C. K. Wong
P. C. Yue

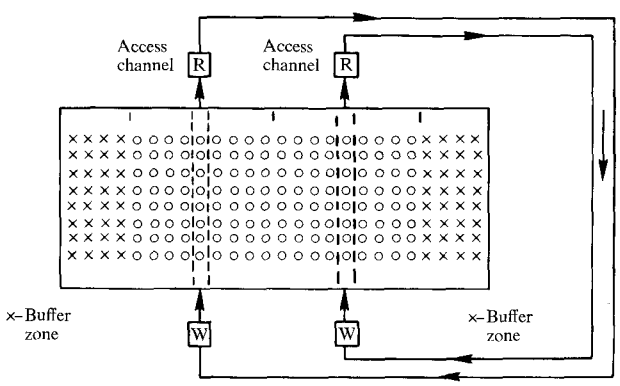
Data Organization in Magnetic Bubble Lattice Files

Abstract: In this paper, we discuss several aspects of data organization in the framework of a recently developed magnetic bubble memory technology known as a bubble lattice file. A dynamic ordering scheme, different from those implemented in conventional T-I bar bubble memories, is proposed. This ordering is particularly suited for bubble lattices. Some performance figures are included for comparison.

Introduction

In the past few years, magnetic bubble memories have become increasingly important due to their potential for nonvolatile, large capacity storage. In addition to having an access time that is two orders of magnitude faster than the moving head disk, the bubble memory has features that can lead to simpler designs in controlling software/hardware and also to more sophisticated methods of data organization. Although there is a large body of literature (see, e.g., [1]) on techniques for making bubble memories faster, cheaper, and higher in density, there has been relatively little discussion of bubble memory data organizations. (For some work in this area, see [2-8].) Furthermore, most of the discussions are restricted to T-I bar bubble memories. In this paper, we examine bubble memories in the light of bubble lattices, which promise much higher density than the traditional T-I bar technology [1]. Our purpose is to contrast the features of bubble lattices with other storage devices and suggest ways to make use of these features.

Figure 1 Conventional BLF.



Magnetic bubble lattices

Bubble lattice files (BLF) operate very differently from T-I bar memories. In BLFs, bit information is encoded in the wall states of magnetic bubble domains, rather than in the absence or presence of bubbles at certain locations. Bit information is decoded by detecting the direction of movement of bubbles at a sensor under a deflecting field gradient.

Bubbles can be accessed through a combination of two operations: 1) lattice translation and 2) column shifting. As shown schematically in Fig. 1, the lattice file consists of an array of bubbles that can be translated as a whole, to the left or to the right, on a piece of magnetic material with buffer zones on both ends. The driving force is provided by an overlaid pattern of current conductors. The pattern of conductors can be sparse compared to that of T-I bar memories, because the lattice is always full and magnetostatic interactions among bubbles tend to keep them aligned with each other. This is one of the major reasons that high density can be achieved in the BLF.

In addition to translation, bubbles in a single column can be shifted out of the lattice at the read (R) end, one at a time, sensed for wall state, and then recreated at the write (W) end. Thus, a column of bubbles can be operated as a shift register. However, shifting can take place only when the column is under one of the access channels. Each access channel consists of a pair of parallel current conductors and is terminated on one side by a read station and the other by a write station (designed R and W, respectively, in Fig. 1).

Data organization for bubble lattices

Given the type of devices just described, we now consider how data are to be organized on bubble lattices for

efficient retrieval. Conceptually, a bubble lattice is in many ways analogous to a moving-head disk file. In a disk, a track is the set of locations at a given head position on a given disk surface, and a cylinder is the set of locations at a given head position.

Since bubble columns are accessed in shift register fashion, each column is analogous to a track, but with latency time and reading time appearing as shift time. The set of columns under all access channels is analogous to a cylinder, and the collection of access channels in a lattice is analogous to the moving head but with seek time appearing as lattice translation time. The BLF differs from the disk in several respects. First of all, the shift is electronically clocked and is thus effectively instantly stoppable, as is a T-I bar memory. However, the shift register is unidirectional as is a conventional disk. To obtain bidirectionality, it is necessary to have a mechanism capable of read and write functions on both ends of the lattice. Another difference is that the records on different tracks of the same cylinder of a disk file move in synchronism, whereas access channels on a BLF are separately controllable. These features of the BLF give rise to flexibility in choosing data organizations, as we shall see below.

Figure 2 shows two kinds of memory organizations in a BLF, namely, the noninterleaved organization and the interleaved organization. In (a) data are stored serially in columns, whereas in (b) a datum is stored in the set of all columns that can be accessed simultaneously by the access channels. Both schemes have similar average latency times, but scheme (b) has shorter read time if a whole block of contiguous data is to be accessed (e.g., in sequential processing). On the other hand, scheme (a) provides more parallel paths for queued requests. Furthermore, if data request lookahead is possible, one can take advantage of the separate control of individual columns under different access channels to line up data for future requests. We refer to [7] for a detailed discussion of anticipatory control features.

In addition to the issue of parallelism, the features of BLF are also reflected in the way data are retrieved. We distinguish between two cases:

1. *Retrieval by address* Because of the stoppability of column shift, it is not necessary to store the address in conjunction with each datum. Instead, we need only a counter to keep track of the current address, i.e., the stopped position of each column. The amount of shift required for an access is then readily obtained by a simple computation. Furtherfore, the risk of over-shifting is negligible compared to the severe timing constraint of disk rotation.
2. *Retrieval by key (associative retrieval)* When the exact address is not available (because of, say, the cost of maintaining indices) or when the address is available but the cost of address counters is not desirable, retrieval by key can be used. Then either the key or the address must be imbedded into the data. In this case, a comparator circuit is required for identifying the datum to be retrieved (see Fig. 3). Note here that although moving-head disks also contain key fields and comparator circuits, one cannot interleave effectively since the records on different tracks cannot be read concurrently.

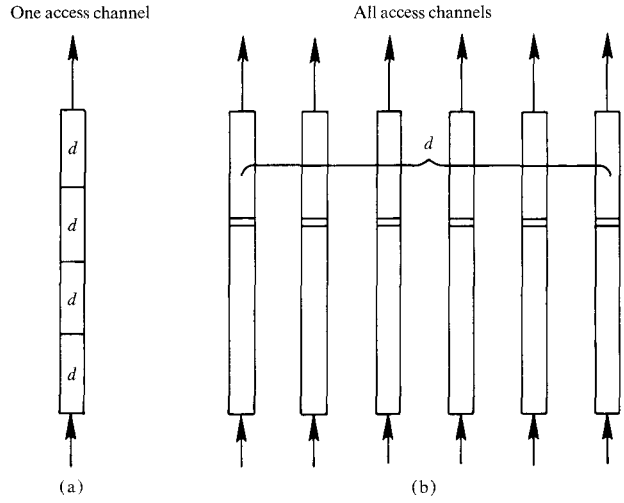


Figure 2 Data organization in BLF. (a) Noninterleaved organization; (b) interleaved organization.

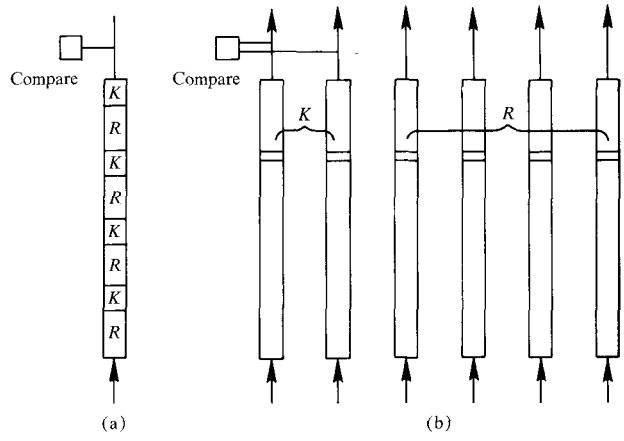


Figure 3 Data organization for associative retrieval (K denotes key, R denotes record). (a) Noninterleaved organization; (b) interleaved organization.

Dynamic ordering in bubble lattices

Another novel feature offered by magnetic bubble memories was first discovered by Beausoleil, Brown, and Phelps [3] and by Bonyhard and Nelson [4] for T-I bar

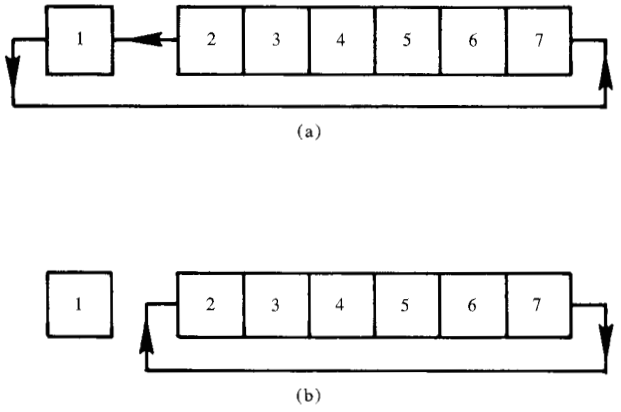


Figure 4 Magnetic bubble memory (a) has seven locations; after accessing the datum at location 1, remaining data can be rotated in the clockwise direction (b).

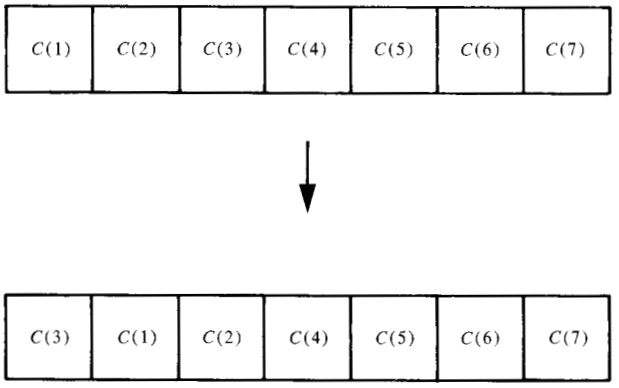


Figure 5 Last Use Ordering.

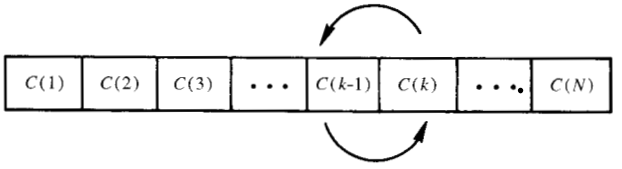


Figure 6 Transposition Ordering.

implementations. Later variations are discussed in [5, 6]. The basic idea is to recognize that bubble memories enable us to alter the relative positions of data with ease. In particular, we can maintain some kind of dynamic ordering among data items in the memory so that the more frequently accessed items are close to the access position.

The scheme proposed in [3, 4] can be explained as follows: Suppose a magnetic bubble memory were represented by a loop with location 1 as the access position, and two operations are available as shown in Fig. 4. The cyclic operation of moving all data to their adjacent locations in a counterclockwise direction is represented at (a), whereas (b) represents the operation in which the datum at location 1 remains intact, but all other data are being circulated in the clockwise direction.

The dynamic ordering to be maintained is graphically described in Fig. 5, where $C(i)$ denotes the datum at location i. After an access to datum $C(k)$ ($k = 3$ in Fig. 5), the datum $C(i)$, $1 \leq i \leq k-1$, is found at location $i+1$, $C(k)$ at location 1, and $C(i)$, $i \geq k+1$, at location i. As a consequence, the most recently accessed data move closer to location 1. This kind of dynamic ordering is referred to as *Last Use Ordering*. If the recently accessed data are the most likely to be reaccessed, then Last Use Ordering will improve access time.

Note here that bidirectionality is essential for implementing such a scheme. Due to the features of the bubble lattices described previously, last use ordering is not feasible for reordering data within a column or for reordering columns among themselves. We demonstrate, however, that another kind of ordering, which is referred to as *Transposition Ordering*, can be used to advantage for reordering among bubble columns so that lattice translation time can be greatly reduced.

Transposition Ordering is an algorithm that approximates ordering by frequency of usage. Suppose we have a list of n items. The one located in the access position is called $C(1)$, the next $C(2)$, then $C(3)$, etc. Transposition Ordering means that every time an access is made to $C(k)$, it changes position with $C(k-1)$ unless $k = 1$. (See Fig. 6.) As a result, the more frequently an item is accessed, the closer it is to the access position. Thus, the average access time tends to be small.

To implement this ordering in BLF, we propose the following modifications to the conventional BLF. First of all, the access channel consists of a shift register that can propagate two adjacent bubble columns at the same time. (See Fig. 7.) Propagation can be accomplished with the same technology as in a conventional BLF (see [12]) by isolating two columns instead of one from the rest of the lattice. Secondly, the access channel is terminated at one end by a pair of write stations (W, W′) and at the other by a pair of read stations (R, R′). The write stations are connected to a switch that has two modes: SET and RESET. In RESET mode, the bubble sensed at R is regenerated at W, thus operating in a manner similar to the conventional BLF (and likewise for R′ and W′). In SET mode, however, the roles of W and W′ are reversed so that the bit that is shifted out through R is shifted back through W′ into the adjacent column, and similarly for R′ and W (see Fig. 8). Thus, in the time taken to shift the full length of a column, the two adjacent columns under the access channel are automatically transposed.

Using the above access mechanism, we specify the operation of a self-organizing BLF as follows.

Addressing scheme The address of each column consists of a sector address and a displacement (Fig. 9).

The sector address, S, is given by the higher-order bits. A sector is defined as the set of bubble columns in a lattice that are accessible by an access channel using a left shift or a right shift. By convention, we assign an even address to sectors on the right hand side of a channel. Therefore, S_0, the lowest order bit of S, provides the translation direction control, whereas the remainder of S identifies the channel.

The displacement d is given by the number of columns between the one being addressed and the one under the channel. Thus the column under the channel has a displacement of zero. Figure 10 is an example of a lattice with two access channels and 16 columns. The addresses of columns from left to right are as shown in the figure.

Access control Every storage reference is serviced in two phases: a translation and an input/output phase. In the translation phase, the desired column is shifted d columns to the left is $S_0 = 0$ and d columns to the right if $S_0 = 1$. The possible operations of the input/output phase are shown in Table 1. At the end of the second phase, the translation is reversed so that the zero-displacement columns are returned to the channel positions. (This last step may not be necessary. If eliminated, however, the translation in the first phase will be given by either the difference or the sum of the new and old values of d depending on whether the value of S_0 has changed. The direction of shift also has to be changed accordingly.)

Note that in this scheme, columns in one half of a sector do not mingle with those in the other half. In each half, Transposition Ordering is maintained. Also, if the column to be accessed is under the access channel (i.e., $d = 0$), then no transposition occurs. Thus if a column is being accessed consecutively, it remains under the channel and no translation of columns is needed.

On the other hand, one can eliminate all the switches and connect the access channels such that they are permanently in the SET mode. If this is the case, then whenever a column is accessed, a transposition between it and an adjacent column always occurs (even for a column with $d = 0$). Consequently, columns in the two halves of a sector mingle, and the more frequently accessed columns are clustered in the middle of the sector.

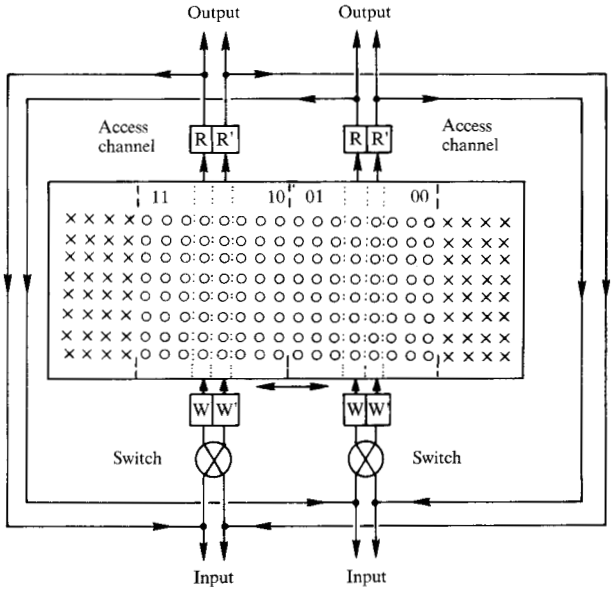


Figure 7 Self-reorganizing bubble lattice file.

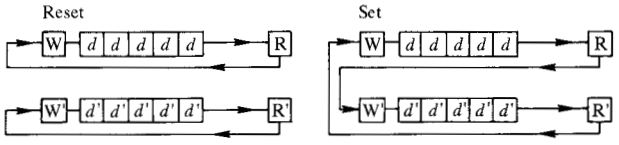


Figure 8 Switching modes.

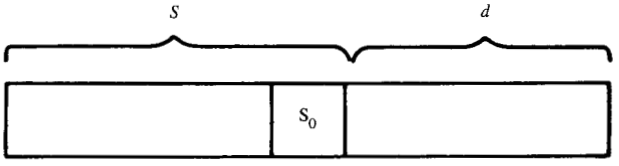


Figure 9 Address format.

Some performance figures

Suppose we have an array of n column locations and a set of n data columns with access probabilities $p_1 \geq p_2 \geq \cdots \geq p_n$, $\Sigma_{i=1}^{n} p_i = 1$, where p_i is the probability for column i. If the exact sequence of accesses is not known beforehand, then the optimal way to allocate data in order to minimize the average access time is to place the column with p_i at the ith location from the access channel, for $i = 1, \cdots, n$, and constantly maintain that order. In this case, the average access time is given by $\Sigma_{i=1}^{n} p_i (i - 1)$. Another extreme is to place the data randomly among the n locations. Then the expected access time is $(n-1)/2$.

In the case of Transposition Ordering, the initial allocation is not important if one is interested in asymptotic average access time, i.e., after a large number of storage references.

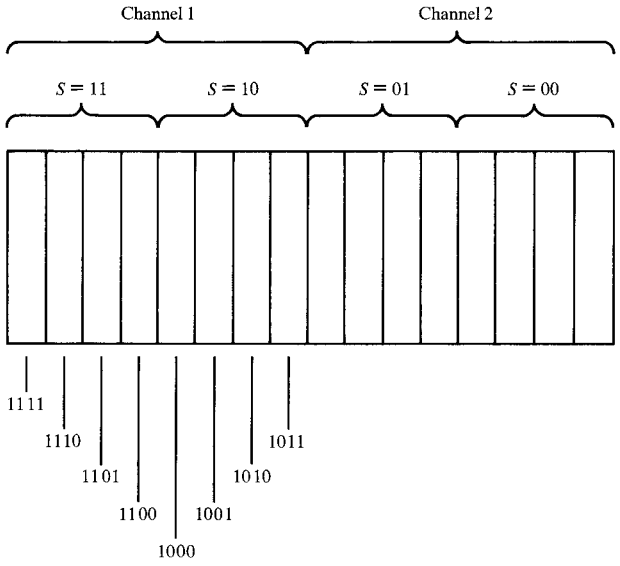


Figure 10 Addressing scheme example.

Table 1 Input/output phase operations.

d	S_0	Output Operation	Output Switch mode	Input Operation	Input Switch mode
0	0	read out through R′	RESET	write in through W′	RESET
	1	read out through R	RESET	write in through W	RESET
	0	read out through R′ while transposing	SET	write in through W′	RESET
				transpose	SET
not zero					
	1	read out through R while transposing	SET	write in through W	RESET
				transpose	SET

Table 2 Anticipated performance.

Columns in sector	Random Ordering	Transposition Ordering	Optimal Ordering
8	3.5	2.226	1.940
16	7.5	4.058	3.733
32	15.5	7.253	6.885
64	31.5	13.048	12.491
128	63.5	23.281	22.505

In many data processing applications, the following assumption is valid [9, 10]: $p_i \propto 1/i$, i.e., the probabilities obey Zipf's law. In this case ([9], p. 398), $\Sigma_{i=1}^{n} p_i(i-1) \approx n/\ln n$.

We include in Table 2 some numerical comparisons that give an indication of the performance merits of the self-organizing BLF. The access time averaged over a simulation period of 100000 storage references, with Zipf's law as the access probability distribution, is seen to be much smaller than random ordering and is close to the optimal. (Access time is measured by the number of columns translated.)

Another way of appraising the asymptotic average access time is to compare it with Last Use Ordering, which has an access time equal to

$$\sum_{i,j} \frac{p_i p_j}{p_i + p_j} - \tfrac{1}{2}.$$

It is shown in [11] that for any given access probability distribution $p_1, p_2, \cdots, p_n$, Transposition Ordering always has a smaller asymptotic average access time than Last Use Ordering, although no simple closed-form expression is available for the former. In the case of Zipf's law,

$$\sum_{i,j} \frac{p_i p_j}{p_i + p_j} - \tfrac{1}{2} \approx 2n/\log_2 n.$$

Thus, if $t(n)$ denotes the asymptotic average access time for Transposition Ordering, then $n/\ln n \leq t(n) \leq 2n/\log_2 n$. Note that the upper bound is about 1.386 times as large as the lower bound. Therefore, Transposition Ordering is always within 38.6% of the optimal. It is easy to demonstrate that the advantage of reordering is even more pronounced if the reference probabilities are more skewed than Zipf's law.

Concluding remarks

In this paper, we discuss some aspects of data organization in BLF. A self-organizing feature based on Transposition Ordering is introduced to suit the special requirements of BLF. This scheme can also be implemented in T-I bar bubble memories without much difficulty. Many variations of Transposition Ordering are possible. For example, instead of interchanging whole columns, one can interchange only portions of columns. This would be particularly suitable for the noninterleaved memory organization described earlier. On the other hand, one could interchange columns at a fixed distance as well as adjacent columns. In this way, one could approximate the ordering by usage frequency much more quickly through a compromise between Transposition Ordering and Last Use Ordering.

References

1. H. Chang, *Magnetic Bubble Technology: Integrated-Circuit Magnetics for Digital Storage and Processing*, IEEE Press, New York, 1975.
2. O. Voegeli, B. A. Calhoun, L. L. Rosier, and J. C. Slonczewski, "The Use of Bubble Lattices for Information Storage," *Proc. 20th Ann. Conf. on Magnetism and Magnetic Materials*, San Francisco, 1974.
3. W. F. Beausoleil, D. T. Brown, and B. E. Phelps, "Magnetic Bubble Memory Organization," *IBM J. Res. Develop.* **16,** 587 (1972).
4. P. I. Bonyhard and T. J. Nelson, "Dynamic Data Relocation in Bubble Memories," *Bell Syst. Tech. J.* **52,** 307 (1973).
5. C. Tung, T. C. Chen, and H. Chang, "A Bubble Ladder Structure for Information Processing," *IEEE Trans. Magnetics* **Mag-11,** 1163 (1975).
6. C. K. Wong and D. Coppersmith, "The Generation of Permutations in Magnetic Bubble Memories," *IEEE Trans. Computers* **Com-25,** 254 (1976).
7. C. K. Wong and P. C. Yue, "The Anticipatory Control of a Cyclically Permutable Memory," *IEEE Trans. Computers* **Com-22,** 481 (1973).
8. D. P. Bhandarkar, "On the Performance of Magnetic Bubble Memories in Computer Systems," *IEEE Trans. Computers* **Com-24,** 1125 (1975).
9. D. Knuth, *The Art of Computer Programming, Sorting and Searching* **3,** Addison-Wesley Publishing Co., Reading, MA, 1973, Sec. 6.1.
10. M. C. Easton, "Model for Interactive Data Base Reference String," *IBM J. Res. Develop.* **19,** 550 (1975).
11. R. L. Rivest, "On Self-Organizing Sequential Search Heuristics," *Commun. ACM* **19,** 63 (1976).
12. B. A. Calhoun, J. S. Eggenberger, L. L. Rosier, and L. F. Shaw, "Column Access of a Bubble Lattice: Column Translation and Lattice Translation," *IBM J. Res. Develop.* **20,** 368 (1976).

Received March 22, 1976

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598.