LOGIC-ENHANCED MEMORIES: AN OVERVIEW AND SOME EXAMPLES OF THEIR APPLICATION TO A RADAR TRACKING PROBLEM

By

W. Michael Denny

Ernest R. Buley

Earl Hatt

General Research Corporation

Santa Barbara, California

CALTECH CONFERENCE ON VLSI, January 1979

1 INTRODUCTION

For the past two years, General Research Corporation has been investigating ways of using very large amounts of active memory to solve some stressing data processing problems encountered in ballistic missile defense systems. Our approach has been to use large numbers of simple logic elements (from 2 thousand to 2 million elements, each of some 200 gates) distributed throughout very large memories (10^{12} bits).

Two aspects of this work give it a unique place in the current milieu of research on von Neumann architectures. First, the work has been directed toward fairly specialized problems in ballistic missile defense. This problem-directed approach has been consciously adopted in view of the relatively undeveloped state of highly parallel algorithms and architectures. We have found that the exigencies of the problem itself guided us through three separate architectural designs to solve one problem. These different designs reflected different algorithmic approaches to the problem and would be implemented by different circuit technologies. (We are now constructing a portion of one design, of sufficient size to permit a more detailed proof-of-principle verification.) Until the algorithmic and architectural theory of parallel structures matures, we expect much practical design to be intuitively guided. Perhaps the experiences reported here will help guide the intuitions of others.

Second, the highly parallel structures we describe are basically memory structures. The ratio of logic gates to memory bits is roughly 10^{-5}. However, in these structures, memory is not treated as a passive functional unit characterized only by storage capacity and access time. Rather, the memory is seen as an active component which cooperates with a more conventional processor in the data processing task. Although the logic-enhanced memory performs much of the processing, we have not insisted that it be responsible for all of it. Much work remains to be done in exploring the appropriate division of labor between the "processor" and its logic-enhanced memory.

1.1 ARCHITECTURAL CONTEXT

Traditional memories have been considered relatively passive functional elements in data processing. They are characterized by their reliability, size, and speed and by the way their contents can be retrieved (referenced) by the processor. Memories which transform the stored data themselves are rare. Although recently memories have gradually taken a more active role, their increased internal activity has generally been in the interests of increased reliability (error detecting/correcting memories), increased storage and retrieval speed (buffer and cache memories), and increased size (hardware "paging boxes" and other hardware-managed virtual memory hierarchies).

These low levels of active memory architectures have themselves spawned a more complex category of active memories. As the size and effective speed of memories increases, processors spend more and more time threading their way through the disorganized and arbitrarily organized data in order to find the data needed at various processing steps. Memories whose internal activity assists the processor in this chore represent a first step toward a more equitable division of the processing load between the processor and storage functions. "Defined field"[1] memories remove much of the arbitrary data organization imposed by the memory's word length. Content Addressable Memories (CAMs) remove the artificial organization imposed on the data by its physical location in memory by allowing the contents of the data itself to express some simple aspects of its organization. FIFO and LIFO memory organizations implement in hardware a few of the more common data structures which asynchronous processes, expression evaluation, block structured languages, and structured control have found useful.[2]

1.2 THE LOGIC-ENHANCED MEMORY

Architectural considerations such as those above have been partially responsible for the direction of our recent research. We are exploring a class of active memory structures which we believe represents another step in the direction of more equitable distribution of labor between the processing and storage elements. One approach to such processor/memory combinations is a simple extension of Content Addressable Memories (CAMs).

Sometimes the simple match/don't-care or bounded search criteria used by CAMs to associate data are not appropriate. We might like to know which data are related by simple functional transformations or satisfy more general relationships than match/don't-care or bounded search. For example, we might ask which sets of data not only are identified by a match/don't-care condition, but also satisfy the criterion that each value-object in the set, when divided by a corresponding value-object in some other set, produces a result approximately equal to some value. The parameters specifying "approximately equal" could be passed to this type of memory along with the value the division should produce for a "hit" to be recorded. Such a memory would make an effective matched filter in signal processing applications. And analogous to the highly parallel match circuits found in a CAM, this example of a "logic-enhanced memory" would have a highly parallel set of divide and compare

---

[1] W. T. Wilner, "The Design of the Burroughs B1700," FJCC, 1972; and "Memory Utilization on the B1700," FJCC, 1972.

[2] Yaoh-an Chu, High Level Language Computer Architecture, Academic Press, 1975, pp. 63-107.

circuits to transform the data and determine which ones should be retrieved. While not reported here, such a logic-enhanced memory has been designed at General Research Corporation in a separate effort. Its internal structure is very much like that of the third solution reported in Sec. 2 of this paper.

Simple and highly parallel transformations applied to the data can be used not only to aid retrieval, but to enable a memory to produce its own data for subsequent processing. Finite difference methods fit nicely with the concept of arithmetic simplicity of the data transformation done by the logic-enhanced memory. An example of a bistatic radar "target finder" using just such an approach is given in Sec. 2 of this paper.

We have found that when traditional processor-memory architectures are used along with such logic-enhanced memories, their computational burden is reduced by several orders of magnitude. For example, the bistatic target finding problem discussed in Sec. 2 can be shown to require a computational power equivalent to about twenty PEPE-class processors. However, a logic-enhanced memory architecture resembling an array of 12,288 Babbage-like differential engines distributed over 6×10^{11} bits of storage reduces the magnitude of the problem to one well within the capacity of a modern minicomputer or microprocessor.

Logic-enhanced memories also promise a mechanism for utilizing very large memories (1 to 10 billion bytes). Because the LEM does much of the manipulation of large amounts of data, and presents the processor with much smaller quantities of data (LEMs give the processor only the "right" data), the processor's address space can be much smaller than that actually spanned by the LEM memory. This is of most value for processors in the mini-micro range and suggests that LEMs and arrays of microprocessors or small minicomputers could be profitably combined. Microprocessor arrays promise relatively large amounts of processor power over a small ($\leq$ 24 bits) address space. A LEM architecture could provide part or all of the storage used by an array of microprocessors.

In the LEM architectures discussed in Sec. 2, it is important to realize that although some 10^{12} bits of storage are used, the minicomputer using the LEM does not directly address any of it except the "top" containing the proper data. In this sense the LEM's role in a problem is analogous to the role played by a stack in expression evaluation: both allow large amounts of storage to be used, but not directly addressed by the processor. This gives rise to the tantalizing notion that LEM architectures might not only reduce the address space requirement of some conventional processors, but under some circumstances, eliminate the notion of addressable storage altogether--and with it the host of address-related problems inflicted on computer architecture.[1]

---

[1] See, for example, W. Lonergon and P. King, "The Design of the B5000," Datamation, Vol. 7, No. 5, pp. 28-32, May, 1961; and P. Christy, "Minicomputer Architecture Links Past and Future Generations," Electronics, July 6, 1978, pp. 98-105.

Furthermore, the logical complexity of the functional elements which we have thus far used in our LEM designs (100-500 gates) is far below that of current microprocessors (4000-7000 gates). (This low level of complexity suggests that, while a microprocessor implementation of a LEM might be a worthwhile research tool, it would be inferior to a custom LSI or VLSI implementation in size-critical applications.) The LEM logic elements not only have one to two orders of magnitude fewer gates than microprocessors, but are an order of magnitude faster. This leads one to consider the LEM elements and microprocessors as forming levels in a hierarchy. The LEM elements are valuable in data-intensive problems, just as ALUs are valuable components in computation-intensive algorithms. Computationally stressing calculations can themselves often be traded off for more memory. With the advent of high-density memories, data-intensive algorithms using some simple, highly-distributed logic to process the data appear to be cost-effective alternatives in problems traditionally requiring very fast processors.

To date, we have developed one LEM design in enough detail to permit prototype manufacture. This design, described in Sec. 2, produces data at a rate that a processor in the PDP-11/45 class can handle. It remains to be seen whether subsequent LEM designs have similar characteristics and whether they are appropriately matched to microprocessor arrays.

Another feature of the simplicity of the logic elements in LEMs is the possibility of tight coupling of the algorithm embodied in the logic to the physical properties of the storage medium itself. At first glance, this philosophy may seem contrary to the trend in hardware/software design toward increasing isolation of the algorithm from the hardware. However, even though the algorithm implemented in the LEM may itself be very hardware-dependent, its effect is very often to isolate the other processing elements from the intricacies of the storage unit's physical organization. The "difference engine" LEM design in Sec. 2 is an illustration of this effect. As different types of novel storage mechanisms proliferate, LEM designs can serve very effectively in allowing the processor to treat storage in an algorithmically structured manner independent of the physical organization of the storage devices. Furthermore, because the LEM logic is itself relatively simple, the fact that it is tuned to the hardware in which it is implemented is not so distasteful as it is with more complex algorithms running on "general purpose" processors.

A final consequence of LEM logic's simplicity is more of a hope than an observed property. If the logic itself needs only 100 gates or so, it is more likely to be implementable in the same technology as the memory itself--probably even on the same chips. The advent of custom VLSI makes this possibility a very economically attractive one.

2 THREE LOGIC-ENHANCED MEMORY DESIGNS

2.1 THE "DEGHOSTING" PROBLEM

One type of radar system that has been considered for ballistic missile defense is a "bistatic multilateration" target locator. Such a system works like this.

Suppose we have a pair of radars--one which transmits and one which receives the pulse reflected from a target, as illustrated in Fig. 1. The only information that the receiving radar extracts from the reflected pulse is the time since it was transmitted. This information is enough to locate the target on an ellipsoid whose foci are the transmitter and receiver and which is rotated about a line joining these foci. In two dimensions, the "bistatic range"--from transmitter to target to receiver--defines an ellipse, and it takes one more receiver to locate the target at the intersection of two ellipses.

Serious problems begin to emerge if there are several targets in the field of view, as in Fig. 2, where there are four intersections of the ellipses defined by the bistatic ranges of two receivers.

In two dimensions, a third receiver is necessary to separate the "real" intersections from the "ghost" intersections. The third receiver

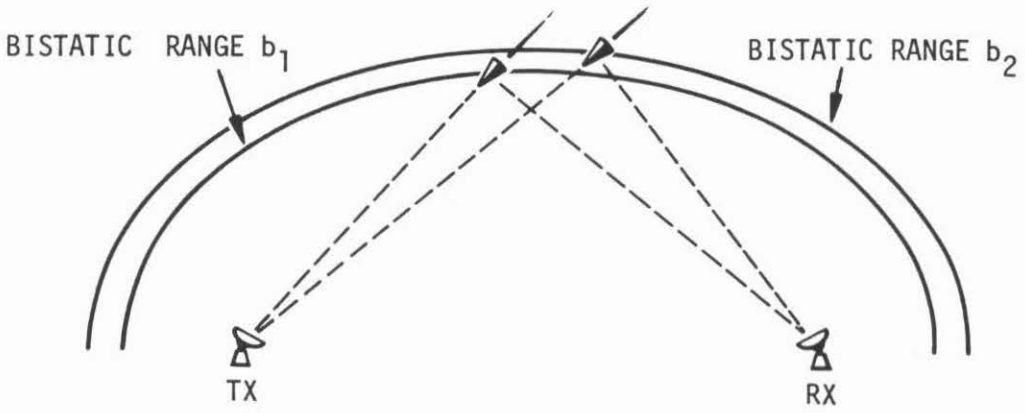

Figure 1. A bistatic range measurement locates an object on the surface of an ellipsoid of revolution.

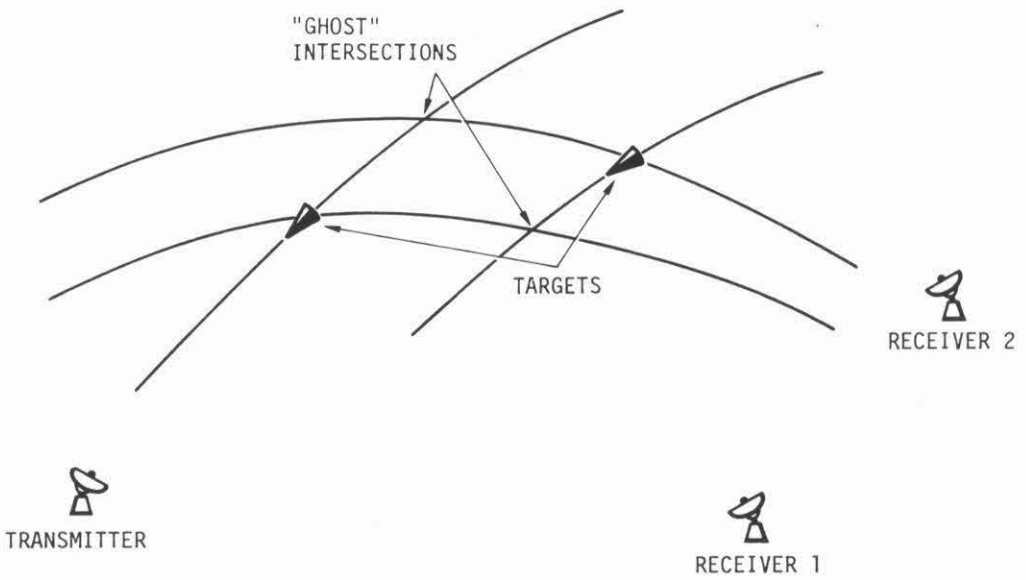

Figure 2. Multiple targets produce "ghost" intersections

---

defines an additional ellipse for each target. Targets are then recognized by the intersection of all three ellipses, whereas "ghosts" are located at intersections of only two ellipses.

In three dimensions, four receivers are required to locate multiple targets. And, in practice, because of noise in the bistatic transmit times and the vagaries of radar transmission and reception, a 5-out-of-6 scheme would be used. It can be shown that the number of "ghost" intersections is of the order of N^3, where N is the number of targets in the common viewing volume of the single transmitter and several receivers.

Forty targets, a realistic threat density, would result in 64,000 "ghosts". The real-time requirements of the defense system dictate that the N real targets be separated from the N^3 ghosts in about 3 ms--imposing an enormous load on a sequential processor. Forty targets would (conservatively) require about 20 PEPE-class processors. This large amount of required processing, coupled with the inherent parallelism of the problem, prompted us to examine the logic-enhanced memory as a possible solution.

2.2 SOLUTION ONE

The past thirty years of computing have given us some intuition about problem-solving on sequential machines, but very little about problem-solving with highly parallel structures. In our approach to this problem we found it advantageous to imagine the volume around the radars as being subdivided into "elemental volumes" in various ways. Mentally, we could picture each such elemental volume of space as associated with a LEM processing element--a simple logic-memory combination. We then imagined each elemental volume being processed in parallel with all the rest. (A side effect of viewing the problem in such spatial terms was that if each LEM element could do its work independent of the number of targets, then the processing time for the whole group of LEM elements could also be nearly independent of the number of targets. This desirable property is in sharp contrast to a sequential processor, where the processing time for the classical algorithm rises exponentially with the number of targets.)

Our first approach to this problem was to imagine the entire volume around the radars divided into cubes (see Fig. 3), each associated with a small processing element. Each element is pre-loaded with the values of bistatic range that determine an ellipsoid passing through its cube. Each element then reports only when five (of the possible six) ellipsoids intersect in its cube of space. A logic-enhanced memory built

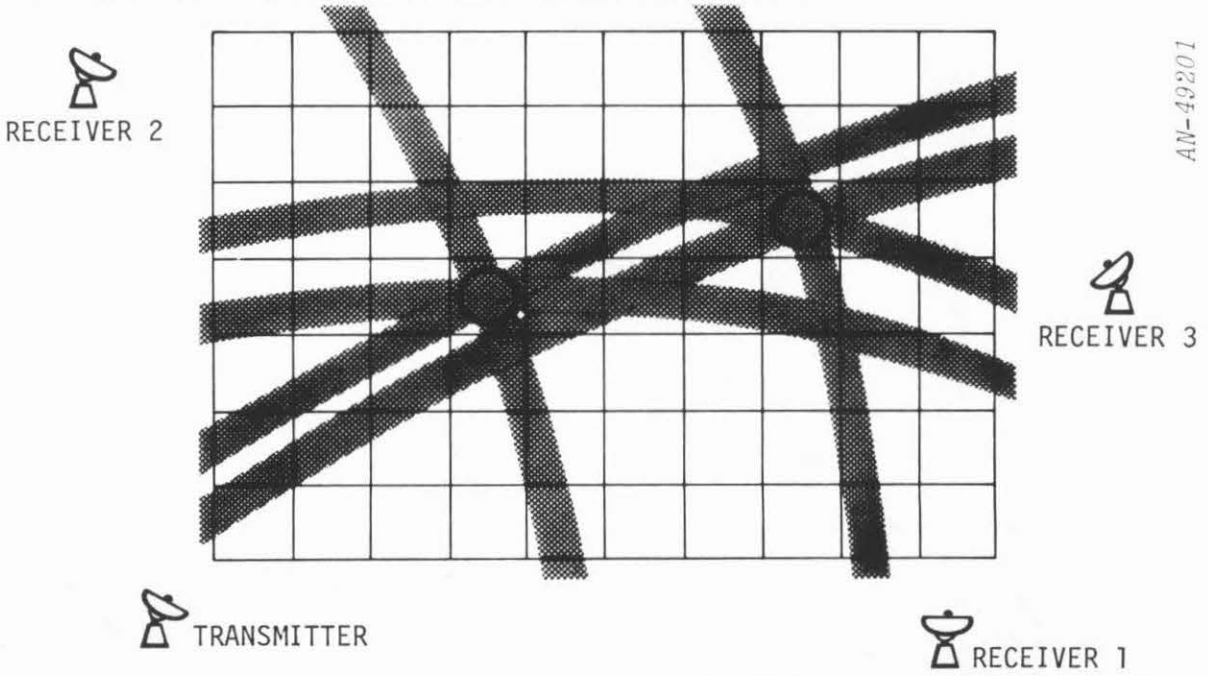

Figure 3. In Solution One, space is divided into regular cubes, and a LEM element is associated with each. The LEM element computes the number of ellipsoids that intersect within its cube.

around this concept was designed and roughly simulated. It was rejected because it was found that adequate performance required approximately 10^{10} elements (10^{13} bits), a number well beyond practical near-term development. In addition, this approach resulted in less than 1% of the elements reporting hits, even though the noise of the radars resulted in many elements whose cubes were outside the transmitter's beam reporting hits. It seemed clear that the information whether or not the cube was in the transmitter's beam should be added to the hit count.

2.3 SOLUTION TWO

An obvious way to do this was to subdivide only the volume of the transmitter's beam into cells--each cell being identified with a LEM element. Whenever the beam position changed, the appropriate LEM cells would be reloaded with precalculated parameters expressing the geometric relationship of the new transmit beam to whichever set of six radar receivers was selected.

To be more specific, the transmitter's beam was divided radially into 2048 "rays" emanating from the transmitter as in Fig. 4. Associated with each ray was a LEM element consisting of six calculation circuits (one for each receiver), six comparison circuits, and a 5-out-of-6 comparator to examine the six comparison circuits for 5 out of 6 "hits" (see Fig. 5). Individual hits were discovered through the following mechanism: we observed that if the different bistatic ranges for an object (obtained by the different receivers) corresponded to a real target (an intersection of 5 out of 6 ellipsoids), then each of the different bistatic ranges would correspond approximately to the same distance, r, from the transmitter. The relationship between r_j, the distance from the transmitter to the target along the jth ray, and B_i, the bistatic range measured by the ith receiver, can be shown to be:

$$r_j = \frac{1}{2} \frac{B_i^2 - P_{ij}^2}{B_i - C_{ij}} \qquad (1)$$

where P_{ij} and C_{ij} are fixed geometric parameters relating the jth ray in the transmitter's beam to the ith receiver.

Given this, it was only necessary for each of the six calculation circuits in each ray to solve Eq. 1 for r_j--given each bistatic range, B_i, it received, and the comparison circuit to check whether five out of six values for r were "close enough." If so, a hit was recorded and a target located. In this design, then, a LEM element is identified with a volume of space corresponding to an elemental ray. Then each LEM element is divided into six smaller subelements, one for each of the six receivers.

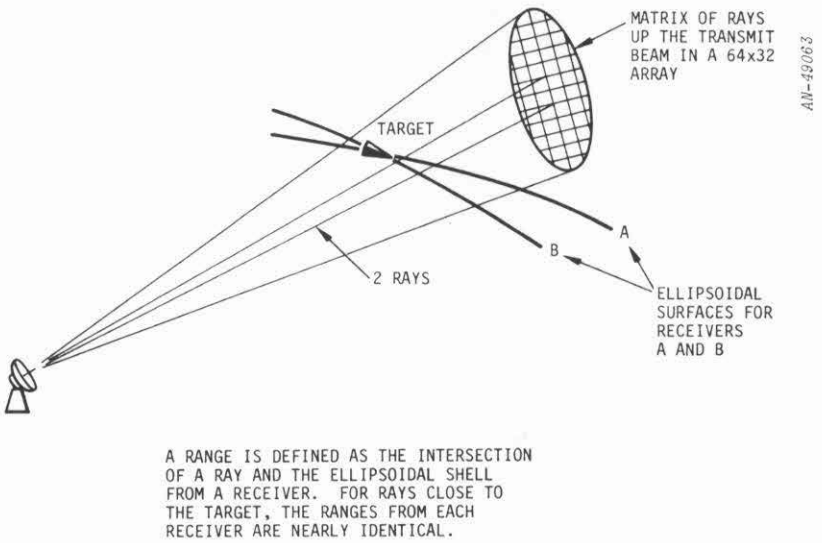

Figure 4. In Solution Two, a range is defined by the intersection of a "ray" and the ellipsoidal shell from a receiver. For rays close to the target, the ranges from each receiver are nearly identical

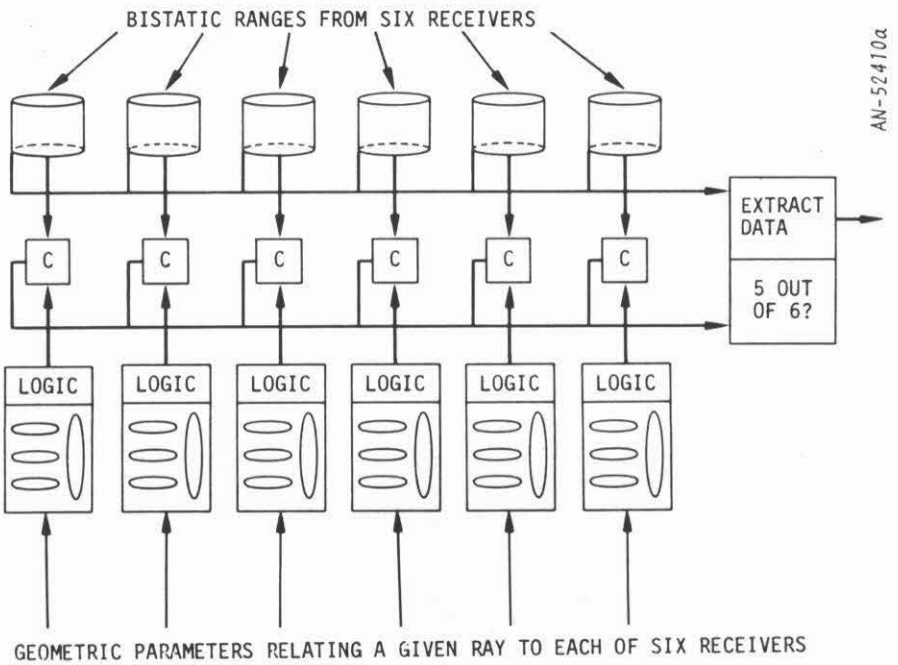

Figure 5. A Single LEM Element for Solutions Two and Three

Simulation of this design indicated that in a realistic scenario with about 40 targets in the beam, the LEM would reject about 99% of the 64,000 ghosts in the beam, and pass 560 ghosts along with the 40 real targets to a postprocessor that would eliminate the remaining ghosts. These residual ghosts are produced by the effects of noise in the radar signals, and are removed by fairly standard signal extraction techniques on a sequential processor (a PDP-11/45). Since relatively small numbers of these residual ghosts were passed to the sequential processor, we did not examine the possibility of removing them with a LEM-like device.

2.4 SOLUTION THREE

Solution Two was on the verge of prototype manufacture when a simpler design was discovered. First, notice that Eq. 1 is monotonic in the bistatic ranges B_i, which arrive at the receivers in ascending order because returns from the closest targets arrive before those from the more distant ones. This implies that if we view the problem in terms of the ranges r from the transmitter to the target, then the bistatic returns arrive in the order of their range distance along the appropriate ray.

This implies that if each ray were to be scanned in an ascending manner beginning at the transmitter, the set of bistatic ranges does not have to be randomly accessed to discover which B_i's correspond to a given range r. Instead, it is only necessary to store the set of B_i's in a FIFO memory, with one FIFO memory for each receiver for each ray. This observation not only allows us to use a simple memory structure for the bistatic range buffers of Fig. 5, but also suggests a way to simplify the logic of each of the six calculation circuits in each ray-associated element. A finite difference method can be used to "scan along" each ray.

Each of the memories of the six logic elements in each LEM element is loaded with a set of parameters describing the first, second, and third differences of the expression in Eq. 1. Then, by simple addition, the corresponding ΔB_i can be calculated for each Δr along the ray. Thus, the logic in each of the six elements can be replaced with the simple "difference engine" logic of Fig. 6.

In operation the initial bistatic range is loaded into register A, its rate of change with respect to r into B, the rate of rate of change into register C, and the rate of rate of rate of change into D. For each incremental advance along the ray, the clock line is asserted, causing the following operations:

$C \leftarrow C+D$
$B \leftarrow B+C$
$A \leftarrow A+B$

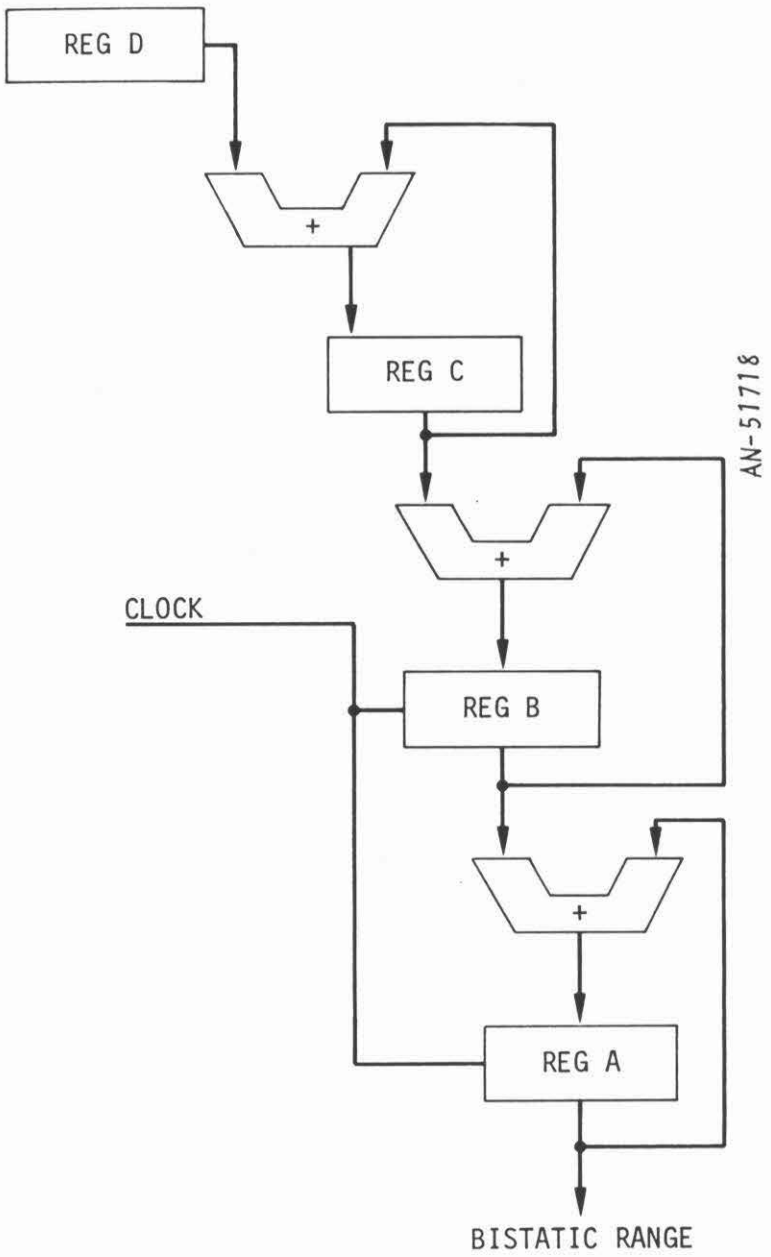

Figure 6. A Target Finder Difference Engine

(Simulation of this design under realistic scenarios has indicated appropriate register sizes and their initial values. More details are given in a General Research Corporation contract report.[1]

Now, at each step along the range r , the corresponding window of bistatic ranges ΔB_i can be calculated by each of the six logic elements in a ray-associated LEM element. The received bistatic ranges in the FIFO memories are compared with the calculated ΔB_i by each of the six comparison circuits and a "probable target" is reported if five of the six comparators report a hit. The 5-out-of-6 condition only indicates a probable target because the noise in the received bistatic returns causes some bistatic returns to only apparently fall within the required window. This effect produces some false targets which are then eliminated by a PDP-11/45 class postprocessor. High-fidelity simulations of this LEM target finder design indicate that with 40 targets in the field of view, this LEM target finder reduces the 64,000 ghosts to a more manageable 600 false targets.[1] Thus, this simple collection of 12,288 Babbage-like differential engines distributed over about 10^{12} bits of memory reduces the calculation load from one requiring about twenty PEPE processors to one requiring only a small minicomputer.

A single element for such a "difference engine" LEM target finder is being constructed now[2] and will be installed at the BMD Advanced Technology Center at Huntsville, Alabama, during the second quarter of 1979 for more exhaustive testing.

2.5 CONCLUSIONS

The following table illustrates some of the major design properties of the three logic-enhanced memory designs. (In fact, two additional LEM designs were developed for the target finder problem, but in the interest of brevity they are not reported here.)

Solution One utilizes considerable memory and very little logic to process the data in each element. Its relatively slow performance is due primarily to the time required to load such an enormous number of elements with bistatic range information and interrogate them for "hits". Such an architecture is communication-bound.

Solution Two uses the least memory and the most complex logic per memory element. The solution exhibits the most processor-intensive architecture and is limited by the processing time of the bit-slice microprocessor used to implement the design.

---

[1] E. Hatt and H. Ostrowsky, Logic-Enhanced Memory: Final Report, Vol. I, "The LEM Target Finder," General Research Corporation CR-2-776 (Contract DASG60-77-C-0066), July, 1978.

[2] By Honeywell Advanced Systems, Minneapolis, Minnesota.

	Number of LEM Elements	Memory Required, Bits	Logic Required Per Element	Time to Deghost 40 Returns, ms
Solution One	10^{10}	10^{13}	1 compare	40
Solution Two	12,288	10^{6}	1 multiply 1 divide 2 adds 1 compare	3
Solution Three	12,288	10^{9}	2 adds 2 compares	0.2

Solution Three uses only a little more logic per element than the communication-bound Solution One. Its speed is memory-bound, limited by the transfer rate from the serial-access bubble memories.

These three architectures each implement a different algorithmic approach to a single problem, and each one has practical performance limits imposed by a different aspect of the architecture. All three designs, however, utilize large amounts of memory in distinctly non-von Neumann ways to reduce the processing load on a conventional processor.[1]

Future research will be directed toward characterizing these unusual memory architectures and the algorithms that utilize them. We are also exploring other computationally stressing problems in ballistic missile defense to discover logic-enhanced memory architectures which can substantially relieve these problems.

[1] For a more detailed description of several LEM designs applied to the radar deghosting problem, see E. Hatt and H. Ostrowsky, op. cit.