Architecture, design, and operating characteristics of a 12-ns CMOS functional cache chip

by Richard Matick
Robert Mao
Samuel Ray

The architecture, design, and implementation of a high-performance cache require a detailed consideration of the overall system functions closely coupled with the proper mapping and integration of these functions into the circuits and arrays. This approach has resulted in a new cache chip which incorporates a number of unique on-chip functions as well as unique design, providing a one-cycle cache in which translation can be overlapped with cache access. In order to achieve high average performance, a cache should give the appearance of being a two-ported array in order to provide high bandwidth both to the processor during normal execution and to the main memory during reload. But a true two-port design is undesirable as well as unnecessary, especially since the reload process is typically limited by the memory speed and other system parameters. A significant improvement can be obtained by judicious choice as well as proper integration of some critical functions placed directly on the cache chips. This paper describes these functions and integration onto a 72K-bit static RAM chip, implemented in 1-μm CMOS technology for high speed and overall system performance. In addition, the chip I/O is selectable for either ECL or TTL compatibility.

Introduction

The advantages of integrating the proper functions on a single chip have been widely recognized with respect to processor design [1, 2]. However, the advantage provided by such integration in cache design has only begun to receive attention [3]. An increasing emphasis on cache design is crucial for all future systems for fundamental reasons, as follows.

The large performance gap between CPU and main memory has made the use of a cache an important factor for any high-performance processor, be it microprocessor, mid-sized or large system. However, a cache is valuable only if the total *average* time to access the cache is much less than that to access main memory for the same data. This total average time includes the usual access time when data is resident in the cache, plus a weighted average time for access misses and subsequent reload.

In cache design, there are many parameters that affect the overall performance. Some of these are determined by the technology and design trade-offs and thus are more closely under the control of the designer, while others are not, being based on statistical averages of the current workload at any moment. Attempts to improve cache performance have appeared over the entire spectrum, from faster array designs to methods for fine-tuning the cache hit ratio using look-ahead and pre-reload. The approaches can be classified into two broad areas, namely those which attack some fundamental hardware limitation which can be well qualified, and those which attack more loosely defined systems problems which are not easily qualified since they depend on the application, programming style, and probability. Both areas are important for performance improvement, but the former, by attacking fundamental problems inherent in all cache design, can be a more generally applicable solution, not limited to a particular set of problems or conditions. However, this approach has traditionally made use of logically simple but fast random-access memory arrays with other necessary or desired functions done elsewhere, using complex logic as needed. In such cases, because the required functions are not well integrated, extra path delays, chip crossings, and additional circuits are encountered. This paper presents the design of a cache array chip with simple, well-integrated functions which greatly enhance the performance with minimal amounts of additional hardware. The major problem solved by this approach is the minimization of average system performance degradation due to a cache miss while still providing the means for fast access during a normal read/write cycle. Not only is this design applicable over a wide range of problems, but it can be used for both minicomputers and large systems.

The architectural and general functional specifications of an optimal functional cache are detailed in [4], which gives particular attention to minimizing the number of machine cycles for accessing and reload of a miss, with little attention to the actual circuits and array design or speed. While there are many ways in which such functions can be designed in the actual circuits and laid out on silicon, the actual design chosen will determine the final speed of the chip. If the system cycle time is not of prime importance, but rather only the functional features for access and reload, the chip and circuit design will be much simpler than that for the case of achieving a fast chip. The approach adopted here was to design the fastest possible cache chip which incorporates all the characteristics of an optimal, functionally integrated cache architecture. A summary of some of the key functions implemented on the chip follows.*

1. *Late-select capability.* For high speed, it is necessary to access the array for the entire congruence class in parallel with the translation via the cache directory. For a set associativity of 4, this requires accessing four words and selection of one of these very late in the cycle, after translation.
2. *Late-write capability.* For a write access, the correct word of the congruence class to be written is not known until very late in the cycle, after a data-read has taken place. Typical array designs require the data to be valid at the beginning of the cycle and hence necessitate two cycles for writing such a cache. This penalty can be avoided by an array design which allows the data to become valid late in the cycle.
3. *Load-through path.* On a cache miss, the reload starts at the word which caused the miss. This word is loaded from main memory to the cache for storage, and also is passed through the cache directly to the processor data bus. If it was a read access, the data is latched in the processor for an immediate load on the same cycle, and the processor can restart before the reload is completed. For this chip, only the first word need be loaded through, since subsequent reloading words can be obtained from the reload buffer.
4. *Cache-reload buffer (CRB).* Words reloading from main memory are placed first into the CRB. While this is taking place, the CPU may access words which have already been loaded into the CRB, or any other data already resident within the cache array. Ideally, all the controls for achieving this should be on the cache chip.
5. *Store-back buffer (SBB) (for store-in cache).* If changes in data are contained only in the cache and not written through to memory, a severe reload penalty is encountered when a cache miss requires reloading of a changed block and hence a write-back to main memory. An off-chip store-back buffer can be and often is used, but typically requires many cycles to load. By placing the store-back buffer on-chip, and, more importantly, by proper integration with the arrays and overall system, the store-back time can typically be overlapped with other functions. This is the design path chosen for this chip.

* See [4] for a more thorough discussion of the overall functional requirements.

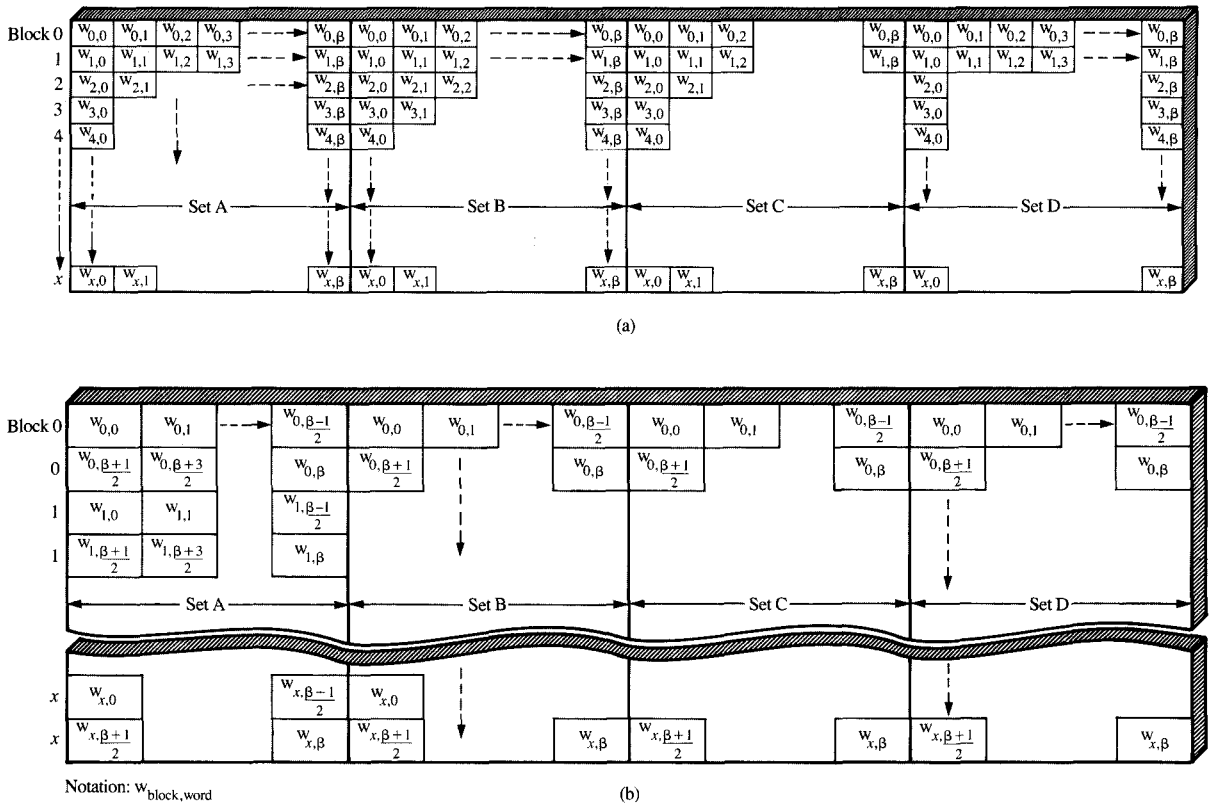

Figure 1

Mapping of sets, blocks, and words to chip units for late-select, four-way set-associative cache having $x + 1$ blocks of $\beta + 1$ words per block for (a) one and (b) two rows per congruence class (from [4]).

6. *Character write-select capability.* Most processors have the capability to manipulate individual bytes while typical memories access on a word boundary. Changing one byte often requires a multi-cycle read–modify–write operation. In this design, one byte per logical island is a natural boundary. Thus, one pin per logical island is used as a byte-select signal which can also serve as the chip-enable signal.

Mapping function onto silicon

Mapping these desired functions onto real silicon in a workable, practical fashion is a task with several parts, as follows.

- *Logical-to-physical mapping of array bits;* partitioning of bits, bytes, and words in some modular fashion to satisfy the system requirements for capacity, set-associativity (in a late-select cache design), and reload bus-width.
- *Chip architecture;* logical/electrical structure, design, and timing for implementing the functions without compromising speed.
- *Floor plan* and *physical design* for overall structure and functionality with high speed in the given technology.
- *Performance measurements* to determine how well the design implements the functions.

- *Logical-to-physical mapping*

The mapping of bits, bytes, and words to the actual physical array requires careful consideration of all the functions which the chip is to perform. This is discussed in considerable detail in [4], which derives a practical mapping well suited for high density, speed, and modularity. This mapping is shown in **Figure 1**; its use in

accommodating the logical structure of the desired functions on a 72K-bit CMOS chip is shown in **Figure 2**. The latter only shows the logical data flow for the major functions which have been added on a chip unit, where a chip unit would be four chips in this case. A chip unit is the group of chips required to give the full CPU logical word width. The chip described here has a nine-bit (one-byte) I/O to the CPU; four such chips in parallel (e.g., Figure 4) make up a chip unit to supply a four-byte word width. The store-back buffer and the cache-reload buffer are similarly partitioned across several chips; hence, only part of each is on any one chip. The chip described here has 72 bits (eight bytes) of the SBB and 72 bits of the CRB on each chip. The full data flow and partitioning for the chip are shown in **Figure 3**. The array consists of 256 physical word lines by 288 pairs of bit/sense lines. Each physical word is logically partitioned (by the decoders) into four groups, with each group having 256 physical words by 72 bits per word. These four groups correspond to the four sets A, B, C, and D in a four-way set-associative design, as shown in Figures 1 and 2. During any access, all 288 bits along a word line are sensed and latched before any bit-line selection takes place. A normal access reads or writes one byte per chip (nine bits) out of or into the array cells through the bottom bit switches and decoder shown as the boxes "Select 4 × 9 out of 288" and "Select 9 out of 4 × 9" in Figure 3. For reload, 72 bits which all belong to the same set A, B, C, or D are read or written from/to the array through the top bit switches and decoder shown as the box "Select 72 out of 288 bits" in Figure 3. The external bus width to main memory for reload is 18 bits, and the array-memory path is buffered by the store-back buffer and cache-reload buffer as shown. Further details of the integration and operation of this design are given in subsequent sections.

This chip can be used in a number of different cache configurations. For instance, a 32-bit CPU logical word requires four such chips as a minimum with a minimum cache capacity of 32K bytes. The cache block size would be the size of the SBB or CRB on the four chips, or 4 × 8 = 32 bytes, and the reload path from main memory would be 4 × 18 bits or eight bytes (two words) per cycle. It is possible to double the cache capacity and block size while still maintaining the 32-bit CPU logical word size by using eight chips partitioned and connected as shown in **Figure 4**. Even words are stored on the left-hand group of four chips, and the odd words are stored on the right-hand group, as shown. The 36-bit CPU data bus from the two groups is dot-ORed, while the reload buses from the two groups run in parallel to double the total bus width. Other capacities, block sizes, and/or reload bus width can be obtained by various combinations, groupings, and use of selective control signals. For example, two

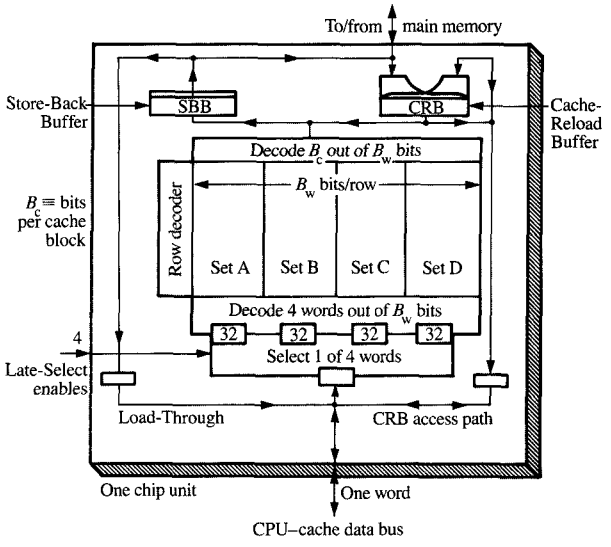

Figure 2

Functionally integrated chip unit showing major functions added for performance (chip unit = four chips).

configurations of that shown in Figure 4 can be used to achieve a 128K-byte capacity with various block sizes and reload bus width. The block size and reload bus width could be held the same as for Figure 4 alone by connecting the four-word reload bus of the two configurations in parallel and using the external control signals described later to select one or the other configuration of eight chips for reload. In a similar fashion, the CPU data bus width can be varied but, of course, the logical words must map correctly to the array and corresponding SBB/CRB partition for correct reload to/from main memory.

• *Chip architecture*

In any computing system or component, speed can usually be obtained by running several paths in parallel and pushing the slower signals required in the logic to be used as late as possible. For example, a late-select cache organization accesses the words from the entire congruence class at the start of the cycle and uses the slower, late-select signal to enable one of these words to/from the CPU very late in the cycle. In this manner, considerable performance improvement can be obtained. The same general principle can be applied to other parts of the system and is used quite extensively on this cache chip. Many functions are started in parallel, and the correct path is selected later, by the slower control signals. Additional speed improvement can also be

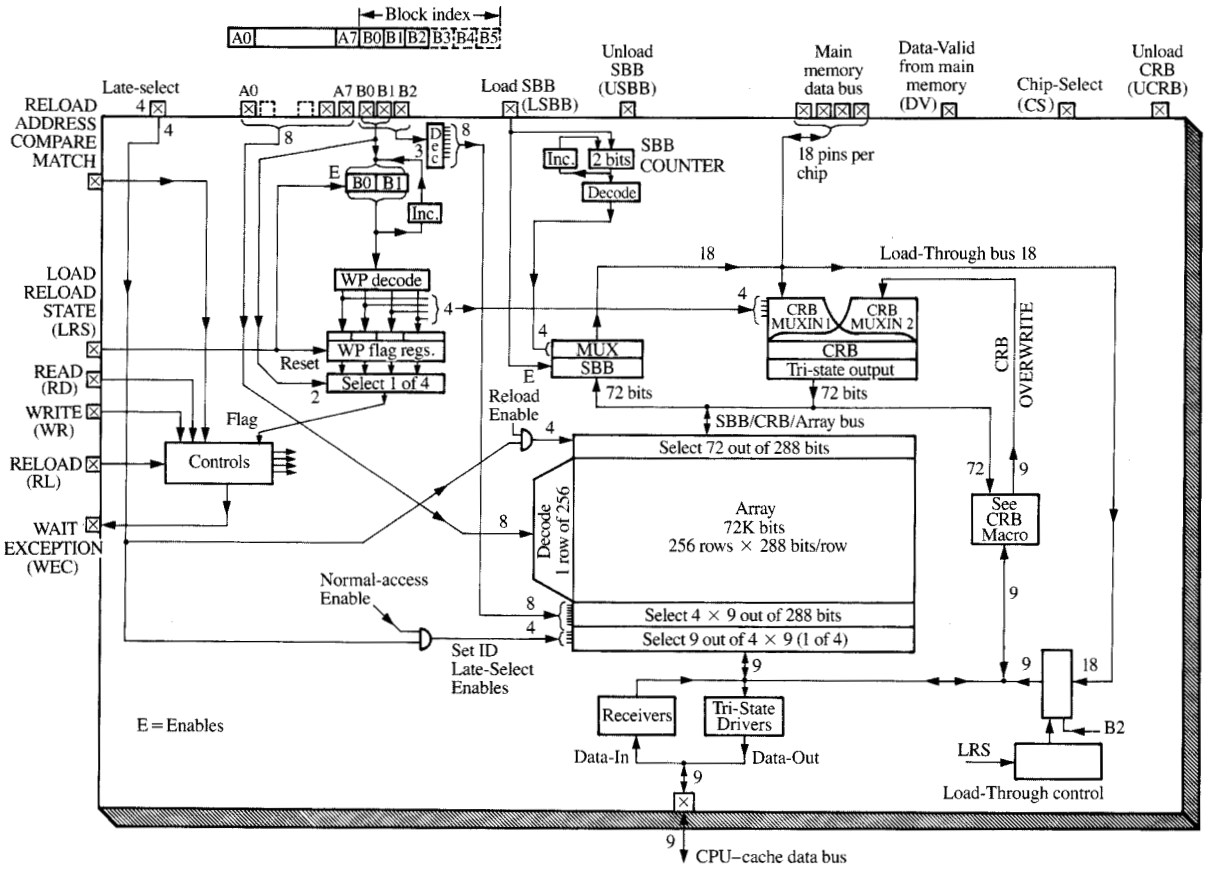

Figure 3

Schematic of functional cache chip showing controls and data flow.

obtained by using self-timing of critical paths, rather than some general clock pulse which guarantees allowance for worst-case delays. The latter must generally delay the clock by an additional amount equal to the difference between the worst- and best-case delay to ensure that an early clock signal does not occur. Hence, unnecessary delays are part of such a design and should be avoided if possible.

The general use of late-select and self-timing techniques to improve speed incurs a cost, namely some additional circuits and additional power. On a cache chip, the total amount of additional circuitry for parallel late-select operation and self-clocking is typically quite small, and the advent of CMOS technology makes the additional power consumption very modest. Hence, the improvement in speed is a good cost/performance trade-off.

A number of operations occur in parallel on this chip. Whenever the cache is in RELOAD mode and a new cycle is initiated by the nongated chip-select signal going valid, this starts an access to the array for the specified word address, and a simultaneous access to the cache-reload buffer. In addition, the off-chip reload address compare logic is also accessed in parallel, to determine whether the access is to the array or to the CRB. At some time in mid-cycle, the latter will gate data out of/into the array or CRB as appropriate. The use of a late-write design allows the identical, parallel operation with late enable for both a read and a write operation, a distinct advantage for such an application. In addition, the array access itself will simultaneously select one byte from each of the four sets; if the final access is determined to be to the array, one of the four late-select-enable signals will select one of these for a read or write.

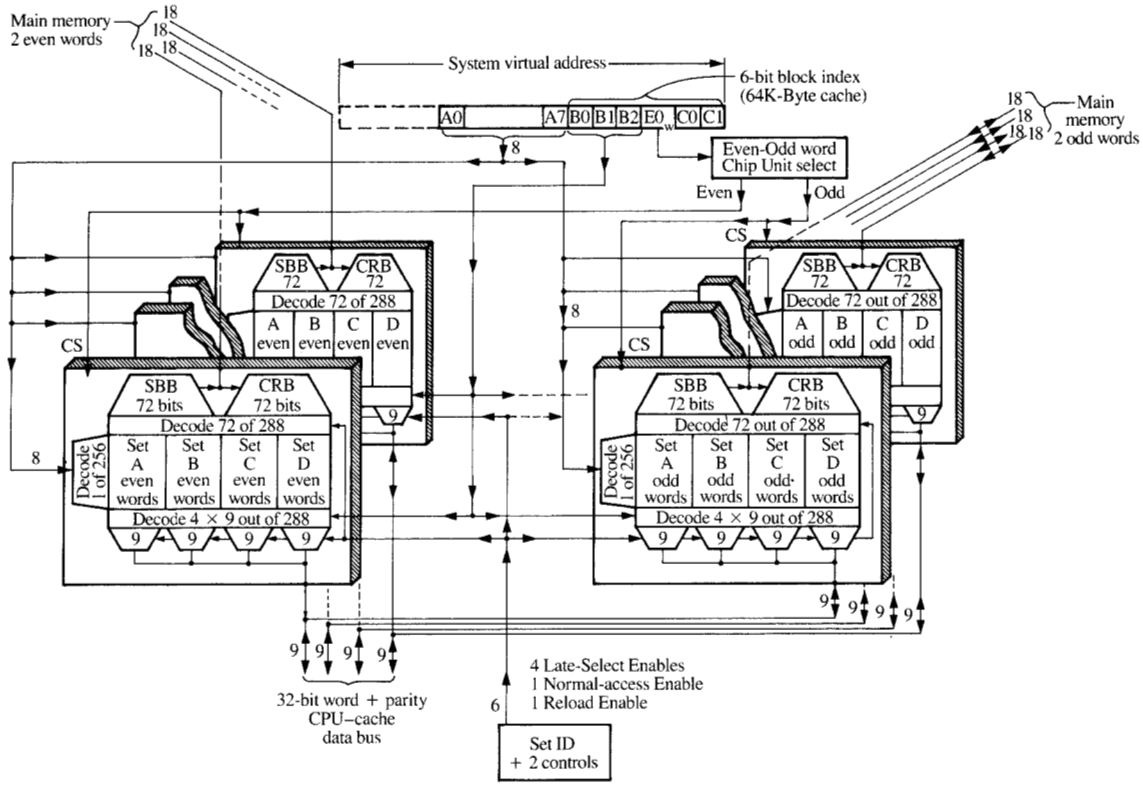

Figure 4

Organization and addressing for 64K-byte cache, 64-byte blocks, late-select with four-word reload path.

Architecture of the basic random-access array

While the array uses a standard six-device CMOS storage cell, the overall architecture, design, and operation of the basic random-accessing system are quite different from those of typical static arrays. A timing chain, similar in some respects to that in a dynamic chip, is initiated by the chip-select signal. However, there are many additional self-triggered timing signals used at various points to give versatility and speed. The general structure of the array accessing is illustrated by the timing diagram of **Figure 5** and proceeds as follows. The word address must be valid during the chip-select active period, since the address input buffer is enabled by $\overline{CS}$ going valid. The address is decoded by a precharged, dynamic NOR pull-down decoder which drives 256 word-line drivers. This circuit, shown in **Figure 6**, has a number of unique features. Whenever chip-select goes valid, the previously selected word-line driver output is quickly reset to ground, and all drivers are enabled for one to be selected by the decoder. This enable function is provided by the CS signal shown in Figure 6. The output stage which drives each word line consists of a cross-coupled latch with an extra pull-up device and a feedback line as shown. By the use of such a circuit, the selected driver is latched by feedback of its output signal (V_{dd} volts) and holds the word line high until the next $\overline{CS}$ signal goes valid. This allows the word decoder to be decoupled from the driver for decoder precharging, while still allowing a late write or read access to the selected word line. In fact, multiple accesses to bits on this latched word line can be made just by changing the bit address. This can be done until $\overline{CS}$ goes valid again. In parallel with this decoding and word-line driving are the bit-line equalization (to $V_{dd}/2$) and precharging of the on-chip, bi-directional (differential) I/O data bus shown in Figure 3, both also initiated by $\overline{CS}$ going valid. This precharging takes less than a quarter of the minimum access time and provides a faster access time. When $\overline{CS}$ goes invalid, all word-line

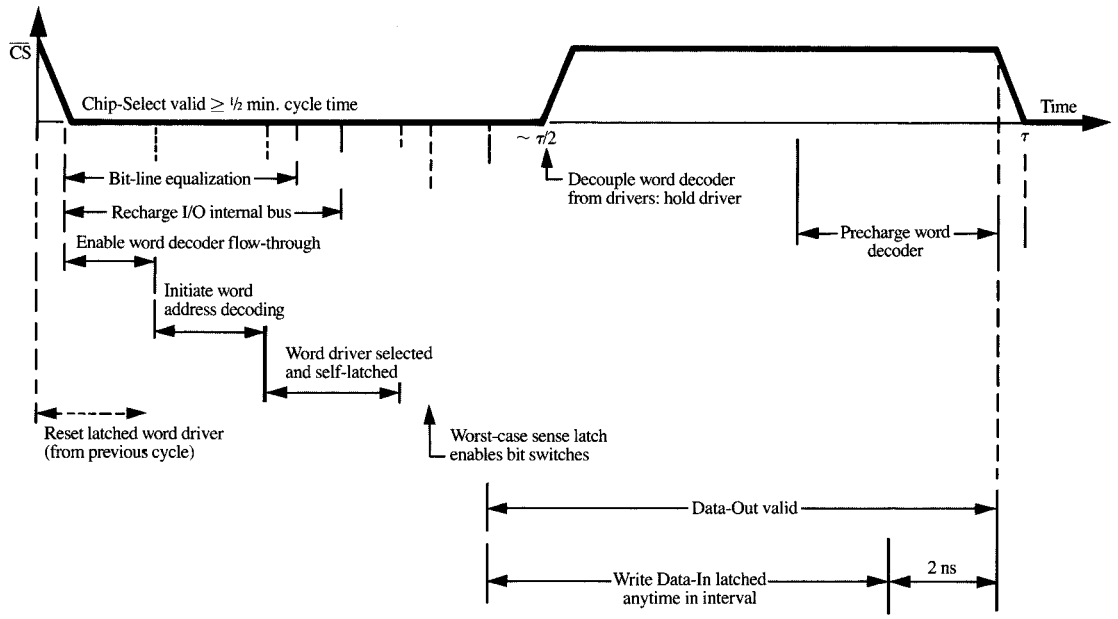

Figure 5

Timing of chip array accessing functions relative to the chip-select signal.

drivers, including the one selected, are decoupled from the word decoder so that the latter can be precharged on the second half of the cycle while the sensing and other functions are taking place.

The word address is latched immediately, at the beginning of a cycle. Thus, about 2 ns after $\overline{CS}$ goes valid, the external word address can be changed on the bus and has no effect on the chip.

In order to implement the desired functions and mapping without the use of complex multiplexors and long data bus lines on-chip, a separate set of bit-switch decoders is used on each end of the bit/sense lines. The bottom set interfaces to the CPU data path and the top set interfaces to the SBB/CRB bus for reload, as shown in Figure 3. These bit switches are disabled until all dynamic sense amplifiers are set—otherwise the excessive load would increase access time and could cause sensing errors. This self-timing is provided by a signal derived from a dummy word line with additional delay which ensures that the worst-case sensing is completed before any bit-line decoding takes place.

The sense amplifiers act as a preamplifier to quickly sense the differential signal across the bit/sense line pair of each cell and then pull the pair either high/low or low/high. Once the sense amplifiers are latched, the appropriate set of bit switches at either the top or bottom of the array are activated. For CPU accesses, the bottom 288 pairs of bit switches are enabled by a two-level decoder. The first level decodes the three bit addresses, B0, B1, B2, to select one of eight groups of switches where each group consists of four bytes (4 × 9 bits), one byte from each of the sets A, B, C, and D. One of the four late-select enables (already decoded) selects one of the sets in each of these groups and places the state of the nine sense amplifiers selected by bit switches onto the on-chip I/O bus. For a normal read operation, an external read signal, RD, will transfer the I/O bus information through the nine static, tri-state data-out drivers. A write operation is identical to a read except that the external write signal, WR, enables nine data-in receivers and drivers while the data-out drivers remain in tri-state. The write drivers are activated by the write signal, WR, in combination with chip-select, and have large devices which overwrite the state of the selected sense amplifiers and cells at any time before $\overline{CS}$ starts going valid for a subsequent cycle—see timing in Figure 5. (Note that WR can be valid at chip-select time, since its effect on the write drivers is internally delayed until about 5 ns into the cycle.) The normal I/O path from the bit switches outward operates in basically a static mode

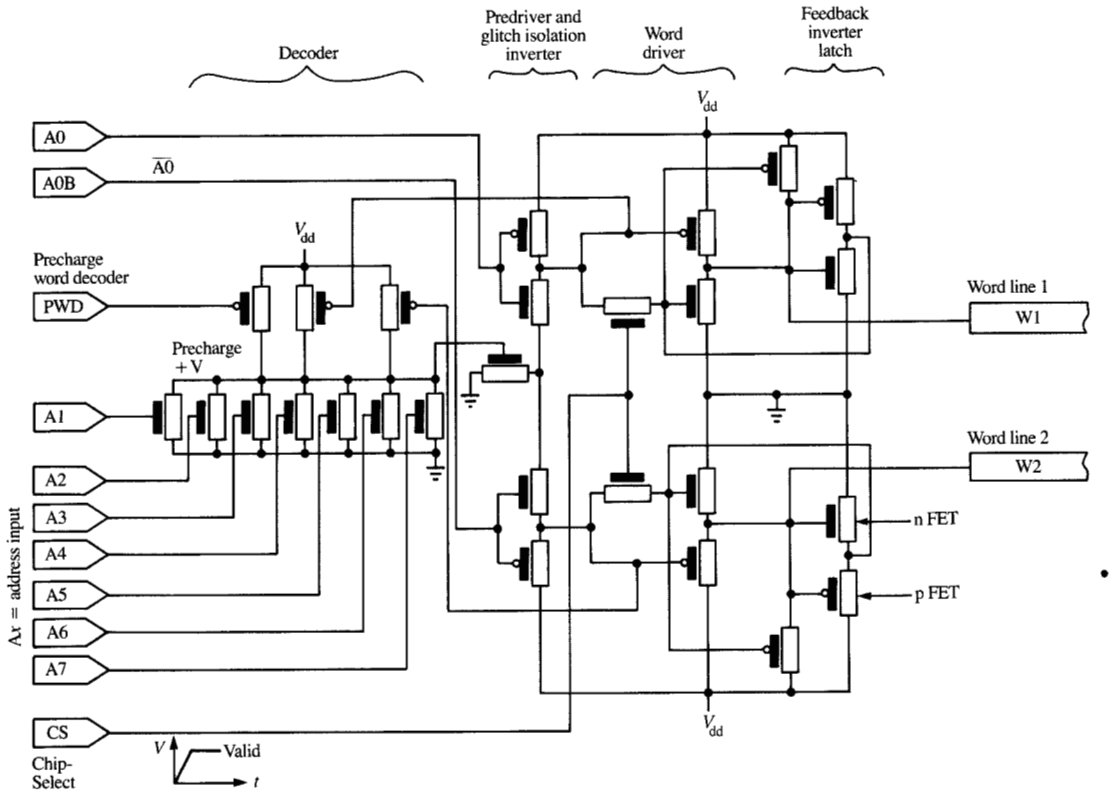

Figure 6

Word decoder and driver, showing decoder–driver decoupling.

with no latching of data in or out. Thus, the chip-select signal can be held valid (or can go invalid, as long as there is no transition from invalid to valid) and the bit addresses, B0, B1, B2, and/or late-select enables can be changed to read/write any of the 32 bytes which correspond to the selected word line within 6 ns without compromising the subsequent access/cycle time. This is due to the unique use of the separate precharged word decoder and self-latching word driver of Figure 6, described previously, in combination with a separate sense amplifier for each bit/sense line; it is comparable to a static column mode on a dynamic RAM. Since precharge timing is hidden inside the active cycle, the chip cycle time can be either less than or equal to the chip access time.

Static RAM chips designed for speed typically have the sense amplifiers located on one end of the bit/sense lines [5, 6]. In addition, there are usually far fewer than one sense amplifier per bit/sense line. More typically, four, eight, sixteen, or more sense lines share one sense amplifier. In this highly functional cache design, such an arrangement is not possible. Thus, to achieve fast access and cycle time in an array which makes use of complex functions on both ends of the bit/sense line, special consideration must be given to the sensing circuit partitioning, layout, and design. The need for one sense amplifier per sense line and different functions at the two ends of the sense lines resulted in a design with the sense amplifiers located in the middle of the array, as indicated schematically in Figure 3. (Figure 11, which gives the actual layout, is shown later.) While such a layout is often seen in DRAM design, it is not typically used in SRAM designs because of speed considerations.

It was previously pointed out that the bit switches at the top and bottom of the array in Figure 3 are not activated until the sense amplifiers are latched. This avoids the very large loading which would otherwise result from the peripheral functions added to this design. However, the goal of this design was speed, so further consideration was given to the problem of large bit/sense-line capacitance. In normal operation, bits located in the bottom half of the array must sometimes drive the

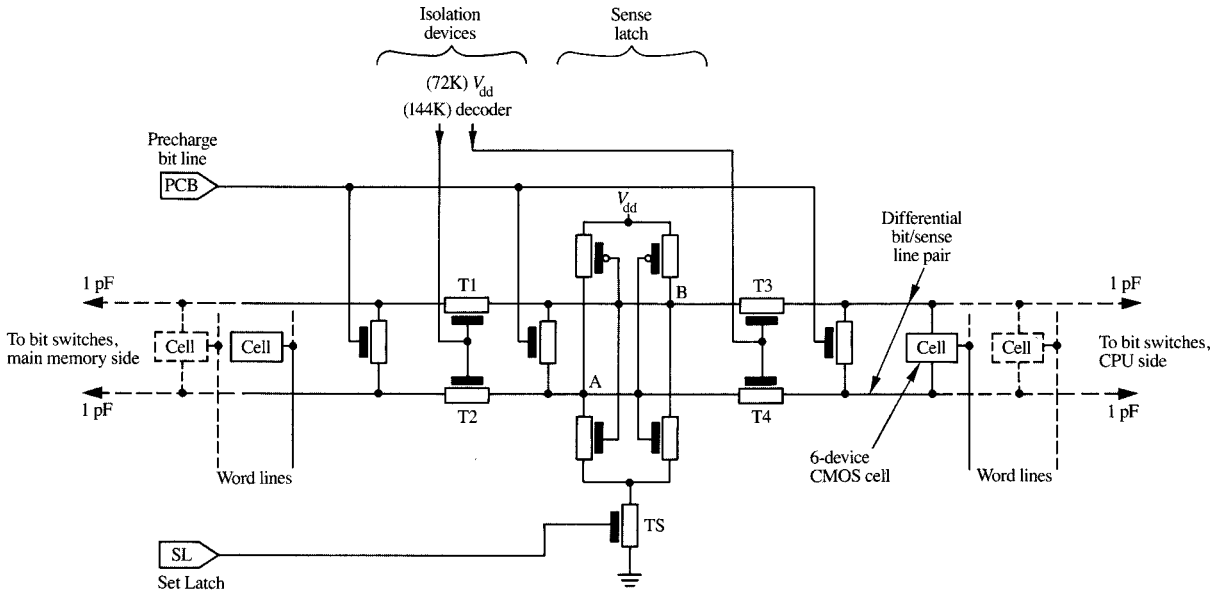

Figure 7

Differential sense amplifier and bit/sense line pair, showing precharge and isolation devices used for improved performance.

buffers at the top of the array. At other times, bits located in the top half of the array must drive the output circuits at the bottom of the array. Thus, each sense amplifier must eventually, at some point in time, drive the entire sense line across both halves of the array. Driving the entire sense line directly, during the cell sensing time, is a slow process. The design chosen isolates one or the other half of the sense-line capacitance until the sense amplifier latch is set. Once again, a similar technique is often used in DRAM designs to minimize the sense signal deterioration due to bit/sense-line capacitance, but it is not seen in SRAM designs. This isolation is achieved with the circuit of **Figure 7** as follows. Each sense amplifier has four isolation devices, two on the top side and two on the bottom side (note that Figure 7 is rotated 90 degrees compared to Figures 3 and 11). The precharging circuits are not shown. If a cell on the bottom array (right side in Figure 7) is being sensed, devices T1 and T2 are turned off, while T3 and T4 are turned on by circuits not shown. Thus, essentially only one half of the bit-line capacitance, the bottom half, affects the sensing during the initial phase. Whenever a substantial offset voltage, about 700 mV, is developed across the node AB in the sense latch, the latch is set via the bottom device, TS. At this time, the isolation devices T1 and T2 are turned on, and subsequently the bit switches at the top and bottom of the array. The entire sense line then rises up to $V_{dd} - V_t$, which is about 4 V. An analogous method is used to sense a cell in the top array.

In the initial 72K-bit design, the devices T1, T2, T3, and T4 had their gates all tied to V_{dd} rather than being selected. This only provides an RC phase-shift isolation between the top and bottom sense-line halves. For designs with larger sense-line capacitance, such as a 144K-bit design, selection of these devices as described above is necessary to maintain the speed. This sensing method increases bit-line differential signal, improves bit-line droop during sensing, and reduces chip power consumption. In addition, small, long-channel p devices in each sense amplifier and large push-pull drivers in the data-in circuits provide a fast bit-line overwrite capability. This late-write operation allows array cell modification to be performed at the back end of any cycle, for a one-cycle read or write, late-select cache.

In any chip design, fast time-changing currents through stray inductive and resistive components can cause ground and supply voltage bounce. Known as the *di/dt* problem, this appears fundamentally as unwanted signals. Since this functional cache chip is very fast and has many sense amplifiers, unlike typical static designs, special care was given to the *di/dt* problem. The use of one-half V_{dd}

precharging for sensing nearly eliminates any external current flow during the sense-line precharge time. Higher speed at low power and reduced di/dt are then possible. This results from the fact that before precharging, the differential sense lines will always have one of the pair at $V_{dd} - V_t$ and one at 0 V. The subsequent precharging of both lines to approximately half $V_{dd} - V_t$ consists of equalization of charge on both lines through a pass device shown in Figure 7. Half the charge from the high end passes to the low end, requiring very little current from the supply voltage, and the same is true for all sense-line pairs. The use of full V_{dd} precharging would require external current. Since one line in each of the 288 sense-line pairs would have to be charged each cycle, the di/dt problem would severely limit the speed.

In this chip design, there is still some ground bounce to consider during sensing. One of each of the 288 sense-line pairs must be pulled to ground at this time, which can cause a significant di/dt in the ground lines. However, the lines need only be pulled to ground from $V_{dd}/2$, rather than from V_{dd}, which cuts the current in half. Additionally, second-metal ground lines of 200 μm width, placed over the sense amplifiers and tied directly to ten ground pads, are used to reduce IR drop. A large second-metal V_{dd} bus supplies current to the bit lines. All 83 ground pads, and similarly all V_{dd} pads, are connected by second-level metal rings around the chip periphery to reduce inductive voltage drop on the module. Separate power pads are designated for output drivers. The combination of these factors reduces the di/dt problem to an acceptable level, as well as providing high speed at low power.

Even though this chip makes use of a number of dynamic circuits to obtain speed at low power, it nevertheless behaves functionally much like a conventional static RAM, with some additional features. In addition to the static column mode, it provides a late-select capability for late-select cache designs, as well as a unique late-read and late-write capability.

Architecture of peripheral functions

The operation of each of the special functions is described separately, with the required control signals (see [4] for background on these functions).

Store-back buffer (SBB) From Figure 2 it can be seen that this buffer is loaded from the top side of the array and unloaded onto the main-cache bus. Loading and unloading of this buffer occur as follows.

1. *Load SBB.* The SBB macro with all its required control and data signals is shown in **Figure 8**. It makes use of three control signals, LSBB to load the SBB from the array and reset its address counter to 0, and USBB (Unload SBB) in combination with DVB (DV bar) to unload the SBB and increment the address counter. In principle, it is not necessary to use DVB, but this is done to prevent burning out the chip—if the SBB tries to dump data on the main-cache bus at the same time that the SCU is doing so, the excessive loading can destroy the chip. The operation of this function is as follows. On a directory translation miss, if the block to be replaced is a modified block, the cache controller will latch the LSBB and RELOAD control signals, both being set valid. At the beginning of the next cycle, the LSBB and RELOAD are broadcasted to all chips. The array, as always, is accessed with the eight-bit word address, and 288 bits are sensed and latched. The set ID from the CPU is used by the top decoder and bit switches in the Array Macro to select 72 of the 288 bits, all 72 bits belonging to the same set. The LSBB signal causes the SBB to latch these 72 data bits, and also resets the two-bit SBB Address register to state 00. Even though this register will normally be in this state, this reset ensures the state on initial start-up.

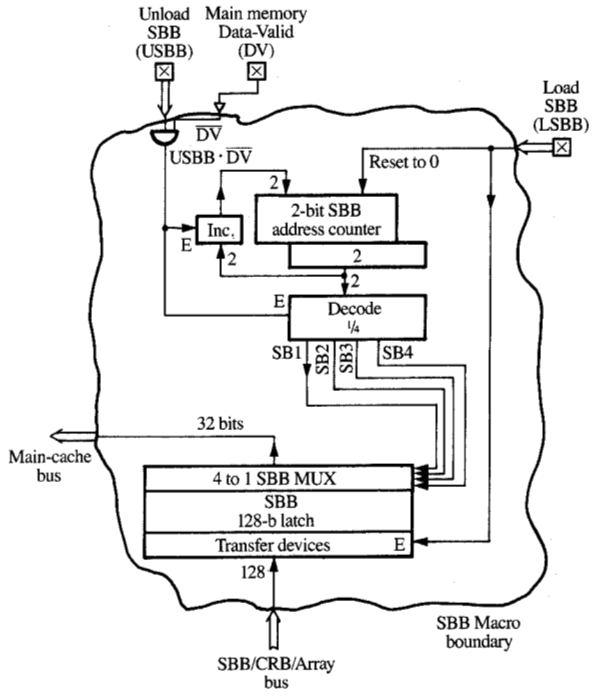

Figure 8

Store-back buffer macro showing data flow and controls.

2. *Unload SBB.* While the loading of the CRB from main memory can and will start on any quad-word boundary, the unloading of the SBB to main memory will always start on a cache-block boundary. Since main memory can be accessed by the CPU for reloading the translation lookaside buffer, or TLB (top priority), a page fault, or for a general I/O, the unloading of the SBB must be controlled externally by some controller which has all this state information. The cache is designed so as to not depend on which unit is in control. After the missed block has been reloaded (to the desired point, either into the array or held in the CRB), the cache controller unit controls the storing back of the SBB to main memory. The USBB signal, Unload SBB, must be issued four times. Each time, the current address in the two-bit address counter is decoded and accesses 18 bits from the SBB to the main-cache bus as shown, provided signal DV is invalid. The address counter is incremented by 1 at the end of each cycle, in preparation for the next store-back cycle. If signal DV is valid, the cache controller unit will inhibit the issuing of the USBB signal *or* reissue the operation, since the cache will not have responded when DV was valid. The operation of the SBB macro is totally independent of the other macros, except for using the same clocks. If the LSBB signal is issued at the wrong time, the SBB will latch whatever is on the SBB/CRB/Array bus.

Cache-reload buffer (CRB) In essence the CRB makes the cache chips look as close to a true two-port array as is necessary, and does so without using two-ported cells. From Figure 3 and **Figure 9** it can be seen that the CRB receives input from either the main-cache bus via CRB MUXIN1 or the chip I/O bus via CRB MUXIN2. The output of the CRB can go to either the array via the SBB/CRB/Array bus, or to the chip I/O bus via CRB DATAOUT MUX. While many of these functions can occur simultaneously, some are obviously mutually exclusive, selected by the control signals as defined below.

1. *Load and access CRB.* Whenever the cache is in the Reload mode, as indicated by the RELOAD signal RL being valid, the CRB is loaded from the main-memory bus each time the SCU data valid (DV) signal becomes valid. This loading is controlled by the two-bit addresses, B0 and B1, which are latched in an on-chip register when a miss occurs, and thus identify the word pair containing the word which caused the miss. These two bits are the lower-order portion of the reloading block address, the higher-order bits being stored in the reload address register (RAR) in the cache controller, as shown in Figure 9. These two bits are decoded by the word-pair (WP) decoder and produce four direct-enable signals, CR1 through CR3, as shown. These select the correct word-pair position in the CRB, starting at the word-pair boundary causing the miss. Also, these four enable signals set valid one of the WP valid flags in the WP flag register—this register is initially reset to invalid by the load reload state (LRS) signal at the beginning of the reload cycle. At the end of each cycle, the two-bit address is incremented by 1, and the four reload cycles will cause the two bits to cycle through four states and reset back to the initial state.
2. *Unload CRB.* After the fourth successive DV signal, the CRB on the eight chips will contain the full cache block. The CRB can be unloaded (written) to the array at any time, depending on the overall system control. For the simplest case, the CRB can be written back on the system cycle immediately after being fully loaded. This may waste a cycle if a cache data access is pending. Alternatively, the cache controller may be designed to defer the unloading of the CRB until another cache miss occurs, when there will be at least one free, available cycle. The design of the cache chip is such that it can be used in either mode. As long as the RELOAD signal is maintained valid, any access to the cache will access both the CRB and the array, with the final selection depending on the reload address compare match signal supplied by the cache controller. The CRB is written back to the array by issuing the Unload CRB signal; the WR signal must be invalid, or an undefined state will result.
3. *Normal read/write to CRB or array.* Once data has been placed in the CRB, it can be read or overwritten in a normal manner, or the array can be accessed for any other data resident in the cache. The effective address and Set ID of the block in the CRB are contained respectively in the RAR and Set ID registers, which are located in the cache controller, as shown in Figure 9. After the first load-through, the CPU can issue a new cache-access request. As long as the RELOAD signal is valid, the chips will check each access to see whether the requested data is valid, i.e., whether it has been reloaded. This is done via the WP flag, which is set valid for each word pair at the end of each quad-word reload cycle. If the current access is to the CRB, as indicated by the off-chip reload address compare match signal COMP1 * COMP2 being valid [with RL * (RD + WR) valid] but the flag is invalid, the word has not yet been reloaded and the cache sets the signal wait-exception current (WEC) cycle valid. This causes the cache controller to retry the last access repeatedly until the access is successful. Note that this allows the cache to work with any mode of main memory reloading, including multiple cycles between successive quad-words.

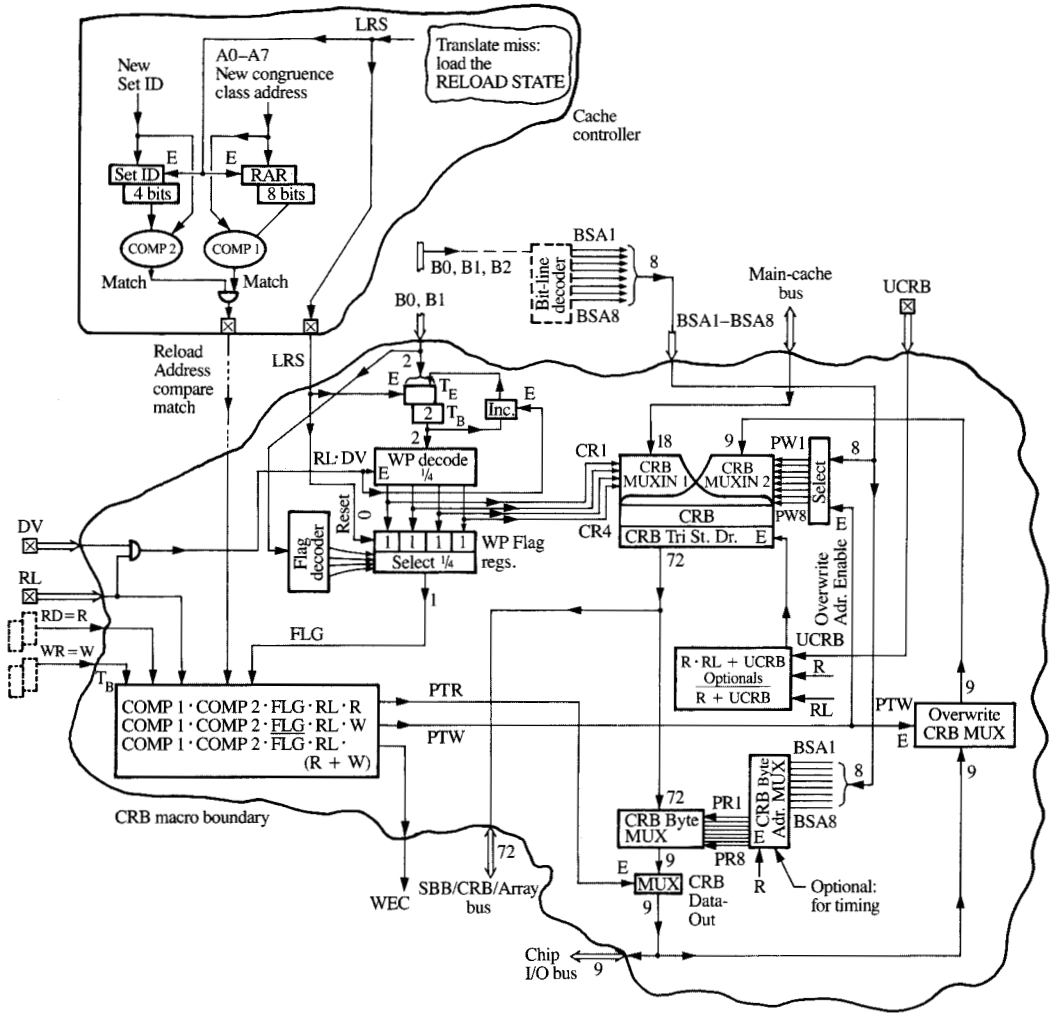

Figure 9

Cache-reload buffer macro showing data flow and controls.

If the current access is to the CRB and the flag is valid, then if the RD signal is valid, the CRB data-out MUX is enabled, and optionally (for timing, not needed functionally) the CRB byte address MUX is enabled. The result is that nine bits of the 72 CRB bits are placed on the chip I/O bus as shown in Figure 9.

If the above occurs, with the exception that the WR signal was initially valid, this enables the CRB overwrite MUX for data and the CRB overwrite address MUX. The previously entered byte will be changed within the CRB. Optionally, the same data could be written into the cache array on the same cycle.

Load-through buffer (LTB) The CRB only allows access (read or write) to words that have been previously reloaded. If the word which originally caused the miss was for a read access, one cycle will be lost while it is first loaded into the CRB and then read on the next cycle. Since read accesses are more common than write accesses, an LTB is provided to eliminate the lost cycle for reads. Note that the cycle is unavoidably lost for an initial write access. The load-through buffer shown in Figure 3 and **Figure 10** consists of an 18- to 9-bit MUX connecting the main-cache bus to the chip I/O bus. The 2-to-1 selection is done by address bit B2. Note that while

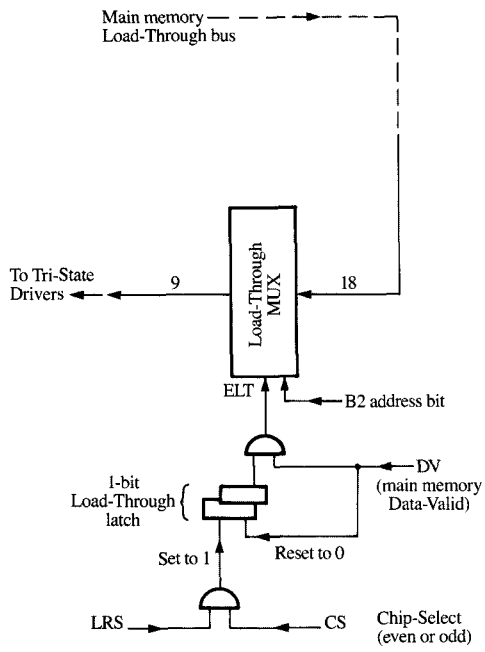

Figure 10

Schematic of load-through function and controls.

all chips receive data from main memory during reload, the load-through function must ensure that only four even or four odd chips activate the cache–CPU data-out bus for a CPU load. This even/odd bank select for CPU accesses is done by two separate external BYC (even/odd) signals—only one such signal per chip, but one for the even bank and one for the odd bank. The load-through process is enabled by an ELT (enable load-through) signal obtained from a one-bit LT latch ANDed with signal DV. This latch is set valid by the signal LRS (load reload state), which occurs only once, at the beginning of a miss/reload cycle, ANDed with the signal BYC. Note that the load-through is activated for either a missed read or write operation—the latter will just not be latched by the receiving unit.

The LT latch is reset invalid by the first DV signal. Thus, at the start of reload, the latch is set valid, waiting for the first word to come in from main memory. When the first quad-word arrives, as indicated by signal DV going valid, the word which caused the initial miss (either read or write) is selected and enabled onto the I/O bus for transmission to the CPU, and the latch is reset to 0. Since LRS is only valid once per miss, only one word is loaded through.

Floor plan and physical design

The basic array cell is a standard six-device cross-coupled static CMOS cell. The cell measures 16 μm × 24 μm and is not a ground-rule-minimum cell. Since the objective of this cache design is performance and reliability, the ratio of the cells' N-channel pull-down device width to the cells' N-channel transfer device width has been increased to 1.5 from the minimum of 1. The poly word line is stitched every 16 cells to reduce the word-line time constant. As shown in **Figure 11**, there are 18 gaps in the array, which are used for word-line stitching as well as for wiring channels to accommodate load-through and CRB data access to the CPU.

Since the CRB is accessed every cycle in parallel with the array, the large on-chip SBB/CRB bus width of 72 bits requires special attention. In order to minimize the ac power consumption and circuit delay, the CRB, SBB, and top bit switches (buffers in Figure 11) are aligned on four-bit pitches and share the same short bus. Grouping CRB and SBB registers in separate macros and running wiring buses across the entire chip could increase load-through and CRB access time approximately 5 ns due to bus-timing skew, and increase chip power consumption approximately 15%. Obviously, proper integration is necessary for maximum performance.

The ECL input buffers consist of a differential amplifier and buffer stages. Chappell et al. [7] have characterized this receiver to have 2 ns nominal delay with 50 mV differential signal input. The output drivers consist of a differential-to-single-ended converter and a tri-state line driver. Depending on the power supplies used, they are capable of driving either a 50-pF-terminated TTL load or a 50-Ω-terminated ECL load in 1 ns. At the beginning of a cycle, both differential input lines to each driver are precharged to V_{dd}, and each line driver output is in tri-state mode to speed up data access.

Performance

Figure 11 shows a photomicrograph of the chip, which is implemented in a 1-μm, 225-Å gate oxide CMOS process and has undergone extensive testing. Functional dice have been mounted on 36-mm ceramic substrates. **Figure 12** shows the external control and data signals at the pins for successive 20-ns cycles consisting of Write-1, Read-1, Write-0, Read-0. Long probe tips caused the ECL waveform of the $\overline{CS}$ (chip-select) signal to degenerate from a square into a sine waveform. The late-write capability of the chip is evident from the timing of the DI (data-in) signal, which becomes valid more than 5 ns after $\overline{CS}$ has become valid (gone low). The DI and WR (write control) signals need not be externally delayed, since WR input is internally disabled until about 5 ns after $\overline{CS}$ has become valid. Since the data-in circuits are static, the data can be applied to the chip at any time up

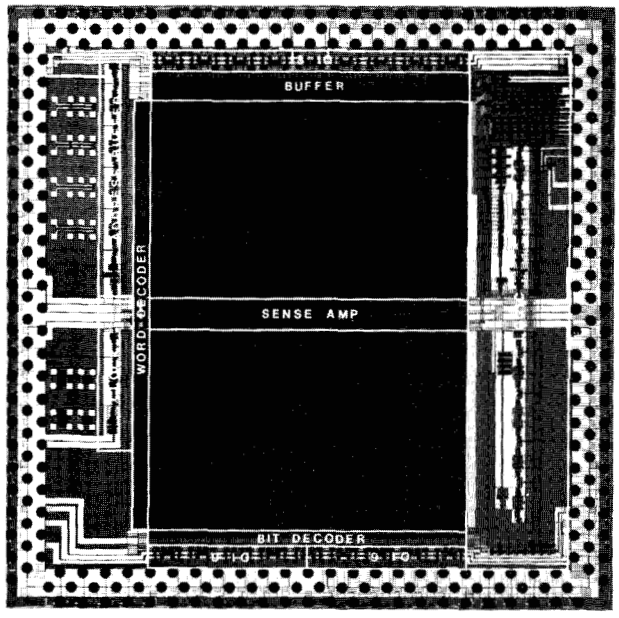

Figure 11

Photomicrograph of functional cache chip.

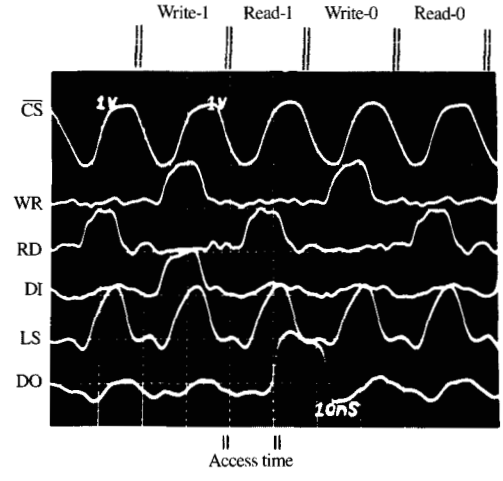

Figure 12

External ECL control and data signals at pins for sequence of 20-ns cycles (x = 10 ns/div., y = 1 V/div). $\overline{CS}$ = chip-select; WR = write control; RD = read control; DI = data-in; LS = late-select; DO = data-out, terminated through 50 Ω to −0.7 V. All signals are +0.5 to −0.5 V.

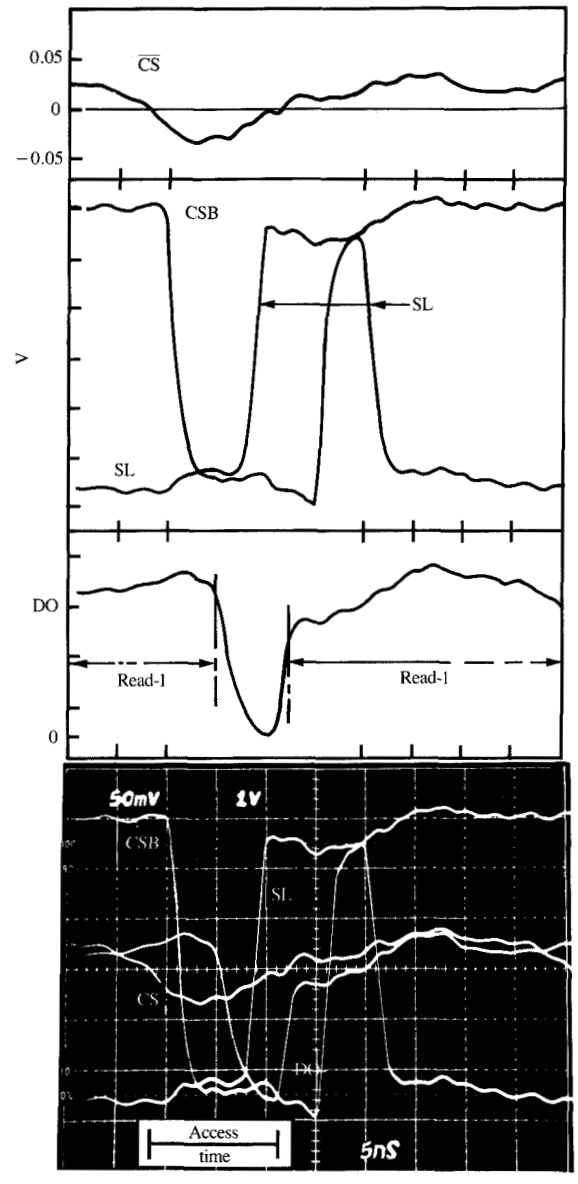

Figure 13

Measured chip-access time for continuous Read-1 cycles, using ECL-input, TTL-output signals: top, waveform identification; bottom, oscillogram. x = 5 ns/div., y = 1 V/div. (except $\overline{CS}$ = 50 mV/div.) $\overline{CS}$ = chip-select ECL signal (±25 mV); CSB = internal, delayed, inverted chip-select; SL = internal set sense-latch signal; DO = data-out signal (100-Ω termination) showing two successive "1" signals (1 = high V).

to about 2 ns before the end of the cycle, as shown in Figure 5. During a read cycle, the data-out (DO) remains valid even when the $\overline{CS}$ signal goes invalid, and data remains valid until about 3 ns after $\overline{CS}$ has gone valid for

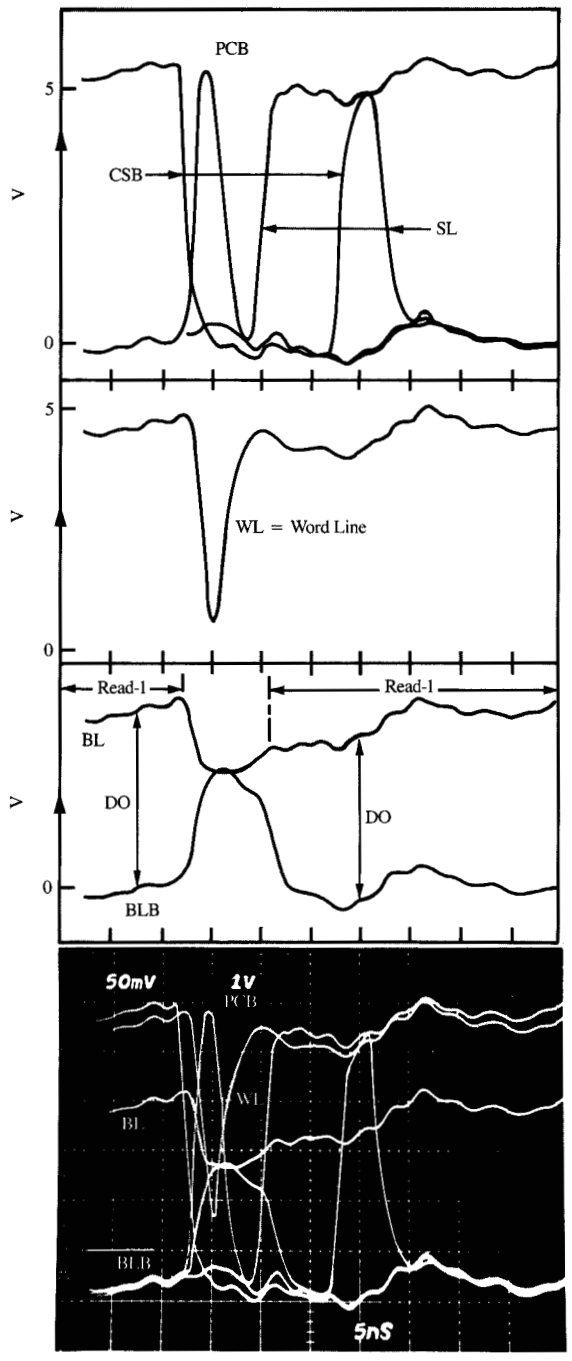

Figure 14

Internal chip signals for continuous Read-1 from same word line: top, waveform identification; bottom, oscillogram. (x = 5 ns/div., y = 1 V/div.) PCB = precharge bit line (bit-line equalization); CSB = inverted, delayed chip-select; SL = set sense latch; WL = selected word-line signal; BL, BLB = single-ended cell bit-line signals; DO = differential sense signal.

the subsequent cycle. This is shown in Figure 12, where DO during a Read-1 cycle goes low (invalid) after $\overline{CS}$ has gone low (valid). With nominal power supplies, ECL external signals, and 50-Ω resistor terminations, typical chip-select data access time is 12 to 13 ns. This is shown in Figure 12 and **Figure 13** as the time from chip-select ($\overline{CS}$) starting to go valid until data-out (DO) is full amplitude. In Figure 13, two internal chip signals, namely CSB (timing signal which is the inverted chip-select, two inverter stage delays after the receiver), and SL (set sense latch) are also shown. Some additional internal chip signals are shown in **Figure 14**, namely the selected word-line signal, WL, bit-line equalization signal, PCB (precharge bit line), and the two single-ended bit-line signals, BL and BLB, of the sensed cell.

Extensive testing has demonstrated full functionality of the store-back buffer, cache-reload buffer, load-through path, and other functions as described above. In addition, the chip can be used in a TTL environment by the use of a 5-V supply and a 2.0-V input receiver reference. The output driver power pads are connected to the TTL supply. This chip can also be operated in the IBM thermal conduction module (TCM) compatible environment. It requires four power supplies: −2.2 V (ground), −0.7 V, +1.4 V, and +2.8 V (V_{dd}). Output drivers use +1.4 V and −0.7 V to generate ±0.5-V ECL signals. The chip measures 9.4 mm × 9.4 mm, has 82 signal I/O pins (not all used for this implementation) and 287 total pins, and dissipates 0.5 W during standby and 2.0 W at a 12-ns chip-select cycle.

Conclusions

A high-performance CMOS static cache chip has been architected, designed, implemented, and tested. It can be used in a late-select cache organization with a one-cycle read and write capability. The introduction of new performance-enhancement features, architectural and electrical, improves overall system performance during normal accesses as well as reload of a miss. In addition to ideal matching of on-chip functions to the overall system requirements, the chip also is versatile and can operate in a TTL or ECL environment. The combined, optimized features give such an approach a distinct edge over ordinary cache designs which do not optimize at the system performance level. This approach to cache design is desirable not only in microprocessors and mid-range systems, but will find use in larger systems in the future.

The fundamental concepts pioneered in this work and in Reference [1] served as the basic building block for the high-speed, functional cache design in the second-generation RISC processor of Reference [8].

References

1. D. Fier, R. Caulk, P. Torgerson, D. Breid, R. Bradley, and K. LeClair, "A 36/72b CMOS Micro-Mainframe Chip Set," *Digest of*

Technical Papers, IEEE International Solid State Circuits Conference, February 1986, p. 26.
2. D. Alpert, D. Carberry, M. Yamamura, Y. Chow, and P. Mak, "32 Bit Processor Chip Integrates Major System Functions," *ELECTRONICS*, pp. 113–119 (July 14, 1983).
3. T. Watanabe, "An 8K Byte Intelligent Cache Memory," *Digest of Technical Papers*, IEEE International Solid State Circuits Conference, February 1987, p. 266.
4. Richard E. Matick, "Functional Cache Chip for Improved System Performance," *IBM J. Res. Develop.* **33**, 15–32 (January 1989).
5. S. Schuster, B. Chappell, R. Franch, P. Greier, S. Klepner, F. J. Lai, P. Cook, R. Lipa, R. Perry, W. Pokorny, and M. Roberge, "A 15-ns CMOS 64K RAM," *IEEE J. Solid-State Circuits* **SC-21**, 704–711 (October 1986).
6. F. Towler, J. Chu, R. Houghton, P. Lane, B. A. Chappell, and S. Schuster, "A 128K 6.5 ns Access/5 ns Cycle CMOS ECL Static RAM," *Digest of Technical Papers*, IEEE International Solid State Circuits Conference, February 1989, New York, pp. 30–31.
7. B. Chappell, T. Chappell, S. Schuster, H. Segmuller, J. Allan, R. Franch, and P. Restle, "CMOS High Speed ECL Receivers," *Digest of Technical Papers*, 1987 Symposium on VLSI Circuits, May 22–23, Karuizawa, Japan, pp. 91–92.
8. H. B. Bakoglu, G. F. Grohoski, L. E. Thatcher, J. A. Kahle, C. R. Moore, D. P. Tuttle, W. E. Maule, W. R. Hardell, Jr., D. A. Hicks, M. Nguyenphu, R. K. Montoye, W. T. Glover, and S. Dhawan, "IBM Second-Generation RISC Machine Organization," *Proceedings of the International Conference on Computer Design (ICCD '89)*, October 2–4, 1989, Cambridge, MA, pp. 138–142.

Received July 7, 1988; accepted for publication August 25, 1988

Richard E. Matick *IBM Research Division, T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Matick received his B.S., M.S., and Ph.D. degrees in electrical engineering from Carnegie Mellon University, Pittsburgh, in 1955, 1956, and 1958. He joined IBM in October 1958 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 Outstanding Invention Award for the invention and development of the thick-film read-only memory. He spent six months at IBM Hursley, England, developing this read-only memory for System/360 applications. Dr. Matick joined the technical staff of the IBM Director of Research in 1965 and remained until 1972, serving in various staff positions that included coordinator of Research Division plans and Technical Assistant to the Director of Research. He took a sabbatical in 1972 to teach at the University of Colorado and at IBM Boulder. During the summer of 1973 he taught at Stanford University while doing research there. He is currently working in the areas of VLSI functional memory chip and microprocessor design. In 1986 Dr. Matick received an IBM Outstanding Innovation Award as co-inventor of "video RAM," a new memory chip that is becoming popular for use in bit-buffered displays and is also used in the high-resolution display announced with the IBM PC RT. Dr. Matick is the author of the books *Transmission Lines for Digital and Communication Networks* and *Computer Storage Systems and Technology*. He is also the author of chapters on memories in *Introduction to Computer Architecture* and *Electronics Engineers' Handbook, Second Edition*. Dr. Matick has written numerous papers on magnetic devices and memories, semiconductor circuits, memory and logic, as well as virtual memory chips and systems. He is the holder of numerous patents and patent publications, and is a member of Eta Kappa Nu and a senior member of the Institute of Electrical and Electronics Engineers.

Samuel T. Ray *IBM General Products Division, 5600 Cottle Road, San Jose, California 95193.*