A BIT-SLICE CACHE CONTROLLER*

B. D. Ackland

Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

Cache storage is a proven memory speedup technique in large mainframe computers. Two of the main difficulties associated with the use of this concept in small machines are the high relative cost and complexity of the cache controller. An LSI bit-slice chip set is described which should reduce both controller cost and complexity. The set will enable a memory designer to construct a wide variety of cache structures with a minimum number of components and interconnections. Design parameters are based on the results of extensive simulation. Particular emphasis is placed on the need for design flexibility.

The chip set consists of three devices - an address bit-slice, a data bit-slice and a central control unit. Circuit design has been completed to a gate level based on an ECL/EFL implementation. The proposed structure will accommodate cache sizes up to 2K words with access times as short as 25 ns.

Introduction

The processing power of a digital computer is directly related to the speed and capacity of its memory system. Large, fast memories are expensive to implement in terms of cost, power dissipation and size, and pose severe high-speed interconnection problems. Accordingly, there is a need to find cost-effective memory speedup techniques which will significantly reduce memory access times without the associated cost and performance tradeoffs.

One such technique is the use of memory hierarchies - in particular the cache store concept. This approach to computer memory speedup has been well proven in the large processor situation, offering the designer cost-effective performance, programmer transparency and main memory isolation. Simulation studies carried out by the author [1,2] and others[3,4] have shown that this technique is also applicable to smaller machines where the economic constraints are more severe. Although cache-based systems have been successfully implemented in larger processors, manufacturers have been slow to employ the cache concept in small machines. It seems strange that a technique offering so much in terms of cost-effective speedup has not found acceptance amongst small computer designers.

There is no doubt that one of the main difficulties associated with the use of cache memory in small machines is the relative complexity of the cache controller. In larger computers, the cost of design and construction is readily absorbed by the high overall cost of the machine. In mini- and microcomputers, however, the implementation of a suitable cache controller can represent a significant proportion of the total design effort. A further problem is that as the speed of bipolar memories increases, so the speed of the cache control logic must be upgraded. This, in turn, may require the use of special construction techniques and elaborate debugging and test facilities. In addition, the advent of relatively fast, high-density MOS RAM's has offered an alternative structure to the memory designer. A single-level MOS memory, whilst not being able to match the performance of a well designed cache system, does offer the designer reasonable, low-cost performance and hardware simplicity. A possible solution to the cache design problem is to integrate some of the control functions on to a number of LSI circuits. This would lead to:

(1) reduced chip count,

(2) reduced interconnection leading to less noise and crosstalk

(3) greater speed due to a reduction in the number of off-chip buffers,

(4) reduced power dissipation, and

(5) design simplicity.

This paper describes the design of such an LSI chip set which will enable a designer to construct a wide variety of cache structures with a minimum number of components and interconnections. Particular emphasis is placed on the need for design flexibility. A proposed implementation of the chip set using emitter-function logic (EFL) is also presented.

Basic Structure

A cache memory system is one in which a small fast cache (or buffer) store is interposed between the main memory and the CPU as shown in Figure 1. The cache store is very fast (typically implemented in bipolar technology) so that it can match the instruction execution rate of the CPU. It is also of limited capacity and can hold only a small portion of the contents of main memory at any one time. Associated with the cache store is a control unit whose task is to control information transfers between main memory, the cache and the CPU. It is of necessity a hardware unit as software control would be intolerably slow. The cache control unit (CCU) aims to keep the most frequently used information in the cache store, thereby ensuring that the majority of accesses are satisfied by the fast buffer. Incorporated in the CCU is a *tag* memory which lists those portions of main memory currently held in cache. Main memory and cache are divided up into a number of equal sized blocks.* Associated with each block of information in cache is a tag word which contains the address of the information in main memory.

* This work completed as part of author's Ph.D. thesis in the Electrical Engineering Department, University of Adelaide, South Australia.

* A block is the unit of transfer between main memory and cache and typically consists of 1, 2, 3 etc., contiguous words.

CH139406/79/0000-0075 $00.75©1979 IEEE

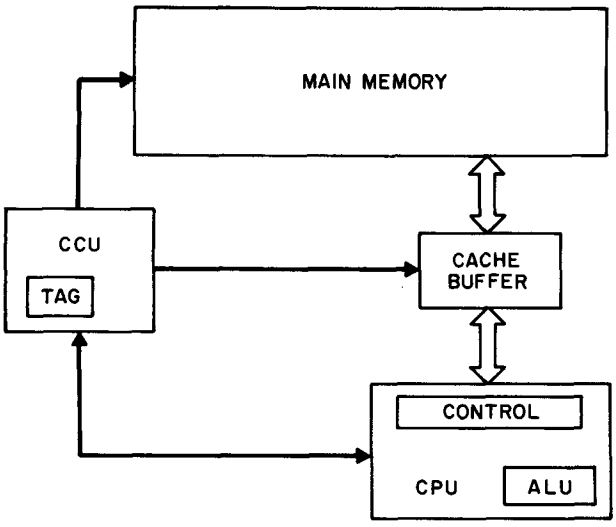

Figure 1 Cache Memory System

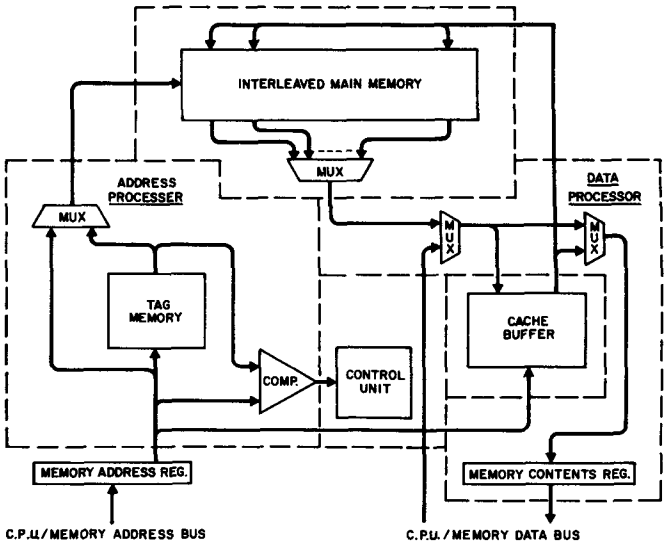

Figure 2 Generalized Cache Structure

The design of a cache controller may be likened to the design of a simple microprocessor. A cache control unit is, in essence, a fixed program address processor using the tag memory for data storage and controlling various peripherals such as main memory, the cache buffer, data multiplexers, etc. Accordingly, some of the problems encountered in microprocessor design are common to the cache control situation.

The task of integrating a cache controller on to a single LSI chip is fraught with difficulties. These include packing density, power dissipation, pin-out limitations and lack of flexibility. Such problems are also encountered in the design of microprocessors. Although the single chip approach has been successfully used in the construction of relatively slow microprocessors, high-speed design has been restricted to an alternative technique: bit-slice architecture. A bit-slice microprocessor divides its arithmetic processing power up into a number of equivalent LSI chips. The advantages of this approach include reduced gate count, power dissipation and pin count per chip in addition to increased design flexibility.

It follows that bit-slice architecture may be suitable for cache controller design. Consider the generalized cache memory system shown in Figure 2. It consists of a number of functional subunits viz.,

(1) Interleaved main memory - including associated latches, decoders and multiplexers.

(2) Cache buffer - including address latches.

(3) Address processor. This section includes the tag memory, tag address comparator and main memory address multiplexer. These devices handle tag (sector) addresses only.

(4) Data processor. This section includes input and output multiplexers and the memory contents register (MCR).

(5) Controller. The control unit communicates with the CPU and controls all other sections of the memory system.

The address and data processing sections may be likened to the arithmetic sections of a conventional bit-slice microcomputer. Similarly the function of the control unit corresponds to that of a microprogram controller. It seems reasonable, therefore, that bit-slice architecture should be applied to the cache control problem. Accordingly a three chip set is proposed comprising 1) an address bit-slice, 2) a data bit slice, and 3) a controller chip.

Design Parameters

Before proceeding to a detailed design of the component chips, it is necessary to describe suitable specifications for an integrated cache system. The following design parameters are based on minicomputer simulation results and the author's understanding of typical small computer system requirements.

Mapping Algorithm

This is the transformation rule by which main memory blocks of information are mapped into cache address space. Simulation results[2] showed that significant speedup can be obtained from both the set-associative (using a two block set) and direct mapping algorithms.* The set-associative scheme gives superior performance at the expense of increased hardware complexity. In the design of an integrated controller, circuit complexity is of secondary importance being limited mainly by gate count/package. Design was therefore based on the set-associative algorithm with a set size of two blocks.

Cache Size

Optimum cache size depends on processor architecture and the software environment in which the machine will operate. Simulation results showed that effective speedup can be gained from buffers ranging in size from 256 words to 1024 words. To be useful in a wide range of applications, therefore, an integrated cache controller should be able to support varying cache sizes up to a maximum of at least 1024 words.

Cache Speed

The access time of the cache buffer is determined primarily by the cycle time of the CPU. An integrated controller must therefore be able to accommodate a wide range of cache access times down to at least 50 ns.

Main Memory Speed

Main memory cycle times are also likely to show considerable variations depending on the application. An integrated controller must therefore be capable of efficiently using any reasonable main memory speed.

---

* For a description of the various cache mapping algorithms, the reader is referred to the literature[5].

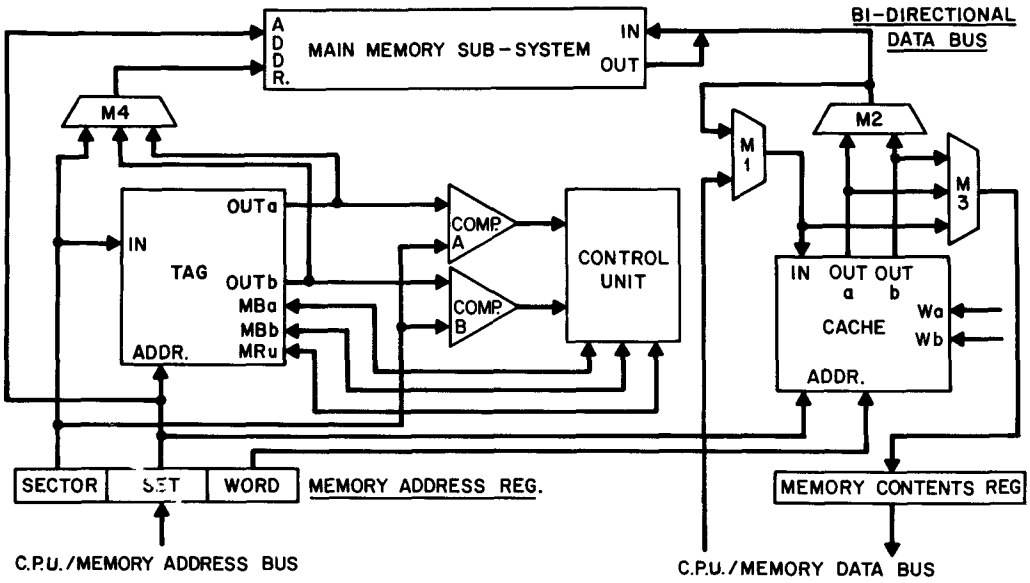

Figure 3 Block Diagram of Proposed Integrated Cache System

Block Size

Minicomputer simulations showed that improved performance could be obtained by using a multiple-word block in conjunction with main memory interleaving. The optimum size was found to be either two or four words depending on cache size and type of program. Accordingly it is felt that the controller should be able to support block sizes of one, two or four words. To minimize main store/cache interconnection problems, a time multiplexed data bus is chosen as the mode of block transfer. This simplifies circuitry external to the chip set at the expense of internal controller complexity.

Other Parameters Main memory size is determined by system programming requirements and not by the cache designer. The controller must therefore be able to accommodate any main memory size. This specification is satisfied by the proposed bit-slice structure of the address processing section. Word width is another system parameter outside the control of the cache designer. The proposed bit-slice structure of the data processing section allows the cache controller to accommodate any work width.

Write-back strategy is the algorithm which determines when and how data is written back from cache to main memory. On the basis of simulation results, a conflicting-modified-writeback (CMW) strategy[2,3] has been adopted. In this scheme, data is written to main store as it is displaced from cache and only then, if the data was modified whilst in cache.

CHIP SET ARCHITECTURE

Figure 3 shows a block diagram of a complete cache system based on the specifications of the previous section. Note that there are two outputs from the tag memory and two address comparators. This allows simultaneous comparisons of the two block tags associated with a given cache set. The cache buffer is also split into two pages - each having a separate output bus and control lines. In this way, buffer and tag access time can be overlapped to achieve minimum cache access delay.

Set associative mapping requires the use of two *modified bits* per set (MB_a, MB_b) - one for each block. In addition, there is a *most-recently-used* (MRU) *bit* associated with each set which is used to implement a 'least-recently-used' (LRU) replacement algorithm. These status bits are stored in tag memory alongside their corresponding address tag.

A bi-directional main memory data bus is employed to reduce external chip interconnection between main store and cache. Transmission delays caused by the need for bi-directional drivers are not likely to be significant due to the buffering property of the cache.

Address Processor

The address bit-slice processing chip needs to be able to handle tag address bits, *modified bits* and the MRU bit corresponding to each cache set. A block diagram of a circuit capable of processing a one-bit slice of sector address information is shown in Figure 4. It consists of two S word by 1-bit memories (where S is the number of sets in cache), two 1-bit comparators, a three-way 1-bit multiplexer and a 1-of-2 decoder. The two memories correspond to the two blocks (designated block *a* and block *b)* in every cache set. They share a common address bus A_0 - A_n and data input line I.

Suppose this one bit-slice has been selected to process bit *k* of

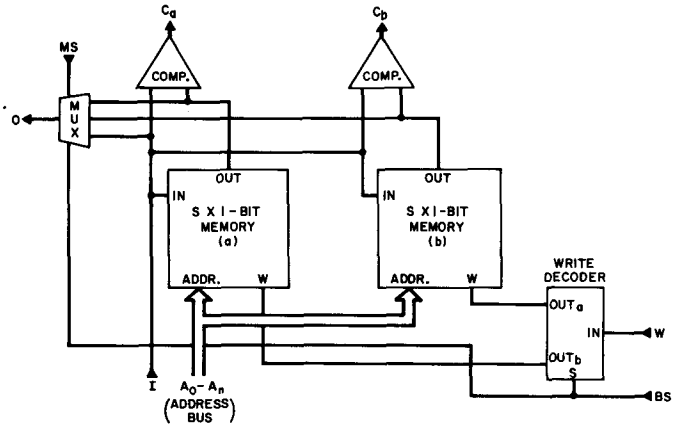

Figure 4 1-bit Tag Address Processor

the sector address field. The memory address bus A_0 - A_n is connected externally to the set address field of the memory address register (MAR). The input line I is connected to bit *k* of the tag address field of the MAR. At the beginning of a cache memory cycle, the specified set address is used to access the tag address of the two blocks currently occupying that set in cache. The processor shown in Figure 4 compares bit *k* of these tag addresses with the corresponding bit of the required tag address (as contained in the MAR). The comparator outputs C_a and C_b are used by the controller chip to determine (1) if the required word is in cache and (2) if so, then its block address with the set. Complete tag address comparison is achieved by 'wire-ANDing' comparator outputs external to the chips.

Once the controller chip has processed the tag comparator outputs, it sends out a block select signal (BS). In the case of a *hit*, this represents the cache block address of the required information. In the case of a *miss*, it represents the address of the cache block selected by the replacement algorithm to be overwritten. In the address processor of Figure 4, this BS line serves two purposes. Firstly, it controls (via the write decoder) tag memory update during miss cycles. Secondly, it drives the main memory tag address multiplexer whose output is externally connected to the input of bit *k* of the main memory sector address latches. The multiplexer select line (MS) is used to switch between old and new tag addresses during writeback *miss* cycles. It enables the main memory sector address latches to be loaded from either tag memory (during writeback) or from the MAR (when bringing a new block into cache).

The one-bit address processor can also be used to handle *modified bits*. In this case, the two memory modules are used to hold the *modified bits* of block *a* and block *b* respectively. Once again, the two memories share a common address bus A_0 - A_n and input line I. As with the tag address processor, A_0 - A_n are externally connected to the set address field of the MAR. In this case, however, the input line I is connected to the WRITE line of the memory system and the output O feeds status information back to the controller chip. Comparator outputs C_a and C_b and the multiplexer select line MS are not used.

This same structure may also be used for processing MRU bits. Only one memory module is needed to store the S MRU bits (one for each cache set). In this case the input I is connected to the block-select signal from the controller chip. During a *hit* cycle, the write-enable line W is strobed by the controller - thereby updating the appropriate MRU bit. During a *miss* cycle, the controller chip reads the MRU bit via output O to determine block replacement. Once again, C_a, C_b and MS are not used. Since only one memory module is required, BS is set to zero.

Having defined a suitable bit-slice structure, it is now necessary to determine how many bits should be handled by each I.C. The size of the bit-slice is limited by packing density and pin count. Referring to Figure 4, a number of external connections (viz. A_0-A_n, MS, C_a and C_b) can be commoned together. In order to be suitable for handling all types of tag information, the other signals (viz. I, O, BS and W) must be separately provided for each bit. The size of the memory modules depends on the technology used and the number of bits processed per chip. This in turn determines the maximum allowable buffer size. As a compromise between cache size, chip count and package size, a 2-bit slice was selected as suitable for integration. Such a structure needs 19 pins (plus supply rails) and will accommodate a cache size of up to 2K words.

Data Processor

Figure 5 shows a 1-bit processor capable of handling memory data information. It consists of two 2-way multiplexers M1 and M2, a 3-way multiplexer M3 and a level sensitive latch which forms one-bit of the memory contents register (MCR).

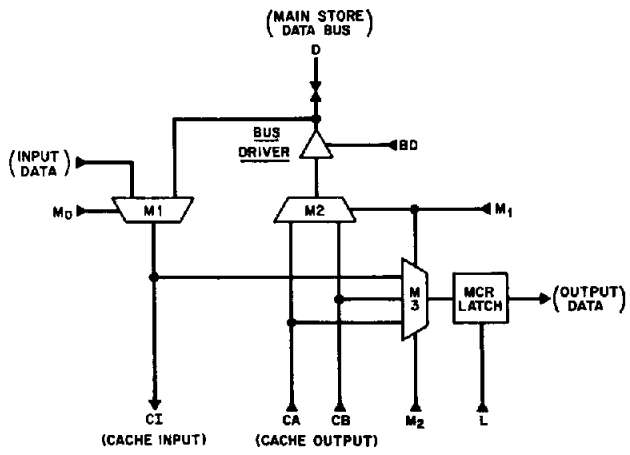

Figure 5 1-bit Data Processor

Multiplexer M1 functions as the input multiplexer. Its output CI is externally connected to the cache input bus. Input I is connected to the input port of the memory system and D forms one bit of the bi-directional main memory/cache data bus. Multiplexer M2 is the main store input multiplexer and is used during write-back. Its inputs CA and CB are connected externally to the outputs of cache pages *a* and *b* respectively. M3 is the output multiplexer driving the MCR latch. It derives its inputs from either the cache output bus CA or CB (during a *hit* cycle) or from the output of M1 (during a *miss*).

The multiplexer select lines M_0, M_1 and M_2 are externally connected to the controller chip. These are used to control data flow within the processor. The bus-direction line (BD) controls data transmission on the main memory/cache bus. The latch-enable line L is used to strobe the data latch.

Packing density is not important in determining the slice-size of this I.C. due to the relative simplicity of the data processor. Rather, it is limited simply by pin count. Once again, a 2-bit slice was chosen as the best compromise between chip count and package size. Such a chip requires 17 external connections (plus supply rails).

Controller

The controller chip coordinates the operation of the cache memory system by interfacing with the CPU and controlling the various bit-slice processors. It needs to be able to accommodate varying cache size, main memory size, cache speed, main memory speed and block size. To achieve maximum speed, the controller is designed around the use of dedicated combinational and sequential logic. Accordingly, it is not possible to represent its structure in block diagram format. Table I lists the various I/O lines required of this chip and briefly describes their function. Note that the lines have been grouped according to which part of the memory system they interface.

Some form of external timing is necessary to match controller

TABLE I CENTRAL CONTROLLER - I/O LINES

1. C.P.U. INTERFACE

REQ	C.P.U. strobes REQUEST to initiate a memory cycle
RDY	Indicates that controller is READY to receive another REQ.
W	Write-enable input
A_0,A_1	Two least-significant memory address bits

2. ADDRESS PROCESSOR CONTROL

Da	Reads tag comparison for block 'a'
Db	Reads tag comparison for block 'b'
BS	Block select output
MB	Reads 'modified-bit'
MRV	Reads 'MRU-bit'
AS	Drives multiplexer select line (MS) on address processor
WT	Write tag word pulse output
WMB	Write modified bit pulse output

3. DATA PROCESSOR CONTROL

IS	Input multiplexer select-drives M_0
BS	Block select-drives M_1
OS	Output multiplexer select-drives M_2

4. CACHE BUFFER CONTROL

$WC_{a,b}$	Write cache pulse output for blocks 'a' and 'b'
$CA_{0,1}$	Two least-significant cache address bits

5. MAIN STORE CONTROL

L_0-L_3	Latch enable strobe for address and data input latches
W_0-W_3	Write main store pulse outputs

speed to cache access time. During a *hit* cycle, operation is asynchronous in order to minimize control delays. A simple 'handshake' interface (comprising a cache select (CSL) and cache complete (CC) line) is provided to allow the designer to program cache access time. During a *miss* cycle, operation is fully synchronous to minimize circuit complexity and provide 'glitch-free' timing signals. The speed of the main memory sequence is controlled by an external clock input (CK). A second 'handshake' is provided to allow the designer to externally program main memory access time. This comprises a main store select (MSL) and main store complete (MSC) line.

Two program input lines P_0,P_1 are provided to enable the designer to select block size. In all, 35 I/O pins are required. Assuming a suitable technology, therefore, the cache control section can be comfortably accommodated in a single 40-in package.

Component Technology

The choice of a suitable component technology is dominated by the need for very high speed operation in conjunction with relatively high packing density. As the speed of bipolar devices

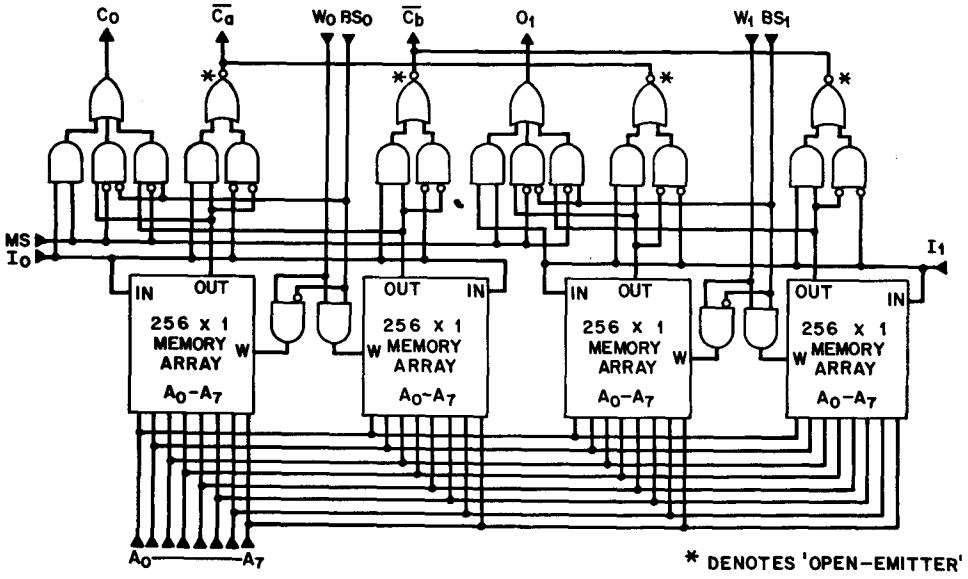

Figure 6 2-bit Address Slice Device

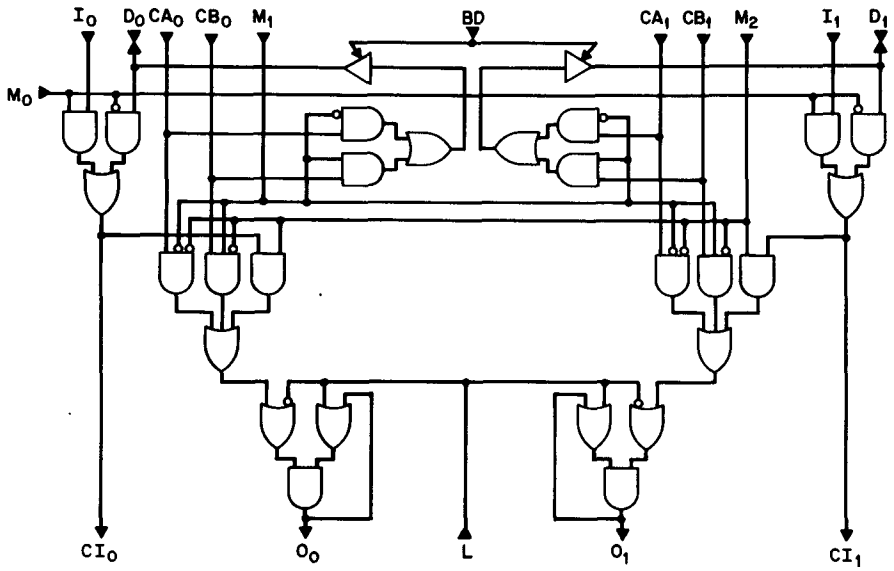

Figure 7 2-bit Data Slice Device

increases, so cache control delay times become a limiting factor in cache performance. In order to efficiently manage buffer access times of less than 50 ns., the controller needs to be able to make cache residency decisions in less than 35 ns. This in turn requires the use of very high speed logic with propogation delays of less than 5 ns. per gate.

The controller chip is a complex collection of random logic comprising some 200 to 300 individual gates. Assuming a maximum power dissipation of 500 mW per chip, this requires a technology which consumes less than 2.5 mW per gate. The address bit-slice processor chip needs 10 to 20 gates in conjunction with IK bits of high speed RAM. The data bit-slice processor is a low-density chip requiring less than 50 gates per package.

Following a review of currently available technologies, a mix of emitter-coupled logic (ECL) and emitter function logic (EFL) was chosen as suitable for hardware implementation. The address bit-slice processor includes a relatively large amount of memory (1K bits) and only a few logic gates. By using ECL in the construction of this chip, tag access times of 15 ns can be achieved. Allowing for a 5 ns comparator delay and 5 ns controller processing delay, such a device would be able to accommodate cache access times as short as 25 ns.

To match this speed, the controller must also be fabricated using a high-speed technology. ECL is not suitable as the large number of logic gates would lead to excessive power dissipation. EFL is an ECL-compatible, high-speed technology suitable for large scale integration. At a gate power level of 2.5 mW, typical gate propagation delay is 1.5 ns, making it a suitable medium for both the controller chip and the data slice processor chip.

Circuit Design

Having selected fabrication technologies, component design at the 'gate-level' is relatively straightforward for the address and data slice chips. Figure 6 shows a 2-bit address slice circuit incorporating 1024 bits of memory and 24 logic gates. Figure 7 shows a 2-bit data slice chip constructed from 28 logic gates. For completeness, the controller chip is described in Figure 8. It comprises some 200 gates of random logic assuming standard EFL implementation. Allowing 1.5 ns per gate, maximum clock frequency is approximately 80 MHz. This is equivalent to a minimum cache cycle time of 25 ns. - consistent with the speed of the asynchronous address processor.

Conclusions

An integrated cache controller, capable of accommodating a wide variety of cache design parameters, has been formulated using bit-slice architecture in conjunction with conventional ECL/EFL circuit technology. Performance specifications of the chip set are summarized in Table II. As an example of chip set implementation, Figure 9 shows the circuit diagram of an 8K by 8-bit cache controller. It features a 256 word cache, set-associative mapping, LRU block replacement, CMW writeback and a block size of four words. Four address-slice devices are used. Three of these operate on the six sector address bits of the system. The fourth slice handles 'modified' and MRU bits. Total chip count of the integrated controller is 11 devices compared to the 50 to 60 MSI and SSI packages that would normally be required to implement such a system. Circuit complexity and interconnection are reduced whilst performance and reliability are increased.

TABLE II PERFORMANCE SPECIFICATIONS

MAPPING ALGORITHM:	Set Associative
SET SIZE:	2 blocks
BLOCK SIZE:	1, 2 or 4 words
CACHE SIZE:	1 - 2048 words
CACHE SPEED:	25 ns. - ∞
MAIN MEMORY SPEED:	any
WRITE-BACK STRATEGY:	Conflicting-Modified
WORD WIDTH	any

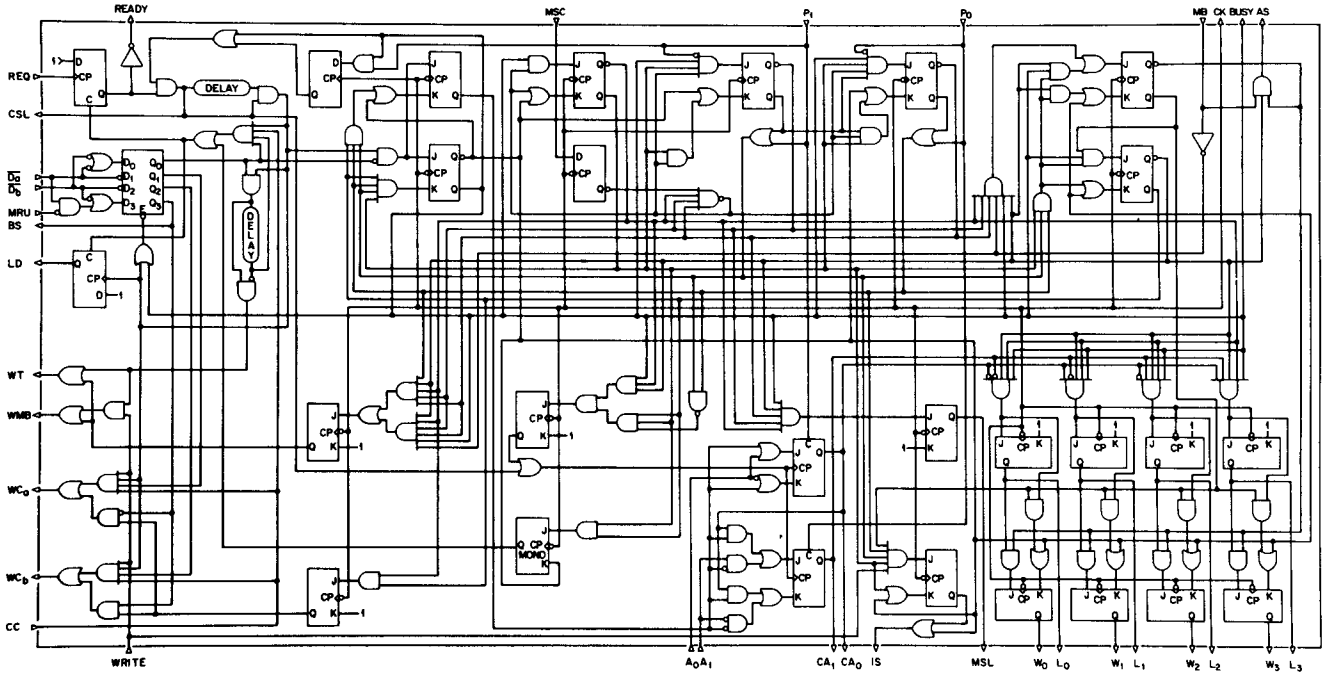

Figure 8 Central Control Device

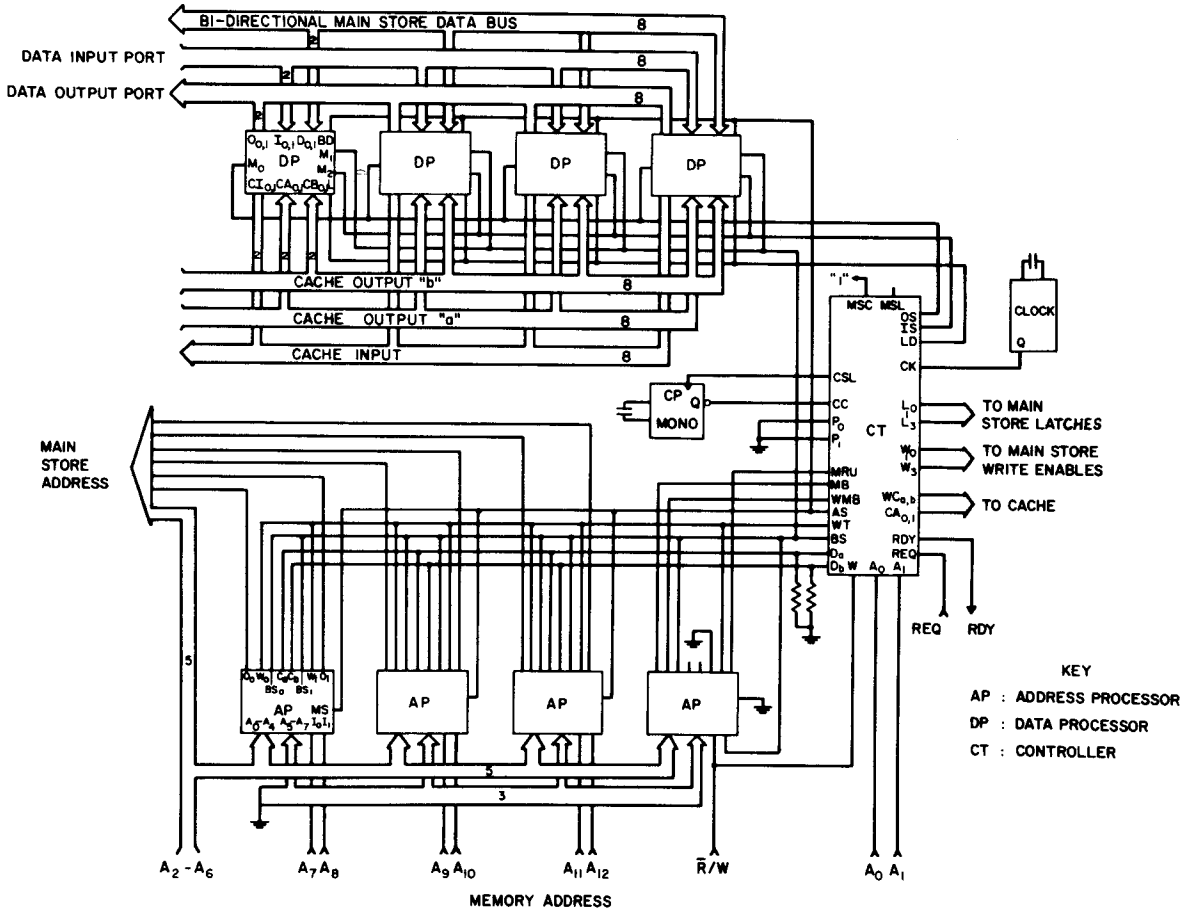

Figure 9 Sample Cache Controller

Design of the chip set has not been taken beyond gate-level circuit diagrams since manufacturing facilities were not available to the author at the time of design. It is the authors belief, however, that the devices described in this paper represent a viable integrated chip set capable of significantly reducing controller design problems in small high-speed cache memory systems.

References

[1] B. D. Ackland and D. A. Pucknell, "Studies of Cache Store Behavior in a Real Time Minicomputer Environment." Electronics Letters, Vol. 11, No. 24, November 1975, pp. 588-590.

[2] B. D. Ackland, "Cache Store Concepts in Small High-Speed Computers." Ph.D. Thesis, Department of Electrical Engineering, University of Adelaide, June 1978.

[3] J. Bell, D. Casasent and C. G. Bell, "An Investigation of Alternative Cache Organizations." I.E.E.E. Trans. on Computers, C-23, 4, April 1974, pp. 346-351.

[4] W. D. Strecker, "Cache Memories for PDP-11 Family Computers." Proc. of the 3rd Annual Symposium on Computer Architecture, 1976, pp. 155-158.

[5] C. J. Conti, "Concepts for Buffer Storage." Computer Group News, 2, 3, March 1969, pp. 9-13.