A Low Power Unified Cache Architecture Providing Power and Performance Flexibility

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

Advances in technology have allowed portable electronic devices to become smaller and more complex, placing stringent power and performance requirements on the device's components. The M•CORE M3 architecture was developed specifically for these embedded applications. To address the growing need for longer battery life and higher performance, an 8-Kbyte, 4-way set-associative, unified (instruction and data) cache with programmable features was added to the M3 core. These features allow the architecture to be optimized based on the application's requirements. In this paper, we focus on the features of the M340 cache sub-system and illustrate the effect on power and performance through benchmark analysis and actual silicon measurements.

1. INTRODUCTION

The M•CORE M340 processor 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). The M340 cache sub-system supports programmable modes of operation to accommodate the varying embedded application environments in order to improve overall cache efficiency. These modes are controlled via a Cache Control Register(CACR) which allow certain features of the cache to be enabled/disabled for power and performance tuning.

To illustrate the effectiveness of the different features of the cache, we have collected data from benchmark simulations and have analyzed the various power and performance improvements. The main contribution of this paper is an evaluation of power consumption and performance on an actual implementation of a 1M transistor commercial low-power CPU.

The following sections describe the enhancements and evaluate their effect on performance and power consumption.

2. PERFORMANCE AND POWER EVALUATIONS

The Powerstone benchmark suite [2] 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
ucbqsort	221583	U.C.B. Quick Sort
v42	2272271	Modem encoding/decoding

3. CACHE WRITE MODES

The M340 cache sub-system supports two write modes: copyback and writethrough. The M340 cache supports both methods for integration flexibility based on the given application.

The write mode for each cache access is determined via the writethrough (WT) bit in the MMU or the Cache Write Mode (CWM) bit in the Cache Control Register (CACR). If WT is set, that access is forced to be a writethrough request. This allows regions of memory to be marked as writethrough without altering the state of the CACR. If the WT bit is clear, the state of the CWM bit determines the write mode of operation: copyback if set and writethrough if clear. It should be noted that all writes to the cache are handled with a no-write-allocate policy.

3.1 Evaluating Cache Write Modes

Address bus utilization was used as a metric to determine the portion of time the processor is utilizing the external bus. 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.

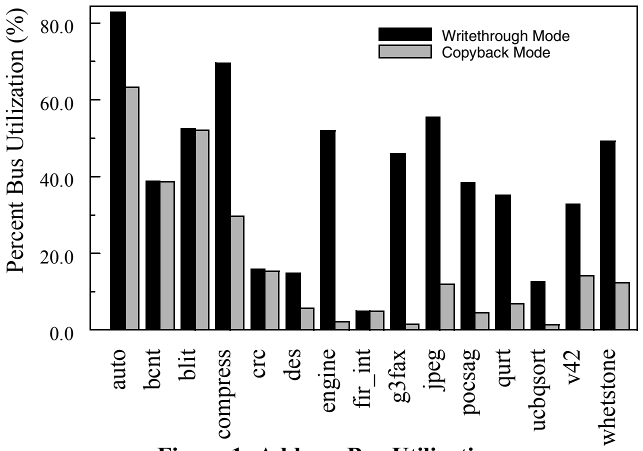

Figure 1: Address Bus Utilization

As shown in Figure 1, the writethrough mode suffers more external memory accesses than copyback. The copyback mode allows multiple writes to the same block to be merged before being written to external memory. Based on the configuration of the other cache features, copyback mode can illustrate tremendous performance improvement.

Due to the larger percentages of external bus traffic, writethrough mode will also yield higher power consumption since most external memories reside off-chip and suffer I/O pad and interconnect loadings. Figure 2 shows the effect of both writethrough and copyback modes on system power. Even though

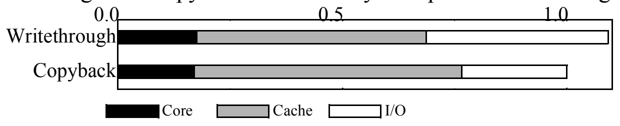

Figure 2: Write Mode Power Chart

the copyback mode causes the cache to consume a larger portion of the overall power, the total power consumption is less for copyback.

Since the M340 does not provide hardware support for monitoring cache coherency in multi-master environments (i.e. cache snooping), writethrough mode would provide more efficient system coherency management. The copyback mode would require flushing of the dirty entries in the cache potentially causing more external bus activity depending on the flushing frequency and cache configuration.

4. WAY MANAGEMENT

Way management 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 CACR. 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.

When cleared, these bits allow locking at the way level as opposed to other techniques such as line locking [3] or half-cache locking [1]. Way level locking reduces the control logic area and complexity and still allows way control flexibility.

4.1 Evaluating Way Management

To study the effect that the locking policy has on performance and power for a given set of applications, each benchmark was run for each way combination[1]. Figure 3 depicts the following three configurations: All Ways Enabled for data and instruction accesses, One Way Enabled for data and instruction accesses (effectively a one-way direct-mapped cache) and “Optimal” Way configuration for the given benchmark (Optimal Way corresponds to the configuration that yielded the least power consumption from the system perspective). Two graphs are shown to illustrate these results. Each graph has been normalized to the “All Ways Enabled” case.

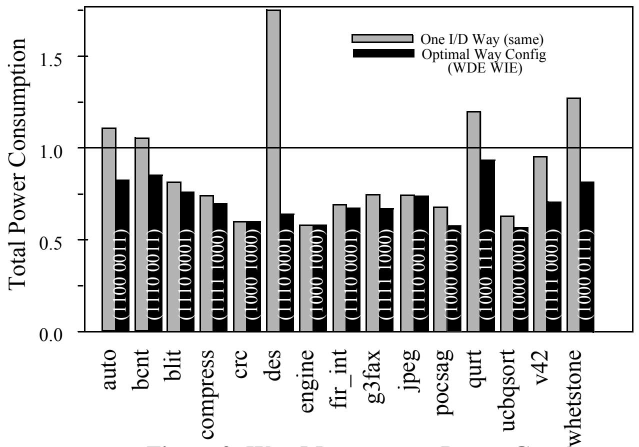

Figure 3: Way Management Power Graph

As shown in Figure 3, 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.

The advantage of way management in the M340 is that the

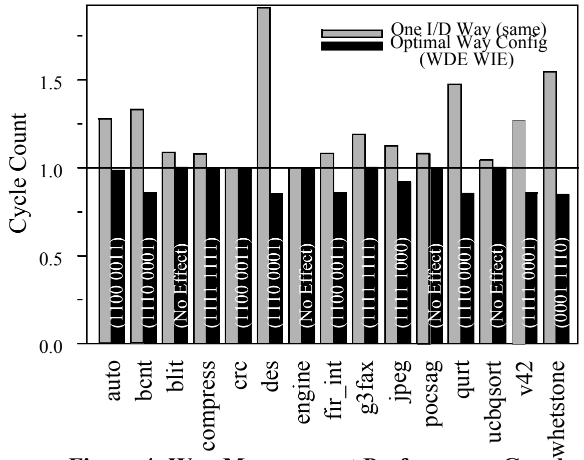

Figure 4: Way Management Performance Graph

[1] Cache control register was configured with all features disabled excluding way configuration.

optimal way configuration can be chosen for a given application. There are numerous factors that are used to determine the optimal configuration for a given application. However, there is one that seems to provide the biggest impact on power and performance: instruction to data conflict miss rate. Code segments that have predictable instruction and data access patterns provide the best opportunity for way optimization.

Figure 4 shows the performance versus way configuration. Fewer enabled ways equate to more conflict misses which result in higher latencies due to increased external memory accesses. The Optimal Way selection (the configuration that provided the fewest number of execution cycles), 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. Applications can achieve even greater performance improvements through compiler optimization that utilizes way manipulation.

5. Store Buffer and Push Buffers

M340 is equipped with an eight word (32 bytes) deep store buffer and a four word (16 bytes) deep push buffer. Store and push buffers increase the overall system performance by reducing the latency for requests made by the core.

5.1 Store Buffer

The store buffer contains a FIFO that can defer pending write misses or writes marked as writethrough. 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. This allows the core to be decoupled from the effect of the external memory latency.

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.

5.2 Push Buffer

The push buffer reduces latency for requested data on a cache miss by temporarily holding displaced dirty data while the new data is fetched from the memory. Once the line-fill from external memory completes (i.e. all four words of the cache line are written into the cache), the cache controller can generate the appropriate line-write bus transaction to write the contents of the push buffer into memory.

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.

5.3 Evaluating the Store and Push Buffers

The impact of store buffer and push buffer on performance and power is shown in Figure 5. The external memory latency used in the buffer performance evaluation was 5 for the writes from the store buffer and 5-1-1-1 (burst write) from the push buffer.

The increase in performance for the store buffer enabled case illustrated in Figure 5 is attributed to the external memory write access latencies seen by the core. This performance improvement is most evident in applications that exhibit a large number of sequential write accesses such as the auto benchmark.

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

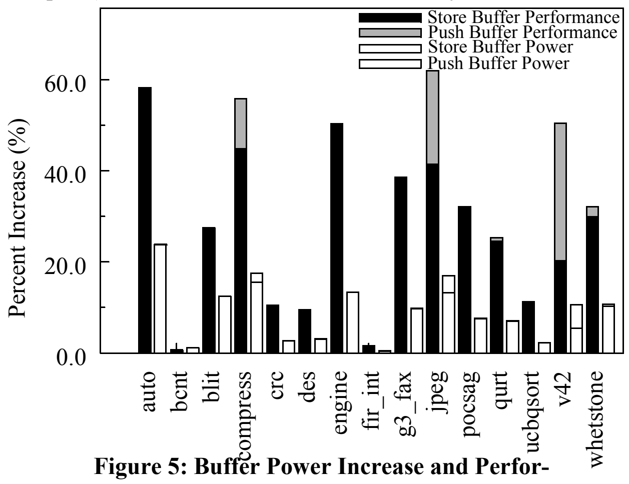

Figure 5: Buffer Power Increase and Performance Improvement

the push latency penalty to be transparent to the core. As shown in Figure 5, the push 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 5 also 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[2], the power contributions from the buffers and corresponding control logic are negligible when the buffers are disabled.

6. CONCLUSION

Many solutions have been proposed that increase performance *or* power efficiency. The M•CORE M340 provides user programmability for power *and* performance tuning. 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 for data preservation. Each of the enhancements show performance and/or power consumption improvements depending on the configuration. The programmable nature of the M340 cache promotes user programming flexibility and easier design trade-off manipulation. The programmable features of the cache discussed in this paper can be configured for optimal performance/power consumption based on the application through appropriate benchmark analysis.

7. REFERENCES

[1] J. Circello et al., "The Superscalar Architecture of the MC 68060*". IEEE Micro*, Vol. 15, No. 2, April 1995, pp. 10-21.

[2] 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.

[3] K. Suzuki, T. Arai, N. Kouhei, and I. Kuroda, "V830R/AV: Embedded Multimedia Superscalar RISC Processor*". IEEE Micro*, Vol. 18, No. 2, April 1998, pp. 36-47.

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

M•CORE is a trademark of Motorola, Inc.