Cache Justification for Digital Signal Processors

by

Michael J. Lee

December 3, 1999

Cache Justification for Digital Signal Processors

By

Michael J. Lee

Abstract

Caches are commonly used on general-purpose processors (GPPs) to improve performance by reducing the need to go to off-chip memory every time program instruction or data is needed. However, digital signal processors (DSPs) traditionally did not incorporate any caches, but instead mainly relied on fast on-chip memory banks. Although some DSPs made use of small instruction caches, none had integrated a data cache. This paper will discuss the justification for having caches on DSP processors and the performance impact of using them. It will mainly analyze Texas Instruments' TMS320C6211, which uses a two-level caching scheme for both instruction and data. The paper will conclude with my findings on cache utilization of the TMS320C6211 by evaluating several commonly used DSP kernels on a commercially available simulator.

INTRODUCTION

More and more consumer products and cost-sensitive systems are incorporating DSPs into their designs. The demand for products with signal processing capabilities is influencing the design goals of some DSPs. Traditionally, there was a fine distinction that set apart DSPs from GPPs; but now, as DSPs are getting more sophisticated, the distinction is becoming more ambiguous. One area that used to set apart DSPs from GPPs was the use of caches. GPPs routinely implemented caches into their designs, but DSPs have traditionally lacked caches, especially data caches. As processor cycle times continue to fall, more proportion of the peak processing power is being lost to the memory system. Caches offer a simple and effective way to combat the latency of accessing instructions and data [1].

MEMORY ARCHITECTURES

Typical DSP operations require high memory bandwidth because they are data-intensive. To satisfy the high throughput requirements, DSPs usually implement the Harvard architecture as opposed to the Von Neumann architecture. GPPs have traditionally used the Von Neumann memory architecture (Figure 1a), which connects one memory space to the processor core by one bus set—an address bus and a data bus. The memory bandwidth was sufficient to sustain many GPP applications with plenty of instructions and data. However, the memory bandwidth requirements of DSPs make the Von Neumann architecture a poor choice. Thus, most DSPs implement a Harvard memory architecture (Figure 1b).

The Harvard memory architecture uses two memory spaces, usually partitioned as program memory and data memory. The two memory spaces are connected to the

processor core by two bus sets, allowing for two simultaneous accesses to memory. This, in effect, doubles the processor's memory bandwidth; and thus, keeps the processor core well fed with large amounts of instructions and data [2]. Modern high-performance GPPs such as the Pentium and PowerPC also implement a similar Harvard architecture to make multiple memory accesses per instruction cycle for superscalar execution and for high data demands [3].

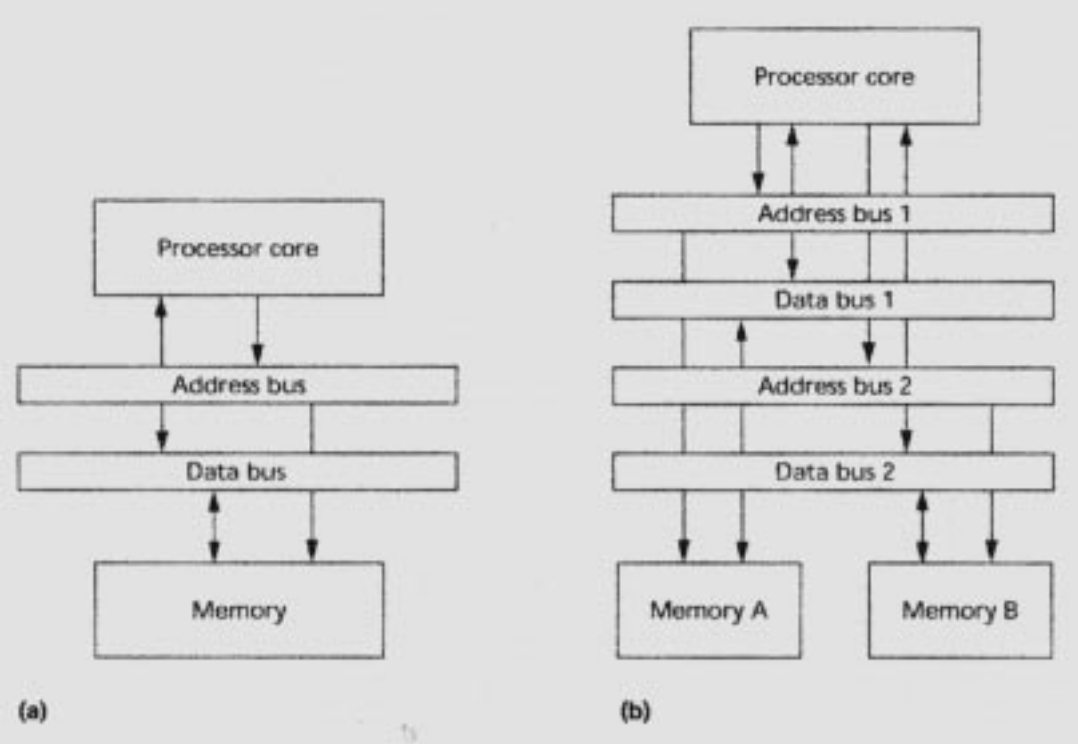

Figure 1: Memory architectures. (a) von Neuman (b) Harvard [3]

GPP AND DSP CACHE NEEDS

Most high-performance GPPs typically contain two on-chip caches—one for data and one for instructions. The performance requirements of GPPs make quick memory accesses an essential criteria in order to meet their design goals, but the high performance nature of these chips often makes it impossible to integrate memory chips capable of keeping pace with the processors. Therefore, caches provide a viable way of sustaining

data demands as well as allowing for data retrieval at full processor speed without accessing slow, off-chip memory. The importance of quick memory access can be seen by IBM's highly anticipated Power4 chip, which is allocating 60 percent of the transistors to cache memory [4].

Unlike GPPs, most DSP processors do not have any cache. Instead, they rely on multiple banks of on-chip memory and multiple bus sets to allow for several on-chip memory accesses per instruction cycle. However, some DSPs do incorporate a small, specialized instruction cache that is used for storing instructions used in small loops so that the on-chip bus sets can be free to retrieve data words. DSP processors almost never include a data cache because the data is typically "streaming." In other words, data samples are often used by the DSP processors to perform computations and then are discarded with little need for reuse [3].

The traditional DSP design has changed with the introduction of Texas Instruments' TMS320C6211, which not only includes instruction and data caches, but also implements them in two-levels. At 150 MHz, it is capable of performing 1200 RISC MIPS, with up to eight instructions per cycle. It has 72 KB of on-chip RAM—4 KB of L1 program cache, 4 KB of L1 data cache, and 64 KB of L2 unified cache [4]. The C6211 is expected to be used for price sensitive applications, such as digital subscriber loop (DSL) clients for small offices or the home as well as high-speed data transmission functions in switches and routers, wireless data clients, imaging, biometrics, remote medial diagnostics, automotive vehicle and drive train control, and security systems [6].

TMS320C6211

The TMS320C6211 utilizes a two-level memory architecture for on-chip program and data accesses (Figure 2). The first level consists of 4 KB of direct-mapped program cache and 4 KB of 2-way set associative data cache. Separate and dedicated L1 caches prevent conflicts that may arise due to fights for memory resources between the program and data busses. A direct map is well suited for the L1P since DSP algorithms consist of small, tight loops. Set associativity is more appropriate for the L1D because data tends to be more random and have more strides than program instructions [8]. Also, the L1D uses a least-recently-used (LRU) replacement scheme, which produces cache allocations that are very close to optimal [9]. The second level is made up of 64 KB that can be used by both the program and data. It can be used entirely as a cache, be directly mapped as internal memory, or be used as a combination of these functions. It is divided into four 16 KB banks, each of which can be programmed as cache or RAM space. With the flexibility allowed by the L2, the user can optimally partition the cache with a balance of RAM, program cache, and data cache [7].

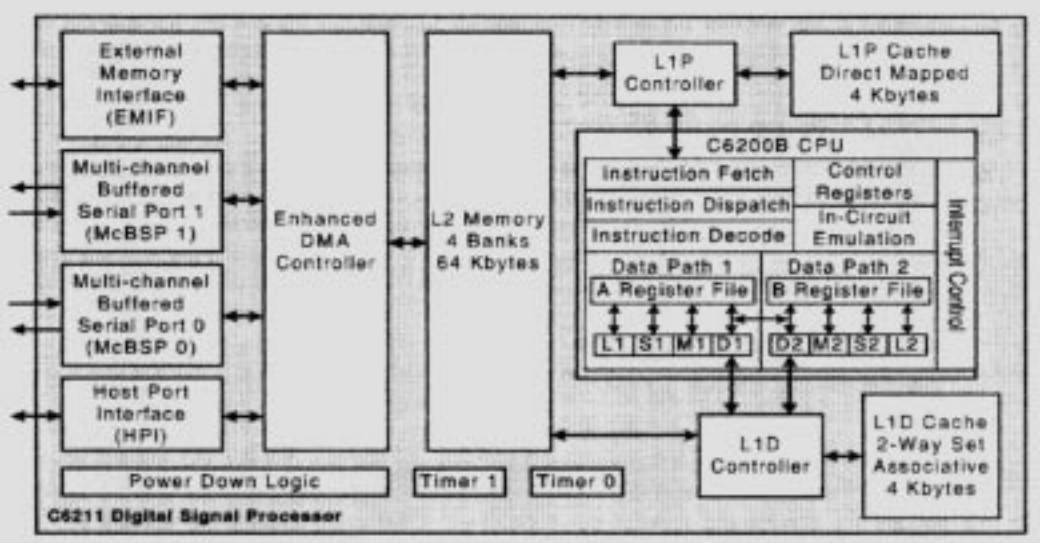

Figure 2: TMS320C6211 Digital Signal Processor [7]

IMPLEMENTATION

Objective

The objective of my study was to substantiate the performance gain realizable by having a cache or caches on a DSP. My approach was to execute several common DSP kernels on an actual DSP with caches to examine its cache utilization to see whether there were true benefits. I chose Texas Instruments' TMS320C6211 for my study because it exploited many features of caching such as using both instruction and data caches, set associativity, and two levels. Also, by using TMS320C6211, I was able to compare my results with Texas Instruments' findings.

My first task was to obtain a tool to conduct my investigation. From Texas Instruments' web site, I was able to download a 30-day free trial version of a commercially available tool, Code Composer Studio. This tool provided a TMS320C6211 simulator that modeled the cache performance of the device. It was able to monitor cache activities and report pertinent information such as cache hits and misses for instructions and data on both levels of caches. I was also able to use several common DSP kernels that came with the tool to perform the evaluation.

Texas Instruments' Findings

Texas Instruments (TI) found that the high L1 hit rate, coupled with the flexible L2 memory organization, allows for a high rate of performance. TI claims that typical applications can operate at more than 80% of the optimal cycle performance [7]. Optimal performance means that the processor has infinite internal program and data memory. TI says that 98% of program fetches hit in the L1P and 91% of data fetches hit in the L1D. With the L2 in the 4-way set associative mode, there is normally a hit 96% of the time

when the L2 is accessed. Overall, more than 99.5% of all CPU cycles execute without going out to external memory [8]. Figure 3 shows the performance efficiency observable on the TMS320C6211 by some DSP applications.

Application	Efficiency
v.34	89%
AC-3 Decoder	90%
Zlib File Compression	96%
Line Echo Cancellation	99%
GSM Frame Encoder	92%
GSM Frame Decoder	88%
ADSL	85%

Figure 3: TMS320C6211 Benchmark Performance [8]

My findings

For my study, I ran several common DSP kernels through the TMS320C6211 simulator and observed the cache utilization. Since one of the most important measure of cache-design effectiveness is the miss ratio, that is what I focused on. The miss ratio is the fraction of the total processor references that are not found in the cache. A low miss ratio (or a high hit ratio) would indicate that the cache is effective since the processor does not have to wait for slower main memory references as often [10]. I was able to examine the L1P and L1D hits and misses, but I was unable to monitor the L2 cache performance. Although the L2 is configurable to 1-, 2-, 3-, or 4-way set associative cache or as a RAM space, I was unable to figure out how to modify the L2 configuration from its default setting as a RAM space. One thing to note, however, is that all the

references that missed in the L1 were found in the L2 RAM space. Table 1 summarizes the results of my investigation.

DSP Kernels	L1P Hit Ratio	L1D Read Hit Ratio	L1D Write Hit Ratio
IIR filter	92493/92517 (99.97%)	891/918 (97.06%)	356/371 (95.96%)
Vector Multiply	225696/225717 (99.99%)	1042/1066 (97.75%)	304/318 (95.60%)
MAC	171135/171158 (99.99%)	1350/1374 (98.25%)	460/475 (96.84%)
FFT	52366/52402 (99.93%)	118408/118805 (99.67%)	69894/69923 (99.96%)
Telecom	240145/240171 (99.99%)	328/338 (97.04%)	153/176 (86.93%)

Kernel descriptions
IIR filter—typical Infinite Impulse Response biquad filter
Vector Multiply—a scalar vector multiply followed by a right shift
MAC—Multiply Accumulate function
FFT—complex radix 4 Fast Fourier Transform
Telecom—implements one stage of Viterbi V32 trellis decoder

Table 1: Summary of Cache Performance

CONCLUSION

As bus speeds continue to increase and the nature of applications changes, the need for caching is becoming more apparent. The analysis of Texas Instruments' TMS320C6211 clearly show the benefits of incorporating a cache structure into the DSP design. My investigation shows a strong correlation to Texas Instruments' findings—mainly, that most program and data accesses hit in the L1. With an L2, the off-chip memory latency penalty can be further reduced. As DSP applications continue to grow, using caches may become the norm instead of being a rarity.

REFERENCES

[1] N. P. Jouppi and S. J. E. Wilton, "Tradeoffs in Two-Level On-Chip Caching," *21st Annual Int. Symp. on Computer Architecture*, April 18-21, 1994, pp. 34-45.

[2] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP Processor Fundamentals*, IEEE Press, ISBN 0-7803-3405-1, 1997.

[3] J. Eyre and J. Bier, "DSP Processors Hit the Mainstream," *http://www.bdti.com/articles/final.pdf*.

[4] J. Markoff, "IBM To Introduce New Computer Processors," *New York Times*, October 5, 1999.

[5] "TMS320C6211," *http://www.horizon-tech.fr/products/hunt/tms320c6000/tms320c6211.htm*.

[6] "TI DSP Will Run at 2,000 MIPS With Lots of Cache," *http://edtn.com/news/sept9/090998pnews1.html*.

[7] Texas Instruments, "How To Begin Development Today With the TMS320C6211 DSP," *Application Report SPRA474*, *http://www.ti.com/sc/psheets/spra474/spra474.html*

[8] Texas Instruments, "TMSC320C6211 Cache Analysis," *Application Report SPRA472*, *http://www.ti.com/sc/psheets/spra472/spra472.html*

[9] H. S. Stone, J. Turek, and J. L. Wolf, "Optimal Partitioning of Cache Memory," *IEEE Transactions on Computers*, vol. 41, no. 9, pp. 1054-1068, Sept. 1992.

[10] D. A. Alpert and M. J. Flynn, "Performance Trade-offs for Microprocessor Cache Memories," *IEEE Micro*, vol. 84, pp. 44-54, Aug. 1988.