CAPACITY GROWTH OF A CDMA2000 BASE STATION DUE TO MOORE'S LAW

John Chapin, Andrew Chiu, Victor Lum and Jeremy Nimmer
Vanu, Inc. Cambridge, MA, USA **info@vanu.com**

ABSTRACT

The software radio technology used by Vanu, Inc. is designed to deliver ongoing performance improvements to customers. An all-software approach, eliminating low-level firmware such as VHDL for FPGAs or DSP assembly code, enables low-cost porting of waveforms to new processors and platforms as they become available. This enables exploiting the Moore's Law growth curve of the semiconductor industry. This paper evaluates the success of this approach by reporting performance improvements delivered for a particular waveform, CDMA2000 1xRTT, across multiple hardware generations.

1. INTRODUCTION

Moore's Law exponential performance improvement charts such as the one shown in Figure 1 are commonplace. The potential benefits for software radio systems and users are immense: ongoing improvements in capacity, and ongoing reductions in size and power consumption, without increase in hardware cost. Yet there is little published data showing whether the raw performance increases offered by semiconductor manufacturers translate into full-system performance increases for users of software radio systems.

This paper analyzes performance of a single waveform across multiple Intel-based systems representing several years of technology improvement. We chose the CDMA2000 1xRTT cellular waveform, more specifically the basestation side implementation of its RC3 voice channel, since it has sufficiently high computation requirements to stress current state-of-the-art processors.

2. CHANNEL CAPACITY

Figure 2 shows the number of voice channels that the Vanu Anywave™ CDMA2000 physical layer can support on a single core. Over the 5.5 years separating the acquisition of the first and last machine in the study, channel capacity per core increased by a factor of 16. Not shown in the figure is a parallel increase in the number of cores. The "sweet spot" for price-performance is a two CPU server, which had 2 cores in the 2001 1.0 GHz system and is now offered with 8 cores in late 2007. If the core count increase were to provide linear performance growth (which in practice it does not) there would be a further factor of 4 capacity increase at the system level, for roughly the same server price.

Figure 2 also shows full-system benchmarks, where a

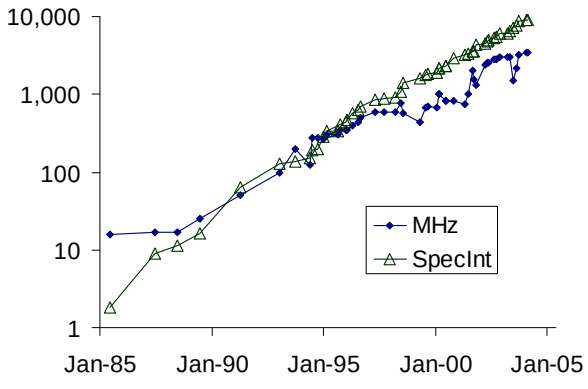

Figure 1. Moore's Law performance growth. Highest CPU clock speed and highest SpecInt benchmark score, 1985–2004. Data and SpecInt normalization from [1].

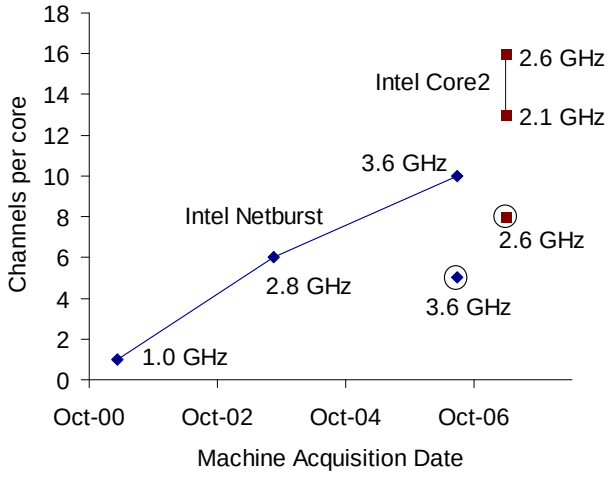

Figure 2. Channel capacity for CDMA2000 1xRTT RC3 (Vanu Anywave software). Circled data points include PHY, control, and I/O. Others are PHY-only CPU benchmarks.

radio head is connected and the BTS runs in an operational software configuration, although still using only a single core. Due to laboratory equipment limitations we were only able to benchmark two of the servers as full systems. These tests show that adding control processing, a BSC connection and radio head I/O to the PHY processing reduces the channel capacity by 50%.

3. PLATFORM EFFECTS

In addition to CPU speed and number of cores, other platform features significantly affect performance. Table 1 describes the systems used for the study, labeled A-E.

One critical part of the platform is the compiler used for the signal processing code. We have found that the Intel C compiler generates substantially more efficient code than the Gnu C compiler for our applications. Figure 3 compares the PHY-only channel capacity for the two compilers (vertical bars and left axis). The quantization into an integer number of channels masks the magnitude of the benefit of using ICC. To show the benefit more clearly, the plotted line and right axis show the performance improvement before quantization. It ranges from 25% for a 2003 vintage machine with a Xeon up to 52% for a current Core2 server. (Since the Core2 machines were introduced only recently, we can expect GCC's code generation for the target to improve over the next year or two.)

Another critical part of the platform is the availability of a pipelined vector execution unit, such as SSE and SSE2 for Intel processors. These execution units are very efficient for the streaming data computations characteristic of signal processing code. SSE2 on the Core2 microarchitecture is particularly efficient because a 128-bit wide operation can be issued every cycle, compared to every two cycles on Netburst. Figure 3 shows an overall capacity benefit of 3x–4x when the vector instruction unit is exploited.

The performance results shown in Figure 2 all use ICC with SSE2, except for platform A whose Pentium III Coppermine does not have SSE capability.

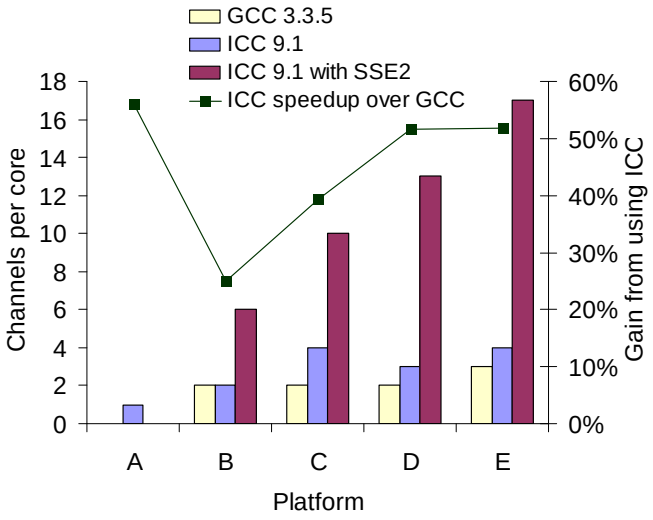

Figure 3. Performance effects due to compiler choice and use of the SSE2 vector execution unit. ICC is the Intel C compiler, while GCC is the Gnu C compiler.

4. KEY SOFTWARE COMPONENTS

We used a nonintrusive statistical sampling tool to investigate where the computation time is spent (Table 2). Unsurprisingly, the bulk of the processing is spent in the receive chain modules closest to the RF head, where the sample rate is highest. The Netburst core (platform C) and Core2 core (platform D) had roughly similar processing costs for the key software modules.

One interesting observation is the significant reduction in relative despreader cost on platform C when comparing the full system test to the PHY-only benchmark. Recall that channel capacity is reduced by about half when MAC and I/O are added to the signal processing computation. We would therefore expect the despreader to take about half as

Table 1. System Characteristics

ID	Mfr	Model	Acq. Date	Total Cores	Core speed (GHz)	CPU Brand	Core arch	Total L2 Cache (KB)	Memory bus (MHz)
A	Dell	GX150	4/1/2001	1	1.0	Pentium III	Coppermine	256	133
B	HP	DL380 G3	9/1/2003	2	2.8	Xeon	Netburst	512	266
C	IBM	xSeries 336	7/1/2006	2	3.6	Xeon	Netburst	2048	266
D	Dell	Optiplex GX745	4/1/2007	2	2.13	Xeon 6400	Core2	2048	667
E	IBM	System x3550	4/1/2007	4	2.6	Woodcrest	Core2	8192	266

Table 2. Software Component Costs (% of application)

Experiment	C PHY only	C full system	D PHY only
Despread	21	5.3	29
Rake finger estimation	12	4.5	13
Rake finger search	6	< 1	6
Equalizer	4	1.7	5

much of the overall CPU in the full system test as it does in the PHY-only test, yet it actually takes only a quarter of the amount. We suspect that this is due to cache effects. In the PHY-only test, the despreader module incurs all the misses required to pull input sample data into the cache. When the full system is running, the I/O processing stack pulls the input sample data into the cache before the despreader is invoked.

5. METHODOLOGY

PHY-only tests: The 1xRTT physical layer application process was configured to use a single thread and to process 16 voice channels within a single 1.25 MHz CDMA2000 carrier. The input I/Q data stream was all zeros and the produced output data was discarded. The process was run for precisely 10 seconds and the number of samples processed was counted. This experiment was repeated four times. Channel capacity *C* was estimated from the averaged sample count *N* as:

$$C = floor\left(\frac{16 \times N}{10 \times 1228800}\right)$$

where 1228800 samples/sec is the required real-time processing rate for the carrier.

Full-system tests: The full 1xRTT BTS was executed, using a stub base station controller rather than a full BSC. The physical layer application was configured to use a single thread. The server was connected via gigabit ethernet to an RF head [2]. The BSC was commanded to establish four voice channels, modeling the case of four independent landline-to-mobile calls. With the system in this steady state, the Linux *top* application was run [3]. It collected system load information every second. The average of 10 reports was taken. Channel capacity *C* was estimated from the averaged load *L* as *floor(4/L)*. We validated that the reported capacity numbers do not change when the platform is booted in single-core mode.

Software module costs: The above 2 test setups were used, except the platform C PHY-only test processed only 4 voice channels to allow better comparison with the platform C full-system test. Performance analysis used the OProfile tool [4], a whole-system statistical sampling profiler that leverages the hardware performance counters provided in the CPU.

6. IMPLICATIONS

This paper shows that the Moore's Law curve of the semiconductor industry can deliver ongoing significant performance growth to users of software radio systems. Platform improvements over a 3.5 year period (September 2003 to April 2007) increased CDMA2000 channel capacity per core from 6 to 16, and number of cores per dual-CPU server from 2 to 4. We expect this trend to continue. For example, 8-core servers are now widely available. A radio operator who invests in software-radio based systems can expect ongoing reductions in cost or increases in capacity (or both) as nodes are incrementally added to a deployed system.

Not every software radio vendor can deliver the benefits of Moore's Law to end users. Waveform software for software radio is expensive to develop. If it is not portable across hardware generations, the vendor is locked in to obsolete components until the software's cost has been fully recovered. This is often a problem when vendors select DSPs or FPGAs as their SDR signal processing platform.

Vanu, Inc. has focused its engineering efforts on developing portable waveform software. All the signal processing executes as an application, on a standard OS, on processors such as Intel x86 and PowerPC. The software is highly modular so individual components can be specialized to exploit processor features (such as the new SSE3 vector instruction set from Intel) without affecting the rest of the software. This system design gives Vanu's products, such as the Anywave™ Radio Access Network, a unique ability to deliver ongoing performance improvements to operators and users.

REFERENCES

[1] M. Ekman, F. Warg, and J. Nilsson. "An In-Depth Look at Computer Performance Growth." Technical Report 2004-9, Chalmers University of Technology, Department of Computer Engineering, Göteborg, 2004.

[2] G. Britton, B. Kubert, and J. Chapin. "RF Over Ethernet for Wireless Infrastructure." SDR05, SDR Forum Technical Conference 2005.

[3] http://linux.about.com/od/commands/l/blcmdl1_top.htm

[4] http://oprofile.sourceforge.net/about/