

A Programmable Unified Cache Support for GPRS Encryption System

Advances in wireless technology have allowed portable electronic devices to become smaller and more complex, placing stringent power, performance and security requirements on the device's components. To address the growing need for longer battery life, higher performance and enhanced security for 3G applications, an 8Kbyte, 4-way set-associative, unified (instruction and data) cache with programmable features was added to the M-CORE M3 core. These features include write mode selection, way management, and buffer enabling/disabling which allow the architecture to be optimized based on the application's requirements. In this paper, we present the features of the unified cache architecture and illustrate the usefulness effect on power and performance for a GPRS encryption module (GEM) through benchmark analysis and actual silicon measurements.

INTRODUCTION

General Packet Radio Service (GPRS) allows packet switched communications by sending packets on more than one time slot to the same user allowing data rates as high as 150 kbps. GPRS allows multiple users to share one physical channel and enables high performance wireless internet access. GPRS enables wireless e-commerce which requires enhanced security for the users. The GPRS encryption module is a hardware accelerator used assist in the encryption and decryption of GPRS data packets. The GEM can be used to generate and verify the Frame Check Sequence (FCS) contained in the GPRS data packets. Our GEM implementation includes both GEA-1 and GEA-2 as specified by the ETSI [10].

The M-CORE M341 processor with GEM (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). The on-chip L1 cache improves system performance by providing low-latency data to the instruction and data pipelines of the core, hence, decoupling the processor and the GEM from the external memory latency. The GPRS data to be manipulated is placed in any memory (RAM, eDRAM etc) in the system. Typically the data is placed in the on-chip memory to avoid off chip latencies. Control and status information is passed back and forth from the GEM and the CPU via a set of memory mapped tasks. A single interrupt from the GEM indicates completion of tasks.

The addition of a cache memory sub-system is a well-known performance (and power) enhancement technique for microprocessors. However, the M341 cache

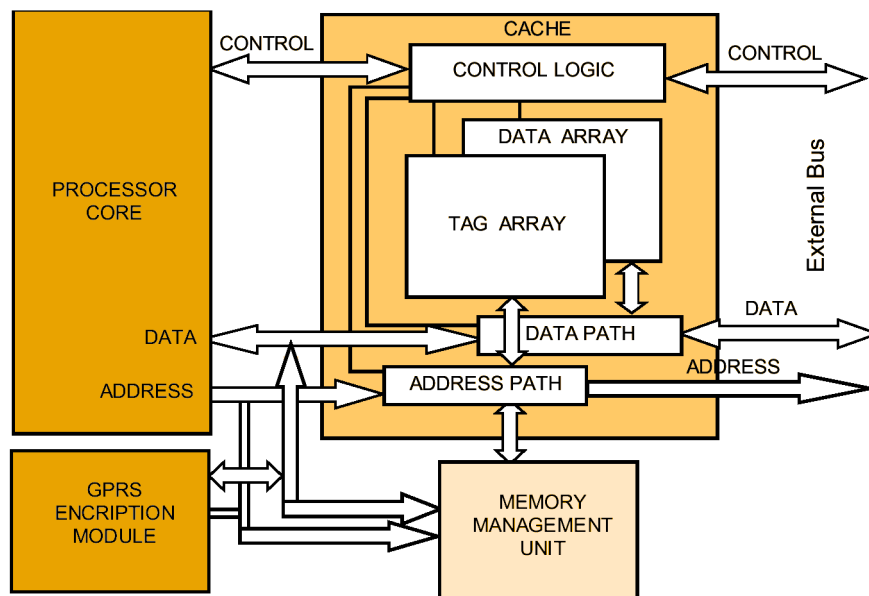


Figure 1. Processor Block Diagram.

sub-system supports programmable modes of operation to accommodate the performance requirement of a 3G GPRS encryption system. These modes are controlled via a cache control register which allow certain features of the cache to be enabled/disabled for power and performance tuning. The M-CORE M341 processor allows writethrough and copyback (writeback) modes for write operations with optional buffer support, streaming mode for sequential instruction/data forwarding to the CPU and the GEM, and way management for controlling individual ways of the cache.

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⁽¹⁾ 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.

FUNCTIONAL DESCRIPTION

Frame Check Sequence (FCS) and data ciphering are the two computationally intensive tasks accelerated by our GPRS encryption module.

The Powerstone benchmark suite [8] was used to evaluate performance and power saving using a unified cache architecture. It contains a collection of embedded and portable applications, including paging, automobile control, signal processing, imaging and fax applications. The simulations were performed with the cache configured with all ways enabled for instruction/data, writethrough mode, and buffers and streaming disabled (this configuration was changed only when simulating a particular feature) and the external burst memory latency of 5-1-1-1 clock cycles. Table 1 lists the applications in the benchmark suite, the number of instruction accesses for each application, and a

short description of the application's function

CACHE WRITE MODES

The M341 cache sub-system supports two write modes: copyback and writethrough. There have been various discussions of both methods and their effectiveness in a system[1,2,3]. The M341 cache supports both methods to provide the option of choosing which mode is best suited for the given application.

WRITETHROUGH MODE

Write accesses can be marked as writethrough by the MMU or a write mode bit in the Cache Control Register (CACR). In either case, writethrough writes are always passed on to the external bus. This mode of operation is desirable in systems such as those with shared memory (coherency concerns) or external memory drives [4].

The obvious downfall with writethrough writes is the processor stall issue due to external memory latency. To avoid this latency issue, the M341 cache utilizes a FIFO store buffer (Section 6.1) that can be enabled via the CACR.

Writes in writethrough mode are handled with a no-write-allocate policy-i.e. writes that miss in the cache are written to the external bus, but do not cause the corresponding line in the memory to be loaded into the cache. Other write allocation policies were considered such as those discussed in [1] and [4], but the no-write allocate policy was chosen in order to minimize control complexity and to avoid processor stalls due to write allocations. Write accesses that hit always write through to memory and update the corresponding cache line.

COPYBACK MODE

Similarly, write accesses can be marked as copyback by the MMU or the write mode bit in the CACR. Copyback writes set the dirty⁽²⁾ bit and update the

Benchmarks	Instr. Accesses	Description
qurt	35273	Square Root calculation using floating point
whets tone	48741	Test for compiler optimization
crc	19896	Cyclic redundancy check
bcnt	965	Bit shifting & anding through 1 K 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

Table 1.

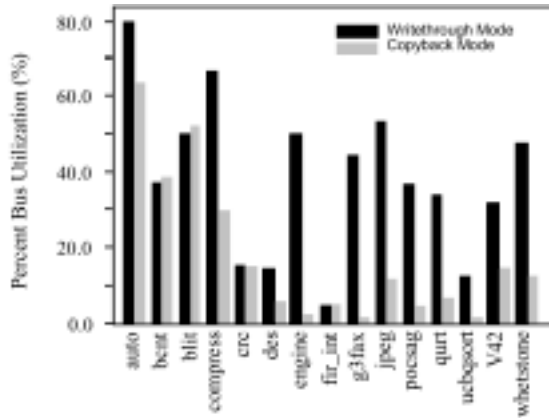


Figure 2. Address Bus Utilization.

cache line on a hit without performing an external bus access. Writes that miss in the cache are handled with a no-write-allocate policy. Copyback regions are typically used for local data structures or stacks to minimize external bus use and reduce write-access latency. The dirty cache data is later written to memory when the line is replaced.

One penalty of copyback is the hardware overhead due to the dirty bit and push logic. With writethrough, this additional logic is unnecessary, thus, reducing the total chip area. Another more critical shortcoming of copyback is the stall(s) incurred by the processor while waiting for the replaced line to be "written back" to external memory. To alleviate these stalls, the M341 cache utilizes a one cache line entry push (copyback) buffer that can be enabled through the CACR (Section 6.2). Once the replaced line is read into the cache, the push buffer contents are written to memory.

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 for higher system performance and overall reduction of system power.

As shown in Figure 2, 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. It should also be noted that some applications yielded very little utilization difference between the two modes (i.e. bcnt, blit,

crc). This arises when an application contains very few, if any, write hits. When this occurs, copyback mode looks more like a writethrough mode and the utilization remains the same.

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 3 shows the effect of both writethrough and copyback modes on system power. Even though the copyback mode causes the cache to consume a larger portion of the overall power, the total power consumption is less for copyback.

However, since the M341 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. To avoid the writethrough bus traffic and power consumption penalties, the M341 MMU allows particular pages to be marked as writethrough to allow copyback operation for references to the remainder of pages.

WAY MANAGEMENT

Way management refers to the ability to control accesses to individual ways of the cache for power and performance tuning. The M341 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 M341 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 [5] or half-cache locking [6]. Way level locking reduces the control logic area and complexity and still allows way control flexibility.

If an instruction or data enable bit is set in the CACR, that particular way is enabled for a lookup as well as a line replacement by an instruction or data access miss, respectively. When clear, the way is disabled for lookups which prevents the row clocks, sense amps, and hit generation logic for the given way from toggling. Locking the way also prevents line fills into the disabled way allowing data to be preserved. For data coherency, the Way Access Mode (WAM) bit in the CACR can be used to override these enable bits for cache lookup accesses only. If this bit is set, all ways are searched for the requested access independent of

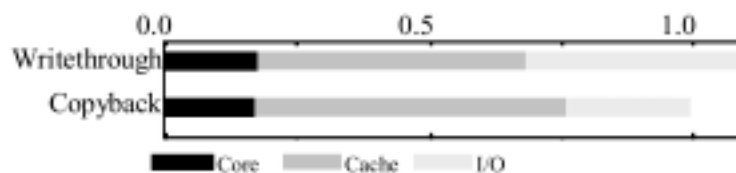


Figure 3. Write Mode Power Chart Description.

VARIOUS

a cleared enable bit. Upon a line fill, this bit has no effect.

The configuration of these locking bits depends on the application environment. The locking bits are configurable to allow a programmer to fine tune the cache performance based on the application requirements. For instance, the cache size can be reduced by locking ways from instruction and data accesses for lower performance targets. The locking bits can also be used to configure the cache to provide cacheable and SRAM functionality (see Cache Management section regarding accessing the cache using direct memory-mapped load/stores) or to be adjusted based on the instruction to data access ratio and conflict miss(3) percentage for performance (and power) optimization.

EVALUATING WAY MANAGEMENT

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 one-way direct-mapped cache) and "Optimal" Way configuration (denoted in parentheses) that yielded the highest performance or lowest power consumption depending on the metric). Two graphs are shown to illustrate these results. Each graph has been normalized to the "All Ways Enabled" case.

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.

The advantage of way management in the M341 is that the 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. For instance, if an instruction set exhibits spatial locality but very little temporal locality (i.e. sequential, non-iterative code) and the data set has the inverse behavior, enabling more data ways than instruction ways would reduce the instruction-data conflict miss percentage resulting in better 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

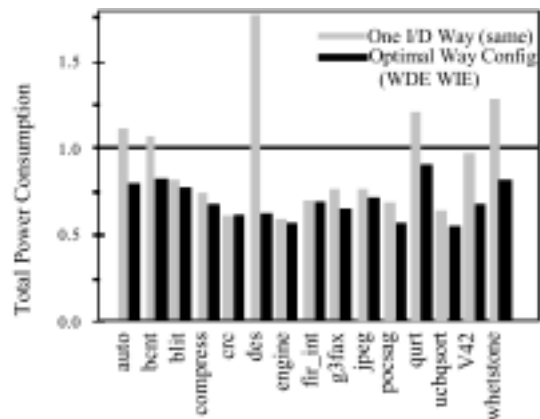


Figure 4. Way Management Power Graph.

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(4) that utilizes way manipulation or in systems where external memory latencies(5) (such as multi-level memory hierarchies) are extremely large.

STREAMING

The M341 cache supports streaming of data from the external bus to the core as soon as data is available from the main memory. This feature can be enabled/disabled by a control bit in the cache control register. If a request from the core results in a cache miss, a burst read operation is performed by the cache to fill the corresponding line. The requested word is first supplied to the core and the cache continues with the line fill. If the next access requested by the core is sequential and same as the next word of the line fill then the word is streamed to the core directly from the external memory bus.

Figure 6 shows the number of words streamed for data and instruction accesses relative to the total number of line fill word transfers. The sequential nature of instructions result in more stream hits (except for the ucbsort

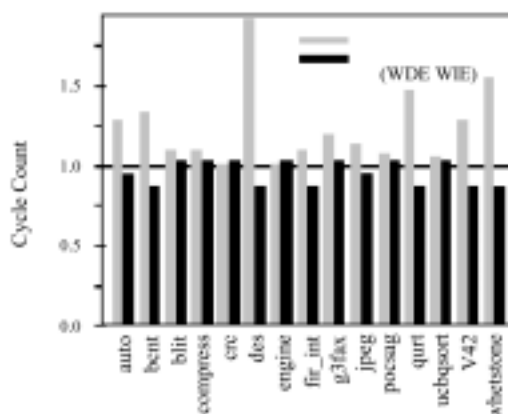


Figure 5. Way Management Performance Graph.

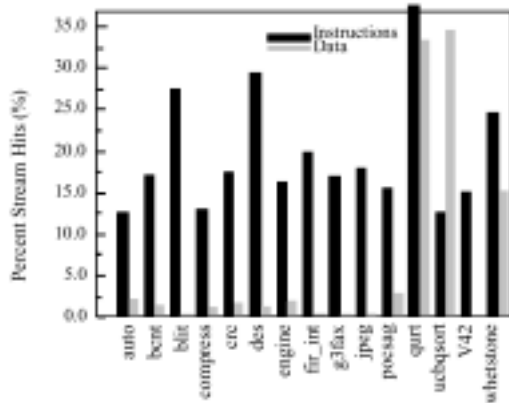


Figure 6. Instruction and Data Streaming.

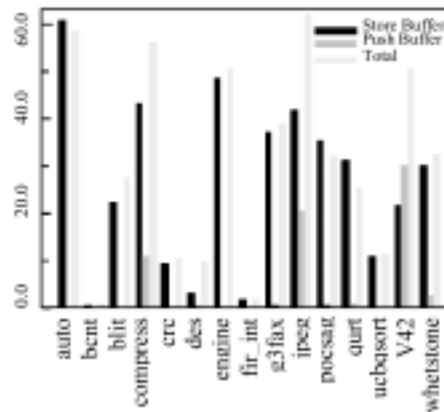


Figure 7. Buffer Performance.

benchmark which has more sequential data accesses). Streaming allows the word being retrieved from external memory to be sent directly to the core, thus reducing latency and also saving a number of lookup cycles in the cache. Depending upon the nature of software code, streaming can result in significant power savings and performance improvement.

STORE BUFFER AND PUSH BUFFERS

M341 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.

STORE 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. This allows the core to be decoupled from the effect of the external memory latency. For coherency, cache-inhibited stores, line-fills, and CACR accesses cause the processor termination to be withheld until the store buffer has been flushed of all entries.

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.

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. If a cache miss displaces a dirty line and the push buffer is enabled, the requested access is immediately placed on the external bus. In parallel to the critical word request, the data line to be replaced is stored into the push buffer. 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.

EVALUATING THE STORE AND PUSH BUFFERS

Once the push buffer or store buffer has valid data, the internal bus controller uses the next available external bus cycle to generate the appropriate write request. In the event that a cache line-fill is required while valid data still resides in the buffers, the pipeline will stall until the buffers are emptied before generating the required external bus transaction for the critical word.

The impact of store buffer and push buffer on performance and power is shown in Figure 7 and Figure 8, respectively (6). 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 7 is attributed to the write access latencies seen by the core. When the buffer is enabled, the core sees a single cycle access latency to the store buffer (if store buffer is not full) as opposed to a 5 cycle access latency to external memory. 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. When a dirty line needs to be replaced, the dirty data is written to the push buffer (when enabled) and the critical word requested from the core is sent immediately to the external memory bus. This allows the push latency penalty to be transparent to the core. As shown in Figure 7, 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.

VARIOUS

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:

```
Pcache = Psb + Ppb + Pctrl + Pdarray + Ptarray
Psb: Power dissipated by the store buffer
Ppb: Power dissipated by the push buffer
Pctrl: Power dissipated by the control logic
Pdarray: Power dissipated by the data array
Ptarray: 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[8], P_{sb} and P_{pb} are negligible when the buffers are disabled, resulting in lower overall system power dissipation.

CACHE MANAGEMENT

The cache control register (CACR) is used to enable and configure the cache. This register is memory mapped for easy load/store access. A hardware reset clears the enable bit, disabling the cache, and removing all configuration information. An additional address region is dedicated to allow particular commands to be executed on a per line basis in the cache. For instance, each line can be flushed with or without invalidation or invalidated without further modifications in a single store operation. These operations are provided to assist context switching, debugging, and software coherency.

The M341 cache arrays are also memory mapped for debugging ease and on-chip SRAM accessibility (data array section only). The data array section is a 8K contiguous memory space. These memory-mapped sections are accessed through single-cycle load/store instructions. For a particular application, the cache can be partitioned into cache/SRAM sections through the use of way management (Section 4). This feature is useful in applications that utilize only a few of the ways (such as the engine benchmark) and need a small, fast section of on-chip memory (i.e. local stacks).

CONCLUSION

A unified cache architecture was discussed that provides user programmability for power/performance tuning. Two write modes (writethrough and copyback) are supported with store and push buffer enhancement options, way management control for instruction versus data caching policy adjustment and for data preservation, and a line fill streaming option for optimal sequential access performance. Each of the enhancements show performance and/or power consumption improvements depending on the configuration. The performance requirement of GPRS systems are growing exponentially. The programmable nature of the M341 cache promotes user programming flexibility and easier design trade-off integration.

Although most of the features of this cache architecture are not new concepts, the paper describes and analyzes an actual programmable implementation of these features in a commercial low-power CPU. The programmable nature of this architecture allows a degree of freedom for the programmer in order to allow

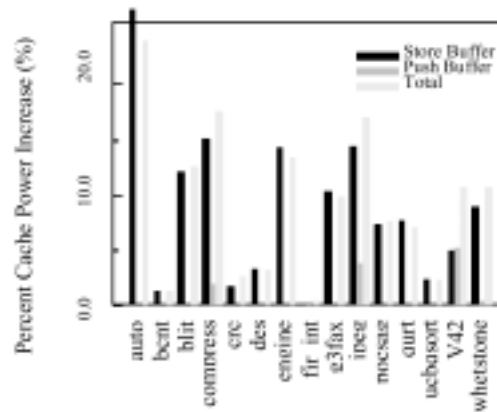


Figure 8. Buffer Power Usage.

further application optimization that could not be realized with a fixed architecture ■

David Ruimy Gonzales is a Senior Member of Technical Staff at the Motorola Embedded Platform Solutions group. He has worked on microcontrollers, DSPs, and micro-RISC processors for 22 years at Motorola. He holds a BSCS from St. Edwards University in Austin, Texas, has published over 50 technical articles, and holds 2 patents in the area of real-time embedded systems development.



REFERENCES

- N. P. Jouppi, "Cache Write Policies and Performance," Computer Architecture News vol.21, no.2, 1993, p.191-201.
- Smith, Alan J. "Characterizing the Storage Process and Its Effect on the Update of Main Memory by Write-Through". Journal of the ACM 26, 1 (January 1979), 6-27.
- Agarwal, Anant. Analysis of Cache Performance for Operating Systems and Multiprogramming. Ph.D. Th., Stanford University, 1987.
- D. W. Miller and D. T. Harper III, "Performance Analysis of Disk Cache Write Policies". Microprocessors and Microsystems, Vol 19, No. 3, April 1995, pp. 121-130.
- 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.
- J. Circello et al., "The Superscalar Architecture of the MC68060". IEEE Micro, Vol. 15, No. 2, April 1995, pp. 10-21.
- Hennessy & Patterson, "Computer Architecture: A Quantitative Approach," Second Edition. Morgan-Kaufmann, San Francisco, CA, 1990, 1996.
- 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.

- M-CORE M341 Reference Manual, Motorola, Inc., 2000, European Telecommunications Standards Institute

FOOTNOTES

- (1) It should be noted that although area is a design concern, die cost will not be evaluated in this paper.
- (2) An entry in the cache is marked dirty when it has been modified by a valid write access. This signifies that the cache contains a more updated copy than the external memory.
- (3) Conflict Miss - If the block placement strategy is set associative or direct mapped, conflict misses will occur because a block can be discarded and later retrieved if too many blocks map to its set. These are also called collision misses or interference misses.
- (4) Compiler techniques are outside the scope of this paper.
- (5) All performance simulations were run with a 5 cycle and 5-1-1-1 burst cycle memory access configuration.
- (6) The performance degradation due to the store-push buffer enabling logic is negligible and is not considered.



Dedicated Systems Experts can assist you with:

- Audits
- Courses
- Consultancy
- Support

Contact Nico Van Wijmeersch at +32-2-520.55.77 or find more information on our Web-site at <http://www.dedicated-systems.com>