Cache Justification for DSP Processors

by

Michael J. Lee

October 22, 1999

Cache Justification for DSP 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, DSPs traditionally did not incorporate any caches, but instead mainly relied on fast on-chip memory banks. 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 a talk on a project implementation idea of investigating a DSP with a cache (or caches) to evaluate its cache utilization and performance enhancement.

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. As applications for DSPs continue to grow, the need for caching is becoming more apparent.

MEMORY ARCHITECTURES

Typical DSP operations require high memory bandwidth to handle the massive amounts of computations that they are called upon to perform. 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 [1]. 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 [2].

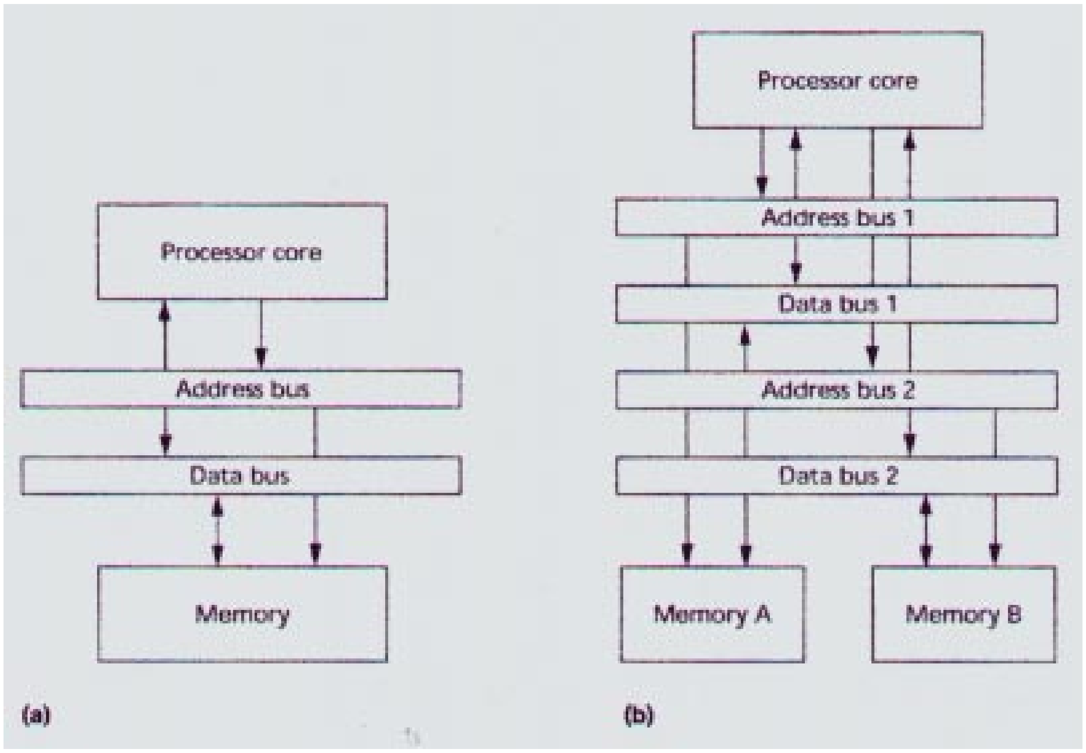


Figure 1: Memory architectures. (a) von Neuman (b) Harvard [2]

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 [3].

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 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 [2].

The traditional DSP design has changed with the introduction of Texas Instruments’ TMS320C6211, which not only includes both instruction and data caches, but also implements them in two-levels. At 150MHz, it is capable of performing 1200 MIPS, with up to eight instructions per cycle. It has 72KB of on-chip RAM—4KB of L1 program cache, 4KB of L1 data cache, and 64KB 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 [5].

TMS320C6211

Cache Structure

The TMS320C6211 utilizes a two-level memory architecture for on-chip program and data accesses (Figure 2). The first level consists of 4KB of program cache and 4KB of data cache. Separate and dedicated L1 caches prevent conflicts that may arise due to fights for memory resources between the program and data busses. The second level is made up of 64KB that can be used by both the program and data [6].

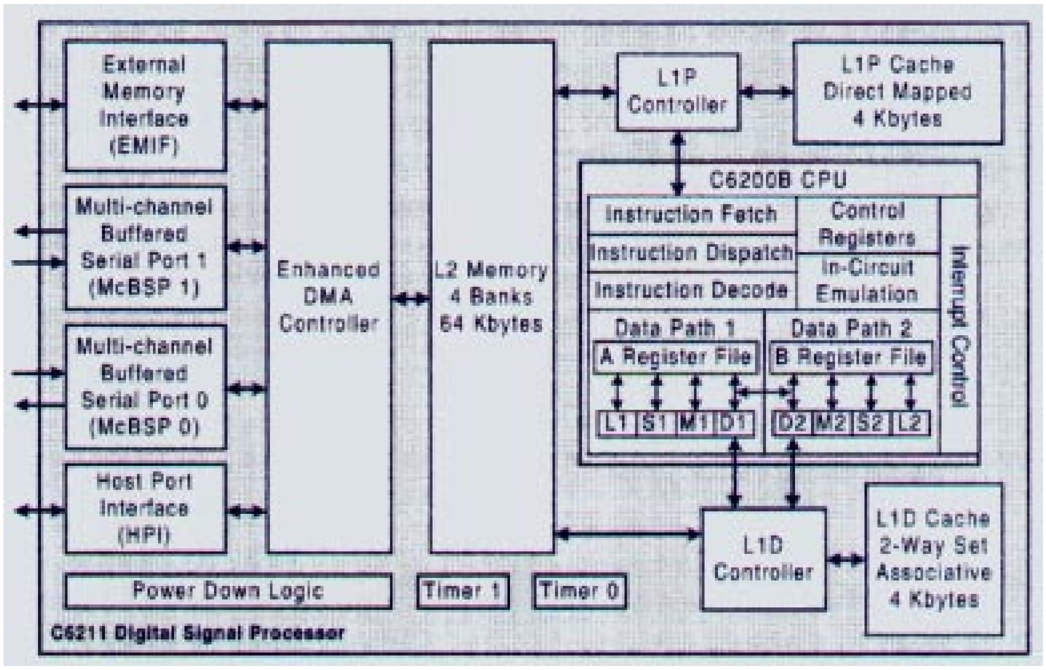

External Memory Interface (EMIF)
Multi-channel Buffered Serial Port 1 (McBSP 1)
Multi-channel Buffered Serial Port 0 (McBSP 0)
Host Port Interface (HPI)
Enhanced DMA Controller
L2 Memory 4 Banks 64 Kbytes
L1P Controller
L1P Cache Direct Mapped 4 Kbytes
C6200B CPU
Instruction Fetch
Instruction Dispatch
Instruction Decode
Control Registers
In-Circuit Emulation
Interrupt Control
Data Path 1
Data Path 2
A Register File
B Register File
L1 S1 M1 D1
D2 M2 S2 L2
L1D Controller
L1D Cache 2-Way Set Associative 4 Kbytes
Power Down Logic
Timer 1
Timer 0
C6211 Digital Signal Processor

Figure 2: TMS320C6211 Digital Signal Processor [6]

The L1P is direct-mapped with a 64-byte line size. A direct map is well suited for the L1P since most DSP algorithms consists of small, tight loops that rarely thrash. The L1P is large enough to accommodate several DSP kernels simultaneously; and since most applications execute sequential instructions, following instructions are already prefetched in the L1P to allow for less number of cache misses.

The L1D is 2-way set associative with a 32-byte line size. Set associativity is more appropriate for the L1D because data tends to be more random and have larger strides than program instructions. The 2-way structure reduces the chance of a cache thrash since two thrashing addresses can be simultaneously stored. The L1D is also dualported to allow for two simultaneous data accesses. The L1D uses a least-recently-used (LRU) replacement scheme to update between the two set [7].

The L2 is a unified 64KB cache that is used for both 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 16KB banks, each of which can be programmed as a cache or RAM space (Figure 3). The amount of program or data in the L2 is also configurable by the user. The amount of associativity is determined by how many of the banks are configured as caches; thus, allowing for 1-, 2-, 3-, or 4-way associativity. With the flexibility allowed by the L2, the user can optimally partition the cache with a balance of RAM, program cache, and data cache [6].

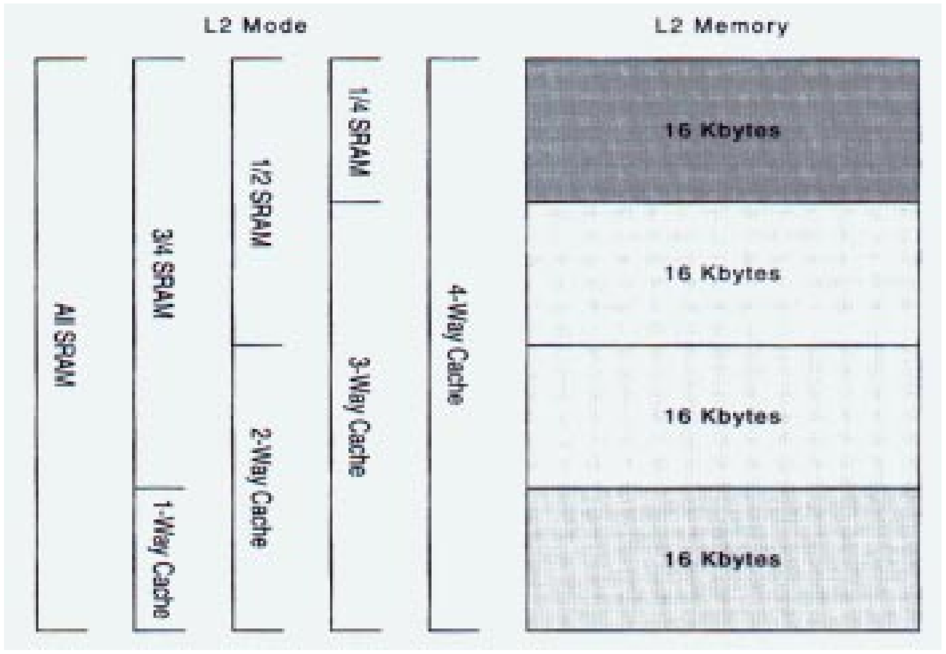
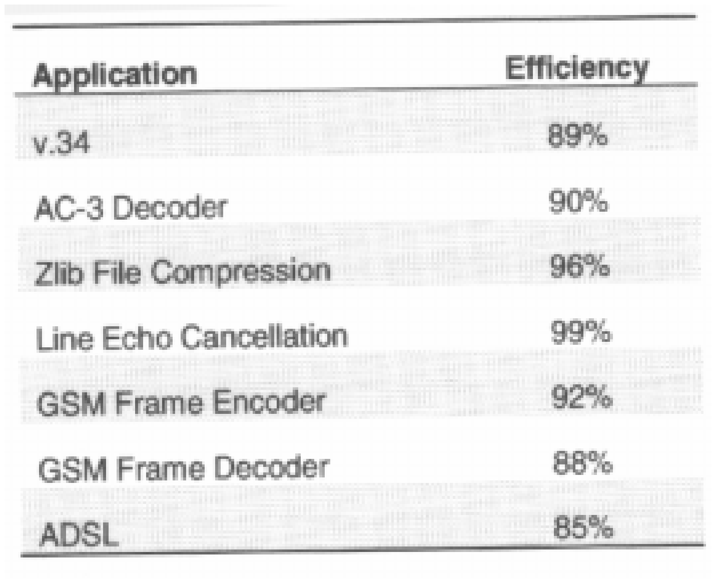


Figure 3: L2 Memory Configuration [7]

Performance

The high L1 hit rate, coupled with the flexible L2 memory organization allows for a high level of performance. Typical applications can operate at more than 80% of the optimal cycle performance [6]. Optimal performance means that the processor has infinite internal program and data memory. Usually, 98% of program fetches hit in the L1P and 91% of data fetches hit in the L1D. In the 4-way set associative mode, there is normally a hit 96% of the time in the L2. Overall, more than 99.5% of all CPU cycles execute without going out to external memory [7]. Figure 4 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 4: TMS320C6211 Benchmark Performance [7]

Benefits of a Two-Level Cache

The two-level cache architecture of the TMS320C6211 offers many benefits over a non-cache or a one-level cache design. The highly efficient two-level cache scheme allows slower, less expensive memories to be used, helping to reduce the overall system cost. In addition, the level of associativity and the high cache hit rate require virtually no optimization to be done, shortening code development and accelerating time to market. Also, the unified L2 allows the user to allocate the appropriate amount of memory for both program and data, tailoring to the application's needs [7].

CONCLUSION

For the most part, DSPs have lacked caches since most applications did not have a need for them. However, as DSPs are called on to perform more and more functions, the justification for having caches is evident. The analysis of Texas Instruments' TMS320C6211 clearly show the benefits of incorporating a cache structure into the DSP design. As DSP applications continue to grow, using caches may become the norm instead of being a rarity.

PROJECT IMPLEMENTATION

For my project, I would like to investigate the true advantages of having a cache hierarchy. Specifically, I would like to benchmark a DSP, perhaps the TMS320C6211, to validate the performance enhancements. If time permits, I would also like to see which type of applications really benefit from the caches. Through this project, I hope to gain a good understanding for the justification of having caches on DSPs.

REFERENCES

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

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

[3] J. Markoff, “IBM To Introduce New Computer Processors,” *New York Times*, October 5, 1999.

[4] “TMS320C6211,” *http://www.horizon-tech.fr/products/hunt/tms320c6000/tms320c6211.htm*.

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

[6] Texas Instruments, “How To Begin Development Today With the TMS320C6211 DSP,” *Application Report SPRA474.*

[7] Texas Instruments, “TMSC320C6211 Cache Analysis,” *Application Report SPRA472.*