

X2000 ADVANCED AVIONICS CHARACTERIZATION STUDY¹

Len Day – len.day@jpl.nasa.gov - 818-354-4308
Don Meyer, Nick Nicolich, Rob Steele, Igor Uchenik, Ken Vines
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

Abstract — The X2000 Advanced Avionics project at JPL was created to provide the core spacecraft avionics for use on multiple future missions. The avionics suite includes a radiation hardened PPC 750 as well as various I/O and mass storage devices.

A key factor in modern computer architectures is the fact that the CPU operates much faster than does the main memory. This has necessitated caching techniques to obtain full performance from the processor, and has also made software performance highly dependent on cache activity.

This dependency on cache activity makes it more difficult to predict whether a given application will perform as expected on a given platform. Simple benchmarks are of limited use since the cache behavior is generally predictable and does not accurately reflect the behavior of an application running in a more complex environment.

A “X2000 Advanced Avionics Characterization Study” was performed in which a set of tests were run, beginning with single tests of both CPU intensive and I/O intensive activities. These tests were then run in combination to build up to a very active complex computing environment in order to expose performance interactions. This should then provide a wide enough range of examples to allow a potential user to more accurately estimate the expected performance of their application. Measurements were collected on a per-task basis for the actual execution rate and cache performance by using the Power PC 750 performance monitoring registers.

The characterization study has shown that adjustments in an application’s data accesses can easily create a performance difference of three or more times in actual applications. In general, by iterating over a small portion of a data set rather than its entirety, execution can remain within cache thereby producing a performance increase. Additionally, the approaches used in performing I/O can make a major performance difference.

As system activity increases differences are seen in a application’s cache performance, but do not appear to be a

major factor in real applications. Contention for the PCI bus can also be a performance issue to consider.

TABLE OF CONTENTS

1	INTRODUCTION	1
2	X2000 OVERVIEW	1
3	PROCESSOR PERFORMANCE FACTORS.....	2
4	CONCLUSIONS	6
	REFERENCES	8

1 INTRODUCTION

As new spacecraft processors become available, their dependency on memory and cache interaction makes it more difficult to predict the actual runtime performance that they will achieve. A "X2000 Advanced Avionics Characterization Study" was performed to exercise a variety of applications on the BAE Systems RAD750 Power PC processor and other mass memory and I/O devices developed as part of the JPL X2000 project. This study characterizes their performance and provides measured results for a wide variety of operational scenarios. This paper will highlight the results of this study and attempt to provide useful information to those that need to do system engineering, design, and implementation of real time systems on new processor architectures.

2 X2000 OVERVIEW

The X2000 hardware that resides in the main chassis are Compact PCI (CPCI) 3U form factor. As discussed below,

¹ 0-7803-7651-X/03/\$17.00 © 2003 IEEE

certain components do not reside in the main chassis but instead connect over the I²C bus.

The X2000 hardware suite consists of the following components:

- SFC (Spacecraft Flight Computer): Developed and manufactured by BAE Systems, the SFC is a radiation and SEU hardened, fully PPC compliant processor. The CPU is a Power PC 750 processor running at 132 MHz with onboard 128 MBytes of RAM and the PowerPCI PCI host bridge. The host bridge is a new development by BAE, based on the Motorola MPC 106 host bridge interface. The PowerPCI contains the memory controller, PCI interface and various ancillary components. The name BAE has given to this board as a product is "RAD750".
- NVM (Non-Volatile Memory): Developed and manufactured by SEAKR Engineering, the NVM board is the mass storage component consisting of two 128MByte banks of non-volatile storage (flash memory) on each card for a total of 256MBytes.
- DIO (Digital I/O): Developed by JPL, the DIO is an ASIC providing an IEEE 1394 (Firewire) interface, two I²C interfaces, a high-speed UART interface, a watchdog timer capable of resetting the CPU and a set of count-down timers. At this writing the DIO ASIC has not been fabricated and all testing has been done with an FPGA version. Unfortunately the FPGA version is limited to half speed PCI (16.5 MHz) and quarter speed 1394 (25 Mbps) and as such impacts the performance numbers presented. Comments will be made where appropriate to indicate expected behavior with fully capable DIO components.
- SIO (System I/O): The SIO is a card containing two DIOs and the associated interface circuitry.
- SIA (System Interface Assembly): Developed by JPL, the SIA is designed to be the instrument and telecom interface. It is a card that provides 4 high-speed synchronous serial instrument interfaces, a single MIL-STD-1553 interface and a synchronous serial telecom interface.
- PSS (Power Switch Slice): The PSS is not a CPCI card, but rather is coupled to the system via redundant I²C interfaces. Developed by JPL, the PSS contains 16 switches and a sophisticated commanding capability. It has been designed to not only control normal spacecraft loads but also for pyro firings and valve drives.
- TRIO (Temperature Remote I/O): Developed by APL, the TRIO is primarily an analog to digital

converter providing up to 32 channels of input and a single I²C interface to the rest of the system. It can be used for either temperature or voltage measurements depending on configuration.

The SFC (RAD750), NVM, SIA and SIO are prototype boards. The PSS and TRIO used for this testing are pre-prototype development implementations.

Additionally, there is a PCS (Power Control Slice) under development at this writing that was not available for testing.

3 PROCESSOR PERFORMANCE FACTORS

CPU performance and the number of instructions that are executed can vary widely depending on the specifics of software execution. The theoretical maximum execution speed of the PPC750 running at 133MHz is approximately 240 MIPS. Other factors as discussed below can then degrade this value.

Previously processors have been primarily rated in MIPS (Million Instructions Per Second) and most applications were serviced at the advertised instruction execution rate. One of the major messages of this document is that in new architectures MIPS becomes extremely variable depending on the processor's interaction with external entities such as RAM and I/O. In this document, the term "MIPS" refers to the execution rate based on actual RAD750 instruction counts, and not to "marketing" ratings for the processor.

Latencies due to memory throughput and PCI accesses are discussed below. It is important to understand that these latencies apply to the CPU. The memory accesses in question cause a delay in completing the current instruction, leaving the CPU blocked. This is then observed as a reduction in the instruction execution rate, i.e. lower MIPS.

3.1 Cache Performance

The first performance factor stems from the relationship of CPU speed to memory speed. In recent times memory speeds have not come close to keeping up with processor speeds. With the processor running substantially faster than the memory it becomes difficult to achieve the full potential performance of the CPU. This is what led to the inclusion of high-speed cache in commercial CPUs in the 1990s, and why commercial enterprises are currently working hard on faster memory architectures such as RAMBUS and DDR.

Processors may have multiple levels of cache; L1 for the cache first accessed, L2 for the next level and so on. The RAD750 contains only L1 cache due to the unavailability of a space-qualified part for L2 cache.

When the CPU references a memory address, the requested data may be in cache. In this case no RAM access is generated and the access is accomplished in a single CPU cycle. If the data are not in cache then RAM must be accessed. In hardware the RAM is arranged into 4K pages. When a RAM access is done to a new page, there is a 2 cycle overhead to "open" the page. If the RAM access is to the same memory page as the previous RAM access the access does not require the additional 2 cycles.

On the RAD750, the memory is accessed by a 64-bit bus, and has timing of 12-1-1-1 for a fetch from a new RAM page, or 10-1-1-1 for a fetch from an already open RAM page (the more common case). This means that the first 64 bits of a fetch requires ten or twelve 30 nanosecond bus cycles and subsequent words that are part of the same fetch require one cycle each. A memory fetch may be up to 4 64 bit values (32 bytes).

If L1 cache is enabled, then all memory fetches are done in 32-byte increments or in groups of 4 64-bit fetches. Thus from an open page, it requires 13 bus cycles or 390ns to fetch a 32-byte cache line. At the theoretical top speed of 240 MIPS, the CPU could have executed 94 instructions in this time. The CPU needs to hang waiting for the instruction or data fetch, effectively resulting in the execution one instruction instead of 94. Also, it is sometimes necessary to write back modified data from a cache line before fetching the new data (called a cast-out, discussed below). Hence, memory throughput and cache hit rate are critical to CPU performance.

It can be seen from the bus timings above that it is extremely important to have the L1 cache enabled. If the cache is not enabled, all memory fetches are done as single 64-bit operations meaning the 10 or 12 cycle penalty is paid for every fetch. At 10 cycles or 300 nanoseconds per 64-bit fetch, this yields a best-case rate of about 6.7 MIPS, not accounting for data load/store.

For a given a cache hit rate it is possible to compute an instruction execution rate that will be close to experimental results. The PPC750 has separate 32 Kbyte caches for instructions and for data. Therefore, in order to compute a cache hit rate it is necessary to aggregate the instruction cache and data cache miss rates. In the tests we have run a reasonable proportion for memory accesses is 70% instructions and 30% data for a compute-bound task as an overall average. Therefore the overall cache miss rate can be estimated as:

$$\text{Miss rate} = (.7 * \text{I-cache miss}) + (.3 * \text{D-cache miss})$$

As one might expect, this can vary substantially based on the software algorithm and instruction mix being executed. The I/O intensive operations tend to be more like 55% instructions and 45 % data.

The other cache-related factor that can influence cache performance is data cast-outs. On the SFC the L1 cache is normally run in write-back mode in which modified data in the data cache are not written to memory until the cache line is needed for different data. A "cast-out" is said to occur when it is necessary to save modified data before reusing a cache line. In this situation the processor must wait while the modified data are saved before it can fetch the new data. A cache line that contains modified data that have not been written to RAM is said to be "dirty".

The timing for writes from cache to RAM is always 5-1-1-1, i.e. 8 cycles to write a cache line. This is 240 nanoseconds or about 58 instructions.

Below is a graph giving the computed instruction rate for various cache miss rates. This graph is the result of analysis, not measurement but it agrees fairly well with test data.

$$\text{rate} = \text{MIPS} / (c + (ir * m1 + dr * m2) * p1 + (m2 * r2 * p2 * dr))$$

- rate: The actual rate of instruction execution
- MIPS: Advertised (theoretical maximum) CPU MIPS Rate (240).
- c: Instruction load rate out of L1 cache. Assuming 1.0.
- m1: L1 I-cache miss rate
- m2: L1 D-cache miss rate
- ir: Fraction of accesses that are instructions
- dr: Fraction of accesses that are data
- p1: Instruction penalty to load L1 line from RAM (94).
- r2: ratio of dirty cache lines (those that must be written before reloading new data into the cache line).
- p2: Instruction penalty to write L1 line to RAM (58).

The above yields a multi-dimensional space controlled by m1, m2, ir, dr and r2 which is inconvenient to use for estimation purposes. Generally one would like to see a graph giving cache rates versus instruction rates. In the plot below an aggregate miss rate is plotted as the X-axis and MIPS is the Y-axis. For purposes of computing the cast-out penalty only a D cache miss rate of 2.5% has been used as well as a data access fraction of 40% has been used. Plots for cast-out rates of 0%, 50% and 100% (r2) are included. The formula then becomes:

$$\text{rate} = \text{MIPS} / (1.0 + m * 94 + (.025 * r2 * 58 * .4))$$

The D-cache miss rate gives results that are a little pessimistic for low aggregate miss percentage and perhaps optimistic for high miss percentage, but is a good estimate for most applications. The tests run show a cast-out rate of about 85% for I/O tasks and 40% to 80% for compute-

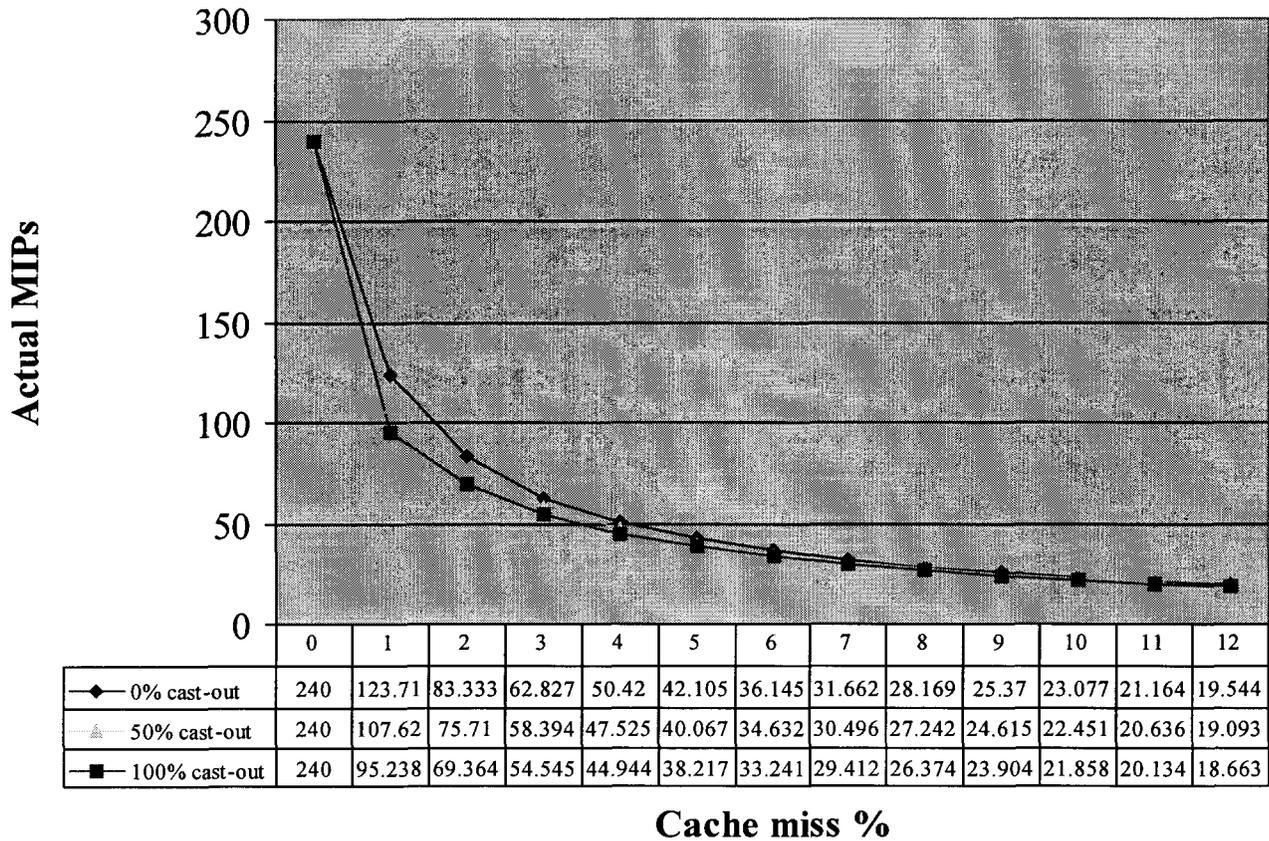
bound tasks. Hence for many cases the higher cast-out

With a cast-out rate of zero, approximately half the instruction execution rate is lost at a 1% miss rate and half the remainder is lost at a 3% miss rate.

The above graph and calculations assume an instruction mix that could execute at the theoretical maximum speed of the CPU, as is the case with most integer instructions. Applications involving primarily floating point instructions

numbers indicated on the graph will be more realistic. will produce substantially lower MIPS numbers since floating point instructions take more time to execute. Specific instruction timings are outside the scope of this document but this effect can be seen in the Gestalt benchmark results below. In this case the difference has nothing to do with the cache and would be similar to any architecture when comparing the performance between floating point and integer operations.

Instruction rate vs Cache miss rate



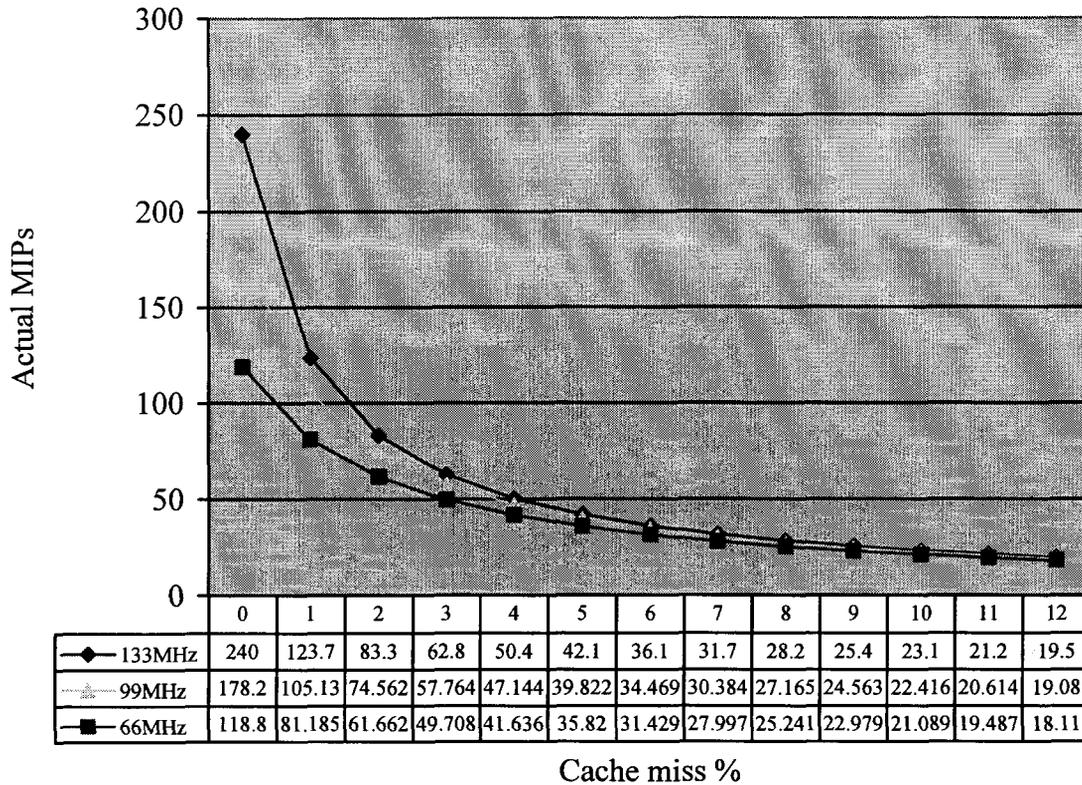
3.2 CPU Clock Rate

By adjusting the divider in the CPU it is possible to achieve various CPU speeds below the maximum without modifying the memory speed. The interesting thing about CPU rate is that, as the CPU rate is reduced, the effect on performance is much greater for cases with good cache performance than for cases with poor cache performance. As the cache performance becomes worse, the contribution of CPU speed

becomes less and the contribution from memory speed becomes greater.

The graph below shows instruction rates for various cache miss rates and CPU clock speeds. It can be seen that as the CPU clock rate goes down the curve flattens out. The numbers below are all calculated for 0% cast-out and the current (not enhanced) SFC.

Cache miss rate vs instruction rate for various CPU speeds



It can be seen that, at a 3% cache miss rate, cutting the CPU speed in half makes only an 18% difference in instruction execution rate. The benefit of higher CPU rates is seen primarily for compute-intensive tasks. The PCI speed does not change with CPU speed so I/O bound tasks will also be less influenced by CPU speed.

Note: One might expect the response to be flat if the processor was running at the same rate as the memory (33MHz). This does not occur due to the 10-1-1 memory fetch timing.

3.3 PCI I/O

The next factor that can have a substantial impact on CPU performance is I/O to the PCI bus. The PCI runs at 33MHz, which is fast for an I/O bus. But like the memory bus there is overhead in initiating a new transaction. Unlike the memory bus the PCI is only 32 bits wide, hence a single PCI transaction transfers 4 bytes.

The overhead for initiating a PCI write is 5 PCI cycles and the overhead for initiating a PCI read is 26 PCI cycles, or 780 nanoseconds. These may seem to be high values but in

fact they are comparable to commercially available PCI bridges.

For both reads and writes the bridge is capable of a continuous burst at one word per cycle after the transaction is initiated, at which point the speed is determined by the speed of the target PCI device.

PCI reads are completely synchronous for the CPU. That is, the CPU must wait for completion before continuing. Therefore any PCI read will cost a minimum of 780 nanoseconds with possibly some extension if the PCI device is not able to respond on the first cycle. The only way to generate a burst of more than one word for a CPU read is to use the CPU's floating point registers. This will generate a two-word burst and can nearly double the throughput.

These effects are not unique to the RAD750 implementation but apply equally to commercial implementations.

PCI writes are more efficient, both because of the lower initiation time and because some pipelining can occur. When floating-point registers are used, bursts of up to four words have been observed on the PCI bus.

The test results show that the instruction execution rate may be less than the expected rate due to cache effects, with this difference explained primarily by PCI I/O latencies.

A PCI read of a single 32-bit value requires a minimum of 27 PCI cycles (810 nanoseconds). If floating-point registers are used, throughput is almost doubled since 64 bits can be read in a minimum of 28 cycles. In reality the target may introduce one or two memory wait-states, but the results observed indicate that throughput does in fact nearly double. A similar principle applies to PCI writes.

The initiation time is the same for PCI transactions of any length. In applications where direct memory access (DMA) can be used for PCI I/O, a major improvement in performance is achieved. This is not only because the CPU load is reduced but also because data may be transferred by PCI burst transactions at a rate very close to the theoretical limit.

3.4 PPC Multiple load / store instructions

Testing was performed to determine if PPC multiple load/store instructions would generate PCI burst transactions and thereby improve throughput. No burst transactions were generated and the performance was equivalent to doing the I/O using single 32-bit instructions.

4 CONCLUSIONS

The full results of the "X2000 Advanced Avionics Characterization Study" (JPL Document D-24447) comprise a report of approximately 140 pages and it is not possible to fully present the information contained to support the conclusions given here.

While a processor will indeed run at the maximum advertised rate under ideal conditions, it to be expected that a real-world application will perform less well. For example it has been shown here that the performance varies substantially depending on how the software interacts with the cache.

Flight software tends to require a high degree of determinism, or at least guaranteed minimum performance. Given the variations in performance due to cache activity, this may become somewhat difficult to achieve. This study shows that algorithm design and associated cache activity makes a much bigger difference in software performance than other factors such as system load. While there may be more performance jitter than in older environments, predicting its extent should still be well within manageable bounds.

The ICER image compression benchmark and the DS1 image compression task clearly show the benefit of designing algorithms that iterate piecewise over small

portions of the data rather than over the entire data set at once. In iterating over an entire image the ICER benchmark achieved a measured performance of ~60 MIPS and it achieved roughly twice that rate by iterating over small image sections. In a performance-critical application substantial benefit can be achieved by designing, when possible, for small locality of reference, completing the processing for a small section of code or data before moving on to further processing. By minimizing I/O and doing larger I/O operations (DMA, floating point register transfers), substantial gains are possible.

In the UART tests a "slipstream" effect was observed in which processing for the second UART in each cycle executes about 20% faster than the first due to the first interrupt having brought the code into cache. Performance is enhanced by executing an algorithm as much as possible for typically short time periods before proceeding on. This benefit is only achieved if an algorithm is small enough to fit mostly or entirely in the cache.

The I/O bound case is simpler to analyze. Given the PCI bus rates (Section **Error! Reference source not found.**), determining the amount of time required by an operation is fairly straightforward. In this case PCI throughput is likely to be the biggest performance factor.

Optimizing RAD750 performance should not be very different than for other processors, except for cache performance being an added and important factor to consider.

4.1 Application performance versus complexity of environment

As discussed earlier, software performance is largely dependent on cache performance. Cache performance in turn is dependent on the range of different memory locations being accessed over time.

The effectiveness of cache is based on the heuristic that memory accesses, either for code or for data, tend to be localized over short time intervals and that memory locations tend to be referenced multiple times over short time periods. At the CPU level there is no distinction between operating system tasks or an application, there is just a series of instructions executing as a stream.

As a result, a given application may have a different performance when run by itself, as compared to being run as part of a system with interrupts and other tasks which intervene and cause flushing of the cache.

Certain restrictions made it impossible to run exactly the same tests in all scenarios, e.g. the requirement for reducing PCI speed in tests involving the 1394 bus. Also, there is a certain amount of noise present in the results due to interrupts from network activity that was necessarily present

during the testing. Therefore, making comparisons between specific tests is difficult.

The NVM write tests show a consistently small reduction in the D-cache miss rate for the interrupt service routine as the data rate increases, however at the same time the difference in MIPS is small. On the other hand the NVM read tests do not exhibit a similar effect.

The Star ID test showed a fairly consistent performance and should not be influenced by the PCI speed. This test showed a 4.6% reduction in the MIPS rate when run as part of the full-rate combination test as compared to running standalone. A small but consistent reduction in performance is observed as system load increases. In the table "I" is instruction cache miss rate and "D" is data cache miss rate.

Star ID test

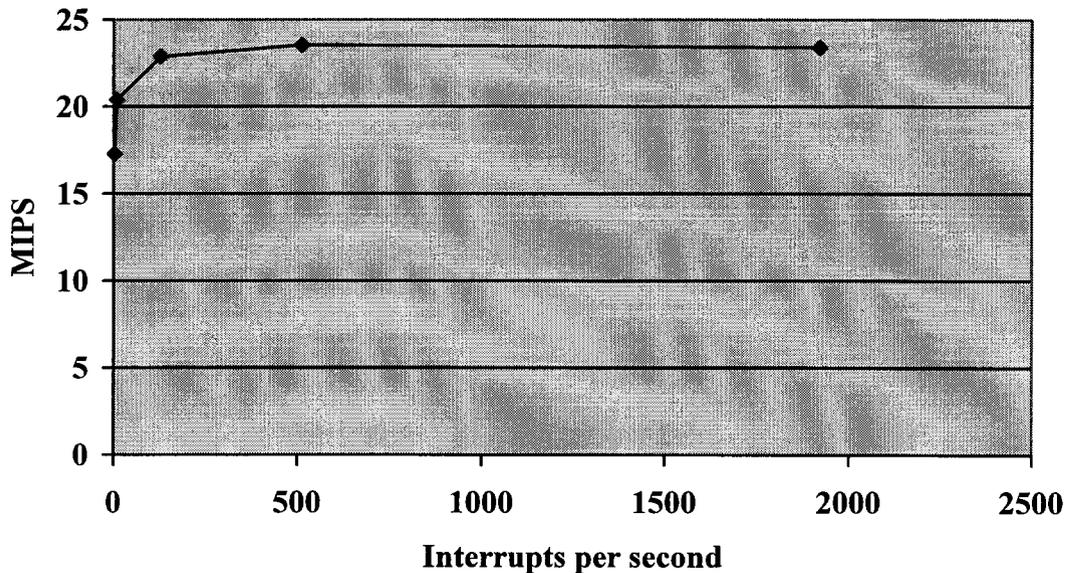
Stand-alone	Half-rate combination	Full-rate combination
128.6MIPS	125.5MIPS	122.7MIPS
I - 0.01	I - 0.01	I - 0.02
D - 2.30	D - 2.44	D - 2.52

The 1394 bus isochronous loopback tests do indicate an interaction between the performance values for the interrupt service routine (ISR), as shown in the following table. As the number of interrupts per second increases, the ISR MIPS increase by about 26%, while the I-cache miss rate decreases.

1394 isochronous loopback tests

Interrupts / sec	MIPS	I-Cache miss %	D-Cache miss %
3	17.3	4.2	3.8
9	20.4	2.5	3.4
129	22.9	1.1	3.0
513	23.4	0.9	3.0
1921	23.4	0.9	3.0

MIPS vs Interrupt Rate, 1394 ISR



In the 1394 tests, most of the processing occurred in the ISR and there is not a lot of system activity (primarily network) during the tests that would flush the ISR out of cache. This effect is not seen as much in a more active system.

There are no other results that indicate a major impact on the performance of applications in a realistic scenario based on interrupts or other system activity.

If performance of an application is measured on an otherwise completely inactive system there will be degradation due to cache interaction in the range of 10% to 15% as compared to an active system. Most of this degradation will

occur quickly as the system becomes more active. The performance interactions are observed to be greater for applications that exhibit higher cache miss rates. This would be expected since a low cache miss— rate application will do more processing between interrupts. Hence the influence of interruptions on cache performance will be a smaller percent.

While tuning a system to reduce the context switches and interrupt activity may be done, the results here indicate that designing algorithms for locality of access will generally be more productive. If code design and implementation cannot be modified, overall it is desirable to arrange the system activity to allow applications to run as long as possible without interruption.

4.2 *PCI bus contention*

Contention is another factor that reduces performance as system activity increases, in particular contention for the PCI bus. This effect can be seen most clearly in the NVM read tests when running standalone as compared to running in combination with other components. The NVM high rate tests experienced approximately a 30% decrease in performance when run with the 1394 bus contending for the PCI. Less than half of this degradation can be explained by the difference in cache miss. The remainder (~20%) is the result of the contention with the 1394 bus in accessing the PCI bus.

For any application that uses program I/O, this contention will be seen as an increase in CPU time consumed since the CPU must wait for the operation to complete. For an application that uses DMA to transfer data this contention will not be reflected in the CPU time but will show in the maximum data rates achievable.

4.3 *MIPS and processor percent*

It has been common in designing systems to allocate to software applications (such as attitude control, science, etc.) portions of the processing capability based on the MIPS rating of the processor. It is clear that this approach is not

valid for the RAD750 architecture since the MIPS achieved varies widely depending on cache activity.

MIPS became an allocation unit because it was a convenient to use a single number. This MIPS rating was for a particular instruction mix, which was not always well understood or considered. For marketing reasons, it would invariably represent “best case”. But even in older non-cached architectures it was not a particularly useful measure. The performance of these systems was also affected by the extent of I/O processing. Allocating MIPS to applications usually did not take into account the instruction mix, e.g. the different execution times of integer and floating-point operations.

Given these failings, the percentage of the processor time taken by a given task is a more meaningful allocation measure. This is what was actually being allocated before, in which MIPS allocations needed to be scaled based on task characteristics such as floating point usage and I/O activity.

The performance data presented in this document is intended as an aid in predicting how a specific type of application will perform on the RAD750 architecture. Unfortunately, scaling performance estimates for processors that follow will be rather approximate. It will still be necessary to run tests to characterize new platforms since performance does not typically scale equally for all types of operations between platforms. For instance, it is likely to be true for some time that memory subsystems’ performance will vary widely, a factor easily overlooked in a manufacturer’s specifications.

It is hoped that this document will provide information sufficient to aid in predicting the performance of applications when hosted on the RAD750 / X2000 platform, and form the basis for comparisons with other platforms.

REFERENCES

- [1] Len Day, et al, JPL Internal Document D-24447, “X2000 Advanced Avionics Project Performance Characterization Study”