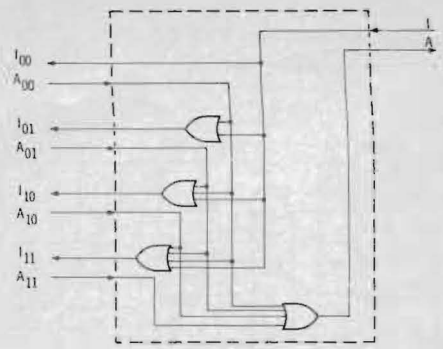


Fig. 8. I-generator and summer block.

Resolver Logic Operation

Determining the number of responders following a search is basically a two-step operation sequenced by the control pulses, E_Y and $E_{Y \cdot I}$:

1) E_Y gates the Y vector out to the $A_0 - A_{192}$ inputs to the tree. A_{out} at the root of the tree then contains the value of ORing Y_i's, y, which is temporarily stored in the combinatorial logic and store block. Simultaneously, the "0" level at the I input to the root of the tree ripples out to the $B_{0,i}$-level block where it is combined with the $A_0 - A_{192}$ inputs to form the I vector.

2) $E_{Y \cdot I}$ gates $Y \cdot I$ over the $A_0 - A_{192}$ inputs to the tree. A_{out} at the root of the tree then contains the value of ORing $(Y \cdot I)_i$'s, which is stored and combined with the value of y as previously described, to determine the number of responders.

CONCLUSION

An optimal framing strategy and an associative processor implementation which can eliminate loss of frame synchronization due to bit errors in the framing pattern and reduce time to reframe following loss of bit count integrity to its minimum value have been described. The technique eliminates the time spent dwelling at false frame positions by simultaneously searching all possible bit positions for the frame pattern. For T1 digital terminals with frame lengths of 193 bits, a period of 125 μs, and a 1 bit frame pattern, the reframe time can be reduced from 49.5 ms to 1.5 ms. The loss in information is thus reduced from 396 frames to 12 frames, providing a significant improvement in system performance.

ACKNOWLEDGMENT

Credit is due to G. A. Anderson for the design of the I-generator tree used for the multiple match resolution logic.

REFERENCES

[1] Technical Staff, Bell Telephone Labs, *Transmission Systems for Communications*, 4th ed. Winston-Salem, NC: Western Electric Co., Tech. Publ., 1970, 759 pp.

[2] L. B. W. Jolley, *Summation of Series*. New York: Dover, 1961.

[3] G. A. Anderson, "Multiple match resolvers: A new design method," *IEEE Trans. Comput.* (Corresp.), vol. C-23, pp. 1317–1320, Dec. 1974.

The Multidimensional Access Memory in STARAN

KENNETH E. BATCHER

***Abstract*—STARAN® has a number of array modules, each with a multidimensional access (MDA) memory. The implementation of this memory with random-access memory (RAM) chips is described. Because data can be accessed in either the word direction or the bit-slice direction, associative processing is possible without the need for costly, custom-made logic-in-memory chips.**

***Index Terms*—Associative processing, corner-turning memories, multidimensional access (MDA) memories, parallel processing, solid-state memories.**

INTRODUCTION

With the current interest in storage schemes and memory-to-processing element interconnections [1], [2], it is timely to describe the multidimensional access (MDA) memory used in the STARAN associative array processor. Conventional random-access memory (RAM) integrated-circuit chips are used in a novel manner to construct a memory that provides access to data in a number of different ways (by words, by bit slices, by bytes, etc.). The use of these standard, high-volume, low pin-count memory devices in place of custom LSI devices reduces the cost of associative processing significantly; the reason is that the ratio of the number of bits stored in a chip over the number of pins on the chip can be much higher.

NEED FOR MULTIDIMENSIONAL ACCESS

In STARAN, vector arithmetic and search operations are performed using many simple processing elements (PE's) [3]–[5]. Memory data are accessed by bit slices (one bit of many items fetched or stored in one memory cycle), with each bit assigned to a distinct PE. For example, consider the vector addition, $A + B \rightarrow B$, where each vector has 256 ten-bit components. First, the least-significant bit (LSB) of every component of A is fetched with a bit-slice access and sent to the PE's (the ith PE receiving the LSB of the ith component of A). Then the LSB of every B component is fetched with a bit-slice access and sent to the PE's (the ith PE receiving the LSB of the ith component).

In every PE, the two LSB's are added to generate a carry and the LSB of a sum component. The sum bits are sent to the LSB position of every component of B with a bit-slice store (the ith component of B receives the sum bit of the ith PE). These basic steps are repeated for each of the other nine bits of A and B, working from the low-order bit positions to the high-order bit positions. The PE's save the carries of each iteration and add them to the two operand bits of the following iteration. The 256 ten-bit additions are performed in 30 memory cycles: 10 bit-slice fetches from vector A, 10 bit-slice fetches from vector B, and 10 bit-slice stores into vector B.

Like conventional computers, input and output operations in STARAN usually access memory data by items or words (many bits of one item fetched or stored in one memory cycle) since most peripherals transmit data in this fashion. Nonvector (scalar) arithmetic operations on single items also access data by words.

Besides bit-slice accesses and word accesses, other ways of accessing memory data are useful in certain situations. For example, a number of alphanumeric records are stored in memory. Input and output operations can access several contiguous bytes of one record in one memory cycle. Operations searching for special codes in key positions can access one byte of several stored records in one memory cycle. Operations searching for alphabetic data, numeric data, lower case data, etc., anywhere in a record can access one bit of several bytes in one memory cycle.

Thus, there is a need for a memory with a number of access modes allowing data to be accessed by bit slice, by word, etc. Such a memory allows the processing power of the system to be applied either to a few bits of many items for parallel vector operations or to many bits of a few items for conventional scalar operations.

In STARAN, MDA memories are implemented with standard RAM integrated-circuit chips—the same chips used by a number of conventional computers for solid-state storage.

MDA MEMORY CONSTRUCTION

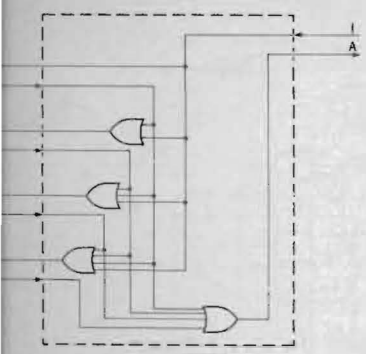
General

The construction of an MDA memory from RAM chips can be illustrated by an example of an MDA memory with 2^n words, each of 2^n bits. The memory uses 2^n RAM chips with 2^n bit locations in each chip. Memory words, RAM chips, word bits, and bit locations in a chip are numbered (addressed) with n-bit binary vectors (n-vectors).

Manuscript received August 22, 1975; revised June 2, 1976.

The author is with the Department of Digital Technology, Goodyear Aerospace Corporation, Akron, OH 44315.

*STARAN is a registered trademark of Goodyear Aerospace Corporation, Akron, OH.



ig. 8. *I*-generator and summer block.

peration

number of responders following a search is basically
n sequenced by the control pulses, E_Y and $E_{Y \cdot I}$:
Y vector out to the $A_0 - A_{192}$ inputs to the tree. A_{out}
ree then contains the value of ORing Y_i's, y, which is
in the combinatorial logic and store block. Simulta-
el at the I input to the root of the tree ripples out to
where it is combined with the $A_0 - A_{192}$ inputs to form

$\cdot I$ over the $A_0 - A_{192}$ inputs to the tree. A_{out} at the
contains the value of ORing $(Y \cdot I)_i$'s, which is stored
the value of y as previously described, to determine
onders.

Conclusion

ing strategy and an associative processor implemen-
minate loss of frame synchronization due to bit errors
ern and reduce time to reframe following loss of bit
s minimum value have been described. The technique
spent dwelling at false frame positions by simulta-
ll possible bit positions for the frame pattern. For T1
ith frame lengths of 193 bits, a period of 125 μs, and
rn, the reframe time can be reduced from 49.5 ms to
information is thus reduced from 396 frames to 12
significant improvement in system performance.

Acknowledgment

G. A. Anderson for the design of the *I*-generator tree
le match resolution logic.

References

Bell Telephone Labs, *Transmission Systems for Communi-*
Winston-Salem, NC: Western Electric Co., Tech. Publ., 1970.

mmation of Series. New York: Dover, 1961.
Multiple match resolvers: A new design method," *IEEE Trans.*
.), vol. C-23, pp. 1317–1320, Dec. 1974.

mensional Access Memory in STARAN

KENNETH E. BATCHER

**RAN® has a number of array modules, each with
al access (MDA) memory. The implementation
ith random-access memory (RAM) chips is de-
scribed. Because data can be accessed in either the word direction or the bit-slice direction, associative processing is possible without the need for costly, custom-made logic-in-memory chips.**

***Index Terms*—Associative processing, corner-turning memories, multidimensional access (MDA) memories, parallel processing, solid-state memories.**

d August 22, 1975; revised June 2, 1976.
he Department of Digital Technology, Goodyear Aerospace
)H 44315.

Introduction

With the current interest in storage schemes and memory-to-processing element interconnections [1], [2], it is timely to describe the multidimensional access (MDA) memory used in the STARAN associative array processor. Conventional random-access memory (RAM) integrated-circuit chips are used in a novel manner to construct a memory that provides access to data in a number of different ways (by words, by bit slices, by bytes, etc.). The use of these standard, high-volume, low pin-count memory devices in place of custom LSI devices reduces the cost of associative processing significantly; the reason is that the ratio of the number of bits stored in a chip over the number of pins on the chip can be much higher.

Need for Multidimensional Access

In STARAN, vector arithmetic and search operations are performed using many simple processing elements (PE's) [3]–[5]. Memory data are accessed by bit slices (one bit of many items fetched or stored in one memory cycle), with each bit assigned to a distinct PE. For example, consider the vector addition, $A + B \rightarrow B$, where each vector has 250 ten-bit components. First, the least-significant bit (LSB) of every component of A is fetched with a bit-slice access and sent to the PE's (the ith PE receiving the LSB of the ith component of A). Then the LSB of every B component is fetched with a bit-slice access and sent to the PE's (the ith PE receiving the LSB of the ith component).

In every PE, the two LSB's are added to generate a carry and the LSB of a sum component. The sum bits are sent to the LSB position of every component of B with a bit-slice store (the ith component of B receives the sum bit of the ith PE). These basic steps are repeated for each of the other nine bits of A and B, working from the low-order bit positions to the high-order bit positions. The PE's save the carries of each iteration and add them to the two operand bits of the following iteration. The 250 ten-bit additions are performed in 30 memory cycles: 10 bit-slice fetches from vector A, 10 bit-slice fetches from vector B, and 10 bit-slice stores into vector B.

Like conventional computers, input and output operations in STARAN usually access memory data by items or words (many bits of one item fetched or stored in one memory cycle) since most peripherals transmit data in this fashion. Nonvector (scalar) arithmetic operations on single items also access data by words.

Besides bit-slice accesses and word accesses, other ways of accessing memory data are useful in certain situations. For example, a number of alphanumeric records are stored in memory. Input and output operations can access several contiguous bytes of one record in one memory cycle. Operations searching for special codes in key positions can access one byte of several stored records in one memory cycle. Operations searching for alphabetic data, numeric data, lower case data, etc., anywhere in a record, can access one bit of several bytes in one memory cycle.

Thus, there is a need for a memory with a number of access modes allowing data to be accessed by bit slice, by word, etc. Such a memory allows the processing power of the system to be applied either to a few bits of many items for parallel vector operations or to many bits of a few items for conventional scalar operations.

In STARAN, MDA memories are implemented with standard RAM integrated-circuit chips—the same chips used by a number of conventional computers for solid-state storage.

MDA Memory Construction

General

The construction of an MDA memory from RAM chips can be illustrated by an example of an MDA memory with 2^n words, each of 2^n bits. The memory uses 2^n RAM chips with 2^n bit locations in each chip. Memory words, RAM chips, word bits, and bit locations in a chip are numbered (addressed) with n-bit binary vectors (n-vectors).

Notation

If X and Y are two n-vectors, then $\overline{X}$
X (each component bit is replaced by
cates the logical product (AND) of X
ANDed with the corresponding compo
the EXCLUSIVE-OR of X and Y (a com
if the corresponding components of X

Storage Pattern

The MDA memory stores data in a
cording to the following rule.

Storage Rule: For any two n-vecto
stored in bit-location B of memory c
two-n-vectors, B and C, bit-location B
word $B \oplus C$.

Fig. 1 illustrates the storage rule for
i is stored in bit-location i of all eight
0 and 7 occupy the main forward and
array, and the other six words occupy th
of certain subarrays. Each bit-slice an
eight memory chips. This scrambling
various ways.

Access Stencils

To access data in the MDA memory,
stencil is specified. A word stencil cov
positioned over any memory word. A
(one bit of each word) and can be po
stencils cover some bits of some words
bits of certain words.

Every stencil covers exactly 2^n bits
fetched or stored in one memory cycl

In general, there are 2^n possible ste
positioned in 2^n different places. Th
n-vector called the access mode, and
vector called the global address. When
are specified, the address bus structu
a local address for each memory chip

Local Addressing Rule: For any th
the specified access mode and G is the
local address fed to the address pins

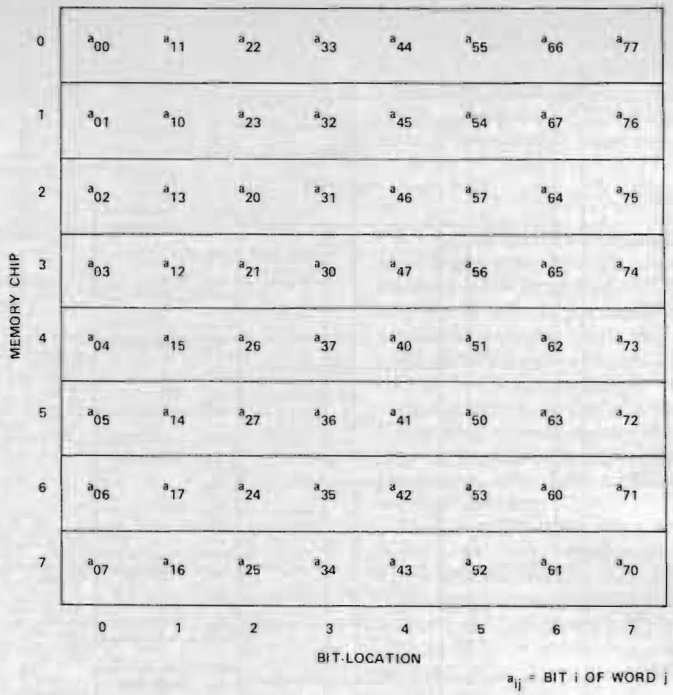


Fig. 1. Storage rule for an 8 × 8 memory.

Notation

If X and Y are two n-vectors, then $\overline{X}$ indicates the logical negation of X (each component bit is replaced by its one's complement), XY indicates the logical product (AND) of X and Y (each component of X is ANDed with the corresponding component of Y), and $X \oplus Y$ indicates the EXCLUSIVE-OR of X and Y (a component of $X \oplus Y$ is 1 if and only if the corresponding components of X and Y have opposite values).

Storage Pattern

The MDA memory stores data in a particular scrambled pattern according to the following rule.

Storage Rule: For any two n-vectors, B and W, bit B of word W is stored in bit-location B of memory chip $B \oplus W$. Conversely, for any two-n-vectors, B and C, bit-location B of memory chip C stores bit B of word $B \oplus C$.

Fig. 1 illustrates the storage rule for an 8 × 8 MDA memory. Bit-slice i is stored in bit-location i of all eight memory chips ($0 \leq i \leq 7$). Words 0 and 7 occupy the main forward and backward diagonals of the square array, and the other six words occupy the forward and backward diagonals of certain subarrays. Each bit-slice and each word are spread across the eight memory chips. This scrambling allows the data to be accessed in various ways.

Access Stencils

To access data in the MDA memory, the shape and position of an access stencil is specified. A word stencil covers all bits of one word and can be positioned over any memory word. A bit-slice stencil covers a bit-slice (one bit of each word) and can be positioned over any bit slice. Other stencils cover some bits of some words and can be positioned over certain bits of certain words.

Every stencil covers exactly 2^n bits of storage. All covered bits can be fetched or stored in one memory cycle.

In general, there are 2^n possible stencil shapes, and each shape can be positioned in 2^n different places. The stencil shape is specified by an n-vector called the access mode, and its position is specified by an n-vector called the global address. When an access mode and global address are specified, the address bus structure of the MDA memory generates a local address for each memory chip according to the following rule.

Local Addressing Rule: For any three n-vectors, M, G and C, if M is the specified access mode and G is the specified global address, then the local address fed to the address pins of memory chip C is $G \oplus (MC)$.

From the storage rule, bit-location $G \oplus (MC)$ of memory chip C c
tains bit $G \oplus (MC)$ of memory word $G \oplus (MC) \oplus C = G \oplus (\overline{M}C)$.

Let g_k be the kth bit of global address vector G, and m_k be the kth
of access mode vector M.

The address bus structure has $2n$ address lines (as opposed to n addr
lines in the conventional RAM). The address lines are labelled x_1,y_1,
$\cdots,x_n,y_n$. Address line x_k is driven by g_k, and address line y_k is driv
by $g_k \oplus m_k$. Address pin k of memory chip $C = (c_1c_2 \cdots c_n)$ is connect
either to x_k if $c_k = 0$ or to y_k if $c_k = 1$. Address line x_k feeds half of t
memory chips, while y_k feeds the remaining half.

This structure implements the local addressing rule since for k
1,2, $\cdots$,n, then $g_k \oplus m_kc_k = g_k$ if $c_k = 0$, and $g_k \oplus m_kc_k = g_k \oplus m_k$ if
= 1. All 2^n local addresses required by the memory chips are comput
from just n EXCLUSIVE-OR logic circuits. Fig. 2 illustrates the addr
bus structure for an 8 × 8 MDA memory.

Scrambling and Unscrambling

Because of the scrambled pattern of stored data, the data fetched fr
the MDA memory are scrambled and, hence, must be unscrambled bef
being read by the processing elements (or the input/output channe
Similarly, data to be stored in memory must be scrambled before be
fed to the data-input pins of the RAM chips. The processing elemer
the input lines, and the output lines are numbered with n-vectors.

Scramble/Unscramble Rule: For any two n-vectors, G and P, wh
storing data into memory under global address G (and any access moc
the data bit from processing element P (or input line P) is applied to
data-input pin of memory chip $G \oplus P$. Conversely, for any two n-vect
G and C, when fetching memory data under global address G (and a
access mode), the data bit on the data-output pin of memory chip C
sent to processing element $G \oplus C$ (or output line $G \oplus C$).

Because of the similarity between the scramble and unscramble ru
and the fact that memory fetches and memory stores never occur
multaneously, one scramble/unscramble network suffices for both
erations. Fig. 3 shows how the scramble/unscramble network fits betwe
the MDA memory elements and the processing elements.

The scramble/unscramble network can be constructed using seve
levels of data-selector integrated circuits. With two-input selectors, th
are n levels of logic with 2^n selectors in each level. The first level tre
bit g_n of the global address G; if the selectors are numbered with n-v
tors, then each selector S, selects either input-data-bit S if $g_n = 0$
input-data-bit $S \oplus (00 \cdots 001)$ if $g_n = 1$. The second level of select
treats global address bit g_{n-1}; selector S in this level either selects fir
level output S if $g_{n-1} = 0$ or selects first-level output $S \oplus (0 \cdots 0010$

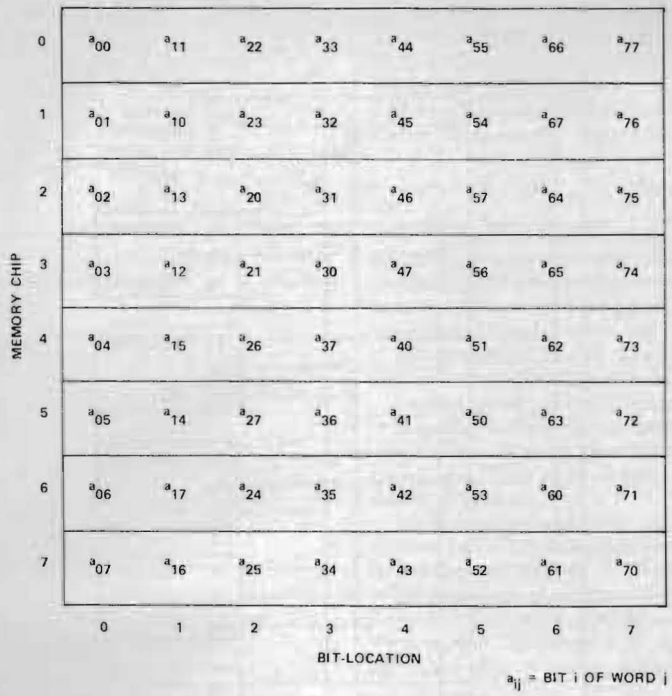


Fig. 1. Storage rule for an 8 × 8 memory.

re two n-vectors, then $\overline{X}$ indicates the logical negation of nent bit is replaced by its one's complement), XY indial product (AND) of X and Y (each component of X is corresponding component of Y), and $X \oplus Y$ indicates -OR of X and Y (a component of $X \oplus Y$ is 1 if and only ding components of X and Y have opposite values).

ern

emory stores data in a particular scrambled pattern acollowing rule.

: For any two n-vectors, B and W, bit B of word W is cation B of memory chip $B \oplus W$. Conversely, for any B and C, bit-location B of memory chip C stores bit B of

tes the storage rule for an 8 × 8 MDA memory. Bit-slice -location i of all eight memory chips ($0 \leq i \leq 7$). Words the main forward and backward diagonals of the square her six words occupy the forward and backward diagonals rays. Each bit-slice and each word are spread across the hips. This scrambling allows the data to be accessed in

ls

in the MDA memory, the shape and position of an access ed. A word stencil covers all bits of one word and can be any memory word. A bit-slice stencil covers a bit-slice word) and can be positioned over any bit slice. Other me bits of some words and can be positioned over certain ords.

covers exactly 2^n bits of storage. All covered bits can be d in one memory cycle.

re are 2^n possible stencil shapes, and each shape can be different places. The stencil shape is specified by an the access mode, and its position is specified by an n-global address. When an access mode and global address e address bus structure of the MDA memory generates or each memory chip according to the following rule.

sing Rule: For any three n-vectors, M, G and C, if M is ss mode and G is the specified global address, then the to the address pins of memory chip C is $G \oplus (MC)$.

From the storage rule, bit-location $G \oplus (MC)$ of memory chip C contains bit $G \oplus (MC)$ of memory word $G \oplus (MC) \oplus C = G \oplus (\overline{M}C)$.

Let g_k be the kth bit of global address vector G, and m_k be the kth bit of access mode vector M.

The address bus structure has $2n$ address lines (as opposed to n address lines in the conventional RAM). The address lines are labelled $x_1,y_1,x_2, \cdots ,x_n,y_n$. Address line x_k is driven by g_k, and address line y_k is driven by $g_k \oplus m_k$. Address pin k of memory chip $C = (c_1c_2 \cdots c_n)$ is connected either to x_k if $c_k = 0$ or to y_k if $c_k = 1$. Address line x_k feeds half of the memory chips, while y_k feeds the remaining half.

This structure implements the local addressing rule since for $k = 1,2, \cdots ,n$, then $g_k \oplus m_kc_k = g_k$ if $c_k = 0$, and $g_k \oplus m_kc_k = g_k \oplus m_k$ if $c_k = 1$. All 2^n local addresses required by the memory chips are computed from just n EXCLUSIVE-OR logic circuits. Fig. 2 illustrates the address bus structure for an 8 × 8 MDA memory.

Scrambling and Unscrambling

Because of the scrambled pattern of stored data, the data fetched from the MDA memory are scrambled and, hence, must be unscrambled before being read by the processing elements (or the input/output channels). Similarly, data to be stored in memory must be scrambled before being fed to the data-input pins of the RAM chips. The processing elements, the input lines, and the output lines are numbered with n-vectors.

Scramble/Unscramble Rule: For any two n-vectors, G and P, when storing data into memory under global address G (and any access mode), the data bit from processing element P (or input line P) is applied to the data-input pin of memory chip $G \oplus P$. Conversely, for any two n-vectors, G and C, when fetching memory data under global address G (and any access mode), the data bit on the data-output pin of memory chip C is sent to processing element $G \oplus C$ (or output line $G \oplus C$).

Because of the similarity between the scramble and unscramble rules and the fact that memory fetches and memory stores never occur simultaneously, one scramble/unscramble network suffices for both operations. Fig. 3 shows how the scramble/unscramble network fits between the MDA memory elements and the processing elements.

The scramble/unscramble network can be constructed using several levels of data-selector integrated circuits. With two-input selectors, there are n levels of logic with 2^n selectors in each level. The first level treats bit g_n of the global address G; if the selectors are numbered with n-vectors, then each selector S, selects either input-data-bit S if $g_n = 0$ or input-data-bit $S \oplus (00 \cdots 001)$ if $g_n = 1$. The second level of selectors treats global address bit g_{n-1}; selector S in this level either selects first-level output S if $g_{n-1} = 0$ or selects first-level output $S \oplus (0 \cdots 0010)$ if

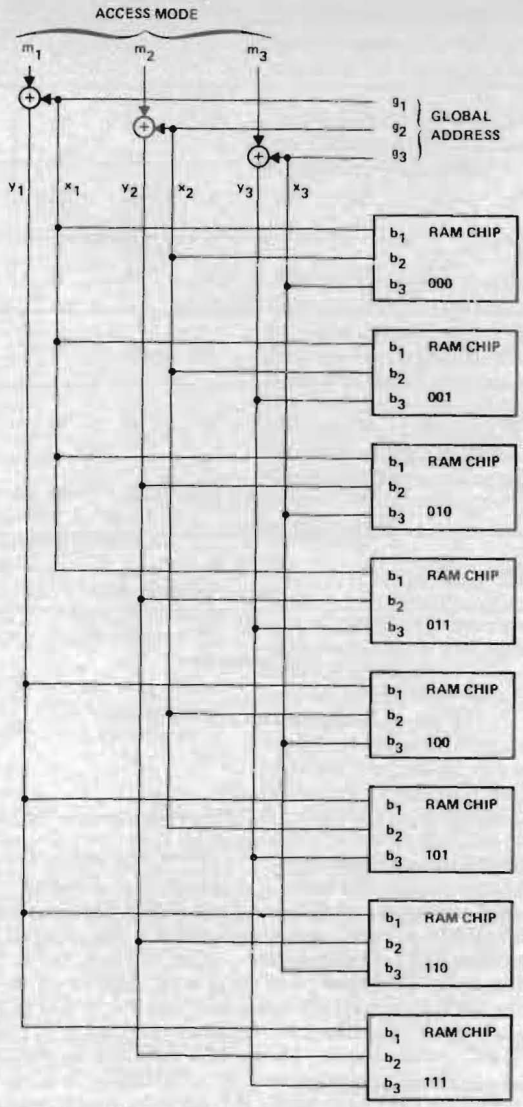


Fig. 2. Address bus structure of 8 × 8 MDA memory.

$g_{n-1} = 1$. Other levels of selectors treat the other bits of the global address in a similar manner. If four-input selectors are used, only $n/2$ levels of selectors are required; each level treats two bits of the global address.

Alternatively, the scramble/unscramble network can be constructed with one level of selection and a perfect shuffle [6]. First, the data are shuffled. Then if g_1 (the first bit of the global address) equals 1, the data in each pair of locations are swapped (data bit S goes to location $S \oplus (00\cdots001)$). Next the data are shuffled again and, if $g_2 = 1$, then data in each pair of locations are swapped. These steps are executed n times—once for each bit of the global address.

Accessing the MDA Memory

General

From the previously stated storage rule, local-addressing rule, and scramble/unscramble rule, one can develop an access rule that shows the memory bit accessed by any particular PE (or input–output line) when the memory is accessed with a given access mode and global address.

Access Rule: For any three n-vectors, M, G, and P, when fetching or storing data under access mode M and global address G, then processing element P (or input–output line P) will access bit $(\overline{M}G) \oplus (MP)$ of memory word $(MG) \oplus (\overline{M}P)$.

For bit-slice access, the access mode M equals $(00\cdots00)$, and the global address G specifies which bit slice is fetched or stored. Processing element P accesses bit G of word P (see Fig. 4).

For word access, the access mode M equals $(11\cdots11)$, and the global address G specifies which word is fetched or stored. Processing element P now accesses bit P of word G.

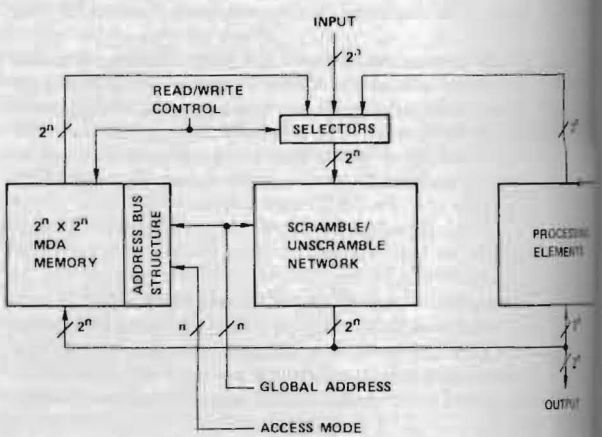


Fig. 3. Scramble/unscramble network connecting MDA memory to processing elements.

If the access mode M is $(00\cdots00111)$ and the global address is $(b_1b_2\cdots b_{n-3}w_{n-2}w_{n-1}w_n)$, then processing element $(p_1p_2\cdots p_n)$ accesses bit $(b_1b_2\cdots b_{n-3}p_{n-2}p_{n-1}p_n)$ of word $(p_1p_2\cdots p_{n-3}w_{n-2}w_{n-1}w_n)$. One 8-bit byte from every eighth word is accessed. The first part of the global address $(b_1b_2\cdots b_{n-3})$ specifies which

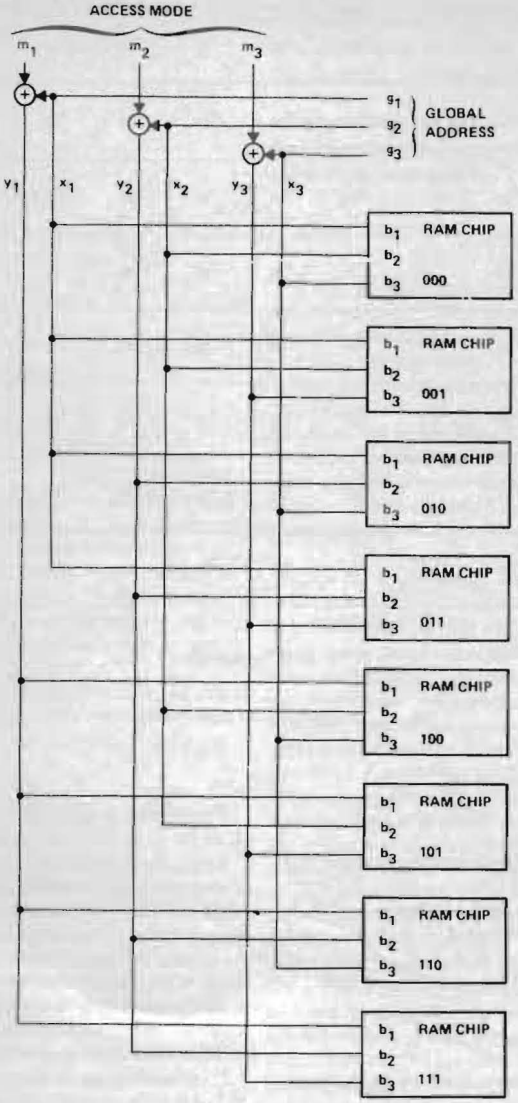


Fig. 2. Address bus structure of 8 × 8 MDA memory.

evels of selectors treat the other bits of the global address
ner. If four-input selectors are used, only $n/2$ levels of
uired; each level treats two bits of the global address.
the scramble/unscramble network can be constructed
selection and a perfect shuffle [6]. First, the data are
g_1 (the first bit of the global address) equals 1, the data
ocations are swapped (data bit S goes to location $S \oplus$
xt the data are shuffled again and, if $g_2 = 1$, then data
ations are swapped. These steps are executed n times—
of the global address.

Accessing the MDA Memory

iously stated storage rule, local-addressing rule, and
nble rule, one can develop an access rule that shows the
sed by any particular PE (or input–output line) when
cessed with a given access mode and global address.
or any three n-vectors, M, G, and P, when fetching or
r access mode M and global address G, then processing
put–output line P) will access bit $(\overline{M}G) \oplus (MP)$ of
$G) \oplus (\overline{M}P)$.
ess, the access mode M equals $(00 \cdots 00)$, and the global
s which bit slice is fetched or stored. Processing element
f word P (see Fig. 4).
, the access mode M equals $(11 \cdots 11)$, and the global

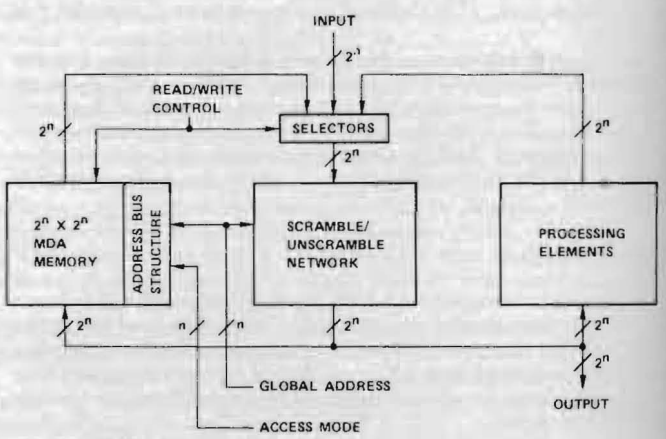


Fig. 3. Scramble/unscramble network connecting MDA memory to processing elements.

address G specifies which word is fetched or stored. Processing element P now accesses bit P of word G.

If the access mode M is $(00 \cdots 00111)$ and the global address G is $(b_1 b_2 \cdots b_{n-3} w_{n-2} w_{n-1} w_n)$, then processing element $(p_1 p_2 \cdots p_{n-1} p_n)$ accesses bit $(b_1 b_2 \cdots b_{n-3} p_{n-2} p_{n-1} p_n)$ of word $(p_1 p_2 \cdots p_{n-3} w_{n-2} w_{n-1} w_n)$. One 8-bit byte from every eighth word is accessed. The first part of the global address $(b_1 b_2 \cdots b_{n-3})$ specifies which of the

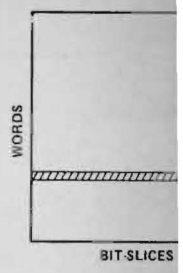


Fig. 4. MDA memory

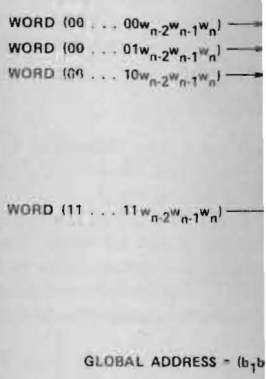


Fig. 5. Byt

2^{n-3} bytes in a word is accessed; th
$(w_{n-2}w_{n-1}w_n)$ specifies the subset
this byte-access stencil. Note that
stored in memory with each record
then the byte-access stencil access
record.

In general, if the access mode has n
covers 2^m bits of each of 2^{n-m} memo
end of the access mode, the 2^m cover
the 2^{n-m} covered words are separate

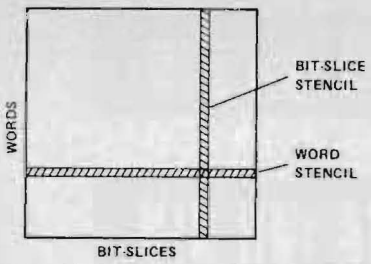


Fig. 4. MDA memory with two-access stencils.

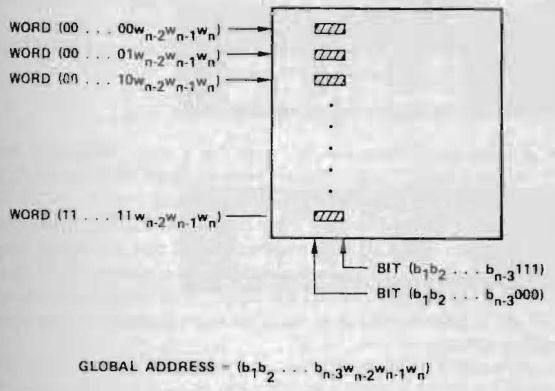


Fig. 5. Byte-access stencil.

2^{n-3} bytes in a word is accessed; the second part of the global address ($w_{n-2}w_{n-1}w_n$) specifies the subset of accessed words. Fig. 5 illustrates this byte-access stencil. Note that if a number of 2^n-byte records are stored in memory with each record occupying eight contiguous words, then the byte-access stencil accesses corresponding bytes from each record.

In general, if the access mode has m 1's and $n - m$ 0's, the access stencil covers 2^m bits of each of 2^{n-m} memory words. If the m 1's are at the right end of the access mode, the 2^m covered bits are in contiguous group, and the 2^{n-m} covered words are separated. Conversely, if the m 1's are at the left end of the access mode, the 2^m covered bits are separated, and the 2^{n-m} covered words form a contiguous group. Where a bit of the access mode is 1, the corresponding bit of the global address is used to position the stencil "vertically" over certain words. Where an access mode bit is 0, the corresponding global address bit is used to position the access stencil "horizontally" over certain bits of words.

Conclusion

By storing data in a scrambled manner, doubling the address lines, and accessing memory through a scramble/unscramble network, one can use standard RAM integrated-circuits to construct a multidimensional access memory. With suitable processing elements, bit-slice access to data allows parallel vector-arithmetic and content-addressable search operations to be executed efficiently. Conventional word access also allows input, output, and scalar arithmetic to use memory efficiently.

An alternative method of achieving MDA would be skewed storage [1]; in this case, skewed storage would require one adder for each memory chip for local address generation. Scrambling the data, as shown here, greatly simplifies local address generation (n EXCLUSIVE-OR circuits versus 2^n adders).

The scramble/unscramble network can be designed to accommodate shifting and other useful permutations of data (such as those required for parallel execution of the fast Fourier transform). It can take the place of the data-routing network required by most parallel-processing architectures and therefore not add a significant cost to the total system. Thus, the cost of MDA storage need not be much higher than the cost of conventional, random-access, solid-state storage.

References

[1] P. Budnik and D. J. Kuck, "The organization and use of parallel memories," *IEEE Trans. Comput.*, vol. C-20, pp. 1566–1569, Dec. 1971.
[2] T. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comput.*, vol. C-23, pp. 309–318, Mar. 1974.
[3] J. D. Feldman and L. C. Fulmer, "RADCAP—An operational parallel processing facility," in *1974 Nat. Comput. Conf., AFIPS Conf. Proc.*, vol. 43, pp. 7–15.
[4] E. W. Davis, "STARAN parallel processor system software," in *1974 Nat. Comput. Conf. AFIPS Conf. Proc.*, vol. 43, pp. 17–22.
[5] K. E. Batcher, "STARAN parallel processor system hardware," in *1974 Nat. Comput. Conf., AFIPS Conf. Proc.*, vol. 43, pp. 405–410.
[6] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.

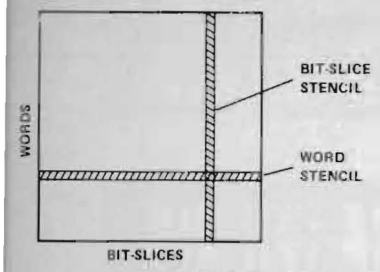


Fig. 4. MDA memory with two-access stencils.

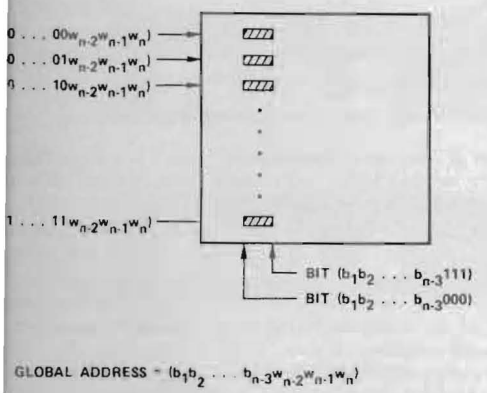


Fig. 5. Byte-access stencil.

a word is accessed; the second part of the global address) specifies the subset of accessed words. Fig. 5 illustrates ss stencil. Note that if a number of 2^n-byte records are ıory with each record occupying eight contiguous words, -access stencil accesses corresponding bytes from each

f the access mode has m 1's and $n - m$ 0's, the access stencil of each of 2^{n-m} memory words. If the m 1's are at the right ess mode, the 2^m covered bits are in contiguous group, and red words are separated. Conversely, if the m 1's are at the left end of the access mode, the 2^m covered bits are separated, and the 2^{n-m} covered words form a contiguous group. Where a bit of the access mode is 1, the corresponding bit of the global address is used to position the stencil "vertically" over certain words. Where an access mode bit is 0, the corresponding global address bit is used to position the access stencil "horizontally" over certain bits of words.

Conclusion

By storing data in a scrambled manner, doubling the address lines, and accessing memory through a scramble/unscramble network, one can use standard RAM integrated-circuits to construct a multidimensional access memory. With suitable processing elements, bit-slice access to data allows parallel vector-arithmetic and content-addressable search operations to be executed efficiently. Conventional word access also allows input, output, and scalar arithmetic to use memory efficiently.

An alternative method of achieving MDA would be skewed storage [1]; in this case, skewed storage would require one adder for each memory chip for local address generation. Scrambling the data, as shown here, greatly simplifies local address generation (n EXCLUSIVE-OR circuits versus 2^n adders).

The scramble/unscramble network can be designed to accommodate shifting and other useful permutations of data (such as those required for parallel execution of the fast Fourier transform). It can take the place of the data-routing network required by most parallel-processing architectures and therefore not add a significant cost to the total system. Thus, the cost of MDA storage need not be much higher than the cost of conventional, random-access, solid-state storage.

References

[1] P. Budnik and D. J. Kuck, "The organization and use of parallel memories," *IEEE Trans. Comput.*, vol. C-20, pp. 1566–1569, Dec. 1971.
[2] T. Feng, "Data manipulating functions in parallel processors and their implementations," *IEEE Trans. Comput.*, vol. C-23, pp. 309–318, Mar. 1974.
[3] J. D. Feldman and L. C. Fulmer, "RADCAP—An operational parallel processing facility," in *1974 Nat. Comput. Conf., AFIPS Conf. Proc.*, vol. 43, pp. 7–15.
[4] E. W. Davis, "STARAN parallel processor system software," in *1974 Nat. Comput. Conf, AFIPS Conf Proc.*, vol. 43, pp. 17–22.
[5] K. E. Batcher, "STARAN parallel processor system hardware," in *1974 Nat. Comput. Conf., AFIPS Conf. Proc.*, vol. 43, pp. 405–410.
[6] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.