CACHED MEMORY PERFORMANCE CHARACTERIZATION OF A WIRELESS DIGITAL BASEBAND PROCESSOR

Srikanth Kannan†, IEEE Member, Michael Allen†, IEEE Member, and Jose Fridman‡, IEEE Senior Member

Analog Devices, Inc.

‡ RSTC
804 Woburn Street
Wilmington, MA 01887

† 6500 River Place Boulevard
Building 4, Suite 300
Austin, TX 78730

ABSTRACT

In this paper we present performance analysis results of the MSP500 *Digital Baseband* (DBB) platform, a system developed at Analog Devices, Inc., targeted at cellular handsets supporting the GSM, GPRS, and EDGE communication standards. We focus on a particular member of the MSP500 family, the AD6532 device, which integrates a Blackfin® core, and examine the execution time performance of a number of wireless physical layer software components from the perspective of an instruction- and data-cached memory hierarchy. The Blackfin is a 16-bit fixed-point core that combines some of the best features of DSPs and micro-controllers, and has support for a cached memory system.[1]

1. INTRODUCTION

Commonly accepted standards for building communications software on wireless Digital Baseband (DBB) systems for cellular handsets rely on detailed management of the code and data components in the available on- and off-chip memory resources. Under this approach, and due to the stringent real-time requirements of these systems, all time-critical software components are typically placed and moved by a system designer according to heuristics derived by profiling execution schedules. For instance, when a particular time-critical module, such as a data equalizer, is required, its corresponding code and data are moved from slow storage into fast SRAM prior to execution. This pre-loading stage ensures that the module executes in the fastest possible time, and within predefined execution time bounds.

However, cellular systems have in recent years grown in complexity to an extreme degree, especially for the emerging generation of handsets, which explicitly require support for multiple modes of operation. For instance, a GSM/GPRS modem must co-exist with an FDD WCDMA modem, or what we refer to as a *multi-mode handset* [1]. In addition to cellular standards, we are also experiencing convergence with multiple wireless connectivity standards in the cellular handset, such as IEEE802.11 Wireless LAN and Bluetooth. The multiplicity of wireless standards places a tremendous burden on what already constitutes a highly complex hardware and software system. And further, in a variety of scenarios, several modes of operation of these standards must co-exist and be active at the same time.

We believe that the conventional model that requires the explicit movement of instruction and data objects is no longer sufficient, and will not support the development of current and future multi-mode handsets. For this reason, the MSP500 DBB has a hierarchical memory model, where the memory layer closest to the DSP can be configured as Harvard-type instruction and data caches. Building a system on a cache-based memory architecture minimizes the need to explicitly move objects according to execution schedules.

However, as a consequence of a cached memory system, two performance aspects are affected: execution time has an additional overhead component (due to cache line fills and stores), and this overhead is not a fixed number, but rather it can fluctuate in response to system behavior taking place outside of the module in execution (due to left-over remnants of other code and data in the caches). In this paper we present results of analysis to quantify the degree to which performance is affected by a cached memory system.

2. METHODOLOGY AND TOOLS

2.1 Metrics

We use the metric *Instructions-per-Clock* (IPC) to denote the efficiency of a module [2]. An IPC of 1.0

[1] Blackfin is a trademark of Analog Devices, Inc.

represents a system with zero overhead, where an instruction executes in one cycle.

There are two components that contribute to degrade IPC of a module. One is due to the stalls of a processor pipeline, and the other is due to the stalls of a memory system.

2.2 Run-Time Model

As mentioned above, one of the complexities introduced by a cached system is the interaction between different modules in a system that arises from one module leaving remnants of code and data in the caches. These remnants affect the behavior of other modules in a manner that is difficult, if not impossible, to predict accurately at run-time.

Since we are interested in the worse-case execution time of a module, we run our experiments under the assumption that the module is interrupted or preempted at a rate *R*, and that during each interrupt both the instruction and data caches are fully flushed, that is, when the module resumes execution, the caches no longer have instructions or data belonging to the target module. This model assumes that the target module is preempted for relatively long periods of time, and that the modules that execute during these intervals completely flush the caches. This represents the worse case run-time condition under which a module is ever going to execute, and hence it yields the worse case execution time.

2.3 Blackfin System Simulation Environment (BSSE)

For the AD6532 DBB, we developed an efficient method to port drivers, applications and benchmarks from any development platform onto the Blackfin-System Simulation Environment (BSSE). We use the Analog Devices VDSP++ software tools. The simulation environment has been instrumented to provide debugging information (e.g., wave dump), performance analysis reports, and comprehensive simulation reports. We made extensive use of this methodology to simulate the source modules shown in this paper.

3. AD6532 DIGITAL-BASEBAND CHIP

The DBB is a key component of the wireless handset. Depending on different terminal architectures and level of integration, the DBB encompasses a complete set of baseband processing functionality with possible mixed signal (digital-to-analog and analog-to-digital converters, as well as numerous auxiliary and voiceband converters) integrated on chip [3]. DBB architectures for wireless communications typically rely on multi-core platforms, including DSP and microcontroller cores, as well as dedicated hardware functionality (accelerators, co-processors) for a given wireless standard, and large amounts of internal memory [4].

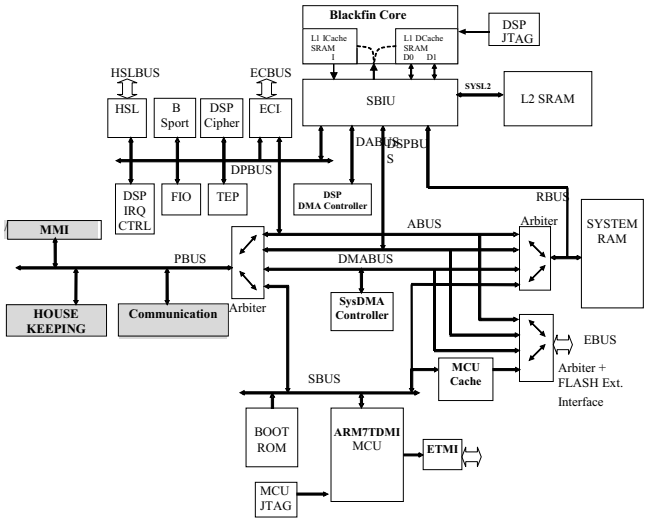


Figure 1: Top-level block diagram of the AD6532 system.

Figure 1 shows a block diagram of the AD6532. At the top-right is the Blackfin sub-system, which consists of the Blackfin core, L1 code and data memories (configurable as cache or SRAM), unified L2 memory, Blackfin DMA controller, and peripherals (Timing and Event Processor, Blackfin interrupt controller, High Speed Logger, Bsport, External Coprocessor Interface). The Blackfin subsystem is connected with the SBIU crossbar. This sub-system is clocked at the maximum speed in the system. At the lower right is the ARM7TDI sub-system, which consists of the ARM7 core, cache and DMA. The lowest level on-chip memory in the system is called System RAM (this is L3 memory), and is accessible by both the Blackfin and the ARM. The rest of the system consists of general connectivity peripherals for control of most devices present in a wireless terminal, as well as control of the *Analog Baseband* (ABB) and Radio systems.

One of the most important aspects of the AD6532 is that is it based on a hierarchical memory system. From the perspective of the Blackfin, L1 memories provide a limited amount of fast zero-wait state storage. Due to the high speed the Blackfin is capable of running, L1 memory is expensive in silicon area, and it is economically feasible to incorporate only a small amount of fast memory. Lower levels of memory provide larger capacity storage, but at slower access times.

The memory system consists of

1. L1 Memory: Caches and SRAM at core frequency, 80KB instruction, 32KB data.
2. L2 Memory: Pipelined 64KB SRAM at half core frequency.

3. L3 Memory: 512KB at ¼ core frequency. Also referred to as SYSRAM.
4. L4 Memory: large capacity off-chip SRAM.

3.1 Blackfin Core Memory System

The Blackfin core contains an L1 instruction memory and a separate, banked, dual-ported L1 data memory. This allows two data load operations or one load operation and one store operation to occur in parallel with an instruction fetch. Both the L1 instruction and L1 data memories may be configured either as caches or as SRAM.

The L1 instruction memory can be configured as either 16KB SRAM or as 16KB of lockable 4-way set-associative instruction cache, and has a dedicated 64KB instruction SRAM bank. The instruction address bus is 32-bits wide; the instruction data bus is 64-bits wide. Misses in the L1 instruction memory are sent to the system bus interface. An external bus master, DMA for example, can access the L1 instruction memory when it is configured as SRAM. The DMA port has a 32-bit address bus, and a 64-bit wide data bus capable of transferring data into the internal SRAM memories.

The L1 data memory consists of two 16KB banks. Each of the 16KB banks can be configured as either 16KB SRAM, or as 16KB of 2-way set associative data cache. The data address busses are 32-bits wide. The load data busses and the store data bus are 32-bits wide. Misses in the L1 data memory are sent to the system bus interface. An external bus master (e.g., DMA) can access the data banks in the L1 data memory when they are configured as SRAM, across the same interface used to access instruction SRAM.

We have found that the most general configuration for the L1 memories of the Blackfin is: bank A configured as data cache, bank B as SRAM, and the instruction memory as cache. This configuration allows the placement of some data objects in L1 memory (bank B), and allows the caching of data objects that have relatively good caching performance, such as the stack and look-up tables (which are cached through bank A).

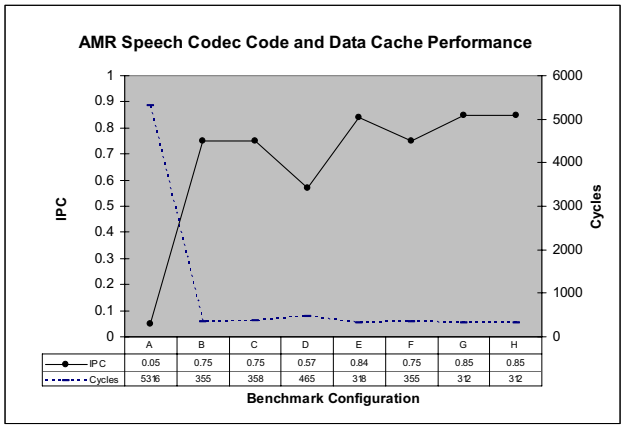
4. ANALYSIS OF PHYSICAL LAYER MODULES

Each of the following modules consists of four objects: *program*, *data* (generally all working buffers that contain temporary and static data), *stack*, and *tables*. *WT* and *WB* refer to write-through and write-back data cache modes [2].

4.1 AMR-NB

We use an implementation of the AMR-NB Speech Codec as a representative benchmark. This is a significant module since it consumes approximately 50% of the total DSP processor capacity during an active voice call. Hence, it remains critical to optimize this module. The following are the test configurations:

	Program	Data	Stack \ Tables	Icache	Dcache
A	L3	L1	L3	Off	Off
B	L3	L1	L3	On	WB
C	L1	L1	L4	n/a	WB
D	L1	L1	L4	n/a	WT
E	L1	L1	L3	n/a	WB
F	L1	L1	L3	n/a	WT
G	L1	L1	L2	n/a	WB
H	L1	L1	L2	n/a	WT



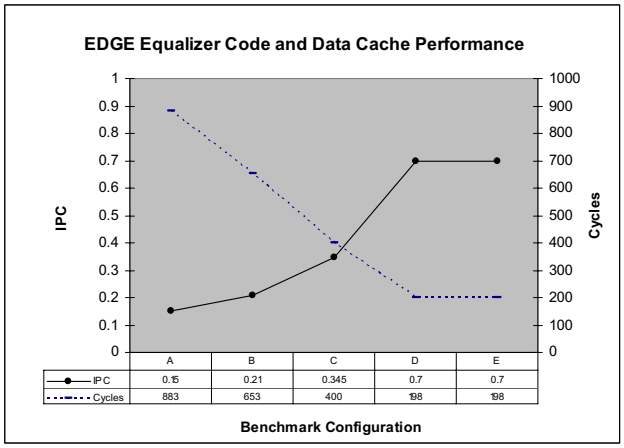
This set of experiments show the difference in performance between WB and WT data cache modes. As can be seen in the above graph, holding the program placement constant in L1, the performance difference between WB and WT data cache modes increases as memory latency increases. We measure over 30% performance increase when going off-chip to access stack-resident data on external SRAM with WB over WT (cases C and D).

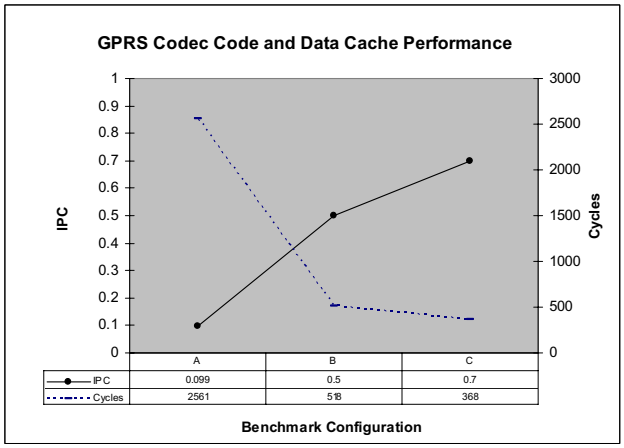
With regard program placement, we can see that there is a 10% performance degradation when the program is executed from L3 storage with the Icache, relative to L1 execution (cases B and E).

4.2 EDGE Equalizer

We use an implementation of the EDGE equalizer based on the DDFSE algorithm.

	Program	Data	Stack	Tables	Icache	Dcache
A	L2	L2	L2	L2	Off	Off
B	L2	L2	L3	L2	On	Off
C	L3	L3	L2	L2	On	Off
D	L3	L2	L2	L2	On	WB
E	L3	L2	L2	L2	On	WT





This test case explores the behavior of a program that makes extensive references to data structures placed in L2 and L3, rather than in L1 as the AMR-NB test case. The Data objects hold most of the working and temporary buffers, while the stack holds small working buffers, input/output parameters, etc. If the data structures are not in L1, then clearly IPC is too low unless data caching is enabled (cases A, B, and C).

When the data structures reside in L2, there is not a significant variation in performance between WB and WT data cache modes (cases D and E). The reason is that L2 memory is fast and pipelined, and there is not enough write traffic in the EDGE Equalizer to back up cache write buffers, hence line stores have little negative impact. It is only with slower, un-pipelined memories (i.e., L3 and L4 as in the AMR-NB case) that WB mode is generally faster.

In general, when allocating space to modules we use the WB mode, and make no distinction for modules that generate relatively light store traffic.

In this benchmark there is little variation in performance relative to changes in program placement. This is due to the fact that the overall EDGE Equalizer is a relatively small code module, compared to the AMR-NB and GPRS Channel Codec modules.

4.2 GPRS Channel Codec

We use an implementation of the GPRS channel codec operating in CS4.

	Program	Data / Stack	Tables	Icache	Dcache
A	L3	L2	L3	Off	Off
B	L3	L2	L3	On	WT
C	L3	L2	L3	On	WB

This test case is similar to the EDGE Equalizer in that it measures the effect of data traffic to L2. However, in this module there is a large difference in performance between a data cache operating in WT or WB mode, with WB improving performance by 30% over WT. The reason for this is that the GPRS Channel Codec has much more store traffic than the EDGE Equalizer, hence backing up write buffers, even to the fast, pipelined L2 storage.

CONCLUSIONS

The AD6532 Digital Baseband device integrates the Blackfin® core, with a flexible memory hierarchy to achieve an optimal balance between performance, power consumption, and cost. The data presented here supports the benefit of cache memory in achieving this balance. The data also suggests that, for several key algorithms of the GSM/GPRS/EDGE physical layer, a significant performance increase can be realized for some modules by operating the data cache in write-back mode, vs. write-through mode, depending on the amount of store traffic that a particular module generates.

REFERENCES

[1] J. Reed, *Software Radio: a Modern Approach to Radio Engineering*, Prentice Hall, Upper Saddle River, NJ, 2002.

[2] J. L. Hennessy and D. A. *Patterson, Computer Architecture: A Quantitative Approach,* Second Edition, Morgan Kaufmann Publishers, 1996.

[3] J. Soerensen, P. Birk and Z. Zvonar, "New Challenges for Integrated Circuit Solution," Wireless Personal Communications, An International Journal, Special Issue on the Future Strategy for the New Millenium Wireless World, Vol 17, No 2-3, pp 291-302, June 2001.

[4] Z. Zvonar, J. Fridman and D. Robertson, "Software-defined radio in wireless handsets: evolution versus revolution," EE Times , August 9, 2002.