A DSP with Caches – A Study of the GSM-EFR Codec on the TI C6211

Tor Jeremiassen
Bell Laboratories – Lucent Technologies
tor@bell-labs.com

Abstract

Texas Instruments has positioned the C6211 as the low cost member of its C62xx family of DSPs. The C6211 differs from the other devices in this family in that it has a new on-chip memory architecture that uses a two-level cache hierarchy, something not typically seen on a DSP.

This paper presents results of a performance study of the TI C6211 running the GSM-EFR speech codec, an important benchmark application in digital cellular telephony. A detailed analysis of the cache performance shows that the caches, although small, are effective in maintaining good performance, even in a multi-programmed workload, where cache pollution affects the memory system performance.

1. Introduction

The Texas Instruments TMS320C6211 [9, 8, 7] is a recent addition to its C62xx family of high performance DSPs. These devices are aimed at computationally intensive multi-channel and multi-function applications, such as that of cellular base-stations.

The C6211 uses the same statically scheduled core architecture as the other C62xx devices [10]. It differs in the memory architecture. Instead of a large amount of on-chip RAM divided into separate data and instruction memories, the C6211 uses separate 4KB instruction and data caches, and four 16 KB banks of on-chip memory that can individually be configured as either local memory or a unified second level cache [7].

Caches have long been used in general purpose microprocessors, and numerous studies have detailed the performance of different cache configurations on a number of important benchmarks [4]. However, DSP designers have typically eschewed caches in favor of on-chip memories. Therefore, few, if any, published studies on cache performance of signal-processing applications running on DSPs exist.

The use of caches on the C6211 can be motivated in part by cost. Assuming reasonable locality of reference in an application, a relatively small amount of cache memory can approximate the performance of a much larger local memory at a significantly lower cost. The 150MHz C6211 cost only about $25, while the 200MHz C6201 (128 KB of on-chip memory), 250MHz C6202 (384 KB of on-chip memory) and 300 MHz C6203 (896 KB of on-chip memory) cost between 3 and 7 times as much [12].

This paper evaluates the C6211 in terms of its performance on the reference version of the GSM Enhanced Full Rate speech transcoder [2], hereafter referred to as the GSM-EFR codec.

The performance numbers were obtained using a cycle accurate instruction simulator for the C62xx architecture, modified to account for cache misses and stall cycles in the memory hierarchy. The results show that, in the base configuration, cache misses account for only 10% to 12% of the total cycle count, barely more than that consumed by NOPs. Despite their small size, the first level caches perform well. This is particularly true for the data cache which exhibit read-miss rates of less than 1%[1].

Introducing periodic invalidations to simulate cache pollution incurred by a multi-programmed workload naturally causes a drop in performance. This drop is minimized however, when the invalidations occur between frames. The performance of the speech encoder decreased by only 7.3% when both levels of cache were invalidated between frames and less than 0.5% when only the level 1 caches were invalidated. For the speech decoder the corresponding numbers were 32.3% and 1.3%.

The remainder of this paper is structured as follows: Section 2 describes the architecture of the C6211 and how it differs from the other C62xx devices. Section 3 explains the methodology of the experiments, Section 4 presents the results, and Section 5 concludes.

2. Architecture

Figure 1 shows the core architecture of the C62xx series of DSPs. It is divided into two (almost) symmetric clusters, A and B. Each cluster consists of four fully pipelined functional units and a 16 general purpose registers [10]. Except for reads/writes to control registers (not shown), the units on the two sides are equivalent in their functionality. The *L*

[1] Write-misses typically do not cause stalls, and the cache uses a no-allocate-on-write policy.

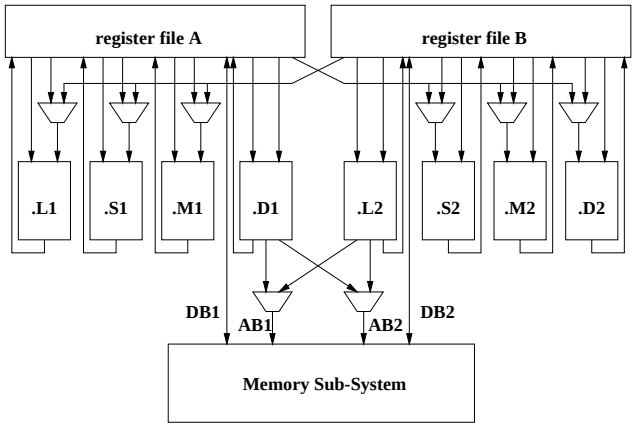


Figure 1. Block diagram of the C6211 architecture.

unit handles saturated and unsaturated basic arithmetic operations, compares, logical operations and some special signal processing operations. The *S* unit handles branches, basic arithmetic operations, shifts and logical operations. The *M* unit handles saturated and unsaturated multiplies. The *D* unit handles simple arithmetic operations and memory operations.

Each functional unit takes two sources from the register file in its cluster. Alternately, the L, M and S functional units can read one source from the opposite cluster, with the constraint that only one cross-cluster source can be supplied in each direction (A to B, B to A) per cycle.

Every instruction in the C6201 can be predicated on one of five predicate registers (B0, B1, B2, A1 and A2) being zero or non-zero, regardless of which cluster the instruction is executing on. Conditional branches are implemented by predicated unconditional branch instructions.

The differences between the C62xx devices are in the memory sub-system. The C6201 has a total of 128 KB of on-chip memory, divided equally between instructions and data. The C6202 has a total of 384 KB of on-chip memory, 256KB for instructions and 128 KB for data [5]. The C6203 has 896 KB of on-chip memory, 384 KB for instructions and 512 KB for data [13]. For the C6202 and C6203, the instruction memory is further divided into two memory mapped blocks, allowing DMA transfers to/from one while the processor operates out of the other. Alternately, 128 KB of the instruction memory (of both devices) can be used as a direct mapped instruction cache. The C6211 differs from the preceding parts in that it provides a two level cache hierarchy, instead of addressable on-chip memory.

Regardless of the memory hierarchy differences between the devices, they are all object code compatible [6]. The C62xx architecture is statically scheduled with fixed functional unit latencies. This means that the program visible memory latency, in processor cycles, remains constant across the devices. That is, every device has four cycles worth of delay slots following the cycle of a load instruction, before the loaded value is available for use. Should a memory access require more time, such as for a cache miss or an external memory reference, the entire processor is stalled.

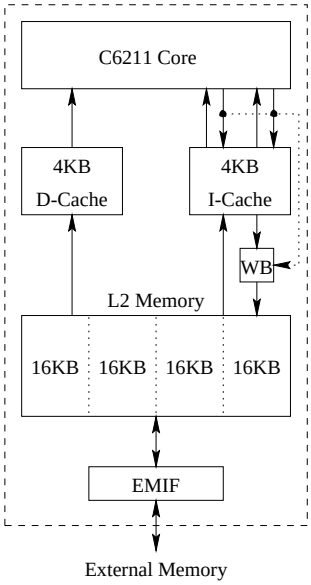


Figure 2. Diagram of the C6211 memory hierarchy.

2.1. C6211 Memory Hierarchy

The C6211 memory hierarchy is illustrated in Figure 2. It consists of a 4 KB instruction cache, a 4 KB dual-ported data cache, and 64 KB of on-chip memory that can be configured either as local memory or second level cache [7].

The 4 KB instruction cache is direct mapped, with a 64 byte block size. The instruction cache is connected to the level 2 cache by a uni-directional 256 bit wide bus. It takes two, 2-cycle transfers on the bus to perform a cache block fill. Including one additional stall cycle, a cache miss (with a level 2 cache hit) stalls the CPU for a total of 5 cycles.

The 4 KB data cache is two-way set associative, with 32 byte block size, and uses an LRU replacement policy. It supports two concurrent memory accesses per cycle. Unlike the on-chip memory in the other C62xx devices, the data cache is dual-ported, not banked, so that the processor is never stalled due to conflicting memory accesses.

The data cache is connected to the level 2 cache by a 128 wide bit bus, and similarly to the instruction cache, it takes two, 2-cycle transfers to perform a cache block fill.

The data cache uses a copy back policy, but does not do allocations on write-misses. Writes that miss in the cache are entered into a 4 entry write-buffer, which then performs the writes into the level 2 cache. Upon a read-miss, the write-buffer has to be emptied before the read can be serviced by the level 2 cache. If two read-misses occur in the same cycle, they are serialized before being serviced by the

level 2 cache. A read-miss stalls the processor for 4 cycles, provided it hits in the level 2 cache.

There is 64 KB of on-chip memory that can be configured, in increments of 16 KB, either as level 2 cache or as local memory. When the memory is configured as a level 2 cache, the cache has 128 byte block size, and an associativity equal to the number of 16 KB blocks configured as cache. That is, if only one of the on-chip memory blocks is configured as cache the level 2 cache is direct mapped; with two blocks configured as cache it is two-way set associative, and so on, up to 4-way set associative.

The level 2 memory is single-ported and configured as four 64-bit wide banks. An I-cache block fill accesses all banks; a D-cache block fill accesses only two. If there is a conflict between I and D cache accesses, the I cache is given priority. Conflicts between level 1 cache accesses and external memory interface (EMIF) accesses are determined based on the value of a bit in the cache configuration register. To simplify the simulation of the memory hierarchy, the results in this paper assume that the EMIF accesses are given priority.

Level 2 cache fills are performed starting with the requested level 1 cache block, and updates both levels of caches simultaneously. Thus, once the first 32 bytes of data has been read (64 bytes for instructions), the processor can resume execution following the cache miss stall. However, subsequent level 1 cache misses may experience bank conflicts with the EMIF accesses until the entire level 2 cache line has been filled.

3. Methodology

The results for the C6211 reported in the next section were obtained through cycle-accurate, execution driven simulation. The simulator ran on a 533 MHz Alpha 21164A-based workstation, and was capable of simulating about 2.4 million target instructions per second. Version 2.10 of the TI C62xx compiler [11] was used to produce the executable.

The GSM-EFR reference codec version 5.1.0 [1] was used for the experiments. The encoder maps blocks (frames) of 160 13-bit samples of input speech to 244 bits for transmission over a cellular channel. The decoder maps the parameters back to 160 speech samples. The encoding process is roughly an order of magnitude more computationally intensive than the decoding process. The sampling rate of the speech is 8 kHz, which makes the bandwidth of the encoded bit stream 12.2 kb/s. The GSM-EFR codec is part of the Global System for Mobile communications (GSM) standard specified by the European Telecommunications Standards Institute (ETSI).

The codec consists of roughly 12,000 lines of C code for the encoder part, and 8,000 lines for the decoder. The codec was compiled with full optimizations with some source-level optimizations applied by hand to the busiest loops. In particular, some outer loops were unrolled and the resulting collection of inner loops were fused to increase the instruction-level parallelism.

The test input for the simulation runs was a sequence obtained by concatenating the TEST10, TEST7 and TEST13 speech codec test sequences [3]. Together they form 1080 frames, or about 22 sec. of both male and female speech.

Table 1. Execution time per frame, GSM-EFR.

Architecture	Frequency	Encode	Decode
TI C6203	300 MHz	0.99 ms	0.11 ms
TI C6202	250 MHz	1.19 ms	0.13 ms
TI C6201	200 MHz	1.49 ms	0.17 ms
TI C6201	167 MHz	1.78 ms	0.20 ms
TI C6211	150 MHz	2.10 ms	0.24 ms

Table 2. CPU and memory system activity – encoder.

Cycles	Encoder	
Total	340,320,270	(100.00%)
Inst. execution	278,316,778	(81.78%)
Nops	29,452,347	(8.65%)
I-Cache Miss	18,603,805	(5.47%)
D-Cache Miss	8,296,612	(2.44%)
Write Buffer Stall	1,403,053	(0.41%)
L2-Cache Miss	4,247,671	(1.25%)

4. Results

4.1. Overall C6211 Statistics

Table 1 shows the execution time per frame for the GSM-EFR codec executing on the TI C62xx devices. Real-time performance of the codec is 20 ms per frame. Except for the C6211, the performance difference between the devices is only due to clock speed differences. The C6211 incurs an additional 6% to 10% performance penalty due to cache effects.

Tables 2 and 3 break down the total number of cycles used by the C6211 by processor activity. For both the encoder and the decoder, almost 10% of the execution time is due to NOPs and roughly 10% due to memory system overhead. Of the latter, instruction cache misses is the largest component. This is particularly true for the decoder. Write-buffer stalls account for less than 0.5% of all cycles. Using all of the 64 KB second level memory as cache results in little time spent on level 2 cache misses. Thus, for the GSM-

Table 3. CPU and memory system activity – decoder.

Cycles	Decoder	
Total	39,424,391	(100.00%)
Inst. execution	31,360,209	(79.55%)
Nops	3,563,283	(9.04%)
I-Cache Miss	4,227,555	(10.72%)
D-Cache Miss	214,996	(0.55%)
Write Buffer Stall	9,119	(0.02%)
L2-Cache Miss	49,225	(0.12%)

Table 4. Memory system statistics.

	Encoder	Decoder
Inst. fetches (256 bit)	164,465,284	15,886,306
Memory loads	186,151,617	18,816,169
Memory stores	33,625,235	4,101,730
I miss rate	2.26 %	5.32 %
I avg. lifetime	5,852 cyc.	2,982 cyc.
D miss rate	3.67 %	0.84 %
D read-miss rate	0.94 %	0.20 %
D write-miss rate	18.74 %	3.77 %
D avg. lifetime	19,389 cyc.	82,398 cyc.
L2 miss rate	0.69 %	0.14 %
L2 I miss rate	1.78 %	0.02 %
L2 D read-miss rate	0.66 %	3.11 %
L2 D write-miss rate	0.06 %	0.07 %
L2 avg. lifetime	280,123	n/a

EFR codec, the I-Cache seems to be the biggest bottleneck in the memory system.

Table 4 shows the breakdown of the C6211 cache statistics. The first part of Table 4 shows the number of references to each of the level 1 caches. There are 33% more data references than instruction references for the encoder (46% more for the decoder), in part due to the large fetch width (32 bytes, or 8 instructions) of the I-Cache. There are about 5.6 and 4.5 loads for every store in the encoder and decoder respectively.

For the encoder, the data cache miss rate is actually greater than the instruction cache miss rate, even though instruction misses account for a greater fraction of the overall execution time. This reversal, however, is easily explained by the no-allocate-on-write policy of the data cache. Since writes that miss in the cache only cause stalls when the write-buffer is full, and as Tables 2 and 3 show, those stalls account for less than 0.5% of the total running time, there is very little cost associated with write-misses. Thus, the miss rate to compare against is the data cache read-miss rate, which is much less than the instruction cache miss rate.

There is an interesting difference between the encoder and the decoder. The instruction cache miss rate is significantly lower for the encoder than for the decoder, even though the code size of the encoder (53 KB) is significantly greater than that of the decoder (22 KB). This suggests that the instructions of the encoder exhibit greater locality than does the code in the decoder. The fact that instruction cache blocks on average stayed in the instruction cache almost twice as long before being replaced for the encoder than for the decoder supports this. The reason is that the encoder contains more and deeper loop nests, resulting in a greater degree of reuse (temporal locality) of the cache blocks.

The difference in the size of the data segment between the encoder (17 KB) and the decoder (14 KB) is much smaller than the difference in text size. However, all of the data cache miss rate components are significantly larger for the encoder than for the decoder. Moreover, the average residency time of a data cache block for the decoder is roughly 4 times that of the encoder, and in fact exceeds the time to decode two frames. Thus, the decoder is actually getting reuse of cache blocks across the processing of multiple frames. This is clearly not the case for the encoder. Thus the decoder exhibits much greater locality of reference within its data segment.

The level 2 cache miss rate is very low for both halves of the codec, as reflected by the small fraction of the cycles spent on serving level 2 cache misses. However, there is a difference in the source of the cache misses that are incurred at this level. For the encoder, the source is mainly instruction misses, while the number of instruction and data misses for the decoder (in the level 2 cache) are about the same[2]. This behavior correlates well with the difference in code size. For the encoder, the combined code and data does not fit in the second level cache, which results in more conflict misses. Since there are more data than instruction references (68% vs. 32%) to the level 2 cache, one can expect that given the LRU replacement policy, more cache blocks containing instructions are replaced than those containing data. Thus, more instruction references are likely to miss.

For the decoder, the entire program fits in the second level cache so the miss rate is due to cold start misses.

4.2. C6211 Memory Allocation

The C6211 has a more complex memory system on-chip than the other C62xx devices. The 64 KB of second level memory is divided into 4 banks that can be configured either as local memory (mapped to a given range of the address space) or second level cache. To explore the perfor-

[2]The rates differ because the number of references of each type reaching the second level cache are very different.

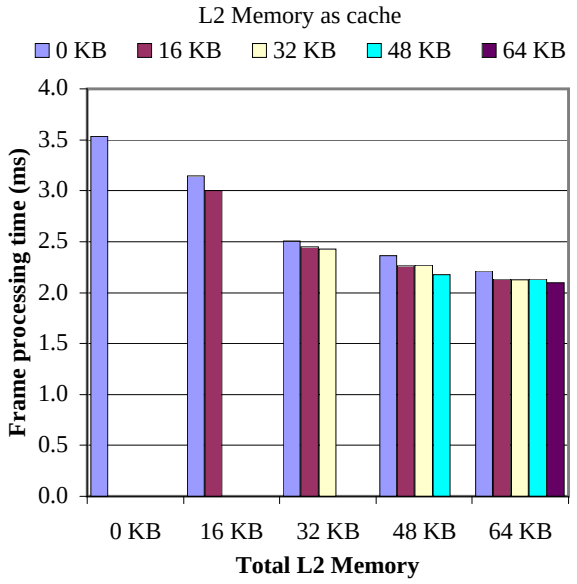


Figure 3. Average execution time per frame for GSM-EFR encoder as function of level 2 memory configuration.

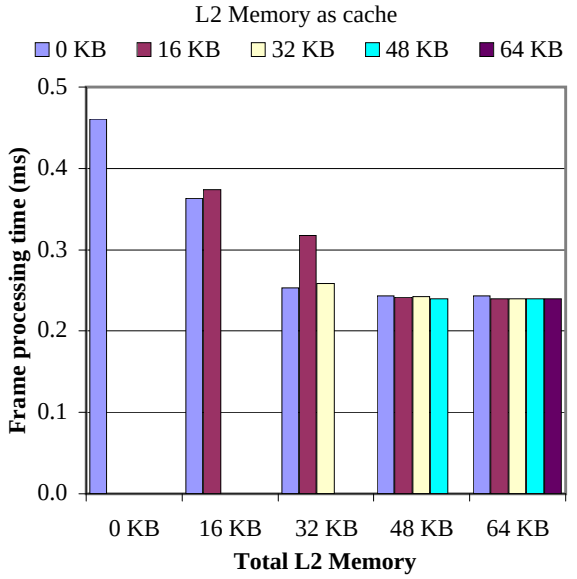


Figure 4. Average execution time per frame for GSM-EFR decoder as function of level 2 memory configuration.

mance trade-offs involved in configuring this memory as cache and/or local memory, the simulator was run with second level cache size varying from 0 KB to 64 KB. For each cache size less than 64 KB, simulations were run with varying amounts of the remaining memory configured as local memory. The performance effect of local memory was simulated by satisfying level 1 cache misses that referenced the allocated local memory with the same latency as a level 2 cache hit, but without affecting the level 2 cache operations. The most frequently referenced 16 KB code and data memory blocks (as seen by the level 2 cache) were assigned to local memory.

Figures 3 and 4 explore the performance trade-offs involved in configuring level 2 memory as local memory or cache for the GSM-EFR encoder and decoder, respectively. From left to right, each group of bars shows the performance of using a given amount of level 2 memory. Within each group, the individual bars show the performance for different allocations of this memory between cache and local memory.

The performance of the encoder (Figure 3) improves, not unexpectedly, with increasing amounts of on-chip memory available to the application. The best performance is obtained by using all of the available memory as cache; any mixed configuration of cache and local memory performs nearly as well, though.

For the decoder, the best performance is again obtained using the full amount of on-chip memory as cache. However, very little is gained by using more than 48 KB of the on-chip memory (either as cache, local memory, or a combination). Additionally, using 32 KB of the on-chip memory for local memory and leaving the remaining half unused comes within 6% of the best performance.

A notable difference between the encoder and the decoder is that when the available on-chip memory is held to 32 KB for the decoder, the mixed cache and memory configuration performs notably worse than either pure configuration. Furthermore, the all local memory configuration edges out the all-cache configuration. The reason for this has to do with the number and type of the external memory accesses. Compared to the all-cache configuration, the mixed configuration incurs roughly 4 times the number of external memory references due to level 2 cache misses. However, the all-local memory configuration incurs even more external memory references. The explanation is that the two most frequently accessed external memory blocks for the decoder contain code. Thus, in the all-local memory configuration these are assigned to local memory and references to them are no longer seen by the level 2 cache. This is important since level 2 cache misses due to instruction references stall the processor for roughly twice as long as misses due to data references, since the larger instruction cache block (64 B vs. 32 B) takes that much longer to transfer over the narrow external bus. Thus, even though there are fewer external memory references in the mixed configuration, the proportion of such references that are due to instruction cache misses is larger (34.02% vs. 0.02%). In addition, most of the external data memory references (about 80%) in the all-local memory configuration are due to data cache write-misses, which do not usually stall the processor.

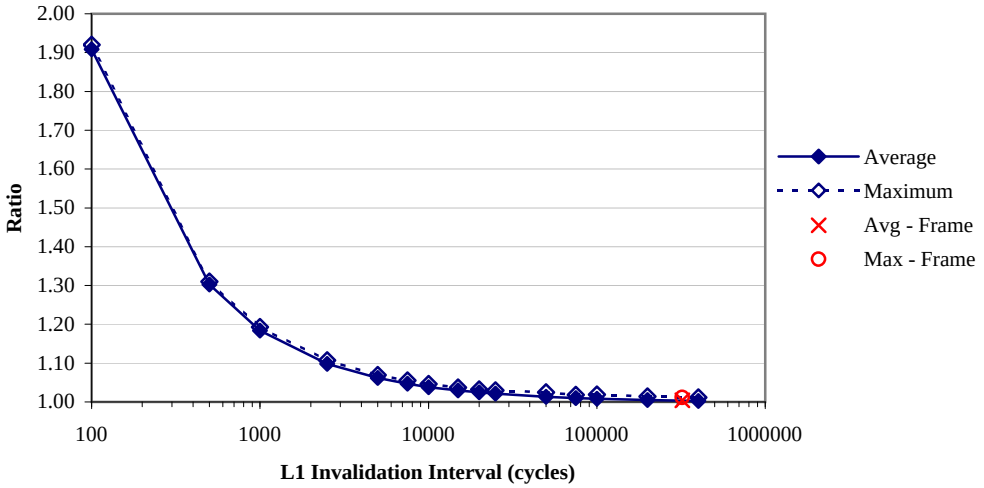


Figure 5. Effect of periodic invalidations of the level 1 caches for the GSM-EFR encoder.

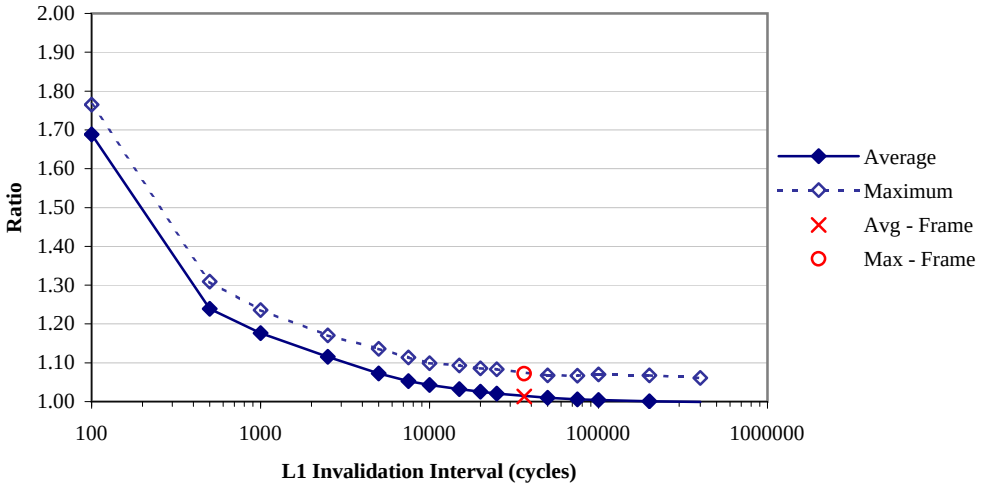


Figure 6. Effect of periodic invalidations of the level 1 caches for the GSM-EFR decoder.

4.3. Effect of Context Switches

As mentioned in the introduction, the C6x series of DSPs is aimed at computationally intensive multi-channel and multi-function applications, such those running on cellular base-stations. Such application domains require that multiple applications, or multiple instances of the same application, run on the same processor. Thus the processor's compute and memory resources, including the C6211's caches, must be shared across these applications. Since it is impossible to maintain the cache state of an application across context switches, the measured performance when running in this environment is bound to be worse than when running alone.

In order to gauge the performance impact of the periodic cache pollution effects of context-switching, the simulator was changed so that the contents of the caches (level one only, or both levels) were invalidated periodically. The invalidation interval was varied over a large range to investigate in detail the performance sensitivity to invalidations of the encoder and decoder and to provide some performance bounds.

Figures 5 and 6 show the running time when invalidating the level 1 caches relative to the no invalidation case. The invalidation interval ranged from 100 cycles to 400,000 cycles. Both the average and maximum[3] encoding (decoding) times per frame are shown. The "X" and "O" show the performance when the level 1 caches are invalidated between frames.

Except for the relative difference between average and maximum frame processing times (the cycle difference is about the same) between the encoder and the decoder, the graphs are very similar. There is a large penalty for invalidating the caches very frequently, but the performance converges to the non-invalidation baseline relatively quickly.

[3]The maximum and average times exclude the time to encode/decode the initial frame, which tend to be longer than the others due to cold start misses.

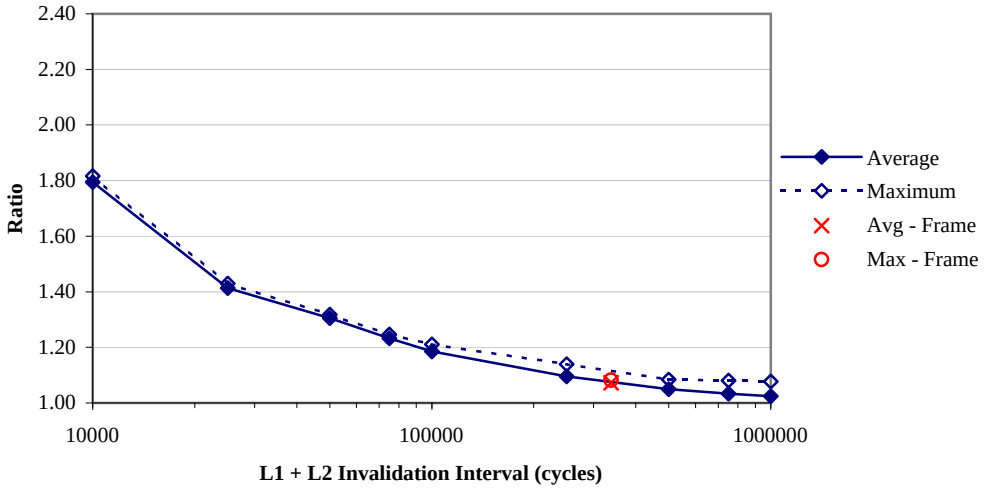


Figure 7. Effect of periodic invalidations of the cache hierarchy for the GSM-EFR encoder.

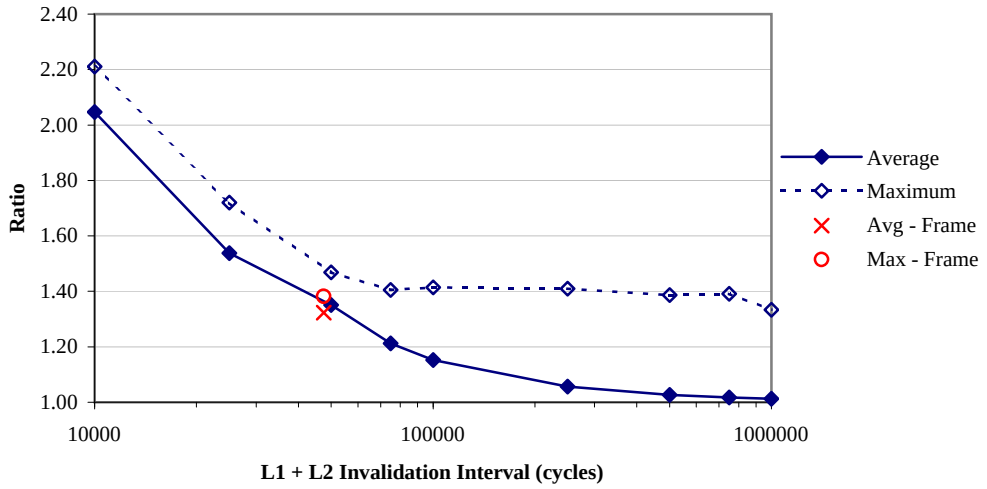


Figure 8. Effect of periodic invalidations of the cache hierarchy for the GSM-EFR decoder.

At the shortest invalidation interval of 100 cycles, the performance is within a factor of 2 of the no-invalidation case. As the invalidation interval increases, the performance penalty rapidly declines. At 1,000 cycles, the performance is only 18% worse than no invalidations at all, and the difference drops to about 4% for 10,000 cycles. When the caches are invalidated between frames, the performance is within 0.5% for the encoder, and 1.3% for the decoder, which is less than the maximum frame processing time with no invalidations. Clearly, unless the context-switch interval is unreasonably short, the cache related impact on performance is negligible when only invalidations of the level 1 caches are considered.

Figures 7 and 8 show the running time when invalidating both levels of caches relative to the no invalidation case. The invalidation interval ranged from 10,000 cycles to 1,000,000 cycles. The “X” and “O” show the performance when the caches are invalidated between frames. Note that the X-axis starts at 10,000 cycles.

It is immediately obvious that invalidating both levels of the cache hierarcy has a much larger impact than just invalidating the first level caches. The performance of invalidating the entire cache hierarchy every 10,000 cycles actually approximates that of invalidating the level 1 caches every 100 cycles. This is not entirely unexpected. The cache miss penalty for a second level cache miss is up to an order of magnitude greater than that of a level 1 cache miss, even more for back-to-back second level cache misses. As the cache invalidation interval increases the performance improves, though not as quickly as when only invalidating the level one caches. Were the caches to be invalidated between frames, the performance would be 7.3% worse for the encoder, and 32.3% worse for the decoder. The decoder’s performance suffers more because the invalidation interval is only a seventh of that of the encoder.

While the average frame processing time decreases as the invalidation interval increases for both the encoder and the decoder, this is not true for the maximum frame pro-

cessing time for the decoder. It levels out at 50,000 cycles and remains relatively constant thereafter. The reason for this behavior is that once the invalidation interval exceeds the frame processing time, the average frame processing time still decreases but the maximum times will occur for those frames that incur the cache invalidation during an early stage of the processing. The same effect is seen for the encoder as well. However, since the frame processing time exceeds 300,000 cycles, and because the relative difference between the average and maximum frame processing times is smaller, the effect is much less visible.

5. Conclusion

The TI C6211 is a DSP that differs from most others by replacing a large on-chip local memory with a smaller cache hierarchy in order to reduce cost while retaining most of the performance. In fact, the simulations show that performance is decreased by less than 10% when 128 KB of on-chip memory is replaced by a total of 72 KB of caches.

A concern with the use of caches is the lack of determinism that follows, and in particular, the effect of context switching. The results show that small first level caches primarily benefit relatively short-term spatial and temporal locality. Thus, the impact of periodically invalidating their contents is small, even at invalidation intervals as short as 10,000 cycles. Moreover, the performance approaches the no-invalidation case asymptotically as the invalidation interval lengthens.

The story is a little different for the C6211's second level cache. Due to its larger size (up to 64 KB), the average lifetime of the cache blocks is much greater than that of the level 1 caches. Coupled with the much longer level 2 cache miss penalty, the performance impact of periodic invalidations is much greater. However, if the invalidation interval is on the order of the frame processing time, the worst and average case performance are both minimized, and for longer running tasks, such as a single frame encoding, the performance impact is less than 10%.

6. Acknowledgements

I would like to thank Brian Kernighan, Nevin Heintze and Lorenz Huelsbergen for their helpful comments on earlier versions of this paper.

References

[1] European Telecommunications Standards Institute. *Digital cellular telecommunications system; ANSI-C code for the GSM Enhanced Full Rate (EFR) speech codec*, March 1997.

[2] European Telecommunications Standards Institute. *Digital cellular telecommunications system; Enhanced Full Rate (EFR) speech transcoding (GSM 06.60)*, March 1997.

[3] European Telecommunications Standards Institute. *Digital cellular telecommunications system (Phase 2+); Test sequences for the GSM Enhanced Full Rate (EFR) speech codec (GSM 06.54 version 5.1.0)*, ets 300 725 edition, December 1997.

[4] J.L. Hennessy and D.A. Patterson. *Computer Architecture A Quantitative Approach*, chapter 5. Morgan Kaufmann, second edition, 1996.

[5] Texas Instruments. *How to Begin Development Today with the TMS320C6202 DSP*, September 1998. SPRA473.

[6] Texas Instruments. *How to Begin Development Today with the TMS320C6211 DSP*, September 1998. SPRA474.

[7] Texas Instruments. *TMS320C6202/C6211 Peripherals Addendum to the TMS320C6201/C6701 Peripherals Reference Guide*, August 1998. SPRU290.

[8] Texas Instruments. *TMS320C6211 Cache Analysis*, September 1998. SPRA472.

[9] Texas Instruments. *TMS320C6211 Fixed-Point Digital Signal Processor – Product Preview*, August 1998. SPRS073.

[10] Texas Instruments. *TMS320C62x/C67x CPU and Instruction Set Reference Guide*, March 1998.

[11] Texas Instruments. *TMS320C6x Optimizing C Compiler User's Guide*, July 1998. SPRU187D.

[12] World's highest performance DSP platform delivers best of both worlds. Texas Instruments News Release, September 1998.

[13] Texas Instruments. *TMS320C6203 Fixed-Point DSP*, May 1999. SPRS086A.