The digital data exchange—A space-division switching system

by E. Hopner
M. A. Patten

The user requirements for an automated switch for computer terminals in a large establishment or laboratory environment are discussed. This is followed by an explanation of the design requirements, and finally by a description of the significant features of the digital data exchange (DDEX), a microprocessor-controlled user-transparent space-division digital data switch of modular design which is capable of connecting 512 to 2048 lines from various types of IBM terminals to different control units, thereby substantially saving interconnection resources.

Introduction

Interactive data processing in the IBM San Jose complex has seen a dramatic expansion over the last decade. Computer communications problems became particularly pressing in early 1978 and our data processing planners came to realize that something had to be done to accommodate growth requirements in cabling to meet our ever-growing needs for more computer terminals.

Up to six cables connected each user terminal to different computer ports, and terminal users could make their choice of programming system options by means of a manual switch. Replacing the manual switch with a microprocessor-controlled switching system at the computer site (installed prior to 1978 at our Santa Teresa programming center) was a significant step in the right direction. One cable per terminal, instead of up to six, certainly provided growth capabilities for the installed base of our computing centers. However, we were still using up to six ports per user, where only one is needed. Furthermore, in case of failure of a port or host processor, the cable connection had to be physically moved; therefore, a more elegant solution was sought.

What we needed was a switching system (called the digital data exchange and hereafter abbreviated to DDEX) which would connect a terminal to the first available computer port of the system chosen by the user. The DDEX should recognize a user's request made at his terminal and make the desired connection once it has established that the user has authorized access to the requested facility. Instead of limiting the user to six different computer ports, we could provide him access to all available ports in the facility.

The DDEX function had to be accomplished without requiring any hardware or software changes in the installation, and for less than the cost of additional cabling. When we considered the development of a new computer system to satisfy our needs within IBM, it became obvious to us that the DDEX should not limit its function to current terminals and their data rates. We also forsaw the need for rapid access to files, sharing of high-speed printers, and switching of computer channels in the future. Thus, the speed requirements for our connections were set at 12 megabits per second (Mb/s) to accommodate a data transfer rate of 1.5 megabytes per second. Once the connection has

been established, the user should be able to keep it permanently if needed. Therefore, simultaneous connections for all users to their facilities should be provided. This connectability requirement is different from that of telephone users, where only 10 to 20 percent of the users need the switching facility at the same time.

As a means of satisfying the speed and occupancy requirements economically, we postulated a high-speed digital data switching system. To accommodate a wide spectrum of speeds and data transmission protocols in our computer installation, transparency in relation to speed and protocol was required. Also, from the point of view of the user, the system must be nonblocking. This means that a user is always guaranteed a data path (connection) through the DDEX if a terminal control unit port is available which can be used to satisfy the user's connection request. Finally, a modular system handling 500–2000 terminals would meet our expected capacity requirements.

A high-speed (75 Mb/s) nonblocking feasibility model of a space-division switch had been demonstrated at the IBM Los Gatos Laboratory in 1977. Thus space-division and time-division switching systems were considered, and loop, bus, and star configuration topologies were studied. Our conclusion in 1978 was that the only configuration to satisfy our current and projected needs was a high-speed-per-connection digital space-division star system with a nonblocking switching capability and a data transparency feature. Such a system was feasible with a total switching capacity in excess of 24 Gb/s (12 Mb/s per connection × 2000 connections). A single time-division loop or bus would have limited us to 100 Mb/s or less, far below our goal.

The DDEX satisfying these needs was proposed in 1978 by our San Jose site, to be built for its internal use. Subsequently, the DDEX was developed [1–3] by the IBM laboratory in Raleigh, North Carolina, from start in 1979 to actual delivery in 1981. The first DDEX installation was operational at our Santa Teresa programming center by the end of 1981. Since then, a number of these systems designed specifically for terminal switching use have been successfully installed at other IBM sites.

The DDEX system operation, design details, an overview of hardware and software considerations, and the user perspective are described in the following sections.

System operation

The DDEX is connected to each user terminal by a single coaxial cable. This same cable originally led back to a terminal control unit port. Instead, all control unit ports at the site are cabled to the other side of the DDEX, which, on user request entered at a user's terminal, makes the proper port connection. The DDEX provides various functions to the users in response to their requests. A user must first indicate to the DDEX his desire to make a request by activating the DDEX *attention* switch attached to the terminal coaxial cable. Activating the switch causes a condition to occur on the cable which is distinguishable from data transfers by the DDEX. Upon detecting this condition, the DDEX sends a menu to the terminal screen. The user selects the desired option from the menu and makes an appropriate entry using his terminal keyboard. The DDEX interprets the entry and executes the related function (usually a request for a connection to a particular service, or a disconnection request).

When a connection is established, the DDEX provides a transparent data path between the terminal and a computer port capable of providing the requested service. The DDEX then takes on the passive role of monitoring the connection for usage. This function is provided by activity detection circuitry. If a connection is not utilized for a period specified in the DDEX system tables, the DDEX automatically breaks the connection. This condition is referred to as an *inactivity timeout.* Connections which are broken in this manner (or by user request) free the computer port for use by other users.

The DDEX provides a limited set of functions to the user which are equivalent to certain telephony features. For example, if a user has been granted the privilege, he may put a connection to a given port on hold while using another port. Also, if no ports are available for a given system, the user's request is put on queue and he is connected to a requested port as soon as one becomes available. Since users are no longer connected to a dedicated port on each system, the DDEX provides another function whereby the user may interrogate the DDEX for the address of the physical port to which his terminal is presently connected. This is useful in failure isolation when a problem is encountered.

A port which has been disconnected by the DDEX from a terminal may be considered free for subsequent connection to another user's terminal. This might, in some cases, result in a data security problem if the previous user were in midsession at the time of a faulty disconnection and the system did not perform a logical disconnect when the electrical path was broken. The DDEX provides a security function to eliminate this problem, called *logoff verification.*

When the DDEX communicates with a user's terminal, hardware known as *controller emulation* circuitry is used to emulate a control unit. In the case of logoff verification, the DDEX emulates a terminal while connected to a control unit port that requires verification. The DDEX emulates various terminal operations to determine whether the system logo or other information indicative of a logged-off condition is present on the particular port. A positive verification results in the port being considered free for subsequent connection to another terminal. If the operation is not successful, it is retried at timed intervals until successful. In the meantime, the port is considered unavailable to all terminals except the terminal that was last connected to the port.

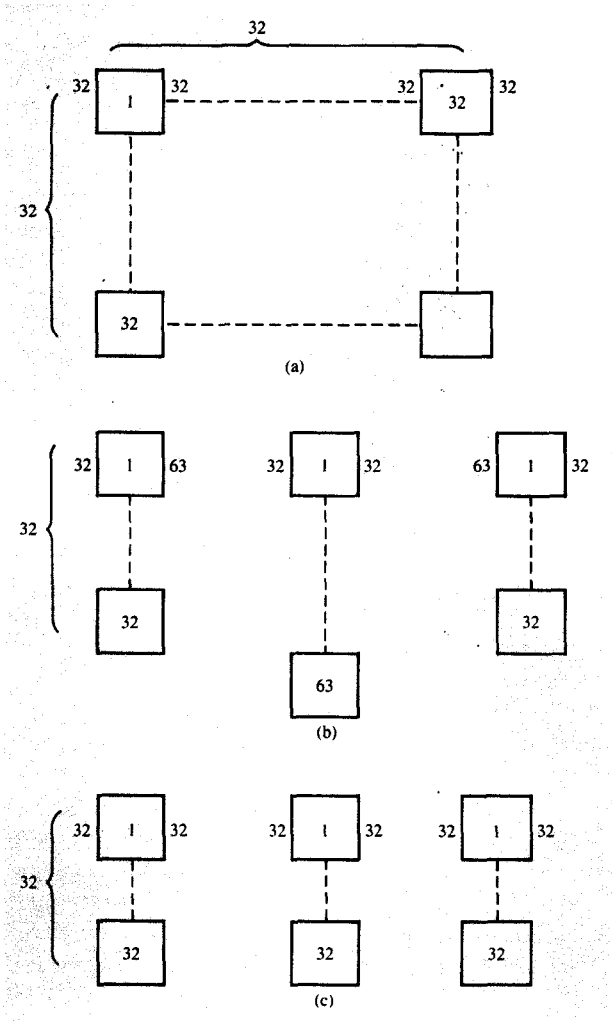

Figure 1

Three network configurations: (a) Single-stage matrix; (b) Three-stage network with $2n-1$ secondary and no call rearrangement; (c) Three-stage network with call rearrangement.

The DDEX provides other security-related functions. Applications may be set up in the DDEX system tables (to be discussed later) to require a password for connection. User may be restricted from accessing certain systems. Certain ports on a given system may be assigned as dedicated for use only by a certain user. This is not only for security but also for purposes of guaranteeing that a critical user, such as a systems programmer, has access to the system he needs.

The DDEX *system tables* are used to define applications, users, and systems parameters. Certain entries simply define on which ports a given operating system or application program is provided. Other parameters define which special functions are implemented and serve to tailor the function of the DDEX to the requirements of the users.

Tables are loaded from a supporting system to the DDEX through a standard control unit port attached to the DDEX. The DDEX utilizes terminal emulation mode to facilitate data transfers from the port. This link is supported by a control program. The control program also provides the means by which the tables are specified and verified. After loading of the system tables, the program is used to provide a variety of administrative functions. One of these functions is the retrieving of activity and error logs from the DDEX, either at the request of the DDEX (log area nearly full), or as the result of a manually or automatically initiated command.

Switch design

From the system operation requirements it was determined that

- A switch matrix had to be designed which would be capable of switching the terminals at their native interfaces, thus being transparent to them.
- The switch matrix had to be designed for modular growth, say from 512 × 512 to 2048 × 2048.
- The switch matrix had to be designed to maintain the connections and display its condition even with a host system crash.
- The switch matrix had to be program-controlled.
- To maintain high system availability, the program had to be executed by a separate control processor and not by the host system.
- Also for system availablity, two control processors were needed.

• *Switch matrix*

Since the switch matrix requirement called for a modular growth capability, an LSI FET chip was designed to serve as the basic building block of the switching network. Three designs shown in **Figure 1** were investigated to provide growth to a 2048 × 2048 matrix:

- single-stage matrix,
- three-stage Clos [4] network with $2n - 1$ secondaries, and
- three-stage Clos [4] network (with call rearrangement) [5].

Call rearrangement requires connections to be dynamically rerouted in the network. That is, old connections are re-established over new paths to make room for new connections in a process known as *call rearrangement.* For our application, call rearrangement must not disturb end-to-end continuity of the connection.

The Clos network was implemented because with it a savings in switching hardware of 5.4 to 1 could be realized in a 1024-line configuration when compared to a large single-

matrix implementation. In addition, call rearrangement was implemented to make the reduction over a single matrix 10.7 to 1. Thus call rearrangement facilities had to be designed into the switch matrices to perform this function.

Another requirement to be met is maintaining the connection independently from the host status; for this, a static memory array of dimensions equal to those of the crosspoint array is needed. Each memory element is designed to be adjacent to the crosspoint device. Thus, each crosspoint device can be controlled from its adjacent static memory element on a one-to-one basis. Furthermore, the static memory element can be interrogated for the status of each interconnection, thereby eliminating the need for storing the status as network maps in the network control processor.

A block diagram of the switch chip is shown in **Figure 2.** The switching array provides connectivity between any of 32 standard inlets and any of 32 standard outlets. Two special inlets may be connected to the standard outlets and the standard inlets may be connected to two special outlets. One special inlet-outlet pair is used for call rearrangement. The other could be used both as a path to control the chip and as a path for the processor to communicate to attached devices. All chip outputs are re-driven by specially designed off-chip drivers.

Connections in the switching array are established, broken, and monitored under control of *bit* and *select* decoders. Addresses which determine a particular path to be controlled are supplied by the *command deserializer*, which receives switch network commands over the control inlet. The deserializer also provides the command processor with operation codes defining the activities to be performed. As a result, the processor generates timing pulses to the decoders, causing appropriate action to be taken in the array. The status serializer generates a status response on the *control outlet* based on status from the network processor and array status latches.

A small number of control lines are required to drive the chip. This keeps the necessary support logic small. All internal timings are derived from a single symmetrical clock input, thereby further reducing system complexity. Since the control of the chip involves high-level commands, the software control of the entire network is simplified.

An abbreviated representation of the switch network is given in **Figure 3.** Blocks labeled PL, PH, SL, SH, TL, and TH represent switch matrices which are configured as a three-stage Clos network. The PL and PH matrices are primary switches in the network and the TL and TH matrices are tertiary switches. In an actual 1024-line DDEX, 32 primaries and 32 tertiaries are contained on 16 primary cards. Also, 32 secondary matrices (SL, SH) are contained on 16 secondary cards.

As stated later, for reliability reasons, the switch matrix is controlled by dual independently running network processors. Thus both primary and secondary cards contain logic for interfacing the network to these processors, SNI P and SNI S, respectively. In addition, the primaries contain

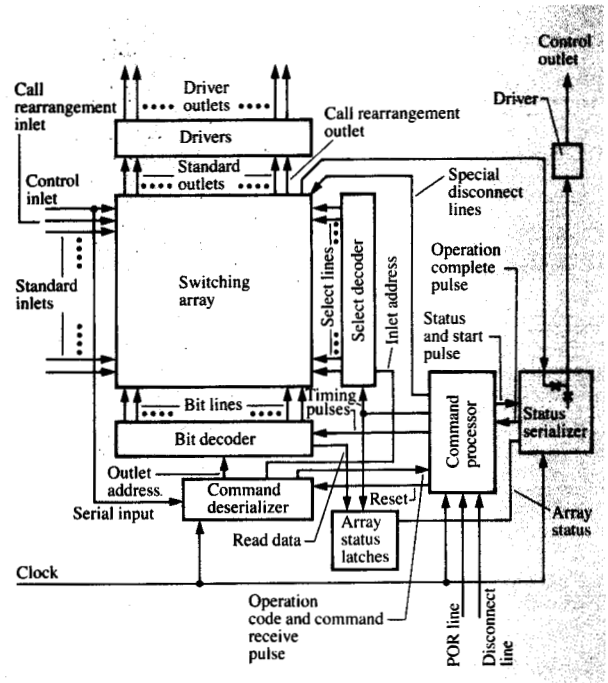

Figure 2

Chip organization.

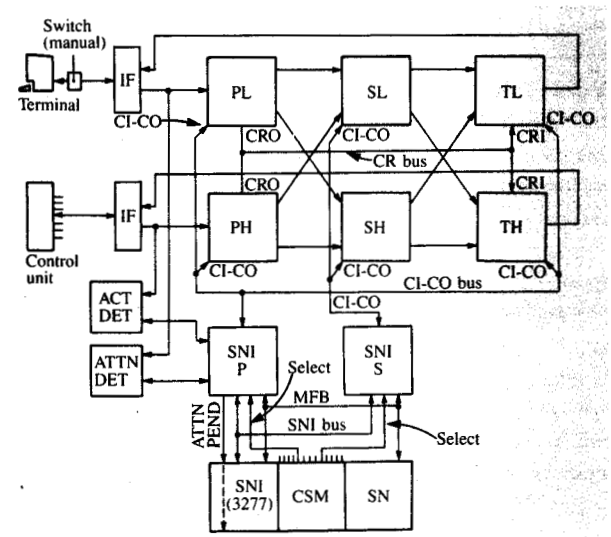

Figure 3

Switch network cross section.

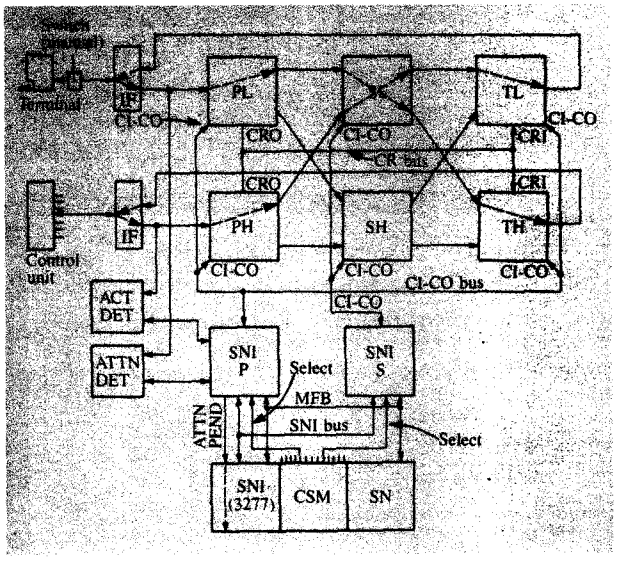

Figure 4

Standard terminal–port connection.

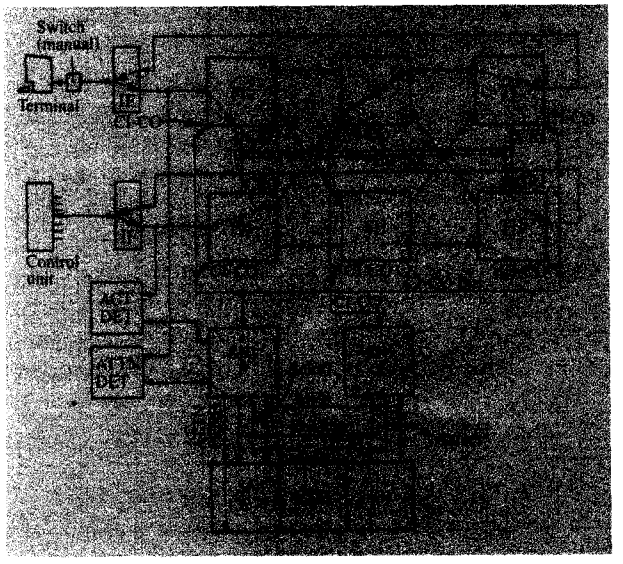

Figure 5

Call rearrangement temporary alternate path.

activity detection (ACT DET) and attention detection (ATTN DET) logic. The primaries also contain interface circuitry (IF) to adapt the attached lines to the switch network.

• *Interface circuits*

The purpose of the interface circuits is to adapt the electrical and functional characteristic of the device coaxial interface to that of the switch network. Appropriate drivers and receivers are implemented to match the electrical characteristics of the coaxial signals to that of the switch network.

Functional adaptation is provided by a digital hybrid which converts the half-duplex coaxial signal into two simplex network channels. Connectivity is thus provided between two coaxial lines when the related interface circuits are cross-connected input-to-output and output-to-input through the switch network. Such a connection is shown in **Figure 4.** Note that the connection is composed of two symmetrical three-stage network paths.

• *Activity and attention detection*

The outputs of the interface circuits to the primary matrices are monitored by detection logic. Lines related to terminals are monitored by attention detection logic which is multiplexed for a group of 32 lines. This logic samples each line to determine whether the associated attention switch is activated. If an attention is detected, an indication is stored in a memory at an address related to the particular line.

Whenever an attention indication is stored, an attention-pending line (ATTN PEND) is asserted to the network processor. The processor microcode responds by polling each switch network primary card for attention indications.

The function of the activity detection logic is similar to that of the attention detection logic. In this case, lines related to control unit ports are monitored in groups of eight for bursts of data indicative of buffer transfers. (Buffer transfers are differentiated from polling sequences by the duration of the data bursts.) Again, an indication is stored in a memory if detection occurred. However, since activity information is used by the processor to time-out idle connections, it is not necessary to send an activity-pending indication to the network processor. Microcode timing routines determine polling intervals for activity indications.

• *Call rearrangement connections*

In order to implement call rearrangement, it is necessary to remove temporarily an existing connection from the switch network. Figure 4, referred to earlier, shows a network connection. **Figure 5** shows that same connection with the terminal-to-control-unit-port path removed from the network. Since end-to-end continuity must be maintained, an auxiliary path is provided over the call rearrangement bus (CR bus). This path, in fact, bypasses the secondary switches to provide a direct connection between a primary outlet and a secondary inlet. Two CR buses are actually implemented since the call rearrangement algorithm requires a maximum of two paths to be removed from the network

simultaneously. Since there exists only one CR outlet on each primary and one CR inlet on each tertiary, switching is required between the CR outlets, the CR inlets, and the two CR buses. This switching, in a sense, represents an additional secondary in the network and is controlled by the switch network interface logic in response to commands issued by the processor.

An illustration of a call rearrangement sequence is given in **Figure 6.** In the first step, a network with two existing connections is shown—C-to-D and E-to-F. A third connection is desired—A-to-B. However, no path exists to complete this connection. The second step shows the C-to-D connection transferred to the CR bus. The original C-to-D connection is broken in the third step. A new C-to-D connection is established in the fourth step, and the parallel call rearrangement path is broken in the fifth step. At this point the condition in the network originally blocking the A-to-B connection no longer exists. Thus, in the sixth step, the A-to-B connection is made.

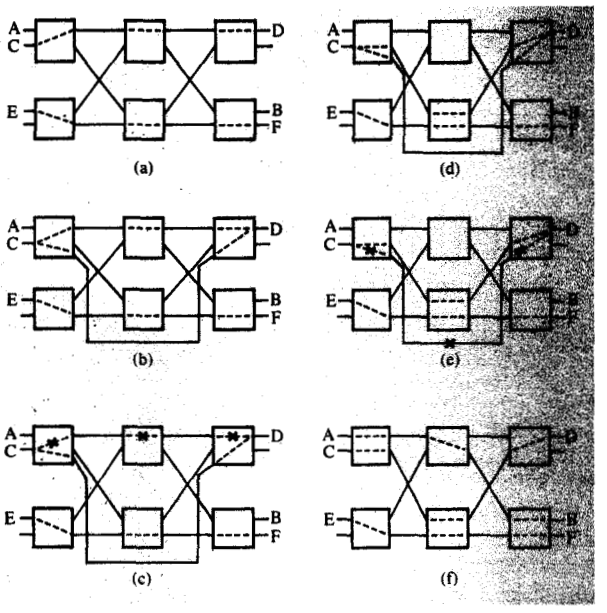

Figure 6

Call rearrangement: (a) Network blocked from making A-to-B connection. (b) Parallel call rearrangement connection made from C to D. (c) Standard connection made from C and D broken. (d) New C-to-D connection made. (e) Parallel call rearrangement connection between C and D broken. (f) A-to-B path no longer blocked; connection established.

• *Machine architecture*

The DDEX structure is illustrated in **Figure 7.** The uppermost blocks represent duplicate processors. Each processor operates independently of the other to minimize the likelihood of a total system failure. Below the processors are two separate groups of adapter logic which convert the data and control information provided by the processors to a format compatible with the switch network and the attached switched devices. The combination of processor and adapter logic is completely duplicated and isolated at all levels. Thus each is capable of controlling the switch network independently, with no reliance on the other.

In the lower half of the figure, the switch network is represented as a vertical and horizontal arrangement of logic cards. This arrangement provides interconnections between each primary and secondary card without reliance upon extensive board wiring. This is an important factor in the physical implementation of space-division switching. The switch-network cards provide all switching, control, monitoring, and signal-conversion functions necessary for interconnection of devices at their native interface. This includes

- functional and electrical adaptation of native interfaces to that of the switch network,
- detection of requests for service and presentation of these requests to the microprocessors,
- providing connectivity between attached devices requesting service and the appropriate microprocessor communications adapter,
- establishing of connections or breaking of connections requested for devices in response to microprocessor commands,
- monitoring of data traffic once a connection is established, and

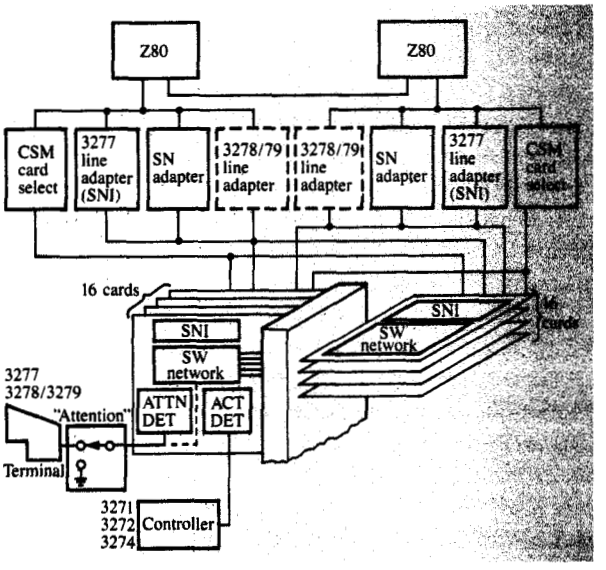

Figure 7

DDEX control structure.

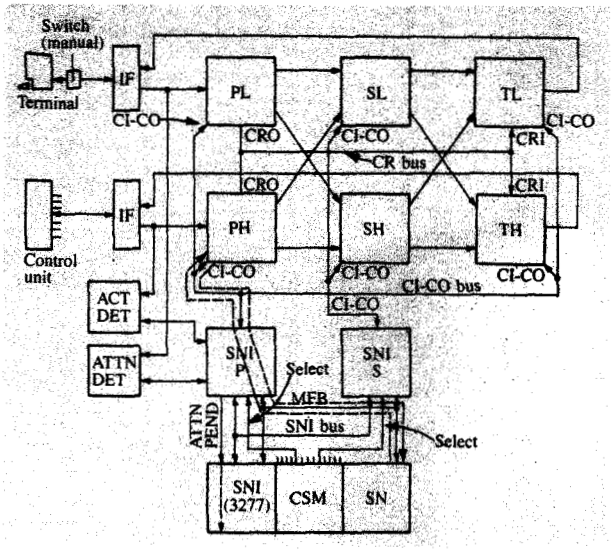

Figure 8

MFB to switch matrix SNI connection.

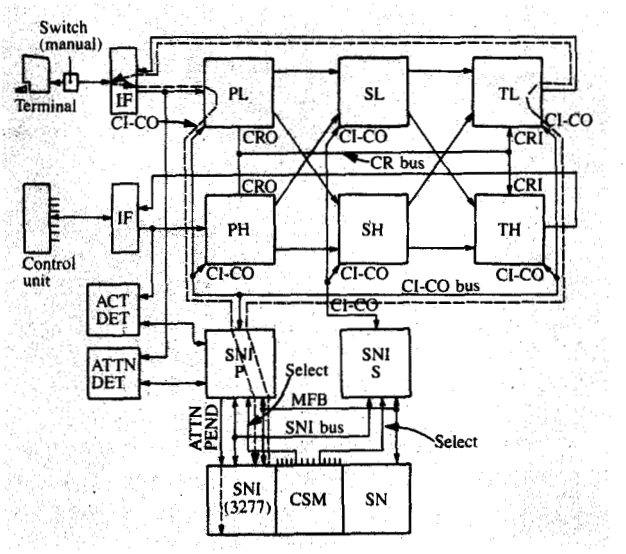

Figure 9

Emulator–device connection.

- presentation of error/status/configuration information to the microprocessors upon demand.

Since the DDEX switches devices at their native interfaces, no modem or adaptor logic is required between the DDEX and the devices. Also, because of the functions provided by the switch network and the device-compatible adapters associated with the processors, the DDEX communicates with attached devices directly without the need of auxiliary control paths. These features combine to provide a switching system which is transparent both from the standpoint of control in obtaining connection request information and from the standpoint of native interface compatibility. Last, and possibly most important, are the truly transparent end-to-end connections provided between devices by the DDEX three-stage nonblocking unclocked space-division switch network.

The duplicate network processors are shown at the top of Fig. 7. The processors as well as the adapters have been duplicated to enhance system availability. One processor controls the network while the other runs background diagnostics and receives messages from the *active* processor indicating the state of the network. A failure in the active processor causes the standby processor to take control. The switchover of control may also be induced manually or under program control.

All control between the processors and the switch network passes through adapter logic. This logic translates the signals of the processors to that required by the switch network and attached devices.

- *Switch-network interfaces*

The switch-network interface (SNI P for primaries, SNI S for secondaries) is the logic that controls various switch-network card functions in response to switch-network processor commands. As mentioned earlier, the SNI controls connections involving the CR buses. It also establishes connections between the multi-function bus (MFB) and the switch matrices for the purpose of controlling the matrices and providing paths between the IBM 3277 or IBM 3278 line adapters and attached devices. (Refer to **Figures 8** and **9**, respectively.)

During power-up initialization, the SNI disables the IF circuits until the network is cleared. Also, during bring-up (and background diagnostics), the SNI controls a wrap function provided by the IF circuits for the purpose of testing network path continuity. One such diagnostic wrap connection is shown in **Figure 10.** By comparing Fig. 10 with Fig. 4, it can be seen that such a connection represents two-thirds of an actual port and terminal interconnection. By repeating the wrap connection for the port, a complete test can be performed, with the secondary tested twice. The wrap function of the interface circuit causes data to be passed from the line driver to the line receiver. This ensures that all components are tested.

Attention and activity indications stored in the ACT DET and ATTN DET logic are presented to the network processor by means of poll commands to the SNI. The SNI also provides status responses to all commands. The status information includes error information and card addresses to verify that the correct card is actually selected. In response to

a diagnostic sense command, additional error information and the card type are provided. The card type is useful in verifying system configuration.

The SNI communicates with the duplicate network controllers over two completely independent interfaces. The active interface is determined according to which processor has asserted *card select* (to be explained later). Since the secondaries do not contain IF circuits, ACT DET or ATTN DET logic, or CR buses, only a subset of the complete SNI command set is implemented for these cards.

• *Adapter logic*

Adapter logic provides a translation between the controls required by the switch network cards and attached devices and the network processors. The adapters are controlled on the processor side by parallel input output microprocessor peripherals (PIO). Adapters designed include a card-select multiplexer (CSM), a switch network interface (SNI), a switch network adapter (SN adapter), an IBM 3277 line adapter, and an IBM 3278 line adapter. These are discussed in more detail below.

The CSM provides a decode function to generate the various card selects. The decoder may be disabled under program control during normal operations or by hardware if control is to be transferred to the backup controller after a failure. Switch network cards cannot drive the interface-to-adapter logic unless their respective card-select lines have been asserted. This function was designed to improve failure isolation.

The SNI converts a bidirectional PIO interface to that of the SNI logic present on the switch network cards. The two interfaces are quite similar and only minimal translation occurs. This logic is packaged on the IBM 3277 line adapter card.

The SN adapter provides a serializer/deserializer function between the bidirectional PIO interface and the serial control inlets and outlets of the switch matrices. The adapter receives parallel data from the PIO which it serializes, at the same time adding framing and parity bits. A serial response is then received from the switch matrix, which in turn is checked for correct framing and parity bits. A parallel response with a check bit is presented on the PIO interface to be read by the microcode. For diagnostic purposes, the SN adapter may be wrapped at the serial interface, either internally as a self-check or externally as a check of a test connection in the switch network. Data generated for normal or wrap operations may also be programmed to contain bad parity so that the adapter and switch matrix parity checkers can be tested.

The IBM 3277 line adapter, in conjunction with processor microcode, provides IBM 3272 control unit and IBM 3277 terminal emulation. The adapter receives parallel commands and data and returns status and data over the processor PIO interface. Device-compatible serial input/output data are

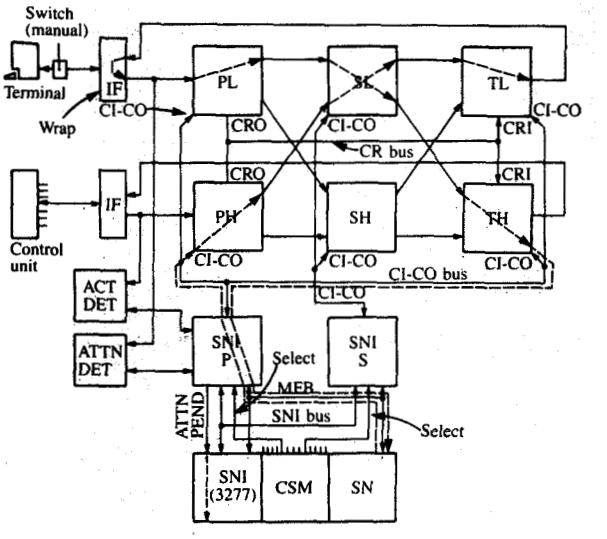

Figure 10

Diagnostic wrap connection.

received and transmitted over the multi-function bus (MFB). A connection between the MFB and a device is provided by a switch network primary card as described previously.

Terminal emulation is used to communicate with the host system for the purpose of loading machine configuration data during initial table loading and for log transfers and other administrative functions during operation. This mode is also used for log-off verification of controller ports after a disconnection and before the ports are reconnected to different user terminals.

Since all switching actions are initiated via the user's terminal, controller emulation mode is used to service user requests. When the *attention* key is depressed at the user's terminal, a menu is sent in response. The user then enters data on the screen and presses *enter*, thereby causing a screen read to occur. Alternatively, the user may enter other predefined function keys to minimize key strokes required to cause an operation to be initiated.

The adapter also performs a wrap function for diagnostic purposes. Other features have been implemented to allow off-line checkout of the adapter during IPL and during normal operation by background diagnostics. The IBM 3278 line adapter provides the same function for IBM 3278 and 3279 terminals as the IBM 3277 adapter does for its terminals. The IBM 3274 controller emulation function is provided by cards designed for that purpose. These cards, together with processor microcode, provide the controller emulation function.

Terminal emulation is accomplished in a somewhat different manner. Existing cards for an IBM 3278 terminal

are used. However, since the cards were not intended to be driven by a controller, it was necessary to develop a converter to simulate keyboard entry and allow reading the terminal logic refresh buffer through the PIOs. With this configuration, the microcode and converter card appear to the terminal logic as an operator entering keystrokes.

As in the case of the IBM 3277 adapter, the serial input/output data path is connected to a device over the MFB and a primary card controller connection. Off-line data wrap and other diagnostic functions are provided for testing during IPL and by background diagnostics.

• *Network processors*

The DDEX is controlled by two completely separate processors and adapters. The processors are identical and drive the switch network over two independent interfaces. The processors incorporate an 8-bit Z80 microprocessor [6] with 32K bytes of ROM and 32K bytes of dynamic RAM. A one-second time base was included to facilitate microcode timers and real-time functions. An operator panel is connected to the processors for control and diagnostic purposes.

The Z80 microprocessor was selected because of its ready availability, speed, and powerful bit-mode instruction set. Due to the lack of error checking (in microprocessors in general), two Z80s were incorporated in each network processor. One Z80 actually runs the machine, the other receives the same inputs and generates the same outputs if the processors are working correctly. If not, compare logic generates an error that disables the controller and causes the standby twin processors to assume control. Parity checking was also added to the Z80 such that each I/O or memory write operation causes parity to be generated. During each read operation, parity is checked. The lower 24K bytes of ROM is patchable in 1K-byte regions by installing a programmable PROM patch card.

All processor input/output operations are performed using standard Z80 PIOs. The PIOs are used to generate system control lines, vector interrupts, interface to the operator panel, collect error information, transfer data processor-to-processor, and interface to adapter logic.

To enhance failure isolation, a parity bit was added to the PIO interface adapter logic. A *no-data-present* indication was also added since that function is not provided by the PIO.

During normal operation, one network processor controls the switch network and the other runs background diagnostics off line. The processor in control sends messages to the other processor each time an action takes place and control tables are modified. In this manner, the standby processor remains up to date on the current state of the machine.

If a hardware-detected catastrophic failure occurs, switchover to the standby network processor and the disabling of the active network processor takes place automatically. If a retryable error occurs and persists, or if an adaptor error or certain switch network errors occur, the switchover is accomplished under software control.

Switchover may also be initiated through the operator panel. This results in a software switchover with the previous active processor becoming standby (or disabled selectively). If a panel switchover is attempted when the standby processor is not running, the operation defaults to a reset and warm restart of the active processor.

DDEX performance

The performance of the DDEX as a system has two aspects. One is the establishment of a connection (in response to a user request). The other is the characteristics of the established path.

First, the shortest time required by the DDEX to make a connection from the time a request is made is about 200 ms. If, however, one percent of the users on a fully populated system requested connections at the same instant, the last of these requests to be serviced would take in excess of two seconds (assuming 1200 users total). Periodically, requests may be somewhat delayed if made concurrent with a DDEX-to-host-system data-transfer operation.

Once a connection has been established, a transparent connection exists. The only delays encountered in the data path are the logic delays through the system, which are less than 200 ns. Assuming the use of a data transmission technique that can tolerate at least 20 percent distortion, the DDEX will pass data at up to a 12-Mb rate. Since the system is digital, crosstalk takes on the form of induced distortion, and in the worst case contributes from 2 to 3 ns to total system distortion.

DDEX summary

The DDEX was designed to switch IBM 3277, 3278, and 3279 keyboard/display terminals to controller ports attached to different computer systems in the IBM San Jose facility at the direct request of the terminal users. It provides

- high-speed application switching without response time delays,
- improved application availability with or without system operator intervention,
- minimum installation effort (transparency), and
- user-friendly menu selection interface.

Handling up to 2000 connections at a rate of 12 Mb/s per connection required the design of a space-division star-configuration switching system rather than a time-division switching system.

The measurable cost savings of a DDEX to an installation are in the following areas:

- reduced terminal control unit requirements,

- reduced multiple coaxial cable requirements,
- freeing up of raised floor space, and
- reduced installation cabling and patch panel work as users move offices or change access requirements.

In addition to the basic cost savings associated with the DDEX, additional benefits are derived from the DDEX as a powerful installation management tool for the terminal attachment environment, both from the daily operation (load sharing) and from the long-range planning (usage statistics) point of view. Also, the reduction in UCB address assignments relieves channel address overloads (since it is no longer necessary to maintain system definitions for unused or under-utilized resources).

Acknowledgments

The authors are indebted to many who worked on and encouraged this project. We wish to recognize the contributions of key technical people, especially Despina Boinodiris, C. Michael Melas, and Richard Talmadge for the architectural and functional definition of the DDEX, Frank Davis, Jack Dickerson, Paul Evrenidis, Richard Gimigliano, Leon McAllister, and ErwinWinz for its hardware development, Gwen Barnes, Quincy McIlwain, Michael Zapata, and C. Michael Melas for the microcode development, and also Paul Heckler, for the host system support software development.

References

1. G. F. Abbott, "The Digital Data Exchange—A Broad Band Data Switching System—An Overview," *Proceedings*, 1982 IEEE Global Telecommunications Conference, Miami, Nov. 29–Dec. 2, 1982, IEEE Order No. 82CH1819-2, pp. 478–481.
2. C. M. Melas, "The Digital Data Exchange—Architecture and Software Description," *Proceedings*, 1982 IEEE Global Telecommunications Conference, Miami, Nov. 29–Dec. 2, 1982, IEEE Order No. 82CH1819-2, pp. 482–485.
3. M. A. Patten, "The Digital Data Exchange—Logical Implementation," *Proceedings*, 1982 IEEE Global Telecommunications Conference, Miami, Nov. 29–Dec. 2, 1982, IEEE Order No. 82CH1819-2, pp. 486–491.
4. C. Clos, "A Study of Non-Blocking Switching Networks," *Bell Syst. Tech. J.* **32,** 406–424 (1953).
5. C. M. Melas, "Path Rearrangement in a Data-Switching Network," *IEEE Trans. Commun.* **31,** 155–157 (1983).
6. J. J. Carr, *Z80 Users Manual,* Prentice-Hall–Reston Publishing Co., Reston, VA, 1980.

Received November 8, 1982; revised June 27, 1983

Emil Hopner *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.* Dr. Hopner joined IBM in 1955 at San Jose, where he is currently an advanced signal processing consultant. His interests at present involve the use of computer simulation in studying the next generation of storage structures and technologies. Since joining IBM, Dr. Hopner has been involved in technical and managerial capacities with automatic speech recognition, modems, magnetic recording, communications, and advanced information retrieval technology. He received an electrical engineering degree from the Swiss Federal Institute of Technology, Zurich, in 1945 and a doctorate in technical sciences from the same institution in 1962. Dr. Hopner has received two IBM Outstanding Achievement Awards for work in data transmission systems and distortion correction, and an IBM Outstanding Achievement Award for technical leadership of the DDEX program. He holds fifteen U.S. patents and the IBM First- and Second-Level Invention Awards. He has published on the subjects of microwave propagation, modems, automatic equalization, and magnetic recording. He is a member of the Institute of Electrical and Electronics Engineers and Sigma Xi.

Michael Allen Patten *IBM Research Division, 5600 Cottle Road, San Jose, California 95193.* Mr. Patten is a telecommunications engineer at the IBM San Jose Research laboratory, currently working on the installation of telecommunications equipment for the Almaden Research facility. He joined IBM in 1967 at San Jose; since that time he has worked on the design of telephony and speech filing systems, high-resolution graphics circuitry, and head electronics for high-resolution monitors, as well as system- and circuit-level design of data switching systems and control processor development for telephone switching systems. Mr. Patten attended Stanford and San Jose State University. He holds IBM First- and Second-Level Invention Achievement Awards, and a 1982 Division Award for his work on the DDEX program.