T. C. Chen
C. Tung

Storage Management Operations in Linked Uniform Shift-Register Loops

Abstract: A new storage structure, called a uniform ladder, consists of a linear array of equal shift-register loops, each holding one record and linked by flow-steering switches. Data exchange across a loop boundary is mandatory if the controlling switch is on and forbidden if off. For MRU (Most Recently Used) storage management, the most important operation is the climbing of data to the top of the ladder from a depth of D loops, which takes only $(D+1)/2$ record periods in the uniform ladder. Program switching is enhanced by efficient schemes for partial environmental exchanges and also by internal block transfers. A pushdown stack can be efficiently implemented by a change in the record storing technique.

Introduction

Although a computer program and its data may occupy a large storage space, it has been observed that, at any given time during execution, only a small subset of the total (commonly referred to as the working set) has a high probability of being needed in the near future. Furthermore, the size and make-up of the working set change only slowly with time [1–3]. This locality of reference can be exploited to reduce average access time by dynamically positioning the members of the working set close to the access mechanism.

Dynamic positioning has been used with great success in the data reassignment between a fast, small cache and a large but slow random access memory. Within each unit, all storage positions are equally accessible.

However, in storage based on shift-registers (such as those using magnetic bubbles and charge-coupled devices) access is governed by simple geometry. Within the same unit there is now a uniform spectrum of access times, capable of spanning two or more orders of magnitude. Dynamic positioning of working sets for access efficiency is feasible, and the resultant multilevel storage can lead to substantial improvement of cost performance.

Recently Beausoleil, Brown, and Phelps [4], and also Bonyhard and Nelson [5], have discussed management schemes in magnetic bubble technology, using ingenious bubble track designs to permit stoppage and reversal of the moving bits by means of stoppage and reversal of the rotating drive field.

Tung, Chen, and Chang [6] discussed a different scheme requiring neither bit movement stoppage nor reversal, using an arrangement with multiple shift-register loops linked by special flow-steering switches. Although the switch design is given in terms of magnetic bubbles, the principle applies to all uniformly driven shift-register storage designs.

We discuss here a new multiloop design using the same switches, having equal-sized loops and being under a more sophisticated switch control. The device is called a *uniform ladder*, and is shown to be very effective in storage management and related applications.

The uniform ladder

A diagram of the steering switch is given in Fig. 1(a). It is a memoryless and practically delayless device with two inputs, A, C, and two outputs, B, D. It has two electronically controlled states. In the avoidance (off) state, inputs A, C are connected respectively to outputs B, D, and the two data streams avoid each other. In the alternative crossover (on) state, A, C are connected respectively to D, B, resulting in crossover of the two streams. Unless stated otherwise, a switch is assumed to be off.

The uniform ladder shown in 1(b) consists of a linear sequence of N shift-register loops $\{L_i\}$, $i = 0, 1, \cdots, (N-1)$, each with a capacity of $2m$ bits. Adjacent loops, say L_i and L_{i+1}, are linked by an internal steering switch S_{i+1}; a top switch S_0 is then used to link the ladder to the outside environment. The collection of switches lies in a straight line, which subdivides the ladder symmetrically into two equal parts.

Any two adjacent loops have opposite flow directions. With no loss of generality, we assume the top loop L_0 to have counterclockwise flow.

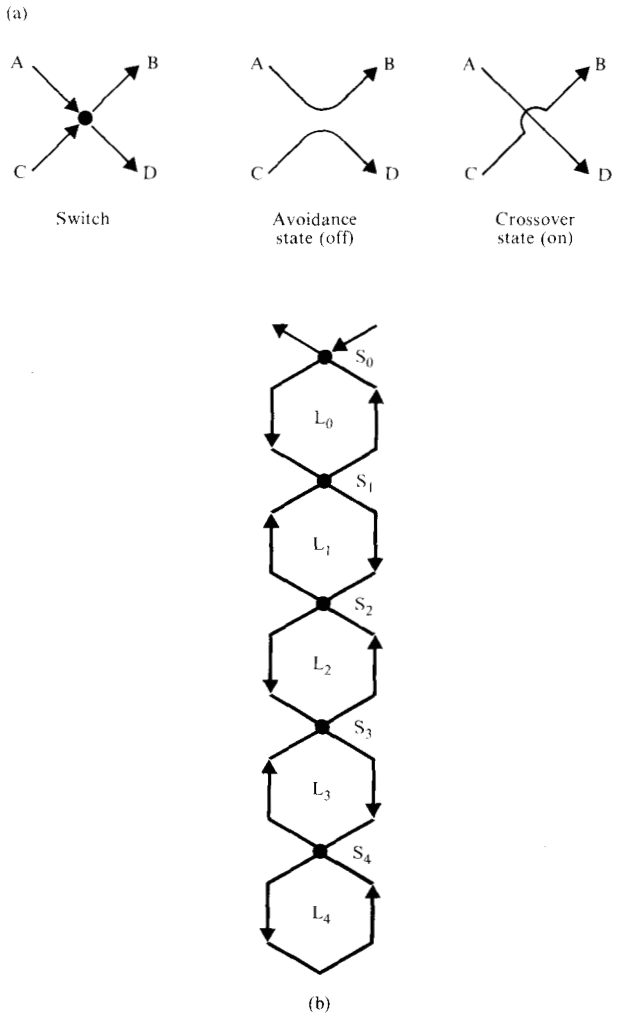

Figure 1 The uniform ladder, consisting of a linear sequence of shift-register loops. (a) Flow-steering switch. Under external electronic control, information streams entering A, C are steered in either of two ways. (b) Configuration of the uniform ladder. Equal sized loops $\{L_i\}$ are linked by switches $\{S_i\}$. Switch S_0 serves as input/output control.

The N-loop ladder will be used to hold N records. Each such record consists of a linear sequence of bits, subdivided into two halves; FR, (front of R) consists of bits r_0 through r_{m-1}, and BR, (back of R), with r_m through r_{2m-1}. The bits r_0, r_m, and r_{2m-1} are called the *head*, *waist*, and *tail*, respectively.

We consider the behavior of the uniform ladder in the following dynamic shifting mode. In one "bit interval," a given data bit will shift from one bit position to an adjacent bit position; if a switch S_i is part of the shift path, the destination position will depend on its setting. If the setting is "off," the data bit will remain in the loop; turning it on will cause the bit to cross the loop boundary. We reiterate the fact that S_i is delayless; its presence in a path does not add to the shift delay. The time for a bit to circulate within a loop is called a *period*; it equals $2m$ bit intervals and is a convenient unit for subsequent discussions.

The detailed positioning of the records within the ladder (or subladder) is called an *arrangement*. It depends on three factors: (a) the starting condition at some reference time t_s, (b) the elapsed time Δt since t_s, and (c) the dynamic setting (on or off) of all the switches $\{S_i\}$.

In discussing the behavior of a single ladder, it should be clear that many (say M) such ladders can operate in unison, and what is called one ladder record here may actually be $1/M$ of a user's data item. The well-known "major-minor loop" bubble storage access scheme, for example, can be invoked, with a ladder serving as a minor loop, so that one bit from each ladder contributes towards an M-bit record, accessible by the major loop.

Idling and neighbor exchange

An "off" setting for S_i effectively partitions the ladder into two uncoupled subladders; an "off" setting for S_0 serves to insulate the entire ladder from its environment.

Regardless of the initial arrangement, if we turn off all switches, the information within each loop will circulate in "holding patterns," hence the same arrangement will recur at the end of each period. This is the analog of "no-operation" for the continuously shifting records, because the information is stationary in the sense that loop contents remain unchanged. The setting with all switches off is called the idle setting even though the duration may not equal an integral number of periods.

Starting from an arrangement with one record per loop, if we turn on exactly one internal switch, say S_i, while all other switches remain off, then the contents of loops L_{i-1}, L_i will flow into each other. After one period, the exchange will be complete; if S_i is then turned off again, the exchanged arrangement will remain. Note that this exchange requires no buffering whatever.

It is clear that any particular assignment of records to loops can be obtained from another by a sequence of exchanges. However, the reordering may be more efficient using alternative schemes, as will be shown subsequently.

Climbing and topping operations

We define the (vertical) distance between loops L_i and L_j to be $|i-j|$. The depth of L_i is the distance between L_i and L_0; it is simply equal to i.

Starting at time t_s, the movement of a record, say R, from L_i to L_j, $j < i$, can be done in $(i-j)$ periods, via a series of neighbor exchanges, using the following settings for $k = 0, 1, \cdots (j-i-1)$:

S_{i-k} is on during $(t_s + k - 1, t_s + k]$, off otherwise.

A similar formula results for $i > j$.

As the record R moves towards L_j, at each loop it interchanges positions with a different neighbor. Consequently, after it has reached the destination, all intervening records initially at L_k, $j \leq k < i$, will have been displaced towards L_i by a distance of one loop.

Such data movement is important in storage management, and is called *climbing* over the distance $(i-j)$. The special case, $j = 0$, of climbing to the top is called *topping*; it has also been termed dynamic ordering [4], or dynamic reallocation [5].

The climbing of a record over distance D, i.e., from L_i to L_{i-D}, can be done in D periods using repeated exchanges. However, an analysis of the data movement requirements shows that the climbing time can almost be halved. Using the following settings for $k = 0, 1, \cdots, (D-1)$, exactly $(D+1)/2$ periods are needed to accomplish the climbing:

S_{i-k} is on during $(t_s + k/2, t_s + k/2 + 1]$, off otherwise.

The slower exchange technique and the fast climbing technique are contrasted for the special case of topping in Fig. 2, where we note that a path to the top involves only one side of each of the intervening loops. The trick in Fig. 2(b) is to fire S_{i-k-1} half a period *before* S_{i-k} is reset to "off;" time is saved by overlapping two crossover settings. The climbing speed is two loops per period, with an extra half-period to complete the action. This is true because the leading bit originally in loop L_i needs only $D/2$ periods to reach the topmost position of L_{i-D}, but at that time only half of the record is in loop L_{i-D}; the remaining half is still in the loop below, taking another half-period to complete the ascent.

For a ladder of N loops, the access of any single record can be done by using topping, the access time ranging from a minimum of zero $(i = 0)$ to a maximum of $N/2$ periods $(i = N-1)$. If accesses are equally probable, the average time would be $N/4$. If the accessed item is to return to the "home" loop, the total time would be $N/2$ periods, equal to the worst-case topping time. As we shall see later the section entitled "Storage management," topping without return is much more desirable.

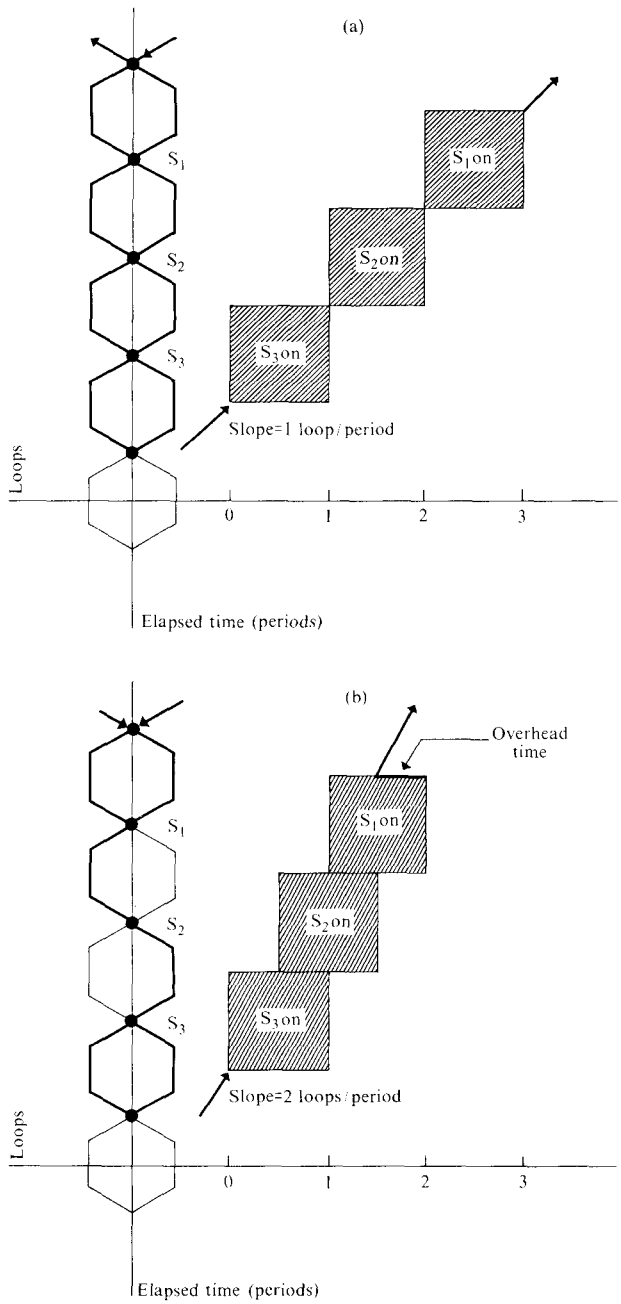

Figure 2 Topping schemes. (a) To top a record at depth D requires D periods by neighbor exchange. (b) Topping the record requires only $(D+1)/2$ periods when redundant paths are avoided.

Loading and unloading

As switch S_i controls the flow into and out of the subladder below it, so does the top switch S_0 control the communication of the ladder with other system components. In order for an external record, say R_j, to reach loop L_j, it must pass successively $S_0, \cdots, S_j$. The minimum requirement is for these switches to be fired in succession at half-period intervals, each individual firing lasting one full period. This is exactly analogous to the fast climbing case, but is a descent from a (hypothetical) environment loop L_{-1}.

When R_j reaches S_{j+1}, and if the latter is set to "off", the record will "bounce" and become filed in L_j a half period later; if S_j is turned off, then the record will bounce

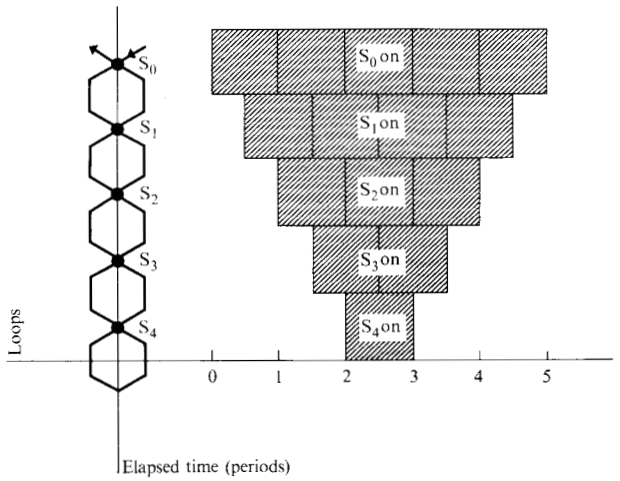

Figure 3 Switch setting during loading or unloading. Loading into the lower subladder of M loops requires turning its top switch on for M periods. All loading is accompanied by simultaneous unloading of the previous contents.

again, and will remain filed as long as both S_j and S_{j+1} remain off. The total time consumed by this filing procedure is $(j+1)/2$ periods.

If R_{j-1} also needs to be filed in L_{j-1}, the same path will be needed, except that S_j should be off when the record reaches it, and S_{j-1} should be turned off at the opportune time. Rather than wait for the R_j loading to complete, R_{j-1} can enter the ladder immediately after the last bit of R_j has entered. $S_0, S_1, \cdots, S_{j-1}$, having been turned on for R_j, can remain on for an extra period to allow the passage of R_{j-1}. When S_j has been turned off just before R_{j-1} reaches it, the latter will be denied access to L_j, and will file in L_{j-1} a half period later; this filed condition will persist if S_{j-2} is turned off immediately afterwards.

This procedure can be extended to fill the entire ladder with a continuous stream of N records, $R_{N-1}, \cdots, R_0$. This is like filling a test tube, the loaded contents serving to reduce the effective length of travel for the new input. The scheme is summarized below for $k = 0, 1, \cdots, (N-1)$:

S_k is on during $(t_s + k/2, t_s + N - k/2]$, off otherwise.

The total time is therefore exactly N periods.

Figure 3 shows the setting sequences to load a four-loop ladder. The diagram is completely symmetric in time; it turns out that the same settings can also unload the entire ladder correctly. The record at the topmost ladder loops is unloaded first; a lower switch need be turned on only at the proper instant for the subladder below it to unload. After N periods, the unloading will be complete.

Both load and unload operations are thus identical, the difference lying merely in user emphasis. *As an outside record is being loaded, a ladder record is simultaneously being unloaded in exchange.* Exchange is thus fundamental to the uniform ladder, not merely in internal data movement, but also in dealing with the environment of other system components.

If a true exchange of records with the environment is to be effected, it is not necessary to unload the ladder first and then reload with new contents. The loading process can simply be done concurrently with unloading, and the total environmental exchange is done in N, rather than $2N$ periods.

We observe that the first record to be loaded sinks to the ladder bottom and is the last record to be unloaded. On the other hand, the last-entered record is the first to come out. Thus the ladder follows the LIFO (last-in-first-out) discipline typical for stacks, without necessarily being a stack itself.

Subladders bordering on the top switch S_0 are called *upper-subladders*. "Partial I/O," namely the environment exchange involving an upper-subladder, follows exactly the same rule as full input/output for a shorter ladder. With S_k turned off to seal off the subladder below, the upper subladder behaves as a full ladder. For most storage applications the most useful partial I/O involves the topmost loop only; the system updates the topmost record by writing into it, incidentally obtaining the old version which can be useful for possible error control. With due care the updating can even be done on part of a record in less than one full period.

The contents of an internal loop must first rise to the top before replacement by external data; this is easily done using the topping operation described in the previous section. We now generalize to the case of moving a contiguous block of p records, not necessarily to the top.

Contents of a subladder consisting of $L_i, L_{i+1}, \cdots, L_{i+p-1}$ can be elevated to occupy $L_{i-D}, L_{i+1-D}, \cdots, L_{i+p-D-1}$ by a sequence of p separate climbing operations, each over a distance of D. The qth operation, for example, allows the contents of L_{i+q-1} to do the climbing. Such a sequence would cost $(D+1)p/2$ periods.

This cost can be reduced to $(D+p)/2$ periods, by starting the $(q+1)$st climb exactly half a period after the qth. The switch settings for this *block climbing* of p records is, for $k = (1-D), (2-D), \cdots, (p-1)$:

S_{i+k} is on during $(t_s + |k|/2, t_s + \min(p - k/2, D + k/2)]$, off otherwise;

and as the block of p records ascends by D loops, simultaneously a block of D records in $L_{i-D}, \cdots, L_{i-1}$ descends by p loop positions.

The input/output of subladders not bordering on S_0 can now be achieved by block climbing, followed by a

partial I/O. An even more efficient method is to use the block climbing algorithm with $D = i$, ignoring the nonexistent switch settings; the subladder of p records will simply climb out of the ladder top. Indeed, the input/output algorithm can be viewed as a special case of block climbing, with $p = D = N$, $i = 0$.

Record integrity

It is important that data manipulations in the ladder do not mutilate (i.e., do not break up into disjointed pieces) any of the records that should eventually be read out in the correct bit sequence.

A record is intact, i.e., not mutilated, if it is filed inside a loop. An arrangement in a subladder with only filed records is designated to be a *filed arrangement*. In order for a record to move across loop boundaries, it must leave the filed state by creating a break, not necessarily at the logical record boundary. If, after the break, bit r_n leads the rest of the record, the resultant sequence

$$r_n, r_{n+1}, \cdots, r_{2m-1}, r_0, r_1, \cdots, r_{n-1}$$

is said to be modulo-serial with offset n. Its apparent head and tail are r_n, r_{n-1}, respecively. The special case with offset 0 is called *strictly serial*, or just *serial*.

Starting at time t_0, N records can be loaded serially into the uniform ladder in exactly N periods. Upon completion time $t_1 = t_0 + N$, the ladder becomes organized as shown in Fig. 4. We note that all records are filed with the front half occupying the right side of the ladder; further, the head bits of all occupants of the odd-subscripted loops are at the bottom, poised for movement downward, and the head bit of all occupants of the even-subscripted loops are on top, ready for movement upward. In particular, the record in the top loop is poised for serial exit. This condition at the top loop recurs at full period intervals, for $t = t_1 + k$, where k is an integer, regardless of the switch settings in the interim.

Thus, the act of loading also serves to initialize the arrangement of all the records in a special, synchronized manner. The integrity of the records can then be preserved by operations that tend to map one such organized arrangement into another. It turns out that any of the non-idling operations, applied one at a time, cause all loop-crossing records to be modulo-serial with the same offset. This has far-reaching implications regarding the global integrity of the ladder file. Further, a ladder so loaded can be subdivided into subladders, each to be handled independently, if due care is exercised in preserving the integrity of the records when the subladders are merged.

The following assertions are provided without proof. Some of the proofs are obvious; the rest will be found in the Appendix.

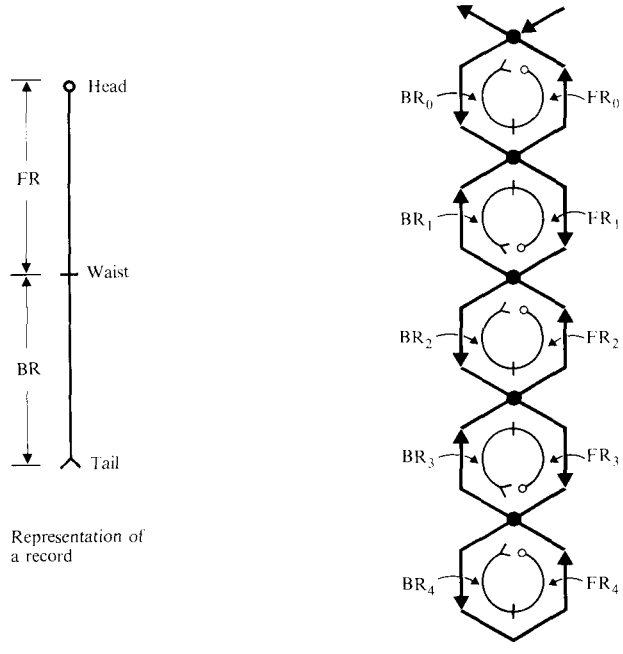

Figure 4 Arrangement of filed records poised for output. The head bit of the record is posed to cross the top switch.

1. The operations defined as

   idle for any duration,
   exchange between adjacent loops ($\Delta t = 1$),
   climbing over distance D(including topping) ($\Delta t = (D + 1)/2$),
   block climbing of p records over distance D ($\Delta t = (D + p)/2$),

   will map filed arrangements into filed arrangements and can start at any instant after t_1, one operation at a time and as many times as needed, as long as the initial arrangement is known to be filed.
2. The load operation for a full ladder can start at any instant, the latter defining a new t_0 for subsequent operations. The completion time will also define t_1.
3. A readout operation will yield serial records for a ladder with intact records if the starting time is $t_s = t_1 + k$. On the other hand, if the readout starts at time $t_s = t_1 + k + (n/2m)$, the record movement will be modulo-serial with offset n. In particular, if $n = m$, the movement will be waist-first.
4. The topping operation for D = odd (even), starting at full (half) periods after input, will occur with the designated record climbing waist-first. The instant of topping completion will be at a full period boundary, and serial readout can start at once. On the other hand, D = even (odd) will lead to serial (head-first) "percolation," and the completion instant will not be opportune for serial readout. The latter obtains by idling for an extra half-period, or better, by starting

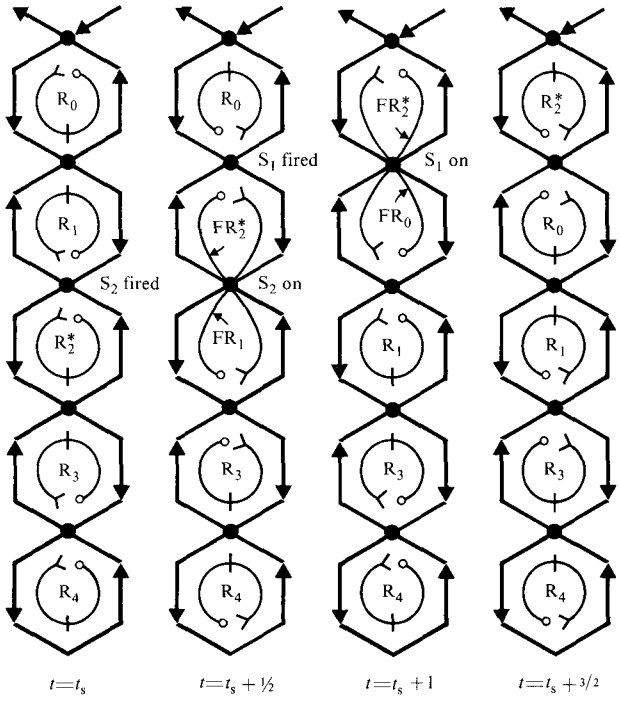

Figure 5 Detail of the topping operation with even $D(=2)$. The record is selected when the head bit is poised to climb. All intermediate steps show head-first movement. Completion time is unfavorable for head-first output, which, however, can be done half a period earlier.

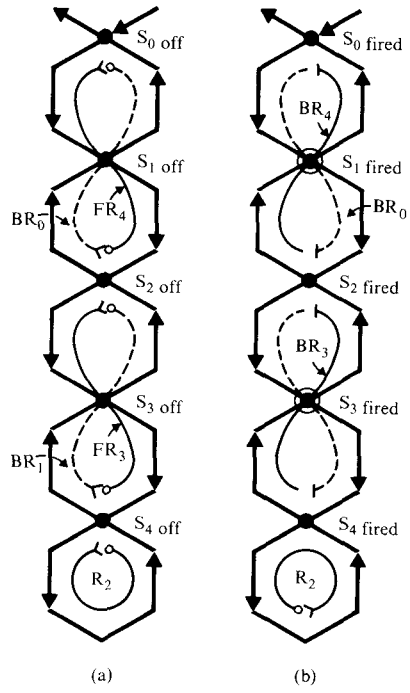

Figure 6 The uniform ladder as a pushdown stack. The lower half of the stack content is shown in dotted lines. (a) Ready for pushdown. $t = t_s + j$. (b) Ready for pop-up. $t = t_s + j + 1/2$. Note the waist-leading exit.

the output half a period sooner (Fig. 5). See Table 1 for details.

5. If these operations are done one at a time within a subladder, the record movements will all be modulo-serial, with no record mutilation. Each operation will result in a filed arrangement. At any time, however, an ongoing operation can be interrupted by the insertion of an idle interval for an integral number of periods, during which time data movement may not be modulo-serial; the interrupted operation can be resumed after an integral number of periods with no visible effect except for the time delay.
6. The setting of some switches to "off" throughout a given time interval effectively subdivides the ladder into uncoupled subladders bounded by these switches and the original ladder boundaries. Operations involving these subladders can be completely independent, except that (partial) I/O can start only an integral number of periods after t_1. Merger of adjacent subladders will be safe if both contain only filed records.

Storage management

Storage access costs can be reduced if data items are arranged dynamically in linear order, based on the relative probability of immediate usage. During execution time, the access pattern reflects changes in these probabilities, and automatic storage management consists in rearranging the records into an order that also reflects these changes.

As mentioned at the beginning of this paper, locality in referencing is a common phenomenon in computing. Hence, in the absence of detailed knowledge of program execution, it is reasonable to expect that the access of a data item should automatically raise its probability for immediate (re)usage; in contrast, the data items previously accorded a higher probability should now have downgraded probabilities. In the arrangement by probability, the accessed data item should move upward and some other data items, previously ranked higher, should drop down.

A very reasonable assumption is that the more recently accessed data item has a higher probability for immediate reuse. Whenever a data item is newly accessed, it acquires the highest probability for immediate reuse, thus should reach the top, while all intervening data items should drop down by one position; in other words, access is always associated with topping. This scheme is called the MRU (Most Recently Used) upgrading algorithm.

Table 1 Fast topping from depth D

	Starting time $t_s = t_1 + k$		*Starting time* $t_s = t_1 + k + \frac{1}{2}$	
Parity of D	even	odd	even	odd
Movement	serial, head-first	modulo-serial, waist-first	modulo-serial, waist-first	serial, head-first
Completion time t_f	$t_s + (D+1)/2$	$t_s + (D+1)/2$	$t_s + (D+1)/2$	$t_s + (D+1)/2$
Time for first serial output	t_f	$t_f - \frac{1}{2}$	$t_f - \frac{1}{2}$	t_f

Incidentally, the same algorithm has been used widely within the context of a two-level hierarchy, each level with equal-access data items. The emphasis there is on purging the least-recently used (LRU) data item from the higher level, and the scheme is called the LRU replacement algorithm, or more precisely, the LRU purging algorithm. In our multilevel case, the top storage slot is unique and is the bone of contention, and the purging is not confined to a single record, but to all intervening data items. The use of "LRU purging" is therefore inappropriate.

The uniform ladder, with a record identified with a data item, is very efficient in topping; it should be an effective device in MRU storage management. "Read from the ladder" should be interpreted as "top and copy," without the need to return the topped record to the "home" loop. Also, "write into the ladder" should mean "top and replace." Replacement is just a partial I/O operation involving one record; copying can be done by attaching a sensing or replicating device at the top loop.

The MRU upgrading algorithm handles only one record at a time and is only moderately efficient during switching between programs involving many records. Here the much more efficient internal block climbing can be invoked, the block in question being the entire new program.

The behavior of a dynamically managed storage using MRU upgrading (LRU purging) is often likened to a pushdown stack [7], but there are major differences. Although the MRU storage, like the pushdown stack, follows the LIFO discipline, there is no real need to do the pushdown and pop-up operations in storage management. On the other hand, topping is the most important operation in MRU management, yet is not even part of the normal pushdown stack repertoire, though included in some hardware implementations as an added feature.

We recall, however, that the use of the uniform ladder discussed thus far is based on the filing of records into individual loops. In this way selected records often can be moved along only one-half of the traversed loops, covering a distance of two loops in one period. Although this scheme, well suited for storage management, can be used to implement the pushdown operation, efficiency cannot be expected. It is easily seen that a pushdown operation on a ladder with the top k loops occupied will take at least $(k+1)/2$ periods, because this involves the topping of the contents of loop L_k, in order to exchange with the environment, and the topping operation already takes $(k+1)/2$ periods.

More satisfactory is a new scheme based on a storage viewpoint distinct from the one-record-per-loop philosophy. In this alternative storage scheme, the entire ladder is treated as a giant, twisted loop. Two adjacent loops, L_{2k} and L_{2k+1}, with $0 \leq k < N/2$, are used to represent pushdown levels P_k and P_{N-k-1}, such that their respective occupants each straddle both loops evenly, with a half record in each loop. If the ladder has an odd number of loops, the bottom loop will represent the middle pushdown level $P_{(n-1)/2}$. When ready for pushing, the contents of P_{N-k-1} will be pointing down, and the contents of P_k will be pointing up, including the middle level. When geared for popping up, the direction of the pointing will be reversed.

The pushdown stack operations are as follows:

- *Pushdown* (loading): Turn on all switches S_0 through S_{N-1} for one period, starting at $t = t_s + j$, where j is a nonnegative integer.
- *Idle*: Turn off all switches S_0 through S_{N-1}, a given arrangement will recur at full period intervals.
- *Pop-up* (unloading): Turn on all switches S_0 through S_{N-1} for one period, starting at $t = t_s + j + (1/2)$, where j is a nonnegative integer.

Figure 6 illustrates these operations on a pushdown stack made of a five-loop ladder. Each pushdown or pop-up for a record takes exactly one period, but the transition from pushdown mode to pop-up mode takes a half-period. It is curious that pop-up is done modulo-serially, waist first.

Summary

We have shown that a uniform ladder, constructed by linking N equal shift-register loops linearly by means of flow-steering switches, can be used to contain and ma-

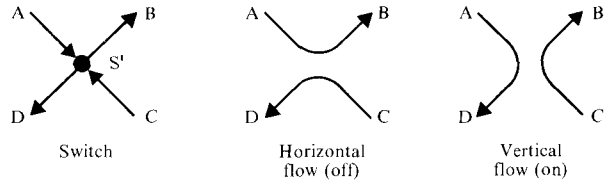

Figure 7 An alternative flow-steering switch, in which there is no crossing of information streams in either state.

nipulate records in various arrangements desirable for computer application. One scheme is particularly attractive in performing the topping operation to consolidate working sets during program execution; another is effective in implementing a pushdown stack.

The block climbing scheme permits the efficient use of multirecord segments in the same ladder and is important in program switching.

Though we have based our discussion on the particular switch in Fig. 1, the type of flow-steering switch shown in Fig. 7 can also be designed, relying on two orthogonal modes of avoidance patterns, with no explicit crossover. However, the analysis of a uniform ladder using these orthogonal switches turns out to be *exactly the same* as the one above, with only the odd-subscripted loops $\{L_{2k+1}\}$ rotated by 180° about the ladder axis, out of the plane of the paper.

The uniform ladder, constructed either way, can probably be implemented in any shift-register technology, although the first design uses the magnetic bubble technology. In any case, the current emphasis on shift registers is for storage applications, for which the uniform ladder seems very well suited.

Our study of the counterflow of contents between adjacent ladder loops has further led to schemes for permutation and is to be reported elsewhere.

Appendix: The preservation of record integrity

• *Integrity of records*

The section on record integrity gives some rules that guarantee the nonmutilation of records within the ladder. We now discuss their origin.

The prevention of file mutilation lies in the prevention of the mutilation of any of the records involved. A record is mutilated if it is not modulo-serial, either because

1. it is broken up into two or more pieces, or
2. it does not obey the modulo $2m$ ordering.

Situation (2) never arises with serially loaded records, and we are mainly concerned with situation (1).

• *Breaks in a record and their immediate remedy*

A properly filed record circulates modulo-serially within the assigned loop; whether it is strictly serial is a moot question. Any attempt to move a record across a loop boundary creates a discontinuity in the storing pattern, making the flow modulo-serial. One more break will mean definite mutilation.

Operations that limit the number of breaks in a record to no more than one are certainly safe. Moreover, all streaming operations within the ladder are reversible; and mutilations can be rectified in principle and may not be disastrous. They do, however, overburden the bookkeeping unless the remedy is applied immediately.

An example of mutilation with immediate recovery is the idling operation. The following two rules deal with the most common idling requirements and can be applied at any bit-time.

Rule 1 (Idling) The switches in a subladder can all be turned off for one period. Any arrangement in the subladder will be guaranteed to recur. Modulo-serial flow on part or all of the subladder will be reinstated despite possible mutilation in the interim.

Rule 2 (Straddle-flip) Given a modulo-serially moving record with the offset n, straddling symmetrically about switch S_i. Then S_{i-1}, S_i and S_{i+1} can all be turned off for a half-period, and the record will remain modulo-serial, but will be moving in the opposite direction with the offset $[(n - m) \bmod 2m]$ at the end of the half-period.

We now give two simple rules to prevent the occurrence of either two or more breaks in a filed record, or one extra break in a flowing record. These are the only cases that can cause mutilation of modulo-serially flowing records.

Rule 3 (Nontearing) If loop L_i contains a filed record, S_i and S_{i+1} should not be fired within the same bit interval. Otherwise, parts of the record will flow upward while other parts will flow downward, creating two breaks.

Rule 4 (Nondiversion) If a record is flowing across a loop boundary, neither of the two switches along the flow path should change its value until the apparent tail has passed it. Otherwise, an extra break is certain to occur.

Although these rules can be violated if mutilations are subsequently remedied, it is advisable that rules (3) and (4) be broken only by the idling rules (1) or (2). This gives enough latitude for data movement with minimum bookkeeping effort.

• *Synchronization*

The N records in a subladder are said to be synchronized with time lag n, calibrated at the reference time t_1 if, at $t = t_1 + (j/2m)$, the bits r_{n+j} are all poised for crossover, those in even-subscripted loops for crossing upward, those in odd-subscripted loops for crossing downward.

Synchronism is distinct from intactness of the individual records not only because synchronism is a property involving all records, but also because records can be synchronized yet mutilated. In addition, records can be modulo-serial but unsynchronized. Synchronism is a constant of the motion in a ladder; once a subladder is synchronized, it remains so with the same time lag as long as no input-output action occurs to disturb it.

Two subladders are said to be in tune if both are synchronized, for the same calibration time with equal time lag. If the subladders also border on the same switch S_i, then they can be considered to belong to the same synchronized subladder formed by the union of the two.

All properly loaded subladders are automatically synchronized, with zero time lag, as calibrated at load completion time t_1. Further, all records are filed and are modulo-serial. Intactness is preserved, if breakout occurs at time $t_1 + k$, where k is an integer. The records are then all modulo-serial with offset 0, adequate for strictly serial output exit; if breakout occurs at time $t_1 + k + 1/2$, movement will be modulo-serial with offset m, (waist-leading).

• *Safety in a ladder*

We now summarize general statements about the entire ladder.

1. A subladder file is intact if all records therein are modulo-serial.
2. A ladder file is intact if all the subladders contain intact files.
3. An intact subladder file remains intact during the application of Rules (3) and/or (4). It may be mutilated by application of Rules (1) and (2), but intactness reappears after the operations.
4. A properly loaded ladder is intact, and synchronized with zero time lag relative to the loading completion time. All unloading starting at time $t_1 + k$ with k an integer, will be strictly serial.

The rules in the section on record integrity are derived from the statements in the Appendix.

References

1. L. A. Belady, "A Study of Replacement Algorithms for a Virtual-Storage Computer," *IBM Syst. J.* **5**, 78 (1966).
2. E. G. Coffman, Jr. and P. J. Denning, *Operating System Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1973, Ch. 7.
3. R. L. Mattson, J. Gecsei, D. R. Slutz, and I. L. Traiger, "Evaluation Techniques for Storage Hierarchies," *IBM Syst. J.* **9**, 78 (1970).
4. W. F. Beausoleil, D. T. Brown, and B. E. Phelps, "Magnetic Bubble Memory Organization," *IBM J. Res. Develop.* **16**, 587 (1972).
5. P. I. Bonyhard and T. J. Nelson, "Dynamic Data Reallocation in Bubble Memories," *Bell System Tech. J.* **52**, 307 (1973).
6. C. Tung, T. C. Chen, and H. Chang, "Bubble Ladder for Information Processing," *IEEE Trans. Magnetics* **MAG-11** 1163 (1975).
7. For a discussion of the pushdown stack see, for example, H. Stone, *Introduction to Computer Organization and Data Structures*, McGraw-Hill Book Co., Inc., New York, 1972, Sect. 6.3.

Received February 24, 1975

The authors are located at the IBM Research Division Laboratory, Monterey and Cottle Roads, San Jose, California 95193.