Advanced DSP Performance Complicates Memory Architectures In Wireless Designs

FUTURE DSP BENCHMARKING AND SELECTION MAY BE DIFFICULT WITHOUT TAKING MEMORY ARCHITECTURE INTO CONSIDERATION.

BY ETHAN BORDEAUX

OVER the years, a wide variety of digital-signal-processor (DSP) core architectures has been developed for wireless applications. Some of these include single-instruction/multiple-data (SIMD), very-long-instruction-word (VLIW), complex-long-instruction-word (CLIW), and static-superscalar-processing cores. While these designs have received a great deal of scrutiny, it is often the memory architecture attached to these cores that decides whether a particular processor is well-suited for a specific application.

What is meant by memory architecture is the way in which internal memory is connected and accessed by the core-processing units. The proper memory architecture for a particular application is primarily defined by the data requirements for the algorithm, along with the necessity for absolute determinism for the specific computation. If the internal memory cannot properly support the data requirements or the raw processing performance of the DSP, then total processing efficiency is compromised.

MULTIPLY ACCUMULATE

The multiply/accumulate (MAC) is the kernel of many signal-processing operations, including convolution, filtering, as well as echo cancellation. Typically, this instruction is contained within a tight loop, causing the processor to repeated multiply and add a series of data values located in separate memory buffers. It can be written in pseudocode as:

result = result + (data1 × data2), data1 = newval1++, data2 = newval2++;

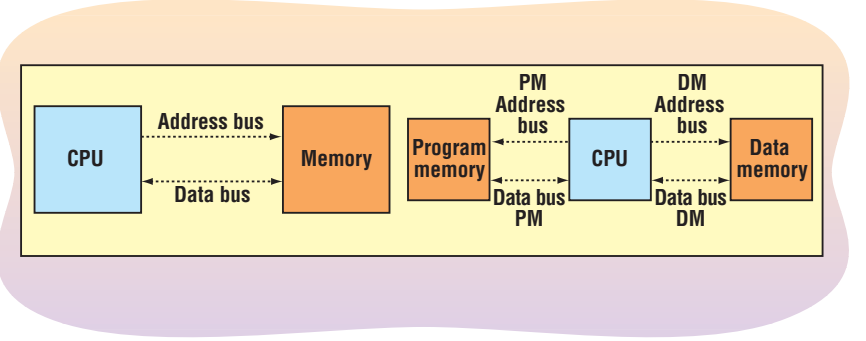
In this case, there are a total of three memory fetches, along with the two computational operations. These memory fetches are opcode (instruction) fetch, data1 (data) fetch, and data2 (data) fetch.

To sustain this operation in a single cycle (assuming the core processor can perform a single-cycle MAC), the processor must fetch three operands from memory in a single cycle. If it cannot, there is a stall and performance is dramatically reduced **(see table)**.

MEMORY ARCHITECTURES

One of the simplest processor memory architectures is known as the Von Neumann architecture **(Fig. 1)**, where a single address and data bus extends from the core processor to internal (and often into external) memory. While this is an old and not particularly complicated internal bus design, it is still common in many microcontrollers and microprocessors where raw performance (especially in data-intensive operations such as a MAC) is not a primary concern. This architecture is well-suited for "command-and-control" applications, where the processor is not continually fetching data for computations. With the MAC as an example, it is clear just how poorly this architecture performs in signal-processing situations. Due to the single address and data bus, there must be three separate fetches to internal memory to collect the instruction and two data operands for the MAC. This is clearly not an efficient architecture for DSP algorithms.

A Harvard architecture is one that provides two separate memory spaces and buses: one for instructions—program memory (PM), and another for data—the data memory (DM). The independence of the PM and DM buses enables fetches from both memory spaces in a single cycle. While this is a dramatic improvement over the Von Neumann architecture for signal-processing applications, it is still not fully optimized for a MAC. The MAC instruction requires a single PM fetch and dual DM fetches. The Harvard architecture in its most basic form only supports a single DM fetch. Therefore, the standard Harvard architecture requires two cycles for this operation. While this seems to be on the right track toward finding a suitable memory architecture for signal processing, some modifications are needed in the standard Harvard architecture to optimize performance. This points to the modified Harvard architecture, which supports dual DM fetches along with a PM fetch in a single cycle.



1. *One of the simplest processor memory architectures is the Von Neumann architecture (left). The Harvard architecture (right) is a dramatic improvement over the Von Neumann system for signal-processing applications, it is still not fully optimized for a MAC.*

While there are a great number of ways this can be done, there are three common methods.

Double-pumped memory: One path to dual data fetches in a single cycle is to make a portion of on-chip memory accessible twice in a single instruction cycle. Typically, this support is added to PM. In the case of the MAC operation, the programmer places one of the data buffers in DM and another in PM. When the MAC operation executes, the processor accesses PM twice in a single machine cycle—once to fetch the next opcode and again to fetch one of the data operands for the next instruction **(Fig. 2)**.

Dual data buses: Another method of modifying the standard Harvard architecture to enable a single-cycle MAC is to split data memory into separate memory spaces and provide separate buses to each data-memory region. Therefore, in a single cycle, the DSP accesses program memory and both data-memory regions.

Program-memory cache: One last method of enabling a single-cycle MAC through a modified Harvard architecture is by using a PM cache. In this case, there are still two physical memory spaces—one for PM and another for DM with a single bus running to each block of memory. If a particular algorithm requires dual data fetches (such as an FIR filter which is essentially a loop of MACs), the programmer places one buffer in PM and another in DM. The first time that the processor executes this instruction, there is a one-cycle stall because it must fetch the next opcode and the next piece of data over the PM bus. However, whenever there is bus conflict, the DSP "caches," or stores, the instruction in a small (typically 16 to 128 locations) memory space. Assuming it has not been removed from the cache, the next time the program sequencer points to this instruction, it skips the opcode fetch and just fetches the data while the opcode comes from the cache over a separate bus. Therefore, the processor achieves three bus performance with two buses. Each of these three methods of modifying the standard Harvard architecture enables a single cycle MAC—hence, the bandwidth needed to enable the highest processor performance. However, each of these methods follows a different path to this goal.

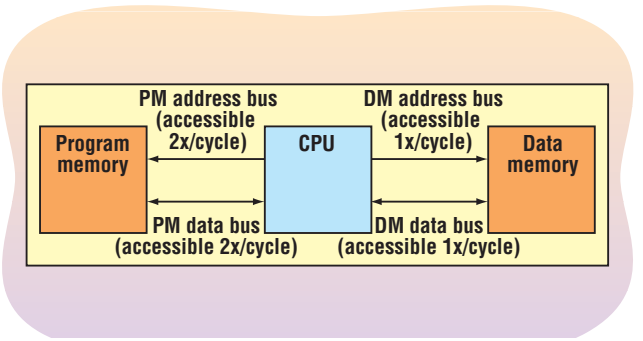
The double-pumped memory and dual data-bus models for single-cycle MAC performance are certainly the simplest methods of increasing performance of the standard Harvard architecture. In both of these cases, there are no restrictions on the location of the MAC to enable single-cycle performance. The big disadvantage of the double-pumped memory model is that this often limits the maximum operational speed of the processor. If the DSP is executing instructions at 75 MHz (13.3 ns), the internal memory must run at 150 MHz (an access time of 6.7 ns). However, as long as the DSP itself executes code at a rate suitable for the specific application, this point is moot and the double-pumped memory model is perhaps the easiest method of enabling single cycle, dual data-fetch execution. The dual data-bus memory architecture is another simple way of increasing the performance of the standard Harvard architecture. The disadvantages of this method are more at the chip-design stage rather than when the original-equipment-manufacturer (OEM) designer programs the chip. Since there are now three separate address and data buses on-chip, this could force the design of the processor to either require additional silicon (Si)/metal layers or expand the total die size.

MAIN DISADVANTAGES

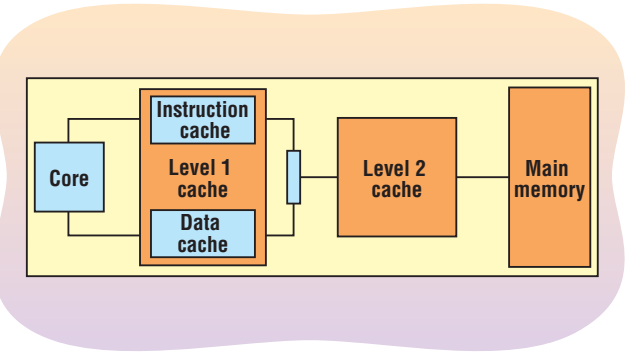
The main disadvantages of the PM cache are increased programming complexity and a performance loss over the other memory architectures. The complexity comes from the fact that the programmer must keep the PM cache in mind when writing code for the processor, especially when looking at the tight inner loops. Code must be optimized so that the cache is accessed as much as possible to enable the highest performance. This means that the number of instructions within looped code that access the cache should be less than the total length of the cache, and that data should not be placed in PM unless absolutely necessary. Fortunately, most modern DSP development tools provide statistical profiling on cache hits versus cache misses when simulating and emulating code.

Overview of common memory architectures

Architecture	Cycles for MAC	Advantages	Disadvantages
Von Neumann	3	None for data-intensive code	Slowest for algorithims based on MAC operation
Harvard	2	Faster than Von Neumann	Slower than all modified Harvard architecture for MAC
Modified Harvard double-pumped memory	1	Coding simplicity	Top processor speed stunted because of two accesses per cycle requirement, higher power dissipation over PM cache design
Modified Harvard dual data bus	1	Coding simplicity	Increased silicon area, higher power dissipation over PM cache design
Modified Harvard PM cache	1	Lower power consumption over other modified Harvard architectures	Slight performance decrease over other modified Harvard architectures, increased coding complexity
L1/L2 cache modified Harvard	1	Faster memory, increased memory flexibility	Loss of data determinism, increased complexity when coding for deterministic behavior



2. *When the MAC operation executes, the processor accesses PM twice in a single machine cycle—once to fetch the next opcode and again to fetch one of the data operands for the next instruction.*



3. *If the required information is not found in the L2 cache, the information is assumed to be external memory and is fetched into the L2 and L1 caches, as shown in this L1/L2 cache-modified Harvard architecture.*

This profiling eases the burden on the programmer, who can focus optimization efforts on areas where the cache is not used efficiently.

It is important to note that even when code is fully optimized to use the PM cache, there are always times when the cache is not preloaded properly and a cache miss occurs. Therefore, execution time is always somewhat slower in PM cache architecture over double-pumped or dual data-bus architectures. However, this difference is often trivial in real-world applications. For example, if a particular algorithm requires a MAC operation to occur 100 times within a single-cycle inner loop, PM cached architecture would have a cache miss the first time it executed the instruction, but a cache hit the next 99 times. This results in 99-percent efficiency versus the other modified Harvard architectures.

There is one important advantage of a PM cache—it often leads to lower core-processor power consumption. This is because the PM cache memory is typically located close to the core and, therefore, does not use the large address and data buses that go to main system memory. In the double-pumped and dual data-bus architectures, the DSP must drive a total of three full address and data buses every time it executes a instruction that requires dual data operands. The PM cached architecture only drives two full buses, along with the cache bus. This is particularly significant if the DSP is used primarily as a computational workhorse, where it is almost exclusively executing algorithmic code (and, hence, continually using the PM cache). The power specifications in processor datasheets are often based on the "typical" instructions the DSP runs in a system. However, if a system is using the DSP primarily for algorithms that take advantage of the PM cache, the difference in power consumption for the PM cached DSP over the other memory models may be significantly greater.

The L1/L2 cache: The L1/L2 cache is a memory architecture that has primarily been used in microprocessor designs, but not in DSPs. This memory arrangement provides for two-memory levels on-chip. The first level (L1) is relatively small and connects directly to the processor core. In a microprocessor, there might be a single L1 memory space where it fetches either code or data. However, in a DSP, it is often a modified Harvard architecture itself with separate memory spaces for code (L1PM) and data (L1DM) along with provisions for transferring an opcode and two data words to the core in a single cycle. If the required instruction or data are not found in the L1 cache, the processor then looks in the L2 cache for the information. If it is found, the information is loaded into the L1 cache, along with the information from a small series of addresses located immediately after the L2 hit location. The assumption here is that if information at address n is required, there is a high likelihood that information at addresses n+1, n+2, etc. is needed in subsequent operations. This is primarily true in tight inner loops with data buffers, but not in control code that involves jumps and calls to a number of different locations. If the required information is not found in the L2 cache, the information is assumed to be in external memory and is fetched into the L2 and L1 caches **(Fig. 3)**.

The primary reason this memory architecture has not yet been widely adopted into DSP applications is due to an inherent lack of execution time determinism in an L1/L2 cache architecture. This is not a big concern in microprocessors that are in personal computers (PCs), because most operations do not require the level of determinism found in embedded DSP applications. It is not of paramount importance if it takes a computer 1.525 or 1.526 s to open a particular application. This is in contrast to embedded applications where the algorithms are based on data being computed within an absolutely rigid period of time. As an example, if a DSP cannot keep up with the data input/output (I/O) requirements of a G.729 vocoder (which are specifically designated in the G.729 specification), the output data are invalid and the vocoder operates improperly. Due to this level of indeterminacy, prototyping of a total system based on a cached DSP is required to realistically see the lowest level of performance for a specific algorithm.

L1/L2 cached memories are expected to grow in popularity on DSPs in the future. This is true because the increased memory requirements of new applications may not allow chip manufacturers to place large enough static random-access memories (SRAMs) onboard to support the entire application. L2 cache does not need to run at the full clock rate of the processor and is configurable to hold either code or data at runtime.

As DSP performance continues to increase, this architecture may no longer be reasonable (or even feasible) and memory designs will appear that go against the old standards. In the future, it may not be as easy to benchmark and select a DSP without taking the memory architecture and the system requirements of the processor into consideration. *WSD*

ETHAN BORDEAUX, *DSP Applications Engineer, Analog Devices, P.O. Box 9106, 3 Technology Way, Norwood, MA 02062; (781) 461-3094, FAX: (781) 461-3300.*