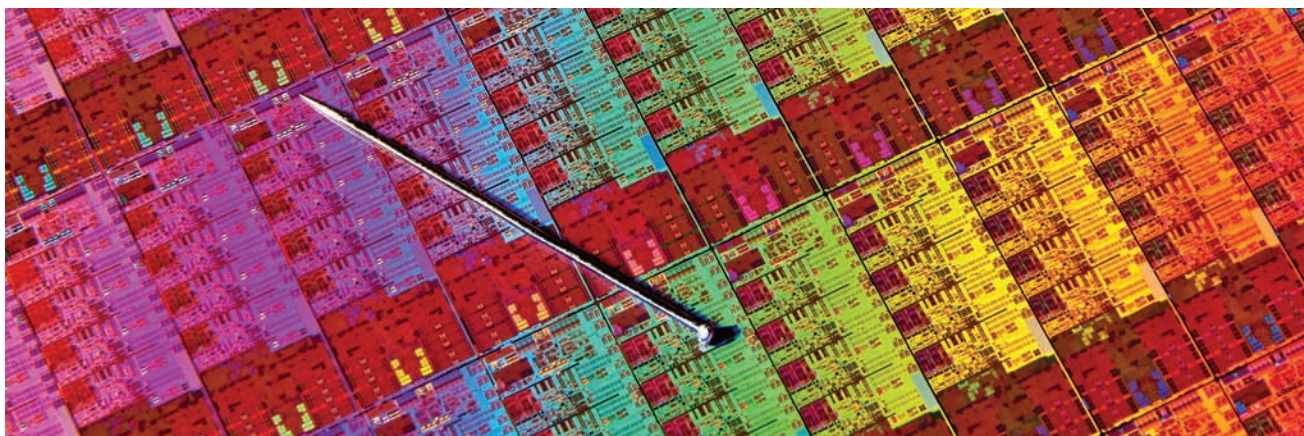


# More Than We Ever Dreamed Possible

## Processor Technology for GNSS Software Receivers in the Year 2015



Wikimedia Commons/Intel Free Press

Recent increases in computational power can be used to build more efficient GNSS software receivers. Experimental benchmarks show how well the currently available technology can be exploited for these purposes. A personal computer (maximum number of channels greater than 1,000) and an embedded board (maximum number of channels less than 100) are tested under various scenarios, with promising results for future applications.

Computational power continues to increase at a rapid pace and unlike other areas of technology, this trend is not expected to slow down in the foreseeable future. Software GNSS receivers fully exploit these developments to steadily increase performance.

Wikipedia lists 21 software receiver projects currently in progress. Meanwhile, more than 1,000 wideband GNSS channels can be tracked in real-time on a conventional PC. Moreover, smartphones containing multi-core central processing units (CPUs) and powerful many-core graphics processing units (GPUs) bring these super computing technologies into our embedded devices.

All these developments can be exploited to produce power-efficient, customized receivers with flexible correlation schemes and more advanced positioning techniques. For example,

promising techniques such as the Direct Position Estimation (DPE) paradigm or usage of tracking solutions based on particle filtering, seem to be very appealing in challenging environments but are likewise computationally quite demanding.

This article sheds some light on recent processor developments and relates them to use in GNSS software radios.

### Handheld Supercomputing

Building a GNSS receiver involves three major tasks: detection of GNSS signals, tracking them, and using the obtained ranging information to compute the user position. In contrast to hardware GNSS receivers, a software GNSS receiver allows engineers to easily adapt the algorithms and design principles employed so as to match the needs of each application domain.

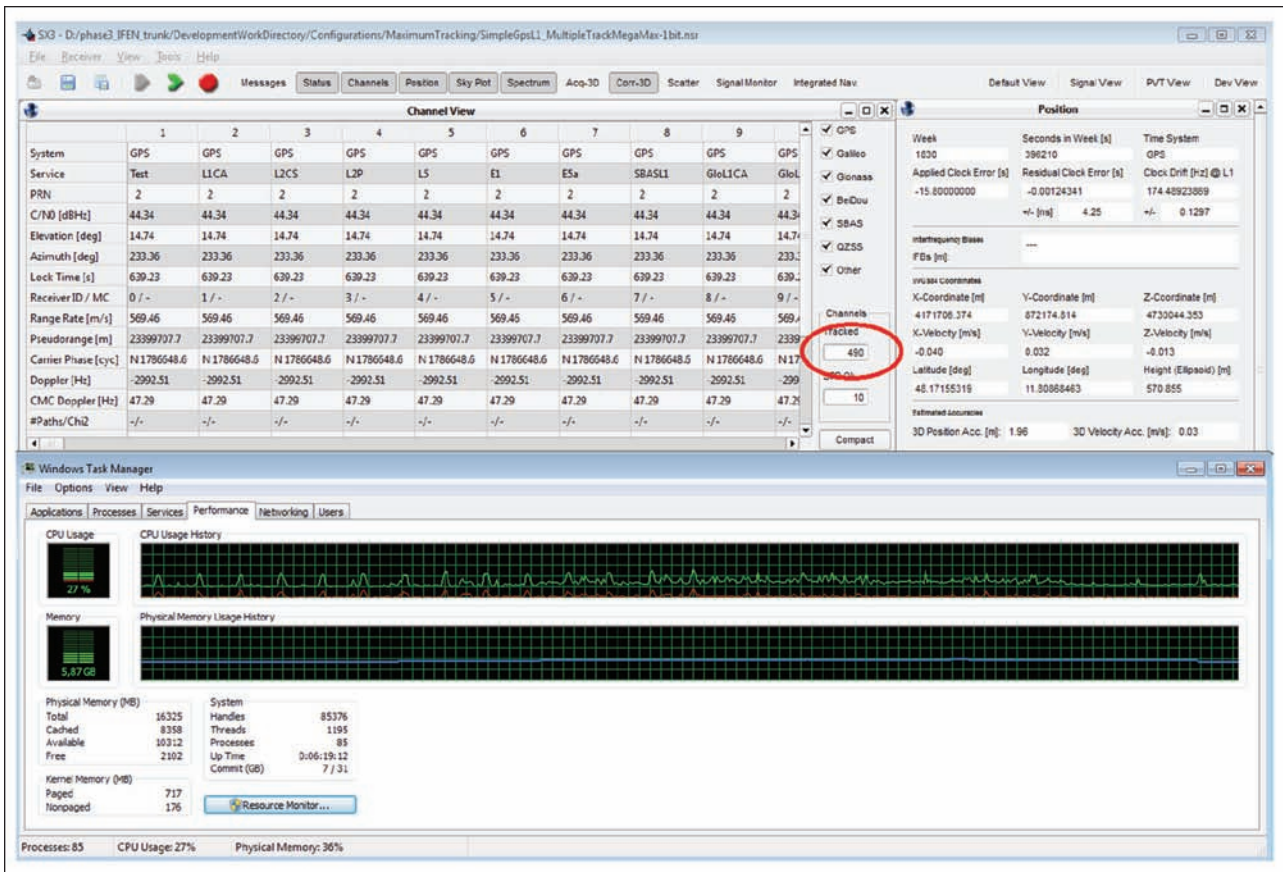
The three core tasks of receiver development have various characteristic numerical operations that must be performed. For GNSS signal acquisition, the use of the fast Fourier transform (FFT) is virtually inevitable. For signal tracking, the correlation of received signals with internally generated replica signals must be performed — either as a dot-product operation with multiply-and-add com-

JÜRGEN DAMPF, THOMAS PANY, WOLFGANG BÄR, JÓN WINKEL, CARSTEN STÖBER  
IFEN GMBH

KARL FÜRLINGER  
LUDWIG-MAXIMILIANS-UNIVERSITY (LMU)  
MUNICH

PAU CLOSAS,  
CENTRE TECNOLÒGIC DE TELECOMUNICACIONS  
DE CATALUNYA

J. A. GARCIA-MOLINA  
HE SPACE FOR EUROPEAN SPACE  
AGENCY/ESTEC



**FIGURE 1** Screen shot of a software receiver tracking 490 wideband channels (20.48-megahertz sample rate) in real-time on a standard PC plus processing load

mands or as an exclusive-or (XOR) operation if one-bit sampling is used. Positioning filters typically make use of various floating point operations to compute, for example, satellite positions or to update the navigation filter.

In mass-market receivers the first two tasks are accomplished by application-specific integrated circuits (ASICs) with the user position computed either on that ASIC or by an external general purpose processor. Current ASIC technology is highly efficient, and ASIC-based GNSS receivers can solve the aforementioned tasks, consuming only milliwatts of power while still allowing for user position tracking in the background. This facilitates applications such as geofencing, user motion detection in wearables, and so forth.

However, the accuracy of such receivers is still on the order of several (dozens of) meters or worse, especially if operated in a mobile phone under degraded signal conditions. Needless to say, mass market receivers have limited capabilities

to be configured for specific applications as the core proprietary algorithms are built into the silicon chip and positioning algorithms are intentionally kept simple to maintain the low power consumption.

Realizing all three receiver tasks in software generally allows implementation of more flexible algorithms by a larger community of engineers and researchers. The time needed to build or adapt a so-called software receiver is significantly shorter because, apart from the RF front-end, no dedicated hardware development is involved. Indeed, nowadays software GNSS receivers are the gold standard in research and development (R&D) especially when developing new algorithms, testing new navigation signals, or fusing GNSS with other sensors.

R&D software receivers typically run on a conventional desktop personal computer (PC) or a laptop, which not only allows real-time processing of hundreds of channels, but also re-pro-

cessing of the same signal many times to test various algorithms and parameters.

This R&D market is definitely not very large, but the technology fully profits from the ongoing developments in the PC sector, including ever more powerful CPUs and graphics processors, which we will outline in this article. Only 18 years after Dennis Akos presented a first post-processing software GPS receiver in his Ph.D. thesis, real-time all-in-view tracking of the complete GNSS constellation is possible on a PC costing the same as it did then.

**Figure 1** shows, as an example, the computational load when tracking 490 wideband channels in real-time on a standard \$1,000 PC. Minor internal modifications in the software receiver were made in order to simulate the availability of such a large number of signals with a rooftop antenna. This is done by simply assigning identical signals to all available channels, which can cause the irritating output of, e.g., Column 7 in Figure 1 (System GPS; Service E5a). The

processing load is below 30 percent with occasional spikes to about 50 percent during signal acquisition.

When Samsung recently introduced the quad-core mobile phone S4 with its embedded powerful graphics processor, it became clear that we have entered the era of mobile super-computing. The graphics chip Adreno 420 in the Samsung Note 4 is able to perform more than one hundred billion floating point operations per second (GFlop/sec), enabling hyper-realistic 3D gaming.

Naturally, the power consumption and form factor of the chips and boards in question are now much more intriguing. Mobile supercomputing devices may be used to design and build a new generation of GNSS software receivers, thus escaping the R&D niche and extending their adoption by a wider application market, with potentially significant benefits for the end-user. For example, precise positioning in degraded or indoor environments is still a challenge not achieved with conventional hardware receiver technology.

In this context, sophisticated methods of GNSS/inertial integration and/or nonlinear estimation filters could make much better use of the available information from the GNSS signals than existing technology does. Later we will discuss how DPE fuses GNSS signal tracking and position estimation and how more flexible software receivers can foster its adoption.

Additional niche markets, such as GNSS space receivers, may emerge for software receivers, as they can better adapt to the challenging environment and specific requirements. For example, launch receivers not only face a poor satellite signal geometry (and thus rather low received signal power) due to the mounting of the antenna, but they also must cope with high vehicle/signal dynamics.

In the remainder of this article, we will first summarize the current state of mobile supercomputing and then match those developments to the GNSS software receiver core activities. Benchmarks are presented for FFT and signal correlation during tracking, evaluated both on a state-of-the-art PC and on an embedded platform purchased at the end of 2014. We will then map the results in terms of the effective number of correlators available for signal acquisition and the number of receiver channels.

The article wraps up with a discussion of the importance of these computing advances in regard to the feasibility of implementing advanced tracking and positioning techniques, both of which would benefit from the supercomputing capabilities of current processor technology.

At the time of publication of this article we continued testing other embedded platforms and an Android-based mobile phone, and will present those results at the ION GNSS+ 2015 conference in Tampa, Florida, in September.

### Faster than Moore's Law

Common wisdom assumes that computers are becoming ever faster, smaller, and more efficient. In the world of supercomputers actual empirical data backs up and quantifies this impression. The Top500 organization has been tracking the performance of the world's 500 most powerful computing systems

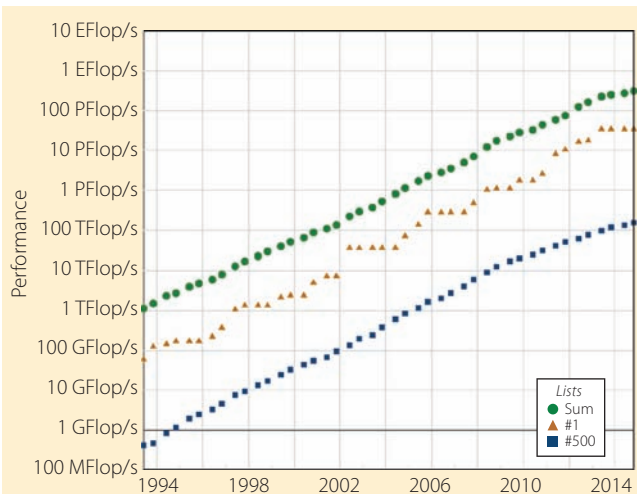


FIGURE 2 The Top500 list of supercomputers

for more than 20 years. The number-one system on the first issue of the list in June 1993 achieved a whopping 59.7 GFlop/sec when solving a linear system of equations using an LU-decomposition approach. This baseline metric is still used today to evaluate performance in terms of the Linpack benchmarks.

When looking at the historic development of the Top500 list provided in **Figure 2**, we can identify remarkable performance growth since 1993. The average performance of the 500 top systems grew by an average of 84 percent each year since 1993, and the performance of the number one-system improved even faster (90 percent per year).

This improvement is fundamentally fueled by advances in semiconductor technology governed by Moore's Law, which states that the number of transistors doubles roughly every 18 months (or, in other words, improves by about 60 percent per year). Computer architecture enhancements (in the past) and increased parallelism (a more recent trend) account for the performance growth beyond what Moore's Law predicts.

The trend towards parallelism is, however, not limited to supercomputers. While the big systems in the Top 500 list today are made up of hundreds of thousands of cores, multi- and manycores are dominating desktop and server systems and have more recently also appeared in mobile and embedded platforms. One speaks of a multi-core system, if it utilizes a few (below or around a dozen) computational units, while a many core system may have a hundred or more units.

Multiple computing cores are, however, not the only form of parallelism employed by modern architectures. A particularly efficient form of parallelism is known as SIMD (single instruction multiple data), which is employed in virtually all computers today. Here, the same operation is performed on multiple data items instead of just one. For some application areas, such as dense linear algebra, this directly translates to increased performance.

SIMD (with the number of data items that can be processed in parallel) is constantly increasing. For desktop CPUs, the SIMD registers are today 256 bits wide and they can thus operate on four double-precision floating point (FP) numbers or

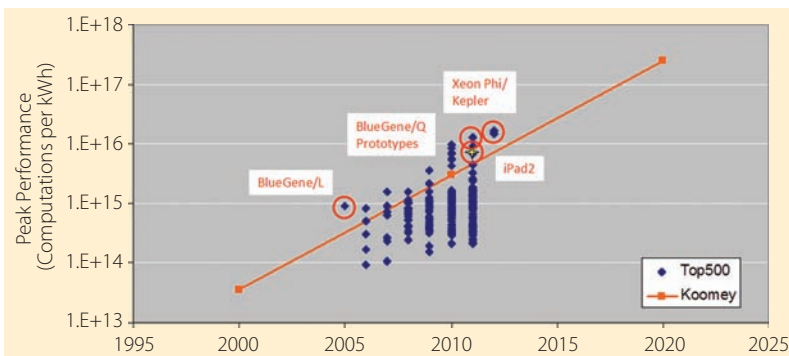


FIGURE 3 Computational power per kilowatt

puting devices. **Figure 3** shows such a comparison between the achieved energy efficiency of systems in the Top 500 list (blue dots), the prediction made by Koomey's Law as a red line, and the Apple iPad2 as an example of a mobile system. The highlighted systems have been specifically designed for energy efficiency (the BlueGene line of systems by IBM) and more recent accelerator platforms based on Xeon Phi and GPU hardware plus the iPad2.

### Our Test Systems

Today, a standard desktop PC consists of two main computational units: the CPU and the GPU. **Figure**



FIGURE 4 Test platforms: PC and embedded system

4 shows our benchmark system featuring a standard off-the-shelf \$1,000 PC. The Intel Core i7 CPU has four cores, each running at 4.0 gigahertz. The graphics card used is an nVidia GTX 770, with the GPU employing 1,536 cores running at a base clock rate of 1.046 gigahertz.

The theoretical peak performance of the GPU reaches the impressive number of 3,213GFlop/sec (=3,213,000,000,000 floating point operations per second). How much of this performance can be exploited for a specific algorithm is a different story, as we will see later in the specific case of an FFT. The Top500 list uses other metrics for ranking purposes.

For benchmarking the software receiver algorithms on an embedded system, we use a ~\$200 Arndale Octa board with a Samsung Exynos 5420 CPU consisting of four 1.8-gigahertz Cortex-A15 cores plus an ARM Mali-T628 MP6 GPU, which appears in the lower portion of Figure 4. The GPU has a remarkable floating point performance of 102 GFlop/s when using all six GPU cores. Our benchmark application uses only four GPU cores in order to minimize the adaptation effort within the Linux kernel, which yields a theoretical peak performance of 68 GFlop/s for the GPU.

Very high processing power on a tiny platform allows us to bring highly demanding software applications into new fields of application. Many mobile applications often have some power constraints, especially in the aviation or space sector, where power-to-weight ratio plays a dominant role. Thus, the power consumption of such an embedded software receiver is an important attribute that we have measured for typical load levels.

### Some Words about Power Consumption

One has to be careful when performing a power consumption measurement for a specific application due to the fact that a peripheral not used for a specific application, such as a video connection, can contribute to the overall power balance significantly.

We set up three measurement scenarios to determine the power consumption of the Arndale Octa development board running a software receiver acquiring and tracking common GPS L1 signals. **Table 1** lists all power measurements during the development phase of an embedded software receiver.

The first measurement acts as a baseline and determines the power consumption of the board with the default periph-

eral settings and only the Linux operating system running. The second scenario determines the increase of power consumption during nominal tracking (after acquiring the signals). This is achieved by setting 12 tracking channels and switching off the acquisition unit. The third scenario is similar to the second, but with the acquisition unit permanently active, which causes an additional increase in CPU load and heavily loads the GPU. The overall power consumption with acquisition activated increases to 4.35 watts with the largest portion attributed to the background power.

All measurements were performed during the early development phase of the embedded software receiver. Especially for the CPU load and, therefore, in the power-to-number-of-tracking-channels ratio, we have not yet implemented many possible optimizations.

The increase in CPU load during signal acquisition is based on a first implementation of the OpenCL acquisition, which pre-computes all Doppler bins for acquisition search on the CPU and then moves the whole data set to the OpenCL-based FFT engine, thus con-

suming a lot of CPU power. Naturally, this recomputation should also happen directly in the OpenCL kernel running on the GPU.

As we will see in the next section, our FFT implementation achieves around 1 GFlop/s and consumes around 0.3 W of power, which translates to around 3 GFlop/s per Watt. This value is typical for the power efficiency of the Top500 supercomputers in November 2014 and supports the validity of Koomey's law: computational performance is mostly determined by the year a chip is built and the power it consumes.

### Signal Acquisition and FFT Performance

For GNSS signal acquisition on the PC, we selected as a test case the cold-start acquisition of all 32 GPS C/A codes. A code resolution of 0.25 chips was used and a primary FFT size of 4,096 points. A Doppler search space of  $\pm 5.5$  kilohertz was considered. We used an optimized FFT algorithm invented by David Akopian and described in section 9.5.6 of the book by T. Pany listed in Additional Resources. This algorithm efficiently exploits the fact that the GPS C/A code

repeats 20 times within one data bit, which means that only relatively short primary FFTs (of 4,096 points) have to be performed.

Table 2 provides an overview of how fast the FFT performed this acquisition with various settings of the coherent and noncoherent integration times on our benchmark PC. The algorithm non-coherently sums up the coherent blocks leading to an overall integration time of about two seconds for all performed measurements.

The total time needed for integration includes all processing steps on the PC, including intermediate frequency (IF) sample selection and decisions about which satellites to acquire. In particular, it includes data transfer to and from the GPU.

Based on the chosen settings, a certain number of correlation values were computed. For a longer coherent integration time, this value increases as the number of Doppler bins increases (in order to be below a certain Doppler correlation loss). The GPU carried out computation of the FFTs together with other operations, which consumed a large portion of the overall integration time. To compute the FFTs on the PC, the so-called cuFFT library is used, which is part of the CUDA 6.5 toolkit (nVidia). This is a highly optimized FFT library tailored for nVidia hardware.

The FFTs for the noncoherent integrations were executed in parallel on the GPU. Thus, for a decreasing number of noncoherent integrations the computational performance expressed in GFlop/sec of the GPU decreases as the parallelism of 1,536 GPU cores is not fully

	System state	CPU load [%]	No. tracked channels	Current [A]	Voltage [V]	Power consumption [W]
1	OS Linux started SW Receiver Off	0.20	-	0.525	5.237	2.75
2	SW Receiver On Tracking only	52.0	12	0.781	5.226	4.08
3	SW Receiver On Tracking + Acquisition	70,6	12 + acq.	0.832	5.223	4.35

Table 1 Power consumption of the embedded software receiver

Coherent Integration time [ms]	Number of noncoherent integrations	Total time needed [s]	No. of correlation values	No. of primary FFTs	FFT time [ms]	GFlop/s within FFT	Effective no. of correlators for GPS C/A [Million]
2	1,024	1.87	3,276,800	819,200	448	450	3.6
4	512	2.0	6,422,528	802,816	465	424	6.6
8	256	2.26	12,451,840	778,240	535	357	11.3
16	128	2.84	24,510,464	765,952	721	261	17.7
32	64	4.0	48,758,784	761,856	1040	180	25.0
16	1,024	18.84	24,510,464	6,127,616	2852	528	21.3

Table 2 FFT acquisition performance on the PC

exploited. Whereas the GPU itself still offers a factor of 5-to-10 of additional computational power (we maximally used 528 GFlop/sec out of the potentially available 3,213 GFlop/sec), with a setting of, for example, 32 milliseconds of coherent and 64 noncoherent integrations, this method enables cold-start acquisition of the whole GPS constellation with a sensitivity around 20 dBHz within only four seconds. To our knowledge this is orders of magnitude faster than any hardware receiver can achieve.

When leaving the CUDA world, one immediately realizes that efficiently implemented FFT libraries do not grow on trees. In this context, the OpenCL standard should be mentioned because it provides a computing language compatible with many CPUs and GPUs. Indeed, our embedded test system was also able to support OpenCL. Fortunately, the Advanced Micro Devices, Inc. (AMD) c1FFT library exists as an open-source project facilitating efficient implementation of FFTs based on OpenCL. This library was initially implemented and optimized for AMD GPUs, but also runs on other OpenCL compatible GPUs, including our embedded test system to perform benchmarks.

**Table 3** shows the results of the FFT benchmarks (not including a full acquisition algorithm), where an FFT length of 8,192 points was used. This length matches well with the Galileo E1 codes that have 4,092 chips and a PRN code duration of four milliseconds. Within a very simplistic FFT-based acquisition algorithm, one FFT can be used to search one Doppler bin, yielding a coarse estimate of the possible number of correlators shown in Table 3.

More sophisticated algorithms (like the one used to benchmark the PC) and an optimized FFT library would strongly increase this number of correlators. Please note that the correlators discussed here are complex correlators including I and Q components. The effective number of correlators corresponds to the number of (hardware) correlators running in parallel in real-time to achieve the same result.

### Signal Tracking and Correlation

For the signal tracking benchmark on the PC, we set up a dedicated program simulating a full receiver tracking channel, including a code and carrier numerically controlled oscillator (NCO) plus correlators. Then, we measured the number of samples the platform can process within one second. As the default mode, the channel was configured to work with five correlators (e.g., very early, early, prompt, late, and very late).

We implemented two correlation schemes. The first one operates with one-bit samples and actually realizes the correlation as an exclusive-OR operation plus bit counting. The second scheme uses 16-bit samples, and the correlation is performed with vectorized (SIMD) multiply-and-add commands. Both are based on the GPS C/A code.

Our measurements on the benchmark PC show that a single CPU core (actually one-half of a CPU core, since hyper threading was activated) is able to process five billion or 5,000 million correlations per second (MCOPS) in the one-bit mode and 2,140 MCOPS in the 16-bit mode at a sampling rate of 20.48

No. of parallel FFTs	GPU execution time [ms]	GFlop/s	Effective no. of correlators for Galileo E1 [Thousands]
1	1.073	0.5	30
2	1.633	0.65	40
4	2.572	0.83	50.6
16	8.575	1.0	60.7
50	24.720	1.1	65.8

**Table 3** FFT performance on the embedded system

megahertz. One correlation corresponds to the processing of one pre-correlation sample for the sake of producing five complex correlation values.

For a sample rate of 20.48 MHz, a single tracking channel working with five correlators consumes 20.48 MCOPS. Doubling the number of correlators per channel increases the MCOPS demand per channel, but this increase is not a linear function of the number of correlators due to the mechanism the CPU schedules the actual operations. For example, two channels with 5 correlators generally would consume slightly more MCOPS than one channel with 10 correlators.

The one-bit correlation task is supported by the Intel architecture with two assembly SIMD commands. A PXOR instruction computes the bitwise XOR of two 128-bit registers, which represents the multiplication of the received and the local replica signal. The POPCNT (population count) instruction counts the number of ones within a 64-bit quad-word and realizes the integration of the product signal. Both make use of the multimedia registers of the CPU whose bit width tends to double from one processor generation to the next.

The 16-bit correlation is supported by the Intel SIMD assembly instruction VPMADDWD (packed multiply and add) that multiplies two 16-bit values and adds pairs of them. It performs this eight-wise in parallel. The VPMADDWD instruction is available for all CPUs supporting the AVX2 instruction set. Within one clock cycle, 16 samples can be correlated.

Apart from the correlation itself, the local replica signals must be generated, which is typically implemented via lookup tables for both the sine/cosine values and the PRN codes. Intel supports this operation by the command VPGATHERDD (gather packed dword values) allowing us to load up to eight values for one instruction. Besides these core operations, the NCO update, discriminator computation, loop update, navigation message decoding, and other tasks also have to be implemented very efficiently to fully exploit the computational resources.

**Table 4** shows the tracking performance on the PC, assuming a certain sampling rate and a certain number of virtual cores. (On the PC, one physical core corresponds to two virtually available cores due to hyper threading.)

For comparison, we carried out similar benchmarks in the year 2003 on a Pentium IV operating at 2.4 gigahertz. This was a single-core CPU. An MCOPS value of 150 (using a slightly different 16-bit tracking channel architecture) was achieved

Type	MCOPS per core	Sample rate	No. of used CPU cores	Channel no. upper limit
1-bit	5,000	20.48 MHz	8	1953
16-bit	2,140	20.48 MHz	8	835

**Table 4** Tracking performance on the PC

No. of correlators	MCOPS per core	Sample rate	No. of used CPU cores	Channel no. upper limit
2	428	20.48 MHz	4	83
2	428	40.96 MHz	4	31
3	326	20.48 MHz	4	63
5	231	20.48 MHz	4	45
21	66	10.24 MHz	4	25

**Table 5** Tracking performance on the embedded board

(described in detail in the article by T. Pany *et alia*, 2003). As we can achieve today a number of  $8 * 2140 = 17120$  MCOPS, this is a performance increase of a factor of 114 over 12 years. In other words, in light of this comparison, the tracking performance of the GNSS software receiver seems to double every 21 months. An observant reader may note that this increase is slower than Moore's Law, expressing the fact that exploitation of hardware technology for a certain application is not a straightforward story and requires constant optimization of source code to match available hardware structures.

The same tracking benchmark was carried out on the embedded system described earlier. In this case we used only the one-bit correlation. **Table 5** shows the corresponding results. Furthermore, we considered a varying number of correlators: 21, 5, 3, and 2. As depicted in Table 5, increasing the number of correlators per channel results in a reduction of the maximum number of channels.

For example, the embedded system easily allows implementation of a single-frequency GPS, Galileo, or GLONASS plus Beidou receiver. Alternatively, wideband tracking (20-megahertz bandwidth and 40-megahertz sample rate) of dual-frequency GPS is possible.

Another conclusion that these numbers suggest is that multi-correlation schemes (that is, more than five per channel) are possible. The latter opens the door for implementing sophisticated synchronization and positioning solutions that typically use large numbers of correlators per unit of integration time.

### Advanced Tracking, Positioning Techniques

Legacy GNSS receivers use phase lock loop (PLL) and delay lock loop (DLL) variations to perform signal tracking. There is a deep literature comparing the various approaches, mainly based on the selection of different discriminator functions that provide enhanced performance to the receiver, for instance, in terms of multipath rejection capabilities.

Typically, these PLL/DLL schemes require three or five correlation points to operate, depending on the specific implementation. Arguably, these approaches provided a good balance between tracking performance and computational complexity.

These savings were particularly important when computational resources were scarce.

As indicated in this article, however, this correlator limitation no longer exists with modern computer platforms or is going to vanish in the near future, at least as long as we are not seeking absolute minimum power consumption, for example, to allow for background operation of a GNSS receiver.

Imagine going beyond the classical PLL/DLL architectures. The possibility of implementing multi-correlation architectures turns this dream into a reality. In recent decades we have seen a number of proposals for more sophisticated tracking and positioning techniques, most of them requiring use of multiple (i.e., more than five) correlation outputs.

Their benefits have been quantified in the literature in terms of performance enhancements with respect to legacy solutions. However, these proposals could not compete against legacy solutions in terms of complexity and feasibility of implementation given the state-of-the-art processing technologies at the time. This is not the case nowadays, and the gap will diminish quickly.

**Figure 5** shows the baseband processing diagram of a generic GNSS receiver with  $K$  channels. Here, we used the complex notation for the sake of clarity, thus avoiding I&Q duplicated arms in the diagram. Our focus is on the tracking and positioning operations, and acquisition is thus assumed as a previously accomplished stage.

In the digital domain, a closed-loop receiver has to perform carrier wipe-off and code correlation, which is fed to a digital signal processing (DSP) block implementing the chosen tracking/positioning algorithm. Notice that we have  $L$  correlation outputs, although this number could vary among channels.

The diagram in Figure 5 is sufficiently generic to represent legacy and advanced tracking techniques. For instance, whether the outputs from the  $K$  channels are processed independently or jointly is handled at the DSP module. The former method corresponds to the usual PLL/DLL schemes, where each channel tracks one satellite (with  $L \leq 5$ ). Then, an additional navigation filter is in charge of delivering position, velocity, and time (PVT) estimates after computing the necessary observables.

Other more sophisticated techniques fall into this category, such as the Multipath Estimating DLL (MEDLL) (see the article by R. Van Nee *et alia*) or the Multipath Estimating Particle Filter (MEPF) (see P. Closas *et alia*, 2009b). Both techniques share the idea of jointly estimating the parameters of the line-of-sight signal and those from the multipath echoes, with the goal of mitigating the effects of the latter on the former. Their main difference is in the statistical principle used to derive the estimators: whereas MEDLL is based on the maximum likelihood (ML) principle, the MEPF resorts to Bayesian filtering theory. Their implementation is consequently different, but they share the fact that  $L$  should be larger than the values considered in PLL/DLL schemes.

Roughly, several dozens of correlators per channel should be considered when implementing these techniques for full exploitation of their capabilities and enhanced performances (see the article by A. Fernández *et alia*). Consequently, the computa-

tional resources required are moderately large, particularly when a large number of particles are used in MEPP. However, the intrinsic parallelization properties of particle filtering should benefit from the computational advances in this field, with the ability to share the load among various processing resources, e.g., the many cores of a GPU.

On the other hand, the receiver depicted in Figure 5 can also accommodate combined tracking structures where the outputs from the  $K$  channels are jointly processed. The output of this processing is the control signal to drive the tracking loops and the PVT solution. In this category we find vector tracking loops (VTL), a convenient modification of the usual PLL/DLL schemes that allows for exploiting the synergies among channels. Therefore, the number of correlators in VTL schemes is driven by the underlying PLL/DLL techniques ( $L \leq 5$ ), and its bottleneck is not in this part of the baseband processing chain.

An alternative, more general approach has been proposed under the name of direct position estimation or DPE described in the article by P. Closas *et alia*, 2009a. DPE was initially derived under the ML principle, but incorporated an implementation based on Bayesian filtering methods as well. Therefore, open- and closed-loop architectures are possible for DPE, in contrast to VTL.

In open- and closed-loop architectures, DPE was seen to enhance the per-

formance of legacy receivers in terms of multipath mitigation, operation in weak signal conditions, or other challenging situations. As a payoff, DPE needs a receiver capable of computing a larger number of correlator outputs  $L$  (on the order of 10s of them, as in the channel-per-channel advanced techniques mentioned earlier).

With regards to important operations to be performed, a fully open-loop DPE scheme must solve a multivariate optimization, which involves systematic evaluation of an operation with complexity asymptotically proportional to  $K^2$ . This systematic evaluation could be relaxed in closed-loop DPE schemes (for instance using particle filtering) (see P. Closas *et alia*, 2010), in which case this computation could be parallelized.

### Conclusions

The increase of computational power can be used to build more efficient GNSS software receivers, and our experimental benchmarks show how well we can exploit the currently available technology for these purposes.

We found that GNSS signal acquisition dramatically benefits from executing the FFT operation on a GPU. High-volume data exchange between the CPU and GPU can cause notable delays, making the GPU suitable for signal acquisition that occurs only occasionally. Performing the acquisition on the GPU also has the benefit of de-stressing the

CPU and enabling more resources for tracking.

On a PC, a homogeneous framework exists with hardware and software optimized for each other and millions of effective correlators can be realized.

In the embedded world, we realize that FFT libraries and GPU hardware exist, which can more or less be brought together to compute FFTs. Although the number of achievable correlators is still impressive (more than 50,000), we noticed that a performance of only ~1 GFlop/sec was achieved, with a theoretical limit of 68 GFlop/sec supported by the hardware. However, we strongly believe that this gap can be significantly reduced, with a potential speed-up of at least an order of magnitude. Furthermore, it turned out that executing several FFTs in parallel did not show a big performance gain over that of at least two FFTs. This is an issue to be discussed within the open source cFFT library project in the context of usage for the ARM Mali GPU.

The tracking and correlation code can and has been designed without making use of dedicated libraries. Correlation with 1-bit or 16-bit samples is directly supported by SIMD instructions of either the Intel or the ARM architecture. On a PC the maximum number of channels is well above one thousand, with the embedded board containing less than one hundred. Remarkably, the low-power Arndale Octa embedded

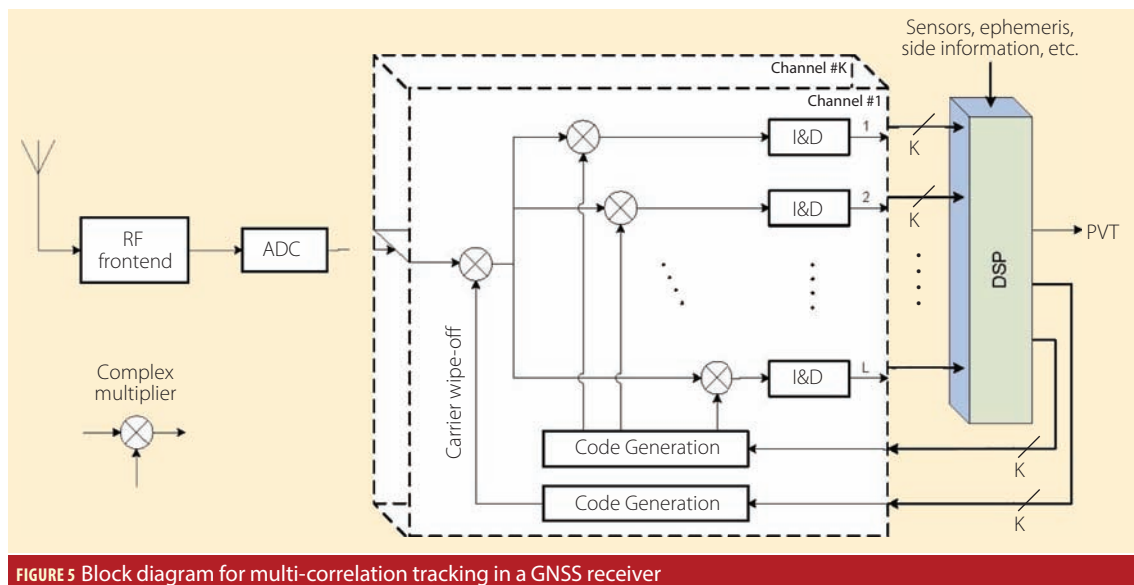


FIGURE 5 Block diagram for multi-correlation tracking in a GNSS receiver



platform would support up to 25 channels with, e.g., 21 correlators covering a rather large ranging uncertainty of  $\pm 585$  meters for high bandwidth signals. These signals could be used for DPE or other advanced tracking or positioning techniques.

We also pointed out that modern techniques coping with well-known impairments have appeared in recent years, promising important improvements at the cost of increased computational burdens. We discussed a brief sample of these techniques, which served to motivate the ideas that: a) multicorrelation strategies are the desired choice in most advanced techniques, and b) these techniques often require complex matrix operations to be computed, which sometimes can be parallelized. Both points are doable either with the current technology or that to be expected to appear in the near future.

We will continue our efforts to better understand and optimize algorithms especially for embedded devices. We expect that acquisition can be optimized by a factor of 5-10 and tracking by a factor of 2 even using the currently available embedded development boards. This will be achieved by controlling the CPU clock frequency, thus avoiding automatic frequency reduction by the Linux kernel, and better exploitation of the CPU and GPU memory caches.

## Acknowledgment

The research leading to a portion of these results has received funding from the European Space Agency (ESA) under ESA Contract No. 4000111113/14/NL/CBi/fk (eSTAR).

ESA is investigating software receiver technology for its application within a spaceborne receiver. The use of GNSS for space applications is manifold and includes launch monitoring or precise injection of satellites into geostationary orbits, applications which exhibit high user dynamics. Standard GNSS space applications include positioning in low Earth orbits or on geostationary orbits where centimeter-level accuracy can be achieved.

Formation flying or rendezvous and docking with the help of GNSS is

demanding in terms of the required accuracy. Navigation in geostationary orbits, on highly elliptical orbits, or navigation to and from the Moon requires a very high sensitivity of the GNSS receiver. All these applications could potentially benefit from a software receiver-based approach fully optimized for a specific purpose.

## Additional Resources

[1] Advanced Micro Devices, Inc., "cIFFT," GitHub, n.d., Web <<https://github.com/clMathLibraries/cIFFT>>, February 2015

[2] Akos, D. M., *A Software Radio Approach to Global Navigation Satellite System Receiver Design*, Ph.D. Dissertation, Ohio University, Columbus, Ohio, August 1997

[3] Broadcom Press Release: Broadcom Announces Industry's First GNSS Location Hub Chip for Smartphones to Support Galileo Satellite System, <<http://www.broadcom.com/press/release.php?id=s885589>>

[4] Closas, P., *Bayesian Signal Processing Techniques for GNSS Receivers: From Multipath Mitigation to Positioning*, PhD Dissertation, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, June 2009 (2009a)

[5] Closas, P., and C. Fernández-Prades and J. A. Fernández-Rubio, "A Bayesian Approach to Multipath Mitigation in GNSS Receivers," *IEEE Journal of Selected Topics in Signal Processing*, special issue on *Advanced Signal Processing for GNSS and Robust Navigation*, Volume: 3, Issue: 4, pp. 695-706, August 2009 (2009b)

[6] Closas, P., and C. Fernández-Prades, "Bayesian Nonlinear Filters for Direct Position Estimation," *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 2010

[7] Dabove, P. and M. Petovello, "What are the Actual Performances of GNSS Positioning using Smartphone Technology?" *Inside GNSS*, November/December 2014

[8] Fernández, A., and M. Wiz, P. Closas, C. Fernández-Prades, F. Zanier, R. Prieto-Cerdeira, and M. Crisci, "ARTEMISA: Preliminary Results of Advanced Receiver Techniques for Multipath Mitigation," *Proceedings of the ION GNSS 2012 Meeting*, Nashville, Tennessee USA, September 2012

[9] nVidia, "cuFFT," nVidia CUDA ZONE, nVidia, n.d., <<https://developer.nvidia.com/cuFFT>>, February 2015

[10] Pany, T., *Navigation Signal Processing for GNSS Software Receivers*, Artech House, ISBN: 978-1-60807-027-5, 2010

[11] Pany, T., and S.W. Moon, M. Irsigler, B. Eissfeller, and K. Furlinger, "Performance Assessment of an Under-Sampling SWC Receiver for Simulated High-Bandwidth GPS/Galileo Signals and Real Signals," *Proceedings of the ION-GNSS 2003 Meeting*, Portland, Oregon USA, pp. 103-116, September 2003

[12] Top500 Project <[www.top500.org](http://www.top500.org)>

[13] Van Nee, R. D. J., and J. Sierveld, P. C. Fenton, and B. R. Townsend, "The Multipath Estimating Delay Lock Loop: Approaching Theoretical Accuracy Limits," *Proceedings of the Position Location and Navigation Symposium 1994*, pp. 246-251, April 1994

## Authors



**Thomas Pany** works for IFEN GmbH as a senior research engineer in the GNSS receiver department. He also works as a lecturer (Priv.-Doz.) at the University FAF Munich and for the University of Applied Sciences in Graz. He obtained a Ph.D. in geodesy from the Graz University of Technology, in Austria. His research interests include GNSS receivers, GNSS/INS integration, signal processing, and GNSS science.



**Wolfgang Bär** is head of mobile solutions department at IFEN GmbH since 2014. He joined IFEN in 2006 as a system engineer. Previously he was a research associate at the Institute for Geoinformatics and Remote Sensing of the University of Osnabrueck where he received his Ph.D.



**Jürgen Dampf** received his M.Sc. in aviation engineering at the University of Applied Sciences in Graz. Since 2012 he has for IFEN GmbH as a system engineer emphasizing GNSS reflectometry, particle filters, and embedded software receivers.



**Jón Ó. Winkel** is head of receiver technology at IFEN GmbH since 2001. He studied physics at the universities in Hamburg and Regensburg. He received a Ph.D. (Dr.-Ing.) from the University of the FAF in Munich in the field of GNSS modeling and simulations.



**Carsten Stöber** works for IFEN GmbH as a system engineer. He received his diploma in geodesy at the Technical University in Berlin. For seven years he has been research associate at the Institute of Space Technology & Space Application of the University of the Federal Armed Forces in Munich.



**Karl Furlinger** is a lecturer and senior researcher at the Ludwig-Maximilians-University (LMU) Munich, working in the area of parallel and high performance computing. He received his Ph.D. from the Technical University of Munich. His research focuses on performance tools and

**Working Papers** continued on page 72

# Editor Glen Gibbons Honored by RIN for GNSS Journalism

**I**nside GNSS editor and publisher Glen Gibbons has been honored by the United Kingdom's Royal Institute of Navigation (RIN) for his "outstanding contribution to navigation" as a journalist and publisher.

Gibbons received the 2015 Harold Spencer-Jones Gold Medal during the institute's annual meeting at the Royal Geographical Society in London. As this issue went to press, the medal was to be presented to Gibbons on July 15 by Prince Philip, Duke of Edinburgh and husband of Queen Elizabeth II. The duke is the patron of the institute.

"Glen Gibbons has probably done more than anyone to raise general awareness and understanding of the emergent satellite navigation technology, including its capabilities and limitations," reads the citation accompanying the medal.

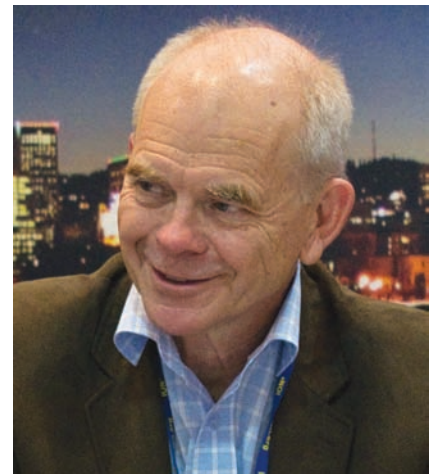
RIN director Peter Chapman-Andrews said the medal is the institute's highest award, which is never made more

than once per year and sometimes not at all. Gibbons shared the spotlight with the European Space Agency and other RIN award recipients. The duke recognized ESA with his own technical award for navigation for the agency's successful comet-chasing Rosetta Mission.

As probably the world's first full-time GPS/GNSS journalist, Gibbons publishes the international trade journal *Inside GNSS* and associated media products. He and his wife Eliza Schmidkunz started the technical magazine for engineers, designers, program managers, and policy makers nearly 10 years ago in Eugene, Oregon USA.

He began his career covering GNSS applications and policies in 1989 as the founding editor of *GPS World*, the first trade magazine on the subject, established in Eugene by the Aster Publishing Company.

Over the years, Gibbons and *Inside GNSS* have often been the first outside of scholarly journals to cover news and



Glen Gibbons

issues of all of the space-based positioning, location, timing and navigation systems. At *Inside GNSS*, these have included such subjects as GLONASS revitalization, the decision for the common GPS/Galileo civil signals, the transformation of EU's Galileo from a public-private partnership to a European Commission-controlled program, the first analysis of the new BeiDou signal, the patent dispute between the United States and the European Union over modernized signal development, the formation of the International Committee on GNSS (ICG) by the United Nations, and the first Galileo-only signal analysis.

The RIN award is named for Sir Harold Spencer-Jones, the UK's astronomer-royal in the 1930s and 1940s, who determined the accurate median distance from Earth to the Sun. He was also the first president of the RIN.

## Working Papers continued from page 70

all aspects parallel programming, algorithms, and systems. Before joining LMU Munich he was a postdoctoral researcher at the University of California at Berkeley, and at the NERSC super-computing center and prior to that he was a senior research associate at the University of Tennessee at Knoxville (UTK).



**Pau Closas** received the M.Sc. and Ph.D. in electrical engineering from the Universitat Politècnica de Catalunya (UPC). He also holds a M.Sc. degree in advanced mathematics and mathematical engineering from UPC. Currently he is senior researcher and head of department at the Centre Tecnològic de Telecomunicacions de Catalunya (CTTC) in Barcelona. His expertise is in statistical signal processing with applications to GNSS receiver design, indoor positioning, and wireless communications. He is the recipient of the EURASIP Best Ph.D. Thesis Award 2014 and the ninth Duran Farell Award for Technology Research, both for his contributions in the areas of signal processing and GNSS.




**J. A. Garcia-Molina** is a radio-navigation engineer at the European Space Agency/ESTEC in Noordwijk, The Netherlands. His primary areas of interest include signal processing, estimation

theory, GNSS receivers, and signals and navigation applications.



**Prof.-Dr. Günter Hein** serves as the editor of the Working Papers column. He served as the head of the EGNOS and GNSS Evolution Program Department of the European Space Agency

and continues to advise on scientific aspects of the Navigation Directorate as well as being a member of the ESA Overall High Level Science Advisory Board. Previously, he was a full professor and director of the Institute of Geodesy and Navigation at the Universität der Bundeswehr München. In 2002, he received the Johannes Kepler Award from the U.S. Institute of Navigation (ION) for "sustained and significant contributions" to satellite navigation. He is one of the inventors of the CBOC signal. 

## NovAtel to Supply QZSS Reference Receivers

**N**ovAtel Inc. has announced an agreement with NEC Corporation to supply reference receiver products for use in Japan's Quasi-Zenith Satellite System (QZSS).

The QZSS equipment will be based on NovAtel's third-generation (G-III) family of reference receivers. Designed for integrity monitoring and reference measurement applications, the receivers