Sorting networks and their applications

by K. E. BATCHER
Goodyear Aerospace Corporation
Akron, Ohio

INTRODUCTION

To achieve high throughput rates today's computers perform several operations simultaneously. Not only are I/O operations performed concurrently with computing, but also, in multiprocessors, several computing operations are done concurrently. A major problem in the design of such a computing system is the connecting together of the various parts of the system (the I/O devices, memories, processing units, etc.) in such a way that all the required data transfers can be accommodated. One common scheme is a high-speed bus which is time-shared by the various parts; speed of available hardware limits this scheme. Another scheme is a cross-bar switch or matrix; limiting factors here are the amount of hardware (an m X n matrix requires m X n cross-points) and the fan-in and fan-out of the hardware.

This paper describes networks that have a fast sorting or ordering capability (sorting networks or sorting memories). In $(\frac{1}{2})p(p+1)$ steps 2^p words can be ordered. A sorting network can be used as a multiple-input, multiple-output switching network. It has the advantages over a normal crossbar of requiring less hardware (an n-input n-output switching network can be built with approximately $(\frac{1}{4})\, n(\log_2 n)^2$ elements versus n^2 in a normal crossbar) and of having a constant fan-in and fan-out requirement on its elements. Thus, a sorting network should be useful as a flexible means of tieing together the various parts of a large-scale computing system. Thousands of input and output lines can be accommodated with a reasonable amount of hardware.

Other applications of sorting memories are as a switching network with buffering, a multiaccess memory, a multiaccess content-addressable memory and as a multiprocessor. Of course, the networks also may be used just for sorting and merging.

Comparison elements

The basic element of sorting networks is the comparison element (Figure 1). It receives two numbers over its inputs, A and B, and presents their minimum on its L output and their maximum on its H output.

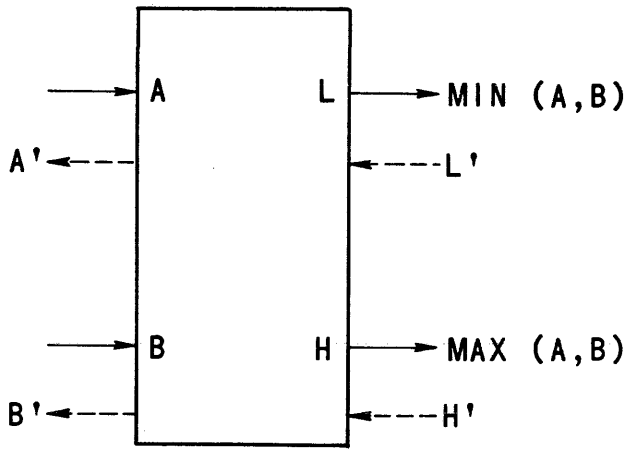

Figure 1 – Symbol for a comparison element

If the numbers in and out of the element are transmitted serially most-significant bit first the element has the state diagram of Figure 2. A reset input places the element in the A = B state and as long as the A and B bits agree it remains in this state with its outputs equal to its inputs. When the A and B bits disagree the element goes to the A < B or the A > B state and remains there until the next reset input. In the A > B state the output H equals the input A and the output L equals the input B. In the A < B state the opposite situation occurs.

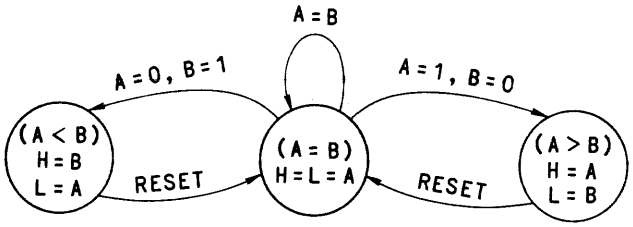

Figure 2 – State diagram for a serial comparison element (most-significant-bit first)

A serial comparison element can be implemented with 13 NORS and can be put on one integrated-circuit chip. When used in sorting networks each H and L output will feed an A or B input of another element so the fan-out is constant regardless of network size; this fact could be used to simplify the design of the chip. With several of the currently available logic families speeds of 100 nanoseconds/bit with a propagation delay from inputs to outputs of 40 nanoseconds are easily achieved.

Faster operation can be attained by treating several bits in parallel in each step with more complex comparison elements.

Some of the applications described below will require "bi-directional" comparison elements. Besides the A and B inputs and the H and L outputs there are H′ and L′ inputs and A′ and B′ outputs (see Figure 1). If A > B then B′ = L′ and A′ = H′, if A < B then B′ = H′ and A′ = L′, otherwise A′ and B′ are left undefined. Information flows from left-to-right over the solid lines and from right-to-left over the dotted lines.

Odd-even merging networks

Merging is the process of arranging two ascendingly-ordered list of numbers into one ascendingly-ordered list. Figure 3 shows a symbol for an "s by t" merging network in which the s numbers of one ascendingly-ordered list, $a_1, a_2, \ldots, a_s$ are presented over s inputs simultaneously with the t numbers of another ascendingly-ordered list, $b_1, b_2, \ldots, b_t$ over another t inputs. The s + t outputs of the merging network present the s+t numbers of the merged lists in ascending order, $c_1, c_2, \ldots, c_{s+1}$.

A "1 by 1" merging network is simply one comparison element. Larger networks can be built by using the iterative rule shown in Figure 4. An "s by t" merging network can be built by presenting the odd-indexed numbers of the two input lists to one small merging network (the odd merge), presenting the even-indexed numbers to another small merging network (the even merge) and then comparing the outputs of these small merges with a row of comparison elements.[1] The lowest output of the odd merge is left alone and becomes the lowest number of the final list. The i^{th} output of the even merge is compared with the $i+1^{th}$ output of the odd merge to form the $2i^{th}$ and $2i+1^{th}$ numbers of the final list for all applicable i's. This may or may not exhaust all the outputs of the odd and even merges; if an output remains in the odd or even merge it is left alone and becomes the highest number in the final list.

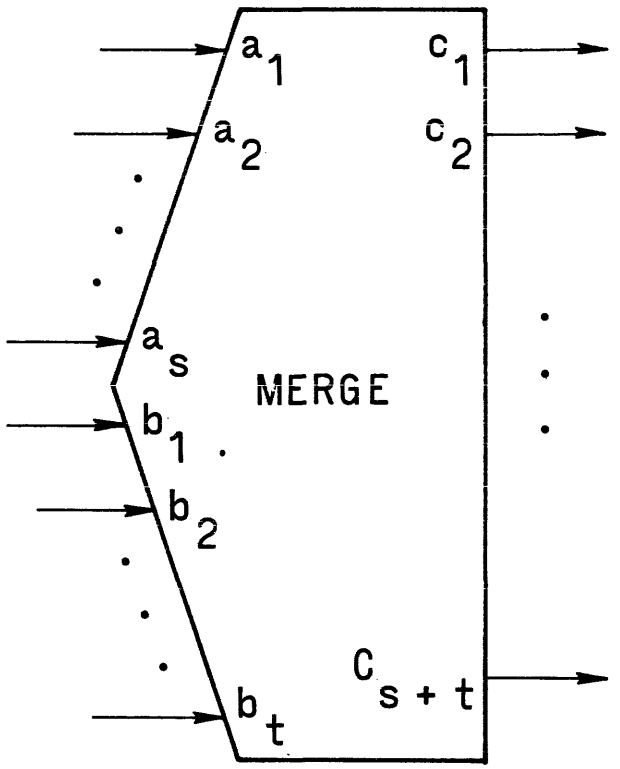

Figure 3 – Symbol for an "s by t" merging network

Appendix A sketches the proof of this iterative rule. Figure 5 shows a "2 by 2" and a "4 by 4" merging network constructed by this rule.

A "2^p by 2^p" merging network constructed by this rule uses $p.2^p+1$ comparison elements. The longest path goes through p+1 comparison elements and the shortest path through one element. Doubling the size of a merge only increases the longest path by unity so the merging time increases slowly with the size of the network.

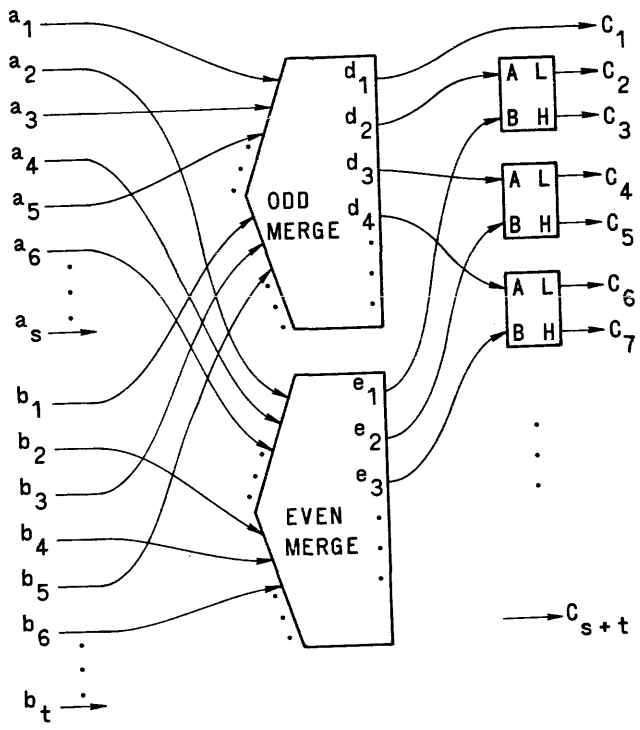

Figure 4 – Iterative rule for odd-even merging networks

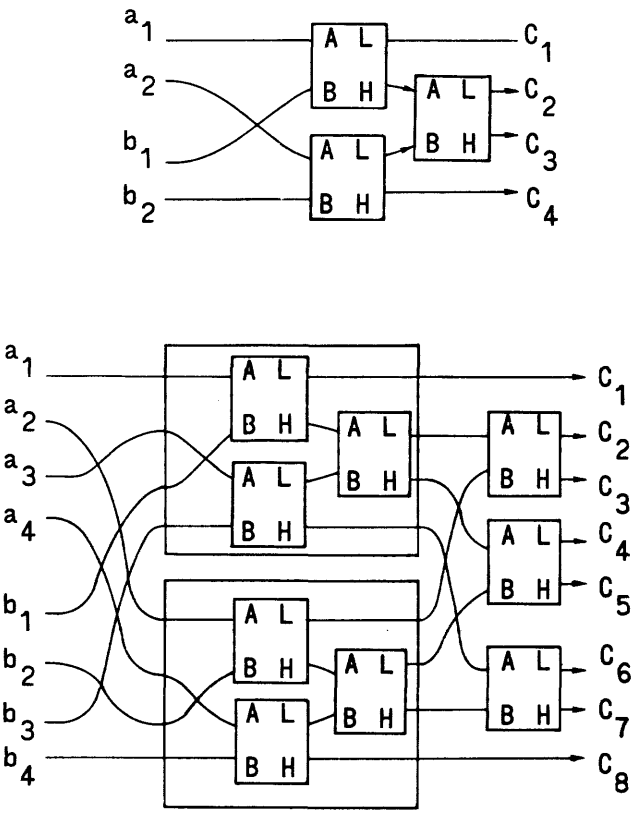

Figure 5 – Construction of "2 by 2" and "4 by 4" odd-even merging networks

Bitonic sorters

Another way of constructing merging networks from comparison elements is presented here. While requiring somewhat more elements than the odd-even merging networks, they have the advantage of flexibility (one network can accommodate input lists of various lengths) and of modularity (a large network can be split up into several identical modules).[2]

We will call a sequence of numbers *bitonic* if it is the juxtaposition of two monotonic sequences, one ascending, the other descending. We also say it remains bitonic if it is split anywhere and the two parts interchanged. Since any two monotonic sequences can be put together to form a bitonic sequence a network which rearranges a bitonic sequence into monotonic order (a bitonic sorter) can be used as a merging network.

Appendix B shows that if a sequence of 2n numbers, $a_1, a_2, \ldots, a_{2n}$ is bitonic and if we form the two n-number sequences:

$$\min(a_1, a_{n+1}), \min(a_2, a_{n+2}), \ldots, \min(a_n, a_{2n}) \quad (1)$$

and

$$\max(a_1, a_{n+1}), \max(a_2, a_{n+2}), \ldots, \max(a_n, a_{2n}), \quad (2)$$

that each of these sequences is bitonic and no number of (1) is greater than any number of (2).

This fact gives us the iterative rule illustrated in Figure 6. A bitonic sorter for 2n numbers can be constructed from n comparison elements and two bitonic sorters for n numbers. The comparison elements form the sequences (1) and (2) and since each is bitonic they are sorted by the two n-number bitonic sorters. Since no number of (1) is greater than any number of (2) the output of one bitonic sorter is the lower half of the sort and the output of the other is the upper half.

A bitonic sorter for 2 numbers is simply a comparison element and using the iterative rule bitonic sorters for 2^p numbers can be constructed for any p. Figure 7 shows bitonic sorters for 4 numbers and 8 numbers.* A 2^p-number bitonic sorter requires p levels of 2^{p-1} elements each for a total of $p \cdot 2^{p-1}$ elements. It can act as a merging network for any two input lists whose total length equals 2^p.

Large bitonic sorters can be constructed from a number of smaller bitonic sorters; for instance, a 16-number bitonic sorter can be constructed from eight 4-number bitonic sorters, as shown in Fig. 8. This allows large networks to be built of standard modules of convenient size.

*-Readers may recognize the similarity between the topologies of the bitonic sorter and the fast-fourier-transform.

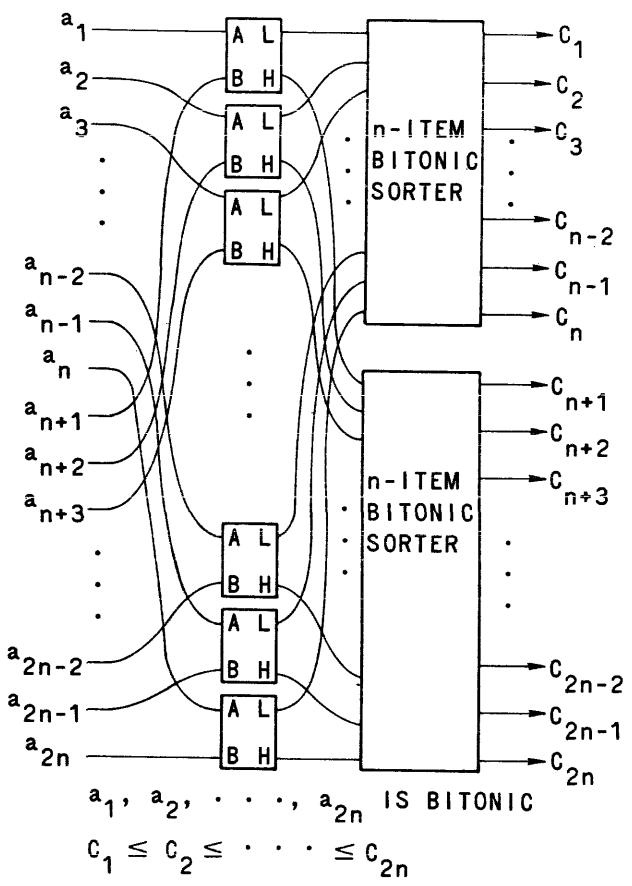

Figure 6 – Iterative rule for bitonic sorters

Sorting networks

A sorter for arbitrary sequences can be constructed from odd-even merges or bitonic sorters using the well-known sorting-by-merging scheme: The numbers are combined two at a time to form ordered lists of length two; these lists are merged two at a time to form ordered lists of length four, etc. until all numbers are merged into one ordered list.

To sort 2^p numbers using odd-even merges requires 2^{p-1} comparison elements followed by 2^{p-2} "2-by-2" merging networks followed by 2^{p-3} "4-by-4" merging networks, etc., etc. The longest path will go through $(\frac{1}{2})p\,(p+1)$ elements and the shortest path through p elements. The network requires $(p^2-p+4)2^{p-2}-1$ comparison elements.

To sort 2^p numbers using bitonic sorters requires $(\frac{1}{2})p(p+1)$ levels each with 2^{p-1} elements for (p^2+p) 2^{p-2} elements. Each path goes through $(\frac{1}{2})p(p+1)$ levels.

A sorter for 1024 numbers will have 55 levels and 24,063 elements with odd-even merges or 28,160 elements with bitonic sorters. With a 40 nanosecond propagation delay per level the total delay is 2.2 microseconds. Serial transmission of the bits would require about this much time between successive bits of the numbers unless re-clocking occurs within the

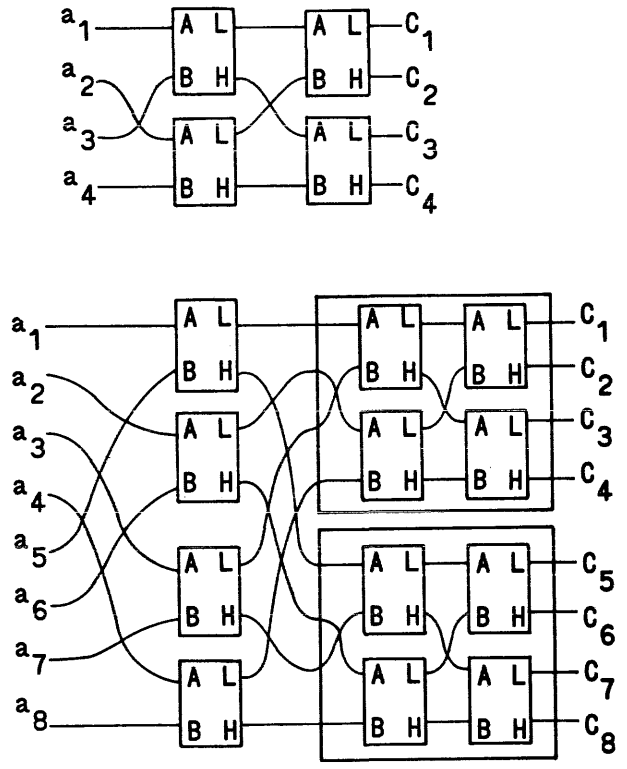

Figure 7 – Construction of bitonic sorters for 4 numbers and for 8 numbers

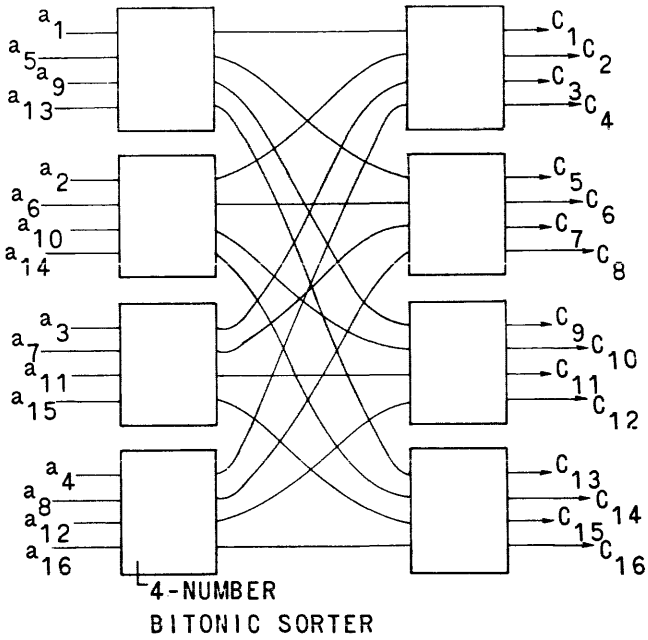

Figure 8 – A 16-number bitonic sorter constructed from eight 4-number bitonic sorters

network. Parallel-input-parallel-output registers of 1024 bits each can be placed between certain levels to perform this task or the re-clocking may be incorporated within each comparison element with a pair of flip-flops on the outputs. The latter scheme does not add to the terminal count of the comparison element so the cost of the added flip-flops on the comparison element chip is small. One can use any of the familiar techniques for driving shift registers such as the "A-B" technique where successive levels are clocked out-of-phase with each other. With present circuit and wiring techniques a bit rate of 10 megahertz may be possible with 50 nanosecond delay per level (2.75 microsecond delay from input to output of a 1024-word sorter).

With re-clocking in the elements and odd-even merges extra elements are needed to balance the unequal-length paths. Bitonic sorters do not have this problem.

Applications

The fast sorting capability of these networks allows their use in solving other problems where large sets of data must be manipulated. Some of these applications are sketched below.

Switching network

A sorting network can connect its input lines to its output lines with any permutation. The connection is made by numbering the output lines in order and presenting the desired output address for each input line at the input. The sorting network sorts the addresses and in the process makes a connection from each input line to its desired output line for the transmission of data. Bi-directional paths will be obtained if bi-directional comparison elements are used.

An alternative permuting network has been shown in the recent literature[3] which has less elements $[(p-1)2^p+1$ versus $(p^2-p+4)2^{p-2}-1$ for permuting 2^p items] but a more complex set-up algorithm.

Switching network with conflict resolution

The aforementioned switching network assumes each input wants a unique output line. In many applications conflicts between inputs occur and must be resolved by inhibiting conflicting inputs. Figure 9 sketches an m-input, n-output network that performs this task. Each input line inserts a word containing the output address desired (or zeroes if the line is inactive), a control bit equal to 1 and a priority number into an m-item sorting network with bi-directional elements. This orders the items so input items with the same output address are grouped together and ordered by their priority number. The ordered set of m-input items is merged with a set of n items, each containing a fixed output address and a control bit equal to O.

At the right side of the m by n merge the m + n items are in one ordered list; each address-inserter item will be directly below any input items with the same address. The adjacent word transfer network, looking at the control bits, connects each address-inserter item to the input item directly above it if one exists (the input item with lowest priority number is picked in each case). The elements in the sort and the merge are bi-directional so two-way paths are formed from input to output. The adjacent word transfer sends back signals over each path to signal each input and output line whether or not a connection has been established. Data can then be transmitted over each of the connected input lines.

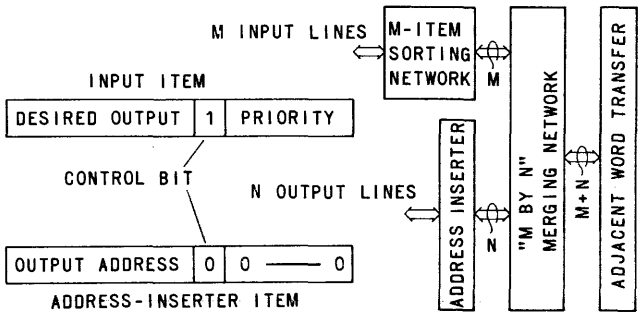

Figure 9 – An m-input, n-output switching network with conflict resolution

Multi-access memory

Re-clocking delays in the comparison elements give a sorting network some storage capability which can be augmented if needed with shift registers on the outputs. When the output lines are fed back to the input lines a recirculating self-sorting store is created (Figure 10). In each recirculation cycle word positions are changed to keep the memory in order.

Inputs to the memory can be made by breaking the recirculation paths of some words and inserting new words. To prevent destroying old information during input we use the convention that words with all bits equal to "one" are "empty" and contain no information; these will automatically collect at the "high-end" of memory where input lines can use them to insert new words.

Outputs from the memory can be accommodated by reserving the most-significant-bit (MSB) of each word; "1" for normal words and "O" for words to be outputted. Words for output will automatically collect at the "low end" of memory where output lines can read them. Selection of which words to output is accommodated by reserving the least-significant-bit (LSB) of each word; "1" for normal words and "O"

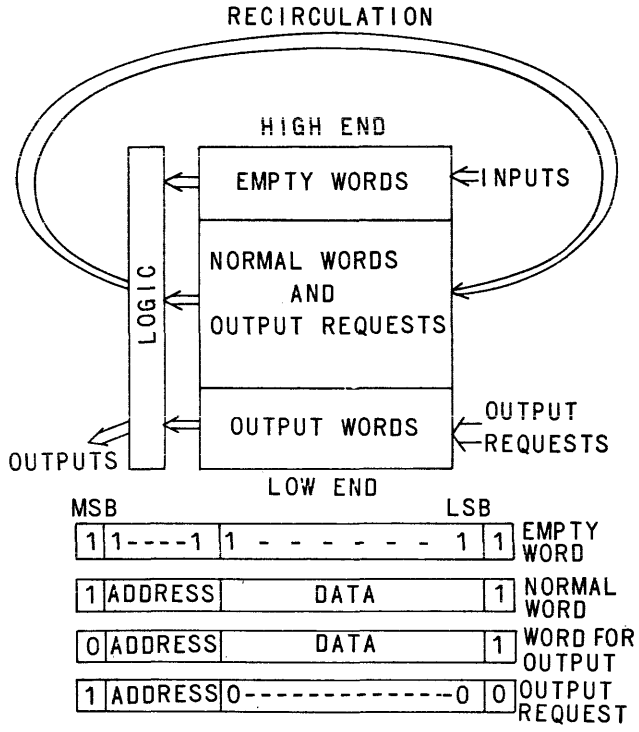

Figure 10 – A multi-access memory

for "output requests". Logic between adjacent words causes an output request to affect the word directly above it.

During one recirculation cycle new words and output requests are entered into memory. During the next recirculation cycle all words are recirculated with no new entries. At the end of the cycle the LSB of each word will precede the MSB of the same word (no reordering occurs in the second cycle). Output requests are identified by a "O" in the LSB and for each request logic performs the following action: if the word above the request is a normal word ("1" in the LSB) change its MSB to a "O" and empty the request (change all its bits to "1" as they fly by), if the word above the request is another request change the MSB of the first request to "O". During the following recirculation cycle the selected words and unfulfilled requests flow to the low end of memory and are read by output lines. Because the request itself is outputted if no word is found, as many outputs as original requests occur. If the original requests were in order the outputs directly correspond to them (a second sorting network can put the original output requests in order).

In use the more-significant part of each word is used as an address and the rest as data. To request a certain address an output request is sent in with that address and zeros for data. The word returned will be at that address or a higher address if the requested address is empty.

While a complete cycle may be long in this memory (50-bit words at 100 nanoseconds/bit = 5 microseconds/recirculation = 10 microseconds/complete cycle) many inputs and outputs can be accommodated in each cycle. An effective rate of 100 nanoseconds/word is achieved with 100 inputs and outputs.

Such a memory could be useful as the "common memory" of multiprocessors. The self-sorting capability could be useful for keeping "task lists" up to date and performing other housekeeping tasks.

Other uses may be as a message "store-and-forward" system and as a switching network with buffering capability. In these uses each output device is given a unique address which it continually interrogates; input devices send their data to these addresses.

Multi-access content addressable memory

By adding facilities for shifting the bits within the words in the aforementioned memory different fields of the words can be brought into the more-significant portions which govern the ordering of the words. Addressing can then take place on any part of the words. As long as the same field positions are being searched more than one search can be accommodated simultaneously.

Multi-processor

By adding processing logic to perform additions, subtractions, etc., on groups of adjacent words of a sorting memory one can implement a multi-processor. The sorting capability is used to transmit operands between processors. Merely by changing address fields the multiprocessor can be reconfigured quickly. Such a multi-processor can keep up with the "dynamic topology" of certain real-time problems.

To simplify the processing logic one might use the same network or another network to perform table look-up arithmetic. It is possible to have all the processors search the same tables simultaneously.

SUMMARY

Sorting networks capable of sorting thousands of items in the order of microseconds can be constructed with present-day hardware. Such fast sorting capability can be used to manipulate large sets of data quickly and solve some of the communications problems associated with large-scale computing systems.

Standard modules of convenient sizes can be picked and used in any size network to lower the cost. Large-scale integration can be applied if the problem of laying out the rather complex topology of the network can be solved. Studies of this problem are being conducted at Goodyear Aerospace.

APPENDIX A—SKETCH OF PROOF OF ITERATIVE RULE FOR ODD-EVEN MERGING

Let $a_1, a_2, a_3, \ldots$ and $b_1, b_2, b_3, \ldots$ be the two ordered input sequences. Let $c_1, c_2, c_3, \ldots$ be their ordered merge, $d_1, d_2, d_3, \ldots$ be the ordered merge of their odd-indexed terms and $e_1, e_2, e_3, \ldots$ be the ordered merge of their even-indexed terms.

For a given i let k of the $i+1$ terms in $d_1, d_2, d_3, \ldots, d_{i+1}$ come from $a_1, a_3, a_5, \ldots$ and $i+1-k$ come from $b_1, b_3, b_5, \ldots$ The term d_{i+1} is greater than or equal to k terms of $a_1, a_3, a_5, \ldots$ and therefore is greater than or equal to $2k-1$ terms of $a_1, a_2, a_3, \ldots$ Similarly it is greater than or equal to $2i+1-2k$ terms of b_1, b_2 $b_3, \ldots$ and hence $2i$ terms of $c_1, c_2, c_3, \ldots$ Therefore

$$d_{i+1} \geq c_{2i}. \tag{A1}$$

Similarly from consideration of the i terms of e_1, e_2 $e_3, \ldots, e_i$ the inequality

$$e_i \geq c_{2i} \tag{A2}$$

is obtained.

Now consider the $2i+1$ terms of $c_1, c_2, c_3, \ldots, c_{2i+1}$ and let k come from $a_1, a_2, a_3, \ldots$ and $2i+1-k$ come from $b_1, b_2, b_3, \ldots$ If k is even we have that c_{2i+1} is greater than or equal to:

k terms of $a_1, a_2, a_3, \ldots$
$(\frac{1}{2})$k terms of $a_1, a_3, a_5, \ldots$
$2i+1-k$ terms of $b_1, b_2, b_3, \ldots$
$i+1-(\frac{1}{2})$k terms of $b_1, b_3, b_5, \ldots$
$i+1$ terms of $d_1, d_2, d_3, \ldots$

and similarly c_{2i+1} is greater than or equal to i terms of $e_1, e_2, e_3, \ldots$
so

$$c_{2i+1} \geq d_{i+1} \tag{A3}$$

and

$$c_{2i+1} \geq e_i \tag{A4}$$

If k is odd, (A3) and (A4) still hold.

Since every item of $d_1, d_2, d_3, \ldots$ and $e_1, e_2, e_3, \ldots$ must appear somewhere in $c_1, c_2, c_3, \ldots$ and $c_1 \leq c_2 \leq c_3 \leq \ldots$ inequalities (A1), (A2), (A3) and (A4) imply that

$$c_{2i} = \min(d_{i+1}, e_i) \tag{A5}$$

and

$$c_{2i+1} = \max(d_{i+1}, e_i). \tag{A6}$$

APPENDIX B—SKETCH OF PROOF OF ITERATIVE RULE FOR BITONIC SORTERS

Let $a_1, a_2, a_3, \ldots, a_{2n}$ be bitonic. Let $d_i = \min(a_i, a_{n+i})$ and $e_i = \max(a_i, a_{n+i})$ for $1 \leq i \leq n$. We want to prove that $d_1, d_2, \ldots, d_n$ and $e_1, e_2, \ldots, e_n$ are each bitonic and

$$\max(d_1, d_2, \ldots, d_n) \leq \min(e_1, e_2, \ldots, e_n). \tag{A7}$$

If $a_1, a_2, a_3, \ldots, a_{2n}$ is split into two parts and the parts interchanged $d_1, d_2, \ldots, d_n$ and $e_1, e_2, \ldots, e_n$ undergo a similar interchange. This does not affect the bitonic property nor affect (A7) so it is sufficient to prove the proposition for the case where

$$a_1 \leq a_2 \leq a_3 \leq \ldots \leq a_{j-1} \leq a_j \geq a_{j+1} \geq \ldots \geq a_{2n} \tag{A8}$$

is true for some j $(1 \leq j \leq 2n)$.

Reversal of the terms of sequences does not affect the bitonic property nor maximums and minimums so it is sufficient to assume $n < j \leq 2n$.

If $a_n \leq a_{2n}$ then $a_i \leq a_{n+i}$ so $d_i = a_i$ and $e_i = a_{n+i}$ for $1 \leq i \leq n$ and the proposition holds.

If $a_n > a_{2n}$ then from $a_{j-n} \leq a_j$ we can find a k such that $j \leq k < 1n$, $a_{k-n} \leq a_k$ and $a_{k-n+1} > a_{k+1}$ (the sequence $a_j, a_{j+1}, a_{j+2}, \ldots, a_{2n}$ is decreasing while the sequence $a_{j-n}, a_{j+1-n}, a_{j+2-n}, \ldots, a_n$ is increasing). Then

$$\left.\begin{aligned} d_i &= a_i \\ e_i &= a_{i+n} \end{aligned}\right\} \text{ for } 1 \leq i \leq k-n \tag{A9}$$

and

$$\left.\begin{aligned} d_i &= a_{i+n} \\ e_i &= a_i \end{aligned}\right\} \text{ for } k-n < i \leq n. \tag{A10}$$

The inequalities

$$d_i \leq d_{i+1} \text{ for } 1 \leq i < k-n, \tag{A11}$$

$$d_i \geq d_{i+1} \text{ for } k-n < i < n, \tag{A12}$$

$$e_i \leq e_{i+1} \text{ for } k-n < i < n, \tag{A13}$$

$$e_n \leq e_1, \tag{A14}$$

$$ei \leq e_{i+1} \text{ for } 1 \leq i < j-n, \tag{A15}$$

and

$$e_i \geq e_{i+1} \text{ for } j-n \leq i < k-n \tag{A16}$$

can be shown which prove that $d_1, d_2, \ldots, d_n$ and $e_1, e_2, \ldots, e_n$ are bitonic and $\max(d_1, d_2, \ldots, d_n) = \max(a_{k-n}, a_{k+1}) \leq \min(a_k, a_{k-n+1}) = \min(e_1, e_2, \ldots, e_n)$.

ACKNOWLEDGMENTS

The help of D. L. Rohrbacher, P. A. Gilmore and others at Goodyear Aerospace is gratefully acknowledged.

Part of this work was supported by Rome Air Development Center under Contract AF30(602)-3550, F. Dion, Administrator.

REFERENCES

1 K E BATCHER
A new internal sorting method
Goodyear Aerospace Report GER-11759 1964

2 K E BATCHER
Bitonic sorting
Goodyear Aerospace Report GER-11869 1964

3 L J GOLDSTEIN S W LEIBHOLZ
On the synthesis of signal switching networks with transient blocking
IEEE Transactions EC-16 5 637-641 1967