The M•CORE™ M340 Unified Cache Architecture

Afzal Malik, Bill Moyer, Dan Cermak

M•CORE Technology Center, Motorola, Inc.
P.O. Box 6000, MD TX77-F51, Austin, TX 78762-6000
{malik,billm,dcermak}@lakewood.sps.mot.com

Abstract

The MCORE M340 architecture was designed to target the low-power, embedded application market. Building upon the MCORE M3 core, the M340 provides enhancements through the addition of an 8K, 4-way set-associative unified (instruction/data) cache and an on-chip Memory Management Unit (MMU) that contains a single unified 64-entry TLB capable of mapping multiple page sizes. To achieve the power and performance requirements that today's portable electronics demand, the M340 provides programmable features that allow the architecture to be optimized for a given application. This paper discusses the features of the M340 cache sub-system and illustrates the power and performance improvements that can be achieved through proper configuration.

1 Introduction

The M•CORE M340 processor (Figure 1) contains an 8-Kbyte, 4-way set-associative, unified L1 cache with 16-byte line size, a M3 processor core, and a Memory Management Unit (MMU).

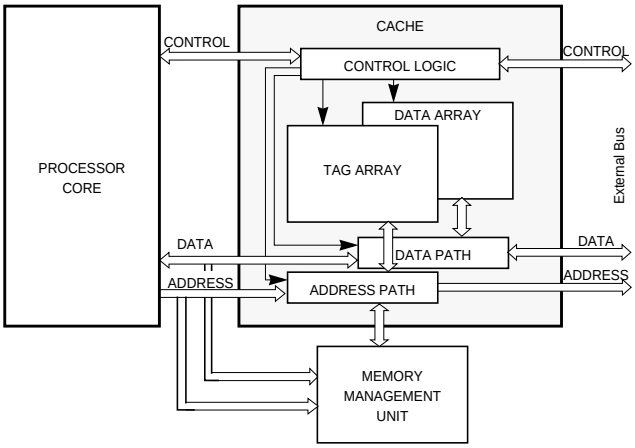


Figure 1: M340 Block Diagram

The addition of a cache memory sub-system is a well-known performance (and power) enhancement technique for microprocessors. However, the M340 cache sub-system supports programmable modes of operation to accommodate the varying embedded application environments. These modes are controlled via a cache control register which allows certain features of the cache to be enabled/disabled for power and performance tuning. The main contribution of this paper is the feature description and power consumption/performance evaluation of an actual implementation of an L1 cache for a 1M transistor commercial low-power CPU.

2 Performance and Power Evaluations

The Powerstone benchmark suite [1] was used to evaluate performance and power savings. It contains a collection of embedded and portable applications, including paging, automobile control, signal processing, imaging and fax applications. These benchmarks were run on the structural gate-level verilog model and the actual silicon (where applicable) of the M340 processor. The simulations were performed with the cache in different modes and the external burst memory latency of 5-1-1-1 clock cycles.

Table 1: Powerstone Benchmark Suite

Benchmarks	Instr Accesses	Description
qurt	35273	Square Root calculation using floating point
whetstone	48741	Test for compiler optimization
crc	19896	Cyclic redundancy check
bcnt	965	Bit shifting & anding through 1K array
auto	266446	Automobile control applications
blit	14198	Graphics applications
compress	102501	A UNIX utility
des	99340	Data Encryption Standard
engine	263946	Engine control application
fir_int	115154	Integer FIR filter
g3fax	824345	Group three fax decode (single level image decompression)
jpeg	2430577	JPEG 24-bit image decompression standard
pocsag	37112	POCSAG paging communication protocols

The Proceedings of the 2000 IEEE International Conference on Computer Design:
VLSI in Computers & Processors
0-7695-0801-4/00 $10.00 @ 2000 IEEE

Table 1: Powerstone Benchmark Suite

Benchmarks	Instr Accesses	Description
ucbqsort	221583	U.C.B. Quick Sort
v42	2272271	Modem encoding/decoding

3 Cache Write Modes

The M340 cache sub-system allows write accesses to be marked as writethrough by the MMU or writethrough/ copyback via a write mode bit in the Cache Control Register (CACR). Based on the application and the system configuration, the write mode can have a dramatic impact on power and performance based on external bus usage. Figure 2 illustrates the differences in external bus usage for writethrough versus copyback modes.

External bus traffic was used as a metric to determine the address and data bus usage for the different write modes. In a multi-processor system in which the main memory is shared by more than one CPU, it is desirable to minimize external bus utilization for higher system performance and overall reduction of system power.

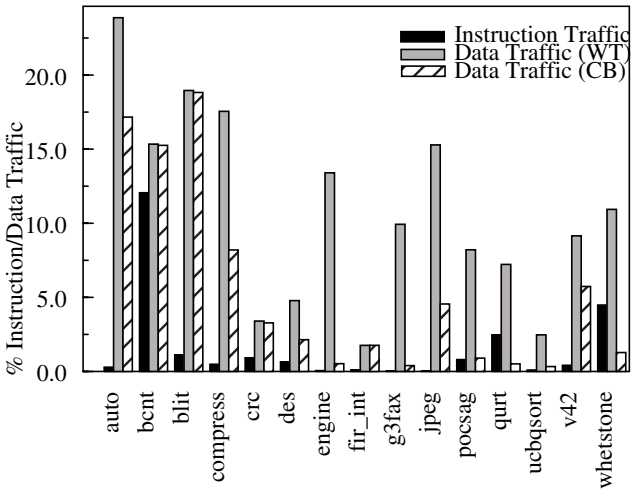


Figure 2: External Bus Traffic Graph

As shown in Figure 2, the data traffic for a given application occupies a majority of the overall bus activity. Furthermore, for writethrough operations, the amount of data transfers exceed that of copyback as expected due to the write merging capability of the copyback mode.

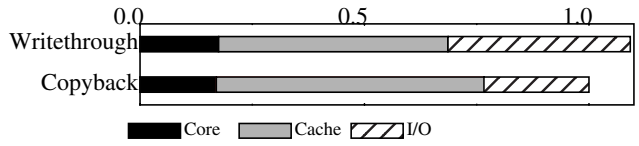


Figure 3: Write Mode Power Chart

The power distribution for the M340 as shown in Figure 3 illustrates the effect of both writethrough and copyback modes on system power. It should be noted that the power chart was based on a typical application that contained a similar number of copyback cycles as writethrough cycles in order to obtain a fair power comparison. For this type of application, copyback poses a larger power consumption requirement on the cache due to the supporting hardware with minimal external I/O access. Therefore, the copyback scenario yields higher cache power dissipation, but lower overall system power for this reason. If an application generates a poor copyback hit rate, the total system level power can be dramatically increased as well as the amount of external bus traffic due to the miss activity.

4 Way Locking

Way locking refers to the ability to control accesses to individual ways of the cache for power and performance tuning. The M340 cache incorporates enable bits that can be controlled via the cache control register. There are 8 bits total, 4 for instruction way enabling (WIE bits) and 4 for data way enabling (WDE bits). Since the M340 cache is a unified data/instruction cache, each way can be enabled or disabled for instruction and/or data access.

The locking configuration in the cache is typically static for a given application. However, with the appropriate benchmarking analysis, the ways can be configured to provide either the most optimal performance for a particular application or a reduction in power consumption with some degradation in performance. To study the effect that the locking policy has on performance and power for a given set of applications, we have accumulated some results using the Powerstone Benchmarks. Each benchmark was run with the following three configurations: All Ways Enabled for data and instruction accesses (nominal mode designated by the normalizing bar on the graphs), One Way Enabled for data and instruction accesses (effectively a

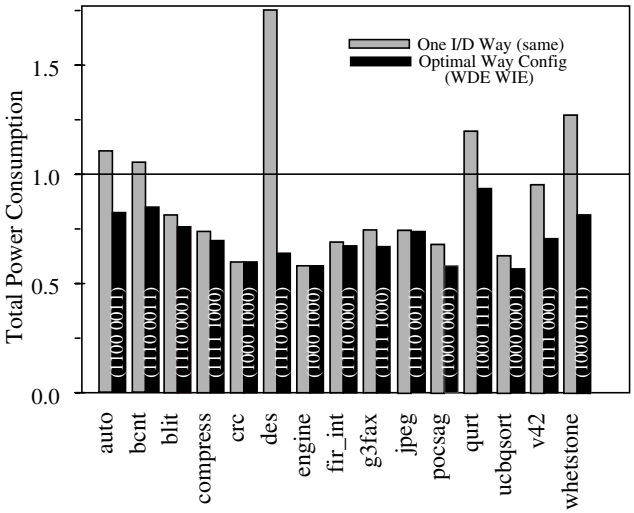


Figure 4: Way Management Power Graph

The Proceedings of the 2000 IEEE International Conference on Computer Design:
VLSI in Computers & Processors
0-7695-0801-4/00 $10.00 @ 2000 IEEE

one-way direct-mapped cache) and “Optimal” Way configuration for the given benchmark (way configuration that yielded the highest performance or lowest power consumption depending on the metric).

As shown in Figure 4, smaller programs will consume less total power since fewer ways can be enabled without suffering the penalty of conflict misses. Fewer ways enabled equates to lower access power in the arrays. Consequently, larger programs benefit from having more ways enabled to avoid numerous conflict misses that result in more high-power external memory accesses.

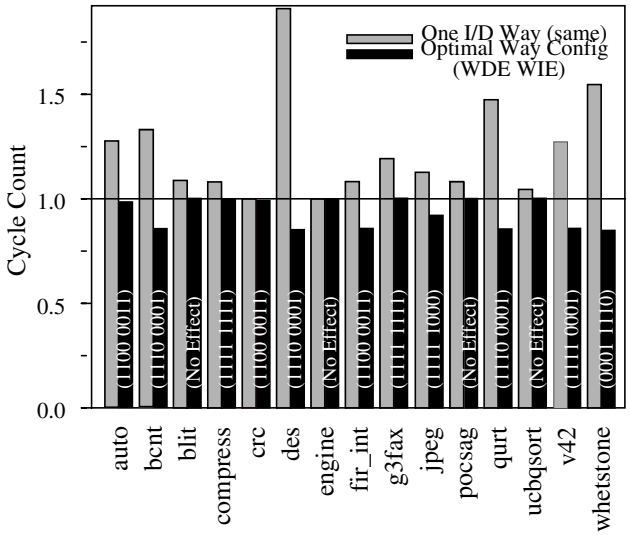


Figure 5: Way Management Performance

Figure 5 shows the performance versus way configuration. The effect of conflict misses is evident in most of the benchmark cases. Fewer ways equate to more conflict misses which result in higher latencies due to increased external memory accesses. The Optimal Way selection, however, showed significant improvement over the All Ways Enabled case for the larger benchmarks. Again, the way organization will vary depending on the code structure. The most improvement is exhibited when the way configuration allows the maximum hit rate for instruction and data accesses. Applications can achieve even greater performance improvements through compiler optimization that utilizes way manipulation or in systems where external memory latencies[1] (such as multi-level memory hierarchies) are extremely large.

5 Store Buffer and Push Buffers

5.1 Description

M340 is equipped with an eight word (32 bytes) deep store buffer and a four word (16 bytes) deep push buffer.

The store buffer contains a FIFO that can defer pending write misses or writes marked as writethrough in order to maximize performance. When enabled, store operations which miss (writethrough or copyback) in the cache or which are marked as writethrough are placed in the store buffer, and the core access is terminated. For systems that require copyback operations or systems that can trade-off performance for power efficiency, the store buffer and corresponding control logic can be gated off by disabling the store buffer bit in the CACR.

The push buffer temporarily stores push data from the cache to be written to external memory to allow the critical data to be immediately retrieved without suffering the external memory write latency. Similar to the store buffer, the push buffer and corresponding control logic can be turned off via the push buffer bit in the CACR. This feature helps to save power consumption in systems that require writethrough only transactions or those systems that can trade-off performance for power savings.

5.2 Evaluating the Store and Push Buffers

The size of the store buffer was set in order to provide maximum decoupling efficiency for the core from the external memory latency. As illustrated in Figure 6, a typical application will only utilize 1 to 3 entries in the buffer. There are numerous factors that can influence the store buffer utilization, for example, external memory latencies, number of sequential or near sequential write requests from the core, etc. Since most of the benchmark application write requests were non-sequential, the buffer was able to write the previous request out to memory as or before the next request was being issued. Some applications, such as the auto benchmark, do require larger buffer sizes due to

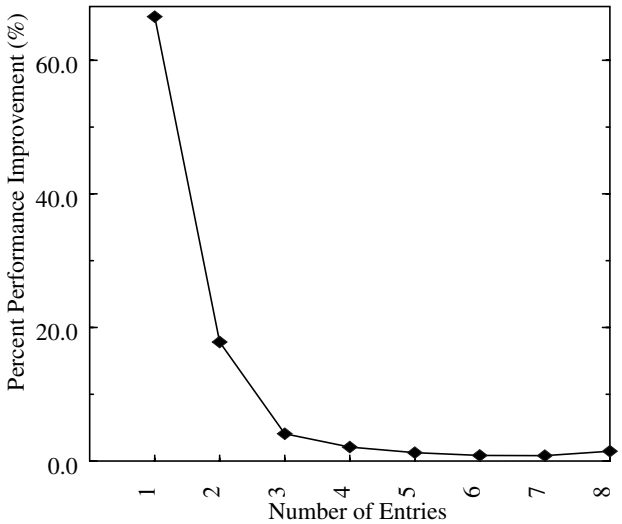


Figure 6: Store Buffer Utilization

1. All performance simulations were run with a 5 cycle and 5-1-1-1 burst cycle memory access configuration.

the multiple sequential write requests that exist. Applications that execute numerous store multiple commands or those with long external memory latencies benefit most from the large depth of the buffer.

The push buffer provides a boost in performance for applications that suffer a large number of conflict misses (v42, jpeg, and compress) on lines that have been marked dirty. This buffer allows the push latency penalty to be transparent to the core. As shown in Figure 7, the push

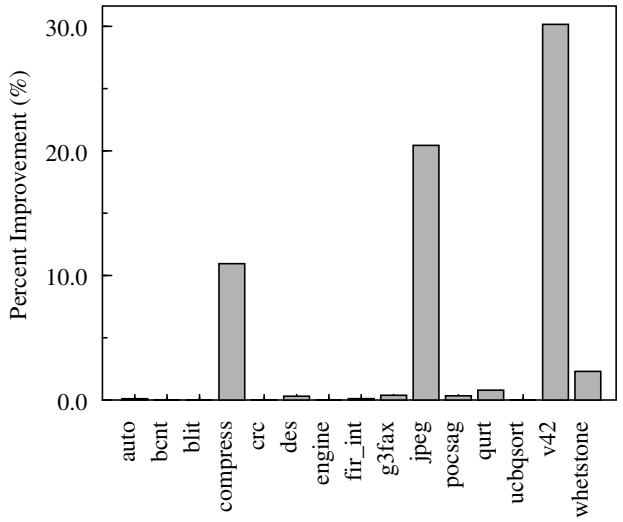


Figure 7: Push Buffer Performance

buffer can provide a significant performance improvement for the appropriate application. The benchmarks that illustrated little to no improvement from the push buffer support can utilize the enable bit to help conserve power.

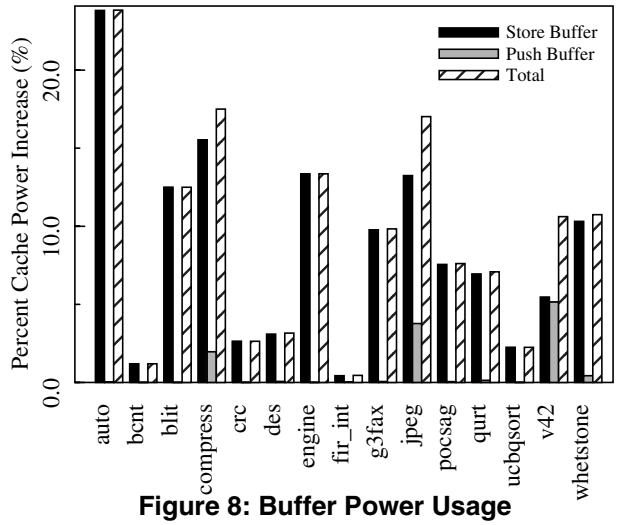


Figure 8: Buffer Power Usage

To show the effect of the buffers on power dissipation, the total cache system power (P_{cache}) can be broken down into five major modules:

$$P_{cache} = P_{sb} + P_{pb} + P_{ctrl} + P_{darray} + P_{tarray} \quad (1)$$

P_{sb}: Power dissipated by the store buffer

P_{pb}: Power dissipated by the push buffer

P_{ctrl}: Power dissipated by the control logic

P_{darray}: Power dissipated by the data array

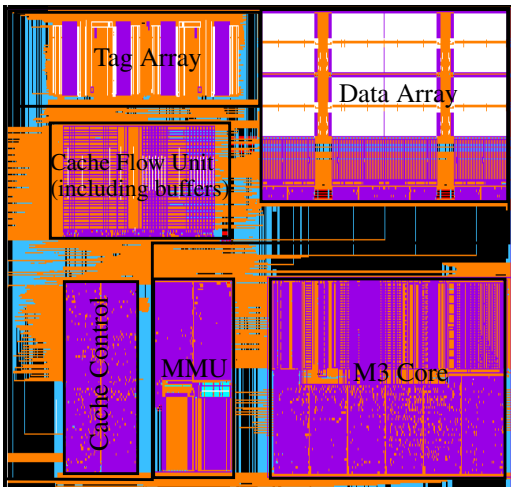
P_{tarray}: Power dissipated by the tag array

Figure 8 illustrates the total increase in power dissipation incurred by enabling the store and push buffers. The store buffer and push buffer can be disabled by clearing the respective enable bit in the CACR. Due to efficient clock gating techniques[1], P_{sb} and P_{pb} are negligible when the buffers are disabled, resulting in lower overall system power dissipation.

6 Conclusion

The M•CORE M340 provides user programmability to allow the architecture to be adaptable to various applications. Two write modes (writethrough and copyback) are supported with store and push buffer enhancement options and way management control for instruction versus data caching policy adjustment and data preservation.

Each of the enhancements described in this paper show performance and/or power consumption improvements depending on the configuration. Further optimization can be achieved through appropriate benchmarking analysis and compiler direction based on the embedded application environment.



7 References

[1] J. Scott, L. Lee, J. Arends, B. Moyer, "Designing the Low-Power M•CORE Architecture," Proc. Int'l. Symp. on Computer Architecture Power Driven Microarchitecture Workshop, Barcelona, Spain, July 1998, pp. 145-150.

[2] M•CORE M340 Reference Manual, Motorola, Inc., 2000.,

M•CORE is a trademark of Motorola, Inc.