Cahners
MICROPROCESSOR REPORT
www.MPRonline.com
THE INSIDER'S GUIDE TO MICROPROCESSOR HARDWARE

EEMBC 1.0 Scores, Part 1: Observations

Single-Number Aggregate Scoring Doesn't Tell the Story

By Markus Levy {8/14/00-01}

More is better. More power savings. More processor choices. More competition. And more benchmark scores. And that's my opinion when it comes to the wide variety of EEMBC (EDN Embedded Microprocessor Benchmark Consortium) benchmarks. (EEMBC is pronounced "embassy.") At first glance, the variety and abundance of EEMBC benchmark scores are bewildering. There are 46 different benchmarks in EEMBC's Version 1.0 benchmark suites, so processor performance and code-size data can quickly get out of hand. But there is a way to get a handle on this situation and use all the data to analyze the real architectural capabilities of processors—in fact, the more data the better.

Part 1 of this article presents an approach for analyzing EEMBC benchmark data. It's not rocket science, but it's effective. Using this approach, Part 1 also lists the observations we were able to make. Table 1 lists the processors and vendors tested for this article. We've presented the respective processor vendors with these observations. In Part 2 of this article, which will appear in one month, we will provide the vendors' explanations for the performance results, as well as our own insight into the processor architectures and the capabilities of the associated compilers, as illuminated by the EEMBC benchmarks. Analyzing the data will be a challenge, simply because the interaction among hardware architecture, memory systems architecture, compilers, and the benchmark code itself is highly nonlinear and interdependent.

Aggregation Is a Matter of Convenience

Many would like to categorize the performance of an embedded processor with a single number. But in the words of Tom Halfhill (see MPR 5/1/00-02, "EEMBC Releases First Benchmarks"), "EEMBC doesn't attempt to distill the raw data into a composite score that yields an easy-to-digest single figure of merit." A single number can hide the distinct advantages of processors; furthermore, it blurs the application-level orientation of the EEMBC benchmarks. However, Halfhill also points out that the value of score aggregation is that it reveals high-level relationships among processors. An aggregation, or average score, is also useful for pointing out individual benchmark scores that deviate from the norm.

MDR derives its unofficial "EEMBCmarks," which provide a first order of approximation, by normalizing all the test results to the slowest processor that participated in all the benchmark suites. Results of this process indicate, for example, that NEC's VR5000 is 38% faster than NEC's VR5432 (for the EEMBC automotive/industrial benchmarks). Another example indicates that, for the consumer benchmarks, National Semiconductor's Geode processor is 12% faster than NEC's V832. (All benchmark scores and datasheets are available on EEMBC's Web site at www.eembc.org/benchmark.)

Inspection Highlights Anomalies and Patterns

What the aggregate score doesn't reveal in the example above is that the VR5000 and VR5432 differ by only 1% on the bit-manipulation benchmark but by 60% on the matrix-arithmetic benchmark. Similarly, it doesn't reveal that the Geode is 78% faster than the V832 on the JPEG-compression benchmark but 45% slower on the RGB-to-YIQ conversion benchmark.

Unless you have more time than you know what to do with, you can't analyze every benchmark score. But if you look for the anomalies, as mentioned above, they will help you

Processor Name	Clock Rate (MHz)	Compiler	Native Data Type	FPU	I/D Cache	I/D Cache Type	L2 Cache Clock	Bus Width	Bus Freq. (MHz)
AMD ElanSC520	133	CAD-UL I381G1X0	32	Yes	16K/16K	2-way SA	n/a	32	66
AMD K6-2	450	CAD-UL I381G1X0	32	Yes	32K/32K	2-way SA	66	32	66
Analog Devices 21065L	60	Analog Devices cc21k V6.0	32	No	500K	On-chip RAM	n/a	32	60
IBM PowerPC 750CX	500	Wind River Diab 4.3b	32	Yes	32K/32K	8-way SA	500	64	66
IDT79RC32364	100	IDT/c 7.2.1 Gnu	32	No	8K/2K	2-way SA	n/a	32	50
IDT79RC64575	250	IDT/c 7.2.1 Gnu	64	Yes	32K/32K	2-way SA	n/a	64	50
Infineon Tricore TC10GP	80	Tasking V1.1r1	32	No	16K/16K	2-way SA	n/a	32	80
Mitsubishi M16C/62A	16	Mitsubishi, NC30, V3.20 R.1	16	No	n/a	n/a	n/a	16	16
Mitsubishi M16C/80	20	Mitsubishi NC308WAV2.00 R.1	16	No	n/a	n/a	n/a	16	20
Mitsubishi M32R/E	40	Wind River Diab Rel4.3 Rev f	32	No	40K	On-chip RAM	n/a	32	40
National CompactRISC CR16B	50	National CRCC (CR16 Dev Toolset V2.2)	16	No	n/a	n/a	n/a	16	50
National Geode GX-1	200	Microsoft MSVC 6.0	32	Yes	16K Unified	4-way SA	n/a	64	66
NEC V832	143	Green Hills V1.8.9	32	No	4K/4K (write-through)	Direct-mapped	n/a	32	47.6
NEC VR5000	250	Green Hills Multi(r)2000 V2.0	64	Yes	32K/32K	2-way SA	100	64	100
NEC VR5432	167	Apogee Software V4.1	64	Yes	32K/32K	2-way SA	100	64	100
TI TMS320C6203	300	TI C6000 Codegen Tools V4.0	16	No	875K	On-chip RAM	n/a	32	NA
Toshiba TMP95FY64F	20	Toshiba Build Manager V1.30	16	No	n/a	n/a	n/a	16	20
Toshiba TMPR3927	133	Green Hills Multi2000 V2.0	32	No	8K/4K	2-way SA	n/a	32	66

Table 1. Comparison of the basic features and compilers of the processors discussed in this article. (n/a = not available)

uncover interesting architectural differences among processors. Spreadsheets are useful for comparing the benchmark scores among a given group of processors with those of every other processor to uncover the irregularities. For example, start by comparing the scores of processor A with those of processor B, processor C, and processor D. The next step is to compare the scores of processor B with those of processor C and processor D, and so on. Obviously, more processors imply more combinations of comparisons, but some of the comparisons will be meaningless. In other words, it doesn't make sense to compare IBM's 500 MHz PowerPC 750CX with Mitsubishi's 20MHz, 16-bit M16C/80.

Once you've set up the spreadsheets, it's easy to pick out the anomalies, such as the matrix-arithmetic benchmark example for the VR5000 and VR5432. Table 2 shows the results of comparing a variety of processors with the VR5000. The numbers in the table are not the absolute scores but score ratios relative to the VR5000. For example, the K6-2 has a score of 0.65 for the floating-point benchmark, indicating that it runs this benchmark 35% slower than the VR5000 does. Unless indicated otherwise, all relative scores presented in this article are based on "out-of-the-box"

Benchmark Scores Relative to the NEC VR5000 (Processor X/VR5000)					
	AMD ElanSC520	AMD K6-2	IBM 750CX	NEC VR5432	NEC V832
Angle-to-Time Conversion	0.27	1.88	3.17	0.68	0.11
Floating-Point	0.09	0.65	1.66	0.43	0.02
Bit Manipulation	0.23	1.36	2.63	1.01	0.24
Cache Buster	0.25	1.04	3.49	0.80	0.23
CAN Remote Data Request	0.20	0.75	3.94	0.76	0.30
FFT	0.17	1.38	3.52	0.86	0.16
FIR	0.18	1.61	3.79	0.96	0.41
IIR	0.32	1.88	3.43	0.58	0.35
IDCT	0.11	0.89	2.82	0.66	0.19
IFFT	0.18	1.43	3.83	0.87	0.16
Matrix Arithmetic	0.06	0.41	2.22	0.40	0.02
Pointer Chasing	0.22	1.93	3.14	0.77	0.23
PWM	0.19	1.72	3.05	0.93	0.25
Road-Speed Calculation	0.23	1.76	3.22	0.78	0.35
Table Lookup & Interpolation	0.19	1.39	3.18	0.85	0.07
Tooth-to-Spark	0.23	1.65	3.56	0.79	0.33

Table 2. Relative comparison of five processors with NEC's VR5000 in the EEMBC automotive/industrial benchmarks shows wide performance variations among both similar and dissimilar architectures.

(unoptimized) results. Unoptimized scores allow vendors to use the compiler and compiler switches they want, but the benchmark source code cannot be altered. Use of unoptimized code in the out-of-the-box benchmarks implies that the processors may not be operating at their peak performance.

Another advantage of these relative comparisons is that they show performance trends, or patterns, and help eliminate any false assumptions that may arise by normalizing for operating frequency. For example, Table 2 shows that the PowerPC is roughly three times faster than the VR5000 for most of the benchmarks. One would expect some of the performance difference to be purely the result of the 750CX's running at twice the frequency of the VR5000, in combination with its 256KB L2 cache. But what architecture, system-level, and compiler factors should also be considered?

Benchmark Scores Relative to the AMD ElanSC520 (Processor X/ElanSC520)				
	IBM 750CX	Infineon TC10GP	NEC VR5432	NEC V832
Angle-to-Time Conversion	11.92	0.94	2.54	0.40
Floating-Point	19.29	0.49	5.07	0.22
Bit Manipulation	11.43	1.59	4.39	1.03
Cache Buster	14.08	1.01	3.24	0.92
CAN Remote Data Request	19.82	1.55	3.81	1.49
FFT	20.17	1.89	4.92	0.90
FIR	20.89	1.50	5.31	2.25
IIR	10.63	1.49	1.80	1.09
IDCT	25.95	1.59	6.09	1.78
IFFT	21.69	1.95	4.92	0.93
Matrix Arithmetic	39.28	0.65	7.08	0.34
Pointer Chasing	14.04	1.08	3.43	1.04
PWM	15.95	1.20	4.86	1.29
Road-Speed Calculation	13.92	1.41	3.37	1.49
Table Lookup & Interpolation	16.72	1.23	4.47	0.37
Tooth-to-Spark	15.15	1.86	3.38	1.39

Table 3. A comparison of automotive/industrial benchmark performance between AMD's ElanSC520 and a variety of other processors illustrates the VR5000's superior floating-point capability, even when its integer performance is less.

The Automotive/Industrial Benchmarks

The EEMBC automotive/industrial benchmarks range from integer-math to floating-point computations to bit manipulation. The names of many of these benchmarks provide clues to their fundamental operation. For most of these benchmarks, NEC's VR5000 outperforms the AMD ElanSC520 by a factor of four to five, as shown in Tables 2 and 3. The difference in clock frequency accounts for some of this difference. Another factor is that the VR5000 is a superscalar processor. But even though both processors have hardware floating-point support, the VR5000 is eleven times faster than the Elan in the floating-point benchmark.

When we compare the VR5000 with the K6-2, we find that the AMD processor is 40–90% faster for most items in the benchmark suite. However, for the CAN remote data request and IDCT, benchmarks are 25% and 11% slower; for the floating-point and matrix-arithmetic benchmarks, the AMD is 35% and 59% slower. Another indicator of the VR5000's strong floating-point capability appears when we compare this processor with IBM's PowerPC 750CX. Operating at twice the frequency of the VR5000, the IBM processor averages three times faster on the benchmarks in this suite, except for the floating-point and matrix-arithmetic benchmarks, where it is only 1.66 and 2.22 times faster, respectively.

When we pit the NEC VR5000 against the VR5432, the performance scores average 40% faster for the VR5000, which you might expect from the 50% higher clock rate. But notice that for the floating-point and matrix-math benchmarks, the VR5000 is 2.3 and 2.5 times faster, respectively. The VR5000's floating-point capability is simply much better than the VR5432's. Yet the winning streak ends on the bit-manipulation benchmark, where the VR5432 comes out ahead. It's possible that the compiler is unable to take advantage of the VR5000's dual pipelines for this benchmark.

Table 4 also shows results from the 16 automotive/industrial benchmarks, but the scores are relative to Infineon's TriCore. In many of the automotive/industrial benchmarks, the scores indicate that Infineon's TriCore makes a good stand against the ElanSC520 (see Table 3 or 4). Of course, that's what TriCore's automotive focus might lead us to expect. TriCore's superscalar architecture, albeit with asymmetrical pipelines, seems to give it an average 34% advantage over the ElanSC520 for many of these benchmarks (TriCore also has a 21% faster memory bus). But the two processors practically tied on the

Benchmark Scores Relative to Infineon TC10GP (Processor X/TC10GP)			
	AMD ElanSC520	Mitsubishi M32R/E	NEC V832
Angle-to-Time Conversion	1.07	0.13	0.42
Floating-Point	2.03	0.05	0.45
Bit Manipulation	0.63	0.36	0.65
Cache Buster	0.99	NA	0.91
CAN Remote Data Request	0.65	0.32	0.96
FFT	0.53	0.30	0.47
FIR	0.67	0.52	1.50
IIR	0.67	0.24	0.73
IDCT	0.63	0.62	1.12
IFFT	0.51	0.32	0.48
Matrix Arithmetic	1.54	0.13	0.52
Pointer Chasing	0.92	0.37	0.96
PWM	0.83	0.59	1.07
Road Speed Calculation	0.71	0.35	1.06
Table Lookup & Interpolation	0.81	0.13	0.30
Tooth-to-Spark	0.54	0.32	0.75

Table 4. Infineon's TriCore, with its superscalar architecture, stands up well to other processors with higher clock rates in EEMBC's automotive/industrial benchmarks.

angle-to-time conversion and cache-buster benchmarks. Furthermore, the Elan beat the TriCore by only a factor of two on the floating-point benchmark, yet the TriCore processor performs floating-point operations in software.

TriCore also beats the NEC V832 by an average of 23%, except for the FIR filter and IDCT benchmarks, where the V832 is faster by 50% and 12%, respectively.

Infineon's TriCore runs the automotive benchmarks an average of 2.8 times faster than Mitsubishi's M32R/E. This is in line with what we'd expect on the basis of operating frequency. But again, the floating-point benchmark is an exception. TriCore runs it 20 times faster than the M32R/E. Similarly, when the M32R/E is compared with NEC's V832, the data are fairly explainable on the basis of clock rate, except for the floating-point benchmark, where the NEC processor is almost 9 times faster (data not shown). All three processors perform floating-point operations in software. TriCore uses Tasking's floating-point library, the M32R/E uses Wind River's Diab compiler, and the V832's floating-point library comes from Green Hills. Perhaps Wind River's floating-point library for the M32R/E isn't as good as it could be, or the M32R/E may not be floating-point friendly. The Wind River Diab floating-point libraries (for all the processors they support) are written in C and are not processor specific; however, Wind River claims that its customers do not complain about the library's performance.

We can make another interesting comparison between Mitsubishi's M16C/80 and M16C/62A microcontrollers, as illustrated by Figure 1. While there's a 25% difference in clock speed between these processors, the M16C/80 averages 70% faster (the tooth-to-spark benchmark is almost three times faster).

Consumer Benchmarks

The EEMBC consumer benchmarks target processors used in digital cameras. Although similar to the other EEMBC benchmarks, one could use the consumer benchmarks in a general sense to represent a variety of workloads. The five consumer benchmarks include JPEG compression and decompression, a high-pass filter, and two color-conversion algorithms. To date, only a few EEMBC members have been bold enough to accept the challenge of the consumer benchmarks, but that situation helps to simplify the initial data analysis.

The JPEG and high-pass filter benchmark scores generate the most surprising results for the processors tested so far. The JPEG algorithm requires the CPU to perform preloading and branch prediction, as well as parallel/vector processing. The high-pass filter explores the target CPU's ability to perform arithmetic and matrix math. Each function takes a point or series of points in the RGB color space (three 8-bit numbers) and applies a transfer function to it.

Table 5 shows that National's GX-1 is 76–78% faster than the V832 for the JPEG and high-pass filter benchmarks, but it's 21–45% slower for the color-conversion benchmarks. This trend holds true for NEC's VR5000 and VR5432. For example, the VR5000 is approximately three to four times faster than the V832 for the JPEG and high-pass filter benchmarks but only about twice as fast for the color-conversion benchmarks.

The high-pass filter benchmark raises another interesting discussion point for NEC's VR5000 when it is compared with the GX-1 and NEC's VR5432. As Table 6 shows, the VR5000 averages 30% faster than the VR5432 for all the consumer benchmarks except for the high-pass filter. Likewise, the VR5000 is about 2.5 times faster than the GX-1 (data not shown), except for the high-pass filter, where it is only 1.7 times faster. Note that the GX-1's scores were handicapped, because the target system was also the host system. This situation implies that the benchmarks for this processor were run on top of the Windows operating system.

Printer Benchmarks

The office automation benchmark suite contains four benchmarks targeting printer applications: Bezier-curve calculation, dithering, image rotation, and text processing. The

Benchmark Scores Relative to NEC V832 (Processor X/V832)			
	National GX1	NEC VR5000	NEC VR5432
Compress JPEG	1.78	4.31	3.12
Decompress JPEG	1.33	3.22	2.39
High Pass Greyscale Filter	1.76	2.97	3.22
RGB-to-CYMK Conversion	0.79	2.20	1.42
RGB-to-YIQ Conversion	0.55	1.67	1.38

Table 5. EEMBC's consumer benchmarks show that National's 200MHz GX-1 can substantially outperform NEC's V832 for some tasks (JPEG compression/decompression and high-pass filtering), but the opposite is true for color conversion.

VR5000 Relative to VR5432 (VR5000/VR5432)		
Consumer	Compress JPEG	1.38
	Decompress JPEG	1.35
	High Pass Greyscale Filter	0.92
	RGB-to-CYMK Conversion	1.55
	RGB-to-YIQ Conversion	1.21
Office Automation	Fixed-Point Bezier Curve	1.01
	Dithering	1.19
	Image Rotation	1.23
	Text Processing	0.91
Telecomm	Autocorrelation	1.10
	Convolutional Encoder - xk5r2dt	1.88
	Convolutional Encoder - xk4r2dt	1.77
	Convolutional Encoder - xk3r2dt	1.51
	Fixed-Point Bit Allocation	1.93
	FFT	0.78
	Viterbi	1.57

Table 6. NEC's VR5000 is faster than the company's VR5432 for all EEMBC consumer, office-automation, and telecom benchmarks, except for the high-pass filter, text-processing, and FFT benchmarks. A composite score in each of these three benchmark suites masks these differences.

Benchmark Scores Relative to ElanSC520 (Processor X/ElanSC520)						
	AMD K6-2	IBM 750CX	NEC VR5000	NEC VR5432	NEC V832	Toshiba TMPR3927F
Fixed-Point Bezier Curve	13.32	NA	10.48	10.36	5.38	6.80
Floating-Point Bezier Curve	12.84	36.92	NA	NA	NA	NA
Dithering	7.48	24.42	7.67	6.43	2.79	4.14
Image Rotation	7.27	18.04	5.04	4.10	1.53	2.52
Text Processing	3.02	6.35	2.06	2.26	0.63	0.56

Table 7. The text-processing benchmark from EEMBC's printer suite appears to take away the performance edge for several processors.

Bezier-curve calculator benchmark, available in fixed- and floating-point versions, exercises a CPU's mathematical computation capability as it calculates points along a Bezier curve. This benchmark relies heavily on multiply and add operations. The dithering benchmark stresses a CPU's indirect references, used for managing internal buffers: its manipulation of large datasets, since large images will stress the cache; its ability to manipulate packed-byte quantities, which hold greyscale pixel information; and its ability to perform four byte-wide multiply-accumulate operations per pixel. The bitmap-rotation algorithm primarily tests a CPU's bit-manipulation, comparison, and indirect-reference capabilities. The algorithm uses a series of indirect references and bit masks to check and set individual bits in a data buffer representing a binary image. The text-processing benchmark tests the byte-manipulation, pointer-comparison, indirect-reference handling, and stack-manipulation capabilities of a processor. Text processing also performs many memory accesses and has the largest code size among the office automation benchmarks.

Table 7 compares AMD's ElanSC520 with the K6-2, two processors with dissimilar architectures but benchmarked with the same compiler. Notice that the K6-2 runs the Bezier-curve benchmark (both the fixed- and floating-point versions) 13 times faster than the ElanSC520 but is only 3 times faster on the text-processing benchmark. The K6-2 seems unable to take advantage of its superscalar architecture in the text-processing benchmark (i.e., there may not be enough parallelism to extract in this benchmark). On the other hand, why does the K6-2 do so well for the Bezier-curve benchmark?

Further, NEC's V832, is 5.4 times faster than the ElanSC520 for the Bezier-curve benchmark but 37% slower for text processing. Additionally, Toshiba's TMPR3927F is 6.8 times faster for the Bezier-curve benchmark, while the ElanSC520 is 1.8 times faster for the text-processing benchmark. The VR5432 is 50–60% faster than the TMPR3927F for all office automation benchmarks except text processing, where it is 4 times faster. Text processing also seems to trouble the VR5000. NEC's VR5000 is 20% faster than the company's VR5432 for the dithering and image-rotation benchmarks, but it is 10% slower for text processing (Table 6).

As we see for the automotive/industrial benchmarks, Mitsubishi's M16C/80 has performance superior to the company's M16C/62A, demonstrating the benefits of a newer

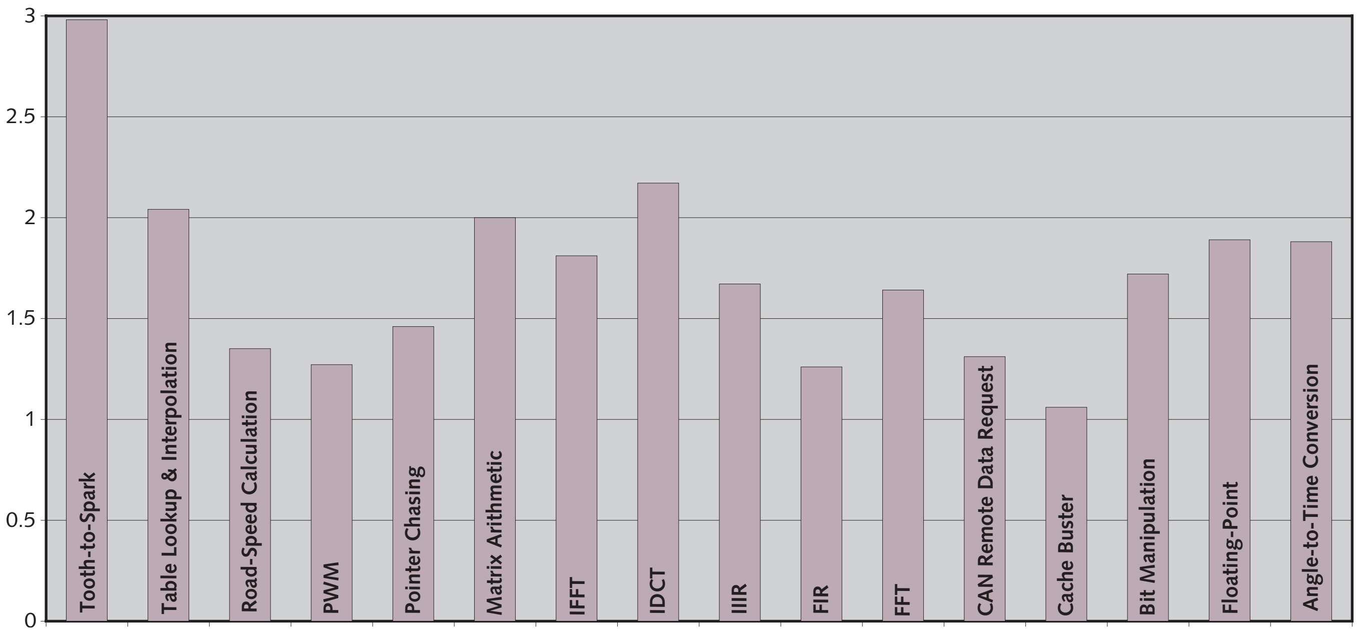

Figure 1. Direct comparison between two similar versions of Mitsubishi's M16C microcontroller—the M16C/80 and the M16C/62A—shows more difference in performance on the EEMBC automotive/industrial benchmark than might be expected from clock-rate differences alone. (M16C/80 relative to M16C/62A)

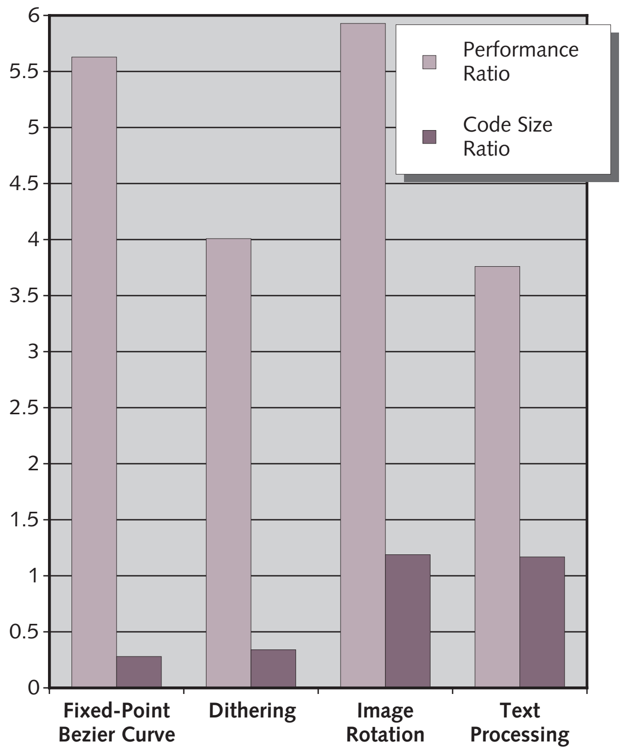

Figure 2. EEMBC office automation printer benchmark ratios show no correlation between code size and performance for the M16C/80 microcontroller relative to the TMP95FY64F. (M16C/80 / TMP95FY64F)

architecture and compiler. The M16C/80 also demonstrates superior performance over Toshiba's TMP95FY64F in the office automation benchmarks. Although both processors were run at the same clock speed, the M16C/80 ranged between 3.7 and 5.9 times faster. Curiously, the code sizes for each of the benchmarks do not provide a clue to the performance difference. Figure 2 shows the inconsistency of the performance and code size ratios for the M16C/80 relative to the TMP95FY64F. Although the M16C/80 runs the Bezier curve 5.6 times faster, its code size is 3.5 times smaller. But for the image-rotation benchmark, its performance is 5.9 times better, while its code size is 20% larger.

RISCs, CISCs, and DSPs Battle in Telecom

EEMBC's telecom benchmarks represent the traditional signal-processing fare. The autocorrelation benchmark stresses a processor's multiplication and summation ability. The benchmark takes a voice-data array and the number of lags (or time delays) as input and generates an array that contains the sum of products for each lag. The bit-allocation benchmark requires the CPU to distribute data into a series of frequency bins and allocates the data bits using a water-level algorithm. It distributes the data such that each bin gets an equal number of bits until the bin is full. The convolutional-encoder benchmark explores the target CPU's ability to perform bit-wise exclusive ORs and table lookups. The Fast Fourier transform (FFT) benchmark tests the complex mathematical and memory-access ability of a processor. This particular implementation uses a decimation in time performed on 256 complex points. The Viterbi benchmark is similar to the convolutional encoder, and it also performs bit-wise operations and table lookups in addition to comparisons. EEMBC runs each of these benchmarks with a variety of data inputs. For example, the autocorrelation benchmark is run with a pulse, sine, and speech wave.

When we compare the ElanSC520 with IDT's 79RC-32364 and NEC's V832, we observe that the IDT device is an average of 2.2 times faster, while the V832 is 2.8 times faster (data not shown). Perhaps this result demonstrates that the CAD-UL compiler (used for the AMD benchmark execution) isn't able to take advantage of the Elan architecture in this particular benchmark. Or is this performance advantage an inherent RISC advantage?

AMD's K6-2 and Analog Devices' 21065L (SHARC) really shouldn't be compared, because it's like comparing an apple and a banana. Nevertheless, a comparison allows us to determine an architecture's efficiency on a clock-by-clock basis. The K6-2 is four to six times faster than the 21065L, except for the autocorrelation benchmark, where performance is almost equal. This result demonstrates that the SHARC is designed for telecomm applications. Unfortunately, we don't have any SHARC benchmark data for the other application areas, as it would be interesting to see how well SHARC would do on networking benchmarks, for example.

In a more realistic comparison, the K6-2 is faster than NEC's VR5000 for the convolutional-encoder, FFT, and Viterbi benchmarks, but it is 10–50% slower for the autocorrelation and bit-allocation benchmarks. The SHARC is 10–30 times faster than National's CompactRISC CR16B, except for the convolutional-encoder and Viterbi benchmarks, where, curiously, it is only twice as fast.

Along the lines of a fairer comparison (DSP-to-DSP), as in the case of SHARC versus the TMS320C6203, the latter has significantly better performance on the autocorrelation and convolutional-encoder benchmarks, even for the out-of-the-box benchmark implementation. On the other hand, if you normalize for frequency, both processors have fairly equivalent performance. (While we've tried to avoid normalizing, both these processors execute from zero-wait-state memory, so performance should scale linearly with frequency.) More information in the bit-allocation, FFT, and Viterbi benchmarks would have allowed the TMS320C6203 compiler to better utilize the architecture, but the "out-of-the-box" benchmark approach is designed to provide the same compiler information to all vendors.

Expanding on this: TI used some fundamental C-level optimizations and gained significant performance benefits, as shown in Figure 3. In fact, bit-allocation performance increased by a factor of 3.8 to 6.5; FFT increased by a factor of 10.5; and Viterbi increased by a factor of 5 after optimization. Furthermore, the convolutional-encoder performance

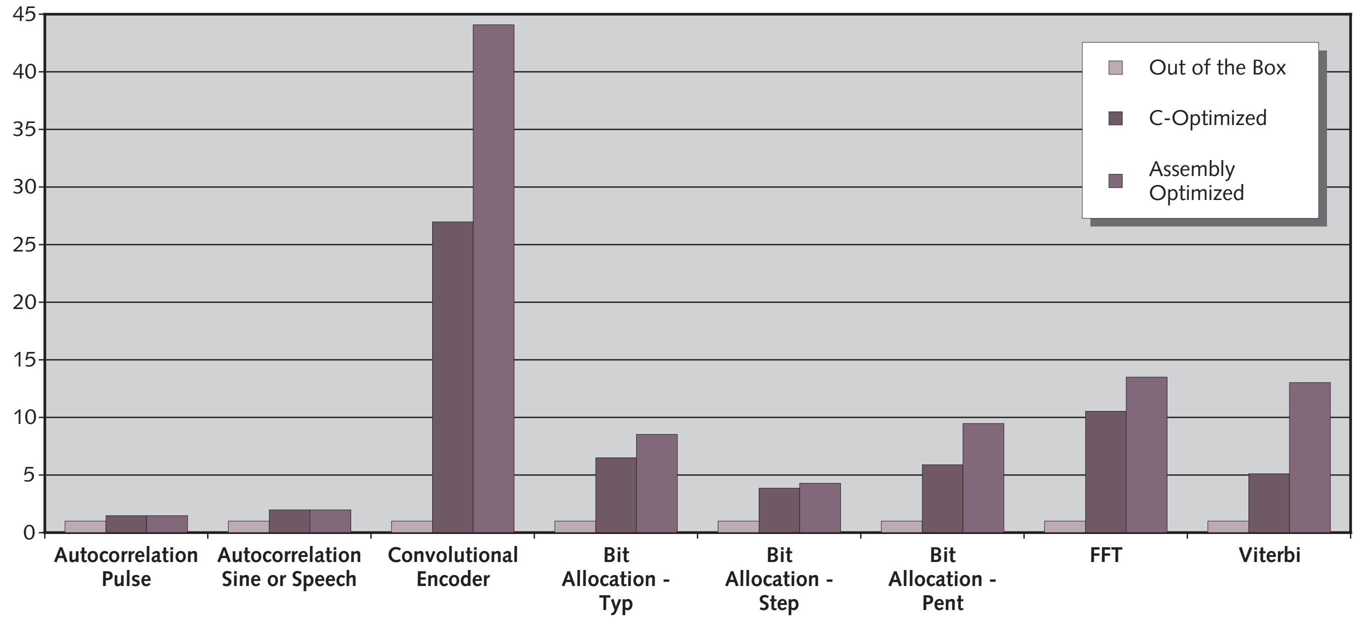

Figure 3. C-level optimizations produce dramatic performance improvements in the TI TMS320C6203 telecom benchmarks, while further assembly-language optimization substantially improves convolutional-encoder and Viterbi performance.

increased by a factor of 27. When TI further optimized the benchmarks, using assembly language to replace critical sections of C, the convolutional-encoder performance went up by an additional 60–70%. The Viterbi performance went up by an additional 250%! Apparently, these two benchmarks require hand coding to unleash the performance capabilities of this processor.

As one might expect from operating-frequency and fundamental architectural differences, NEC's VR5000 is an average of 50% faster than the VR5432. However, as Table 6 shows, the VR5432 is 29% faster than the VR5000 in the FFT benchmark.

Networking Benchmarks Keep Processors Busy

As the name implies, EEMBC's networking benchmarks probe a processor's ability to perform network-router-like functions. Fundamentally, these functions involve moving lots of data around and making routing decisions based on information stored in dynamically generated tables. The OSPF (open shortest path first) benchmark implements the Dijkstra shortest-path-first algorithm widely used in routers and other networking equipment. The Dijkstra algorithm builds a table of nodes, where each node is a router, and computes the shortest route to all nodes in the network relative to the source node. The route-lookup benchmark is a distillation of the fundamental operation of IP datagram routers: receiving and forwarding IP datagrams. It implements an IP lookup mechanism based on a Patricia Tree data structure. The packet-flow benchmark implements some of the processing required in RFC1812 ("Requirements for Routers"). It's designed to emulate the way systems store IP datagrams, using descriptors that are separate from the datagram. As each datagram is received and processed by the benchmark algorithm, it is removed from the receive queue and placed in a holding queue.

In a direct comparison, AMD put two compilers to the test on the same platform. Using its K6-2, AMD pitted Microsoft's C/C++ Optimizing Compiler V12.00.8168 against

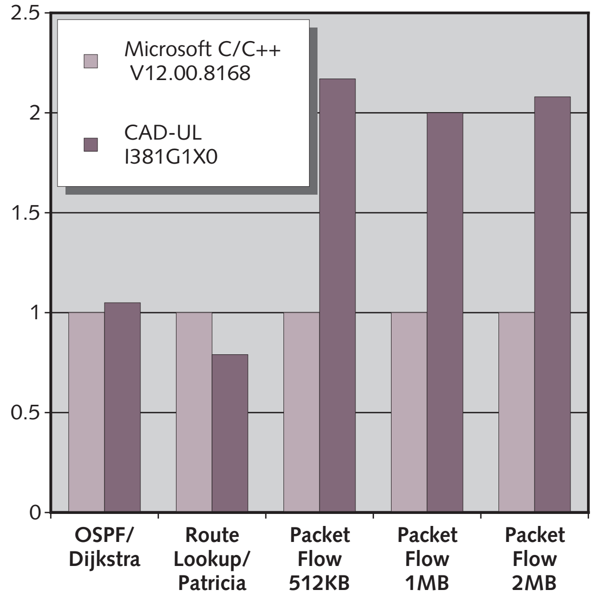

Figure 4. The same processor running the same benchmarks (networking in this example) can deliver substantially different performance when the benchmark code is compiled using different compilers.

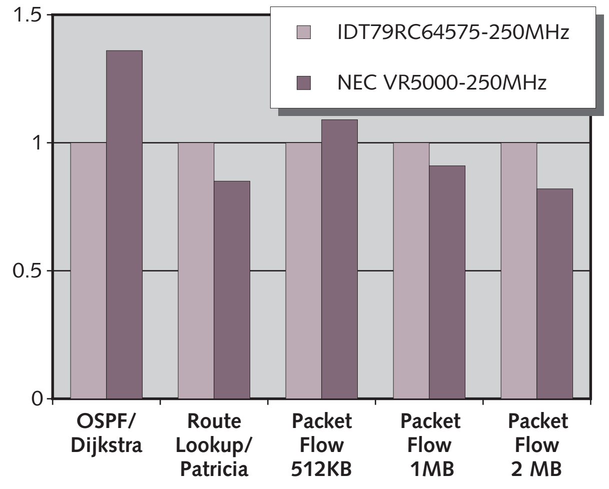

Figure 5. An aggregate EEMBC benchmark score would not highlight the differences between similar processor architectures as does this benchmark-by-benchmark comparison of NEC's VR5000 and IDT's 79RC64575, both running at 250MHz. (IDT79RC6457/VR5000)

CAD-UL's I381G1X0. The results, depicted in Figure 4, show the relative difference between the two compilers run on each benchmark. Hand-tuned code would probably present very different results. But this comparison, using the same processor but different compilers, surely points out some of the benefits of one compiler over another. The CAD-UL compiler performs overwhelmingly better for the packet-flow benchmarks and slightly better for the OSPF. But Microsoft's compiler performs 26% better for the route lookup. Perhaps the CAD-UL compiler isn't able to efficiently schedule the K6-2 resources for the packet-flow benchmark. This is further evidenced in a comparison with the ElanSC520 (data not shown). Both processors use the same compiler, albeit the optimization switches are set to match the processor. Although the K6-2 averages 7.5 times faster than Elan, the route-lookup benchmark on the K6-2 is faster by only a factor of four.

In another comparison involving Elan, this time against NEC's V832, the Elan runs the OSPF and route-lookup benchmarks 6% and 28% faster, respectively. But, surprisingly, for the packet-flow benchmark, it is approximately 40% slower than the V832. Furthermore, Elan runs OSPF twice as fast as Toshiba's TMPR3927, but it is 30% and 55% slower for the route-lookup and packet-flow benchmarks, respectively.

Note that an aggregate EEMBC benchmark score wouldn't point out the interesting differences between NEC's VR5000 with the Green Hills Multi2000 V2.0 compiler and IDT's 79RC64575 with the IDT/c 7.2.1 Gnu compiler shown in Figure 5. Both processors have fundamentally the same architecture, but the VR5000 runs the OSPF and packet-flow benchmarks (the 512K versions) 36% and 9% faster, respectively. In the other networking benchmarks, the VR5000 runs 9–18% slower.

Now, stay tuned. We realize that this article has presented far more observations than explanations, but its purpose is to uncover the interesting performance variations. EEMBC has provided the foregoing information to the respective processor vendors, and in Part 2 of this article, we will provide you with the vendors' explanations—along with our analysis.