Part II
Continuous Real-Time Signal Processing

Comparing Continuous cFFTs on the TigerSHARC and PowerPC

By Jeffry Milrod, BittWare and Chuck Millet, Analog Devices

November 2003

This is the second in a series of articles that explores the complex processor tradeoffs and evaluations required to choose the most effective processor for continuous real-time signal processing applications as typified by the 1024-pt complex Fast Fourier Transform (cFFT). Part I appeared in the December 2002 issue of COTS Journal, and is available on their web site (www.cotsjournalonline.com).

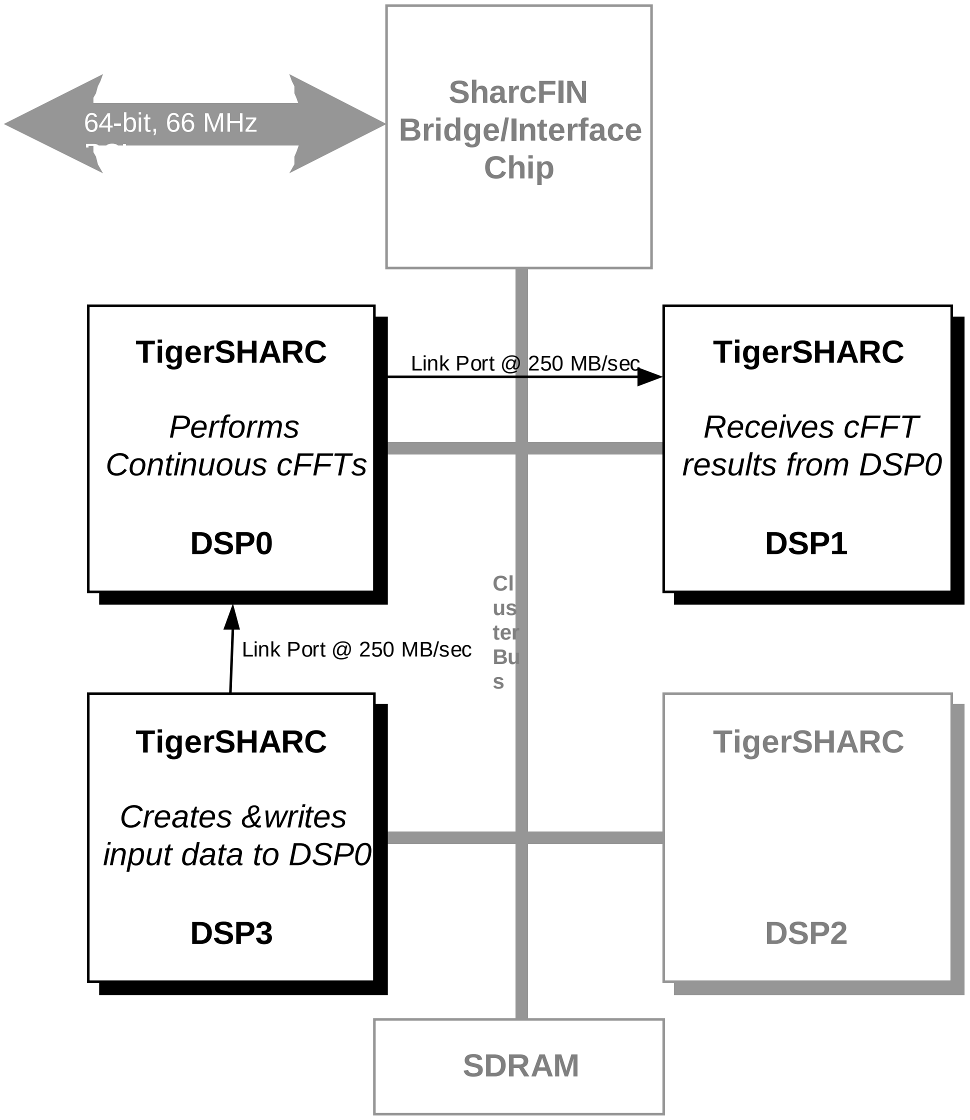
The previous installment of this series provided detailed overviews of the TigerSHARC (ADSP-TS101) and the G4 PowerPC with Altivec (MPC7410 & MPC7455) processors, along with a discussion of the importance of I/O bandwidth and the bandwidth-to-processing ratio (BPR) in real-time signal processing applications. It was shown that oft quoted peak performance benchmarks can be misleading when used to infer real-world performance, and that measuring sustained, or continuous, algorithm performance is much more indicative and informative regarding the comparison of these processors. Finally, it was proposed that the continuous 1024-pt cFFT algorithm is an excellent indicator of real-world performance, and predictions were made for the performance of these processors, both at the processor and board level.

For this article, continuous 1024-pt cFFT algorithm benchmarks were actually coded, optimized, and measured on the ADSP-TS101 at 250 MHz and on the MPC7410 at 400 MHz. Unlike some other "system-level" benchmarks, these implementations of the continuous cFFT benchmarks took advantage of all the features of each processor that could improve performance - since this approach is more like what would be implemented by an engineer designing a real system. The results are reported in detail, along with extrapolations for other variants and speeds, as well as board-level implications.

1. TigerSHARC Implementation

The continuous cFFT algorithm was implemented on a BittWare Tiger-PCI board (TSPC) that features a cluster of four (4) ADSP-TS101 TigerSHARCs running at 250 MHz. The benchmark code was written in C and was developed using Analog Devices' VisualDSP++; it was loaded on to the board using BittWare's BittWorks toolkit.

Figure 1: TigerSHARC Continuous cFFT Data Flow on TSPC Board



As was shown in Part I, it is expected that running continuous cFFTs on the TigerSHARC will be processor limited, implying that the continuous benchmark performance will be driven by the performance of the cFFT algorithm itself. It is, therefore, critical that a highly optimized cFFT routine be used for the benchmark. To that end, the benchmark implementation calls the 1024-pt cFFT function from the EZ-DSP TSlibs function library, which is handcrafted and fully optimized in assembly language.

The TigerSHARC boasts link ports which, as discussed in detail in Part I, provide a unique and balanced data movement path, and this is the obvious way to move data on and off the processor. Therefore, as shown in Figure 1, a single TigerSHARC (DSP0) was used to continuously perform cFFTs. Two additional TigerSHARC were used as test instrumentation to generate the input data (DSP3) and receive the results (DSP1) via link ports. Note that the link ports could also be used to move data from/to other I/O devices such as FPGAs, using the other TigerSHARCs on the TSPC board was simply a convenience.

Since the TigerSHARC can support I/O DMAs from/to internal memory in background, dual input and output buffers were used to implement a ping-pong scheme shown graphically in Figure 2.

While the cFFT routine is processing data from input buffer A and writing the results to output buffer A, the DMA engines are simultaneously moving the data for the next cFFT in to input buffer B from a link port while the results from the previous cFFT are being written out the other link port from output buffer B. After the cFFTs and both DMAs complete, the ping-pong buffers are swapped; the cFFT now processes out of/into buffer B while the DMAs operate on buffer A.

Using this ping-pong scheme, the internal memory of the TigerSHARC must hold the benchmark code, as well as dual input and output buffers. It was verified that how the buffers are placed in memory could dramatically impact the benchmark performance. As shown in Figure 3, the TigerSHARC's internal memory consists of three (3) banks of 2 Mbits each. Bank 1 is used for the program code (with plenty of room to spare). Even though both data buffers would easily fit in to a single bank, it was found that for optimal performance each input/output buffer set needed to be placed in a separate bank; i.e. buffers A placed in Bank 2, and buffers B placed in Bank 3.

Figure 2: Ping-Pong Buffer Scheme for TigerSHARC Implementation

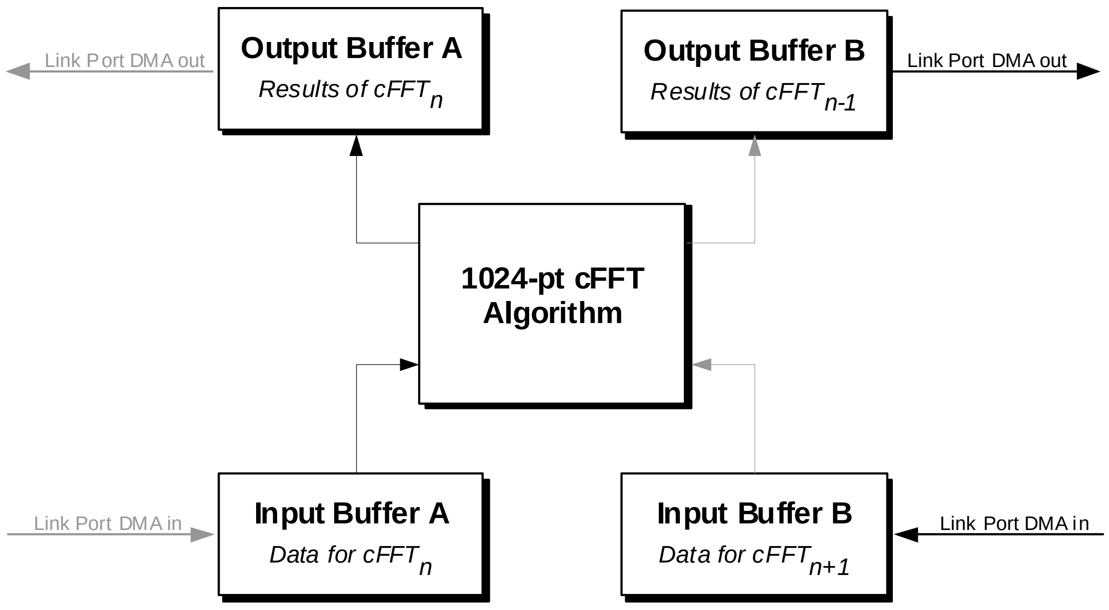
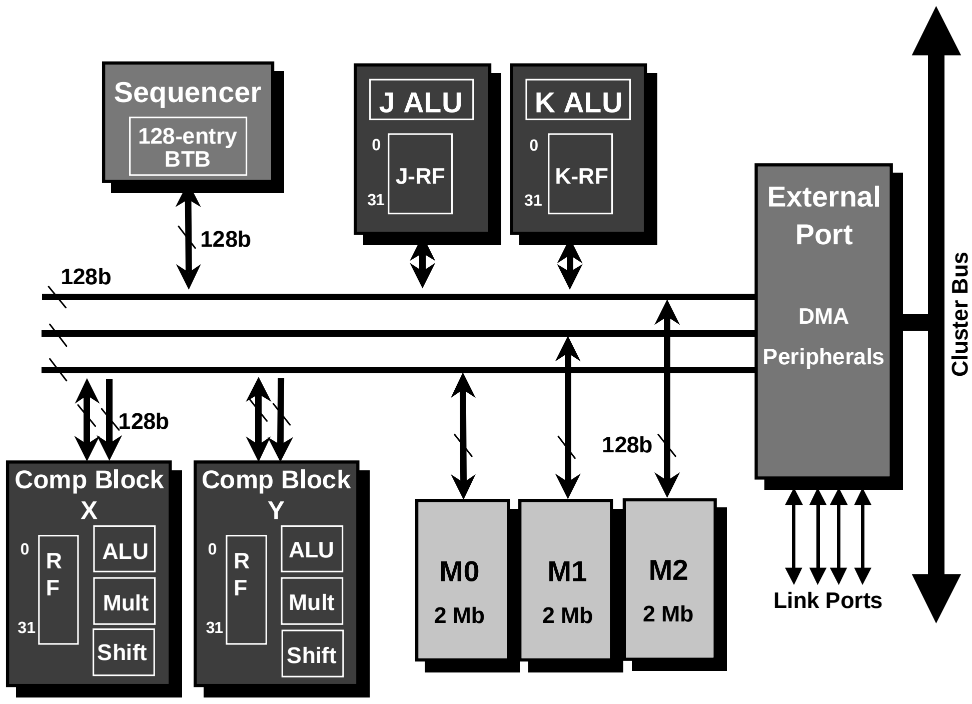


Figure 3 Figure 3: TigerSHARC Block Diagram



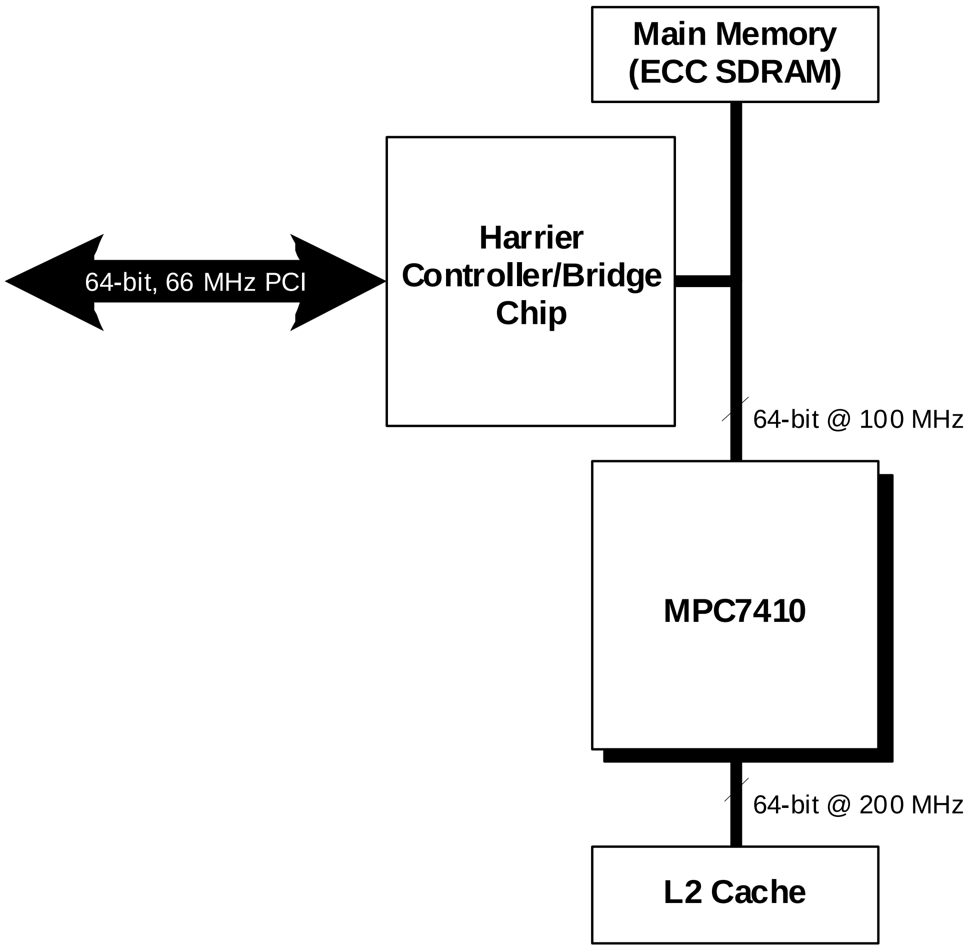
2. PowerPC Implementation

The continuous cFFT algorithm was implemented on a Motorola PrPMC800 PMC that has a single MPC7410 running at 400 MHz with a 100 MHz front-side bus; the PMC board was placed on a PCI carrier card for ease of use. The benchmark code was written in C and assembly, and was compiled using the GNU C compiler version 2.96; it was loaded on to the board using the PPCBUG monitor via a serial port.

As was shown in Part I, it is expected that running continuous cFFTs on the G4 PowerPC with AltiVec will be bandwidth limited, implying that the continuous benchmark performance will be limited by I/O, not the performance of the cFFT algorithm itself. Therefore, the optimization of the cFFT routine used for the benchmark is not critical since it does not drive the performance. A public domain 1024-pt cFFT algorithm implementation was used that offered good, but not stellar performance.

As with most PowerPC boards, the primary mechanism for data movement on the PrPMC800 is through the external node controller and PCI bridge. The PrPMC800 uses Motorola's Harrier chip that features a 64-bit, 66 MHz PCI interface, as shown in Figure 4. Since the cFFT algorithm operates “in-place” (i.e. the input data buffer is overwritten with the results), only two buffers (one I/O, one for cFFT) is required to implement a ping-pong scheme. These buffers were placed in the main memory. While a cFFT is performed on buffer A, data output from the previous cFFT calculation is written out via a DMA from buffer B to the PCI bus; data for the next cFFT calculation is then read in via a DMA from the PCI back into buffer B. As with the TigerSHARC implementation, the buffers simply switch when the whole process is complete. The two DMAs were combined using a chained DMA, which allows a sequence of DMA operations to be combined into a single operation using a linked list of DMA descriptors

Figure 4: PrPMC800 PowerPC Board Block Diagram



Most programmers use an operating system when programming the PowerPC due to the complexity and difficulty of creating stand-alone programs. However, since the overhead of an operating system could reduce the benchmark performance, the implementation configured the PowerPC manually, including managing the cache, MMU, and DMA engine on the external node controller.

Since the PowerPC is bandwidth limited and uses cache to speed access to data, cache management was considered critical in the algorithm implementation. Note that the cache is not addressable memory, it is merely a fast buffer holding a copy of portions of the memory and is controlled by the cache manager. Once a memory address range (in this case, an I/O buffer) has been accessed by the processor, it will be "cached" for quicker subsequent accesses. Therefore, input data that is read in from the PCI bus into memory is not necessarily the same data that the processor sees when it attempts to access the data – because the PowerPC will look to the copy of data that is stored in cache rather than the copy that is in memory. One way to eliminate this problem is to enable "snooping", which causes the hardware to keep the contents of cache and main memory coherent. But because snooping can severely hinder performance, the cache was managed manually by 'invalidating' the address range containing the data received via DMA.

An "invalidate" operation forces the cache to reload the data in the invalidated address range from memory the next time it is accessed, causing the data in cache to be consistent with the data in memory. Likewise, when the cFFT writes its results into an output buffer, this buffer is located in cache rather than in memory. It must then be 'flushed' from cache to memory before a DMA is performed;

otherwise the DMA engine will move stale data located in memory rather than the results of the calculations that are stored in cache.

3. Discussion of Results

Table 1 shows the specifications of the processors, and the predictions made for the performance of the continuous 1024-pt cFFT implementation on the TigerSHARC and the PowerPC from Part I; Table 2 shows the actual results of the benchmarks tests. While neither processor did as well as predicted, the TigerSHARC dramatically outperformed the PowerPC.

Table 1: Processor Specifications and Performance Predictions from Part I

	TigerSHARC	**PowerPC**	
Parameter	ADSP-TS101S	MPC7410	MPC7455
Core Clock	250 MHz	500 MHz	1,000 MHz
Peak Floating-pt Performance	1,500 MFLOPS	4,000 MFLOPS	8,000 MFLOPS
Memory Bus Size/Speed	64-bit/100 MHz	64-bit/125 MHz	64-bit/133 MHz
External Link Ports	4@250 MB/Sec	None	None
I/O Bandwidth (inc. memory)	1,800 MB/Sec	1,000 MB/Sec	1,064 MB/sec
Bandwidth-to-Processing Ratio	1.20 B/FLOP	0.25 B/FLOP	0.13 B/FLOP
1024-pt cFFT Benchmark	39 µsec	22 µsec	13 µsec (est.)
Approx Cycles for 1024-pt cFFT	9,750 cycles	11,000 cycles	13,000 cycles
Predicted 1024-pt cFFTs/chip	25,641 per Sec	26,053* per Sec	64,941* per Sec

*. Assumes 100% of peak I/O used for continuous cFFTs, and neglects cache & data movement overheads due to inability to predict - real-world performance could be much less.

Table 2: Results of Continuous 1024-pt cFFT Benchmark Implementation

Parameter	**TigerSHARC ADSP-TS101S**	**PowerPC MPC7410**
Core Clock	250 MHz	400 MHz
Actual 1024-pt cFFTs/chip	22,923 per sec	7,899 per sec

Further examination of the TigerSHARC results and implementations revealed that the prediction made in Part I neglected to allow for any DMA management overhead. Setting up the link port DMAs, handling the DMA done interrupts, and checking for DMA completion adds an overhead of approximately 10% that accounts for the performance difference. For the sake of convenience the DMA management code was written in straightforward C, and since the improvement of the overall benchmark would be small (10% at best), no attempt was made to minimize this overhead.

Evidently, the PowerPC overhead is much more significant. Although the benchmark tests were only performed on a 400 MHz processor, the results are approximately a factor of three (3) less than predicted. This was somewhat anticipated, however, and gave rise to the asterisked note that the effects of cache and data movement overheads were neglected and the associated warning that real-world performance could be much worse. This concern proved warranted.

To ensure accurate and optimized results, several variants of the full benchmark implementation were run on the PowerPC. As expected, with the cache disabled, performance decreased by about an order of magnitude. Benchmarks were also run with the cFFT disabled, and it was discovered that just moving the data in and out, as well as managing the cache and MMU, resulted in no performance improvement at all! This clearly indicates that even if a considerably faster cFFT algorithm were used, there would be no performance improvement of the continuous cFFT benchmark. Similarly, eliminating the chained DMAs and forcing all data movement over PCI to be writes could possibly result in a small, but relatively insignificant improvement.

More complicated yet, is the possible impact of an operating system to manage the cache and MMU. It is assumed that the overhead would be increased, however, this may be a false assumption - it is possible that commercial operating systems could provide an improvement over this manual implementation, but it seems unlikely due to the inefficiencies and cache overhead associated with the OS services and context switches.

Despite best efforts, this benchmark implementation on the PowerPC might be proven to be non-optimal and thus not truly indicative of the real-time signal processing capabilities of the PowerPC. However, unless and until a greatly improved implementation is provided, it must be concluded that the TigerSHARC is far superior at processing continuous cFFTs and is, therefore, a better real-time signal processor.

4. Extrapolations and Board Level Implementations

Since it is processor limited, the TigerSHARC benchmark results should scale linearly with processor speed as long as the I/O bandwidth remains balanced (i.e. the link port bandwidth also increases linearly). Thus extrapolation for the faster ADSP-TS201S is straightforward.

This is not the case for the PowerPC, as this processor is bandwidth limited. Further, the cache management issues dominate the performance of continuous cFFTs. Thus, the speed of the front-side memory bus is more indicative of continuous cFFT performance than processor speed. Increasing processor speed of a MPC7410 from 400 MHz to 500 MHz, without increasing the speed of the front-side memory bus, will have little or no effect on the benchmark. Oddly, the speed of the front-side memory bus is typically not specified on COTS boards, but in practice, all seem to run at 100 MHz (possibly due to the external controller/bridges), as did the PrPMC800 used for the testing.

Since other PowerPC versions used on COTS boards (MPC7455, MPC7447) have different data movement and cache schemes than the MPC7410, it's difficult to directly infer their performance from these results. However, it seems reasonable to infer that the performance of these processors and boards will be similarly limited by the cache and front-side memory bus speed in addition to the bandwidth limitations discussed in Part I.

As discussed in Part I, multi-processor COTS boards are readily available with TigerSHARCs and PowerPCs. The original board-level predictions accounted for the I/O bandwidth scalability of the TigerSHARC, and further bandwidth limits imposed for the PowerPC boards. However, the reduced processor benchmark performance of the PowerPC indicates that the boards should no longer further limit the I/O required to keep all the PowerPCs feed when running continuous cFFTs. Therefore, as shown in Table 3, the board level performance is projected to simply be the number of continuous cFFTs per processor multiplied by the number of processors per board. It was originally projected that an Octal TigerSHARC board (ADSP-TS101S @ 250 MHz) could perform three times as many continuous cFFTs as a quad PowerPC board (MPC7410 @ 500 MHz). The results of this study indicate that the TigerSHARC board will actually outperform the PowerPC board by more than seven times, and that the new TigerSHARC (ADSP-TS201S) will outperform the PowerPC boards by a factor of 15!

Table 3: Board-Level Implications and Extrapolations

	TigerSHARC		PowerPC
Parameter	ADSP-TS101S	ADSP-TS201S	MPC7410
Core Clock	250 MHz	500 MHz	500 MHz
Typical # Processors/Board	8	8	4
Peak FLOPS/Board	12 GFLOPS	24 GFLOPS	16 GFLOPS
Memory Bus Size/Speed	64-bit/83.3 MHz	64-bit/100 MHz	64-bit/100 MHz
Typical Off-board I/O	2 PMC + 16 Links	2 PMC + 16 Links	2 PMCs
Peak Off-board I/O (not back-plane)	5,056 MB/Sec	9,056 MB/Sec	1,056 MB/Sec
Bandwidth-to-Processing Ratio	0.42 B/FLOP	0.38 B/FLOP	0.07 B/FLOP
Projected 1024-pt cFFT/Board	180,000* per Sec	360,000* per Sec	24,000* per Sec

*. Estimated based on previous results

5. Original Conclusions Supported

The benchmark implementations and testing supported the conclusions from Part I that the TigerSHARC is a superior real-time signal processor. In fact, the results make them even more emphatic, such that they bear repeating:

"If the application requires a lot of number crunching with little data movement, typical of so-called back-end data processing, then the PowerPC's higher clock rate and more powerful core will probably be more effective. For continuous real-time signal processing such as imaging, radar, sonar, sigint, and other applications that require high data flow or throughput, however, the TigerSHARC can dramatically outperform the PowerPC and is probably the preferred choice."

Clearly these processors are designed and optimized for different applications. In the case of the TigerSHARC, virtually all data movement is done in background and the performance is driven by algorithmic speed; in the case of the PowerPC, literally all of the processing is done in background for this application, and the performance was driven by I/O and cache overhead.