SEL-TR-149

VLSI Design of a Crossbar Switch

B.A. Makrucki
T.N. Mudge

January 1981

Prepared under
National Science Foundation Grant MCS-8009315.

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
THE UNIVERSITY OF MICHIGAN, ANN ARBOR 48109

Abstract

This report describes two monolithic n-m crossbar switch designs suitable for use as building blocks in high speed interconnection networks. They use the advantages of VLSI implementation; high gate density and reliability. The designs are oriented towards NMOS technology. Two approaches to address transmission are given: one allows a high logic to pin ratio to be achieved; the second design is oriented toward faster setup times with a lower logic to pin ratio. Both designs attempt to minimize (not formally, but heuristically) network package count within a given pinout constraint. This in turn makes the construction of large networks feasible. Additions to the basic designs are discussed. The space complexity of a single n-m crossbar is derived. The setup time for a network of the crossbars is derived, and the number of n-m crossbars required to build the network is also derived. Interconnection networks constructed from basic n-m crossbars are suitable for constructing high speed multiprocessor systems.

TABLE OF CONTENTS

1. Introduction.

With the continuing decline in the cost of hardware the widespread use of multiprocessor systems is possible. Their use in appropriate situations can improve the performance of a wide variety of computing tasks particularly with respect to speed and reliability. To construct a multiprocessor requires an interconnection network (ICN) that allows fast processor-to-processor communication and/or fast processor-to-memory communication. Throughout this report we shall only consider ICN's for the processor-to-memory case, because in most cases, these can be readily adapted for processor-to-processor connections.

An obvious candidate for an ICN is a shared bus--several processors and memories connected to a shared time-multiplexed bus. However, a shared bus only provides high speed interconnection if it is very fast relative to processors and memories. If it is desired to connect N processors to M memories a shared bus' performance decays as N increases. To improve performance over the shared bus a crossbar switch may be used to establish multiple bus connections between processors and memories. A crossbar allows any set of simultaneous interconnections in which at most one input bus is connected to each output bus. Unlike the shared bus its performance does not decrease with increase in N. However, the component complexity of the crossbar grows as O(NM). For this reason past proposals for tightly coupling multiple processors to multiple memories have steered away from crossbar switches. In their place a whole range of ingenious networks have been proposed ranging from the early ideas of Clos and Benes [Clo53,Ben65] through the Omega network of Lawrie [Law75], the indirect binary n-cube of Pease [Pea77], the Delta network of Patel [Pat79], and the Data Manipulator network of Feng [WuF80], which have many of the connectivity properties of a crossbar without the component complexity. For a good survey of the state-of-the-art in networks see Siegel [Sie80]. The component complexity of these networks all grow as $O(N\log_2 N)$ rather than $O(N^2)$ (assume N = M).

In the context of VLSI technology it is no longer clear that reduced component complexity is an advantage **within** a single IC. For example, preliminary layouts for a Delta network and a crossbar carried out by us suggest that the reduced complexity networks do not appear to translate into more efficient space utilization in an IC layout. The unimportance of component complexity or device count as a measure of design efficiency in ICs has been discussed by Thompson and Franklin [Tho77, Fra80] among others. Furthermore, although the reduced complexity networks preserve some of the connectivity properties of a crossbar they do not preserve bandwidth [Pat79]. At the present time the limiting parameter is the number of pins required of any package that the IC might be put into. For these reasons we have decided to explore monolithic n-m crossbar switches more fully as a basic building blocks for ICN's. This is in contrast to other proposals for reduced complexity networks (for example [CiS81]). The monolithic n-m crossbar switch is a single chip IC capable of connecting n input ports (typically a port is a few pins) to m output ports. This report details a study into the design of an n-m crossbar. A probabilistic analysis of an n-m crossbar that derives figures of merit such as expected bandwidth can be found in [MaM81]. The first of the two designs proposed in this report was carried through to final layout as part of an NMOS VLSI circuit design course offered at the Electrical and Computer Engineering Department of the University of Michigan in the Fall of 1980.

Section 2 describes the general structure and operation of an ICN that could be constructed from either of the two designs. Although the two designs are both crossbars it is assumed that the ICN's constructed from them are

configured as Delta networks since the reduced component complexity argument holds at the IC package level (i.e. outside the IC). The design goal is to provide n-m crossbars that allow ICNs to be constructed with a "reasonable" amount of hardware. "Reasonable" here means that the network is economically implemented using present packages, technologies, and pinout limitations without gross inefficiencies. That is, packages should be used efficiently.

The first design is presented in Section 3. It uses serial address transmission during network setup so that pins may be conserved. The serial addressed design uses only 2 pins per input port and output port, this in turn allows large size crossbar switches to be constructed on a single chip (i.e. large n and m). This makes large ICN's feasible at low package counts. The serial addressed design is suitable for an environment where the uninterrupted connection time between processors and memory modules is large. This includes multiprogrammed multi-user general purpose modular machines.

Section 4 describes the second design. It uses parallel addressing for those applications where machine size is smaller and/or setup speed is important. Obviously, parallel addressing requires more pins per input/output port of the module; so within the same number of pins, smaller but faster modules may be constructed.

Section 5 discusses design options that may be added to the basic designs presented in sections 3 and 4. The additions make the basic design more useful for specific applications. The additions require design modifications to be implemented. The modifications, though, are not great.

Section 6, 7, and 8 discuss space complexity, network setup times, and package counts respectively for the two designs. Section 9 is the conclusion.

The Appendix shows several example ICN's constructed from various n-m crossbars. They are Delta networks.

2. Architecture of the ICN.

This section discusses, in general terms, the structure and operation of an ICN composed of our proposed n-m crossbars. A typical application is shown in the multiprocessor architecture of Figure 1. Each processor (labeled Proc. in the figure) is attached to a system ICN that connects N processors to M memory banks. Each processor has its own private memory (PM). This can used to store program and temporary data in MIMD operations, or just temporary data in SIMD operation. The central controller is needed to broadcast instructions in SIMD operation and for system control, I/O, etc.

The processors execute instructions using operands out of the M data memory (DM) banks or modules. In general, each processor may operate on data elements contained in all of the DM's. To read operands from one of the DM modules, a connection from the processor to its desired memory module must be made. Obviously a machine running efficiently will have many such paths constructed or used in parallel.

It was noted in the introduction that to reduce package count our proposed ICN's will be constructed as Delta networks [Pat79]. In common with other reduced complexity networks Delta networks achieve their reduced complexity over full crossbars at the expense of having to make their connections across multiple stages or levels of basic n-m crossbars, i.e. at the expense of time (see the diagrams in the Appendix). With this in mind two different modes of ICN operation are possible: packet switching; and circuit switching.

In packet switching, the network operates by moving *packets* from input to output ports. A packet is a fixed sized item of information and a packet switched ICN serves to move these packets from their source to destination in a stage-to-stage fashion. Note the similarity with pipelining in that many packets may move between consecutive stages simultaneously, so that high transfer rates are possible. Also, with packet switched networks no feedback is

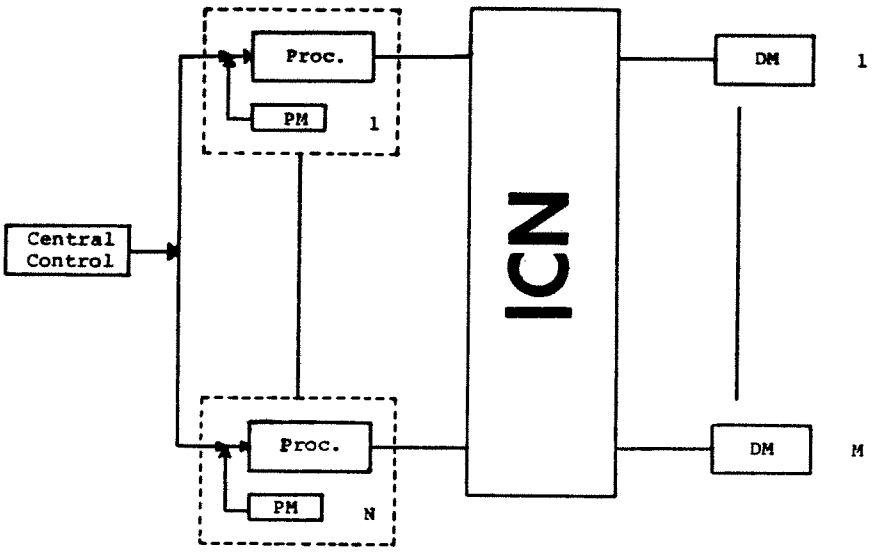


Figure 1. A Multiprocessor Architecture.

associated with packet transmission; processors simply emit packets destined for memory banks, the network and memory modules then take care of the routing and acceptance, and no acknowledge is returned to the processor. In an environment where single memory words are transferred, and lookahead units are employed in processors, packet switching is very efficient. For example, memory reads may be done by having a processor emit a "read" packet which is routed to the proper memory module, the memory module performs the read and then routes the data back through another packet switched network. A lookahead unit can often improve the apparent read time by pre-requesting memory words so that the network has time to respond before the word is actually needed. Under the packet switching mode of operation, the ICN appears to be a flow routing network where the particles of flow are packets.

Circuit switched networks operate differently in that messages are usually variable in length and there is an uninterrupted connection from source to destination after the path is setup. Because of this, circuit switching is most efficient when connections are long in length relative to the setup time. Circuit switching is the mode of operation for the ICN designs described here.

Our proposed multi-stage networks are setup as follows: the first stage module is presented with an output port address, the module then sets the required connection (if it is not busy) for that stage; the remaining addressing information (required for successive stages) is transferred through the connection just established; the process continues through all the stages of the network (typically there are approximately $\log_c N$ stages when c-c crossbar switches are used); at the final stage no address information needs to be transferred after the connection is established, so the path is now complete and the processor is informed that it may begin to use the connection. From this it may be seen that the network is *circuit switched.*

The above process may be inefficient if it is required for every data access or every instruction access, in that much time is spent setting up the network, especially if the network has many stages. In order to aid in relieving this setup inefficiency, it is desirable to use a connection for long periods of time relative to its setup time. As connection time increases, so does the connection efficiency, which may be defined as:

$$CE = \frac{T_{use}}{T_{use} + T_{setup}}.$$

Thus a high CE means that little time is used to setup a path, relative to its usage time. To make CE high, long connections to a particular memory bank are needed. This can be accomplished if the instructions and data that are referenced closely in time are constraint to a minimum number of memory banks. However, if the instructions are spread across memory banks then CE will be low because a new path will have to be setup for each instruction. This is the typical address mapping for interleaved memory. If, though, memory banks are addressed using the upper $\log_2 M$ bits of addresses generated by processors, then instructions and data that are referenced closely in time can more easily be stored in the same bank. This will result in a more efficient use of the ICN (the average value of CE will be close to 1). A further improvement can be made if the PM is used as a cache for program and temporary data. The cache management algorithm should then be organized to update the cache from a single memory bank to avoid setting up new paths during update.

To compare various ways of constructing an ICN, consider using the Delta network to construct a 32x32 system using 2-2 crossbar switches where the

processor/memory connection path requires 64 bits. The network requires 80 2-2 crossbar switches, each 64 bits wide. The amount of logic required could be very large if discrete gates are used. A 32 bit 2-2 crossbar switch could be placed in a 128 pin package (ignoring power supply, clock, etc.), so the total number of packages required for the network would be 160.

This report proposes two alternative partitions of the network from that mentioned above. They are based on the parallel addressed n-m crossbar chip and on the serial addressed n-m crossbar chip respectively. The two approaches can be illustrated using the above example of constructing a 32x32 ICN with a 64 bit wide connection path. First, consider the following: construct an individual 32-32 crossbar with a 1 bit data path on one chip, then stack 64 of them to produce a 64 bit connection path. If a 32-32 crossbar is to be placed on a chip and full parallel routing addresses are to be presented to the chip 32x7 + 32x1 = 256 (5 address, 1 data, 1 request at the input; 1 data at the output) pins are required. Second, consider the following: the 5 bit routing address is sent into 2 pins on a 32-32 crossbar chip (1 pin for address/data and 1 for control) serially in 5 clock cycles. In this case only 32x2x2 = 128 pins are be needed. This scheme also brings the number of chips for the 32x32 ICN down to 64. Furthermore, if connections to memory are long enough, the cost of serial address transmission may be overcome. Note that once a connection is made, data is transferred in parallel words, not serially, and the transfer takes place at propagation speeds.

The network organization for either serial or parallel addressing can be seen in Figure 2. It is easily expanded to arbitrarily large word widths by stacking more planes. The n-m crossbars are directly connectable to make multistage networks. Since word widths are easily varied the network can be designed to deliver multiple words in parallel (a superword) on single memory reads.

Notice that in Figure 2 an extra bit plane is shown labeled "Acknowledge Plane". This 1 bit wide plane is used to signal a connection request acknowledge back to the requesting processor. When a processor needs to establish a connection, it inputs the address of the desired memory (serial or parallel as the case may be) into an n-m crossbar; this is its access port for any of the DM's. When the path inside this crossbar is established by looking at the first $\log_2 m$ bits of the system $\log_2 M$ bit address, the module sends any remaining address bits to the next stage after the connection to the next stage has been made. The last stage, once the connection is made, emits a control signal which the acknowledge plane then propagates backwards through the acknowledge network (note that all planes are set up simultaneously); thus the connection from a data bit plane to the Acknowledge Plane shown in Figure 2. The processor which emitted the request waits until it sees this acknowledge signal before it makes any memory references.

For space efficiency, the n-m crossbar switches have bidirectional data transmission capabilities. The change of direction signaling is done using the control line once a path has been set up (see later). Because conflicts will undoubtedly occur, facilities are incorporated that handle conflict resolution and request waiting in a "clean" manner. Other useful additions to the network's behavior may be found in section 5.

2.1. Interfacing to the Network.

Once the connection has been setup the processor has a parallel connection path to its desired memory bank. That is, a full parallel connection bus of W bits is available for the transfer of data, control, and address information to

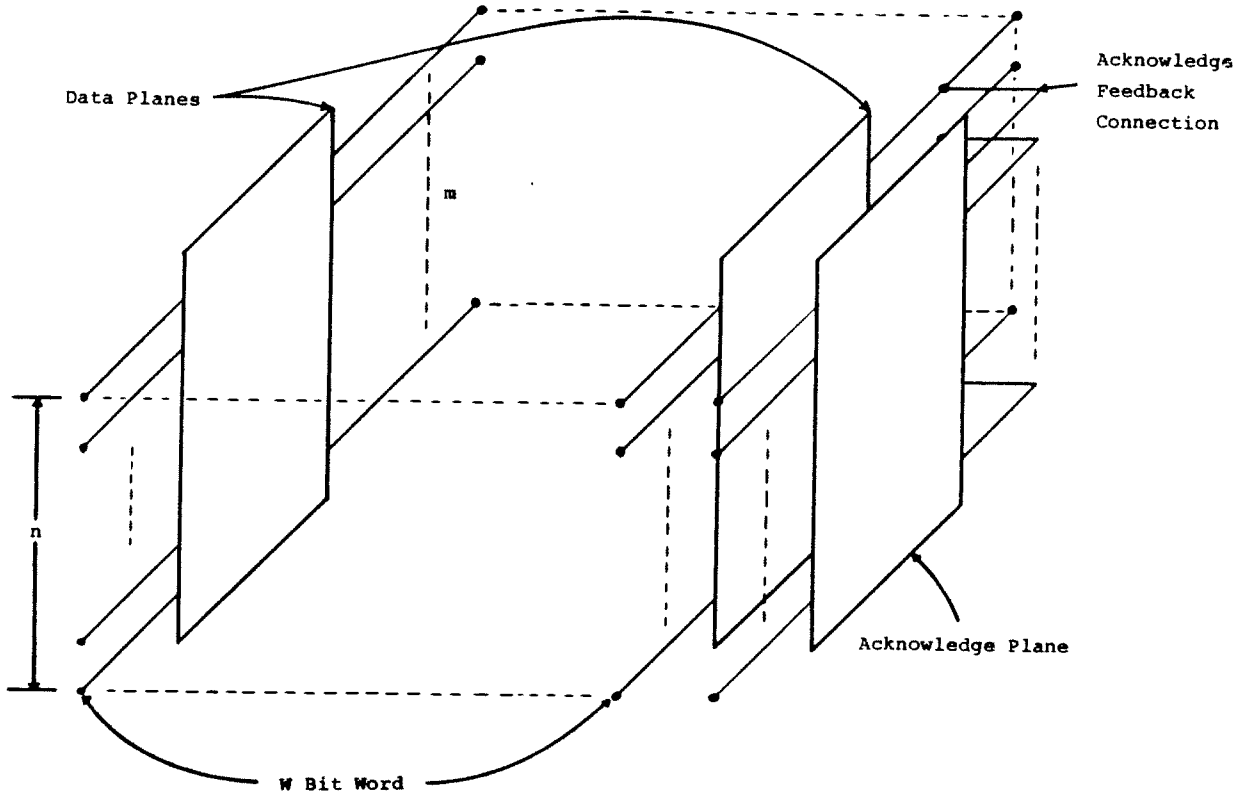


Figure 2. The Network Organization.

specify memory locations within the desired bank. (Note that data and some control lines need to be bidirectional.)

Two possible connection words that go out on the connection bus look like those in Figure 3. These formats provide fast, parallel read/write capabilities.

The parallel addressed connection word is slightly different but will be discussed later. However, in any connection word the basic components still look like Figure 3. Furthermore, it may be desirable to include additional bus lines to allow for error detection/correction of the bit patterns that are transferred on the bus.

The following sections of this report detail the design of the serial addressed crossbar and the parallel addressed crossbar.

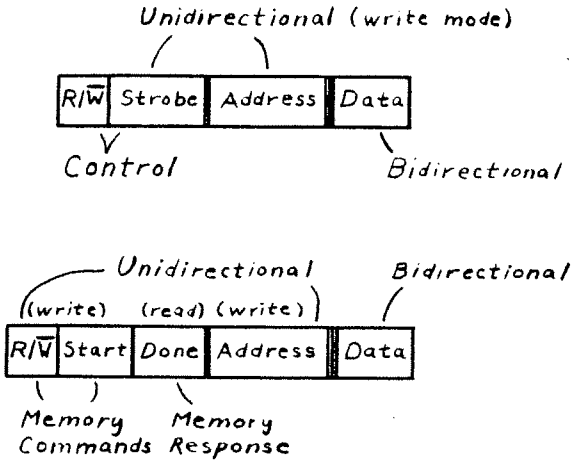


Figure 3. Two Connection Word Formats.

3. Structure and Design of the Serial Addressed n-m Crossbar.

This section describes the architecture of a serial addressed n-m crossbar. The description applies to a basic non-interruptible design. Modifications are needed to implement the useful features of section 5 but complicate the basic architecture, so for simplicity consider a simple n-m crossbar. This design could be used for applications where pin count is highly constrained and/or where the size of the switching module must be large. Serial addressing may also be appropriate where the network runs faster than processors.

Figure 4 shows a block diagram of the basic n-m connector. The figure also indicates the floor-plan of the chip as it could be laid out in NMOS technology. There are n identical rows each of which consists of:

1) a Primary Register.

2) a Secondary Register.

3) a Finite State Machine (FSM) row controller.

4) m Row-Column connection cells.

These four components will be discussed individually along with their interactions.

3.1. The Primary Register.

This register of length $\log_2 m$ bits is used to hold the first $\log_2 m$ bits of the incoming address stream. The bit serial address stream is distinguished from

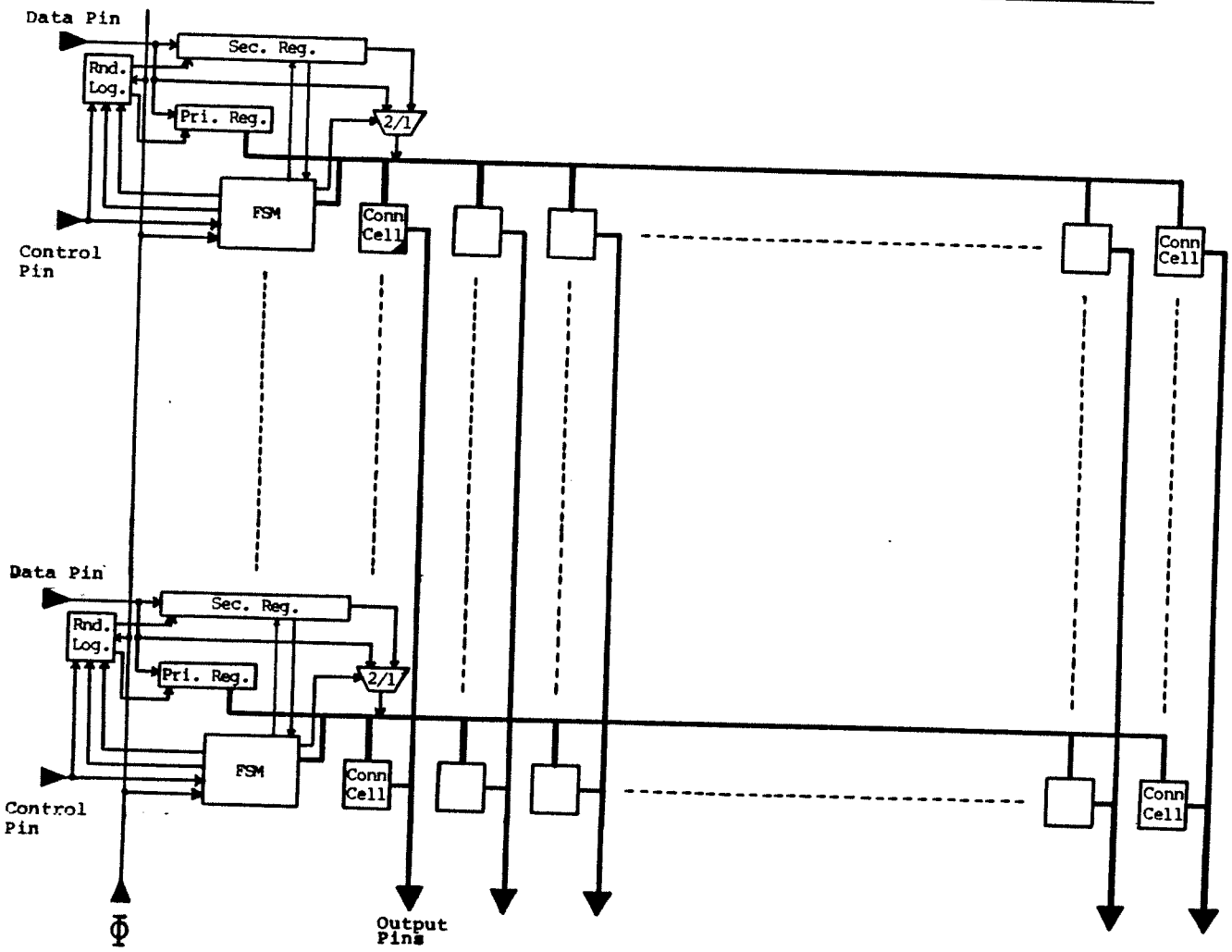


Figure 4. The n-m Serial Addressed Connector Architecture.

data by the CONTROL pin associated with the input port being active. Thus the activation of the CONTROL pin indicates the onset of an incoming address stream. The stream enters serially and after $\log_2 m$ clock cycles the desired output port address is available, in parallel, in this register. That is, this register contains the address of the output port selected for connection. This convention requires that the output port address be sent into the system network most-significant-bit first. Since the first bits streamed into the chip are those used for output port selection, the selection/connection process may run concurrently with the loading of next stage address bits into the Secondary Register (see below).

The Primary Register is controlled by the outputs of the random logic block (Rnd. Log.). The inputs to the random logic are: a control input signal, S1, from the row FSM; the input port CONTROL signal; and the master clock.

When the CONTROL signal and the FSM S1 signal are active, the Primary Register shifts right 1 bit on each clock cycle, shifting in address bits that arrive at the input port DATA pin.

The combination of input port pins (DATA,CONTROL) actually represents a 4 state logic variable and if (or when) 4 valued logic is available these two pins could be compressed into 1 to give twice the module size capability for the same number of pins.

3.2. The Secondary Register.

The Secondary Register is used to hold address bits streaming into the chip that are destined for successive stages. This register will begin to fill after the Primary Register has been filled, assuming more address bits are being sent in for the next stages. When the row-column connection has been set-up by the row FSM, address bits held in the Secondary Register are sent to the successive stages of the network. When this stream is initiated the Secondary Register is either simply being emptied toward the output port or it is being emptied toward the output port while simultaneously being filled from the input port. The former situation occurs when the remaining system memory bank address has been shifted into the chip before a connection could be made. The latter situation occurs when a connection is made before the remaining system memory bank address has been shifted into the chip.

The Secondary Register is controlled by both the FSM and the small amount of random logic. The Secondary Register comprises an address stream buffer, selector logic, and a pointer register. A logic digram of the Secondary Register may be seen in Figure 5. The pointer register and selector logic are used for extracting bits (reading) from the appropriate bit position in the address stream buffer for transmission to the next stage in the network while simultaneously allowing the loading of the address stream buffer with incoming address bits.

The pointer register has one bit set that points to the bit position in the address stream buffer where the next bit in the outgoing address stream must come from. The pointer register's control signal, DEC, originates from the FSM. When DEC is active, the pointer register shifts its bit pointer 1 bit position left if nothing is being shifted into the address stream buffer's left end. If a shift into the address stream buffer is taking place, then the pointer remains at its present position.

The address stream buffer shift right control signal is derived from the clock, a control line from the FSM called S2, and the CONTROL input line. It indicates the presence of additional incoming address bits at the DATA pin.

A status signal, ZERO, is fed back to the FSM from the pointer register. It indicates that the pointer is at its leftmost bit position. One cycle after ZERO becomes active the FSM knows that the remaining address has been transmitted to the next stage so that now it can proceed with data transmission. Note, it always takes more than 1 clock cycle to request a column and get a connection, therefore if any address was shifted into Secondary Register then the address stream buffer will always contain at least 1 bit before the FSM begins to empty it.

The length of the Secondary Register determines the maximum number of network stages that may be cascaded. If the Secondary Register length is L, then a total system memory bank address of $L + \log_2 m$ bits may be accommodated. If all stages are constructed from the same size crossbars then at most $\left\lfloor \frac{L}{\log_2 m} \right\rfloor + 1$ stages may be used. Nothing prohibits the designer from design-

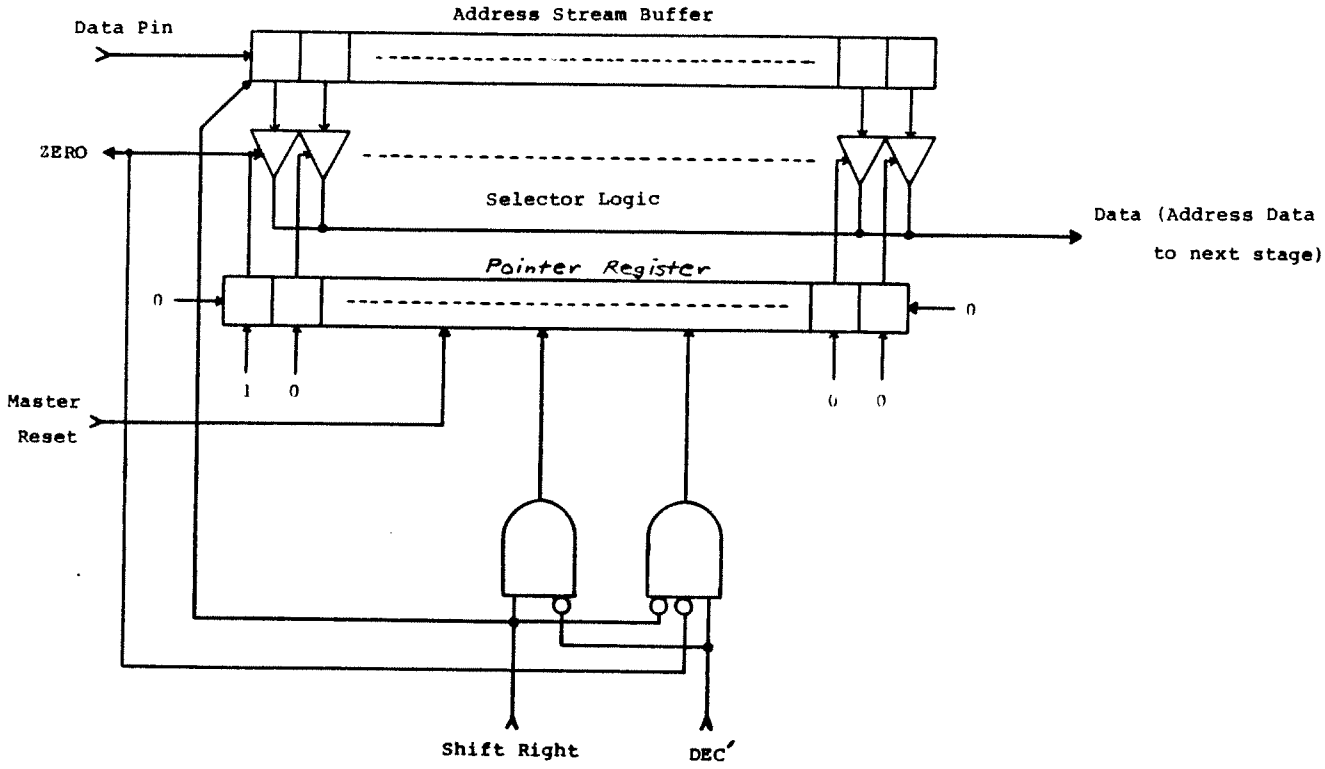


Figure 5. The Secondary Register.

ing a network from dissimilar crossbars (i.e. crossbars with different n-m values). The chip is versatile enough to accommodate all combinations of crossbar sizes as long as their Secondary Register's are capable of buffering the successive stage address bits.

3.3. The FSM.

Each row is controlled by its own FSM. Figure 6 shows the input/output signals for a PLA-based FSM. The FSM controls the handling of the aforementioned registers plus the request/acknowledge operation.

Consider a sequence of operations that creates a connection from an input port (i) to an output port (j). Initially assume that the FSM (and therefore row) i is in an inactive state and an address is going to be sent into input port i, including subsequent stage address bits. Initially, the FSM is idle with its S1 (the control line that affects the primary register shifting) line active. When CONTROL goes active, the primary register begins to shift $\log_2 m$ address bits in, when it is full, the FSM deactivates S1 and activates S2 to begin filling the Secondary Register.

As the Secondary Register is being filled the $\log_2 m$ bit address in the Primary Register propagates down the primary register bus section of the row bus in double railed form, to be decoded by one of the m row-column connection cells. Double railed logic makes decoding simpler and faster in the row-column cells.

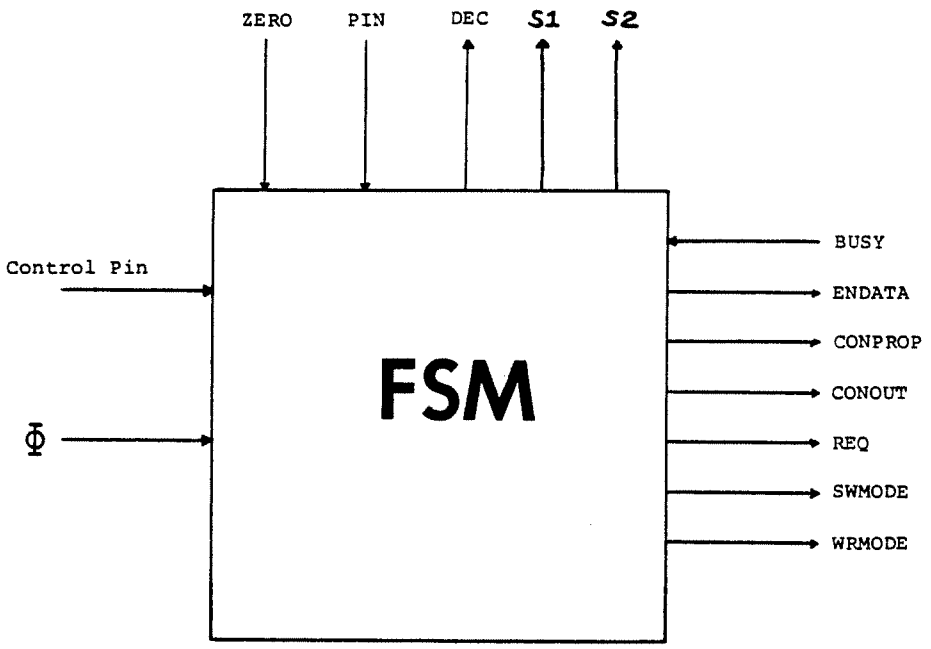


Figure 6. The FSM I/O Signals.

Some of the output signals of the PLA-based FSM are fed into random logic to generate the row signals for the row-column connection cells (see below). The state diagram for the FSM is shown in Figure 7. Note that this particular state diagram is for an n-8 crossbar, hence the initial three states having output S1 = 1.

The random logic used to generate the row signals is shown in Figure 8. This figure also serves to logically relate the signal names from the output of the FSM to those input to the row-column connection cells. This random logic is not shown in Figure 4. It is essentially part of the FSM and was introduced to reduce the size of the FSM.

3.4. The Row-Column Cell.

The row-column connection cells serve to connect column and row signals, that is, they form the crossbar switch connections. They connect the following signals:

1) Row Control signal --> Column Control signal.

2) Row Data signal <--> Column Data signal.

3) Row R/$\overline{W}$ signal --> Column $\overline{R}$/W signal.

4) Row Request signal --> Column Busy signal.

5) Row Busy signal <-- Column Busy signal.

6) Row Priority In signal <-- Column Priority In signal.

Figure 9 shows the signal connection logic. When the row-column connection is made initially only connections 4-6 are made. This is done so that if the column is busy (indicated by BUSY active), the column data stream will not be corrupted.

The signals and their functions are:

Row Control Signal - This signal serves to propagate the CONTROL pin value when the connection has been established. This signal is controlled by the FSM when it is sending successive stage address bits out of the output port. As the FSM is streaming address bits out from the address stream buffer it uses the output port CONTROL line to signal to the next stage that an address bit stream is present. The FSM determines, from the ZERO signal, when the outgoing address stream is finished in order to deactivate the output port CONTROL line.

Row Data Signal - This signal is the data connection path – note that if the chip were designed to handle b bit paths this would be a b bit connection path -- used for the bidirectional propagation of data between the input port and the output port after the connection has been made. This signal is also used during connection setup to propagate the successive address bits to the next stage.

Row R/$\overline{W}$ Signal - This signal indicates the direction of the data propagation. It must be connected to the input and output port DATA pin drivers to control the direction of data signal propagation. Thus this signal must be connected from the input row to the output column (column $\overline{R}$/ W signal).

Row Request Signal - This signal is activated when the FSM requests control of the output column addressed by the Primary Register. This signal is the basis for the column BUSY signal. A row FSM activates the Row Request signal before and during its use of the column, this allows other row FSM's to test the column BUSY line for the column busy condition. This signal also serves to cutoff the Priority In signal (PIN') of lower priority requesters when simultaneous request conflicts occur. See the description of the Priority In signal (PIN) below.

Row Busy Signal - This signal is the BUSY input to the Row FSM, it is the selected column BUSY signal. Again, the FSM checks the signal to tell whether or not the desired column is busy.

Row Priority In Signal - This is the priority input signal (PIN) to the FSM from the selected column priority chain When this signal is active, the FSM is able to request the column if it is not busy. If multiple requests occur during the same clock cycle only the highest priority FSM will see PIN active and thus be the only FSM to proceed. Another priority structure could be used, but a chain was chosen for simplicity and compactness.

Connect Signal - This signal is used to make the final connection of the signals in 1-3 above. It is generated by the logic in the output of the FSM. The Connect signal guarantees that the present user of signals 1-3 can go ahead without being corrupted.

The FSM interacts with the above signals in the following way. After signals 4-6 are connected and row i's FSM sees BUSY active, then it does not

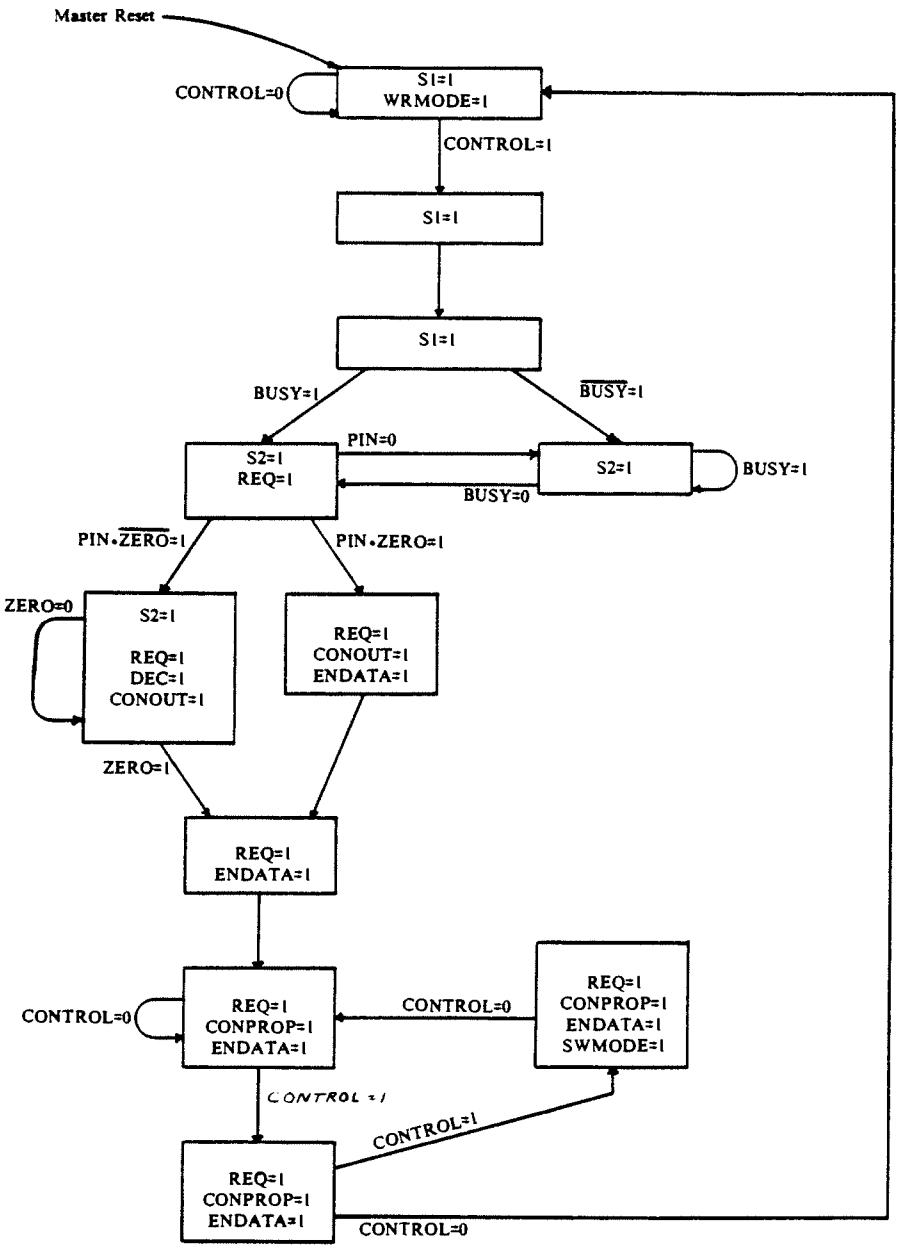


Figure 7. State Diagram of the FSM.

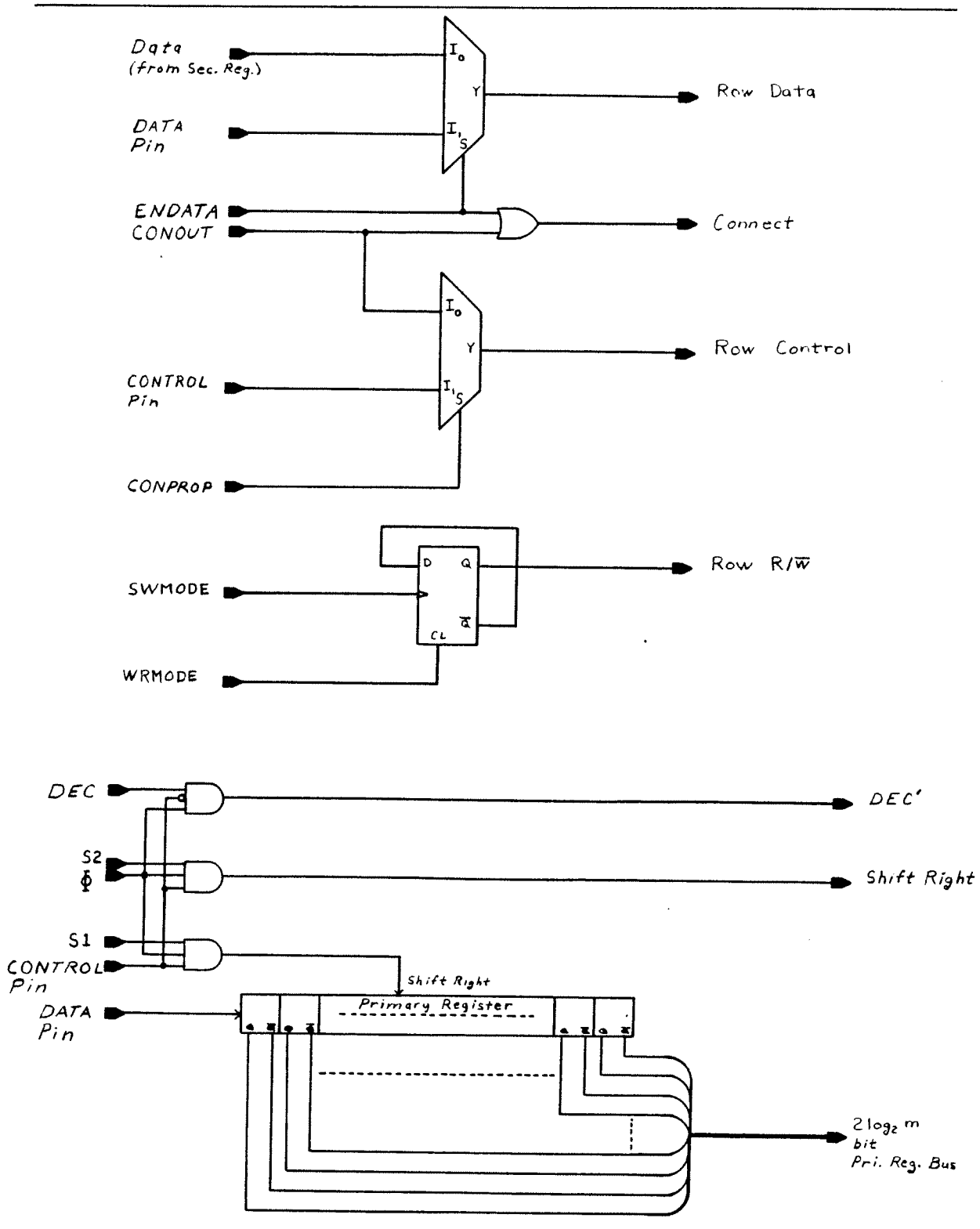


Figure 8. Output Logic of the FSM.

affect any other signals. When BUSY goes inactive (i.e., the present user of the column finishes), all rows desiring the column activate their respective Row Request signals. A column priority chain then indicates to the highest priority FSM, using the Row PIN signal, that it has control of the column. This priority chain is such that the highest priority column-row connection is on the forward diagonal of the array of cells. The trailing end of the chain wraps around the column ends where appropriate. The priority chain is shown in Figure 10. With the diagonal priority organization the identity permutation is favored (if m=n). We chose this simple daisy chain type of priority scheme for simplicity and compactness as mentioned above, however, if the daisy chain proves to be too slow (determined by chip layout and implementation technology) a parallel priority encoder can be used. Implementing this would take more chip space.

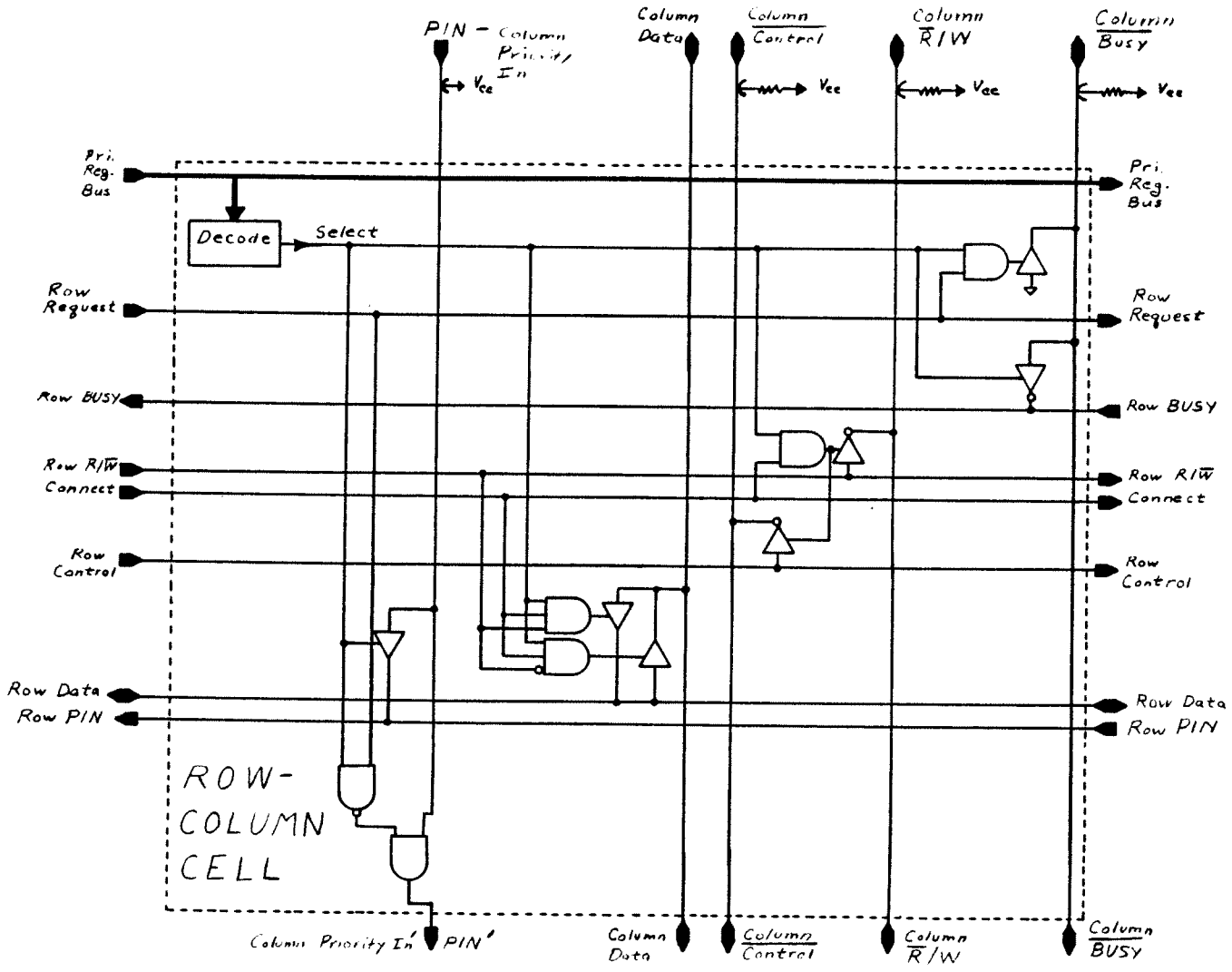


Figure 9. The Row-Column Connection Logic.

The FSM that receives control of the column has control of the Column Data and Control lines. It maintains its assertion of Row Request for as long as the path is established. Once a path is established and the next stage address has been transmitted, the input port CONTROL pin's value is continuously propagated through the Row and Column Control lines to the output port CONTROL pin. Initially the FSM connects the input port DATA pin to the output port DATA pin through the Row and Column Data lines, and pin buffers on the chip are set to transfer data from input to output port. This is the write mode of signal direction.

When the requesting processor gets its acknowledge, the network is in write mode. To change it to read mode the requesting processor activates the CONTROL pin for two clock cycles (see the FSM state diagram, Figure 7), the FSM then toggles the connection to read mode. Repetition of this action at a later time toggles the connection to write mode.

To terminate a connection the requesting processor simply activates the CONTROL pin for one clock cycle then deactivates it for one clock cycle (see the FSM state diagram, Figure 7). Provided it is active long enough for all stages to detect, any protocol will suffice.

In the case of the serial addressed crossbar the acknowledge bit plane of the ICN must relay an acknowledge signal, thus its n-m crossbars must establish a path and then automatically switch to read mode so the acknowledge signal may propagate back immediately. This necessitates that two chip types be designed, however, their only difference is a slight change in the FSM state design, i.e. in the FSM's PLA.

Along with the connection logic, each row-column cell must contain a decoder that detects when its appropriate $\log_2 m$ bit code is on the bus from the Primary Register.

An alternative to distributed decoding is the use of a central $\log_2 m$-to-m line decoder (implemented with a PLA in NMOS). The m decoder output lines would be fanned out to the m row-column cells. This, though, is less space efficient than fanning out the $2\log_2 m$ lines because the rows of row-column cells would each have to be far enough apart so that $(m-2\log_2 m)$ more lines may pass between them. It is better to layout the rows as close together as possible to keep column signal propagations delays as small as possible, as well as to reduce the overall size of the layout.

A serial addressed 16-16 crossbar was designed by us in NMOS. The design was performed as part of an NMOS VLSI circuit design course offered by The Electrical and Computer Engineering Department of the University of Michigan in Fall 1980. The layout contained approximately 25,000 devices.

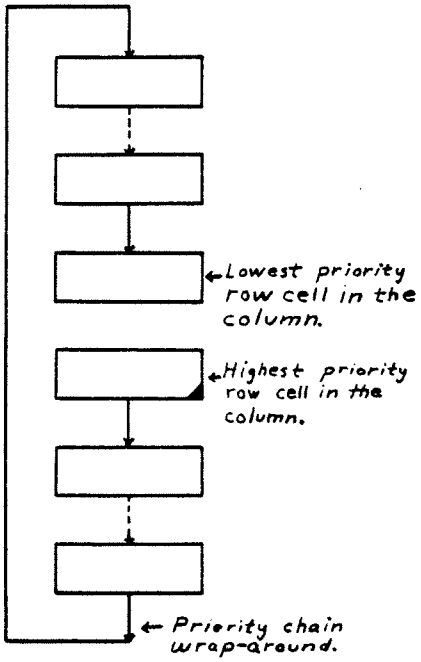


Figure 10. The Priority Chain.

4. Structure and Design of the Parallel Addressed n-m Crossbar.

This section describes the architecture of a parallel addressed n-m crossbar. The operation of the device is similar to the serial address design, but the address bits that select the desired output port are presented in parallel. That is, each input port of the n-m crossbar is presented with the $\log_2 m$ bit address of the desired output port.

Parallel addressing yields a smaller setup time (i.e. improved CE) at the cost of extra pins per port. In the design presented crossbar setup time is two clock cycles during which the path request is transmitted through the crossbar and data stabilizes. Thus crossbar setup time is constant, it is not a function of the crossbar size. Network setup time is equal to $2l-1$ clock cycles, where l is the number of crossbars the signal has to propagate across. An acknowledge must be returned to complete the connection. Each module, once a connection has been made, is capable of bidirectional data transmission for hardware efficiency.

To change read/write modes an input port FSM could be used. It could use a time count on a "control" line (similar to the serial addressed control protocol) or a separate R/$\overline{W}$ pin could be used for each input port. This offers switchover speed advantages and simplifies the row control algorithm. The design described uses a separate R/$\overline{W}$ pin for speed and ease of interfacing to the network. Either approach could be used but parallel control was chosen for speed:

since parallel addressing is used for speed it seemed contradictory to use serial control with parallel addressing.

Due to the parallel addressing, stage address propagation in this design is quite different from the serial addressed design. In parallel addressing processors present a parallel static memory module address to the network. The pertinent first stage crossbar uses an appropriate $\log_2 m$ bit field of this address as its output port address. This $\log_2 m$ bit field is shared by all the ports in each bit plane belonging to the same bus (see Figure 11). Transferring successive stage address bits in parallel is done by treating them as data bits until the stage of the network at which they are needed for addressing. At that stage $\log_2 m$ of the address bits are used, assuming each stage is composed of crossbars with m output ports, and are *not* transmitted further through the network. Thus the number of bit planes of the network (see Figure 11) is reduced by $\log_2 m$ bits at every stage (assuming each stage is composed of crossbars with m output ports).

The port I/O signals are shown in Figure 12. The address/control propagation will be described later. Each input port uses:

$\log_2 m$ address pins,

a $R/\overline{W}$ direction control pin,

a REQIN request control pin,

b bidirectional data pins (b = 1 in Figure 12).

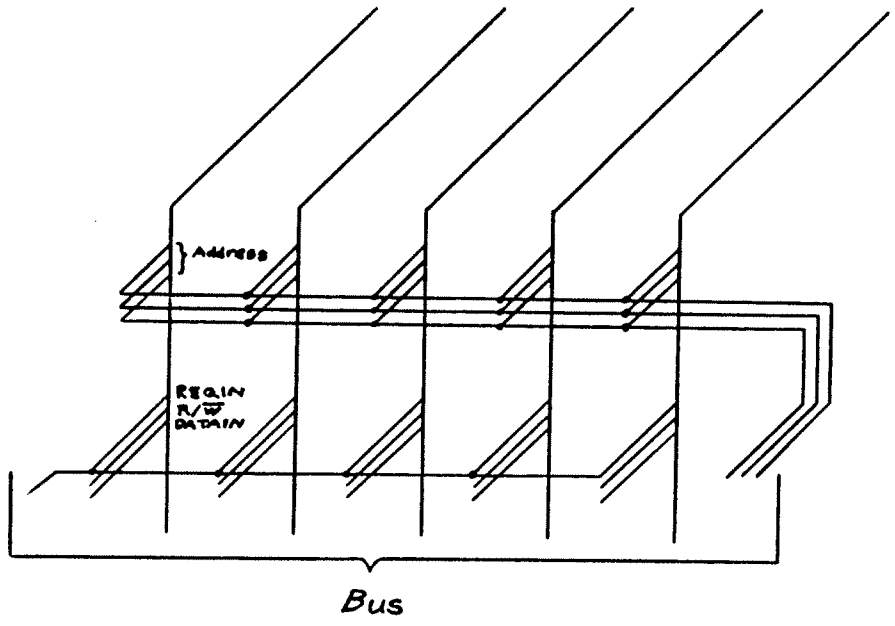


Figure 11. Address Fanout to Each Bit Plane.

The output port uses a single bidirectional data pin.

The REQIN signal is an input used for control. It indicates that a stable address is present on the address lines and a connection is requested. At the first stage REQIN is received from the processor interface. At successive stages REQIN may be obtained from a "data" bit plane in a similar manner to parallel address propagation mentioned above. This approach to request propagation saves a pin on each output port at the expense of higher package count. Although we have chosen not to do so, another output pin might be included with bidirectional capability so that it may communicate information backwards through a bidirectional REQIN pin. Backward communication may be used for stage communications in pipelining the network, or packet switch request/acknowledge operations when a data transfer between stages is to take place.

At the first stage within each bus all bit planes have their address bits connected to the appropriate bits of the memory module address generated by the corresponding bus at the processor interface. Within each bus the interface also generates a request signal which is connected to all bit plane REQIN lines in that bus as it enters the network. The request signal is also connected to a data bit in the "Request" plane. This plane was not needed in the serial addressed crossbar (see Figure 2). Another new bit plane is needed for R/$\overline{W}$ propagation. It operates in a similar manner to the Request plane.

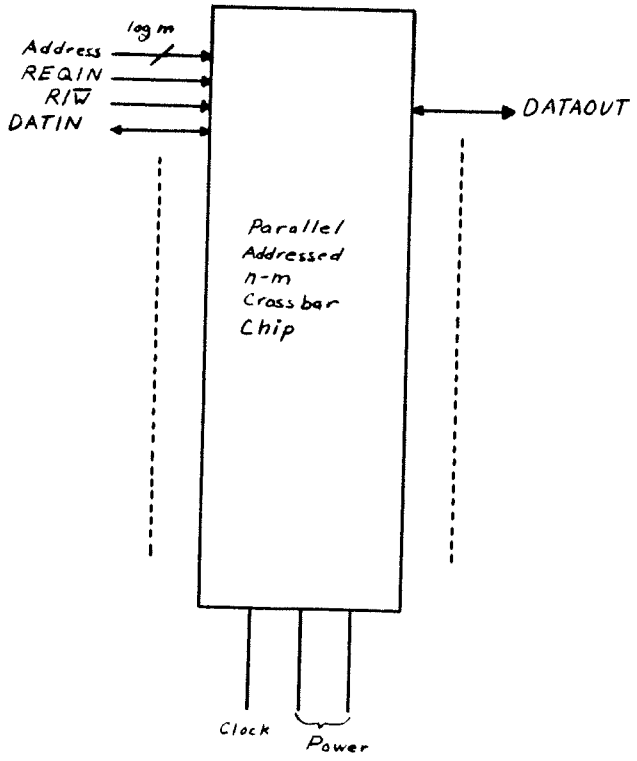


Figure 12. Parallel Addressed Chip I/O Signals.

The remaining discussion applies to all individual interstage bus connections. In other words, if mention is made of a signal fanning out it is to be understood that it is with respect to that bus only. As mentioned above, at any given stage the values present on the DATAOUT lines of the stage that are to be used as the network address information represent a reduced memory module address specification. It is actually the memory module address relative to the present stage of the network. Therefore, the appropriate address lines from this set are connected to all bit plane address lines at the next stage. That is, the DATAOUT line of each appropriate address bit plane is fanned out across the required address inputs of *all* bit planes in the next stage.

The DATAOUT line of the Request plane at the present stage is fanned out to the REQIN lines of *all* bit planes in the next stage. The DATAOUT line of the Request plane is also connected to the DATAIN line of the next stage of the Request plane. The DATAOUT line of the R/$\overline{W}$ plane at the present stage is also fanned out to R/$\overline{W}$ input lines on the next stage. The R/$\overline{W}$ bit is *not* connected to all bit plane R/$\overline{W}$ inputs, it is connected only to those bit planes that are to be bidirectional. In particular, the R/$\overline{W}$ output bit is connected to the DATAIN line of the R/$\overline{W}$ plane at the next stage and the R/$\overline{W}$ control input of the bidirectional data bit planes. The remaining R/$\overline{W}$ inputs are hardwired to the levels discussed below.

All those planes whose data is used as network address information, as well as those planes that carry address information used to access within memory modules are write mode only so their R/$\overline{W}$ pins are wired inactive (write mode). This also applies to the R/$\overline{W}$ and Request planes, they are write mode only. The Acknowledge plane, though, is read mode only, so all modules in this plane would have their R/$\overline{W}$ lines hardwired active. The row-column cell design (see later) provides correct logic levels so that noise-free backward acknowledge propagation may take place. At the last stage the Request plane DATAOUT line is directly connected to the Acknowledge plane DATAOUT line. Since the Request plane is hardwired to write mode, and the Acknowledge plane is hardwired to read mode, when the full network path has been created, the request signal propagates back through the Acknowledge plane to inform the requesting processor that the path has been created.

4.1. The Row Control Structure

The chip design for parallel addressing is similar to the serial addressed design except that new port I/O signals are used and the control algorithm is simpler. Figure 13 shows the row control structure at each input port. Notice that it is simpler than the serial addressed control logic.

The row FSM operates in a similar fashion to the FSM in the serial addressed design. Its state diagram may be found in Figure 14.

The FSM I/O signals and their functions are:

LOAD - This signal controls the loading of the double railed latch that holds the crossbar output port address. This load signal is a level signal (not an edge triggered one). The latch provides the same service that the Primary Register provides in the serial addressed design.

REQ - This signal is identical in operation to the Row Request signal in the serial addressed design.

CONNECT - This signal serves to connect appropriate row-column signals when needed. It is the same as the "Connect" signal in the serial addressed design,

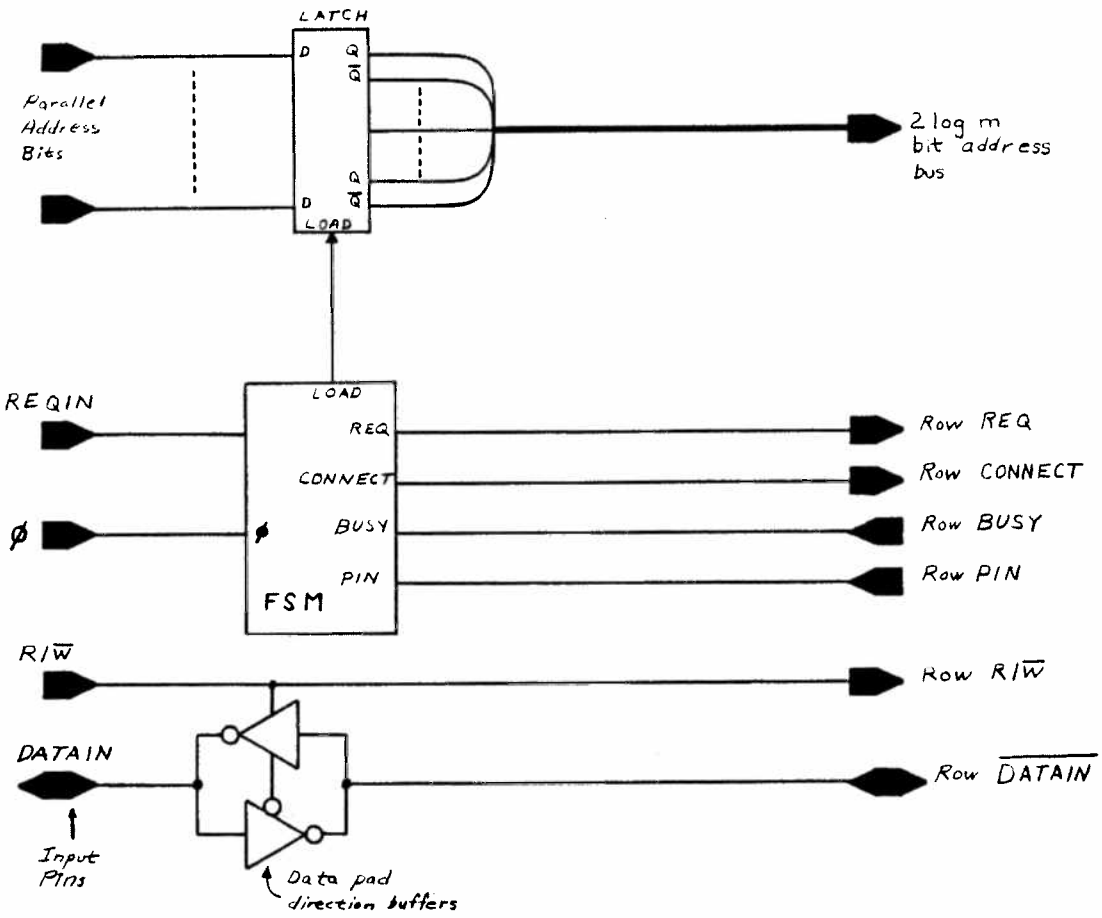


Figure 13. Parallel Addressed Row Control Structure.

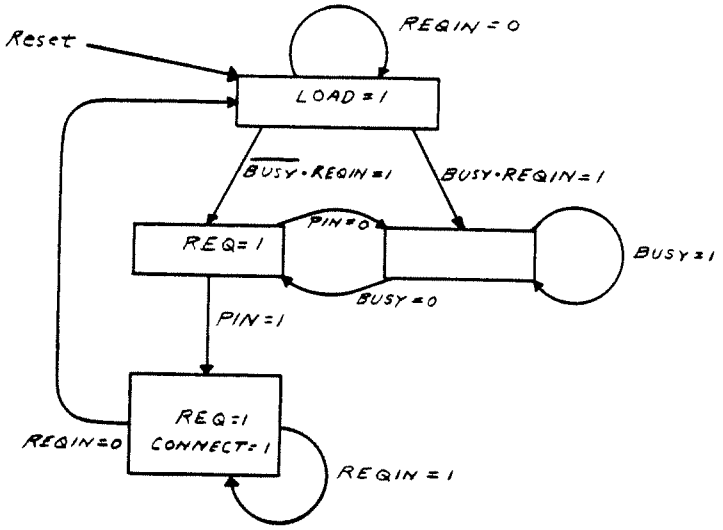


Figure 14. Parallel Addressed FSM State Diagram.

however, the generation of Connect in the serial addressed design is derived from gating logic seen in Figure 8, here it is obtained directly from the row FSM.

BUSY - This signal is identical to the BUSY line in the serial addressed design.

PIN - This priority test signal is identical to the PIN signal of the serial addressed design.

The R/$\overline{W}$ input signal is used to control the direction of the data signal(s), as such it is connected to bidirectional pin buffers at the input and output ports. Therefore, it must be switched through the row-column cell. It must also control data signal flow in the row-column connection logic. This control is accomplished with a simple pass transistor in NMOS.

Inverting buffers are used on the input and output pins so that when an output port is not activated, its DATAOUT line is held Low. This is also true of the DATAIN line when read mode is asserted. This is important for noiseless Request/Acknowledge signal propagation.

4.2. The Parallel Addressed Row-Column Connection Cell

The cell may be seen in Figure 15. It is similar to Figure 9, the only difference is that only one output pin signal is propagated to the output port.

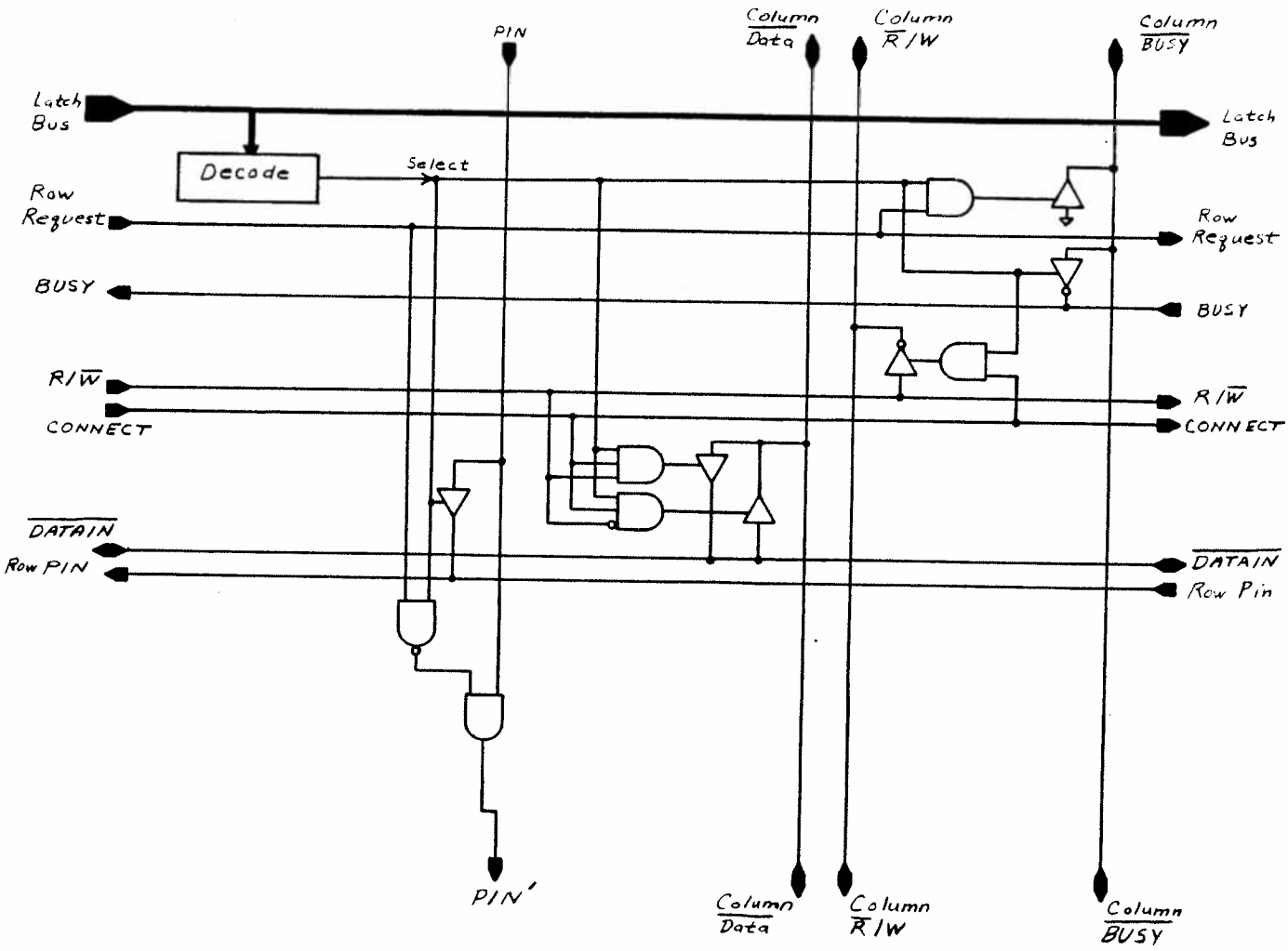


Figure 15. Parallel Address Row-Column Cell.

5. Additions to the Basic n-m Crossbar.

By modifying the FSM it is possible to add "intelligent" features to the basic n-m crossbar. Several useful additions are discussed in this section. The list, though, is by no means complete. The additions to be discussed are:

1) Connection status test.

2) Interrupt/Preempt capabilities with priorities.

3) Connection timeout capabilities.

4) Scalar broadcast capabilities.

5.1. Connection Status Test.

This feature would enable a processor to test the status of any of the M memory bank connections. The processor would essentially "ask" the network if a given memory bank is being used. This would require a different network setup behavior than before. The feature would be useful for general purpose MIMD machines, and an "ICN test" instruction could be included in the processor's instruction set. For example, a low priority process running in one processor may occasionally test to see if a particular memory is free. If it is not, the processor may switch in and run another process for a while before retesting. This avoids having requests pending that block other high priority processes.

The Status Test option is intended for use in general purpose machines where separate processors may be used for running separate operating system processes, or where separate processors may be shared among many users. For example, a pool of central processors may be allocated among several users so that each user gets his/her own CPU. In addition, common system resources such as secondary memory and common peripherals; printers, plotters, etc. can be allocated their own small processor which multiprograms low priority processes (printing, etc.) whose parameter/data blocks may be kept in global memory.

Using the n-m crossbar allows the system designer to design modular general purpose machines in an economical fashion. In such applications the serial addressed crossbar appears to be the logical solution, in that a multiprogrammed multi-user machine can easily be run so that it has the block structured addressing behavior that yields a high CE. For example, each process and its environment can be constrained to use a few memory modules. This also assists the implementation of user security.

5.2. Priority Interrupts.

In a general purpose machine the capability of one processor to interrupt another is of critical importance. For example, a lower priority process may be using a particular memory that a high priority process requires. In this case the high priority process should be able to interrupt the lower priority process' connection. Obviously, this could be done using an additional "interrupt" bus not related to the network, but interrupt facilities may also be implemented in the network.

Priorities could also be imposed on the connection interrupt structure in order to make it more useful. This capability calls for a priority comparator for each row of the n-m connector, its size depends on the number of levels of priority that are desired. A priority bus would be needed for each row and column, thus the row-column cells will increase in size. Dynamic priority structures could also be accommodated.

Interrupts are difficult to implement in the network unless a control protocol is developed that ensures that a processor using a connection is informed of a preemption. That is, no processor's connection path should be disturbed without warning beforehand. This requirement demands a more complicated protocol and, as such, use of the network will be more complicated. The silicon area occupied by a crossbar will be further increased due to the added FSM complexity.

5.3. Connection Timeouts.

Although not explicitly designed for use in SIMD machines, the n-m crossbar can be used in such environments. To simplify its use in an SIMD

machine (where the network acts as a permuter with a short connection time, e.g. one operand transfer time) a programmable time counter may be placed at the base of each column bus. The timer would be responsible for monitoring the connection time of the column and then after reaching a limit, disconnecting the row-column connection. This makes the SIMD ICN setup-fetch cycle slightly more efficient because the connections are automatically destroyed after the specified timeout. The specified timeout length could be programmed, if desired, through extra pins on the chip.

Note that in SIMD mode of operation, conflict problems may be ignored for the sake of setup speed. The user is responsible for assuring that no conflicts occur, thus the FSM's job is simpler because it need not check for a busy column condition. In addition, the acknowledge signal can be omitted.

5.4. Scalar Broadcasting.

This addition can be useful in some MIMD algorithms, however is especially useful for SIMD algorithms where the need arises to expand a scalar into a vector [e.g. for row normalization in Gaussian Elimination]. This facility, though, is more difficult to implement than the other additions above.

To broadcast to *all* outputs is straight forward. A special address that does not correspond to any output may be used to signify such a request. However, if it is desired to broadcast to arbitrary outputs, then each output address must be sent into the network individually and the row FSMs must have the capability to "add" connections to already existent ones. Thus each row-column cell needs a "selection" flip-flop that may be set by the row FSM so that many columns may be selected sequentially as the path is built. Establishing an arbitrary broadcast will be time consuming.

A compromise is to specify the output address with possible "don't care" conditions. This requires twice the number of bits to specify an address which is an improvement over the arbitrary broadcast case. It allows some subsets of the output ports to be addressed, specifically any group of outputs whose addresses are adjacent on a Karnaugh map. The price paid for speed is flexibility. Write mode broadcasts require that all columns to be broadcasted to are inactive before the next stage can be setup: the successive stage address must be transmitted synchronously to all desired output columns. The time spent waiting for all columns to clear may be long in an MIMD environment. It is less of a problem in an SIMD environment.

Broadcasts during memory reads are more complicated than broadcasts during memory writes. To perform a memory read broadcast requires that all involved processors establish a connection prior to the actual memory read operation. This implies that a crossbar allow, "taps" to be placed on already existent connections (to tap a read operation, the requesting tap must be in read mode). The final problem to be overcome in read broadcasts is that of initiating the read operation. Only one processor is allowed to perform the operation to avoid electrical conflict problems if the suggested connection word formats are used.

6. Size Complexity of the Design.

Because the constraint on crossbar size is essentially a pin limitation, the silicon area of the network is not a critical issue at this point in time. However, when VLSI technology reaches the point where processors, memories, and crossbars can all be placed on a chip area will start to become an issue.

Let S be the space required for an n-m crossbar design. Then from examination of the logical structure it is seen that:

$$S = Amn + Bn + Cm + D.$$

where,

A is the size complexity of a row-column cell. A is a function of protocol complexity, the connection path width, b, and the decoder which has size proportional to $\log_2 m$.

B is the size complexity for the elements of a row controller, i.e. the FSM, registers, busses, etc. The FSM has size proportional to $\log_2 m$, since, among other things, it must count up to $\log_2 m$. Furthermore, the Primary and Secondary registers also have size proportional to $\log_2 m$ and L respectively (see section 3). However, for small m the $\log_2 m$ dependency of B can be neglected and B can be regarded as a constant.

C is a fundamental size complexity for column elements such as output pin drivers, connection timeout counters, etc.

D is a constant term representing the size of the shared chip elements such as central clock drivers, reset signal drivers, etc.

For small m and n, and b=1, it has been found that B is the dominating coefficient in the above expression and thus the chip size complexity is approximately linear in n. As b increases the Amn term starts to dominate and the size complexity becomes $O(nm\log_2 m)$. Note that as b increases more efficient use is made of the CONTROL pin in that for b=1 half of the port pins are used for control, but as b increases a lower fraction of the port pins are devoted to control. This is even more apparent in parallel addressing where the input port overhead is $\log_2 m + 2$ pins ($\log_2 m$ address pins; 1 request pin; 1 R/$\overline{W}$ pin). If a small connector is required it is more efficient to increase b in each chip as long as the pinout is within the given constraint.

The pin number for the serial addressed design may be expressed as:

$$P = \left(b + 1\right)\left(m + n\right) + k \ ,$$

where k is a constant representing connections for power, clock, etc. Two level logic is assumed.

For the parallel addressed design, P is:

$$P = \left(b + \log_2 m + 2\right)n + bm + k \ .$$

7. Network Setup Times.

This section discusses network setup times for the serial addressed and parallel addressed designs. Setup times are given in clock cycles. Throughout the discussion assume no conflicts occur.

As mentioned previously setup time for a parallel addressed crossbar is constant and equal to 2 clock cycles. This is the interstage setup times. At the last stage the first clock cycle sets up the connection and after a data settling delay during the second clock cycle the acknowledge signal propagates back through the Acknowledge plane. Thus sometime during the (2l-1)th clock cycle (l is the number of stages in the network) the acknowledge signal will be received by the processor interface. This assumes the propagation back through the Acknowledge plane does not require more than 1 clock cycle.

The setup time for a network constructed from serial addressed crossbars is slightly more complicated. Assume that the network is a Delta network with l stages and where b-b crossbars are used. A timing diagram for l=4, b=16 is shown in Figure 16. From this it may be seen that without blocking the acknowledge signal will be returned to the processor interface sometime during the ($\log_2 M+2l-1$)th clock cycle after the request stream initiation.

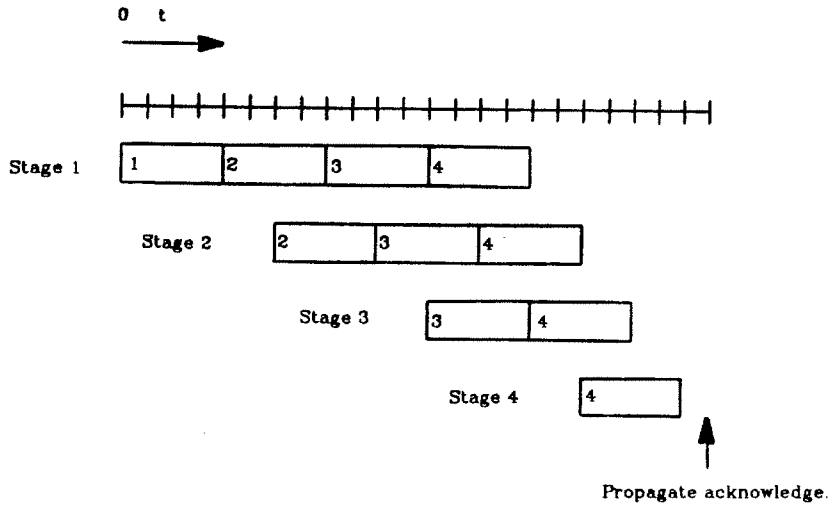


Figure 16. Serial Address Stream Diagram.

8. Network Package Count.

Consider using a Delta network for the topology of an NxN ICN. This section gives a package count for any such Delta network constructed from c-c crossbars.

Let W be the desired memory-processor data path connection width, and assume that $N = c^k$, $c = 2^j$ where j and k are integers. Let P be the number of available pins (not counting power, reset, the clock, etc.).

The number of W data bit wide c-c crossbars required to construct an NxN Delta network is:

$$\frac{N}{c}\log_c N .$$

For the serial addressed crossbar the package count G is:

$$G = \frac{N}{c}\left(\left\lceil\frac{W}{b}\right\rceil + 1\right)\log_c N ,$$

where again b is the data path width per monolithic c-c crossbar. Note the "+1" for the Acknowledge plane.

This can be expressed in terms of package parameters by noting that the number of pins P is $2c\left\lceil b + 1\right\rceil$ so we get:

$$G = \frac{N}{c}\left(\left\lceil 2c\,\frac{W}{P - 2c}\right\rceil + 1\right)\log_c N .$$

For the parallel addressed design where the network width changes at each level, the situation is slightly more complicated. The planes for data and control contribute $\frac{N}{c}\left(\left\lceil\frac{W}{b}\right\rceil + 3\right)\log_c N$ packages. Note the "+3" for the Acknowledge, Request, and $R/\overline{W}$ planes. These planes are assumed to be implemented from components having b=1. The planes for network addressing contribute:

$$\frac{N}{c}\sum_{k=1}^{\log_c N - 1} k \log_2 c$$

packages. Since $\log_2 c$ bits are used per stage and they need not be routed past the stage at which they are used. Furthermore, there are $\log_c N$ stages. As with the control signals, crossbars having b=1 are assumed. Therefore, for the parallel addressed design it takes:

$$G = \frac{N}{c}\left(\left\lceil\frac{W}{b}\right\rceil + 3\right)\log_c N + \frac{N \log_2 c \log_c N \left\lceil\log_c N - 1\right\rceil}{2c}$$

$$G = \frac{N}{c}\left\{\left(\left\lceil\frac{W}{b}\right\rceil + 3\right)\log_c N + \frac{\log_2 N \left\lceil\log_c N - 1\right\rceil}{2}\right\}$$

packages assuming b=1 for all bit planes except the data path, i.e. the path that the processor uses after setup.

b may be expressed as:

$$b = \frac{P - \log_2 c - 2}{2}$$

9. Conclusions.

The problem of designing ICN's in VLSI is actually a problem of mapping the network into a feasible implementation structure. For example, for "on-chip" networks, where processors and their ICN are on a single chip, silicon area is an important issue and is a major constraint on the design. But when the network and processors are on different integrated circuits or boards (or chassis for that matter), pin limitations are the major constraint on the network design. Furthermore, in such cases pin constraints actually influence the set of network topologies that are feasible for the given space and cost.

This report described the design of two n-m crossbars; one that uses serial address transmission, and one that uses parallel address transmission.

To make large networks feasible (say 1024x1024) a design for a basic n-m crossbar was presented that exploits VLSI circuit densities and yet uses pins efficiently. In order to make the logic to pin ratio high, and thus exploit VLSI densities, it was decided to transmit routing information through the network in bit serial fashion. This has the advantage of requiring only two, two level logic, pins per connection port of the n-m crossbar. The effect of the resulting long network setup time may be minimized if connections exist for sufficient time to make CE high. Low pin count allows for high density monolithic switches which in turn lends itself toward more package conservative ICN designs.

The parallel addressed design allows the network to be setup more quickly, but the basic parallel addressed n-m crossbar requires more pins per port than the serial addressed design. This pin cost may very well become a less significant problem if VLSI technology development drives corresponding development of large scale packaging techniques.

Both designs presented make high density crossbars feasible. Their size is only limited by pin constraints. For example, it was noted earlier that approximately 25,000 devices were required for a monolithic NMOS 16-16 crossbar that was serially addressed. This is well within current commercial IC densities (consider the 64k RAM), however, a serial addressed crossbar requires a package with >64 pins. The basic design presented is versatile in that many additions may be made to make the n-m crossbar more useful for a particular environment. However, the additions made may require increase setup time.

The designs are oriented toward large multiprocessor that may be used in general or special purpose computing.

10. References

[Ben65]
V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic Press, New York 1965.

[CiS81]
L. Ciminiera, A. Serra, "Modular Interconnection Networks with Asynchronous Control," *Proc. of the 14th Hawaii International Conf. on System Sciences*, 1981, pp. 210-218.

[Clo53]
C. Clos, "A Study of Non-blocking Switching Networks," *The Bell System Technical Journal*, Vol. 32, March 1953, pp. 406-424.

[Fra80]
M. A. Franklin, "VLSI Performance Comparison of Banyan and Crossbar Connection Networks," in [Sie80], pp. 20-28.

[Law75]
D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Trans. Computers*, Vol. C-24, No. 18, December 1975, pp. 1145-1155.

[MaM81]
B. A. Makrucki, T. N. Mudge, *Probabilistic Analysis of a Crossbar Switch*, SEL Report No. 150, Department of Electrical and Computer Engineering, University of Michigan, March 1981. [To appear.]

[Pat79]
J. H. Patel, "Processor-Memory Interconnections for Multiprocessors," *Proc. 6-th Annual Symposium on Computer Architecture*, IEEE, April 1979, pp. 168-177.

[Pea77]
M. C. Pease, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Trans. Computers*, Vol. C-26, No. 5, May 1977, pp. 458-473.

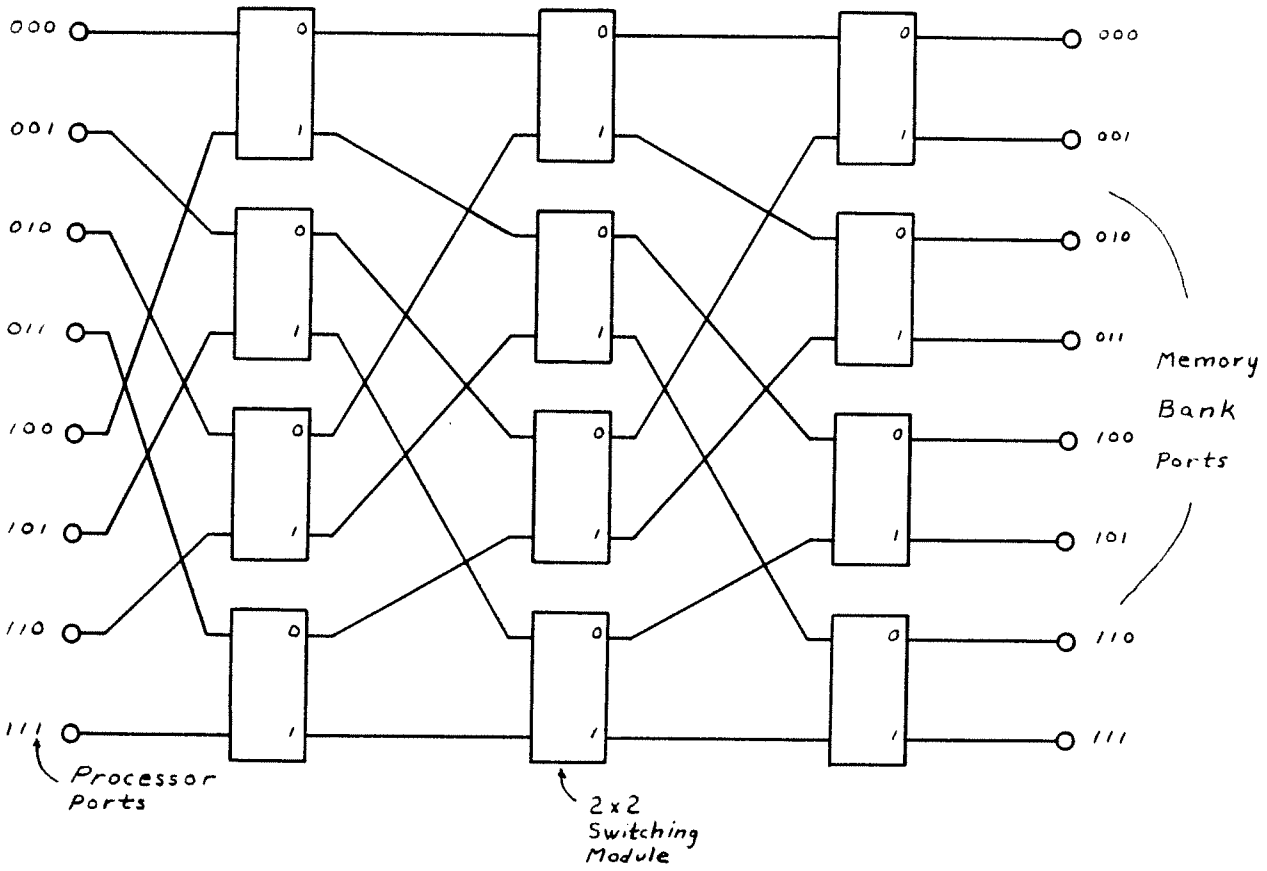
[Sie80]
H. J. Siegel ed., *Proc. Workshop on Interconnection Networks*, Purdue University, April 1980.

[Tho80]
C. D. Thompson, *A Complexity Theory for VLSI*, Ph.D. Thesis, Carnegie-Mellon Univ., Computer Science Dept, August 1980.

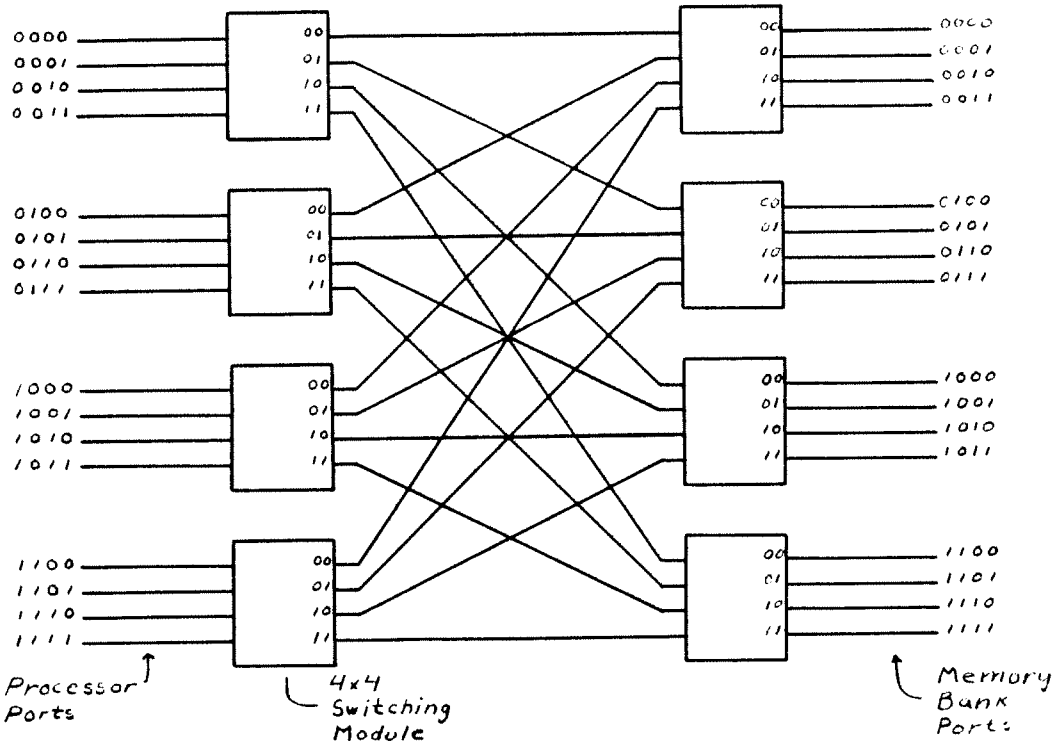
[WuF80]
C-L. Wu, T-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. Computers*, Vol. C-29, No. 8, August 1980, pp. 694-702.

11. Appendix.

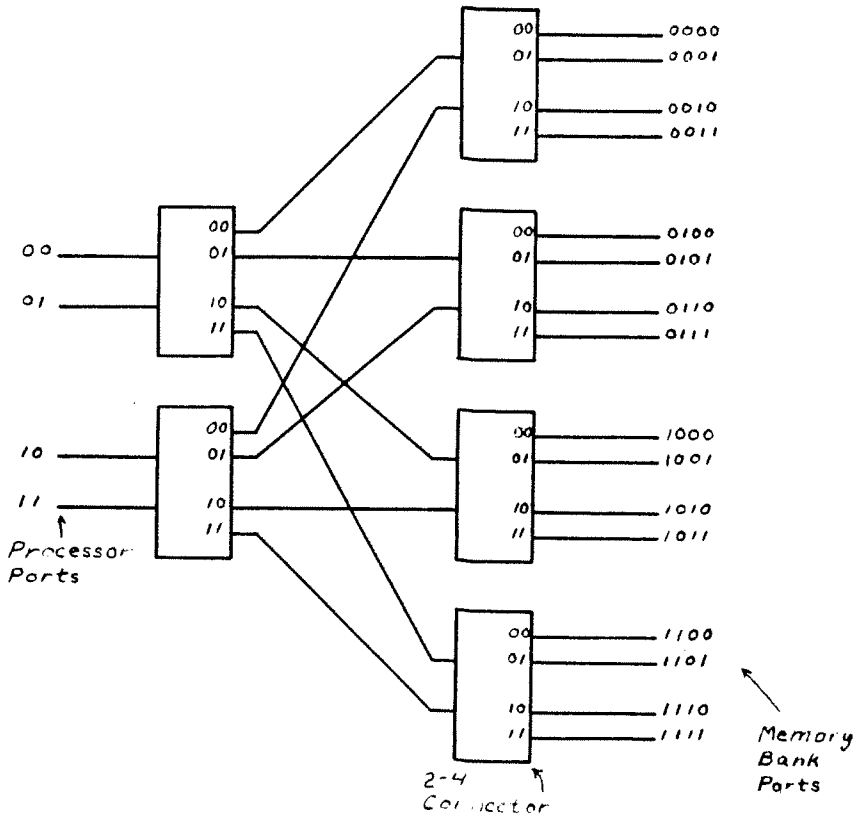
The Appendix contains examples of Delta Networks.



An 8x8 Delta Network.



A 16x16 Delta Network Constructed from 4-4 Crossbars.



A 4 Processor, 16 Memory System Constructed From 2-4 Crossbars.