Working set restoration—A method to increase the performance of multilevel storage hierarchies

by PETER SCHNEIDER
Siemens AG
Munich, Germany

ABSTRACT

The emergence of new storage technologies such as Charge Coupled Devices (CCD) and Bubbles with access times which lie in the access gap between semiconductor memories and rotating magnetic storage media is another important step toward implementing multilevel storage hierarchies.

However, a comparison between a three-level storage system, consisting of cache, page buffer and CCD main memory, and a conventional two-level main memory system will show that the three-level hierarchy using the transfer on demand strategy has an effective access time which is higher by about a factor of 2.

Yet, through better use of the program locality the access time of a three-level system can be reduced to that of the two-level cache/page buffer system. Using this method, the so-called working set restoration, the working set of pages of the next program to be run is loaded into the page buffer during execution of the active program. The required page transfer operations are executed concealed and are thus not time-critical for the processor. This means that for program processing only the access time to the two-level system becomes apparent.

The advantage of a three-level system of this type lies not so much in the improved performance but rather in the lower costs, since it permits the use of a large-capacity main memory on a technology level which is cheaper by a factor of 2 to 4 as compared with MOS RAM.

INTRODUCTION

In the planning of large computer systems, there is a pronounced trend toward increased use of multilevel storage hierarchies. The first step in this direction was the development of the software-controlled virtual memory.

The emergence of the various semiconductor technologies such as the bipolar and MOS technologies, which brought about random access memories of various densities, speeds and costs, led to the development of directly addressable two-level main memory structures. These consist of a cache designed in fast and hence expensive bipolar technology and a main memory designed in cheaper, albeit slower, MOS technology; see Figure 1. By making use of the locality of the active programs, the effective access time of the main memory system can be reduced to nearly that of the cache. The introduction of a two level main memory system in machines with a virtual memory was the next step toward a system enhancement through the use of storage hierarchies. Recent developments include new storage technologies such as Charge Coupled Devices (CCD) and Magnetic Bubble Domain Devices (MBD) of which some were announced as products. Their access times lie in the long-existing access gap between the semiconductor technologies used in the main memory and those of the secondary storage media. The per-bit costs are also estimated to fall in between the price categories of the two known technologies.

Because the new storage technologies have an operation mode different from that of the random access storages, they are called Block Access Memories (BAM).1 Whereas with RAMs any bit can be addressed within an equally short time, RAMs require long access times for single bits while, for a sequence of bits, the longer access time occurs only at the beginning with the remaining bits following sequentially at a high data rate.

In the system considerations outlined in the following paragraphs, the CCD technology will be used as an example representative of the above-mentioned new technologies.

CCD TECHNOLOGY—ITS COST AND PERFORMANCE

The operation of Charge Coupled Devices (CCD) is founded on the basic concept of storing information in MOS capacitors in the form of charge packets and to
shift it at the semiconductor surface from one capacitor to the next. This technology is thus suitable for generating shift registers for the storage of analog as well as digital information. The inherent advantages of the CCD technology are that it is based on the tried MOS technology, storage locations can be extremely small and a high production yield is expected. As compared with MOS RAMs, memory densities greater by a factor of 2 to 4 and per-bit costs lower by a corresponding factor are anticipated for the CCD memory chips.

Memory chips implemented in this technology contain randomly addressable shift registers which are closed via read/write terminals. Since, in the CCD technology as well as in the conventional MOS technology, information is stored dynamically, a refresh of the entire information written is needed after several milliseconds. This is also done by the read/write terminal. Due to shift-frequency-dependent transfer losses, which occur when the information is shifted from one capacitor to the next, the length of shift register loops is limited. As an example, lengths of up to 256 bits are believed to be realizable for a shift frequency of 10 MHz.3

For the following system considerations, devices with a capacity of 65,536 bits, a setup of 256 loops with 256 bits each, and a shift frequency of 5 MHz are assumed. The time required for a complete pass of the information stored in a loop is then 51.2 µs; the mean value for accessing an arbitrarily chosen bit is after one half pass, corresponding to 25.6 µs. Memory chips incorporating these characteristics will be available in the near future.

Some system-engineers believe that these new technologies will challenge only drum and fixed-head disk storages, and thus solely consider the possibility that these technologies replace the conventional paging device in the virtual storage system. By contrast, a discussion in Reference 4 is based on the assumption that the use of a CCD main memory within a three-level directly addressable main memory might obviate the need for a paging device.

However, since the CCD technology, viewed price-wise, can probably compete with MOS RAMs but not with secondary-memory technologies, the following question arises: Can costs be cut maintaining the same system performance if a two-level main memory is replaced by a three-level main memory structure with a CCD main memory, assuming that the virtual storage concept with a conventional paging device is retained?

VIRTUAL STORAGE SYSTEM WITH THREE-LEVEL MAIN MEMORY

To clarify this question, we must first look at the functions performed by today's main memory.

To ensure efficient utilization of the central unit, it is necessary to keep the current pages, the co-called working set4 of several programs in the main memory even if processing of these programs is interrupted due to secondary storage accesses. If, for instance, program pages are missing which have to be fetched from the paging device and entered in the main memory, processing of the active program has to be interrupted. The page entry takes several milliseconds during which time the central processor would have to wait with nothing to do if no other executable programs were available.

However, if several processes are in main memory, the central unit moves on to another process if the active process has to be displaced. Among the processes

---

Figure 1—Two level working memory hierarchy with paging device.
kept in main memory are those waiting for a page to be entered from the paging device. This ensures that, upon reactivation, processes will not have to be deactivated because pages are missing which were overwritten just before.

This holding of working sets of several programs plus the necessity of also having some operating system routines reside in main memory already require main memory capacities of 1 MByte or more for today's timesharing or multiprogramming environments. Due to the mounting storage requirements of individual users as well as the increased use of the installations for multiprogramming applications, an increased memory capacity of over 16 MByte will be needed.

This very fact presents a strong argument in favor of using a cheaper memory technology in the main memory. However, measures must be taken to guard against a loss of system performance. Due to the CCD memory access time, which, compared with random access working memory, is longer by several orders of magnitude, an outright replacement of the main memory is not possible because resident operating system routines must remain readily accessible and the time of access to data of active programs must likewise not be adversely affected.

Considerable savings can, however, be achieved in terms of expensive random access storage capacity by dividing the main memory capacity into two categories using the following criteria:

1. A large capacity memory, which will continue to be called main memory and is designed in CCD technology, for storing non-active working sets;
2. A small-capacity random access memory, the so-called page buffer containing the currently active process as well as the resident and a small number of exchangeable operating system routines.

A strategy equivalent to the virtual memory's paging on demand might also be used here as a load strategy between main memory and page buffer. Pages could then be transferred to the page buffer each time they have been found missing by a built-in hit/miss logic. This method will be referred to as transfer on demand.

This division of the main memory results in a three-level main memory system consisting of cache, page buffer and main memory; see Figure 2. Originally, this division was made under the aspect of cost advantages alone, but we have to examine now whether a procedure of this kind will not result in a loss of performance vis-a-vis a conventional system.

**TWO-LEVEL AND THREE-LEVEL MAIN MEMORY SYSTEM PERFORMANCE COMPARISON**

For a comparison of the performance rate of different main memory systems, we use the effective access time for read operations

\[ T_{eff} = h_1 \cdot t_1 + (1-h_1) \cdot [h_2 \cdot t_2 + (1-h_2) \cdot t_3] \]

since only read operations are time-critical whereas write operations can in general be run concealed while the processor is busy. \( h_1 \) and \( h_2 \) are the hit probabilities for read accesses, \( t_1, t_2, t_3 \) the access times to the different hierarchy levels.
Let us first consider the two-level memory hierarchy cache/MOS main memory. Assuming
\[ h_1 = 95\%, \quad t_1 = 100\text{ ns}, \quad h_2 = 100\%, \quad t_2 = 1000\text{ ns}, \]
the effective access time \( T_{eff} \) will be 145 ns. By comparison, a longer mean access time of 250 ns results for the three-level hierarchy with CCD main memory, page buffer and cache, when \( h_1 = 95\% \) and \( t_2 = 51.2\text{ ns} \).

Since the time of access to the CCD memory \( (t_2) \) is much greater than that of the page buffer \( (t_1) \), the effective access time of the overall memory is extremely susceptible to changes in the page buffer miss rate. If this increases heavily, the effective access time assumes values that are too high to be tolerated.

These considerations lead to the tentative conclusion that under adverse conditions the performance of the three-level hierarchy will be worse than that of its two-level counterpart when using the transfer-on-demand strategy assumed here, which works on the principle that pages are fetched from the main memory only after a page miss has occurred. In order to allow the cost-saving three-level hierarchy to be used without degradation, some means must be found to either prevent or at least significantly reduce page misses, which are responsible for longer access times. This may be achieved by use of the working set restoration method.

**WORKING SET RESTORATION**

This term will be used to designate a method designed to provide a program, upon its reactivation, with the specific memory environment it requires for optimum processing. This is where the locality of a program, referred to as working set of pages, comes in. Under the conventional method, deactivated programs are, upon their reactivation, provided with that number of page frames for entering the pages which corresponds to the actually required number of pages during the preceding processing interval. In contrast to this, the method proposed here attempts to retrieve the actual pages addressed in the preceding interval rather than mere page frames.

A similar method is discussed by Tung who suggests that shorter transfer times between two memory hierarchy levels be achieved by interleaving the main memory in as many ways as there are page frames in the page buffer, i.e., that the number of main memory modules should match the number of page frames in the page buffer. The interface width is equated with the page size. Nevertheless, there are processor wait times under this system during process changeover until a new process can be started. Besides, this concept would require a main memory with an extremely large capacity and the use of very wide interfaces would also pose a problem. Further, it should prove difficult to ensure an optimum distribution of all active pages over all possible memory modules—a necessity if the method proposed by Tung is to work properly. For these reasons, implementation of such a system cannot be advocated.

Since it does not make any difference in the time total whether all pages are transferred consecutively or individually upon request, it is suggested here to load the working set of a program concurrently with the processing of another process so that it does not occur time-critically for that process. To allow this, the page buffer capacity must be expanded by an additional area so that one area will always be available for program processing and another for loading the working set of the successor process. Further, the time available for concealed loading must be sufficient. However, it can be safely assumed that it is, because it is made up of the runtime section of the active program and the operating system execution time for the subsequent process change.

Through the use of this procedure, the access time of the three-level system is reduced to the access time of the two-level cache/page buffer system since the majority of page misses were forestalled in a non-time-critical manner. The third level will have to be accessed only when pages which reside in the main memory but not in the page buffer are added to the working set of the active program.

**SYSTEM CONFIGURATION**

Having explored the theories and facts suggesting that the use of the working set restoration concept as a load strategy between page buffer and main memory seems practical, let us proceed to discuss a sample system configuration and its management.

The memory system, Figure 3, has a cache with generally known characteristics and of conventional size at the top level. Since the cache is independent of the load strategy used between second and third level, it is excluded from the following discussions.

The 512 KByte \((K=1024)\) page buffer is divided into four modules: Modules 1 and 2 are designated for program processing or for loading the successor process working set; Module 3 contains resident operating system routines; Module 4 contains various routines of the exchangeable operating system.

Each of these modules has a capacity of 128 KByte. At any given time, the active program is present in only one of the two modules slated for processing. Both modules contain the page buffer addresses from 0 to 31 \((\text{page size } 4\text{ KByte})\). This ensures that, during processing of a program in, say, module 1, the successor process can be entered in module 2 without affecting program execution. Besides this, the allocation of main memory and page buffer addresses is simplified in so far as all main memory addresses are always allocated to page buffer addresses 0 through 31.

The page size of 4 KByte is identical to that used in the virtual memory system. The storage technology employed in the page buffer must be able to accommo-
date the data rate transferred from the CCD memory over a data path with a width of, say, 8 Bytes.

A CCD memory of modular design is assumed as main memory: 144 CCD memory devices with a capacity of 65,536 bits contribute one bit each to a 16-Byte word which is transferred via the 8-Byte interface using the two-way streaming method. Consecutive words of a 4 K-Byte page are successively stored in the CCD loops; thus a page can be read out or written in 51.2 μs with one parallel cycle of all 144 loops. This corresponds to a data rate of 80 MByte/s. The module comprises 256 pages, i.e., it has a total capacity of 1 MByte, Figure 4. Depending on the desired system performance, the CCD main memory can be implemented with capacities from 1 MByte to 16 MByte or more with the aid of these modules.

A hit/miss logic implemented in the form of an associative memory is visualized as a means of checking whether the pages requested by the processor are entered in the page buffer. The associative memory always describes the specific page buffer module (1 or 2) containing the program (addresses 0 through 31), and module 4 containing the operating system (addresses 32 through 63). This ensures that the entries, by virtue of their being located in the hit/miss logic as they are, point to the page buffer address. Another section of the hit/miss logic—again with addresses 0 through 31—will be used for storing the working set table of the successor process. In addition to the main memory address entries, there will be a write bit which indicates whether a page entered in the page buffer was modified and therefore no longer match the original in the main memory.

In the system discussed here (cf. Figure 3), the exchange of pages between main memory and paging device is not performed via the central processor as in so many existing systems, but rather via a storage processor (SCU) assigned to the memory. This processor has its own memory for storing the pages to be exchanged between main memory and page buffer until they can be entered without interfering with the processes taking place between main memory and page buffer. In addition, the storage processor is responsible for loading the working set of the successor process.

After this overview of the system’s hardware components, let us now turn to the specific software requirements for working set restoration.

Each process will have a table assigned to it describing these pages (up to 32) that were entered in the page buffer by that process during the most recent processing period. These tables are stored in the memory of the storage processor according to their internal process code numbers and can be directly addressed by means of these numbers.

In the order to start the working set restoration procedure, the process requiring loading operations must be known to the storage processor. This informa-
tion is extracted from the process queue maintained by the operating system and is passed by the central processor to the storage processor as soon as this interrupts a program and changes over to another one. This information enables the storage processor to control working set restoration proceedings.

FUNCTIONS

We will now discuss page miss handling and activities involved in working set restoration, Figure 5.

Page miss

After address translation in the central processor, a request address is sent to the memory system. In most cases, an access can already be satisfied in the cache. If a cache miss occurs, the address is switched through to the page buffer. The hit/miss logic then determines whether the requested address is present in the page buffer. If the desired page is not available there, it must be fetched from main memory. This is done by switching the address through to the main memory, after which page transfer is started. A processor-requested word within a page is switched through to the processor during page entry and, on the basis of the foregoing, will be received after a mean time of 256 μs.

In addition to pages which were read only, the page buffer also contains pages which were modified. Since these pages no longer match their original in the main memory, they must be written back to main memory before being overwritten. If both the requested page and the page to be written back are contained in the same module, this may require two page transfer times. In order to avoid such displacement procedures with page miss, one buffer page is always reserved into which the requested page can be loaded immediately. If the page released for overwriting by the replacement algorithm was modified, it is subsequently written back in non-time-critical fashion to main memory, and then becomes a new buffer page. Thus, 32 pages are active in the page buffer at all times. Yet, since one of these pages is being marked free, a page entry due to page miss can be handled in the shortest possible time.

Working set restoration

A working set is to be transferred from the CCD main memory to the page buffer whenever an active process is displaced. The central processor immediately switches over to the next process while the storage processor starts loading the working set of the next executable process; see Figure 5. The storage processor first reads out the 32 hit/miss logic entries describing the page buffer contents of the program just displaced, and updates the associated page table. In the course of this, it determines which pages have to be displaced from the page buffer and simultaneously checks to see, by way of a comparison with the successor process page table, whether any successor process pages fall into other modules than the pages to be displaced.

Figure 5—Flow diagram of the working set restoration activity
The page buffer and main memory addresses are then switched through by the storage processor to the memory control, which is instructed to handle page transfer. Transferred first are those pages that can be entered in the page buffer simply by overwriting unmodified pages.

Transferred next are pages which lie in other modules than the pages to be displaced. This is done in the following manner: the memory locations are read out of the page buffer in the Read/Update/Write mode and transferred to the main memory, while data are simultaneously written in from another main memory module.

The last pages to be entered are the ones which lie in the same module as the pages to be displaced. Here too, the data are entered in the page buffer via Read/Update/Write, but the page to be displaced from the page buffer must first be transferred to the storage processor for intermediate buffering. Pages of this type are entered in main memory after working set restoration has been completed.

Concurrently with page loading, the main memory addresses of the respective pages are entered in the currently unused portion of the hit/miss logic at a location corresponding to their page buffer address such that, at process change-over time, not only will the pages be in the page buffer, but the pertaining address information will also be stored in the hit/miss logic.

After this discussion of the three-level storage hierarchy's structure and functions, let us now examine its capability as compared with other systems.

CAPABILITY

To measure the capability of the three-level memory structure, a simulation program was created which allows an examination of the page miss rate in the page buffer for the transfer-on-demand strategy as well as under working set restoration. The different handling of page fault interrupts as compared with other interrupts has been taken into account. When a page fault occurs, all pages already entered in main memory are retained, whereas with all other interrupt causes the number of pages found after reactivation will be less. This is where the size of the main memory comes in; it can be calculated as follows: Assume that 100 users are concurrently attached to a computer system. Five of these are to be in the active process queue at any given time. Based on measured values, a mean working set size of 100 KByte can be assumed, which adds up to a memory capacity of about 500 KByte for the processes in the active queue. The remaining 95 users share the rest of the main memory capacity. If main memory capacity is 4 MByte, each process will find an average of 40 KByte available upon its reactivation. If each process is to find a greater capacity available, the main memory capacity must be increased if the length of the process queue remains the same. A greater main memory capacity will then result in increased capability since pages have to be overwritten less frequently in main memory and some of the page faults for previously used pages can be prevented. This is taken into account by the simulation program.

The analysis covered address sequences of several different programs. Table I uses two typical address sequences (program A and program B) to demonstrate the impact of main memory size on the page faulting rate and the page miss rate in the page buffer for both load strategies.

These results yielded by simulation were used for calculating the effective access times of useful processor accesses. The term "useful access" is employed to define only the number of accesses used for actual program processing, whereas the time needed for accesses during operating system activity is viewed as waiting time and is thus prorated to the useful accesses. Using this effective access time, the systems illustrated in Figures 1, 2, and 3 were analyzed and compared. The effect of the cache was ignored because it is the same in all systems.

The following formulae yield the effective access times for useful accesses.

For a two-level system

\[ T_{\text{eff}} = h_{\text{pf}} \cdot t_{\text{int}} + h_{\text{int}} \cdot t_{\text{int}} + (1-h_{\text{pf}}-h_{\text{int}}) \cdot t_{\text{int}} + t_{f} \]

and for a three-level system

\[ T_{\text{eff}} = h_{\text{pf}} \cdot t_{\text{int}} + h_{\text{int}} \cdot t_{\text{int}} + (1-h_{\text{pf}}-h_{\text{int}}) \cdot t_{\text{int}} + t_{f} + m_{f} \cdot t_{f} \]

The same formula applies for the three-level systems which differ only in the load strategy. Depending on the load strategy, different miss probabilities \( m_{f} \) or \( m_{n} \) must be expected in the page buffer. \( h_{pf} \) and \( h_{int} \) represent the page faulting and interrupt rates, respectively; \( m_{f} \) is the miss probability in the page buffer and \( t_{\text{int}} = 1\, \text{ms} \) represents the operating system execution time for process changeover. \( t_{f} = 1\, \mu\text{s} \) is the access time to the second level of the two-level system, and \( t_{t} = 160\, \text{ns} \) and \( t_{p} = 51.2\, \mu\text{s} \) are, respectively, the access times to the second and third levels of the three-level system. Figures 6a and 6b show the cost curves of the compared systems: cost

<table>
<thead>
<tr>
<th>Program</th>
<th>Main memory capacity (MB)</th>
<th>Interrupt frequency (h_{pf})</th>
<th>Page fault transfer on demand (h_{int})</th>
<th>Page miss rate (m_{f})</th>
<th>Working set restoration (m_{n})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>0.07</td>
<td>1.67</td>
<td>4.66</td>
<td>0.93</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.07</td>
<td>1.27</td>
<td>4.51</td>
<td>1.13</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.07</td>
<td>0.93</td>
<td>4.41</td>
<td>1.30</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>0.07</td>
<td>0.58</td>
<td>0.61</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.07</td>
<td>0.24</td>
<td>0.60</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>0.07</td>
<td>0.18</td>
<td>0.60</td>
<td>0.01</td>
</tr>
</tbody>
</table>

From the collection of the Computer History Museum (www.computerhistory.org)
curve C₂ applies for the two three-level systems differing only in their load strategy, while C₁ represents the conventional two-level system. As is clearly demonstrated, the costs of the three-level main memory are substantially lower. The figures also plot the effective access time curve of “useful access” for all comparison cases as a function of main memory capacity.

The effective access time of a three-level system with working set restoration T^WSR is almost equal to that of the two-level system T^eff1, Figure 6a, or can, according to Figure 6b, even be lower if no page buffer misses occur and the page buffer’s significantly shorter access time, as compared with a conventional main memory, can thus be made the most of. The effective access time of a system with transfer on demand T^TOD, on the other hand, is always less favorable.

CONCLUSION

Through the use of the working set restoration strategy, a three-level system can in fact always achieve nearly the same performance level as a conventional two-level system, provided that systems with equal main memory capacity—points 1 and 2—are juxtaposed. If, on the other hand, systems with equal cost levels are compared, the three-level system will always offer a more favorable effective access time (cf. points 1 and 3) due to the lower page faulting rate resulting from the greater main memory capacity.

ACKNOWLEDGMENT

The work has been supported under the Technological Program of the Federal Department of Research and Technology of the FRG. The author alone is responsible for the contents.

REFERENCES