGURE 1 for each time slot.

Connections are made between input gate 76 and output gate 86 immediately using the present connection technique. This technique sets up a connection by using a route through input switch 74, center switches 80 and output switches 84 that attempts to use the most used part of a matrix 72 first and the least used part of the matrix 72 last. By using the heavily-loaded part of the matrix where possible in making a given connection, when rearrangement of matrix 72 is necessary, the rearrangement process of the preferred embodiment encounters elements with the smallest number of pre-existing connections. This minimizes the volume of necessary computations for matrix rearrangement.

The preferred embodiment uses a dedicated time slot when rearrangement is necessary in the time domain. For example, referring to FIGURE 1, the 128th time slot may be dedicated as an extra time slot for use during matrix 50 rearrangement. Similarly, in the space domain, rearrangement uses a dedicated or extra center switch. For example, referring to FIGURE 2, center switch 17 may be the dedicated or extra center switch. The present connection technique automatically uses the dedicated or extra element (either the dedicated or extra time slot in

24 APR 1997

- 8 -

the time domain or the dedicated center stage switch in the space domain) when a rearrangeably blocked condition exists.

FIGURE 3 illustrates that $N(17,16,16)$ matrix 72 of FIGURE 2 may be generalized to matrix 88 for which the expression $N(m,n,r)$ describes the configuration. Following the convention of the $N(17,16,16)$ matrix 16, the generalized notation $N(m,n,r)$ describes generalized matrix 88. In the $N(m,n,r)$ notation, m equals the number of $r \times r$ center stage switches (as well as the number of outputs from each $n \times m$ input stage and the number of inputs to each $m \times n$ output stage switch); n equals the number of inputs to each $n \times m$ input stage switch (as well as the number of outputs from each $m \times n$ output stage switch); and r equals the number of $n \times m$ input stage switches (as well as the number of $m \times n$ output stage switches). Note also that, in matrix 88, m at least equals $n+1$. This limitation on m for a given n assures that there exists at least one extra center stage switch.

Referring more particularly to FIGURE 3, generalized matrix 88 has n input gates 90 that connect to input switches, such as input switch 92, having m output connections 94. In generalized matrix 88, there may be, for example, r input switches 92. The input switch-to-center switch connections 94 connect to the $r \times r$ center switches such as center switch 96 where each center switch 96 receives r center switch connections 94 and provides r output connections 98. In the example of FIGURE 3, there are m center switches 96. Each center switch-to-output switch connection 98 from center switches 96 connects to an output switch 100. Each output switch 100 receives m center switch-to-output switch connections 98 and outputs to n output gates such as output gate 102.

24 APR 1997

- 9 -

Both the $N(17,16,16)$ matrix 72 of FIGURE 2 and the $N(m,n,r)$ matrix 88 of FIGURE 3 are said to be rearrangeable matrices if, respectively, for a given state of the matrix and any given idle pair of input gates 76 or 90 and output gates 86 or 102, the existing connections within the matrices may be reassigned to new paths, if necessary, to allow connecting the respective idle pairs. The matrices are rearrangeable if and only if $m \geq n$. For convenience in notation, in a network described by the expression $N(m,n,r)$, the notation $\Delta(m,n,r)$ denotes the number of connections that must be rearranged in order to connect an idle pair of input and output gates. Paull's Method derives from the property that $\Delta(n,n,n) \leq n-1$. In other words, for the $N(n,n,n)$ matrix, it is necessary to move at most $n-1$ connections to connect an idle input gate to an idle output gate. A particular application of this property means, for example, that for matrix 72 of FIGURE 2, rearrangement requires at most $n-1=17-1=16$ connection changes in order to connect an idle input gate to an idle output gate.

In the following discussion, FIGURE 4 introduces notation to assist in understanding the preferred embodiment. FIGURE 5 provides a flow chart to illustrate the basic concepts of the immediate connect method and system of the present invention. Thereafter, FIGURES 6 through 14 show an example of the operation of the method and system in the presence of a rearrangeably blocked condition of a matrix 72 of the three center stage space matrix 56. For the purpose of understanding the present invention in yet more detail, the operation of the method and system are described by way of FIGURES 15 through 34 that use a simplified $N(5,4,4)$ rearrangeable matrix having the ability to perform immediate connection in a rearrangeably blocked condition.

In FIGURE 4, notation beneficial for understanding the present invention uses square matrix 110 to represent

24 APR 1997

- 10 -

connections existing in a space domain matrix such as matrix 88 of FIGURE 3. Square matrix 110 has rows 112 designating input switches (such as input switch 92 of FIGURE 3) and columns 114 representing output switches (such as output switch 100 of FIGURE 3.) The simplified example of FIGURE 4 shows the number of rows 112 and columns 114 equals 8. Note that to correspond exactly to the $N(17,16,16)$ matrix of FIGURE 2, FIGURE 5 could show a matrix having 16 rows representing input switches 74 and 16 columns representing output switches such as output switch 84. Within each square matrix position such as matrix position 116, there are m possible symbols that may appear. These m symbols correspond to the center switches such as center stage switch 96 of FIGURE 3. For example, matrix position 116 may have m possible entries, one for each of the center stage switches 96 appearing in FIGURE 3.

In FIGURE 4, the ordered pair $(3,1)$ denotes matrix position 116, where 3 is the row designation, i.e., row 3 having reference numeral 118, and 1 is the column position, i.e., column 1 having the reference numeral 120. The entry B ($1 \leq B \leq m$) in matrix position $(3,1)$ corresponds to a connection from input switch 3 through intermediate switch B to output switch 1. No entry in a matrix position such as the empty matrix position $(3,2)$ that reference arrow 122 points to indicates the absence of any connection between input switch 3 and output switch 2.

Referring, for example, to FIGURE 3, since there may be only n input gates 90 to each input switch 92 and n output gates 102 from each output switch 100, only n symbols may appear in any row 112 or column 114. Since each input switch 92 has only one connection 94 to each center switch 96, no two symbols in any row 112 may be the same. Similarly, each output switch 100 has only one connection 98 from each center switch 96. No two symbols

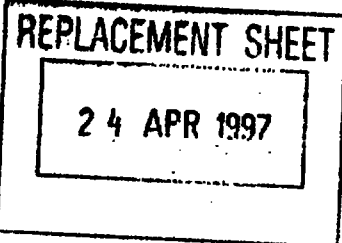
24 APR 1997

- 11 -

connections existing in a space domain matrix such as matrix 88 of FIGURE 3. Square matrix 110 has rows 112 designating input switches (such as input switch 92 of FIGURE 3) and columns 114 representing output switches (such as output switch 100 of FIGURE 3.) The simplified example of FIGURE 4 shows the number of rows 112 and columns 114 equals 8. Note that to correspond exactly to the $N(17,16,16)$ matrix of FIGURE 2, FIGURE 5 could show a matrix having 16 rows representing input switches 74 and 16 columns representing output switches such as output switch 84. Within each square matrix position such as matrix position 116, there are m possible symbols that may appear. These m symbols correspond to the center switches such as center stage switch 96 of FIGURE 3. For example, matrix position 116 may have m possible entries, one for each of the center stage switches 96 appearing in FIGURE 3.

In FIGURE 4, the ordered pair $(3,1)$ denotes matrix position 116, where 3 is the row designation, i.e., row 3 having reference numeral 118, and 1 is the column position, i.e., column 1 having the reference numeral 120. The entry B ($1 \leq B \leq m$) in matrix position $(3,1)$ corresponds to a connection from input switch 3 through intermediate switch B to output switch 1. No entry in a matrix position such as the empty matrix position $(3,2)$ that reference arrow 122 points to indicates the absence of any connection between input switch 3 and output switch 2.

Referring, for example, to FIGURE 3, since there may be only n input gates 90 to each input switch 92 and n output gates 102 from each output switch 100, only n symbols may appear in any row 112 or column 114. Since each input switch 92 has only one connection 94 to each center switch 96, no two symbols in any row 112 may be the same. Similarly, each output switch 100 has only one connection 98 from each center switch 96. No two symbols



- 12 -

in any column 114 may be the same. With the notation of FIGURE 4, a square matrix 110 satisfying these restrictions is said to be "legitimate" since it has only legitimate entries.

Note, that a trivial blocking case occurs if all of the input lines to an input switch such as input switch 1 designated by row 1 are already connected. This would correspond to having n symbols (e.g., 16 symbols in the case of FIGURE 2) in row 1. Similarly, if all the output lines to output switch 3, for example, were already connected corresponding to n symbols in column 3 (e.g., 16 symbols for the example of FIGURE 2) the network matrix 72 of FIGURE 2 and 88 of FIGURE 3 is trivially blocked and square matrix 110 will represent this fact.

In FIGURE 5, flow chart 140 describes the steps and functions of the preferred embodiment of the present invention. Beginning at start step 142, the method is first to query, as query block 144 indicates, whether there is a pair of ports presented for connection. In the example of FIGURE 3, the step is to query whether there is an idle input gate 90 to connect to an idle output gate 102. If there is such a pair, then flow goes to block 146 to begin time domain connections. If there is not a pair of ports presented for connection, then flow goes to block 148 which is described below. At begin time domain step 146, flow goes to block 150 where the method selects the lowest numbered time slot available for connection. Then, at block 152 space domain operations begin for selecting the lowest numbered center stage input switch available for connection at step 154. Having selected the lowest numbered center stage for connection, the connections are made at block 156 and square matrix 110 or a similar status table designating connections is updated at block 158.

At block 160, a query of whether space rearrangement is required takes place. If so, then flow goes to block

162 where the step is to select two center stage switches for rearrangement. Next, at block 164, the step is to find the connections to be rearranged and, from the connections, the query at block 166 is of which path requires the least number of rearrangements. If the path X-A requires the least number of rearrangements, then program flow takes path 168. If X-B requires the least number of rearrangements, then program flow takes path 170. Regardless of which path program flow takes, at step 172 connections are rearranged, one at a time, using a "hitless roll." The discussion below defines the term "hitless roll." And flow returns to step 174 to end space domain operations. Note that if no space rearrangement was necessary, then flow from query block 160 goes directly to end space domain step at block 174.

At query 176, the question is whether time rearrangement is required. If so, then flow goes to step 178 at which point the method is to select two time slots for rearrangement. Then, at block 180, the step is to find connections to be arranged. At block 182, a query occurs of which path requires the least number of rearrangements. If the X-A path requires the least number of rearrangements, flow goes along path 184. On the other hand, if the X-B path requires the least number of rearrangements, then flow follows path 186. The next step, regardless of whether program flow takes path 184 or 186, is to rearrange connections one-at-a-time, again using a hitless roll in the time domain at step 188. Program flow then goes to step 190 to end time-domain operations. Note that if query block 176 determines that no time rearrangement was required, flow proceeds directly to end time domain block 190.

From end time domain block 190, flow goes to query 148 where the method determines whether a pair of ports is presented for disconnect. If not, then flow returns to query block 144 to proceed as previously described.

If a pair of ports is presented for disconnect, at this point the method is to remove connections as requested per block 192 and then update data tables reflecting the connections for the matrix at step 194.

At query 196, the method determines whether a "pack connections after a disconnect" function is enabled. If so, then flow goes to query 198 to examine whether a rearrangement is in progress. If so, then flow will loop via path 200 until rearrangement is no longer in process. Although FIGURE 5 shows disconnect as occurring subsequent to completing any necessary rearrangement, the present method and system may process a disconnect request in parallel with rearrangement. Therefore, if the system of the present invention seeks to direct connections to the most used part of matrix first so as to pack calls after a disconnect, the rearrangement-in-progress query of block 198, prevents attempting to pack during rearrangement.

If a rearrangement is not in progress, from query block 198, the next step is to determine the connections to the packed in the space at step 202. The next step of flow chart 140, then, is to pack connections in the space domain using a hitless roll at block 204. And then, at block 206, the step is to determine the connections to be packed in time, after which the connections are packed at block 206 in time using an automatically hitless roll. Program flow then returns to point 210 to continue the previously described operations of flow chart 140.

The following discussion describes how steps of flow chart 140 of FIGURE 5 relate to the square matrix 110 of FIGURE 4. Recall that after updating square matrix 110 or similar table at step 158, the method of the preferred embodiment detects whether a rearrangeably blocked condition exists. Thus, when the dedicated element (either a time slot in the time domain or center stage

24 APR 1997

- 15 -

switch in the space domain) in a matrix connection configuration is used to make an immediate connection between an idle input gate and an idle output gate, the existence of a rearrangeable blocking condition is automatically detected.

Steps 162 and 178 execute Paull's Method to find the path requiring the fewest rearrangements. Specifically, with the blocked cell denoted (r_1, c_1) such as cell 121 of FIGURE 4, the method tests to find all symbol pairs (A, B) such that if A is in row 1, but not in column 1, and B is in column 1, but not in row 1. For given pairs of symbols, say (A, B) , there are two possible rearrangement sequences, one of which will be shorter. Making this determination is part of step 16 in the space domain and step 182 in the time domain.

The sequence "X→A" denotes a sequence that starts with an immediate connection through the dedicated element (e.g., center stage switch 17 of FIGURE 2 for the space domain and the time slot 128 in FIGURE 1 for the time domain) and ends with the blocked cell being assigned to element A. The sequence begins, for example, by starting with the blocked cell (r_1, c_1) at 121 of FIGURE 4). Since there is no B in row r_1 , the search for an A in r_1 takes place. There must be an A in row r_1 , otherwise the matrix is not blocked. Once the A in row r_1 is found, it is circled or tagged.

In general, then, suppose that the circled A is in (r_j, c_k) , $1 \leq j, k \leq 8$. Then, a search for a B in column c_k takes place. If there is no B in column c_k , the sequence ends and then the next step is to find connections to be rearranged at block 164 or 180, as appropriate. If a B appears in column c_k , the next step is to circle the B and continue. If the circled B is in the (r_k, c_j) position, then a search for an A in row r_k takes place. The sequence X→A continues to alternate between circling B's in columns in A's in rows, until there exists a

24 APR 1997

- 16 -

column with no B to circle or a row with no A to circle. Note that not all A's and all B's get circled, only those encountered in the above search.

Generating the X-B sequence occurs in similar fashion, as follows. Starting with the blocked cell (r_1, c_1) 121 of FIGURE 4, for example, there is no A in column 1, so the search begins for a B in column c_1 . There must be a B in column c_1 , otherwise the matrix would not be blocked. Then, procedure is to circle that B. Since the circled B is in position (r_3, c_1) , the search for an A in row r_3 begins. If there is no A in r_3 , then this sequence ends and program flow continues to step 164 or 180, as appropriate for the space or time domain. Otherwise, the procedure is to circle the A and continue. In this example, the circled A is in position (r_3, c_3) , then the search begins for a B in column c_3 . If there is no B in column c_3 , then the sequence ends. Otherwise, the sequence is to circle the B and continue. This procedure continues to alternate between circling B's in columns and A's in rows until either a column with no B to circle or a row with no A to circle is found. Note, not all A's nor all B's get circled, only those encountered in the above search.

For a given pair of symbols (A,B), the next step is to choose the path generated by the sequence (either the X-A sequence or the X-B sequence) that contains the least number of circled symbols (i.e., A's or B's). For all such pairs of symbols, the step is to choose the pair (A,B) that contains the least number of circled symbols (either A's or B's). This sequence determines the minimum number of rearrangements necessary to establish a path from the input to the output of the rearrangeable matrix.

Rather than completely generating all the X-A and X-B sequences, the preferred embodiment generates the shortest sequence. This is accomplished in block 164 in

the space domain and block 180 in the time domain using a technique of the preferred embodiment that shall hereinafter be called a flooding algorithm. The flooding algorithm starts by finding the first element of the X-A sequence. If the X-A sequence is not ended, the first element of the X-B sequence is found. Then, if the X-B sequence is not ended, the next element of the X-A sequence is found. This process continues until the shorter of the two sequences is generated. At the time that the shorter of the two sequences is generated, the search terminates.

The point at which flooding algorithm terminates determines the path of the minimum number of rearrangements as indicated in block 166 in the space domain and block 182 in the time domain.

In step 172 for the space domain and step 188 for the time domain, rearrangements take place using a hitless roll. The sequence of circled symbols, either A's or B's, that the above steps determined are then changed to clear the blocked cell using the appropriate X-A or X-B roll depending on which yields the minimum number of rearrangements.

The following Table 1 and accompanying text describe the steps of making the connections to yield rearrangement of the center stage switches when X-A results in the minimum number of connection rearrangements:

TABLE 1	
STEP	ACTION
1	$X-(r_1, c_1)$
2	$\textcircled{B}-\textcircled{B}, X$
3	$\textcircled{B}, X-X$
4	$\textcircled{A}-\textcircled{A}, B$
5	$\textcircled{A}, B-B$
6	$X-X, A$
7	$X, A-A$

If the X-A sequence yields the minimum number of rearrangements, then, referring to square matrix 110 of FIGURE 4, step 172 or 188 of FIGURE 5, as appropriate, the method is to put an "X" in the (r_1, c_1) position indicating the connection was made through the extra center stage switch, (TABLE 1, Step 1 indicates this action). Then for the circled B's, the roll first adds X's (Step 2) and then removes the circled B's (Step 3). Next, for the circled A's, the roll first adds the B's (Step 4) and then removes the circled A's (Step 5). Then, for X's appearing in the matrix, A's are first added (Step 6) and then the X's are removed (Step 7).

The following Table 2 and accompanying text describe the steps of making the connection to yield rearrangement of the center stage switches when X-B results in the minimum number of connection rearrangements:

TABLE 2	
STEP	ACTION
1	$X-(r_1, c_1)$
2	$\textcircled{A}-\textcircled{A}, X$
3	$\textcircled{A}, X-X$
4	$\textcircled{B}-\textcircled{B}, A$
5	$\textcircled{B}, A-A$
6	$X-X, B$
7	$X, B-B$

The X→B roll occurs in similar fashion to that of the X→A roll. For example, the first step is to put an X in the position (r_1, c_1) indicating the connection was made through the extra center stage (TABLE 2, Step 1). For the circled A's, the roll first adds X's (Step 2) and then removes the circled A's (Step 3). Next, for the circled B's, the roll adds A's (Step 4) and then removes the circled B's (Step 5). Then, for the X's, the roll adds the B's (Step 6) and finally removes the X's (Step 7).

Rearrangement is performed as necessary using a "head-end-bridge" and a "receive-end-switch" to effect a hitless roll (including the initial connection placed in the designated or extra element) to the lower numbered elements of the matrix. The terms head-end-bridge and receive-end-switch are defined and explained illustratively in connection with the example of FIGURES 15 through 34 below. These steps leave the extra element

clear and ready for the next immediate connection in a rearrangeably blocked condition.

Although it is believed that the present specification and drawings fully and accurately describe the system and provide sufficient information so that anyone skilled in the art may practice the inventive concept, "Unpublished Appendix A" is provided herewith to remain unprinted. Unpublished Appendix A includes a listing of the working source code to enable a computer to operate and perform the immediate connect method of the present method and system.

OPERATION

Having explained generally the processes of the preferred embodiment, the following illustrative example shows how the preferred embodiment provides an immediate connect in the presence of a rearrangeably blocked condition in the space domain. The time domain method becomes obvious upon understanding implementation of the space domain method.

FIGURES 6 through 14 illustrate an example of the immediate connect method of the preferred embodiment. In the square matrix 110 of FIGURE 6, cell (1,1) is marked as blocked because the present connection algorithm detected that the dedicated center stage switch is the only center stage switch having both an idle link to input switch 1 and an idle link to output switch 1.

In response to this condition, the connection algorithm in effect puts an X at (r_1, c_1) to make the connection, as FIGURE 6 shows. The use of the notation "X" herein denotes the use of the dedicated or extra center stage switch in a rearrangeably blocked condition. When the designated dedicated center stage switch of the large matrix configuration is used by the connection

24 APR 1997

- 21 -

algorithm, a rearrangeable blocking condition is automatically detected.

Since the connection pair (1,1) is rearrangeably blocked, in order for a connection to be possible, there must exist an idle link between input switch 1 and some center stage switch, say A. Furthermore, there must exist an idle link between output switch 1 and some center stage switch, say B. The pair (A,B) is one pair of center stage switches to be considered for rearrangement. In this example, the pair (A,B) is the only pair of center stage switches that meets the above criteria. Since the pair (A,B) constitutes the entire set of pairs to be searched, (A,B) is the pair that requires the fewest rearrangements.

FIGURE 7, shows the X→A sequence and FIGURE 8 shows the X→B sequence of the immediate connect algorithm of the preferred embodiment. Together, FIGURES 7 and 8 show the operation of the flooding algorithm of the preferred embodiment. For example, the first step starts off in FIGURE 7 at row r_1 and finds an A in (r_1, c_2) which is circled. FIGURE 8 shows the second step of the flooding algorithm which uses the X→B sequence and starts off in column c_1 to find a circled B in (r_3, c_1) . Then, the third step takes place back in FIGURE 7 wherein column c_2 is examined for a B. This is found at position (r_2, c_2) and circled. Since there are no A's row r_2 , this X→A sequence is complete. The two symbols in FIGURE 7 indicate two rearrangements are required.

Although the flooding algorithm of the present method ends at the third step, if it were to continue, in FIGURE 8 row r_3 would be examined for an A, which would be found in position (r_3, c_3) . This procedure would continue to alternate between circling B's in columns and A's in rows until either a column with no B to circle or a row with no A to circle. The four circled symbols in

algorithm, a rearrangeable blocking condition is automatically detected.

Since the connection pair (1,1) is rearrangeably blocked, in order for a connection to be possible, there must exist an idle link between input switch 1 and some center stage switch, say A. Furthermore, there must exist an idle link between output switch 1 and some center stage switch, say B. The pair (A,B) is one pair of center stage switches to be considered for rearrangement. In this example, the pair (A,B) is the only pair of center stage switches that meets the above criteria. Since the pair (A,B) constitutes the entire set of pairs to be searched, (A,B) is the pair that requires the fewest rearrangements.

FIGURE 7, shows the X→A sequence and FIGURE 8 shows the X→B sequence of the immediate connect algorithm of the preferred embodiment. Together, FIGURES 7 and 8 show the operation of the flooding algorithm of the preferred embodiment. For example, the first step starts off in FIGURE 7 at row r_1 and finds an A in (r_1, c_2) which is circled. FIGURE 8 shows the second step of the flooding algorithm which uses the X→B sequence and starts off in column c_1 to find a circled B in (r_3, c_1) . Then, the third step takes place back in FIGURE 7 wherein column c_2 is examined for a B. This is found at position (r_2, c_2) and circled. Since there are no A's row r_2 , this X→A sequence is complete. The two symbols in FIGURE 7 indicate two rearrangements are required.

Although the flooding algorithm of the present method ends at the third step, if it were to continue, in FIGURE 8 row r_3 would be examined for an A, which would be found in position (r_3, c_3) . This procedure would continue to alternate between circling B's in columns and A's in rows until either a column with no B to circle or a row with no A to circle. The four circled symbols in

24 APR 1997

- 23 -

FIGURE 8, therefore, indicate four rearrangements are necessary in the example.

Since the X-A sequence requires only two rearrangements, and the X-B requires four rearrangements, the preferred embodiment chooses the X-A sequence for the pair (A,B). Since the pair (A,B) is the only pair of center stage switches being considered, the X-A sequence yields the minimum number of rearrangements.

The next step is to execute the rearrangements using the previously mentioned "hitless roll." Since the X-A sequence yields the minimum number of rearrangements, the preferred embodiment uses the X-A roll in this example. The hitless roll includes the initial connection placed in the dedicated center stage switch. This roll takes place by first performing the necessary rearrangement using a head-end-bridge and receive-end-switch.

FIGURES 9 through 14 illustrate executing the X-A hitless roll. The method changes the sequence of previously circled symbols A and B to clear the blocked cell as follows:

For the circled B's add X's as FIGURE 9 shows. In this example, this step corresponds to setting up a head-end-bridge through the center stage switch at the extra switch X for the existing connection from input switch 2 through center stage switch B and a receive-end-switch through center switch X for the existing connection and from center stage B to output stage 2.

The next step is to remove the circled B's as FIGURE 10 shows. In this example, when a connection exists through center stage switch X, and output switch 2 receives a good signal, the next step is to take down the connection from input switch 2 through center stage switch B to output switch 2.

Then, as FIGURE 11 shows for the circled A's add B's. In this example, this step corresponds to setting

24 APR 1997

- 24 -

up a head-end-bridge through the center stage switch B for the existing connection from input switch 1 through 2 enter stage switch A and a receive-end-switch through center stage switch B for the existing connection from center stage switch A to output switch 2. Then, as FIGURE 12 shows, the circled A's are removed. In this example, when a connection exists through center stage switch B and output switch 2 receives a good signal, the next step is to take down the connection from input switch 2 through center stage switch A to output switch 2.

Then, the next step is to add A's to the positions containing the X's, as FIGURE 13 shows. In this example, this corresponds to setting up a head-end-bridge through center stage switch A for both the connection from input switch 2 through center switch X to output switch 2 and the immediate connection from input switch 1 through center switch X to output switch 1.

The final step is to remove the X's as FIGURE 14 illustrates. In this example, when each connection through center stage switch A exists and the corresponding output switch receives a good signal, the corresponding connection from the input switch through center switch X to the corresponding output switch may be taken down. Note that the extra center stage switch X is left idle, since all connections are rolled off the extra enter stage switch X to the appropriate center stage switch.

Having described in detail the operation of the preferred method and system, the following discussion illustrates the connection path selection method and system of the preferred embodiment. For simplicity in describing this aspect of the present invention, the example considers connections in a further reduced $N(5,4,4)$ space matrix 220 of FIGURE 15.

24 APR 1997

- 25 -

FIGURE 15 illustrates the state of a simplified $N(5,4,4)$ space matrix 220 after initialization. Note that the fifth center stage switch 224 is labeled X to indicate that it is the dedicated or extra center stage switch. As legend 226 shows, short-dashed lines 228 indicate an idle link, continuous line 230 indicates a busy link, and long-dashed line 232 shows a switch connection. Since there are no inputs connected to any outputs in FIGURE 15, there are no input stage, center stage, or output stage switch connections and all of the links between the switches are idle.

With the simplified $N(5,4,4)$ space matrix in mind, it is possible to detail how to implement computer software to accomplish the connection path selection and other purposes of the present invention. Note, however, that the size of space matrix 220 does not limit the utility or applicability of the software implementation. To assist in understanding the software implementation, the following discussion introduces data definitions and notations relevant in manipulating the matrices 222, 234, 236, 238, and 240 of FIGURE 16. Note, also, that for ease in understanding connections, FIGURE 15 uses the same reference numerals for input gates 90, input stage switches 92, etc. appearing in the generalized $N(m,n,r)$ matrix of FIGURE 3. The following data definitions are useful in the description of the present connection method and system.

In a software implementation, input/output gate or line conversions occur. For example, a connection may be requested in the form of IL-m-OL-n. In other words, input line m to output line n. The input line number is converted to an input switch stage number i and an input number for that input switch stage j such that IL-m-I(i,j). Similarly, the output line number is converted to an output switch stage number k and an

24 APR 1997

- 26 -

output number for that output switch stage 1 such that $OL-n \rightarrow O(k, l)$.

More formally, $I(i, j)$ is determined from $IL-m$ and $O(k, l)$ is determined from $OL-n$ by:

$$i = 1 + \text{Quot}_N(m-1)$$

$$j = 1 + \text{Rem}_N(m-1)$$

$$k = 1 + \text{Quot}_N(n-1)$$

$$l = 1 + \text{Rem}_N(n-1)$$

where N is the number of first (input) stage switches, which is equal to the number of third (output) stage switches;

Quot_N is the integer quotient of the argument divided by N ; and

Rem_N is the integer remainder after the argument has been divided by N .

The following statements and data definitions may be understood with reference to FIGURES 15 and 16 to provide computer-generated matrices to accomplish the purposes of the present invention.

Data Definition:	$\text{space_in_mat}(\text{in_stage}, \text{in_num})$
------------------	---

In FIGURE 16, the space input matrix 222 for the FIGURE 15 $N(5, 4, 4)$ example is a 4×4 matrix which represents the connection of the input stage switches in the SSS such as switch 92. Each row of space input matrix 222 corresponds to one of the input stage switches 92. The columns of space input matrix 222 correspond to the input gates 90 to each of the first stage switches. The entries in the space input matrix correspond to the outputs of the first stage switches such as outputs 94. Since the output of each of the input stage switches is

connected to the center stage switch of the same number, the entries in the space input matrix may be viewed as the center stage switch to which that particular input stage switch (row), input (column) is connected.

As a result, the statement

statement:	<code>space_in_mat(i,j) = c</code>
------------	------------------------------------

causes space input matrix 222 to indicate a connection from the j^{th} input of the i^{th} first stage switch to center stage switch c .

The entries in space input matrix 222 range from 0 to X using the simplified example of FIGURE 15. A 0 indicates that no connection has been assigned. Entries of 1 through 4 are normal connections. A numerical entry in matrix 222 of 5, in this example corresponds to the entry of "X" in FIGURE 6 and indicates that the extra center stage switch 224 has been assigned and the rearrangement process has been invoked.

Data Definition:	<code>space_cnt_mat[cnt_stage,in_stage]</code>
------------------	--

Space center matrix 234 is a 5x4 matrix which represents the connection map of the center switches in the space domain such as center stage switches 96. Each row of the space center matrix corresponds to one of the center stage switches 96. The columns of the space center matrix correspond to the inputs to each of the center stage switches such as inputs 94. The entries in the space center matrix correspond to the outputs of the center stage switches such as output 98. Since the output of each of the center stage switches 96 connects to the output stage switch of the same number, the entries in the space center matrix 234 may be viewed as

24 APR 1997

- 28 -

the output stage switch to which that particular center stage switch (row), input column connects.

As a result, the statement

Statement:	<code>space_cnt_mat(c,j) = k</code>
------------	-------------------------------------

causes space center matrix 234 to indicate a connection from the j^{th} input of the center stage switch "c" to the x^{th} output stage switch. The entries in space center matrix 234 range from 0 to 4. A 0 indicates that no connection has been assigned. Entries of 1 through 4 are normal connections. A zero entry in x^{th} row (i.e., the 5th row in this example) indicates that the extra center stage switch 224 has been assigned and the rearrangement process has been invoked.

Data Definition:	<code>space_out_mat[out_stage,out_num]</code>
------------------	---

Space output matrix 236 is a 4x4 matrix which represents the connection map of the output stage of the space matrix in the space domain. Each row of space output matrix 236 corresponds to one of the output switches such as output switch 100. The columns of space output matrix 236 correspond to the output from each of the output switches such as output gates 102. The entries in space output matrix 236 correspond to the input to which that output connects such as connections 98. Since the input of each output switch connects to the center stage switch of the same number, the entries in space output matrix 236 may be viewed as the center stage switch to which that particular output switch 100 (row), output gate 102 (column) connects.

As a result, the statement

Statement:	<code>space_out_mat(k,l) = c</code>
------------	-------------------------------------

24 APR 1997

- 29 -

causes space output matrix 236 to indicate a connection from center stage switch "c" to the 1th output of the kth output stage switch. The entries in space output matrix range 236 from 0 to X. A 0 indicates that no connection has been assigned. Entries of 1 through 4 are normal connections. An entry of X (i.e., 5 in this example) indicates that extra center stage switch 224 has been assigned and the rearrangement process has been invoked.

Continuing with reference to FIGURES 15 and 16, the following discussion details how to implement the preferred embodiment to represent an idle link between an input gate and an output gate. In normal operation, the present connection method attempts to make a given connection through the lowest-numbered center stage switch. In order to make a connection from a given input stage switch, such as input switch 92, to a given output stage switch such as output switch 100, there must exist a certain center stage switch, such as center switch 96, with the property that there is an idle input link from input switch 92 to center switch 96 and an idle output link from center switch 96 to output switch 100.

The present method and system track idle input links and idle output links by using two different bit map arrays (one for input links and one for output links). Each bit array map has a separate row for each row of the space center matrices. The method uses a group of primitives to operate upon the bit maps to effect the connect and disconnect operations. The primitives, in turn, invoke the rearrangement process when required.

Data Definition:	space_in_link[in_stage,cnt_stage]
------------------	-----------------------------------

For the example of FIGURES 15 and 16, space input idle link bit map 238 is a 4x5 bit array that represents the idle input links 94 between the input switches 92 and the center switches 96 in the space domain. Each row of

24 APR 1997

- 30 -

space input idle link bit map 238 corresponds to one of the input stage switches 92. The entries in space input idle link bit map 238 for a given row (input stage) and column (center stage) indicate the idle state of the input link 94 between that input stage and center stage. An entry of 1 indicates that the input link is idle and available for assignment. A 0 value indicates that the input link is in use and is not available for assignment. A 0 entry in column X indicates that the extra center stage switch 234 is in use and the rearrangement method is in progress.

Data Definition:	space_out_link[out_stage,cnt_stage]
------------------	-------------------------------------

Space output idle link bit map 240 is a 4x5 bit array that represents the idle output links between the output and the center switches 96 in the space domain. Each row of space output idle link bit map 240 corresponds to one of the output stage switches 100. The entries in the space output idle link bit map 240 for a given row (output switch) and column (center switch) indicate the idle state of the output link 98 between that output switch and center switch. An entry of 1 indicates that the output link is idle and available for assignment. A 0 value indicates that the output links is in use and is not available for assignment. A 0 entry in column X indicates that the extra center stage switch 224 is in use and the rearrangement method is in progress.

Having described the maps for representation idle links, the following describes primitive operations useful to manipulate the space idle link bit maps 238 and 240 for connect and disconnect operations.

Primitive:	init_links()
------------	--------------

24 APR 1997

- 31 -

The initialize idle link bit maps primitive (used only for start-up operations) creates both space input idle link bit map 238 and space output idle link bit map 240 for the preferred embodiment. All entries in the idle link bit maps are initialized to 1, indicating that all links are idle.

Primitive:	set_link(i/o, j, k)
------------	---------------------

This primitive sets the k^{th} bit of the j^{th} row of the bit map specified by the "i/o" parameter to 1. If the bit map contains n elements and the value of j exceeds n , then the computer returns an error flag. Otherwise, the k^{th} element is set to 1.

Primitive:	clear_link (i/o, j, k)
------------	------------------------

This primitive clears (resets) the k^{th} bit of the j^{th} row of the bit map specified by the i/o parameter to 0. If the bit map contains n elements and the value of j exceeds n then the computer returns an error flag. Otherwise, the k^{th} element is set to 0.

Primitive:	pump_link(space_in_link[i],space_out_link[k])
------------	---

The preferred method and system connect between an input switch stage i and an output switch stage k using the lowest numbered center stage switch available. This is accomplished by a process that hereinafter shall be called "pumping" space_n_link[i] and space_out_link[k]. That is, space_in_link[i] is logically "ANDed" with space_out_link[k] to determine c , the lowest numbered center stage switch with both an idle link to input stage

24 APR 1997

- 32 -

i and an idle link to output stage k. The index of the first non-zero entry in the result of the logical AND is c, the lowest numbered center stage switch with both an idle link to input stage i and an idle link to output stage k. The two links to c are then marked as "busy" in space_in_link[i] and space_out_link[k] and the center stage number c is returned. Table 3 illustrates an embodiment of this primitive macro.

TABLE 3
PUMP LIST MACRO FLOW CHART
<pre> pump_link(space_in_link[i], space_out_link[k] int c, i, k, x[xtra_stage]; { int j; x = space_in_link[i] && space_out_link[k]; for (j = 1; j <= xtra_stage; j++) { if (X(j) == 1) { c = j; break; } } clear_link(space_in_line, i, c); clear_link(space_out_link, k, c); if (c == xtra_stage) { rearrange = 1; } return(c); } </pre>

FIGURE 16 illustrates the implementation data matrices after initialization. The corresponding entries

24 APR 1997

- 33 -

in the connection path selection matrices `space_in_mat` 222, `space_cnt_mat` 234, and `space_out_mat` 236, for the example, are illustrated in FIGURE 16. Since there are no switch connections, all of the entries are 0. The contents of the `space_in_link` 238 and `space_out_link` 240 bit maps are as shown at initialization. Note that each input stage switch and each output stage switch has a 1 element for each idle center stage switch. In this example, after initialization, all five center stage switches are marked as "idle".

FIGURE 17 shows the implementation data after the first connection. Suppose the first connection requested is to connect input line 1 to output line 1, or more simply, IL-1-OL-1. The first step is to convert the input line number to an input switch stage number and an input number for that input switch stage. In this case, IL-1-I(1,1). Similarly, the output line number is converted to an output switch stage number and an output number for that output switch stage. In this case, OL-1-O(1,1). So, we are going to make a connection between input stage 1 and output stage 1.

The present method and system attempt to make the connection between input stage 1 and output stage 1 by using the lowest numbered center stage switch available, which in this case is center stage 1. This is accomplished by using the `pump_list` macro of TABLE 3 to compare `space_in_link[1]` with `space_out_link[1]` in FIGURE 16 and set the idle connects of the lowest center stage number common to both bit maps to "0". Then, the `space_in_mat(i,j)` and the `space_out_mat(k,l)` are set equal to that center stage number, and the `space_cnt_mat(c,j)` is set equal to k. The corresponding space matrix data entries are shown in FIGURE 17 and the resultant state of the space matrix 220 is shown in FIGURE 18.

The following statements accomplish this process:

24 APR 1997

- 34 -

TABLE 4
CONNECT STATEMENTS
<pre> x = pump_list(space_in_link[i], space_out_link[k]); space_in_mat(i,j) = c; space_cnt_mat(c,i) = k; space_out_mat(k,l) = c; where, in this case, i=j=k=l=m=c-1. </pre>

Assume the procedure of adding connections continues until n^2-2 connections have been made (presumably without rearrangement) with the resultant space matrix shown in FIGURE 19 and connection path selection matrix data shown in FIGURE 20.

Then, the request to connect IL-13 to OL-7 is received. The request to connect IL-13 to OL-7 converts to a request connect $I(4,1) \rightarrow O(2,3)$. The present method and system attempt to make the connection between input stage 4 and output stage 2 by using the lowest numbered center stage switch available, which in this case, is center stage switch number 4, as illustrated in FIGURE 20. The pump_list macro of Table 1 compares space_in_link[4] with space_out_link[2], determines the lowest center stage number common to both bit maps, and sets the idle status to "busy" in both bit maps. Then the space_in_mat(i,j) and the space_out_mat(k,l) are set equal to that center stage number and the space_cnt_mat(c,j) is set equal to k.

The corresponding space matrix data entries are shown in FIGURE 21 and the resultant state of the space matrix is shown in FIGURE 22. This process is accomplished using the following statements from the connect routine as before:

24 APR 1997

- 35 -

TABLE 5

CONNECT STATEMENTS

```
x = pump_list(space_in_link[i], space_out_link[k]);  
  space_in_mat(i,j) = c;  
  space_cnt_mat(c,i) = k;  
  space_out_mat(k,l) = c;  
where in this case, i=4 j=1 k=2 l=3 m=13 n = 7 and c=4.
```

Suppose the next request is to disconnect IL-13 from OL-7 (which takes down the connection that was just made). The request to disconnect IL-13 from OL-7 converts to a request to disconnect I(4,1)→O(2,3). The put_back macro accomplishes the disconnect by breaking the connection between input stage 4 and output stage 2 and putting the freed-up center stage switch, number 4, back on the input/output stacks for subsequent use. More specifically, the put_back macro determines the freed-up center stage switch number and puts it back in the proper positions on both of the ordered-linked lists in space_in_link[4] and space_out_link[2]. Then, the space_in_mat(i,j), the space_cnt_mat(c,j), and the space_out_mat(k,l) are set equal to 0.

The corresponding space matrix data entries are shown in FIGURE 23 and the resultant state of the space matrix is same as shown before FIGURE 18. The disconnect is accomplished using the following statements from the put_back macro:

24 APR 1997

- 36 -

TABLE 6

DISCONNECT STATEMENTS

```

c = space_in_mat (i,j);
  set_link(space_in_link, i, c);
  set_link(space_out_link, k, c);
  space_in_mat(i,j) = 0;
  space_cnt_mat(c,i) = 0;
  space_out_mat(k,l) = 0;
where in this case, i=4 j=1 k=2 l=3 m=13, n=7, and c=4.

```

Next, suppose we receive a request to connect IL-13 to OL-12. The request to connect IL-13 to OL-12 converts to a request to connect I(4,1)→O(3,4). The present method and system attempt to make the connection between input stage 4 and output stage 3 by using the lowest numbered center stage switch available. In this case, the x^{th} or extra center stage switch 224 is the lowest number available center stage switch, as illustrated in FIGURE 25.

As before, the pump_list macro accomplishes this by comparing space_in_stack[4] with space_out_stack[3] in FIGURE 23, determining the lowest center stage number common to both bit maps (which in this case is center stage switch "X"), and setting the idle status to "busy" in both bit maps. Then, the space_n_mat(i,j) and the space_out_mat(k,l) are set equal to that center stage number and the space_cnt_mat(c,j) is set equal to k. An entry of 5 indicates that the x^{th} center stage switch has been assigned and the rearrangement process is automatically invoked. The corresponding space matrix data entries during rearrangement are shown in FIGURE 24 and the resultant state of the space matrix during arrangement is shown in FIGURE 25.

24 APR 1997

- 37 -

The following immediate connect statements accomplish this:

TABLE 7
IMMEDIATE CONNECT STATEMENTS
<pre> x = pump_list(space_in_stack[i],space_out_stack[k]); space_in_mat(i,j) = c; space_cnt_mat(c,i) = k; space_out_mat(k,l) = c; where in this case, i=4 j=1 k=3 l=4 m=13, n=12, and c=5. </pre>

As a result of the rearrangement algorithm the connection $I(4,3) \rightarrow O(2,1)$ that FIGURE 26 shows is "rolled" from center stage switch number 2 onto center stage switch number 4. This roll is performed hitlessly by creating a head-end-bridge at input stage switch 4 and a receive-end-switch at output stage switch 2 as shown in FIGURE 27. When the correct path is received at output stage switch 2 from center stage switch 4, the original connection through center stage switch 2 is dropped as shown in FIGURE 28 and the space matrix data is updated as shown in FIGURE 29.

The previous roll now allows the immediate connection, $I(4,1) \rightarrow O(3,4)$, as FIGURE 31 shows, which is on center stage switch X 224, to be rolled onto center stage switch 2.

This roll is performed hitlessly by creating a head-end-bridge at input stage switch 4 and an receive-end-switch at output stage switch 3 as shown in FIGURE 31. When the correct path is received at outputs stage switch 3 from center stage switch 2, the original connection through center stage X 224 is dropped.

24 APR 1997

- 38 -

Rolling the immediate connection, $I(4,1) \rightarrow O(3,4)$, off the extra center stage switch completes the rearrangement process for this connection. Note that this process leaves the extra center stage switch clear and ready for a next connection when a rearrangeable blocking condition occurs. FIGURE 32 shows the resulting state of the space matrix after rearrangement. FIGURE 33 shows the corresponding space matrix data entries after rearrangement.

In FIGURE 34, the rearrangement aspects of the preferred embodiment are applied to the above example appear. Once the preferred embodiment uses dedicated center stage switch 224 to make the immediate connection, the system of the preferred embodiment invokes the rearrangement process automatically. First, the system determines two center stage switches which will participate in the rearrangement process. This is accomplished using Paull's Method which is restated as follows:

If (r_1, c_1) is blocked, test all symbol pairs (A, B) , such that A is in row r_1 , but not in column c_1 and B is in c_1 , but not in r_1 , to find which pair will require the fewest changes. The method then carries out the changing algorithm on that pair.

Returning to the space matrix example of FIGURE 25, it is preferred to first transform the space_cnt_mat matrix data in FIGURE 24 into the format of FIGURE 4. FIGURE 34 illustrates the results of this transformation.

The use of the dedicated center stage switch is indicated by the X in (r_4, c_3) of FIGURE 34. The next step is to find all symbol pairs (A, B) such that A is in row r_4 , but not in column c_3 , and B is in column c_3 , but not in row r_4 . In this example, there is a 2 in row r_4 , but no 2 in column c_3 , so A is set to 2. Similarly, there is a 4 in column c_3 , but no 4 in row r_4 , so B is set to 4. Since there is only one value for A and one value for B,

the (A,B) pair that requires the least changes is, obviously, the pair (2,4).

For a given pair of symbols (A,B) there are two possible rearrangement sequences. One of these two generally will be shorter. The following steps generate the X→A and X→B sequences for this example.

The X→A sequence starts with the immediate connection through the extra element and ends up with the blocked cell being assigned to element A or 2, in this case. Starting with the blocked cell (r_3, c_4) , there is no 4 in row r_3 , so this step requires a circle around the 2 in row r_3 . Now, the circled 2 is in (r_4, c_2) . Then, a search for a 4 in column c_2 is necessary. Since there is none, this sequence is complete. (Note that not all 2's get circled.)

The X→B sequence starts with an immediate connection through the dedicated element 224 and ends up with the blocked cell being assigned to element B or 4, in this case. Starting with the blocked cell (r_3, c_4) , there is no 2 in c_4 , so a circle goes around the 2 in c_4 . Now, the circled 4 is in (r_3, c_3) . Then, a search for a 2 in r_3 takes place. Since there is none, the sequence generation process is complete. (Note that not all 4's get circled.)

The flooding algorithm starts by finding the first element of the X→A sequence. If the X→A sequence is not ended, the first element of X→B sequence is found. Then, if the X→B sequence is not ended, the next element of the X→A sequence is found. This process continues until the shorter of the two sequences is completed. (So, the X→B sequence shown in FIGURE 34 is not really generated. It is shown here as an illustration of the alternate path.)

In summary, the method and system of the present invention determine the rearrangement sequence for a rearrangeably blocked communications matrix that for a minimum number of rearrangements of the communications

24 APR 1997

- 40 -

matrix by representing the communications matrix as a square matrix having a first dimension that represents input stages of the communications matrix and a second dimension that represents output stages of the communications matrix, and where cells at the intersection of the first dimension and the second dimension represent center stage switches of the communications matrix so that the matrix identifies a blocked center stage switch of the communications matrix in terms of the first dimension and the second dimension and wherein a pair of center stage switches where one of the center stage switches is not associated with the same output stage switch as the blocked center stage switch and the other center stage switch of the pair is not associated with the same input stage switch as the blocked center stage switch. In parallel, then, the number of rearrangement steps necessary for a rearrangement sequence using one of the center stage switches of the pair and the rearrangement sequence through the communications matrix using the other center stage switch are determined. With these step numbers being determined in parallel, the method and system terminate the determining operations upon first determining the shorter of the rearrangement sequences using the pair of switches.

As a result of the above, although the invention has been described with reference to the above embodiments, its description is not meant to be construed in a limiting sense. Various modifications of the disclosed embodiment, as well as alternative embodiments of the invention will become apparent to person skilled in the art upon reference to the above description. It is therefore contemplated that the appended claims will cover such modifications that fall within the true scope of the invention.

24 APR 1997

- 41 -

APPENDIX A

```

.....
.....
/*
/*      Bowdon's Immediate Connect Algorithm (BICA)
/*
/*
.....
.....

/* To begin proceed to the main program at the end of the the listing. */

#include <stdio.h>
#include <math.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <sys/time.h>

/* Global variables for immediate connection algorithm. */
int      in_line,
         out_line;
int      in_stage,
         in_num;
int      out_stage,
         out_num;
int      n,
         m;
int      command_count,
         command_end = 10;
int      rearrange_space,
         rearrange_time;

int      space_in_mat[16][16];
int      space_cnt_mat[17][16];
int      space_out_mat[16][16];
int      space_in_link[16][17];
int      space_out_link[16][17];
int      xtra_stage = 17;
int      traffic[2][28672];

/* Global variables for rearrangement algorithm. */
int      BTOA_MV[31][3],
         ATOB_MV[31][3];
int      flag_A = 0,
         flag_B = 0;

int      *sort_array ();
int      *chk_space_in_status ();
int      *chk_space_out_status ();
int      *chk_space_blocking ();
int      *shuffle ();
int      *implement_PM2 ();

char      name_in[13];
char      name_out[13];

static int in_file[22];
FILE      *fp_in,
         *fp_out;
int      seed_array[10];
int      p_level;

```

24 APR 1997

- 42 -

APPENDIX A

```

/* Structure to hold current network permutation before rearranging. */
struct p_coordinate
{
    int      x;
    int      y;
    int      z;
};

struct p_mat_pos
{
    struct p_coordinate position[256];
};

struct p_mat_pos rearrange_3d;

/* Structure to hold the output of the rearrangement sequence. */
struct seq_len
{
    int input_sw;
    int old_mid_sw;
    int new_mid_sw;
    int output_sw;
};

struct rearrange_seq
{
    struct seq_len move[15];
};

struct rearrange_seq rearrange_moves;

/*****
/*
/* This is the subprogram for BICA's number of moves from B calc.
/*
*****/

/*****
/* This function first reads the rearrangement structure and picks out
/* the connections in the middle switch pair (A, B). The arguments to
/* this function are the starting point (RB, CB) and the pair (A, B).
/* The function returns the number of moves starting from B. For
/* reference, the starting point is described by Paull as (R1, C2).
*****/

chk_Paull_mat_BTOA (RB, CB, B_least, A_least)
int      RB,
          CB;
int      A_least,
          B_least;
{
    int      i,
             j,
             m = 0,
             n = 0,
             y1 = 0,
             y2 = 0,
             y3 = 0,
             n1 = 0,

```

24 APR 1997

- 43 -
APPENDIX A

```

int      n2 = 0;
         x1 = 0;
         x2 = 0;
         x3 = 0;
         k1 = 0;
         m1 = 0;
         m2 = 0;
int      flag1 = 0;
         flag2 = 0;
         flag3 = 0;
         temp_flag = 0;
         k2 = 0;
int      r_temp = 0;
         c_temp = 0;
         F2 = 0;
int      chosen_msw_A[16][3];
         chosen_msw_B[16][3];

c_temp = CB;
m = 1;
n = 2;

zero (chosen_msw_A, 48);          /* Initialize target matrices. */
zero (chosen_msw_B, 48);

for (i = 0; i < 256; i++)
{
    x1 = rearrange_3d.position[i].x; /* Read rearrangement struct. */
    x2 = rearrange_3d.position[i].y;
    x3 = rearrange_3d.position[i].z;

    if (x3 == B_least)
    {
        chosen_msw_B[m1][0] = x1; /* Pick out connections from */
        chosen_msw_B[m1][1] = x2; /* middle switch B and store */
        chosen_msw_B[m1][2] = x3; /* in chosen_msw_B. */
        m1++;
    }

    if (x3 == A_least)
    {
        chosen_msw_A[m2][0] = x1; /* Pick out connections from */
        chosen_msw_A[m2][1] = x2; /* middle switch A and store */
        chosen_msw_A[m2][2] = x3; /* chosen_msw_A. */
        m2++;
    }
}

while (temp_flag == 0)
{
    k1 = 0;
    for (i = 0; i < 16; i++)          /* Start from (R1, C2), check */
    {                                  /* A's in the same column. */
        x1 = chosen_msw_A[i][0];
        x2 = chosen_msw_A[i][1];
        x3 = chosen_msw_A[i][2];

        if (x2 == c_temp && x3 == A_least) /* Found an B? */
        {
            BTOA_MV[m][0] = x1; /* Yes, then store current */

```

24 APR 1997

- 44 -

APPENDIX A

```

      BTOA_MV[m][1] = x2;      /* connection : BTOA_MV. */
      BTOA_MV[m][2] = x3;
      m++;
      r_temp = x1;
      break;
    }

    k1++;
  }

  if (k1 == 16)
  {
    flag1 = 1;
    temp_flag = 1;
    F2 = m;
  }

  if (flag1 == 0)
  {
    k2 = 0;
    for (i = 0; i < 16; i++)      /* Start from (R1, C2), check */
    {                               /* B's in the same row. */

      x1 = chosen_msw_B[i][0];
      x2 = chosen_msw_B[i][1];
      x3 = chosen_msw_B[i][2];

      if (x1 == r_temp && x3 == B_least)      /* Found an B? */
      {
        BTOA_MV[m][0] = x1;      /* Yes, then store */
        BTOA_MV[m][1] = x2;      /* current connection */
        BTOA_MV[m][2] = x3;      /* in BTOA_MOV. */
        m++;
        c_temp = x2;
        break;
      }

      k2++;
    }

    if (k2 == 16)
    {
      flag2 = 1;
      temp_flag = 1;
      F2 = m;
    }
  }

  return (F2);      /* Return the number of moves from B. */
}

/*-----*/
/*
/* This is the subprogram for BICA's number of moves from A calc.
/*
/*-----*/

/*-----*/
/* This function first reads the rearrangement structure and picks out
/* the connections in the middle switch pair (A, B). The arguments to
/*

```

24 APR 1997

- 45 -

APPENDIX A

```

/* this function are the starting point (RA, CA) and the pair (A, B). */
/* The function returns the number of moves starting from A. For */
/* reference, the starting point is described by Paull as (R2, C1). */
/.....

```

```
chk_Paull_mat_ATOB (RA, CA, B_least, A_least)
```

```
int      RA;
          CA;
int      B_least;
          A_least;
{

```

```

    int      i,
             j,
             m = 0,
             n = 0,
             y1 = 0,
             y2 = 0,
             y3 = 0,
             n1 = 0,
             n2 = 0;
    int      x1 = 0,
             x2 = 0,
             x3 = 0,
             k1 = 0,
             m1 = 0,
             m2 = 0;
    int      flag1 = 0,
             flag2 = 0,
             flag3 = 0,
             temp_flag = 0,
             k2 = 0;
    int      r_temp = 0,
             c_temp = 0,
             F1 = 0,
             F2 = 0;
    int      chosen_msw_A[16][3],
             chosen_msw_B[16][3];

```

```

    r_temp = RA;
    m = 1;
    n = 2;

```

```

    zero (chosen_msw_A, 48);          /* Initialize target matrices. */
    zero (chosen_msw_B, 48);

```

```
    for (i = 0; i < 256; i++)
    {

```

```

        x1 = rearrange_3d.position[i].x; /* Read rearrangement struct. */
        x2 = rearrange_3d.position[i].y;
        x3 = rearrange_3d.position[i].z;

```

```
        if (x3 == B_least)
        {

```

```

            chosen_msw_B[m1][0] = x1; /* Pick out connections from */
            chosen_msw_B[m1][1] = x2; /* middle switch B and store */
            chosen_msw_B[m1][2] = x3; /* in chosen_msw_B. */
            m1++;
        }
    }

```

- 46 -

APPENDIX A

```

if (x3 == A_least)
{
    chosen_msw_A[m2][0] = x1; /* Pick out connections from */
    chosen_msw_A[m2][1] = x2; /* middle switch A and store */
    chosen_msw_A[m2][2] = x3; /* in chosen_msw_A. */
    m2++;
}
}

while (temp_flag == 0)
{
    k1 = 0;
    for (i = 0; i < 16; i++) /* Start from (R2, C1), check */
    { /* B's in the same row. */
        x1 = chosen_msw_B[i][0];
        x2 = chosen_msw_B[i][1];
        x3 = chosen_msw_B[i][2];

        if (x1 == r_temp && x3 == B_least) /* Found an A? */
        {
            ATOB_MV[m][0] = x1; /* Yes, then store current */
            ATOB_MV[m][1] = x2; /* connection in ATOB_MV. */
            ATOB_MV[m][2] = x3;
            m++;
            c_temp = x2;
            break;
        }
        k1++;
    }

    if (k1 == 16)
    {
        flag1 = 1;
        temp_flag = 1;
        F1 = m;
    }

    if (flag1 != 1)
    {
        k2 = 0;
        for (i = 0; i < 16; i++) /* Start from (R2, C1), check */
        { /* A's in the same column. */
            x1 = chosen_msw_A[i][0];
            x2 = chosen_msw_A[i][1];
            x3 = chosen_msw_A[i][2];

            if (x2 == c_temp && x3 == A_least) /* Found an A? */
            {
                ATOB_MV[m][0] = x1; /* Yes, then store */
                ATOB_MV[m][1] = x2; /* current connection */
                ATOB_MV[m][2] = x3; /* in ATOB_MV. */
                m++;
                r_temp = x1;
                break;
            }
            k2++;
        }

        if (k2 == 16)
    }
}

```

24 APR 1997

- 47 -
APPENDIX A

```

    {
        flag2 = 1;
        temp_flag = 1;
        Fl = m;
    }
}
return (Fl); /* Return the number of moves from A. */
}

/*****
/* This is the subprogram for BICA's rearrange output.
/*
*****/

/*****
/* This function is the output of the rearrangement procedure. The
/* arguments to this function are the required number of moves and the
/* pair (A, B). The sequence of moves is written into the output
/* structure rearrange_moves.
*****/

/*****
/* The output of the rearrangement process is a structure of the form:
/*
/*      struct seq_len
/*      {
/*          int input_sw;
/*          int old_mid_sw;
/*          int new_mid_sw;
/*          int out_sw;
/*      }
*****/

int      rearrangement_output (no_moves, AR, BR)
int      no_moves;
int      AR;
int      BR;
{
    int      i,
             j,
             k,
             m,
             n;
    int      x1 = 0,
             x2 = 0,
             x3 = 0;
    int      y1 = 0,
             y2 = 0,
             y3 = 0;

    for (j = 0; j < no_moves; j++)
    {
        if (flag_A == 1 && flag_B == 0) /* Is sequence from A smaller? */
        {
            y1 = ATOB_MV[j][0]; /* Read Input switch */
            y2 = ATOB_MV[j][1]; /* Read Output switch */

```

24 APR 1997

- 48 -

APPENDIX A

```

    y3 = ATOB_MV[j][2]; /* Read Middle switch */
}

if (flag_B == 1 && flag_A == 0) /* Is sequence from B smaller? */
{
    y1 = BTOA_MV[j][0]; /* Read Input switch */
    y2 = BTOA_MV[j][1]; /* Read Output switch */
    y3 = BTOA_MV[j][2]; /* Read Middle switch */
}

rearrange_moves.move[j].input_sw = y1; /* Store Input switch */
rearrange_moves.move[j].old_mid_sw = y3; /* Store Old Middle switch */
rearrange_moves.move[j].output_sw = y2; /* Store Output switch */

/* Store New Middle switch */
if (y3 == AR) /* Change A's to B's. */
    rearrange_moves.move[j].new_mid_sw = BR;

if (y3 == BR) /* Change B's to A's. */
    rearrange_moves.move[j].new_mid_sw = AR;
}
return;

/*.....*/
/*
/* This is the subprogram for BICA's zero function.
/*
/*.....*/

int      zero (A, n)
int      *A;
int      n;
{
    int      i;

    for (i = 0; i < n; i++)
        *(A + i) = 0;
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's hunt algorithm starting point set.
/*
/*.....*/

/*.....*/
/* This function sets the starting points for the hunting algorithm
/* from A & B and executes the algorithm. The set of connections to be
/* moved are stored in either ATOB_MV or BTOA_MV if either the sequence
/* from A is shorter or the sequence from B is shorter, respectively.
/* The arguments to this function are the middle switch pair (A,B),
/* an index and the blocked matrix position (R1,C1). The function
/* returns the length of the shortest sequence.
/*.....*/

```

24 APR 1997

- 49 -

APPENDIX A

```

int      rearrange_3d (rl_B, cl_A, sw_index, R1, C1)
int      rl_B;
int      cl_A;
int      sw_index;
int      R1;
int      C1;
(
    int      i,
              j,
              k = 0,
              x1 = 0,
              x2 = 0,
              x3 = 0,
              m,
              n,
              temp_mov = 0,
              temp = 0;
    int      num_moves = 0,
              mr = 0,
              mc = 0,
              w_seq = 0;
    int      R[3],
              C[3],
              undef_flag = 0;
    int      num_moves_A = 0,
              num_moves_B = 0,
              F1 = 0;
    int      F2 = 0,
              R1_CONN[16][3],
              C1_CONN[16][3];

    zero (R, 15);
    zero (C, 15);
    zero (ATOB_MV, 93);
    zero (BTOA_MV, 93);
    zero (R1_CONN, 48);
    zero (C1_CONN, 48);

    m = 0;
    n = 0;

    R[0] = R1;
    C[0] = C1;

    BTOA_MV[0][0] = R[0];
    ATOB_MV[0][1] = C[0];

    for (i = 0; i < 256; i++) /* Pick out middle switches in (R1,C1). */
    {
        x1 = rearrange_3d.position[i].x; /* Read structure. */
        x2 = rearrange_3d.position[i].y;
        x3 = rearrange_3d.position[i].z;

        if (x1 == R1)
        {
            R1_CONN[m][0] = x1; /* Store middle switches in R1. */
            R1_CONN[m][1] = x2;
            R1_CONN[m][2] = x3;
            m++;
        }
    }

```

24 APR 1997

- 50 -

APPENDIX A

```

if (x2 == C1)
{
    C1_CONN[n][0] = x1;          /* Store middle switches in C1. */
    C1_CONN[n][1] = x2;
    C1_CONN[n][2] = x3;
    n++;
}

for (i = 0; i < 15; i++)        /* Find the matrix position R2, C2. */
{
    x1 = R1_CONN[i][2];          /* R1_CONN is the connections in R1 */
    x2 = C1_CONN[i][2];          /* C1_CONN is the connections in C1 */

    if (x1 == r1_B)
        C[1] = R1_CONN[i][1];
    if (x2 == c1_A)
        R[1] = C1_CONN[i][0];
}

BTOA_MV[0][1] = C[1];           /* Set start point for search from A&B. */
ATOB_MV[0][0] = R[1];           /* A is from (R2, C1); B from (R1, C2). */

BTOA_MV[0][2] = r1_B;           /* The 3rd element of BTOA_MV & ATOB_MV */
ATOB_MV[0][2] = c1_A;           /* is alternating sequence of A's & B's */
                                /* A's are changed to B's & B's to A's. */

/* Set starting point for A. */
mr = R[1];
mc = C[0];

/* Hunting algorithm from A. */
num_moves_A = chk_Paull_mat_ATOB (mr, mc, r1_B, c1_A); /* REPLACE */
                                                    /* w/ FLOOD */

/* Set starting point for B. */
mr = R[0];
mc = C[1];
/* Hunting algorithm from B */

num_moves_B = chk_Paull_mat_BTOA (mr, mc, r1_B, c1_A); /* REPLACE */
                                                    /* w/ FLOOD */

F1 = num_moves_A;
F2 = num_moves_B;

if (F1 < F2)                    /* Is num_moves_A < num_moves_B ? */
{
    w_seq = 1;                  /* If yes, set flag_A = 1, flag_B = 0. */
    num_moves = F1;
    flag_A = 1;
    flag_B = 0;
}

if (F2 < F1)                    /* Is num_moves_B < number_moves_A ? */
{
    w_seq = 2;                  /* If yes, set flag_A = 0, flag_B = 1. */
    num_moves = F2;
    flag_A = 0;
    flag_B = 1;
}

```

24 APR 1997

- 51 -

APPENDIX A

```

    }
    if (F1 == F2)
    {
        w_seq = 1;
        num_moves = F1;
        flag_A = 1;
        flag_B = 0;
    }
    return (num_moves);
}

/*
.....
/* This is the subprogram for BICA's implement change algorithm.
.....
/*
.....
/* This function implements the hunting algorithm to find the sequence
/* of moves required for rearrangement. The arguments to this function
/* are the middle switch pair (msw_A, msw_B) and the index.
.....
int      implement_change (msw_B, msw_A, index)
int      msw_B;
int      msw_A;
int      index;
{
    int      i,
            j,
            k;
    int      len_of_seq = 0;

    len_of_seq = rearrange_seq (msw_B, msw_A, index);
    rearrangement_output (len_of_seq, msw_A, msw_B);
    return (len_of_seq);
}

/*
.....
/* This is the subprogram for BICA's calculate number of moves.
.....
/*
.....
/* This routine uses the implement_change_algorithm function to find
/* the number of moves required for the sequence of moves from sw_A and
/* and for the sequence of moves from sw_B for the existing connections
/* in the center stage switch pair (A, B) and returns the number of
/* moves required for the smaller of the two sequences.
.....
int      calc_nom (sw_B, sw_A, index)
int      sw_B,
        sw_A,
        index;
{

```

24 APR 1997

- 52 -

```

int          num_moves;
num_moves = implement_change (sw_B, sw_A, index);
return(num_moves);
}

/*.....*/
/* This is the subprogram for BICA's matrix model conversion. */
/*.....*/

/*.....*/
/* The present implementation of Paull's method needs the connect data */
/* in the following structure. */
/*.....*/
/*      struct p_coordinate */
/*      { */
/*          int input_sw; */
/*          int output_sw; */
/*          int cnt_sw; */
/*      } */
/*.....*/
/*      struct p_mat_pos */
/*      { */
/*          struct p_coordinate{256}; */
/*      } */
/*.....*/

void          matrix_model_conversion ()
{
    int          i,
                 j,
                 x1,
                 x2,
                 x3,
                 m1 = 0;

    for(i = 1; i <= 17; i++)
    {
        for(j = 1; j <= 17; j++)
        {
            x1 = i;
            x2 = j;
            x3 = space_cnt_mat[i][j];
            if(x3 != 0)
            {
                rearrange_3d.position[m1].x = x2; /* Input switch */
                rearrange_3d.position[m1].y = x3; /* Output switch */
                rearrange_3d.position[m1].z = x1; /* Middle switch */
                m1++;
            }
        }
    }
    return;
}

/*.....*/
/* This is the subprogram for BICA's version of Paull's method II. */
/*.....*/

```

24 APR 1997

- 53 -

APPENDIX A

```

/* ..... */
/* This function uses Paull's method II to search for (A, B) pairs. */
/* ..... */

int      *implement_PM2 ()
{
    int      i,
             j,
             k,
             x1,
             x2,
             m = 0,
             n = 0;

    int      *selected_pair,
             R1_cnt_sw[16],
             C1_cnt_sw[16];

    int      listofB[16],
             listofA[16],
             A_temp = 0,
             B_temp = 0;

    int      cnt_stage_pairs[8][2],
             op_flag;

    int      num_moves,
             least_moves,
             least_pair;

    /* convert from 3 matrix model to rearrangement structure. */
    /* Return pointer to structure or Global must be visible. */

    matrix_model_conversion ();      /* Converts 2nd of 3-matrix model. */

    selected_pair = (int *) calloc (2, sizeof (int));

    /* Given a connect request (I,O), where I is an INPUT SWITCH and O is
    /* an OUTPUT SWITCH, read the row vectors in matrices 1 & 2. Then, XOR
    /* the two vectors to get a list of candidates for (A,B). */

    for (j = 1; j < 17; j++)
    {
        R1_cnt_sw[j] = space_in_mat[in_stage][j];
        C1_cnt_sw[j] = space_cnt_mat[out_stage][j];
    }

    for (j = 0; j < 16; j++)
    {
        x1 = R1_cnt_sw[j];
        x2 = C1_cnt_sw[j];

        if (x1 == 0 && x2 != 0)          /* The XOR operation. */
        {
            listofA[m] = x2;
            m++;
        }

        if (x1 != 0 && x2 == 0)
        {
            listofB[n] = x1;
            n++;
        }
    }
}

```

24 APR 1997

- 54 -

APPENDIX A

```

    }
}

m = 0;
n = 0;
i = 0;
while ((x1 = listofB[i]) != 0)      /* Create all combinations of */
{                                   /* elements of listofA's */
    j = 0;                          /* & listofB's. */
    while ((x2 = listofA[j]) != 0)
    {
        cnt_stage_pairs[m][0] = x1;
        cnt_stage_pairs[m][1] = x2;
        m++;
        j++;
    }
    i++;
}

least_moves = xtra_stage;
for (k = 0; k < m; k++)              /* For each combination of middle */
{                                   /* switches, calculate least num. */
    B_temp = cnt_stage_pairs[k][0]; /* of moves that are required. */
    A_temp = cnt_stage_pairs[k][1];

    if (B_temp != 0 && A_temp != 0)
    {
        num_moves = calc_nom (B_temp, A_temp, k);
    }

    if (num_moves < least_moves)
    {
        least_moves = num_moves;
        least_pair = m;
    }
}

*(selected_pair + 0) = cnt_stage_pairs[least_pair][1];
*(selected_pair + 1) = cnt_stage_pairs[least_pair][0];

return (selected_pair);
}

/*****
/*****
/*
/* The following are the revised subprograms and main program for BICA. */
/*
/*****
/*****

/*****
/*
/* This is the subprogram for BICA's hitless time rearrangement. */
/*
/*****

int          hitless_time_rearrange ()

```

24 APR 1997

- 55 -

APPENDIX A

```
{
    /* Use Include File. */
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's flood time algorithm.
/*
/*.....*/

int          flood_time ()
{
    /* Use Include File. */
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's select two time slots.
/*
/*.....*/

int          select_time_slots ()
{
    /* Use Include File. */
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's hitless space rearrangement.
/*
/*.....*/

int          hitless_space_rearrange ()
{
    /* Use Include File. */
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's flood space algorithm.
/*
/*.....*/

int          flood_space ()
{
    /* Use Include File. */
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's pump link macro.
/*
/*.....*/
/*.....*/
```

24 APR 1997

- 56 -

APPENDIX A

```

/* This macro logically "and's" space_in_link[i] with space_out_link[k]
/* to determine the lowest numbered center stage switch, "c," having
/* both an idle link to input stage "i" and an idle link to output
/* stage "k." The first non-zero entry in the result of the logical
/* "and" is "c" and the two links to "c" are reset to "0" (marked as
/* busy) in space_in_link[i] and space_out_link[k]. The macro returns
/* the value "c."
/*.....

```

```

int      pump_link ()
{
    int      c,
             j,
             x;

    for (j = 1; j <= xtra_stage; j++)
    {
        x = space_in_link[in_stage][j] && space_out_link[out_stage][j];

        if (x == 1)
        {
            c = j;
            break;
        }
    }

    clear_link(space_in_link, in_stage, c);
    clear_link(space_out_link, out_stage, c);

    if (c == xtra_stage)
    {
        rearrange_space = 1;
    }

    /* Return center stage. */
    return(c);
}

```

```

/*.....
/*
/* This is the subprogram for BICA's clear idle link bit (to 0).
/*.....

```

```

/*.....
/* This primitive is used to clear (reset) the kth bit of the jth row
/* of the bit map specified by io to "0." If the bit map contains n
/* elements and the value of j > n, then an error flag is returned.
/* Otherwise, the kth element is reset to "0."
/*.....

```

```

int      clear_link(io, j, k)
int      io[16][17],
             j,
             k;
{
    int      error_flag;

    if(k <= xtra_stage)
    {

```

24 APR 1997

- 57 -

APPENDIX A

```

        io[j][k] = 0;
        error_flag = 0;
    }
    else
    {
        error_flag = 1;
    }
    return(error_flag);
}

```

```

/*****
/*
/* This is the subprogram for BICA's set idle link bit (to 1).
/*
*****/

```

```

/*****
/* This primitive is used to set the kth bit of the jth row of the bit
/* map specified by io to "1." If the bit map contains n elements and
/* the value of j > n, then an error flag is returned. Otherwise, the
/* kth element is set to "1."
*****/

```

```

int          set_link(io, j, k)
int          io[16][17],
            j,
            k;

```

```

{
    int          error_flag;

    if(k <= xtra_stage)
    {
        io[j][k] = 1;
        error_flag = 0;
    }
    else
    {
        error_flag = 1;
    }
    return(error_flag);
}

```

```

/*****
/*
/* This is the subprogram for BICA's select two center stage switches.
/*
*****/

```

```

int          select_cnt_stage ()
{
    int *center_stage_pair;

    center_stage_pair = (int *)calloc(2, sizeof(int));

    /* select middle stage switch pair */
    center_stage_pair = implement_PM2 ();

    return(*center_stage_pair);
}

```

24 APR 1997

- 58 -

APPENDIX A

```

.....
/*
/* This is the subprogram for BICA's time rearrangement.
/*
.....

/* Node D */
void          time_rearrange ()
{
    int          XtoA = 0,
                 XtoB = 0;
    char          last;

    /* Select Two Time Slots for Rearrangement. */
    select_time_slots ();

    /* Find Connections to be Rearranged. */
    flood_time ();

    /* Q6: Which Path Requires Least Number of Rearrangements? */
    if (last == 'A')
    {
        XtoA = 1;
    }
    else
    {
        if (last == 'B')
        {
            XtoB = 1;
        }
    }
    hitless_time_rearrange ();
    rearrange_time = 0;
    return;
}

.....
/*
/* This is the subprogram for BICA's space rearrangement.
/*
.....

/* Node C */
void          space_rearrange ()
{
    int          XtoA = 0,
                 XtoB = 0;
    int          *cnt_stage_pair;
    char          last;

    /* allocate memory for pointer to list of connections to be moved. */
    cnt_stage_pair = (int *)calloc(2, sizeof(int));

    /* Select Two Center Stage Switches for Rearrangement. */
    *cnt_stage_pair = select_cnt_stage ();

    /* Find Connections to be Rearranged. */
    flood_space ();

```

24 APR 1997

- 59 -

APPENDIX A

```

/* Q5: Which Path Requires Least Number of Rearrangements? */
if (last == 'A')
{
    XtoA = 1;
}
else
{
    if (last == 'B')
    {
        XtoB = 1;
    }
}
hitless_space_rearrange ();
rearrange_space = 0;
return;
}

/*.....*/
/*
/* This is the subprogram for BICA's remainder function.
/*.....*/
/*.....*/
/* This function produces the remainder after the divisor has been
* subtracted from the dividend as many (integer) times as possible.
/*.....*/

int          rem(dividend, divisor)
int          dividend,
            divisor;
{
    int          remainder;

    if (dividend <= 0)
    {
        remainder = 0;
    }
    else
        remainder = dividend;
    while (remainder >= divisor)
    {
        remainder = remainder - divisor;
    }
    return(remainder);
}

/*.....*/
/*
/* This is the subprogram for BICA's quotient function.
/*.....*/
/*.....*/
/* This function produces the quotient by subtracting the divisor from
/* the dividend as many (integer) times as possible.
/*.....*/

int          quot(dividend, divisor)
int          dividend,

```

24 APR 1997

- 60 -

APPENDIX A

```

divisor;

int      quotient,
         remainder;

quotient = 0;
if (dividend <= 0)
{
    remainder = 0;
}
else
    remainder = dividend;
while (remainder >= divisor)
{
    remainder = remainder - divisor;
    quotient++;
}
return(quotient);
}

/*.....*/
/* This is the subprogram for BICA's convert I/O line numbers. */
/*.....*/

/*.....*/
/* This subprogram uses the quotient and remainder functions to convert */
/* the I/O line numbers for a connect or disconnect to a pair of I/O */
/* stage switch numbers and I/O stage switch pin number. */
/*.....*/

void      convert_line ()
{
    int      r = 16,
             in_line_minus_1,
             out_line_minus_1;

    in_line_minus_1 = in_line - 1;
    in_stage = quot(in_line_minus_1, r) + 1;
    in_num = rem(in_line_minus_1, r) + 1;

    out_line_minus_1 = out_line - 1;
    out_stage = quot(out_line_minus_1, r) + 1;
    out_num = rem(out_line_minus_1, r) + 1;

    return;
}

/*.....*/
/* This is the subprogram for BICA's print statistics - n and m. */
/*.....*/

int      print_stats ()
{
    /* Test case for print_stats. */
    printf ("BICA performed %d connect(s) and %d disconnect(s).\n", n, m);
}

```

24 APR 1997

- 61 -

APPENDIX A

```

return;
}

/*.....*/
/*
/* This is the subprogram for BICA's disconnect - input from output. */
/*.....*/

/* Node B */
void disconnect ()
{
    int cnt_stage, /* csf for Center Stage switch. */
        csf;

/* Convert IL-m -> OL-n Disconnect Request to I(i,j) >< O(k,l) Format. */
    convert_line ();
    convert_line ();

/* Remove Connections as Requested. */
    csf = space_in_mat[in_stage][in_num];
    space_in_mat[in_stage][in_num] = 0;
    space_cnt_mat[csf][in_stage] = 0;
    space_out_mat[out_stage][out_num] = 0;

/* Update Tables. */
    set_link(space_in_link, in_stage, csf);
    set_link(space_out_link, out_stage, csf);

/* Q7: Pack Connections After A Disconnect Enabled? */
    if (rearrange_space == 1)
        /* A3 = YES -> Node C. */
        {
            space_rearrange ();
        }
    /* A3 = NO. */
    else
        /* End Space Domain. */
        {
            /* Q4: Time Rearrangement Required? */
            if (rearrange_time == 1)
                /* A4 = YES -> Node D. */
                {
                    time_rearrange ();
                }
            /* A4 = NO. */
            /* End Time Domain. */
        }
    return;
}

/*.....*/
/*
/* This is the subprogram for BICA's connect - input to output. */
/*.....*/

/*.....*/
/* This routine connects an input line to an output line through the
/* space matrix. Variables are defined as follows: */

```

24 APR 1997

- 62 -

APPENDIX A

```

/*
/*   in_line:   an input to the space matrix,
/*               0 <- in_line <- 255
/*   out_line:  an output from the space matrix,
/*               0 <- out_line <- 255
/*
/*   in_stage:  an input stage of the space matrix,
/*               0 <- in_stage <- 15
/*   in_num:    an input to an input stage of the space matrix,
/*               0 <- in_number <- 15
/*
/*   out_stage: an output stage from the space matrix,
/*               0 <- out_stage <- 15
/*   out_num:   an output from an output stage of the space matrix,
/*               0 <- out_number <- 15
/*
/*   cnt_stage: a center stage of the space matrix,
/*               0, 1 <- center_stage <- 17
/*   NOTE:      center_stage = 0, means that no center stage of the
/*               space matrix has been assigned.
/*               center_stage = 1, ..., 16 means that the corresponding
/*               center stage of the space matrix is assigned.
/*               center_stage = 17 is the extra stage of the space
/*               matrix, and rearrangement is in progress.
/* .....
/* Node A */
void      connect ()
{
    int          csw;                /* csw for Center Stage switch. */

/* Convert IL-m -> OL-n Connection Request to I(i,j) -> O(k,l) Format. */
    convert_line ();

/* Begin Time Domain.
/* Select Lowest Numbered Time Slot Available for Connection.
/* Begin Space Domain.
/* Select Lowest Numbered Center Stage Available for Connection.
    csw = pump_link ();

/* Make Connection and Update Tables.
    space_in_mat[in_stage][in_num] = csw;
    space_cnt_mat[csw][in_stage] = out_stage;
    space_out_mat[out_stage][out_num] = csw;

/* Q3: Space Rearrangement Required? */
    if (rearrange_space == 1)
        /* A3 = YES -> Node C. */
        {
            space_rearrange ();
        }
/* A3 = NO.
    else
        /* End Space Domain.
        {
            /* Q4: Time Rearrangement Required? */

```

24 APR 1997

- 63 -

APPENDIX A

```

if (rearrange_time == 1)
/* A4 = YES -> Node D. */
{
    time_rearrange ();
}
/* A4 = NO. */
/* End Time Domain. */
}
return;
}

/*-----*/
/* This is the subprogram for BICA's get command. */
/*-----*/

int          get_command ()
{
    int          command;

/* Test case for get_command. */
if (command_count < Command_end)
{
    /* command = 1 -> connect. */
    command = 1;
    command_count++;
    in_line = command_count;
    out_line = command_count;
}
else if (command_count == command_end)
{
    /* command = -1 -> disconnect. */
    command = -1;
    command_count++;
}
else
{
    /* command = 0 -> end. */
    command = 0;
}
return(command);
}

/*-----*/
/* This is the subprogram for BICA's space rearrange initialization. */
/*-----*/

void          initialize_rearrange ()
{
    int          i;

    for (i = 0; i < 256; i++)
    {
        rearrange_3d.position[i].x = 0;
        rearrange_3d.position[i].y = 0;
        rearrange_3d.position[i].z = 0;
    }
}

```

24 APR 1997

- 64 -

APPENDIX A

```

    for (i = 0; i < 15; i++)
    {
        rearrange_moves.move[i].input_sw = 0;
        rearrange_moves.move[i].old_mid_sw = 0;
        rearrange_moves.move[i].new_mid_sw = 0;
        rearrange_moves.move[i].output_sw = 0;
    }
    return;
}

/*.....*/
/* This is the subprogram for BICA's space matrix initialization. */
/*.....*/

/*.....*/
/* This routine is used initially to clear (reset) the space matrices. */
/*.....*/

void                initialize_space_mat()
{
    int i, j;

    for(i = 0; i < xtra_stage; i++)
    {
        for(j = 0; j < xtra_stage; j++)
        {
            space_in_mat[i][j] = 0;
            space_cnt_mat[i][j] = 0;
            space_out_mat[i][j] = 0;
        }
        space_cnt_mat[xtra_stage][i] = 0;
    }
    return;
}

/*.....*/
/* This is the subprogram for BICA's initialize idle link bit maps. */
/*.....*/

/*.....*/
/* The initialize link bit maps primitive (used only for start-up */
/* operations) initializes both the space input idle link bit map and */
/* the space output link bit map. All entries in the idle link bit */
/* maps are initialized to "1," indicating that all links are idle. */
/*.....*/

void                init_links()
{
    int i, j;

    for(i = 0; i < xtra_stage; i++)
    {
        for(j = 0; j < xtra_stage; j++)
        {
            space_in_link[i][j] = 1;

```

24 APR 1997

- 65 -

APPENDIX A

```

        space_out_link' '[j] = 1;
    }
    return;
}

/* ..... */
/* This is the subprogram for BICA's initialization. */
/* ..... */

void          initialize ()
{
    /* Test data for commands. */
    n = 0;
    m = 0;

    command_count = 0;
    rearrange_space = 0;
    rearrange_time = 0;

    /* Initialize all matrices. */
    init_links();
    initialize_space_mat ();
    initialize_rearrange ();

    return;
}

/* ..... */
/* This is the main program for Bowdon's Immediate Connect Algorithm. */
/* ..... */

main ()
{
    int          command;

    /* Start Program Flow Here. */
    initialize ();
    do
    {
        command = get_command ();

        /* Q1: Pair of Ports Presented for Connection? */
        /* command = 1 -> connect. */
        if (command == 1)
            /* A1 = YES -> Node A. */
            {
                connect ();
                n = n++;
            }
        /* A1 = NO. */
        else
            /* Q2: Pair of Ports Presented for Disconnect? */
            {
                /* command = -1 -> disconnect. */

```

24 APR 1997

- 66 -

APPENDIX A

```
if (command == 1)
    /* A2 = YES -> Node B. */
    {
        disconnect ();
        m = m++;
    }
    /* A2 = NO. */
    continue;
}
/* command = 0 -> end. */
while (command != 0);
{
    print_stats ();
}
```

24 APR 1997

- 67 -

WHAT IS CLAIMED IS:

1. A method for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising the steps of:

representing the communications matrix by an array having a first dimension and a second dimension, said first dimension representing a plurality of input stage switches of the communications matrix, said second dimension representing a plurality of output stage switches of the communications matrix, said array having a plurality of cells, each of said cells representing an intersection of said first dimension with said second dimension and representing a possible center stage switch connection through the communications matrix;

identifying a blocked center stage switch of the communications matrix in terms of said first dimension and said second dimension;

identifying a pair of center stage switches comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining, in parallel, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by first using said first center stage switch and a second rearrangement sequence through the communications matrix from said input switch to said output switch by first using said second center stage switch; and

terminating said determining step upon first determining the shorter of said first rearrangement sequence and said second rearrangement sequence.



* C L M S *

2. A system for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising:

an array for representing the communications matrix, said array having a first dimension and a second dimension, said first dimension representing a plurality of input stage switches of the communications matrix, said second dimension representing a plurality of output stage switches of the communications matrix, said having a plurality of cells, each of said cells representing an intersection of said first dimension with said second dimension and representing a possible center stage switch connection through the communications matrix;

means for identifying a blocked center stage switch of the communications matrix in terms of said first dimension and said second dimension;

a pair of center stage switches comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining means for determining, in parallel, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by first using said first center stage switch and a second rearrangement sequence through the communications matrix from said input switch to said output switch by first using said second center stage switch; and

terminating means for terminating operation of said determining means upon first determining the shorter of said first rearrangement sequence and said second rearrangement sequence.



3. A method for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising the steps of:

representing the communications matrix by a square matrix having a plurality of rows and a plurality of columns, said rows representing a plurality of input stage switches of the communications matrix, said columns representing a plurality of output stage switches of the communications matrix, said having a plurality of cells, each of said cells representing an intersection of said rows with said columns and representing a possible center stage switch connection through the communications matrix;

identifying a blocked center stage switch of the communications matrix in terms of said rows and said columns;

identifying in terms of said rows and said columns a pair of center stage switches comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining, in parallel, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by plurality of steps through said square matrix to first use said first center stage switch and then a second rearrangement sequence through the communications matrix from said input switch to said output switch to first use said second center stage switch; and

terminating said determining step upon first determining the shorter of said first rearrangement



REPLACEMENT SHEET

24 APR 1997

- 70 -

sequence and said second rearrangement sequence as determined by the smaller number of steps through said square matrix.



* C L M S *

24 APR 1997

- 71 -

4. A system for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising:

a square matrix for representing the communications matrix, said array having a plurality of rows and a plurality of columns, said row representing a plurality of input stage switches of the communications matrix, said columns representing a plurality of output stage switches of the communications matrix, said having a plurality of cells, each of said cells representing an intersection of said row with said columns and representing a possible center stage switch connection through the communications matrix;

means for identifying a blocked center stage switch by establishing a connection through a dedicated center stage switch of the communications matrix in terms of said rows and said columns;

a pair of center stage switches identified by said rows and said columns and comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining means for determining, in parallel, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by a plurality of steps through said square matrix first to use said first center stage switch and then use a second rearrangement sequence through the communications matrix from said input switch to said output switch that first uses said second center stage switch; and



* C L M S *

24 APR 1997

- 72 -

terminating means for terminating operation of said
determining means upon first determining the shorter of
said first rearrangement sequence and said second
rearrangement sequence as determined by the smaller
number of steps through said square matrix.



24 APR 1997

- 73 -

5. A method for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising the steps of:

representing the communications matrix by an array having a first dimension and a second dimension, said first dimension representing a plurality of input stage switches of the communications matrix, said second dimension representing a plurality of output stage switches of the communications matrix, said array having a plurality of cells, each of said cells representing an intersection of said first dimension with said second dimension and representing a possible center stage switch connection through the communications matrix;

identifying a rearrangeably blocked center stage switch of the communications matrix in terms of said first dimension and said second dimension;

identifying a plurality of center stage switches comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining, in parallel for each of said pairs, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by first using said first center stage switch and a second rearrangement sequence through the communications matrix from said input switch to said output switch by first using said second center stage switch; and

terminating said determining step upon first determining the shorter of said first rearrangement sequence and said second rearrangement sequence.



* C L H S *

24 APR 1997

- 74 -

6. A system for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix, comprising:

an array for representing the communications matrix, said array having a first dimension and a second dimension, said first dimension representing a plurality of input stage switches of the communications matrix, said second dimension representing a plurality of output stage switches of the communications matrix, said array having a plurality of cells, each of said cells representing an intersection of said first dimension with said second dimension and representing a possible center stage switch connection through the communications matrix;

means for identifying a rearrangeably blocked center stage switch by indicating a connection through a dedicated center stage switch of the communications matrix in terms of said first dimension and said second dimension;

a plurality of pairs of center stage switches comprising a first center stage switch and a second center stage switch, said first center stage switch not being associated with the same output stage switch as said blocked center stage switch, said second center stage switch not being associated with the same input stage switch as said blocked center stage switch;

determining means for determining, in parallel for each of said pairs, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix from said input switch to said output switch by first using said first center stage switch and a second rearrangement sequence through the communications matrix from said input switch to said output switch by first using said second center stage switch; and



* C L M S *

24 APR 1997

- 75 -

terminating means for terminating operation of said
determining means upon first determining the shorter of
said first rearrangement sequence and said second
rearrangement sequence.

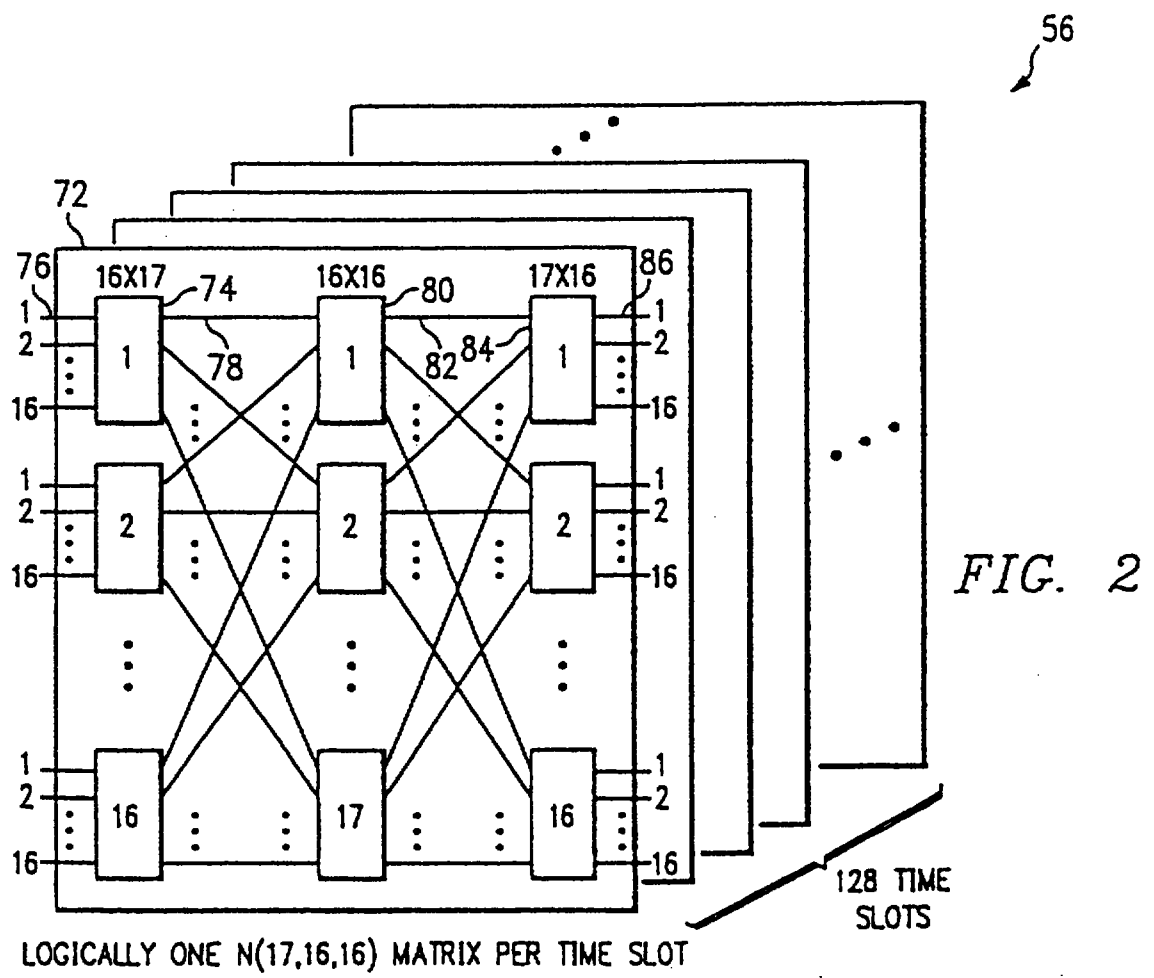
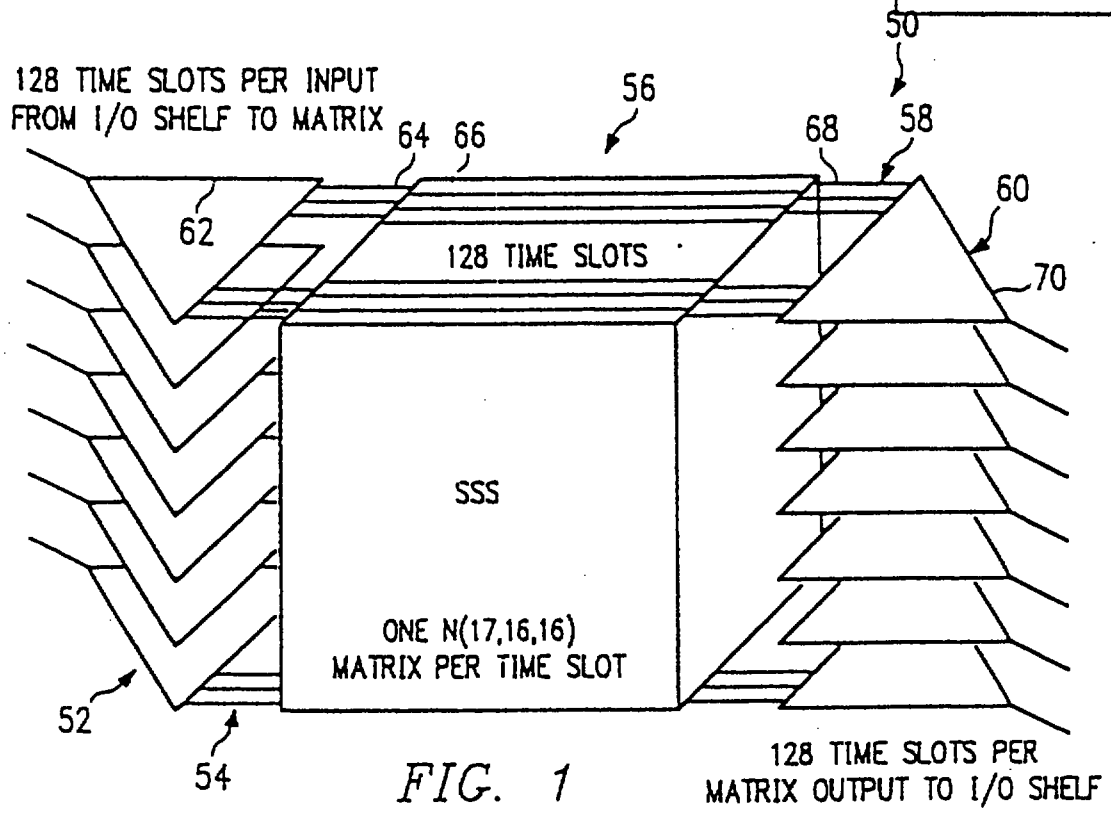


* C L M S *

METHOD AND SYSTEM FOR SELECTING AN
OPTIMAL REARRANGEMENT SEQUENCE FOR
A CROSS-CONNECT COMMUNICATIONS MATRIX

ABSTRACT OF THE DISCLOSURE

A method and system for determining the minimum number of rearrangements for connecting an input gate to an output gate through a rearrangeably blocked communications matrix represent the communications matrix by a square matrix having a first dimension that represents the input stage switches of the communication matrix, and a second dimension that represents the output stage switches of the communications matrix, and wherein the square matrix has cells each representing an intersection of the first dimension with the second dimension and indicating a possible center stage switch connection through of the communications matrix. The method and system identify a blocked center stage switch of the communications matrix in terms of the first dimension and the second dimension and identify a pair of center stage switches, where one of which does not have the same output stage switch as the blocked center stage switch, and the other of which does not have the same input stage switch as the blocked center stage switch. In parallel, the number of rearrangement steps necessary for a rearrangement sequence through the communications matrix that uses the first center stage switch and, then, the number of rearrangement steps for a rearrangement sequence that uses the other center stage switch are determined. The method and system terminate these number determining functions upon first determining the rearrangement sequence requiring the smaller number of rearrangement steps.



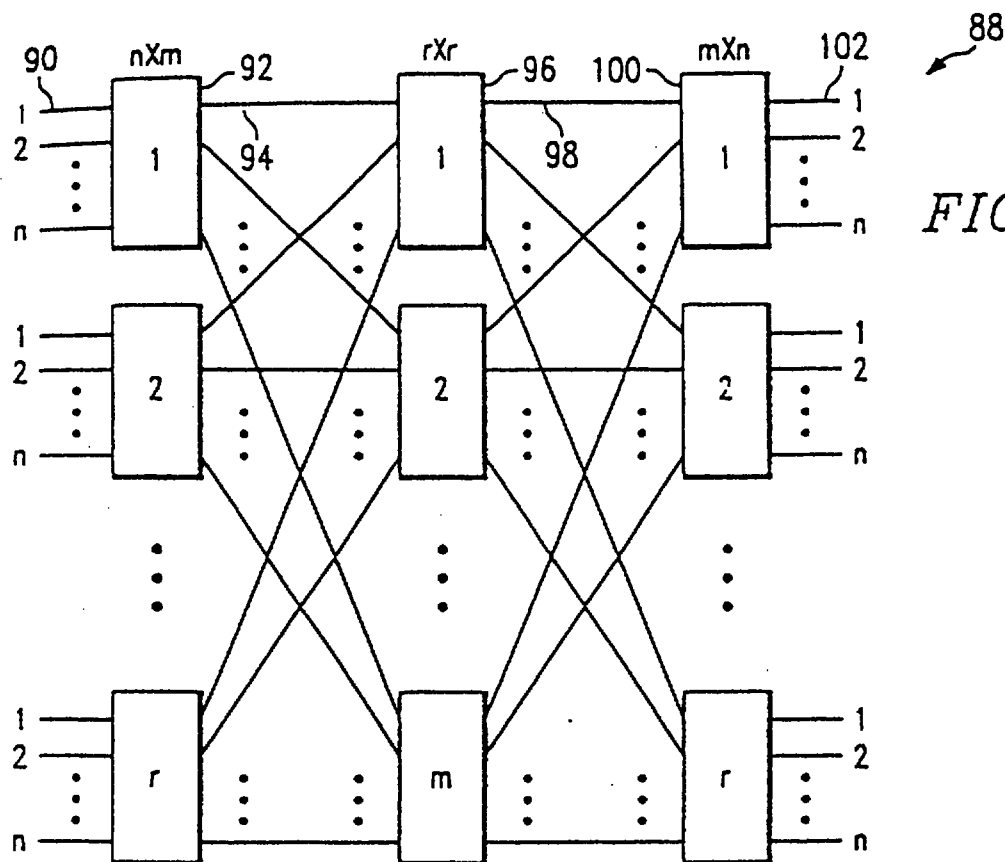


FIG. 3

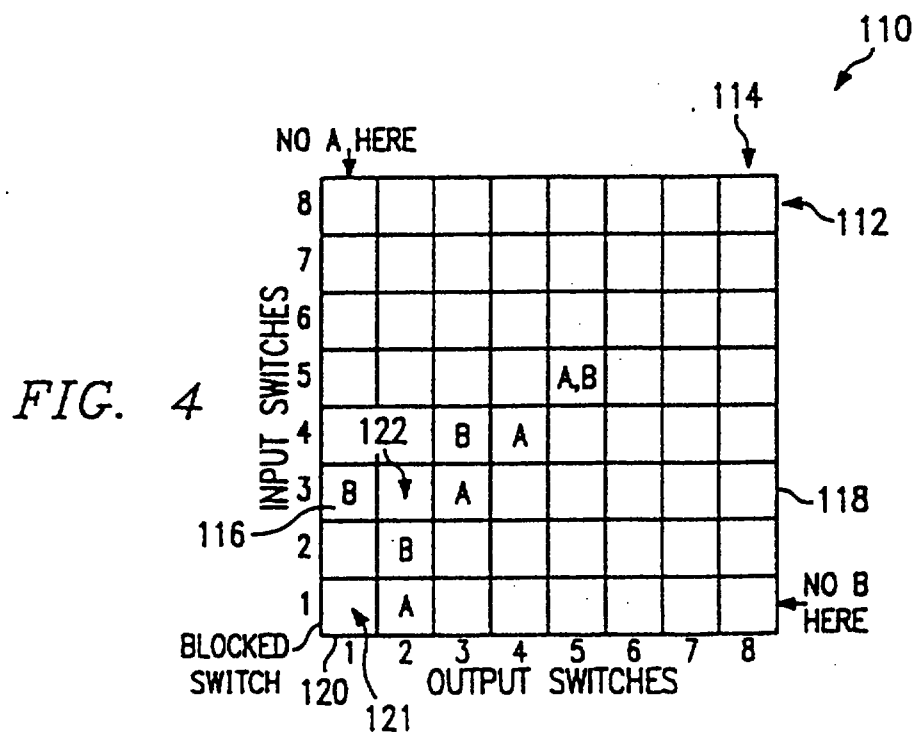


FIG. 4

24 APR 1997

- 3 -

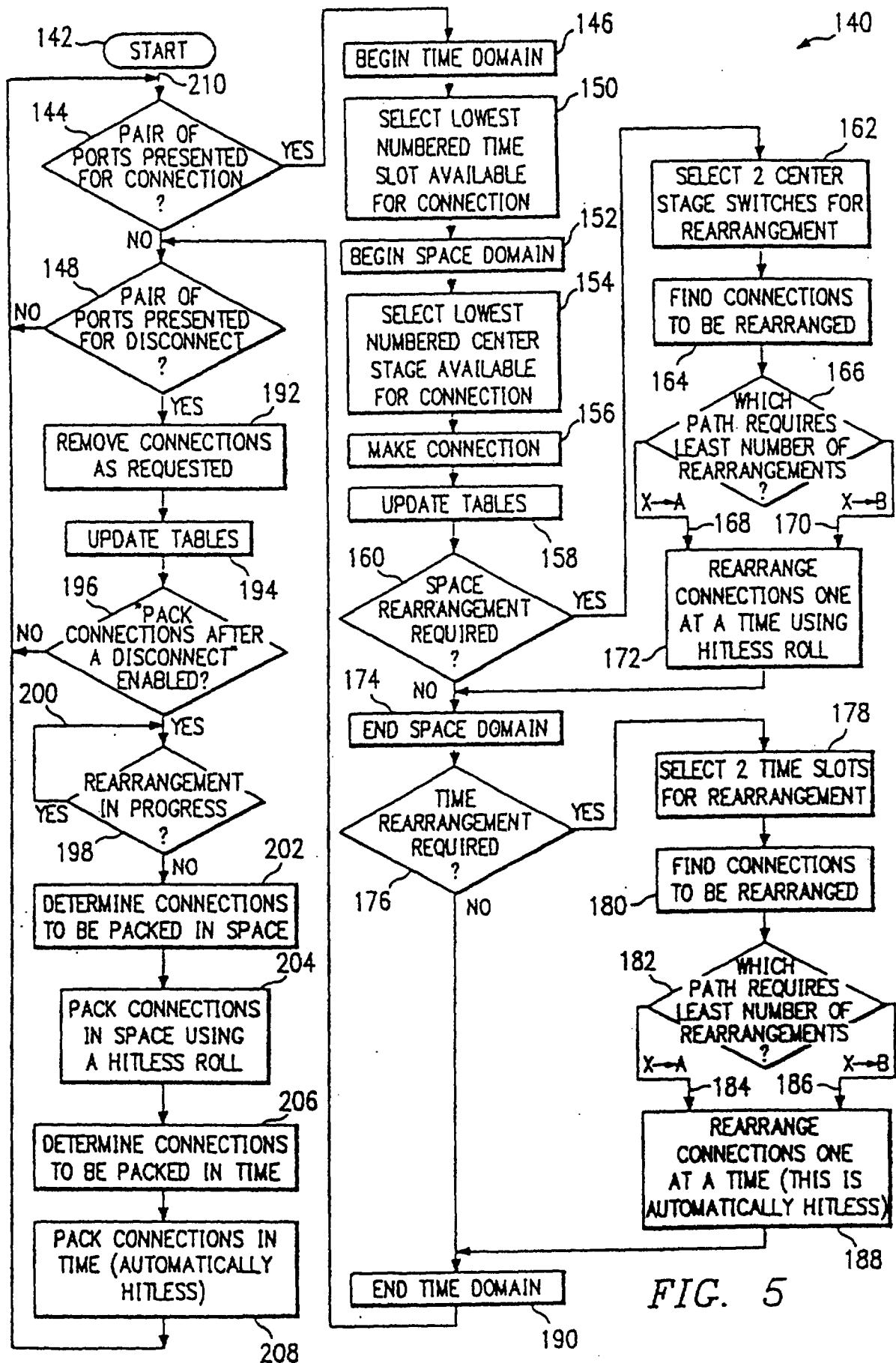


FIG. 5

FIG. 6

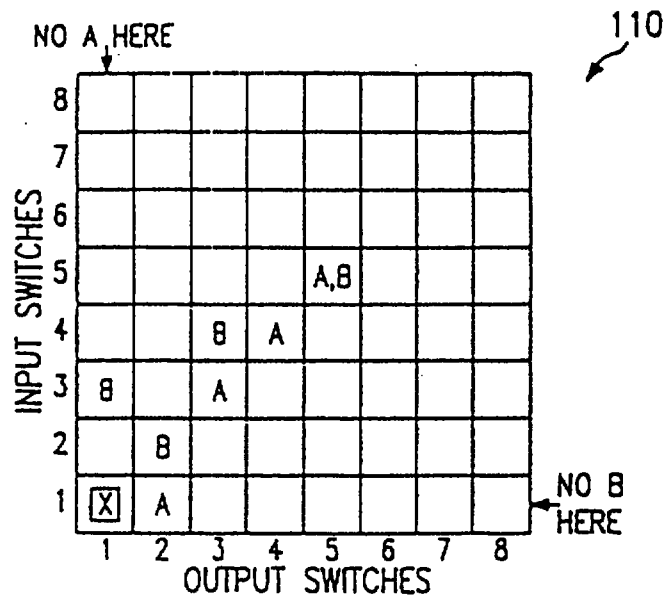


FIG. 7

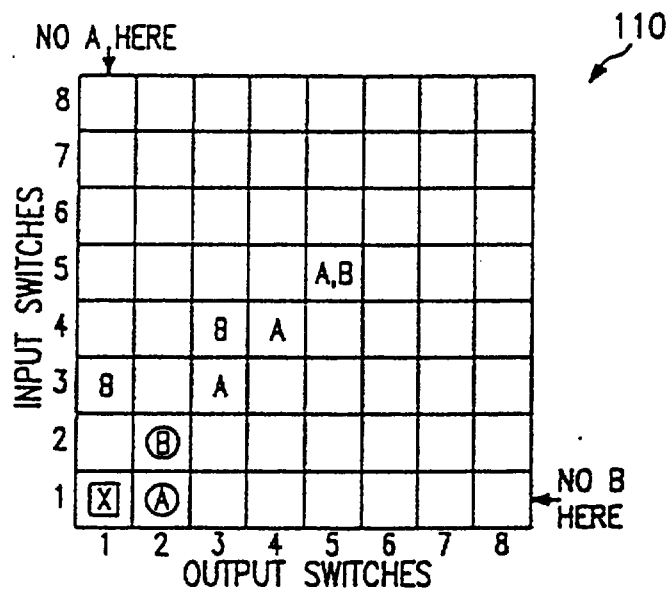
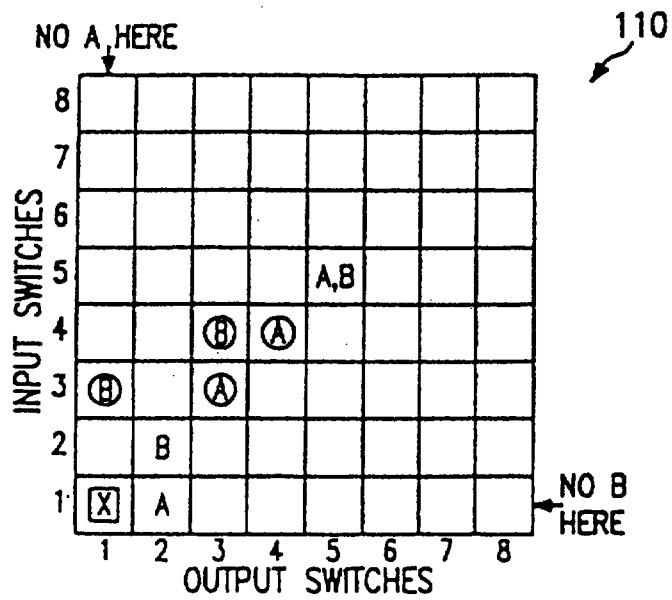


FIG. 8



24 APR 1997

- 5 -

FIG. 9

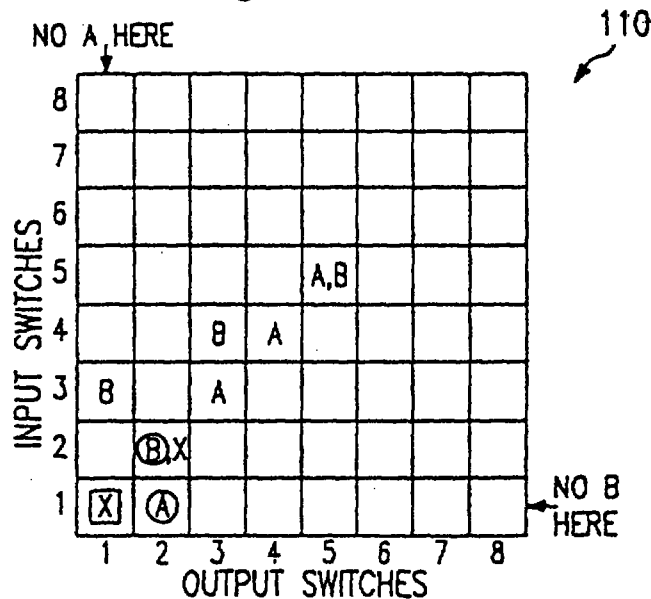


FIG. 10

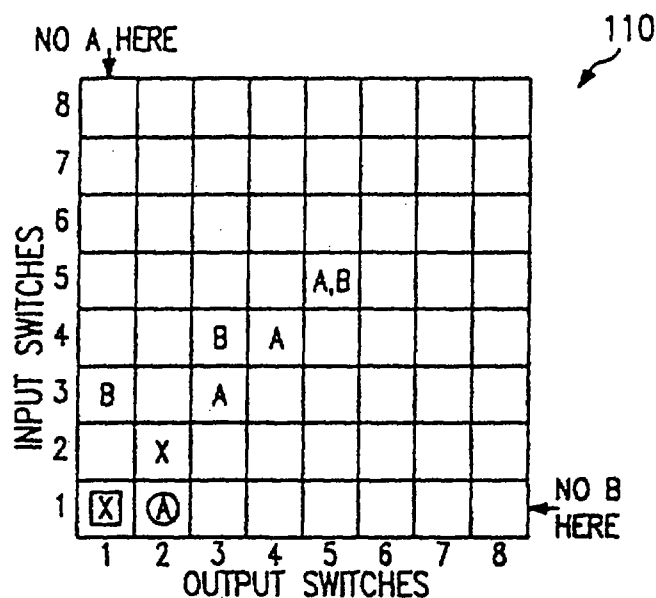
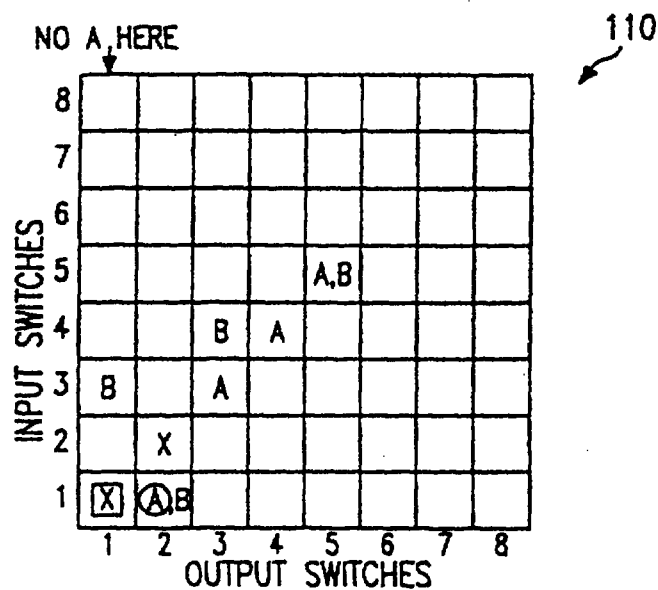


FIG. 11



24 APR 1997

- 6 -

FIG. 12

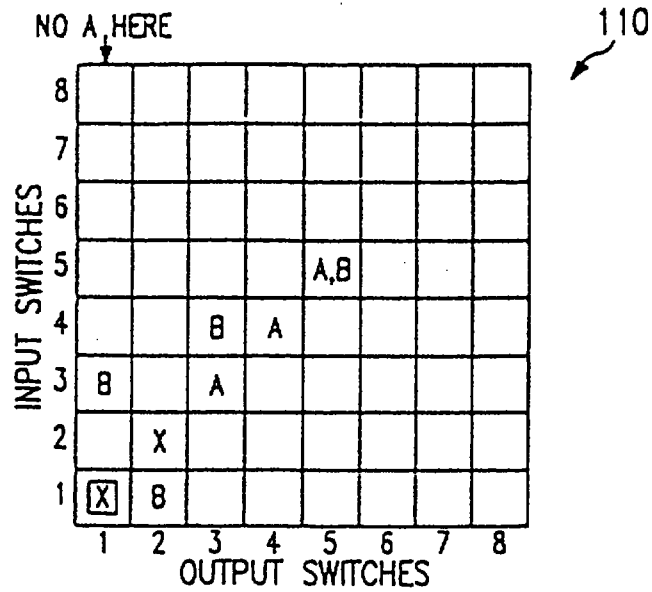


FIG. 13

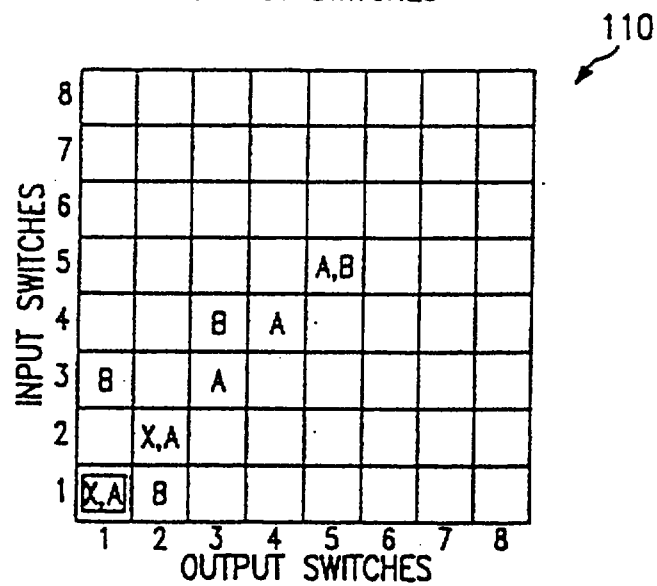
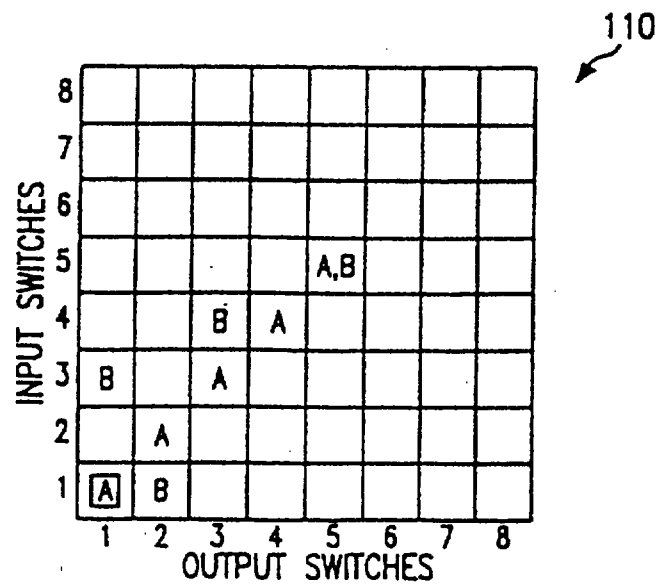


FIG. 14



220

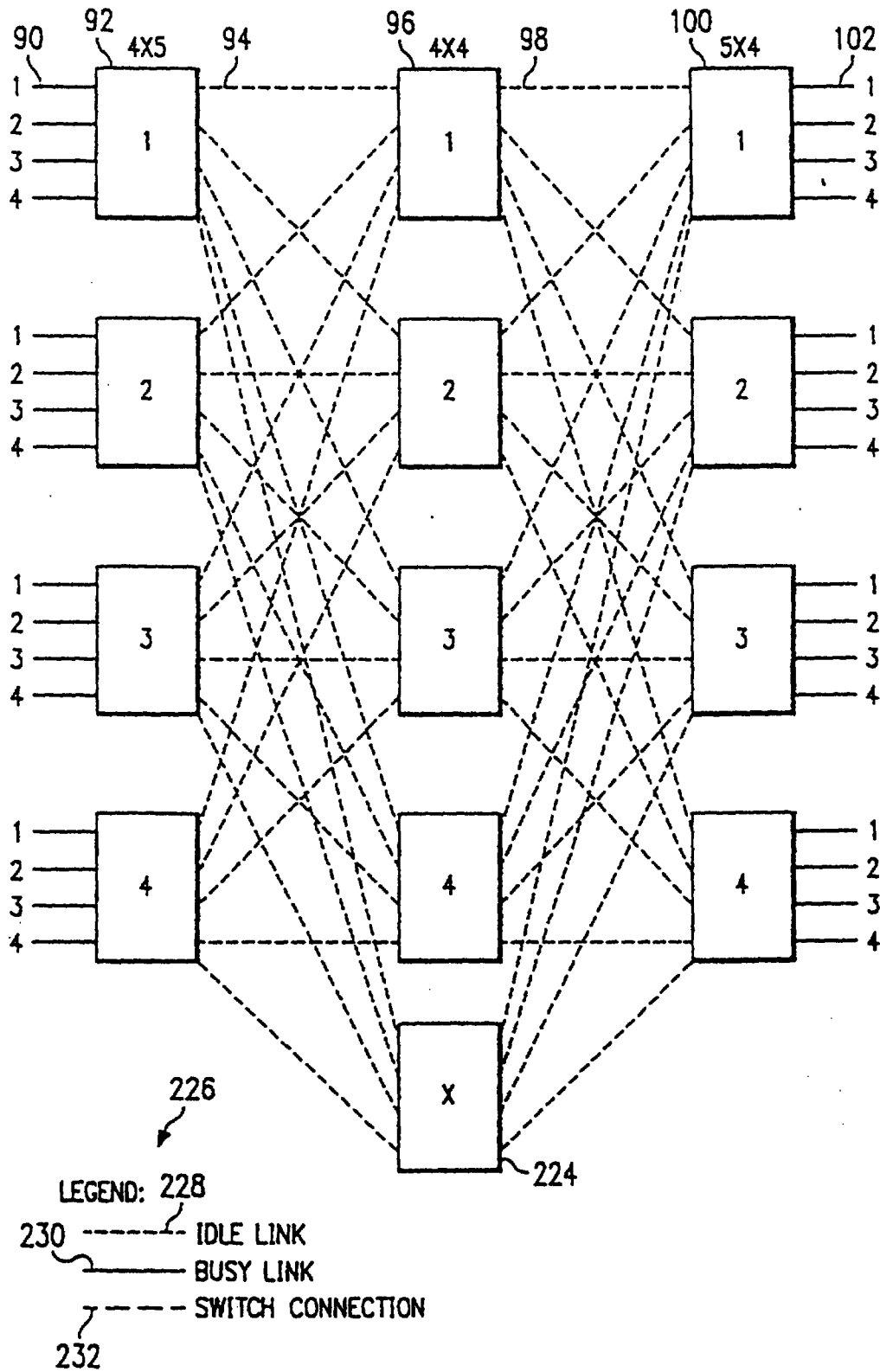
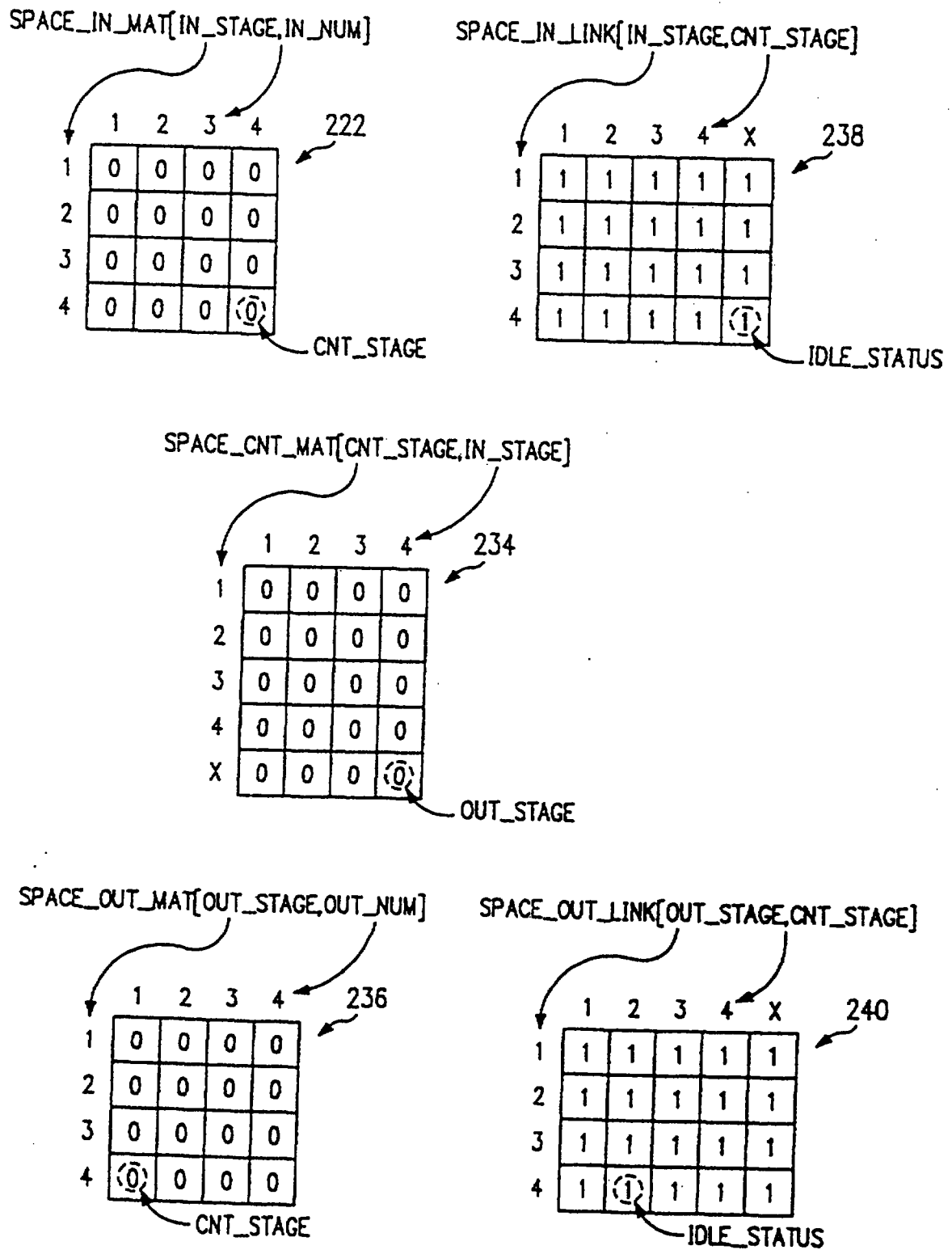


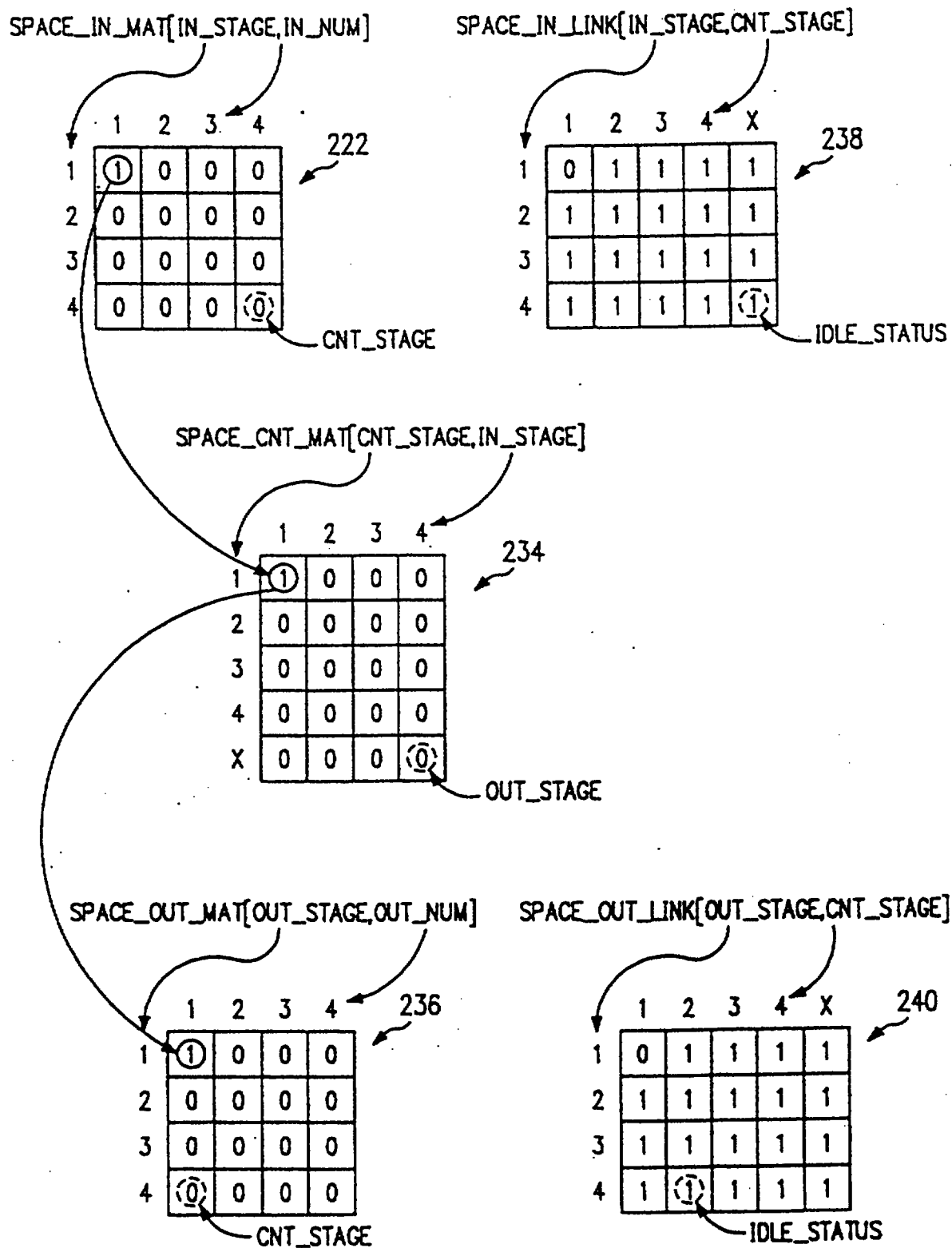
FIG. 16



24 APR 1997

- 9 -

FIG. 17



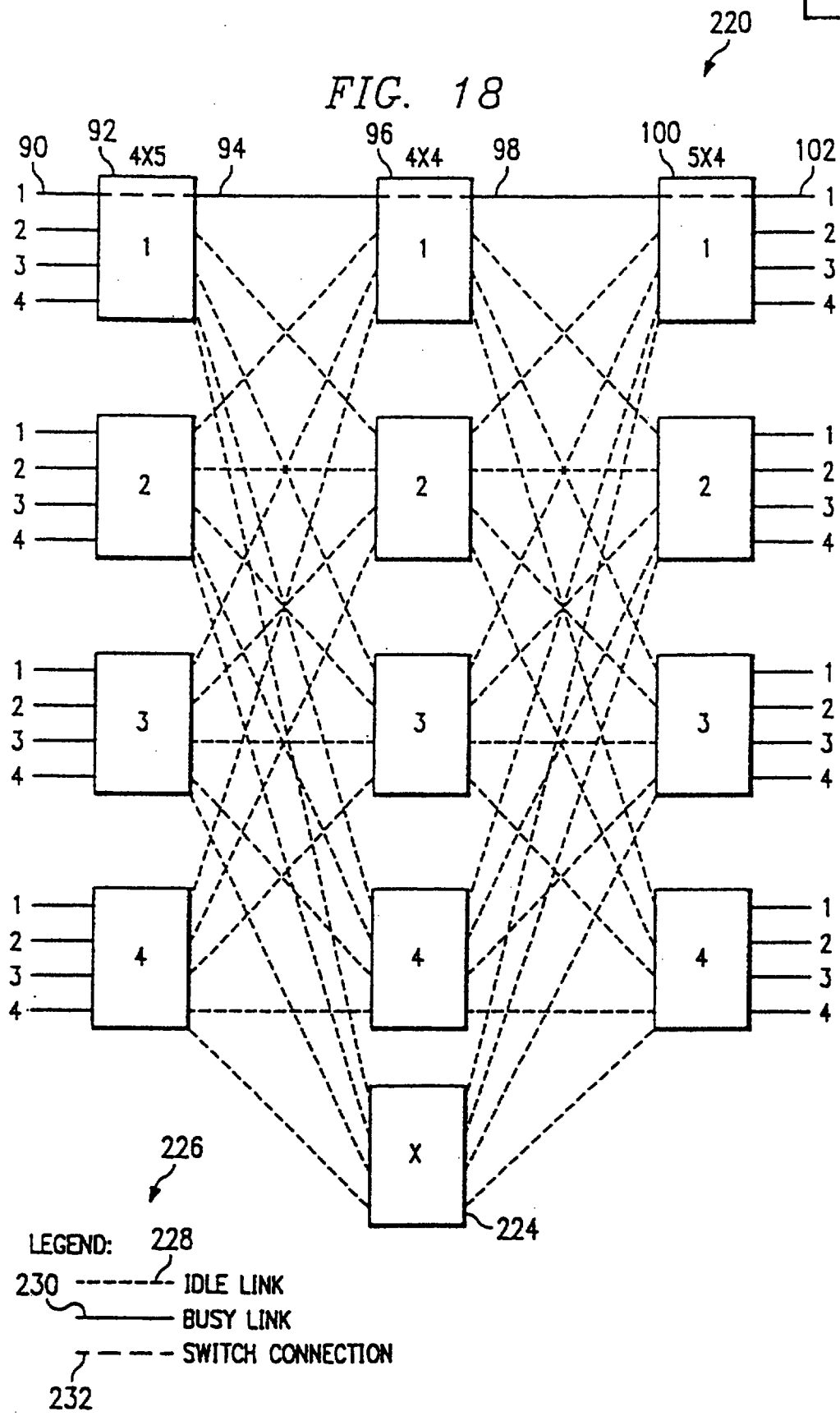


FIG. 19

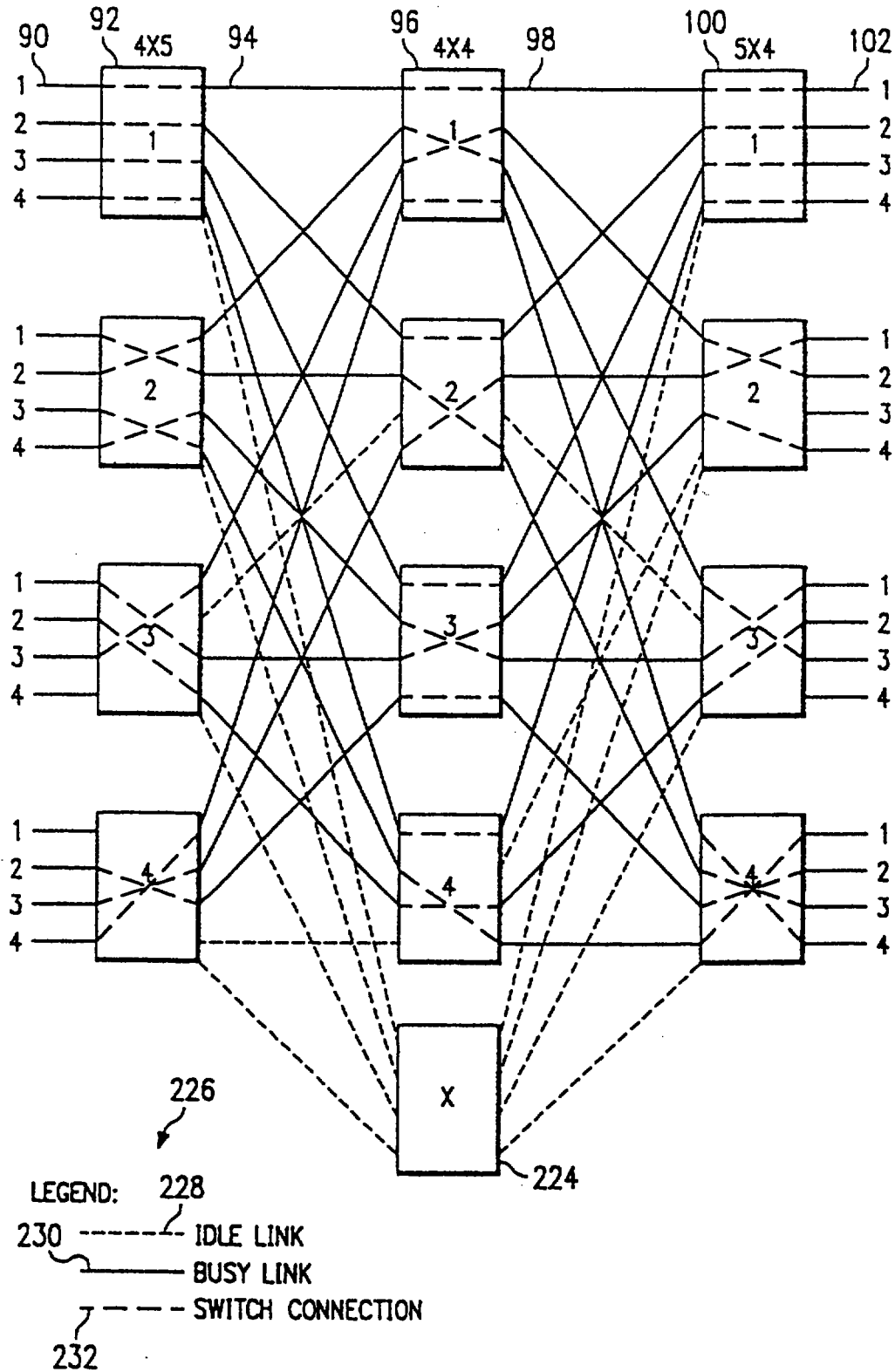


FIG. 20

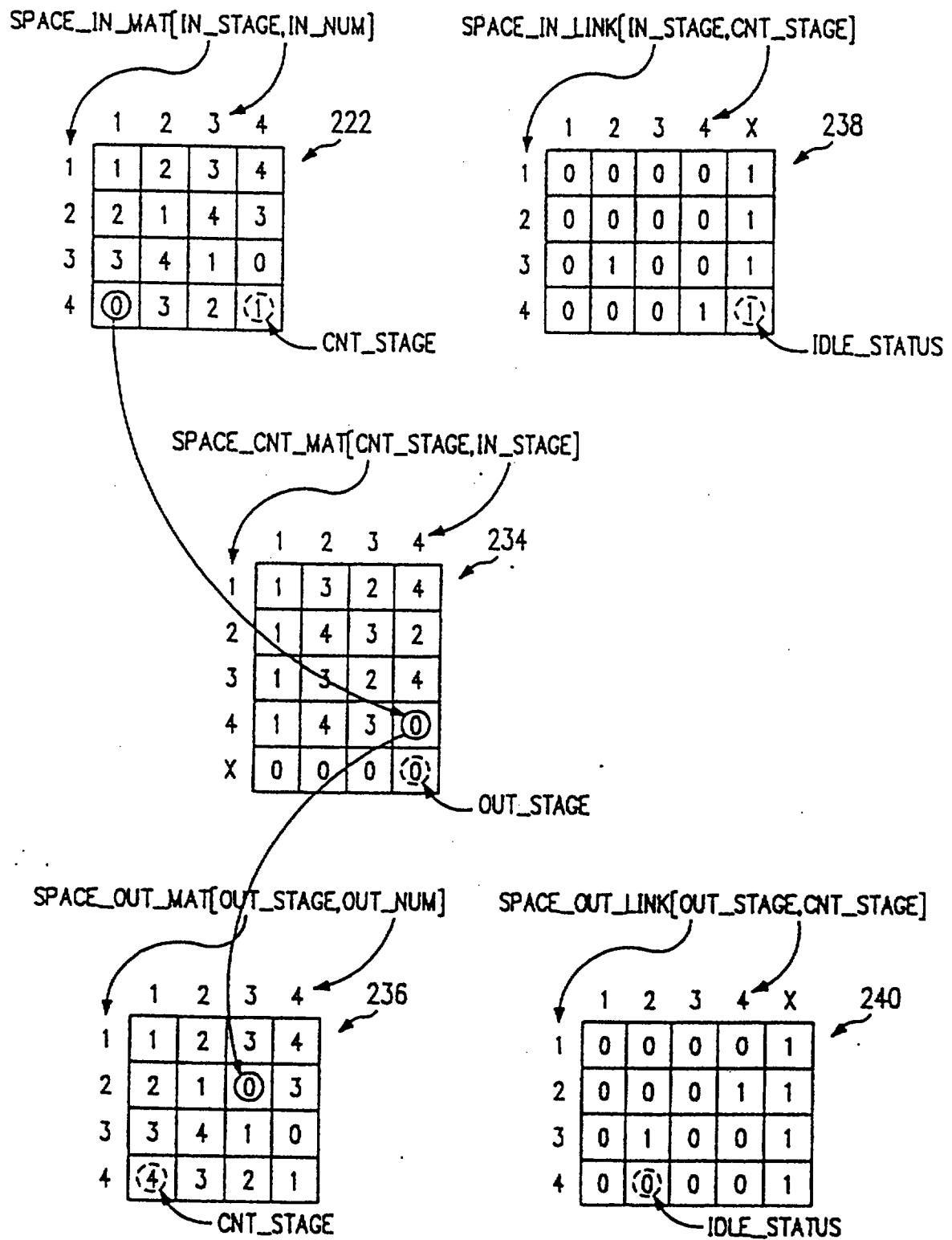
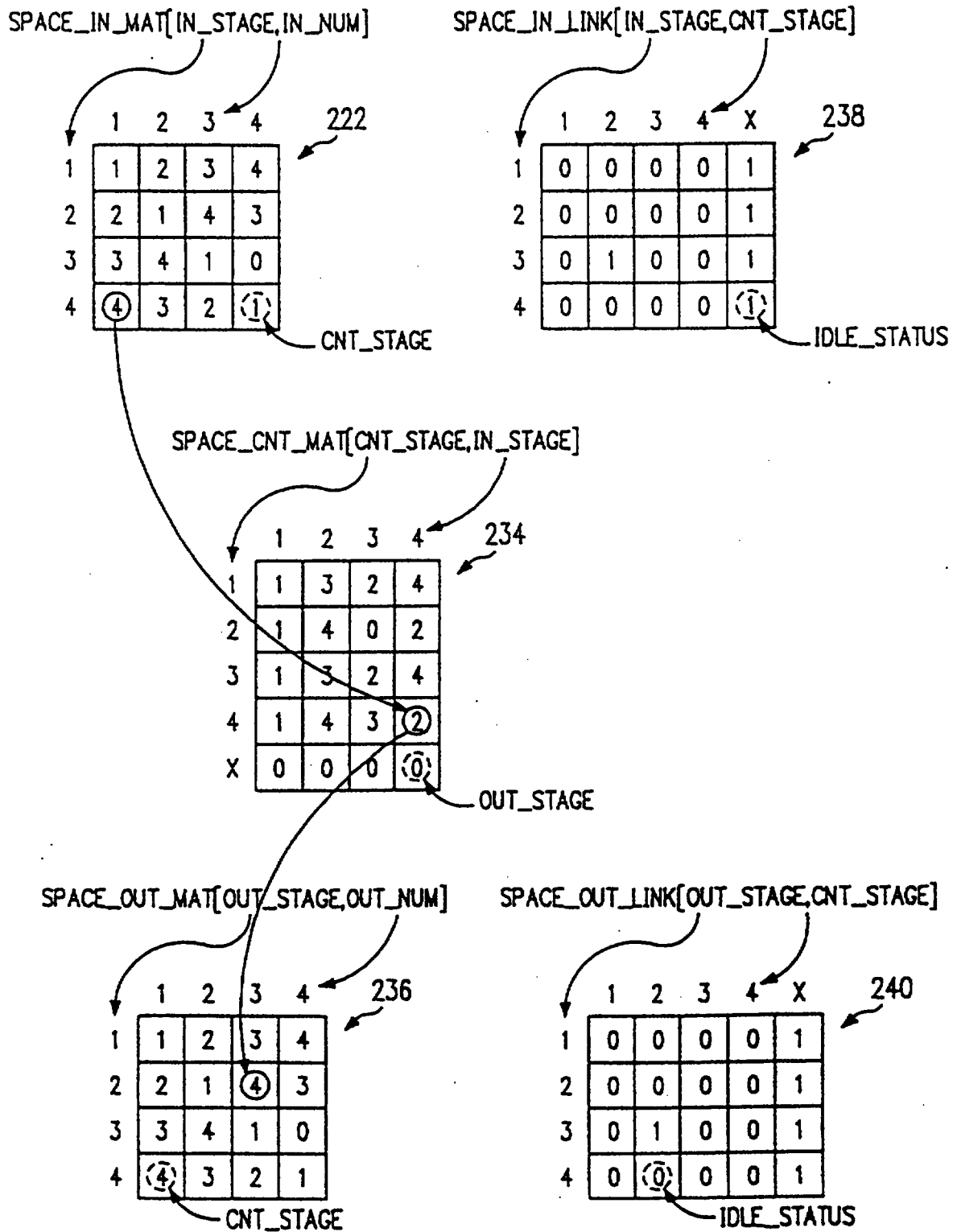


FIG. 21



24 APR 1997

- 14 -

FIG. 22

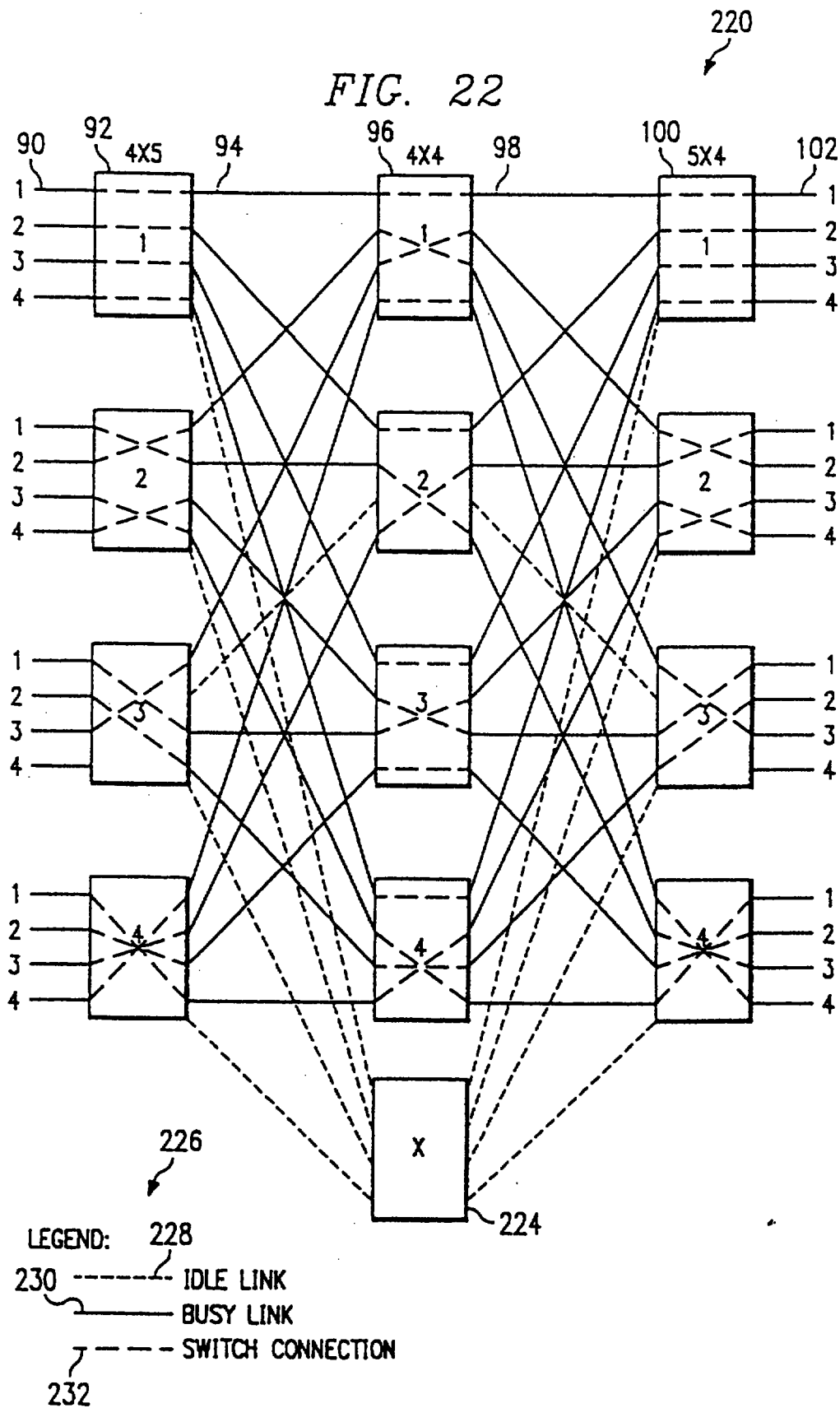
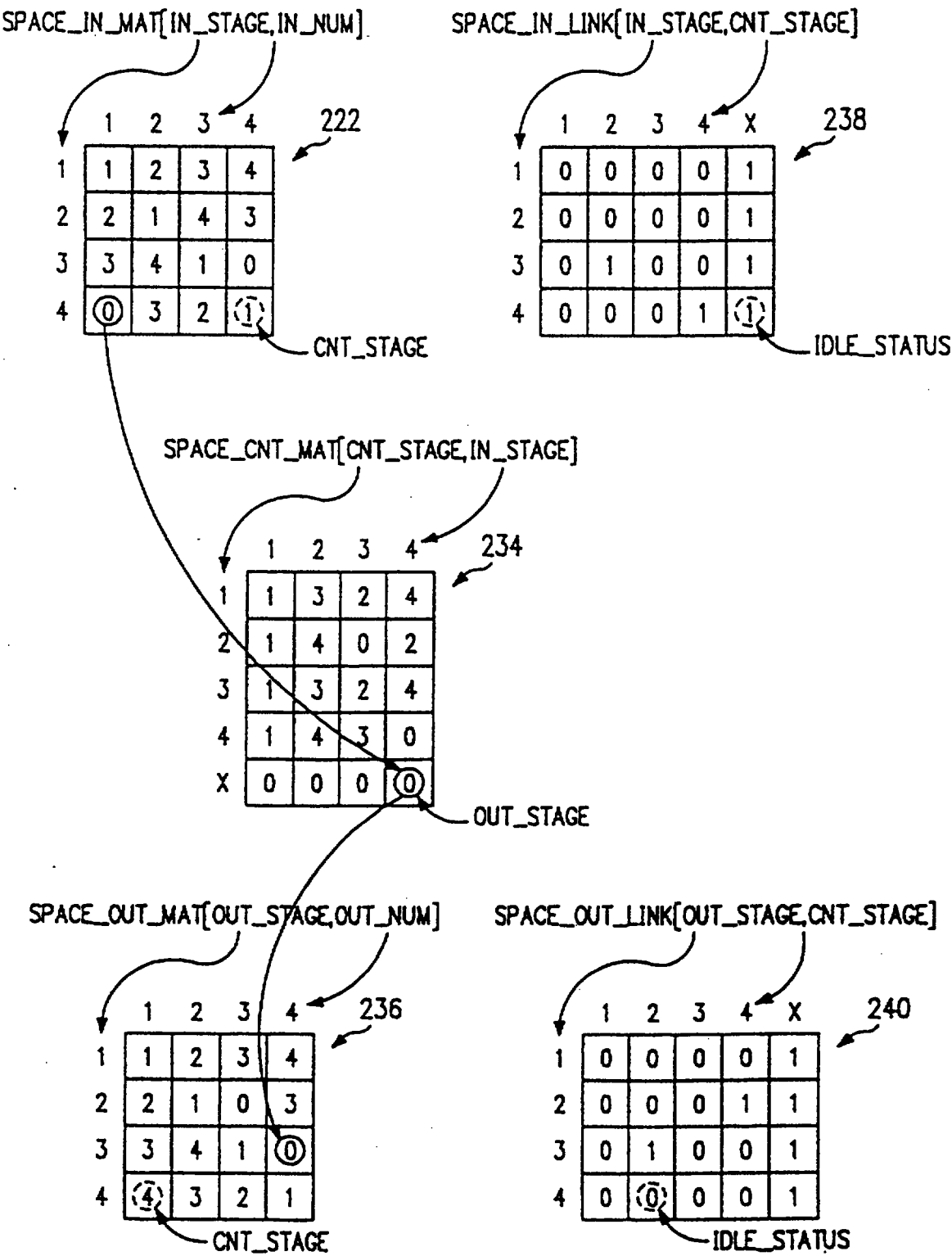


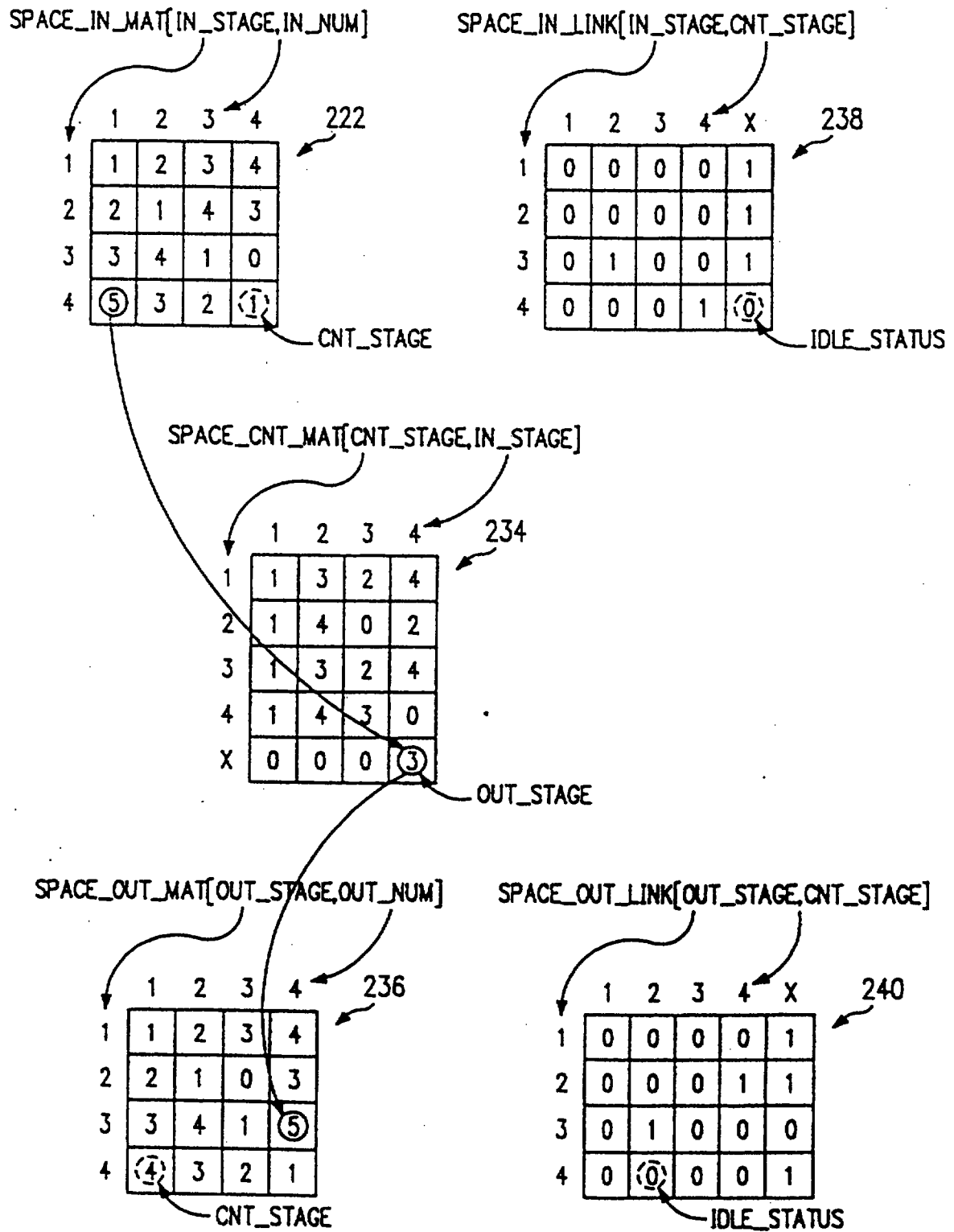
FIG. 23



24 APR 1997

- 16 -

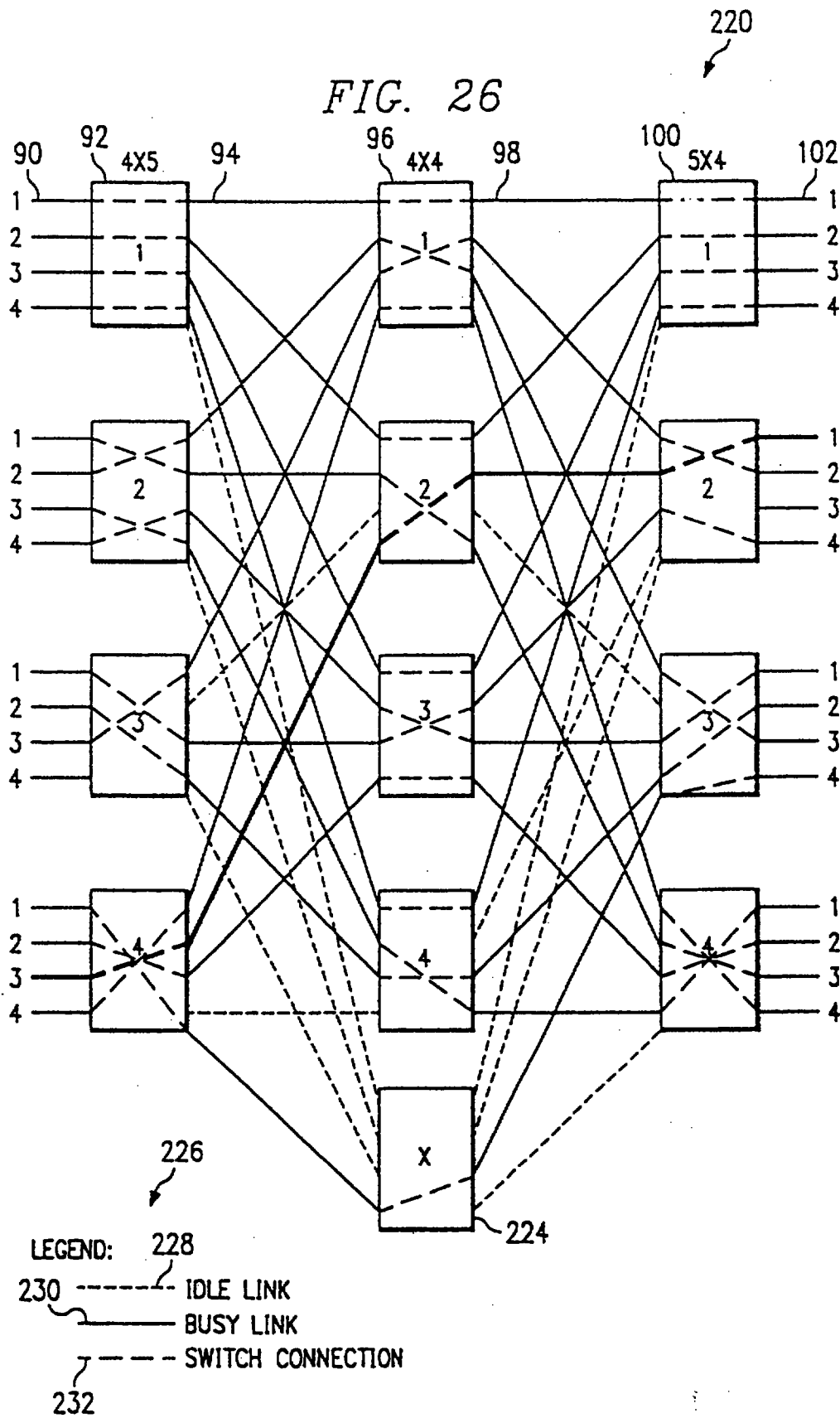
FIG. 24



24 APR 1997

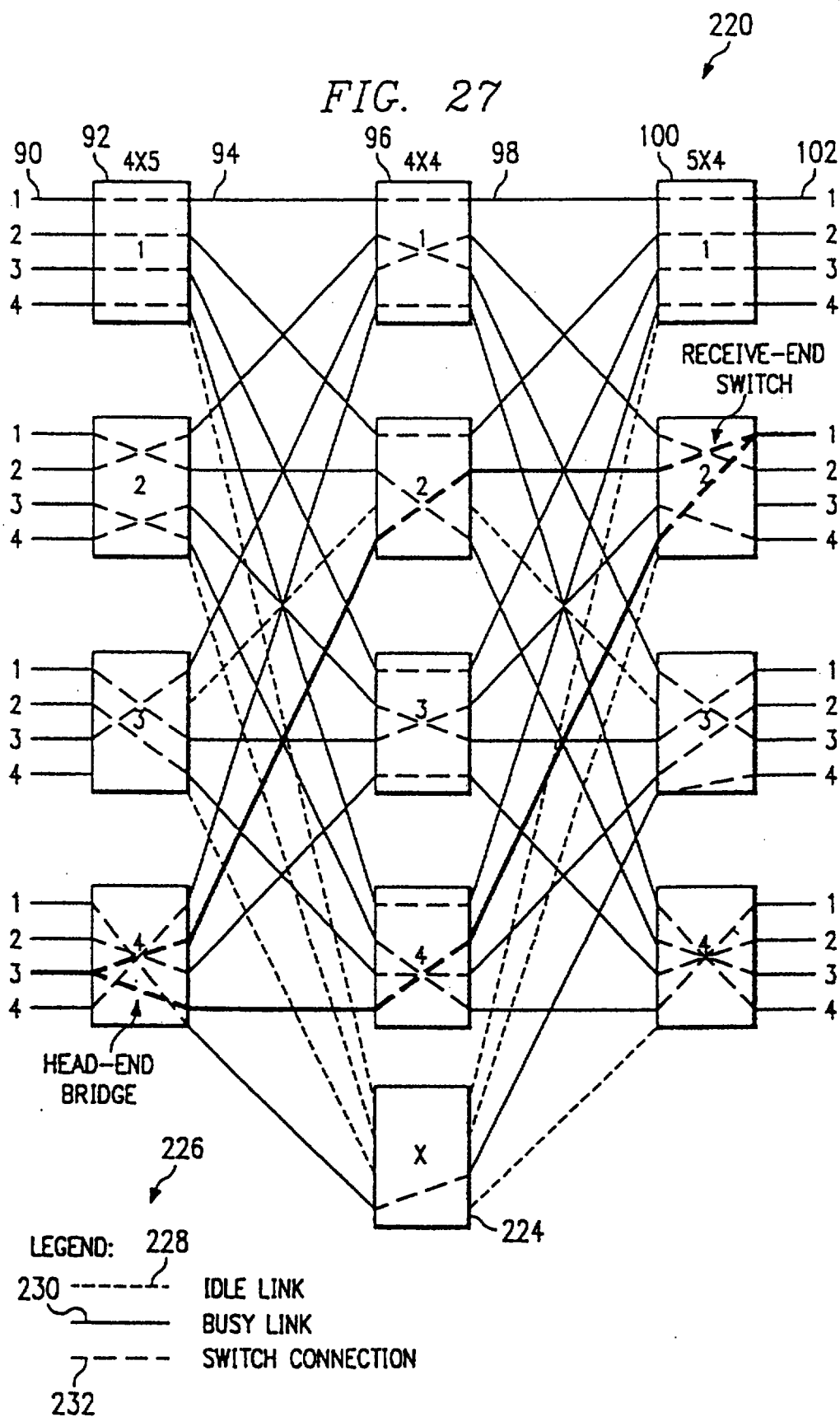
- 18 -

FIG. 26



24 APR 1997

FIG. 27



24 APR 1997

- 20 -

FIG. 28

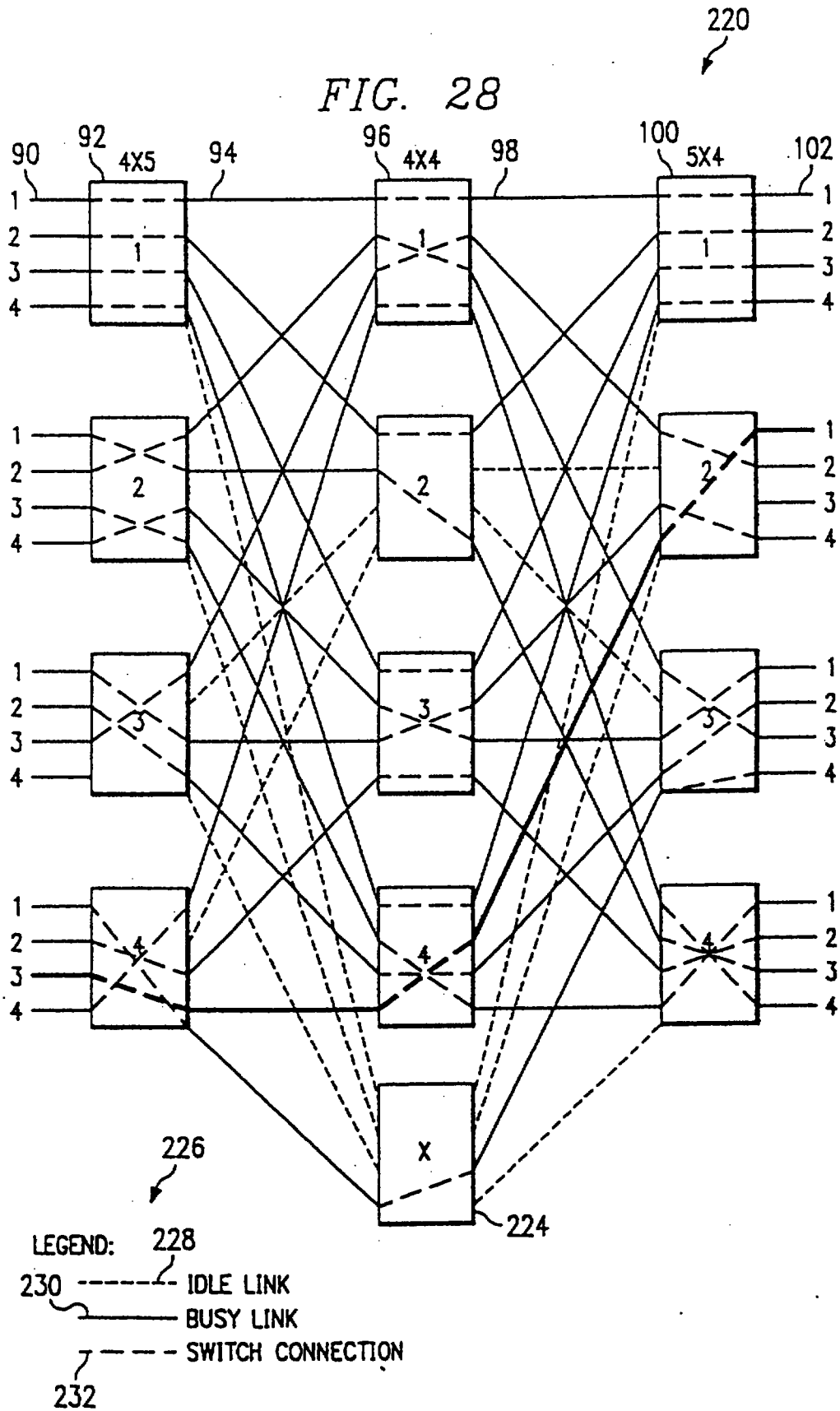


FIG. 29

