Logic-based eDRAM: Origins and rationale for use

R. E. Matick
S. E Schuster

The IBM logic-based eDRAM (embedded DRAM) technology integrates a trench DRAM (dynamic random access memory) storage-cell technology into a logic-circuit technology, merging the two previously separate technologies. Since its introduction in the 1970s, the DRAM technology has been driven by cost while the logic technology has been driven by speed, leading to an ever-widening gap between slower memory and faster logic devices. That has led to the need for increasingly complex levels of memory hierarchies, resulting in considerable degradation of system performance despite many design and architecture compromises. DRAM can provide six to eight times as much memory as SRAM (static random access memory) in the same area, but has been too slow to be used at any cache level. Our studies, highlighted in this paper, indicated that the use of logic-based DRAM could resolve that difficulty—and was necessary for integrating systems on a chip. This has led to the inclusion of logic-based eDRAM as a memory option in the IBM ASICs (application-specific integrated circuits) product.

Introduction

The principles on which the logic-based eDRAM advantages are based are easy to understand. Most important is that the density advantage of DRAM permits replacement of the same area of SRAM with DRAM,[1] which is from four times up to as much as eight times larger in capacity [1]. A factor of 16 to 20 or more in DRAM vs. SRAM capacity per unit area is obtained if DRAM is designed, in the traditional way, for achieving optimum density. The factor decreases to approximately 4 to 8 when the DRAM is designed for improved speed because of the need for shorter bits lines, faster sensing, etc. For cache applications, such as for an L2 cache, the miss ratio decreases approximately as the square root of the capacity increase. Hence, the use of a cache that is larger by a factor of 4 leads to a decrease in the miss ratio by a factor of 2; some relevant studies are described later. Fewer misses to an on-chip cache result in a decrease in the number of cache reloads needed from the slower, external (off-chip) memory system. The larger-capacity on-chip DRAM may not provide a performance improvement if the access speed is too slow. However, the DRAM need not be as fast as the replaced SRAM, but must be faster than a typical DRAM.

Since DRAM was designed for density, not speed, it was clear that a substantial speed improvement should be possible. Questions that we decided to address were how much of an improvement should be possible and how much would be required. In a memory hierarchy, the combination of the miss ratio and the first access time is most crucial for achieving performance. Since this relationship is nonlinear, with many contributing factors, some memory hierarchy analysis was needed in order to determine what speed improvement should be possible. The net result of the analysis was that if the DRAM speed could be made sufficiently fast, its larger capacity should enable it to outperform SRAM over a wide range of cases. A brief discussion of the fundamentals of cache reload and the associated performance impact on a simple system is presented later as an introduction to memory

[1] SRAM uses six field-effect transistors that retain the stored information as long as power is maintained, and is nondestructively read, requiring no rewrite; DRAM uses one field-effect device and one capacitor and is destructively read, requiring write after read, and also requiring refreshing because of capacitor charge leakage.

0018-8646/05/$5.00 © 2005 IBM

hierarchy analysis. First, we present a historical perspective of the evolution of eDRAM and how the industry unknowingly generated a need for it.

Historical perspective

One of the most serious and fundamental limitations on system performance during the late 1980s and 1990s was the growing "speed gap" between logic and DRAM memory technologies. Ideally, the memory system attached to a processor should be large enough to contain the operating system and several complete applications, both code and data, and be randomly accessible in one processor cycle. This was true in some early computers, but it is not generally possible now, because operating systems and applications have increased rapidly in size as processors have become faster. This results from the fact that a fast processor can process more data in a given period of time and hence requires larger amounts of data and programs in order to stay busy. Thus, as processors have become faster, their attached "main memory" capacity has increased, requiring increasingly more processor cycles for an access (see Appendix A for an expanded discussion). The relatively inexpensive "main memory" portion of a system (its DRAM) has become increasingly remote from its processor, requiring an ever-increasing number of levels of cache to bridge the gap. This has increased not only system complexity but also cost, and has prevented the incremental performance gains from being as large as they could otherwise have been.

During the period in which the speed gap was growing, the remoteness of main memory was due to two trends—the increasing gap between logic and DRAM technologies and the ever-increasing size of the required DRAM. The increasing electrical and logic paths associated with the latter resulted in increased delays. Both added to the total memory access delay for reloading a cache miss, and thus had a significant impact on system performance. (See Appendix B for a discussion of this impact.) The primary emphasis for reducing delays was on overall system architecture, primarily because closing the technology gap would have increased the cost of DRAM (now regarded as a "commodity"). While the gap between logic and DRAM was obvious and well known, its significance with respect to overall system performance and possible solutions was not fully appreciated.

The value of having logic of various kinds on a DRAM chip is a relatively old concept. For example, as early as 1976, proposals were studied for placing directory translation directly on main memory chips in order to eliminate page tables [2]. Also, early in the evolution of DRAM, improvements in data transfers into and out of DRAM chips were achieved by various forms of page mode buffers, too numerous to discuss here.[2] Video RAM [3, 4] was a somewhat more advanced and specialized means for improving DRAM bandwidth, and it helped fuel the proliferation of personal computers during the late 1980s and 1990s. The proposed use of a separate distributed cache on a DRAM chip [5] led to the current ESDRAM (enhanced synchronous DRAM) technology. Numerous means were considered for merging SRAM and DRAM on the same chip. Many were unsuccessful because of a lack of proper integration of key functions. For example, in [6], an experiment was described in which the DRAM and SRAM were interconnected with a somewhat standard, narrow bus that was not much different from a bus that would be used if the SRAM were on a separate chip. For the on-chip hierarchy to function effectively, significant integration of the data transfer processes was necessary, such as that used with the "Supercache," discussed below. This was a significant omission in many of the early proposals; i.e., associated functions were not integrated to make use of on-chip advantages.

Despite the rather clear need for more highly integrated functions on a chip, logic technology continued to be driven by speed considerations and memory technology by cost considerations. As the levels of integration continued to increase, the inclusion of fast logic and DRAM on the same chip became feasible. However, there were different views about how to optimize system performance and cost. Within IBM, there was considerable interest in using sophisticated packaging techniques to achieve this—permitting separate logic and DRAM chips to be more tightly integrated on a carrier with large numbers of interconnections [7, 8]. However, this would have led to additional delays and cost.

Since the early introduction of semiconductor memories, there had been numerous proposals to use two-, three-, or four- device memory cells [9, 10], which could be fabricated using standard logic technology, resulting in densities far higher than those of normal six-device SRAM, but not as high as DRAM densities. This would provide improved SRAM density without the need for a new technology. However, the performance still lagged behind what could be obtained with the eDRAM approach.

We had been working extensively on caches, DRAM/SRAM arrays, and memory hierarchies [11–18] and thus had hands-on experience with the wide range of diverse problems encountered in optimizing total system performance. Early in the evolution of the growing speed gap, it became evident to us that 1) the continued advances in integrated circuits would soon make possible fully integrated systems (large memory and processor) on a single chip; 2) the speed gap between DRAM and logic

[2] For more information, search the Internet for *page mode*.

circuits would impose serious performance limitations on integrated systems on a chip; and 3) DRAM could be nearly as fast as SRAM in the same technology. In such integrated systems, large memory capacity on-chip would result in better performance, but if the historical trend were to continue, such systems-on-a-chip would not be able to make use of the larger density provided by DRAM—a serious limitation. Thus, early in the 1990s, we became convinced that the optimum performance of future systems would require the use of a logic-based DRAM technology. We reached this conclusion through various types of analysis which began in 1990 and continued for more than seven years in numerous forms and complexities. A substantial amount of the initial work was "engineering analysis" of memory hierarchies based on insights and educated assumptions. While this type of analysis proved to be correct, it was nevertheless, by itself, insufficient to dispel the considerable doubt about our conclusion that the use of logic-based eDRAM would be essential for continued system evolution. Several options were considered, but when designed "on paper" they clearly indicated the difficulties with a DRAM-based eDRAM. Two of these are discussed later: the so-called "Supercache" option and a full system-on-a-chip option implementing a matrix multiply unit.

Initially, the consensus was that the use of *DRAM-based* eDRAM with its cheaper processing but slower device speed was the correct strategy for system evolution. The basic issue was whether logic should be implemented via the DRAM process (DRAM-based eDRAM)[3] or DRAM implemented via the logic process (logic-based eDRAM, seemingly more difficult and expensive).[4] Despite opposition, our view was that closing the speed gap via the use of *logic-based* DRAM would be the most viable means to ensure system evolution. The timely commencement of an implementation effort in the IBM Technology Group[5] and continued system studies ultimately led to the inclusion of logic-based eDRAM in the IBM ASICs (application-specific integrated circuits) product; see for example [19–26]. This history unfolded as described next.

Early environment of the IBM metal-oxide field-effect transistor (MOSFET) technology

The MOSFET technology used by IBM in the late 1960s and early 1970s for manufacturing semiconductor logic and memory devices and circuits was based on the use of n-channel MOSFETS (also designated as "n-channel devices," "n-MOSFETs," "n-FETs," "n-type devices," etc.), metal gates, and associated depletion loads (ratio logic levels without p-channel devices) [27, 28]. Thus, logic and memory functions could be integrated on a single chip. At the time, main memory used SRAM arrays, as illustrated by the 2-Kb Riesling chip [29], which contained six-device cells. Subsequently, use was made of the DRAM-based technology, based on the use of cells containing one MOSFET device and one capacitor [30]. That technology has become the dominant memory technology because of its improved density and cost.

After the introduction of the DRAM technology but before the availability of the CMOS (complementary metal oxide semiconductor)[6] technology, there was continued use of n-channel devices for both logic and memory. However, a small but noticeable difference between logic and DRAM device speed appeared. The reason for this was that the DRAM technology required the use of devices having a *higher* threshold voltage V_t to ensure low leakage, resulting in an increase in the time interval between refresh cycles.[7] The logic technology involved the use of a similar but modified process to obtain devices having a slightly *lower* threshold voltage that facilitated higher overdrive, $V_{dd} - V_t$ (where V_{dd} is the power-supply voltage), and thus greater circuit speed.[8] The different V_t values were achieved by slightly different channel doping. However, since the magnitude of V_{dd} was 5 V, the difference in overdrive for devices with "high" V_t (~1 V) vs. "low" V_t (~0.7 V) was not very large (5 to 10%); thus, the speed increase was small. The technology required the use of only one type of gate with only minor differences needed to speed up the logic circuitry (lower V_t) or reduce the DRAM leakage (higher V_t).

CMOS technology

During the late 1970s to early 1980s, CMOS technology became viable in IBM manufacturing. The technology required the use of p-channel load devices. Two methods available for introducing such devices were

1. A low-cost approach based on the use of heavily n-doped polysilicon (n^+-poly) gates and a buried p-doped channel to produce its p-channel load devices

[3] DRAM-based eDRAM involves retaining the current, single-work-function technology (described later) and relatively slow DRAM technology and re-mapping logic books and libraries to its ground rules. Thus, embedded processors with DRAM can be designed in same technology, but they are slower than those designed in logic-based (dual-work-function) technology (described later).

[4] Logic-based eDRAM involves adding the DRAM deep trench and other processes to the logic process, requiring extra masks and processing steps. However, the logic libraries do not have to be remapped to new ground rules, which is a significant extra advantage.

[5] Undertaken by the IBM Academy under the leadership of Russell Lange (IBM Technology Group).

[6] The "M" in MOS and CMOS originally referred to the use of a metal gate. Although the metal gate has mostly been replaced by an n-doped or p-doped polysilicon gate, the "M" has been retained.

[7] A DRAM cell stores data as charge on a capacitor. This charge continually leaks away through the field-effect access device and must be refreshed about every 16 milliseconds. Refresh time can interfere with normal accesses and must be controlled. The use of higher-V_t devices reduces this leakage.

[8] Overdrive is a measure of the amount of force exerted on the carriers in the channel of a field-effect device in order to accelerate them during switching; thus, it affects switching speed.

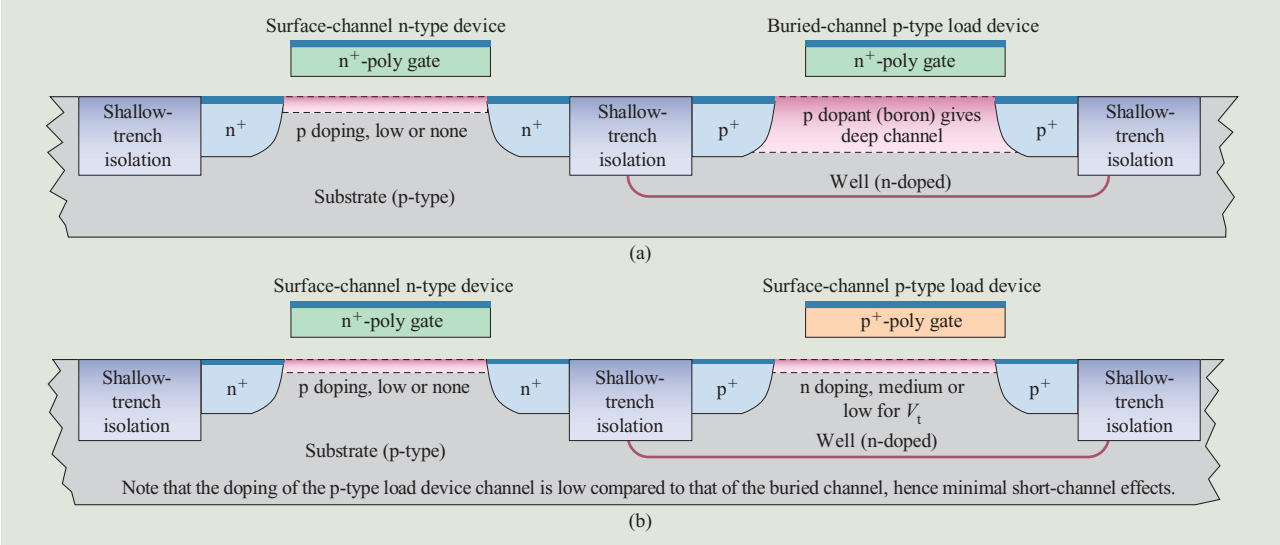

Figure 1

Illustrative cross sections of CMOS configurations fabricated via (a) single- and (b) dual-work-function technologies. In (a), n^+-poly gates are used for the n- and p-channel devices; the p-channel device has dopant added to its surface channel region (to achieve lower V_t), resulting in the formation of a relatively deep channel and associated short-channel degradation effects (single-work-function technology). In (b), an n^+-poly gate is used for the n-channel (normal surface channel) device and a p^+-poly gate for the p-channel (normal surface channel) device (dual-work-function technology).

("single-work-function technology"), as illustrated by the configuration in **Figure 1(a)**.

2. A more costly approach based on the use of n^+-poly gates for its n-type devices and p^+-poly gates for its p-channel load devices ("dual-work-function technology"), as illustrated in **Figure 1(b)**.

At that time, the state of the art employed rather large devices (~1-μm channel length). The single-work-function (buried-channel) approach was suitable for producing both logic and DRAM structures. The first CMOS technology in IBM manufacturing, CMOS 2, was based on the use of that approach (with a channel length of ~1 μm) for both logic and DRAM devices, and was designed to function at V_{dd} = 5 V. The use of this relatively large operating power-supply voltage resulted in only a small difference between logic and DRAM device speeds. Different thresholds were obtained for the devices through the use of different doping levels. For information on CMOS 2 and subsequent generations of IBM CMOS device technologies, see for example [31] and cited references.

While this single-work-function technology was suitable for the relatively large devices being used at the time, it was very difficult to scale the buried-channel devices to smaller channel lengths and concurrently obtain the desired scaled speed improvement; i.e., the speed did not scale well. This difficulty is due to short-channel effects resulting from the deep diffusion incurred for a buried channel. Short-channel effects arise because the implanted boron layer has a finite and relatively large thickness, as shown for the device to the right in Figure 1(a), instead of being an infinitesimally thin sheet, as for the (dual-work-function technology) surface-channel devices of Figure 1(b).

To scale to smaller devices and obtain a lower V_t for logic operation, use was made of a dual-work-function process which functioned at V_{dd} = 2.5 V and V_t ~0.5 V (CMOS 5X). The use of a p-poly surface channel for the p-type devices resulted in the lower value of V_t. However, because the process was more expensive, the use of a slower, buried-channel p-type device (single-work-function technology) was continued for DRAM. Thus, DRAM devices continued to be slower than their logic counterparts. (A buried-channel n-type device has not been used because its p^+-poly gate has a relatively high resistance and its processing is difficult.) Other device and technology requirements which tended to make DRAM technology slower than logic are the following:

1. The need for a thicker gate oxide to allow word-line boost for writing and reading at higher overdrive in order to obtain a higher signal-to-noise ratio (the thicker oxide is needed to prevent electrical breakdown of the oxide at the higher overdrive).

2. The need for a relatively large threshold voltage in order to reduce leakage current (and thus obtain a longer data retention time and longer time between refresh cycles).

Thus, the small speed gap between the logic and DRAM technologies became significantly wider than previously. For more on buried-channel devices and short-channel effects, see [31], part (a), p. 188 and [32], p. 291.

New IBM DRAM strategy for on-chip systems

In the early to mid-1990s, the industry continued to use the above strategy, i.e. the use of separate technologies for DRAM (single-work-function technology) and logic (dual-work-function technology). This caused the speed gap between logic and DRAM to continue to widen, as illustrated in **Figure 2**.

In the early 1990s our engineering analyses of simple systems showed that as the levels of integration increased, significant amounts of memory could and should be placed on a chip to improve performance. The analyses indicated that for memory hierarchies on a chip, if the DRAM speed could be sufficiently improved, the significantly higher density of DRAM compared to SRAM (four to eight times more DRAM in the same area) would offset any speed difference. This would effectively allow DRAM to replace SRAM because of the improved miss ratio of the larger capacity. In addition, further analysis convinced us that the difference in access speed between SRAM and DRAM was limited mainly by the technologies used for implementation, not by fundamental elements. (See Appendix C for a discussion of this issue.) Thus, if DRAM were to be built in logic technology, its access time could be much closer to that of SRAM (rather than being significantly different, as was prevalent at that time).

Our subsequent analysis of memory hierarchies in the early 1990s indicated the various speed ranges required for on-chip DRAM to compete with faster but less dense SRAM (presented later) over a wide range of system parameters. It was found that significant improvements in system performance could result from replacing SRAM with improved DRAM and that the required DRAM parameters were well within the realm of possibility. Thus, it appeared that it should be possible to decrease the speed gap between logic and DRAM technology, as indicated by the dotted curve in Figure 2.[9]

Our first memory-hierarchy-on-a-chip design

While we were aware of the potential and the need for fast DRAM in the early 1990s, there was little capability for

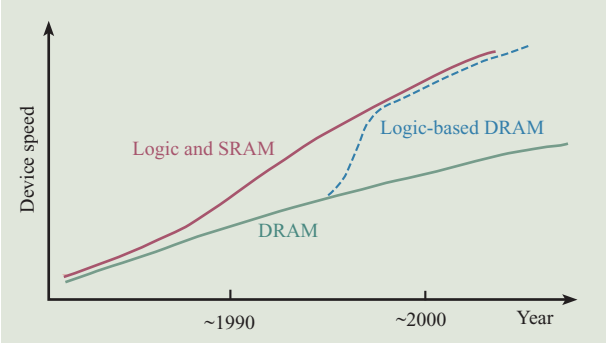

Figure 2

Increasing speed gap between IBM logic and DRAM technologies resulting from cost vs. performance design objectives.

any actual implementation. Thus, our first design was a memory hierarchy consisting of an SRAM in front of a standard DRAM, all on the same DRAM chip. We chose as the base the highest-density DRAM available at the time, the IBM 16-Mb chip [33]. We removed half of the DRAM and replaced it with SRAM and many custom interface circuits and buffers (store-back buffers, reload buffers, etc.). The base design consisted of two 8-Mb islands with built-in error correction and other intervening circuits, thus making it easy to divide. Our modified design contained several buffers and circuits similar to that used in the first IBM RS/6000* cache [18, 34] to support high-speed simultaneous transfers between the L1, L2, and L3 levels. By integrating the two-level cache hierarchy on a single chip, high bandwidth, reduced latency, and better performance were achieved. Substantial circuit analysis, projected speeds, and power calculations were carried out. A schematic diagram of the chip functions and their placement is shown in **Figure 3**. This chip, designated as the Supercache[10] chip, became part of an advanced RS/6000 multiprocessor product plan of record in the early 1990s. It offered substantial performance improvement but lacked sufficient product design support at that time. Thus the Supercache design was abandoned, but the same basic idea is currently used for the on-chip L3/L2 cache of the IBM Blue Gene* high-performance processor [35], implemented in IBM logic-based eDRAM technology.

Matrix multiply unit (MXU)—Full system on a chip

A subsequent study was undertaken to evaluate the performance potential of a special vector processor [36] designated as the matrix multiply unit, or MXU. The concept required a relatively large memory and a high

[9]Note that this does not necessarily imply that large, off-chip main memory systems should be designed with the same fast logic-based eDRAM chips. This is a separate issue that is not part of this work.

[10]U.S. Patents 5,388,072 (1995) and 5,890,215 (1999).

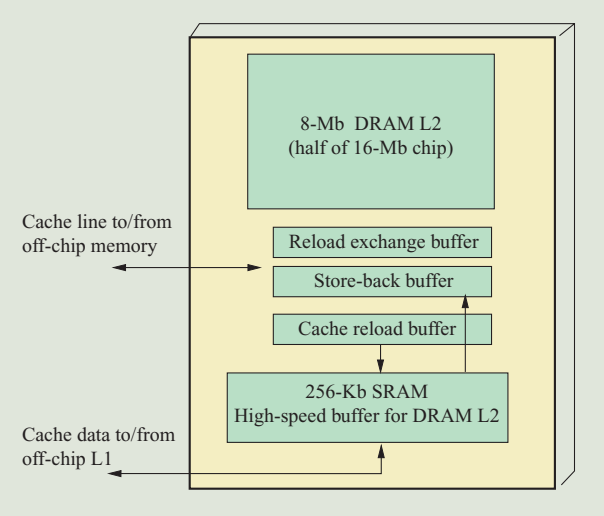

Figure 3

Placement of functions included on Supercache chip.

bandwidth for the parallel processing of vectors. We immediately recognized that the use of embedded DRAM with an SRAM buffer similar to Supercache could provide these, and configured a possible implementation consisting of a processor, MXU, with associated memory on one chip, as shown in **Figure 4**. The MXU includes everything except the memory; in a non-integrated design, the MXU as well as the memory would be located on separate chips. This study proved to be pivotal; up to that time, our arguments for a logic-based eDRAM had been based primarily on engineering analysis of very general memory hierarchies, and we did not have an actual, convincing example. The Supercache system mentioned above did not have an on-chip processor/memory hierarchy. The MXU provided our first, definitive such case. We intended the MXU study to be realistic, and thus carried out the initial system integration using the only appropriate technology available at the time, the DRAM technology (single-work-function, slow, lower-cost chips). The results showed that such an integrated MXU system on a chip fabricated in a DRAM-based eDRAM technology could *not* win in the marketplace. The fundamental problem was that the CPU and matrix-processing unit logic would be fabricated in a technology that was too slow compared with a conventional system using fast logic and SRAM chips with a standard memory hierarchy. *The use of larger-capacity DRAM to replace SRAM could not compensate for the degradation resulting from the slower, DRAM-based CPU/L1 cache/vector logic.* It thus became clear that in order to achieve better performance than that possible with a conventional technology, enhanced device speed would be needed. The analysis which led us to this conclusion proceeded as described in limited detail in Appendix D.

The MXU study and subsequent related work showed the need for logic-based eDRAM in order to achieve higher *system* performance, not necessarily higher DRAM performance. Thus, if DRAM-based eDRAM is used as the base, system performance is always limited by the CPU/logic and will be inferior even for the best achievable DRAM speed; i.e., the larger DRAM cannot compensate for the lower processor performance. However, the use of a logic-based eDRAM facilitates the fabrication of fast CPU/logic for a competitive position and the replacement of on-chip SRAM with eDRAM, which can now also be relatively fast—thus leading to an improved overall system.

Engineering analysis of generic systems

Throughout our more than seven years of study, we made use of different techniques and methods of analysis to make our case for logic-based eDRAM. The essence of the case is the use of miss rates and spreadsheets to calculate relevant parameters. An overview of the approach is presented next, followed by a discussion of its use in comparing the performance of memory hierarchies when SRAM for the L2 and L3 levels of cache is replaced by DRAM of various speeds.

Analysis using miss rates and spreadsheets

The ideal, raw processing power of a processor is measured in cycles per instruction for an infinite cache [37]; i.e., its first-level cache functions as if there were no cache misses and thus no reload penalties. This is equivalent to having attached to the system only one ideal memory of very large size and using a cycle time equal to that of the processor. Since such an ideal memory is not feasible, designers attempt to approximate it by using a memory hierarchy comprising caches of small size and high speed at the first level (the processor level), and designing for a gradually increasing capacity with a decreasing speed, up to main memory. Typically there may be one to four such cache levels between the processor and main memory. Since the level closest to the processor (L1) is smaller in capacity, it cannot retain all of the information which may be required by the processor at any instant. Address translation is required to determine whether any requested information is actually present. Thus, some accesses "miss," i.e., are not present, and require a "reload," which is a transfer of the full block that contains the required information, from the downstream memory to L1. An L1 miss requests the data from L2. If L2 has a miss, it requests the data from L3, etc. until the information, which may be in main memory as the last level, is found. An access to any level downstream from L1 results in the transfer of a block or

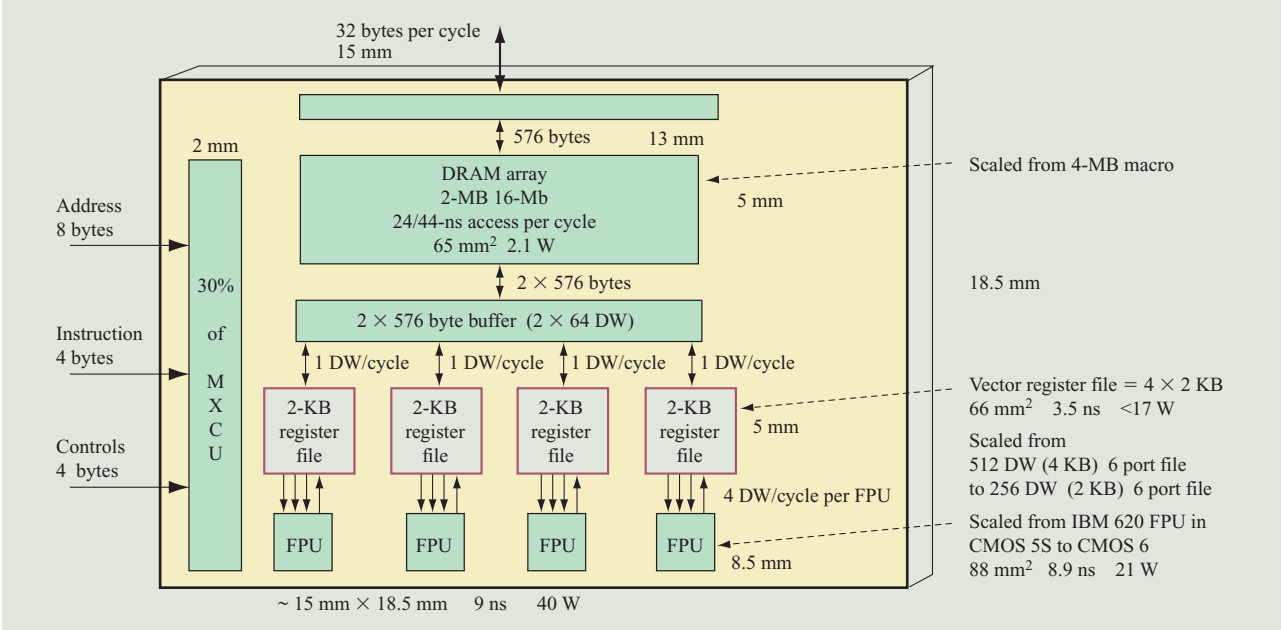

Figure 4

Layout and busing of matrix multiply unit accelerator chip having four FPUs and 2-MB on-chip DRAM L2.

line, typically 32 to 128 bytes, of information to L1 since this likely contains the next piece of information required by the processor. At the beginning of this transfer, the initially requested information which caused the miss (typically a doubleword, 8 bytes) is the first data accessed and is simultaneously "loaded through" to the processor so that it can resume processing. Thus, the average, equivalent memory access time for any system with a memory hierarchy is considerably longer than one system cycle because of stalls and delays caused by cache misses and reloading. This additional delay, measured in cycles per instruction executed, is typically known as the finite cache penalty (FCP). These two parameters are added together to obtain the cycles per instruction (CPI) for the system:

$$CPI[system] = CPI[infinite\ cache] + FCP. \quad (1)$$

The CPI [infinite cache] is independent of the memory hierarchy and is assumed to be given. The attached memory hierarchy affects only the FCP term. The memory hierarchy analysis of only a simple uniprocessor is presented below. Analyses of more complex multiprocessor systems as well as a fuller treatment of this subject are presented in [37].

A uniprocessor system consisting of a single processor having an n-level memory hierarchy is illustrated in **Figure 5**. A processor and first-level cache (L1) are typically designed as a single self-contained unit and thus are not available for optimization as part of the hierarchy analysis. The L1 capacity is typically limited by the processor cycle time and silicon space available, and thus is not a degree of freedom. Rather, the processor and L1 together are the source which generates misses at a given miss rate of mr_1 misses per instruction executed. If there were *no* misses (i.e., infinite L1 cache) or if the miss latency time were 0, the resulting CPI value would be *CPI[infinite cache]*. Assuming that this parameter is given, our task is to determine the additional number of cycles per instruction required for the given memory hierarchy. Miss rates are typically used as the measure of the miss characteristics of each level of the hierarchy:

$$Miss\ rate = number\ of\ misses\ per\ instruction\ executed\ by\ the\ processor. \quad (2)$$

It is possible to express the FCP in terms of a miss ratio for a level, defined as the number of misses per access to that level. These parameters are quite different and are not interchangeable. The miss ratios can be expressed in terms of miss rates for all cache levels except the L1 cache, which requires an additional parameter, as discussed in Appendix B of [38]. The FCP expression is derived below in terms of both miss rates and miss ratios.

In the figure, the miss latency time at each level is assumed to be a fixed constant T_n, where $n = 2, 3, 4$, etc. for each downstream level, as indicated. This time is the total effective time to reload a hit at any level, including

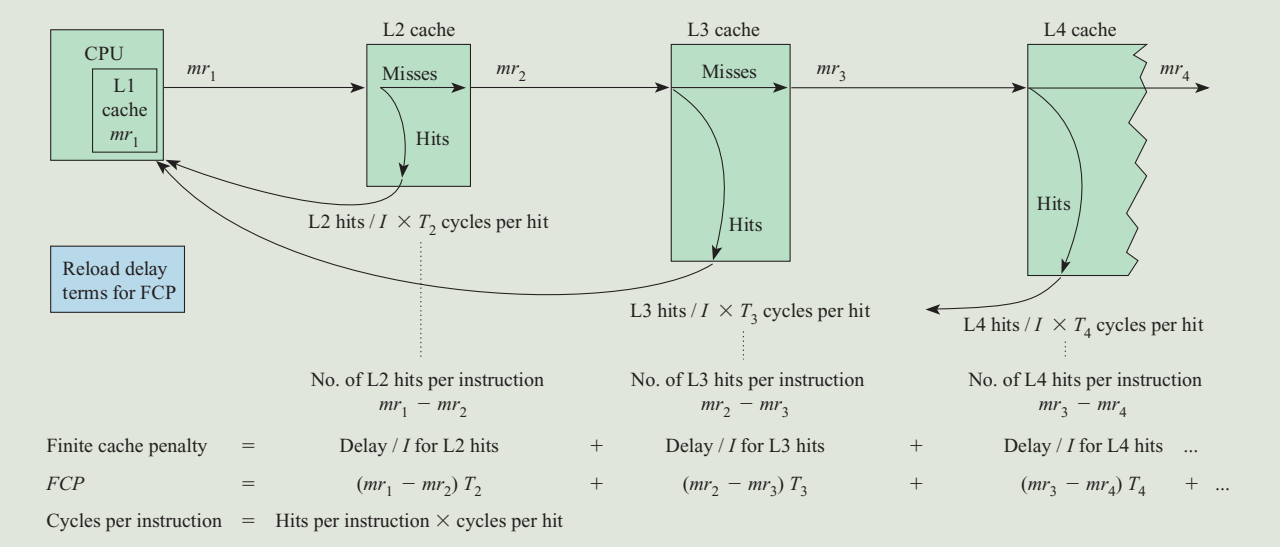

Figure 5

Uniprocessor with simple n-level memory hierarchy. (I: no. of instructions; T_n = miss latency time, where n = 2, 3, 4 etc. for each downstream level.)

all electrical and logical delays; to a good approximation, it can be taken as the time to first access at that level. In a more complex system, particularly a multiprocessor configuration, there are additional queuing delays due to multiple requests to any cache from multiple sources. Such delays are a nonlinear function of the requests at each level and greatly complicate modeling and analysis. A good design minimizes these delays so that they are second-order effects. Thus, this simple analysis is a good approximation.

The L1 miss rate mr_1 serves as input to the memory hierarchy for the FCP calculation. It is necessary to determine only the average number of cycles per instruction required to reload these mr_1 misses per instruction. All L1 misses trigger an interrogation of L2, producing some L2 hits, with the remainder being L2 misses. The L2 misses propagate to L3, producing some L3 hits, with the remainder being L3 misses. This hit–miss behavior continues to the level which produces only hits, which is main memory in this model. A specific hit at any level is reloaded with appropriate delay, and terminates any further miss-request interrogations downstream.[11] Portions of the total average reload delay come from each level. For a general case, this includes hits in L2 with delay T_2, hits in L3 with delay T_3, hits in L4 with delay T_4, etc., as shown in the figure. These amounts are easily determined as follows: The hit rate hr_n at any level n is given by

$$\begin{aligned} hr_n &= \textit{inputs per instruction} - \textit{outputs per instruction} \\ &= \textit{misses per instruction}\ [\textit{previous level}] \\ &\quad - \textit{misses per instruction}\ [\textit{current level}] \\ &= mr_{n-1} - mr_n\,. \end{aligned} \tag{3}$$

The portion of the total FCP contributed by each level n of the hierarchy is the hit rate multiplied by the average effective reload time per hit of that level. The FCP is expressed in units of processor cycles per instruction; hence, all delays T_n are expressed in units of number of processor cycles per hit rather than absolute time. Thus, the contribution to the FCP for any level of the hierarchy is given by

$$FCP_n = hr_n \cdot T_n = (mr_{n-1} - mr_n)T_n\,. \tag{4}$$

The total finite cache penalty of an n-level hierarchy is the sum of all individual terms, or

$$FCP = \sum_{n=2}^{z}(mr_{n-1} - mr_n)T_n\,. \tag{5}$$

Assuming no misses in main memory, the FCP for a hierarchy with four levels below main memory would be

$$\begin{aligned} FCP = {} & (mr_1 - mr_2)T_2 + (mr_2 - mr_3)T_3 \\ & + (mr_3 - mr_4)T_4 + mr_4 T_{\text{main}}, \end{aligned} \tag{6}$$

where mr_1, mr_2, $\cdots$ are the miss rates at levels 1, 2, etc., in misses per instruction executed, and T_2, T_3, $\cdots$ are the average effective reload time in units of the number of

[11] Misses flow downstream from L1 to L2 to L3, etc., while reloads flow upstream.

processor cycles per miss (i.e., the actual delay divided by the cycle time of the processor). This equation can be written in terms of miss ratios as follows. As shown in Appendix B of [38], miss rates can be expressed as

$$mr_1 = A_I MR_1,\ mr_2 = mr_1 MR_2,\ mr_3 = mr_2 MR_3,\ \text{and}$$
$$mr_n = mr_{n-1} MR_n, \tag{7}$$

where A_I is the average processor memory accesses per instruction executed, and MR_1, MR_2, etc. are the miss ratios of L1, L2, etc. in misses per accesses to that level. Substituting these into Equation (6) gives

$$\begin{aligned} FCP &= A_I MR_1(1 - MR_2)T_2 + A_I MR_1 MR_2(1 - MR_3)T_3 \\ &\quad + A_I MR_1 MR_2 MR_3(1 - MR_4)T_4 \cdots \\ &= A_I[MR_1(1 - MR_2)T_2 + MR_1 MR_2(1 - MR_3)T_3 \\ &\quad + MR_1 MR_2 MR_3(1 - MR_4)T_4 \cdots]. \end{aligned} \tag{8}$$

The value of A_I is generally unknown, since the number of memory accesses required by program instructions such as load or store-multiple, etc. are unknown until the program is compiled or executed—and can be data-dependent. However, a value of unity is valid for many cases. Assuming that value, we obtain

$$mr_1 = MR_1,\ mr_2 = MR_1 MR_2,\ mr_3 = MR_1 MR_2 MR_3,$$
$$mr_n = MR_1 MR_2 MR_3 \cdots MR_n \tag{9}$$

and

$$\begin{aligned} FCP &= MR_1(1 - MR_2)T_2 + MR_1 MR_2(1 - MR_3)T_3 \\ &\quad + MR_1 MR_2 MR_3(1 - MR_4)T_4 \cdots, \end{aligned} \tag{10}$$

where T_2, T_3, etc. are the effective reload access times for L2, L3, etc. in units of processor cycles.

At the processor/L1 cache level, the L1 cache speed is included indirectly by way of the CPI[infinite cache] parameter, which is assumed to be given; hence, it is not a variable in this analysis, as can be seen from Equations (6) and (10). The only L1 variable in these equations is the L1 miss ratio, which is varied below. Equations (6) and (10) are the fundamental equations used for most of our subsequent analyses of memory hierarchies of a simple uniprocessor. (See for example [38].)

Spreadsheet analyses of memory hierarchies

The crucial performance parameter in any memory hierarchy is the sum of the products of hit rate times the reload time at each cache level. The reload time consists primarily of chip access time plus all additional packaging and transfer delays.[12] The miss rate or the miss ratio at

[12]Trailing-edge effects may add additional delays if the transfer of a block or line is done piecemeal, requiring many cycles. This results from the fact that the first reload cycle restarts the CPU and subsequent cache accesses may be to a doubleword that is in the same line but not yet reloaded—see [37] for more details.

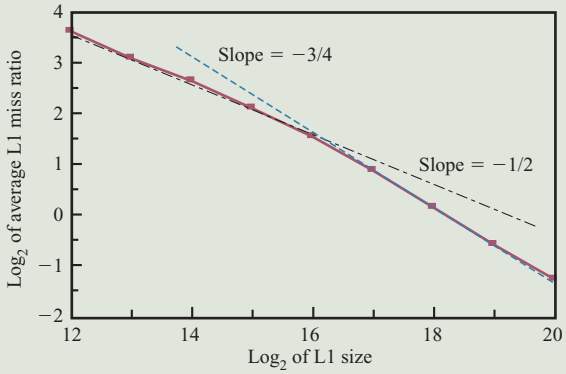

Figure 6

Calculated values of $\log_2$ of average L1 miss ratio vs. $\log_2$ of L1 size for four-way set-associative, 128-byte cache blocks (lines), showing 1/2 and 3/4 power dependence of miss ratio on cache size, for a commercial transaction processing trace workload.

any cache level improves (decreases) with increasing cache size (capacity); detailed analyses have shown that the miss ratio varies approximately as the square root of the cache size, as can be seen, for example, in **Figure 6**. Plotted in that figure is the miss ratio as a function of cache size for a transaction-processing-type commercial trace on a log scale, indicating that the miss ratio varies as the square root of the cache size between 4 and 64 KB (the "square root" rule). Above 64 KB, this and other analyses suggest that for larger caches, the slope may be −3/4. However, this analysis was not extended sufficiently to reach a more definitive conclusion about the slope variation. If the miss ratio vs. size were actually to vary as 3/4 power, the eDRAM approach could have an even greater advantage. This square root rule is assumed to apply in subsequent analyses unless stated otherwise.

At a given technology level, assuming that the square root rule applies, analyses indicate that DRAM can typically provide anywhere from four to eight times more bits than SRAM in the same area, or two to nearly three times lower miss ratios for the same amount of silicon area consumed. However, DRAM has traditionally been slower than SRAM, and in the past had not been considered as a candidate to replace SRAM. With System Scale Integration (SSI), DRAM array and packaging/transfer delays for cache reloads are greatly diminished so that for many practical cases, the hit ratio × reload time and the resulting FCP can be substantially better for a DRAM than for an SRAM. This is seen to be true in the following analysis.

When the concept of logic-based eDRAM was first proposed, the technical community was of the opinion that better performance could be obtained via either of

the following: a DRAM-based eDRAM in which the processor, SRAM, and DRAM were fabricated in the normal DRAM-based technology, or the standard approach involving fabrication of logic and SRAM in the high-speed logic technology and DRAM in the standard DRAM-based technology. The following analysis demonstrates that this is not the case over a wide set of parameters representing typical cases.

Clearly, the performance improvement is highly dependent on the DRAM access time achievable. Also, the reload times T_2 or T_3 include delays other than the array access times, such as logic delays, translation delays, and fixed wire delays. These aspects are not easy to quantify without some specific system design. Nevertheless, we were confident during the initial phases of this work that we could reduce the DRAM access time to within a few cycles (roughly three to five) of an SRAM. Thus, whatever values of T_2 or T_3 are assumed for an SRAM, the best DRAM effective reload access would be that SRAM time plus a few cycles. This is typically the assumption used in the following analysis. However, it is currently expected that DRAM access times will approach and possibly exceed those of SRAMs.[13]

Processor plus SRAM L2 vs. processor plus DRAM L2

In the first comparison, the basic system is assumed to consist of a processor with its given L1 cache (and given CPI[infinite]) plus an L2 cache connected to a main memory. We compute the overall performance of such a system as given by the total system CPI vs. L1 miss ratio MR_1 for three cases: a system consisting of a logic-technology-based processor plus an SRAM L2 (traditional, standard design); a DRAM-technology-based processor plus a DRAM L2; and a logic-based processor plus a logic-based DRAM L2. This is similar to the comparison done for the one specific design point in the MXU study detailed in Appendix D, but now done more generally and for a range of miss ratios. We assume a speed scaling factor of 1.7× between the DRAM-based and logic-based technologies, which is similar to that used in the MXU study.[14]

We start with a logic-based processor with its L1 cache and arbitrarily assume a value of one cycle per instruction for its CPI[infinite].[15] The same processor plus L1 cache in a DRAM-based technology would then have a CPI[infinite] value of 1.7 cycles per instruction (i.e., one cycle per instruction times the speed scaling factor of 1.7). To the former, logic-based processor/L1, we attach an on-chip SRAM L2 having a T_2 value of 13 cycles and a miss ratio of 30%. The main memory (L3 in this case, with $MR_3 = 0$) is assumed to have a reload access time T_3 of 134 cycles. The total system CPI is equal to $1 + FCP$, where the FCP is evaluated from Equation (10) with MR_1 varying from 0 to 10%. The results are given by the blue curve in **Figure 7**. At $MR_1 = 0$, obviously the FCP is 0, so the intercept on the y-axis must be at 1, as shown. Clearly, this represents the traditional standard system using high-speed logic technology for processor/L1 and SRAM L2.

The same system implemented using a DRAM-based technology for processor/L1 and DRAM would yield a DRAM L2 which would be eight times larger in capacity and 1.7 times slower, or would have a reload access time of $T_2 = 1.7 \times 18 = 31$ cycles (i.e., the logic-based DRAM discussed below is chosen at 18 cycles). For this case, the processor/L1 would similarly have a CPI[infinite] value of 1.7 compared to 1 for the logic-based case. A plot of the calculated system CPI vs. MR_1 for this case using the same main memory is shown in Figure 7 as the green curve. The value of the system CPI at $MR_1 = 0$ starts out at a larger value of 1.7 (re: use of slower technology). However, as MR_1 increases, the larger L2 DRAM gives a much more slowly increasing FCP than the previous logic-based processor + SRAM L2 case. As can be seen, the two curves cross at a value of MR_1 of about 10%, indicating that the DRAM-based technology should have an inferior performance at L1 miss ratios less than 10% and superior performance above 10%. Since most applications run with an average L1 miss ratio less than 10%, the DRAM-based system is not as attractive as a standard processor with an SRAM L2. This was the situation and general view of DRAM performance potential for memory hierarchy applications before the introduction of the concept of logic-based DRAM. We next introduce such a DRAM into the same system.

The calculated performance for a logic-based processor plus a DRAM L2 with a capacity of eight times that of the SRAM but with a T_2 of 18 cycles is plotted in Figure 7 as the red curve; i.e., the SRAM of the first case above is replaced by a DRAM eight times larger and five cycles slower. At $MR_1 = 0$, the system CPI must start at a value of 1. However, the larger but slower DRAM L2 gives a smaller FCP at all values of MR_1. In addition, the logic-based DRAM L2 appears to be increasingly superior to the SRAM L2 as MR_1 increases; i.e., the slope of the DRAM curve is smaller than that of the SRAM. This is a result of the lower miss ratio. Similarly, the logic-based DRAM L2 appears to be superior to the DRAM-based

[13] For the same memory capacity, because of its significantly smaller cell, a DRAM occupies considerably less area than an SRAM. Additionally, because its interconnections are shorter, its loading is less, etc., its speed can potentially become faster than that of an SRAM. In addition, as we scale to smaller dimensions, in the sub-0.1-μm range, SRAM cells become unstable because of "read-disturbs." As a result, the SRAM cell size must be increased to larger than minimum size. Since this does not occur for DRAM cells, DRAM scales more easily, adding to its area advantage. (SRAM scaling problems raise many interesting issues pertaining to power leakage, speed, etc.)

[14] A scaling factor between normal DRAM and logic speeds would increase as the technologies evolve, making the case for logic-based eDRAM even more compelling.

[15] CPI[infinite cache] is very dependent on the specific processor and L1 cache design (see Appendix B).

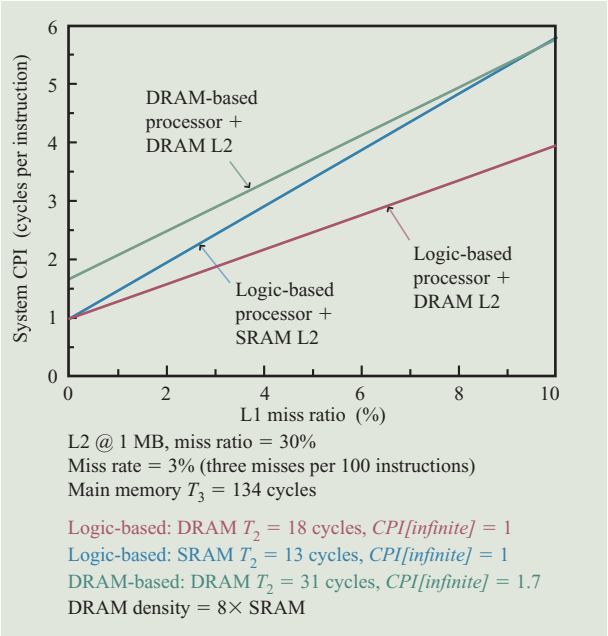

Figure 7

Calculated values of system CPI vs. L1 miss ratio for logic-based systems-on-chip (red and blue curves), contrasted with a DRAM-based system-on-chip (green curve). Statements under the plots pertain to assumed parameters; those that pertain to only one of the plots are color-coded.

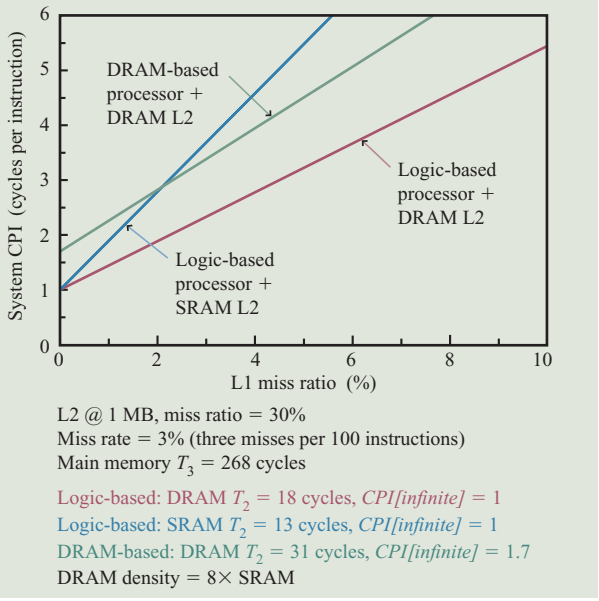

Figure 8

Calculated values of system CPI vs. L1 miss ratio for logic-based systems-on-chip (red and blue curves), contrasted with a DRAM-based system-on-chip (green curve). Statements under the plots pertain to assumed parameters; those that pertain to only one of the plots are color-coded. Assumed parameters identical to those of previous figure except that T_3 = 268 cycles (twice as slow).

L2 at $MR_1 = 0$ because the processor is faster, and for $MR_1 > 0$ because the logic-based DRAM access time is shorter ($T_2 = 18$ vs. 31 cycles).

Thus, we see that not only should the logic-based DRAM system give the best performance, but it should also be less sensitive to variations in MR_1, an important consideration. While this has been shown for only one set of parameters, the same trends remain valid over a very wide, representative range of parameters. For example, suppose we wish to attach a much larger and thus slower main memory to the above system, keeping all other parameters constant. The calculated performance for such a case with a main memory of $T_3 = 268$ (twice as slow as previously) is shown in **Figure 8**. Again we see that the logic-based eDRAM system should be far superior and also less sensitive to variations. However, the calculated FCP is beginning to become so large that the use of an L3 before main memory should be considered, as discussed later.

Replacing SRAM with DRAM: Performance comparisons

The finite cache penalty vs. cache size (capacity) is another dramatic, graphical method of contrasting DRAM performance vs. an SRAM for an L2, L3, or any level cache. Such analyses can quickly show the performance advantage of the larger DRAM capacity for various assumed DRAM and SRAM access times.

For the first comparison, as previously, a memory hierarchy is chosen which consists of L1, L2, and main memory. The calculated FCP is plotted vs. memory capacity for various assumed T_2 access delays as in Equation (10), where the L2 miss ratio MR_2 is assumed to vary with capacity according to the square root rule. DRAM and SRAM are distinguished only by their different T_2 values and are plotted on the same figure with all other parameters held constant.

We start by assuming an extremely slow L2 DRAM (or a DRAM off-chip, or both) with a T_2 of 46 cycles, and a relatively fast SRAM L2 with $T_2 = 13$ cycles. The L1 cache is assumed to have a fixed miss ratio of 5%, as indicated in the figure. The SRAM and DRAM are assumed to have the same miss ratio: 10% at the 1-MB size (e.g., one base unit);[16] and it is assumed that the miss ratio scales as the square root of memory size. The results

[16] Actual size is fundamentally irrelevant—the base unit is whatever size gives the assumed miss ratio. We use a base of 1 MB for convenience.

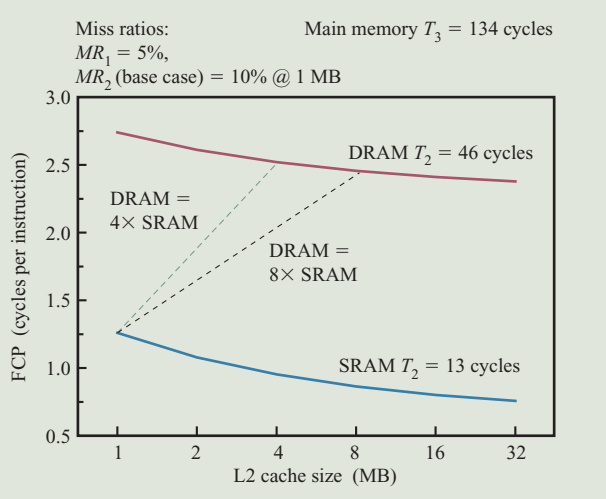

Figure 9

Calculated values of FCP (finite cache penalty) vs. L2 cache size for SRAM and slow (standard) DRAM, starting from base system with L2 MR_2 = 10% at 1 MB for L2 vs. L2 size, with MR_2 varying as the square root of the L2 size ratio, and with the L1 miss ratio = miss rate = 5%.

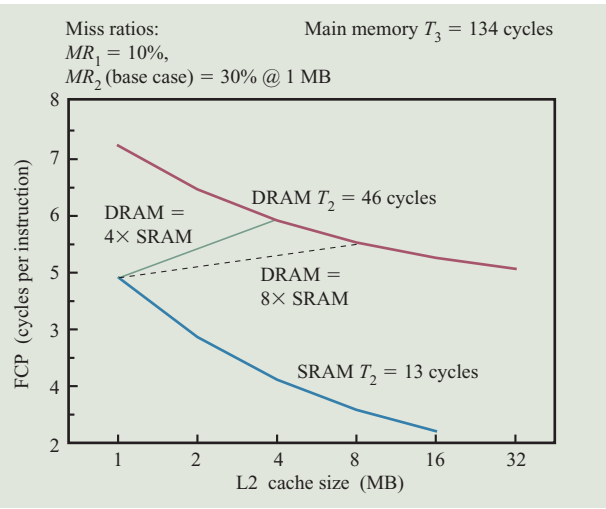

Figure 10

Calculated values of FCP vs. L2 cache size for SRAM and slow (standard) DRAM, starting from base system with L2 MR_2 = 30% at 1 MB and increasing L2 size, with MR_2 varying as the square root of the L2 size ratio, and with the L1 miss ratio = miss rate = 10%.

obtained are shown in **Figure 9** for a relatively fast main memory having T_3 = 134 cycles.

Clearly, at any given memory size such as 1 MB, the finite cache penalty of 2.75 cycles per instruction for

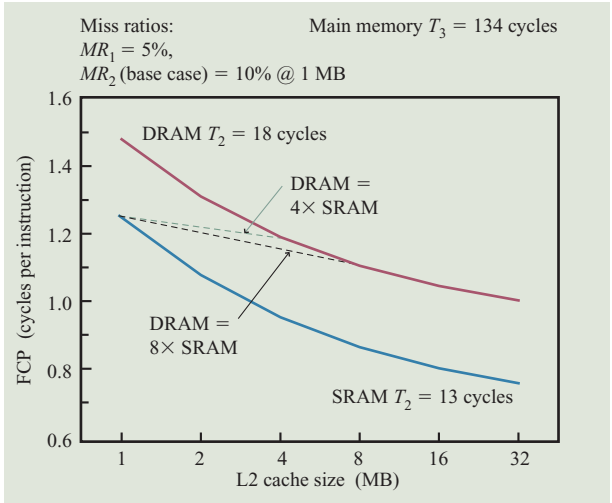

Figure 11

Calculated values of FCP vs. L2 cache size for SRAM and fast (logic-based) DRAM, starting from base system with L2 MR_2 = 10% (miss rate = 0.5%) at 1 MB and increasing L2 size, with MR_2 varying as the square root of the L2 size ratio, and with the L1 miss ratio = miss rate = 5%.

DRAM is worse than the 1.25 cycles per instruction for SRAM. If we assume that the DRAM size can be increased by a factor of 4 or 8 over the SRAM (in the same area), the FCP calculated for the DRAM drops to 2.5 or 2.4, respectively, as shown by the dashed lines connecting the two curves. *The DRAM would be inferior in all of these cases because it would have been implemented with a very slow, standard DRAM technology and/or would have been packaged off-chip*. If we were to start with a larger SRAM, say 2 MB in size, the predicted improvement in FCP for a 4× or 8× DRAM is even less, as expected. If the L1 miss ratio and base L2 miss ratio were to be increased to 10% and 30%, respectively (reasonable values in some cases), the FCP gap between the SRAM and DRAM should decrease, as indicated in **Figure 10**. *However, if the DRAM could become somewhat faster by placing it on-chip to reduce bus delays and using logic devices to improve the access time of the DRAM chip, the situation would change appreciably, as shown next.*

A comparison of calculated FCPs for the same SRAM with different DRAM L2s, fabricated using on-chip, logic-based eDRAM is shown in **Figures 11** and **12**. The L1 and L2 base miss ratios are assumed to be the same as previously (5% and 10%, respectively, in Figure 11, and 10% and 30%, respectively, in Figure 12). However, *the DRAM is assumed to have an access time only 5 cycles greater than the SRAM, namely 18 cycles. The results indicate that the improvement in FCP by using a larger*

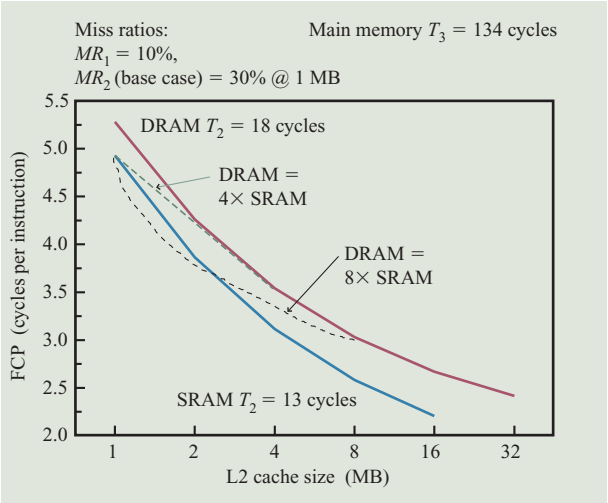

Figure 12

Calculated values of FCP vs. L2 cache size for SRAM and fast (logic-based) DRAM, starting from base system with L2 MR_2 = 30% (miss rate = 3%) at 1 MB and increasing L2 size, with MR_2 varying as the square root of the L2 size ratio, with the L1 miss ratio = miss rate = 10%, and with the main memory T_3 = 134 cycles.

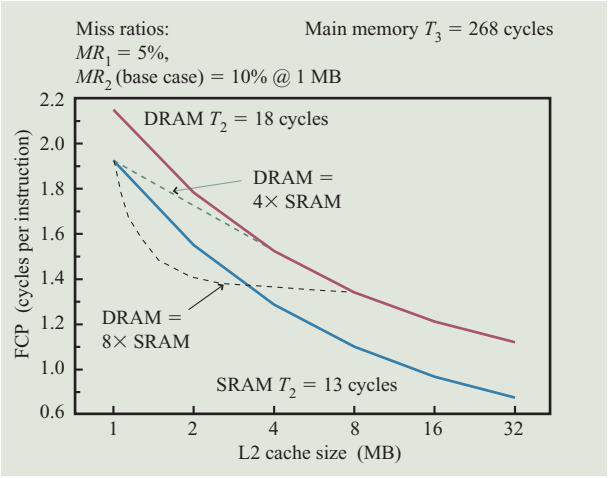

Figure 13

Calculated values of FCP vs. L2 cache size for SRAM and fast (logic-based) DRAM, starting from base system with L2 MR_2 = 10% (miss rate = 0.5%) at 1 MB and increasing L2 size, with MR_2 varying as the square root of the L2 size ratio, and with the L1 miss ratio = miss rate = 5%; identical to Figure 11 except that main memory T_3 = 268 cycles (twice as slow).

DRAM in the same area as an SRAM should be considerable.

For example, as indicated in Figure 11, a 15% improvement in FCP should be attainable by replacing a 1-MB SRAM with an 8-MB DRAM. A more dramatic improvement should be possible, as indicated in Figure 12; i.e., it should be possible to reduce the SRAM FCP of 4.75 cycles per instruction at 1 MB to 3.5 or 3 cycles per instruction by replacing the SRAM (in the same area) with 4 or 8 MB of DRAM.

Further improvement should be possible if the off-chip main memory were slower (reducing cost). For instance, if T_3 for the main memory of Figures 11 and 12 were assumed to be 268 cycles, hence two times slower than the 134 processor cycles assumed in the above cases, the trends shown in **Figures 13** and **14** should apply—indicating not only that the DRAM should have a substantial performance advantage, but that its performance should be less sensitive to variations in the L1 miss ratio (as discussed previously in the *Processor plus SRAM L2 vs. processor plus DRAM L2* section).

Similar curves for using an eDRAM L3 in place of an SRAM L3 are illustrated in **Figures 15** and **16** for two cases of L1, L2, and L3 miss ratios: 5, 15, and 15% and 10, 20, and 30%, respectively. These two cases represent a wide range of possible applications. The following were assumed: The reload access time, T_2, of the L2 was assumed to be 13 cycles (same as previously), the T_3 of the

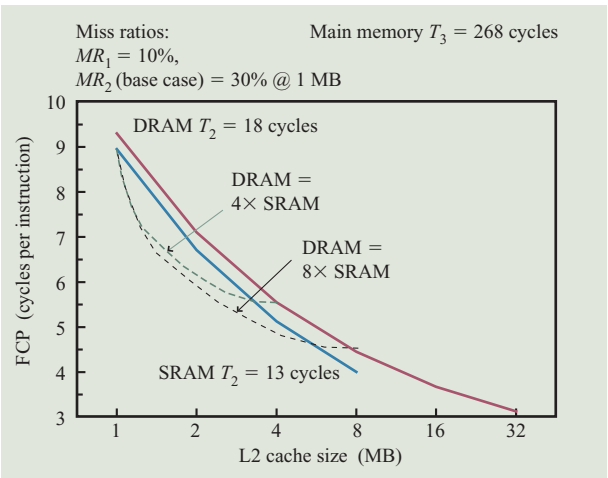

Figure 14

Calculated values of FCP vs. L2 cache size for SRAM and fast (logic-based) DRAM, starting from base system with L2 MR_2 = 30% (miss rate = 3%) at 1 MB and increasing L2 size, with MR_2 varying as the square root of L2 size ratio, and with L1 miss ratio = miss rate = 10%; identical to Figure 12 except that main memory T_3 = 268 cycles twice as slow as previously.

SRAM L3 was assumed to be 40 cycles, and the T_3 of the DRAM L3 was assumed to be 45 cycles. The results obtained were very similar to those obtained previously for L2—viz., replacement of the SRAM L3 with a higher-

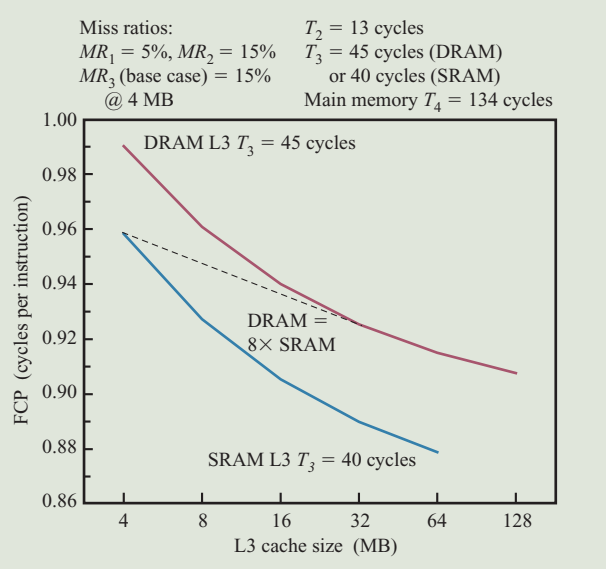

Figure 15

Calculated three-level hierarchy FCP vs. L3 cache size for SRAM and fast DRAM, with L1 and L2 miss ratios of 5% and 15%, respectively, and with L2 delay T_2 of 13 cycles. The L3 miss ratio is 15% at 4 MB and varies as the square root of capacity, with delays (as shown).

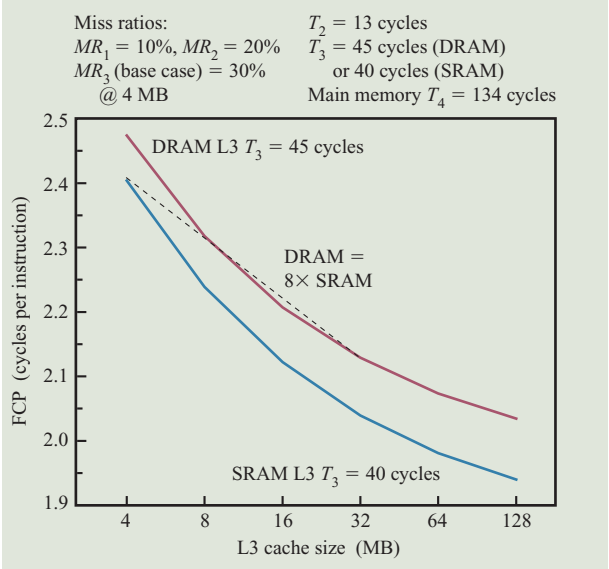

Figure 16

Calculated three-level hierarchy FCP vs. L3 cache size for SRAM and fast DRAM, with L1 and L2 miss ratios of 10% and 20%, respectively, and with L2 delay T_2 of 13 cycles. The L3 miss ratio is 30% at 4 MB and varies as the square root of capacity, with delays (as shown).

density, fast logic-based eDRAM should result in a substantial improvement in performance. An even greater advantage should be possible if a slower off-chip main memory were to be used in the hierarchy.

Access time for SRAM L2 vs. DRAM L2

In the past, DRAMs had traditionally been designed for large capacity at low cost; speed was relatively unimportant. A key objective at the beginning of this work was to estimate the speed improvement that would make DRAM attractive for replacing SRAMs in L2 and L3. This could be done by simply plotting the FCP vs. L2 access times for an SRAM and DRAM of assumed capacity for various cases over a range of design points. Assume, for example, that a 1-MB SRAM L2 of given area and miss ratio were to be replaced by one of three DRAMs of the same area having capacities of 4, 6, and 8 MB. For each possibility, the FCP of a two-level hierarchy consisting of L1 and L2 caches between main memory and processor vs. access time could be calculated via Equation (10).

For the first case, we assume the following: $mr_1 = MR_1 = 5\%$; a base miss ratio MR_2 of 10% for a 1-MB-capacity SRAM; and L2 miss ratio scaling as the square root of the size ratio for the larger DRAM, i.e., MR_2 at 4 MB $= 1/2 \times 10\%$, etc. The calculated FCP vs. DRAM L2 access time is shown in **Figure 17** for a moderately fast main memory having $T_3 = 134$ processor cycles (same as previously). For the 1-MB SRAM, at a T_2 of 10 cycles, the FCP is 1.125 cycles per instruction, as indicated in the figure by an **X**. If a horizontal line is drawn from that point (i.e., at 1.125 cycles per instruction, shown dashed, the intersections with the DRAM curves give points of equal performance. These intersections occur at DRAM access times of 17 cycles for the 4× size, and at 19 cycles for the 8× size. Hence, the DRAM should begin to outperform the SRAM when its access time falls below these values. Similarly, we can choose any SRAM access time, draw a horizontal line at the FCP value corresponding to that access time, and determine the DRAM access times for the same FCP. Obviously, for all access times below these latter values, the DRAM should outperform the SRAM. The smaller the DRAM L2 access times, the larger the percentage of improvement of DRAM over SRAM. In this case, for any given FCP, the SRAM curve maintains a fixed incremental access time difference of about 8 cycles to any of the DRAM curves. Thus, for better performance, the DRAM should have an access time less than 8 cycles larger than the SRAM at any design point. If the DRAM access time is 3 to 5 cycles slower than the SRAM, a significant percentage improvement in FCP is obtained.

Larger improvements in FCP are indicated when a given application exhibits a larger miss ratio. For

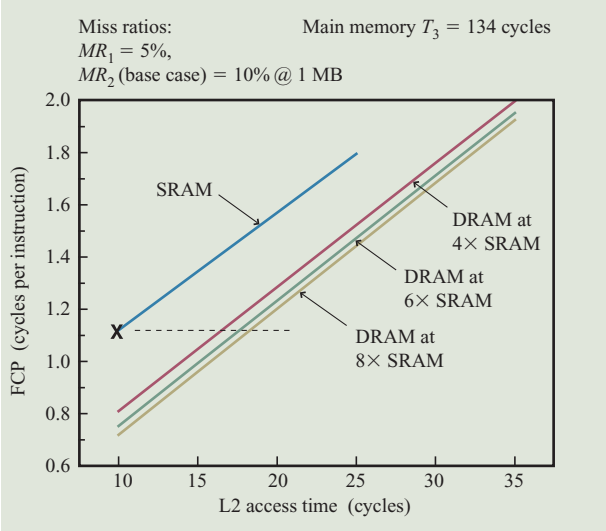

Figure 17

Calculated FCP vs. L2 access time for SRAM and DRAM of various densities, for L1 miss ratio = miss rate = 5% and L2 miss ratio = 10% (miss rate = 0.5%) at 1 MB.

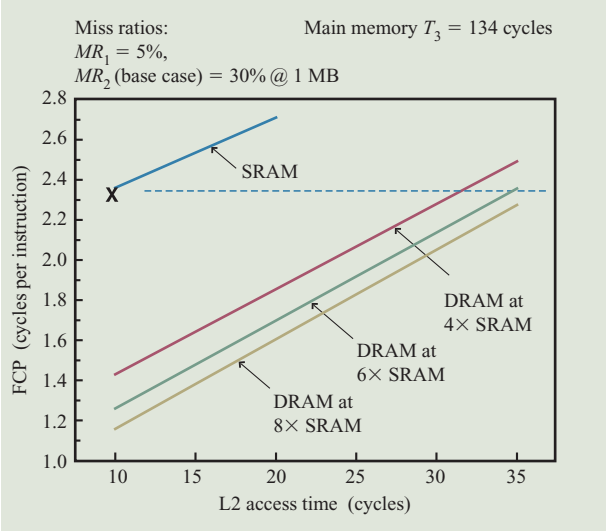

Figure 18

Calculated values of FCP vs. L2 access time for SRAM and DRAM of various densities, for L1 miss ratio = miss rate = 5% and L2 miss ratio = 30% (miss rate = 1.5%) at 1 MB.

example, if the L2 miss ratio increases to 30%, the curves of Figure 17 change to those of **Figure 18**. The horizontal time spread between the SRAM and DRAM curves becomes approximately 25 cycles, indicating that the DRAM L2 access time would have to be less than 25 cycles larger than that of the SRAM in order to be competitive. If that access time were only 3 to 5 cycles larger than the SRAM, a very large performance improvement should be attainable. For example, for a 13-cycle SRAM, the calculated FCP is 2.45 cycles per instruction, whereas for an 18-cycle DRAM at 6× the SRAM density, the calculated FCP is 1.6 cycles per instruction—an improvement by a factor of 2.45 ÷ 1.6 = 1.53. If the DRAM and SRAM each have an access time of 10 cycles, the DRAM could provide about a factor of 2 improvement in FCP, as can be seen from the figure. Thus, the performance improvement could be quite substantial, depending on the miss ratio of the application.

The expected miss ratios of actual systems would be some weighted average of these cases shown, with the weights dependent on the exact workloads. However, whatever values are used for miss ratios or for the base-case SRAM size and performance, the results have the same general trend, namely, with a relatively modest increase in DRAM performance, it should begin to outperform SRAM. If the access times of a DRAM could be within a few cycles of those of an SRAM, analysis shows that a very significant performance advantage should be possible over a wide range of parameters.

Of course, a cost would be incurred to achieve the increased performance of the DRAM. How this cost would affect the cost/performance was a concern. Initially this was difficult to determine, since a logic-based DRAM technology had never been considered. However, as our study unfolded, this proved to be less difficult than anticipated, indicating only about a 20% increase in process complexity [20]. Other important performance enhancements are also inherent in logic-based eDRAM. For example, wide, fast buses can be provided on-chip between L2 and L1 and between L3 and L2 to minimize the cache reload penalty and minimize other factors that can significantly affect performance. As discussed previously and shown in Appendix B, the total reload time can have a very significant effect on system performance and is often not well understood nor appreciated.

Potential performance improvement was the main impetus for advocating a change in strategic direction toward logic-based eDRAM. Also, we felt that the lower power dissipation of DRAM vs. SRAM, especially with scaling to smaller dimensions, could become a significant advantage. Although our work showed considerable potential for logic-based eDRAM, making it a reality was a separate and significant task carried out by others. Its introduction into the IBM high-performance CMOS 7 technology was a first in the industry, making possible

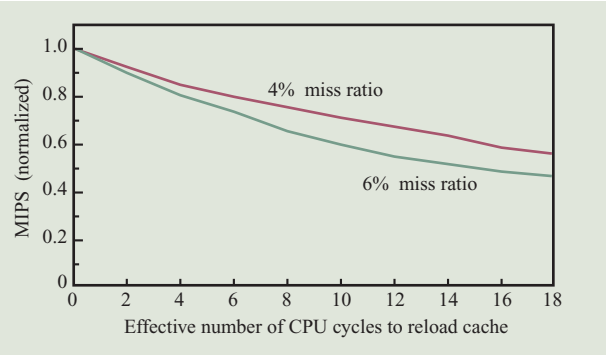

Figure 19

Calculated MIPS vs. effective number of cycles to reload L1 cache at two values of miss ratio; based on simulations of an early RISC system with *CPI[infinite]* = 1.22 cycles per instruction (see [15]).

not only a significant improvement in DRAM performance, but also, by providing a larger memory closer to the processor, a significant improvement in system-on-a-chip performance.

Concluding remarks

Technology evolution typically starts down a particular path on the basis of cost and performance tradeoffs that are valid at the time of decision. The evolutionary trends can become entrenched and are thus believed to be correct, even when the environment changes substantially. DRAM became entrenched as a separate semiconductor technology early in the history of integrated circuits and remained dominant for several decades. When the performance factors driving the evolution changed, it was difficult for the industry to recognize this. Over a period of several years, we were able to demonstrate the need for a strategic change more convincingly, resulting gradually in a shift toward logic-based eDRAM.

In the evolution of a new engineering direction, initial phases are greatly enhanced by simple analysis based on a few fundamentals with good assumptions about the unknowns. Such early analysis is also important for making meaningful projections when details of actual implementations cannot possibly be available. Some preliminary system design and performance tradeoffs then become essential in order to resolve critical problems. Needless to say, implementation often requires an advocate with the ability to obtain resource commitments.

Our initial work on the path to eDRAM began with simple engineering analysis of memory hierarchies using miss ratios, similar to that described in this paper. The first simple "implementation" evaluation was via the Supercache memory hierarchy on a chip, and then via the subsequent matrix multiply unit, which contained a full processor plus an on-chip memory hierarchy. This work convinced us that the correct path for system evolution necessitated the use of a logic-based eDRAM. Many subsequent analyses were carried out in order to refine and further verify our ideas and to convince others of their relevance.

The implementation of such a change in technology and strategy required the skills of a large number of individuals. We were fortunate in that several key people in the IBM Technology Group became advocates and were able to secure funding for actual implementation (see the Acknowledgments).

Appendix A: Memory speed scaling

As computer technology continues to scale to ever smaller and faster switching devices with higher packing density, the data processing speeds of the central processor continue to improve, but the main memory access speeds do not scale proportionally. In a certain sense, this seems counterintuitive, but it occurs for the following reasons.

To a first approximation, within a given instruction set architecture, the critical logic path of a processor which sets the fundamental processor cycle time has remained relatively fixed over a long period of time. In other words, the number of logic stage delays in the critical path which sets the cycle time has remained relatively fixed. While this may change because of possible evolutionary changes in pipeline structure, changes in design have not substantially changed the number of logic stages in the critical path. The cycle time is thus, to a first approximation, set by a nearly fixed number of logic delays, including the associated wiring time of flight. As technology scales to faster devices, both the logic and wiring delays decrease, so the total critical path delay decreases, resulting in improved processor cycle times.

Faster processors require more instructions and data per unit of time to keep busy.[17] This necessitates an ever-increasing main memory capacity as processor performance increases. Unfortunately, such increases in capacity require an ever-increasing number of "logic delays" for access. This increase results from the need for more decoding to select, typically, a doubleword (8 bytes) from an increasing number of total words. Additionally, the need for translation of the virtual address increases the total access delay. Also, although sizes of devices, wires, and associated logic gates and memory cells have decreased, a larger capacity requires more of each in the access path. Thus, the access delay does not scale with technology, but may actually increase, depending on the capacity chosen. If the memory capacity

[17]Processors typically require at least one MB of memory per MIPS (million instructions per second) of processing power; see [14], Chapter 1, Figure 1.5-4.

remained fixed, the access delay would improve as technology scaled to smaller dimensions.

Another problem which aggravates the memory delay is that as the speed gap between processor and memory widens, more levels of cache are required between the processor and main memory to bridge this gap. However, although there is a net improvement, the cache levels introduce some additional levels of logic delay in the path to main memory. Thus, even though much is gained, a small loss also occurs.

Appendix B: Cache reload time vs. system performance

The importance of cache reload time to system performance is illustrated in **Figures 19** and **20**. Figure 19 represents the simulated performance of an actual early system design, designated internally as ROMP-E, which was an early attempt to include a cache on the original IBM PC RT RISC processor [15, p. 277]. The curves show the system MIPS (million instructions per second) vs. the effective cache reload time for two different cache miss ratios. At a reasonable design point of about 8 cycles used for reload, each additional cycle of reload degrades the system performance as much as 6%. This is for a processor with a maximum processing speed (infinite cache) of 1.2 cycles per instruction (proportional to 1/MIPS). The reload time, which is part of the finite cache penalty, degrades this and can be significant, as shown. If the effective reload time is 16 cycles (not unusual at this time), the processor MIPS can decrease to 50% of its maximum value. The actual curve of MIPS vs. reload time depends on a number of parameters as well as the full memory hierarchy, and can degrade more dramatically.

As processors improve in speed, the number of cycles per instruction decreases (current systems function at under one cycle per instruction). Figure 20 illustrates typical MIPS degradation vs. effective reload time for several cases with ideal cycles per instruction and one typical, average cache miss ratio. As expected, with decreasing cycles per instruction (increasing MIPS), each cycle of delay for cache reload has a larger impact on the overall MIPS. Thus, it is essential to continually reduce the effective reload time as the number of cycles per instruction decreases. This can be done quite effectively with eDRAM.

Appendix C: DRAM vs. SRAM speed

If an SRAM and a DRAM array are both designed in the same technology with the same organization and objectives, the major difference in access time is in the sensing delay, because of the smaller sense signal available in DRAM. For similar arrays and loads, all other delay components of any access (address-in,

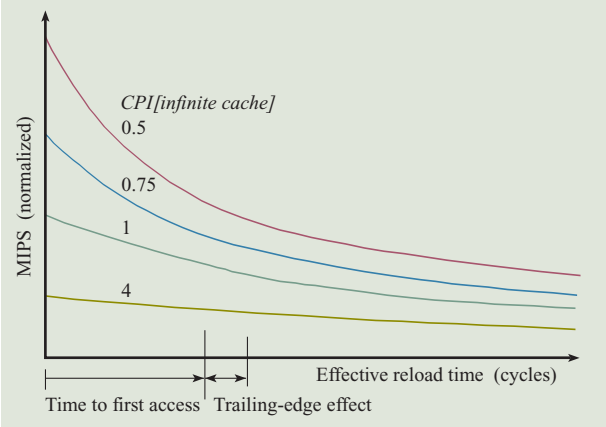

Figure 20

Typical dependence of MIPS on effective reload time, calculated for several values of *CPI[infinite cache]*, showing increasing sensitivity to reload time as *CPI[infinite cache]* decreases.

decode, wordline drive, signal routing, and I/O bus) are fundamentally not very different. In actual designs, there are typically other small or even large differences which can affect many of the delay components. Nevertheless, the sense signal has been a continuing, major component.[18] Examples of estimated access times for a DRAM macro in several different base technologies, all at the CMOS 7 (0.225-μm) lithographic level, are shown in **Figure 21**. The top example corresponds to the use of a standard DRAM design. The on-chip DRAM macro case corresponds to the same normal DRAM design without address multiplexing and without off-chip drivers and receivers, since they are not required; the logic-based eDRAM macro case corresponds to the same design as the latter in logic-based eDRAM technology. The access time of the logic-based eDRAM macro is further reduced by the addition of an improved sense amplifier [11, 13], as indicated in the improved SA case.

Similar SRAM designs were not available for comparison; however, if they had been, the delay components would be roughly similar, except for the sensing delay, which is typically larger in DRAM because of a smaller sense signal. However, with the continued scaling of technology to smaller dimensions and currents, the SRAM sense signals are decreasing because of the poor scaling of device thresholds and other factors.

Currently, SRAM and DRAM sense signals are of nearly the same amplitude, thereby allowing DRAM access time to approach that of SRAM for arrays of the same size.

[18] Actual numbers are very dependent on technology, and various tradeoffs are made in array and circuit design—tradeoffs related to the number of bits per wordline and per bitline, the number of inputs and outputs, device driver sizes, whether self-timing is used, etc.

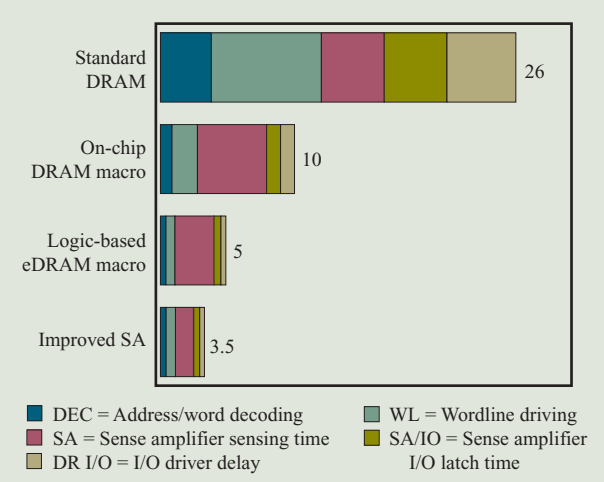

Figure 21

Improvement of DRAM access time with integration and merged logic at CMOS 7 (0.225-μm lithographic level). Logic-based devices are 2.8× faster than DRAM-based devices in this comparison.

Appendix D: Matrix multiply unit (MXU)

The MXU system-on-a-chip study made use of the technologies and component macros available at the time. Figure 4 shows the logical structure and macro sizes of a single-chip MXU containing four floating-point units (FPUs), an 8-KB fast vector register file, a matrix multiply control unit (MXCU), and 2 MB of DRAM, all scaled from a CMOS 5 (see for example [33] and [39]) to a CMOS 6[19] DRAM-based eDRAM. These individual units were all scaled from other known designs in CMOS 5. A direct scaling into the CMOS 6 technology gave a projected cycle time of 9 ns. An estimated improvement in CMOS 6 DRAM-based technology of up to 15% would give a projected improved system cycle time of about 7.5 ns. Unfortunately, neither the 9-ns nor the 7.5-ns MXU systems were competitive for the following reason.

The competition would be a standard system in which the processor units were fabricated on standard logic chips with a standard memory hierarchy (SRAM on-chip, DRAM off-chip. Two standard available (noncustom MXU) floating-point units which could have been used at that time to provide system cost/performance advantage over the custom, DRAM-based MXU system on a chip were

1. An IBM PowerPC* floating-point unit (FPU) fabricated via the CMOS 5X technology and operating at 5 to 6 ns.
2. An IBM PowerPC FPU fabricated via the CMOS 6X technology, operating at about 3 ns.

Various scaling studies were carried out in order to derive reasonable standard alternatives. A high-performance PowerPC processor with a 64K instruction cache (SRAM) and 64K data cache (SRAM) would allow two FPUs to be included on-chip. Such a system might have a clock cycle time as low as 5.4 ns. The matrix multiply performance depended on both the clock time and the number of on-chip FPUs. With only two FPUs and running at about 5 ns, the performance would be about 0.4 (2 ÷ 5) floating-point operations per nanosecond.

The custom MUX-DSRAM accelerator of Figure 4 had four FPUs and operated at about 7.5 ns, so its performance would have been about 0.53 (4 ÷ 7.5) floating-point operations per nanosecond. This is only marginally better than the 0.4 floating-point operations per nanosecond of a standard system designed in existing technology, and is therefore unattractive. The problem with this custom design is the MXU speed, which is limited by the DRAM-based technology. If the system-on-a-chip were to be fabricated in a logic-based technology, it should be possible to reduce the MXU-DSRAM cycle time to about 3 ns, thus improving its system performance to about 1.33 (4 ÷ 3) floating-point operations per nanosecond, which would be three to four times better than that of the standard system. Although the system is clearly interesting, its implementation would require the use of a logic-based eDRAM. It thus became clear that for any applications requiring processors and logic with embedded DRAM, the use of a logic-based memory technology would be essential in order to be competitive.

Glossary

CMOS (complementary metal oxide semiconductor) configuration: A field-effect transistor configuration which provides p-type devices, which are normally "on" for load devices, and n-type devices (normally "off") for pulldown devices. A p-type device and an n-type device in series with gates connected in parallel produce an inverter. The p-type device has a very large ratio of "off" to "on" resistance, unlike the earlier depletion load devices, for which the ratio was low. The latter produced an "off" current which was a significant fraction of the "on" current, giving an inferior logic circuit and higher power dissipation.

Cache block or line: The unit of transfer between levels of cache or from memory to cache. This unit is larger than the normal requested data size, which is typically a doubleword, or 8 bytes. The block (line) size is typically 32 to 128 bytes at the lower levels of cache closer to the

[19] CMOS 5 and CMOS 5X (0.5-μm lithographic level) could be used to fabricate a 16-Mb/chip DRAM; CMOS 6 and CMOS 6X (0.35-μm lithographic level) could be used to fabricate a 64-Mb/chip DRAM.

processor, and sometimes larger—256 bytes, at higher levels closer to main memory.

CPI[infinite]: Cycles per instruction, assuming an infinite cache (no misses); the average, minimum number of cycles required by a processor per instruction executed if there were no cache misses, i.e., if the cache were infinite.

FCP (finite cache penalty): The average reload delay penalty in cycles per instruction required for restarting a processor after a cache miss. This penalty is added to the CPI[infinite] to obtain the average execution speed of the processor.

Doubleword: Word consisting of 8 bytes (64 bits or 72 bits with parity), which is two normal words of 4 bytes each, in IBM architecture.

Miss rate: For any given level of a cache, the average number of access misses (loads or stores) per instruction executed by the processor.

Miss ratio: For any given level of a cache, the average number of access misses (loads or stores) per total number of access requests to that level of cache. Miss ratios can be expressed in terms of miss rates for all levels of cache except L1 (the level providing loads and stores to the processor) for reasons described in Appendix B of [38].

Page mode: A mode which is typically used for fast data transfer of a reload request from a DRAM chip. A full cache line or block is loaded into an on-chip page buffer on one DRAM row access cycle; the smaller units of data (typically 8 to 16 bytes) are clocked out of the buffer at the bus rate, which is much faster than the DRAM access or cycle time. This can significantly reduce the FCP.

SSI (System Scale Integration): The integration level at which essentially a full processor with memory is located on one semiconductor chip.

VideoRAM: A special type of DRAM used as a pixel buffer for all of the graphics from an SVGA (Super Very high Graphics Adapter). An industry standard from the mid-1980s to the late 1990s, containing two I/O ports which allowed simultaneous update of the stored pixel image and copying of the stored image to the screen. It provided a significant increase in the graphics bandwidth required in high-performance graphics.

Acknowledgments

Initially there was serious doubt that replacing on-chip SRAM with logic-based eDRAM would lead to enhanced performance. This was the initial and primary concern addressed by the authors in their study. The second concern pertained to its cost—and its analysis required the additional study and support provided by the Technology Driver Task Force of the IBM Technology Group. The authors would like to acknowledge this important contribution.

Around 1995, Russell C. Lange, then the Director of Semiconductor Strategy for the IBM Microelectronics Division, who headed both that task force and the IBM Academy at that time, became highly interested in the possibility of merging logic with DRAM. Without his subsequent involvement and unique skills, the logic-based eDRAM strategy would have been a tenuous endeavor.

Significant players in making the technology a reality were Tze Chiang Chen, who directed the original development effort, and Scott Crowder, Sang Dhong, Bijan Divari, Bud El Kareh, Tom Heller, Mike Ignatowski, Subramanian Iyer, Tak Ning, Scott Stiffler, and numerous others.

The matrix multiply unit study was initiated by Maurizio Arienzo (formerly an IBM employee) and Jurij Paraszczak, who foresaw the implications of Supercache for future systems-on-a-chip and thus provided the opportunity for us to complete a major stepping stone.

We respectfully acknowledge all of these valuable contributions.

References and associated notes

1. H. Pilo, A. Darren, J. Barth, S. Burns, P. Corson, J. Covino, R. Houghton, and S. Lamphier, "A 5.6 Random Cycle 144 Mb DRAM with 1.4 Gb/s/pin and DDR3-SRAM Interface," *IEEE J. Solid-State Circuits* **38**, 1974 (2003).
2. R. Matick, "Hybrid Memory with On Chip Associative Page Addressing, Page Replacement and Control," U.S. Patent 4,084,230, 1978; DRAM (or SRAM cache) with translation directory and on-chip control.
3. R. Matick, D. T. Ling, S. Gupta, and F. Dill, "All Points Addressable Raster Display Memory," *IBM J. Res. & Dev.* **28**, 379 (1984).
4. F. Dill, D. Ling, and R. Matick, "Random Access Memory Having a Second Input/Output Port," U.S. Patent 4,541,075, 1985; DRAM with added logic and buffers to speed up graphics (video RAM).
5. R. Matick and D. T. Ling, "Distributed, On-Chip Cache," U.S. Patent 4,577,293, 1986; included a small, fast SRAM on a memory chip as a cache, and was a precursor to and first version of current ESDRAM (enhanced synchronous DRAM) chips.
6. M. Asakura, Y. Matsuda, H. Hidaka, Y. Tanaka, and K. Fujishima, "An Experimental 1 Mbit Cache DRAM with ECC," *IEEE J. Solid-State Circuits* **25**, 5 (1990).
7. B. Pogge, "The Next Chip Challenge: Effective Methods for Viable Mixed Technology SoCs," *Proceedings of the 39th Conference on Design Automation* (ASM, IEEE), New Orleans, June 10–14, 2002, p. 84.
8. F. Bozso and P. Emma, "Clock Skew Minimization and Method for Integrated Circuits," U.S. Patent 6,040,203, 2000;

substrates connected together, face to face, using flip-chip technology.
9. E. R. Hnatek, *A User's Handbook of Semiconductor Memories*, John Wiley & Sons, Inc., New York, 1977, p. 360; multiple device memory cells.
10. S. E. Schuster, L. M. Terman, and R. L. Franch, "A 4-Device CMOS Static RAM Cell Using Sub-Threshold Conduction," presented at the IEEE Symposium on VLSI Technology, Systems, and Applications, Taipei, Taiwan, 1987; *Research Report RC-13171*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, 1987.
11. T. Chappell, B. Chappell, S. Schuster, J. Allen, S. Klepner, R. Joshi, and R. Franch, "A 2 ns Cycle, 3.8 ns Access 512 kb CMOS ECL SRAM with a Fully Pipelined Architecture," *IEEE J. Solid-State Circuits* **26**, 1577 (1991).
12. B. El-Kareh, G. B. Broner, and S. E. Schuster, "The Evolution of DRAM Cell Technology," *Solid-State Technol. Mag.* **40**, 89–101 (May 1997).
13. S. E. Schuster, B. Chappell, V. DiLonardo, and P. E. Britton, "A 20ns 64K (4K × 16) NMOS RAM," *IEEE J. Solid-State Circuits* **SC-19**, 564 (1984).
14. R. Matick, *Computer Storage Systems and Technology*, John Wiley & Sons, Inc., New York, 1979.
15. R. E. Matick and D. T. Ling, "Architecture Implications in the Design of Microprocessors," *IBM Syst. J.* **23**, 264 (1984).
16. R. E. Matick, "Impact of Memory Systems on Computer Architecture and System Organization," *IBM Syst. J.* **25**, 274 (1986).
17. R. E. Matick, "Functional Cache Chip for Improved System Performance," *IBM J. Res. & Dev.* **33**, 15 (1989).
18. R. Matick, R. Mao, and S. Ray, "Architecture, Design, and Operating Characteristics of a 12-ns CMOS Functional Cache Chip," *IBM J. Res. & Dev.* **33**, 524 (1989).
19. D. Lamers, "IBM Embeds DRAM in 0.18-Micron ASICs," *Electronic Engineering Times*, February 22, 1999; Web-published at *http://www.eetimes.com/showArticle.jhtml?articleID=18300993*.
20. S. Deffree, "IBM Pushes eDRAM as SRAM Replacement," *Electronic News*, February 13, 2003; Web-published at *http://www.reed-electronics.com/electronicnews/article/CA276970?text=deffree*.
21. M. Clendenin, "IBM Makes Another Run at Embedded DRAM," *Electronic Engineering Times*, February 13, 2003; Web-published at *http://www.eetimes.com/showArticle.jhtml?articleID=18308029*; 144-Mb, 5.6-ns eDRAM to replace SRAM.
22. A. Cataldo, "IBM Pushes Embedded DRAM as NEC Changes Tack," *Electronic Engineering Times*, February 15, 2003; Web-published at *http://www.eetimes.com/showArticle.jhtml?articleID=18303603*.
23. V. Klee, J. Norum, R. Weaver, S. S. K. Iyer, C. R. Kothandaraman, J. Chiou, M. Chen, N. Kusaba, S. Lasserre, C. Liang, J. Liu, A. Lu, P. R. Parries, B. J. Park, J. Rice, N. Robson, D. Shum, B. Khan, Y. Liu, A. Sierkowski, C. Waskiewiscz, P. Wensley, T. Wu, J. Yan, and S. S. Iyer, "A 0.13 μm Logic-Based Embedded DRAM Technology with Electrical Fuses, Cu Interconnect in SiLK™, Sub-7ns Random Access Time and Its Extension to the 0.10 μm Generation," *Technical Digest, IEEE International Electron Devices Meeting*, 2001, p. 407.
24. J. Barth, D. Anand, J. Dreibelbis, and E. Nelson, "A 300 MHz Multi-Banked eDRAM Macro Featuring GND Sense, Bit-Line Twisting and Direct Reference Cell Write," *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, 2002, p. 156.
25. Y. Agata, K. Motomochi, Y. Yoshifumi, M. Shirahama, M. Kurumada, M. Kuroda, H. Sadakata, K. Hayashi, T. Yamada, K. Takahashi, and T. Fujita, "An 8ns Random Cycle Embedded RAM Macro with Dual-Port Interleaved DRAM Architecture (D2RAM)," *Digest of Technical Papers, IEEE International Solid-State Circuits Conference*, 2002, p. 392.
26. C.-L. Hwang, T. Kirihata, M. Wordeman, J. Fitfield, D. Storaska, D. Pontius, G. Fredeman, B. Ji, S. Tomashot, and S. Dhong, "A 2.9ns Random Access Cycle Embedded DRAM with a Destructive-Read Architecture," *Digest of Technical Papers, IEEE Symposium on VLSI Circuits*, 2002, p. 174.
27. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley Publishing Co., Inc., Boston, 1980, p. 6.
28. P. Richman, *MOS Field-Effect Transistors and Integrated Circuits*, John Wiley & Sons, Inc., New York, 1973, p. 220.
29. "Riesling 2K bit, 6 Device SRAM," IBM Engineering Specification No. 5123326, September 19, 1974; a ~4-μm technology chip that was used in main memory for the IBM System/360–158, 168, etc. high-end computers.
30. R. H. Dennard, "Field Effect Transistor Memory," U.S. Patent 3,387,286, filed July 14, 1967.
31. (a) Y. Taur and T. Ning, *Fundamentals of Modern VLSI Devices,* Cambridge University Press, New York, 1998; (b) *IBM J. Res. & Dev.* **39**, No. 1/2 (1995); (c) *IBM J. Res. & Dev.* **46**, No. 2/3 (2002).
32. S. Wolf, *Silicon Processing for the VLSI Era, Vol. 3—The Submicron MOSFET*, Lattice Press, Sunset Beach, CA, 1995, p. 291.
33. H. Kalter, C. Stapper, J. Barth, J. DiLorenzo, C. Drake, J. Fifield, G. Kelley, C. Lewis, W. van der Hoeven, and J. Yankosky, "A 50 ns 16 Mb DRAM with a 10 ns Data Rate and On-Chip ECC," *IEEE J. Solid-State Circuits* **25**, 1118 (1990).
34. IBM RISC System/6000 Technology; Publication SA23-2619. The cache described on p. 12 of "RISC System/6000 Hardware Overview" and on p. 44 of "Data Cache and Storage Control Units" is based on the cache described in Reference 14 of this paper.
35. IBM Blue Gene/L Team (IBM and Lawrence Livermore National Laboratory), "An Overview of the BlueGene/L Supercomputer," *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, 2002, p. 1; see also a forthcoming issue of this journal (2005).
36. R. C. Agarwal, F. G. Gustavson, and M. Zubair, "A High-Performance Matrix-Multiplication Algorithm on a Distributed-Memory Parallel Computer, Using Overlapped Communication," *IBM J. Res. & Dev.* **38**, 673 (1994).
37. P. G. Emma, "Understanding Some Simple Processor-Performance Limits," *IBM J. Res. & Dev.* **41**, 215 (1997).
38. R. E. Matick, T. J. Heller, and M. Ignatowski, "Analytical Analysis of Finite Cache Penalty and Cycles per Instruction of a Multiprocessor Memory Hierarchy Using Miss Rates and Queuing Theory," *IBM J. Res. & Dev.* **45**, 819 (2001).
39. P. Bakeman, A. Bergendahl, M. Hakey, D. Horak, S. Lute, and B. Pierson, "A High Performance 16-Mb DRAM Technology," *Digest of Technical Papers, IEEE Symposium on VLSI Circuits*, Honolulu, HI, June 1990.

Received January 26, 2004; accepted for publication June 15, 2004; Internet publication January 31, 2005

Richard E. Matick *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (matick@us.ibm.com).* Dr. Matick is a Research Staff Member in Systems Technology and Microarchitecture. After receiving his B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Carnegie Mellon University in 1955, 1956, and 1958, respectively, he joined the IBM Research Division and worked in the areas of thin magnetic films, memories, and ferroelectrics. As manager of the Magnetic Film Memory group from 1962 to 1964, he received an IBM Outstanding Invention Award for the invention and development of the thick-film read-only memory. He joined the Technical Staff of the IBM Director of Research in 1965 and remained until 1972, serving in various staff positions and as Technical Assistant to the Director of Research. In 1972 he took a one-year sabbatical to teach at the University of Colorado and IBM in Boulder, Colorado. Dr. Matick spent the summer of 1973 teaching and doing research at Stanford University. In 1986, he received an IBM Outstanding Innovation Award and in 1999 an IBM Corporate Patent Portfolio Award as co-inventor of the industry-standard video RAM memory chip, used to provide the high-speed, high-resolution display bit buffer in personal computers and many workstations. His work in high-density CMOS cache memory design, for which he received an IBM Outstanding Technical Achievement Award in 1990, served as the foundation for the high-speed cache system used in the IBM RISC/6000 series processors. He is co-initiator of the concept of logic-based embedded DRAM, which has become a key IBM strategy for systems-on-a-chip. It was initially conceptualized in 1990 and became a reality in the late 1990s, and is now being offered to all IBM ASIC customers. Dr. Matick is the author of the books *Transmission Lines for Digital and Communication Networks*, McGraw-Hill, 1969 (reprinted as an IEEE Press Classic Reissue in 1995 and in paperback in 2001), and *Computer Storage Systems and Technology,* John Wiley & Sons, 1977. He is also the author of chapters on memories in *Introduction to Computer Architecture* (H. Stone, Editor), SRA 1975 (First Edition), 1980 (Second Edition) and in *Electronics Engineers' Handbook*, Second and Third Editions, McGraw-Hill, 1982 and 1989. He also contributed the "Cache Memory" entry in the *Encyclopedia of Computer Science*, Third Edition, Van Nostrand Reinhold, 1993. Dr. Matick is a member of Eta Kappa Nu and an IEEE Fellow.

Stanley E. Schuster *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (schustr@us.ibm.com).* Mr. Schuster received B.S. and M.S. degrees in electrical engineering from New York University in 1962 and 1969, respectively. In 1965, he joined the IBM Research Division, where he currently is a Research Staff Member working in the VLSI Design Department in the area of low-power high-speed digital circuits and logic-based eDRAM for memory hierarchies. Upon joining IBM, he initially worked on n-MOS device characterization and circuit design for logic and memory. The work was part of the effort that led to the IBM n-MOS technology for main memory. He was also involved in the application of semiconductor technology to communication systems. During the early 1970s he did extensive work that demonstrated the leverage of word- and bit-line redundancy for semiconductor memory yield. His work on redundancy included the development of laser personalization techniques for integrated circuits. This technique was also used to achieve fast turnaround times in the personalization of PLAs. In the late 1970s he started a research effort on the design and application of a series of n-MOS and CMOS memory chips for very-high-speed operation. This work included the design of very-high-speed CMOS ECL-compatible SRAMs with cycle time less than access time. A paper describing this work received the 1992 IEEE International Solid-State Circuits Conference Lewis Winner Award for outstanding paper. This work was instrumental in demonstrating that CMOS could move into the high-speed arena that was previously the domain of bipolar technology. Mr. Schuster was an Associate Editor of the *IEEE Journal of Solid-State Circuits* (ISSCC) from 1988 to 1992 and was Guest Editor of its Special Issue on Logic and Memory in October 1986. He served on the ISSCC program committee from 1985 to 1988. He was Co-chairman of the IEEE Solid-State Circuits and Technology Workshop Committee and a member of the AdCom of the IEEE Solid-State Circuits Society from 1994 to 2003. Mr. Schuster has 41 issued patents and more than 54 published papers. He is a Fellow of the IEEE and a member of the IBM Academy of Technology. He has received six IBM Outstanding Invention and Contribution Awards and a Sixteenth Level IBM Invention Achievement Award. He is also the recipient of an IBM Corporate Award for his work on high-speed CMOS SRAMs.