Getting better all the time.
The Continued Evolution of the GNSS Software-Defined Radio

James T. Curran, Carles Fernández-Prades, Aiden Morrison, and Michele Bavaro

1 Introduction

Software Defined Radio (SDR) has an infinite number of interpretations depending on the context in which it is designed and used. By way of a starting definition the authors choose to use that of 'a reconfigurable radio system whose characteristics are partially or fully defined via software or firmware'. In various forms, SDR has permeated a wide range of user groups, from military, business, academia and to the amateur radio enthusiast.

It has evolved steadily over the decades following its birth in the mid 1980s, with various surges of activity being generally aligned with new developments in related technologies (processor power, serial busses, signal processing techniques, SDR chipsets). At present, it appears that we are experiencing one such surge, and GNSS SDR is expanding in many directions. The proliferation of collaboration and code-sharing sites such as GitHub has enabled communities so share and co-develop receiver technology; the rise in the maker-culture and crowdsourcing has led to the availability high-performance radio-frequency front-ends; and the adoption of SDR by some major telecommunications industry has led to the availability suitable integrated circuits. Combined, these contributing factors have seen an increased uptake of GNSS SDR in military, scientific, and commercial applications. In this article, we explore some of the recent trends and the technology behind them.

2 SDR Topologies

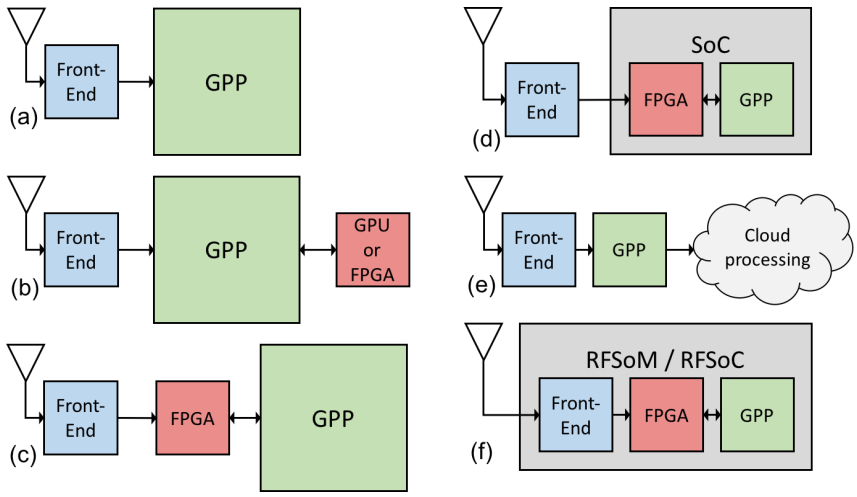

Figure 1: A simplified depiction of different SDR topologies

Software defined radio for GNSS has evolved over the past decade, both in terms of the adoption of new frequencies, new signals and new systems, as they have become available;

as well as the adoption of new processing platforms and their associated processing techniques. Depicted in Figure 1 is a (simplified) depiction of how the topology of the software-defined GNSS receiver has evolved over the years (a-d) with a hint at where it might go next (e,f).

In a traditional SDR GNSS receiver, as depicted in Figure 1 (a), the radio-frequency front-end typically interfaces with the general-purpose processor (GPP) through a standard bus, and intermediate-frequency (IF) samples are streamed to a buffer. Once on the GPP, basic operations such as correlation, acquisition/tracking, measurement generation and positioning were performed.

Of all of the operations performed by a GNSS receiver, correlation is (by some orders of magnitude) the most computationally intensive. However the correlation operations are relatively simple, often requiring only integer arithmetic, and can be easily parallelized. When running on modern processors, optimized software receivers can avail of multithreading (task parallelism) or the operations can be vectorized to exploit data parallelism (single-instruction, multiple data).

Beyond a certain number of GNSS signals, and a certain bandwidth, a GPP simply cannot cope, and many SDR receivers looked to hardware-acceleration for the correlation process. This either took the form of a graphics processing unit (GPU), or a field-programmable gate-array (FPGA), as depicted in Figure 1 (b), both of which are well suited to highly parallel tasks. These processing platforms can be powerful, and efficient, and so can almost alleviate all challenges associated with correlation. This is not the only way to alleviate the processing burden, as it also possible to delegate the correlation task to a network of computers. This `cloud' receiver architecture, depicted in Figure 1 (e) received particular attention of late, showing promise for certain niche applications. This trend has partially reverted with the proliferation of many-core desktop and mobile processors, but at a certain level of signal or processing complexity the extensions remain applicable.

At this point, data throughput becomes an important consideration. When considering multi-constellation, multi-frequency receivers, the objective is often to preserve signal quality, which implies high bandwidth and high digitizer resolution. A triple-frequency front-end might easily produce in excess of 100 or even 500 MB/s. When this data is delivered to the GPP or somewhere in the host computer, and then offloaded to the GPU (or any other hardware acceleration), it might be `handled' twice, exacerbating the bottle-neck. To overcome this problem (and for other practical architectural reasons) it can be preferable to interface the front-end directly with the accelerator, where correlation was performed, and leave the `brains' of the receiver (loop-closure, data processing, PVT, etc.) on the GPP. This is a particularly convenient approach when using an FPGA accelerator, as is shown in Figure 1 (d).

A very similar architecture can be achieved using modern system-on-chip (SoC) integrated circuits (ICs), which can offer a large FPGA and a powerful GPP on the same silicon, as depicted in Figure 1 (d). Indeed, a number of receivers using this architecture have seen commercial and scientific success, having many of the benefits of dedicated silicon while retaining the benefits of software-defined radio, for example the Swift Navigation Piksi

Multi GNSS Module. Recent developments in the field have seen the world's first radio-frequency system-on-module (RFSoM), or system-on-chip (RFSoC), targeting 5G applications. With an architecture similar to that of Figure 1 (f), the IC touts up to 8x8 MIMO with 12 bit ADCs/DACs running at rates of 2/4 Gs/s. Depending on how this trend evolves (assuming `lighter' versions become available), this might offer an exciting new platform for SDR for GNSS, simultaneously capable of multifrequency and multi-antenna operation.

3 RF Hardware: The enabler

GNSS SDR `sees' the world through a hardware peripheral, and the capability of this hardware defines the perimeter between what the receiver can and cannot do. In essence, the front-end peripheral converts one or more analog RF signals at the antenna to a stream or sequence of packets of digital baseband/IF data to the GPP.

Software-defined radio for GNSS benefits greatly from being flanked on both sides by signals that are of interest to the civilian population. Applications such as DVB-T/DVB-S2 receivers have resulted in the availability of a wide range of low-cost RF ICs that are tunable to GNSS (typically spanning 900 MHz to 2.1 GHz) which, along with dedicated GPS ICs, were at the heart of early GNSS SDR front-ends. Later developments in ICs designed around the 2/3/4G mobile communications standards brought another generation of ICs, bringing higher instantaneous bandwidth, higher ADC resolution and MIMO, and re-transmit capability. With the the increase in popularity of software defined radio for cognitive-radio, WiFi, 3G, LTE, and enjoying the benefits of a crowdfunding movement, a wide range of front-end peripherals quickly appeared. Many of these front-ends are compatible with GNSS, offering significantly increased performance relative to their predecessors. A selection of some GNSS-compatible SDR peripherals (both new and old) is shown in Table 1.

Front-End	Interface	Freq. Range	Bandwidth	ADC Resolution	IC	Mode
ADALM-Pluto	USB 2.0	325 MHz - 3.8 GHz	61.44 MHz	12 bit I/Q	Analog Devices AD9363	Full Duplex
Airspy Mini	USB 2.0	24 MHz - 1.8 GHz	6 MHz	12 bit I/Q	Rafael Micro R820T2	Rx
BladeRF	USB 3.0	300 MHz - 3.8 GHz	40 MHz	12 bit I/Q	LimeMicro LMS6002D	Full Duplex
FreeSRP	• USB 3.0g	70 MHz - 6 GHz	61.44 MHz	12 bit I/Q	Analog Devices AD9364	Full Duplex
HackRF	USB 2.0	1 MHz - 6 GHz	20 MHz	8 bit I/Q	Maxim MAX2837 / 5864	Half Duplex
LimeSDR	USB 3.0/PCIe	100 kHz – 3.8 GHz	61.44 MHz	12 bit I/Q	Lime Micro LMS7002M	Full Duplex
RTL-SDR	USB 2.0	24 MHz - 1.766 GHz	2.4 MHz	8 bit I/Q	Realtek RTL2832U	Rx
SDRplay	USB 2.0	1 kHz - 2 GHz	6 (10) MHz	14 (8) bit I/Q	Mirics Msi001	Rx
Sidekiq	Mini PCIe	70 MHz - 6 GHz	50 MHz	12 bit IQ	Analog Devices AD9361*	Full Duplex
SiGe GN3S	USB 2.0	1.57542 GHz	8 MHz	4 bit I/Q	SiGe SE4110L	Rx
USRP N210	Gigabit Eth.	0 - 6 GHz	25 MHz (50 MHz)	14 (8) bit IQ	Analog Devices AD9361	Full Duplex

USRP X310	Giga. Eth.	0 - 6 GHz	200 MHz	14 bit IQ	Analog Devices AD9361	Full Duplex
XTRX	Mini PCIe	100 kHz – 3.8 GHz	160 MHz	14 bit IQ	Lime Micro LMS7002M	Full Duplex

Table 1. A selection of GNSS-compatible SDR front-ends.

3.1 Reference Oscillators

Although many of the requirements of modern telecommunications ICs are beyond what is needed for GNSS (i.e. ADC resolution, frequency-range, bandwidth, linearity), clock stability is often inadequate. Communications signals are generally received at high SNR and so the carrier can be easily recovered, even given very poor clock stability.

In contrast, clock stability can be critical for GNSS applications, due to comparatively long coherent integration period (i.e. 1+ ms). Firstly, because the `search-space' granularity is related to the integration period and the size of the search space to the frequency uncertainty, clock accuracy is important, as an uncertainty of some tens of kHz might increase acquisition time. Secondly, the short-term stability is important as a large amount of phase wander can be challenging when attempting to track the carrier phase with a loop-update rate below 1 kHz. In fact, this issue was so pronounced on early RTL-SDR DVB-T front-ends, that later revisions upgraded the quartz reference oscillator to a more respectable 0.5 ppm temperature-compensated crystal oscillator (TCXO). Typically a TCXO with an accuracy of better than 1 ppm is preferable, but this metric alone is far from sufficient.

Depending on the class of signals which the SDR front-end will be used for, the characteristics of the oscillator, the configuration of its support electronics, and even whether the mixers and analog to digital conversion process utilize the same reference can vary. For example, not all TCXOs are suitable for GNSS applications due to the way in which they internally apply their temperature compensations. If a given TCXO uses a stepwise compensation configuration based on any form of digital feedback the size of the resulting steps can severely impact the GNSS tracking loops. Even if a given TCXO has a suitable compensation curve and implementation, as well as a low and acceptable intrinsic phase noise, every other link in the clock 'chain' must preserve this performance. In some front-end implementations swapping out a low quality clock for a higher quality one is sufficient, but on others there can be design limitations in one or more of the oscillator power supply, the oscillator signal conditioning, subsequent clock generation steps, or distribution routing that can prevent the design from ever being suitable for GNSS use. This can be critical in cases where the carrier phase is of interest, for example, where phase coherence between channels is important for multi-frequency linear combinations, or for multi-antenna systems.

Fortunately, many modern SDR front-ends support the use of an external clock. This feature can also be important when attempting to combine two front-end peripherals to effect a dual-frequency or dual-antenna software receiver.

3.2 The BUS

An intrinsic bottleneck for any SDR system is the fact that some form of connection or bus is needed to carry data from the collection point to the processing element. In a fully

integrated system this connection still exists, but is typically a trace on a circuit board or even within an integrated device. In contrast, in an SDR this often takes the form of a cable or connector between the physically discrete system modules. In cases where the devices are discrete, it is often necessary to implement some data buffering on both ends of the bus.

The suitability of a particular bus is often determined by the sustained data throughput rate required by the application, and in some cases, the latency of the bus. An example of a number of interfaces popular in modern SDR front-ends is shown in Figure 2, illustrating the nominal throughput and the minimum latency of each. In the case of GNSS SDR, the minimum conceivable throughput required would be hundreds of MB/s, but a system could easily use in excess of 200 MB/s for multi-frequency, high bit depth data. Of course, in post-processing applications, bus latency is not a factor, however in certain applications may require that this latency is small, or bounded, or somehow deterministic. Applications such as closed-loop vehicle control or certain safety systems might impose tight requirements on latency. High or unpredictable latency in GNSS measurements might lead to loop instability, in the case of a control system, or might erode safety margins. Although the trend in modern interfaces is for higher throughput, only certain interfaces offer low latency.

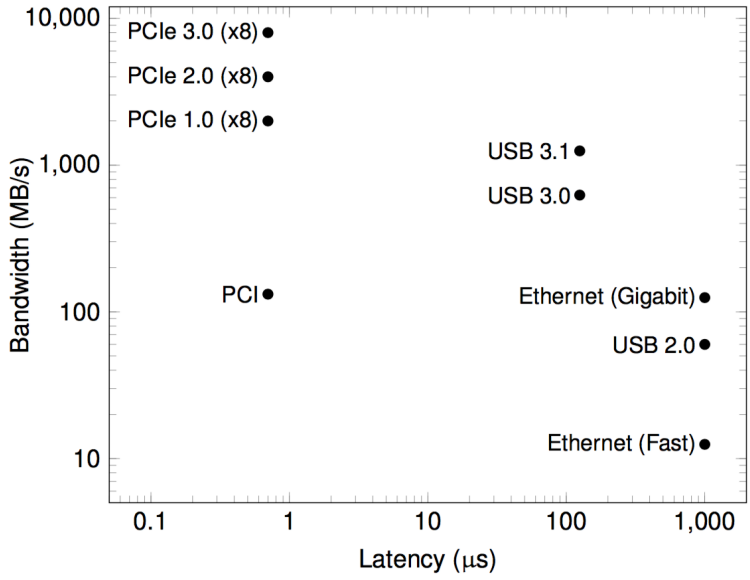

Figure 2. Bandwidth/Latency Scatter-Plot for Popular Buses

3.3 The Silicon

In comparison with less flexible fixed function GNSS receiver chips, GNSS SDR hardware platforms provide the opportunity to exchange 1-3 orders of magnitude of power consumption and system size to gain substantial control over the characteristics of the design. Moreover, one of the other main differences between GNSS front-ends and general purpose SDR front-ends are the number of bits and the linearity. Both contribute to power consumption. However, may be worth considering that GNSS-specific front-ends have not received as much attention as comms front-ends and, consequently, there is at least a generational gap in silicon mask technology (most GNSS products are at the 350 nm level).

In terms of GNSS-specific devices, products such as the SiGe SE4110L, the Maxim MAX2769, Saphyrion's SM1027U, providing a solution for slightly flexible L1 GPS, Galileo or in some chip revisions GLONASS operation, these kinds of chips support a few sampling rates and filtering configurations.

In the middle ground are the much more flexible chips from Maxim including the MAX2120 and MAX2112 which provide total L-band coverage, a myriad of filtering options, and adjustable gain control all within a 0.3 Watt power budget per channel (RF portion only). These chips allow for single band coverage of adjacent GNSS signals such as GPS and GLONASS L1 or L2 in a single non-aliased RF band.

In terms of multi-channel options, devices such as the Maxim MAX19994A, or the NTLab NT1065, respectively offer dual, or quad channel functionality. Similar functionality can be achieved by pairing downconversion and IF receiver ICs, such as, for example, the Linear Technologies, LTC5569 dual active downconverting mixer, and the Analog Devices AD6655 IF receiver, which might offer sufficient performance for high-accuracy dual-frequency positioning.

Higher up the cost, power and complexity structure radios designed explicitly to support SDR applications that happen to cover GNSS bands such as the Lime LMS6002d/LMS7002M or the Analog Devices AD9364. Notably, these provide RX and TX channels and frequency coverage up to 6 GHz. While another interesting and relevant trend is in the use of direct RF sampling ICs, which offer the possibility of full L-band coverage and/or multi-antenna support. Examples of these include the Texas Instruments ADS54J40, which offers dual-channel, 14-bit, 1.0-GSPS ADC, or the ADS54J40 offering a 7.6 bit, quad-channel, 1.25 GSPS, ADC.

3.4 Future Trends, Limitations & Opportunities

The majority of innovation in software defined radio peripherals has taken place in the telecommunications domain, focusing on systems like DVB, WiFi, 2/3G, BLE, and more recently 4/5G. Because of the large community that such a wide spread of technologies can attract, efforts to develop low-cost and high-performance SDR transceivers have enjoyed support from crowdsourcing or venture capital. The GNSS SDR community, being comparatively small, has benefited from these innovations, insofar as they were applicable, but has had little influence over the design.

Looking at the bigger picture, it is clear that GNSS SDR will simply have to follow the road paved by telecoms SDR. We will have to use what is made available, and so future trends in GNSS SDR will likely be driven by the needs of the telecoms SDR community.

So what are they, and will they be aligned with GNSS trends? The answer seems to be yes, and no. One of the bigger trends in modern GNSS receivers is the move to dual or multi-frequency and a second trend is towards multi-antenna receivers for attitude determination, or multi-element antennas for interference management. Meanwhile telecoms are almost universally using multi-input multi-output (MIMO) transceivers, however they don't seem to be using multiple (simultaneous) carriers.

What is particularly interesting, is that the requirements for a MIMO transceiver are well aligned with that of a null-steering antenna: namely high linearity and ADC-resolution, and phase-coherence between channels (for example, the Lime Microsystems LMS7002M or the Analog Devices AD9361). As a result, it is possible (or even likely) that we will see more innovation in GNSS SDR in the area of multi-antenna processing than in multi-frequency processing in the near future.

4 Signal Processing Techniques for SDR

As mentioned above, the signal correlation for the acquisition/tracking is the most computationally intensive operation conducted by the GNSS receiver. In software receivers, many signal acquisition strategies are built around FFT algorithm, while signal tracking `rake' of 3 or more correlators per signal. When targeting real-time processing, these operations need to be applied to a stream of signal samples arriving at a rate of many Msps. This is a challenge for GPPs when implementing a multi-constellation, multi-frequency GNSS receiver.

The processing task can either be alleviated or accelerated. Assistance data can allow the receiver to skip or reduce the search acquisition space, thereby dramatically reducing the overall computational load. In many cases, the software receiver is running on a host computer with many connectivity options. Alternatively, a variety of options are available for accelerating the tasks.

4.1 Parallelization

The main approach to accelerate GNSS signal processing is parallelization. Shared-memory parallel computers can execute different instruction streams (or threads) on different processors, or by interleaving multiple instruction streams on single processor (simultaneous multithreading, or SMT), or both. This approach is referred to as task parallelism, and is well supported by the main programming languages, compilers and operating systems. This approach fits naturally with the architecture of a GNSS receiver, which has many channels (one per satellite and frequency band) operating in parallel over the same input data. When programmed with the appropriate design, execution can be accelerated almost linearly with the number of processing cores. However, the spreading of processing tasks along different threads must be carefully designed in order to avoid bottlenecks (either in the processing or in memory access).

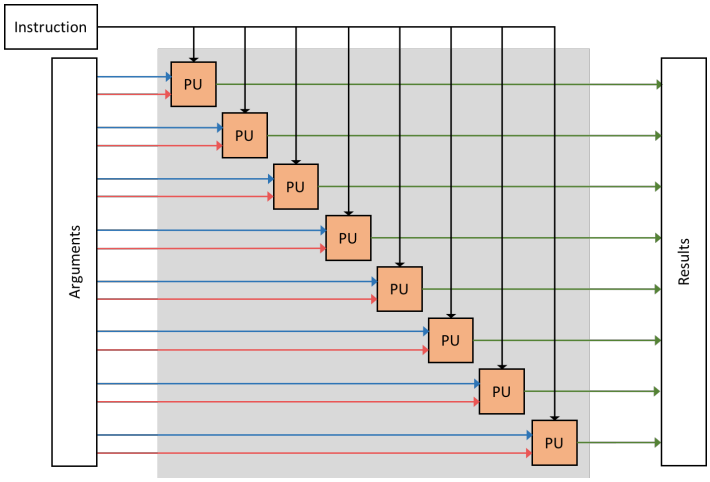

Figure 3: Illustration of the operation of SIMD processors, which take a multiple data (arguments) and produces multiple results, given a single instruction operated in parallel in a set of processing units (PUs).

In combination with task parallelization, software-defined receivers can still resort to another form of parallelization: instructions that can be applied to multiple data elements at the same time, thus exploiting data parallelism. This computer architecture is known as Single Instruction Multiple Data (SIMD), where a single operation is executed in one step on a vector of data, as illustrated in Figure 3. In GNSS receivers this type of instruction can implement operations like multiply-and-accumulate in across multiple (16, 32, 64, etc.) samples in a single clock-cycle. Intel introduced the first instance of 64-bit SIMD extensions, called MMX, in 1997. Later SIMD extensions, (SSE 1 to 4), added multiple 128-bit registers. AMD, quickly followed and SIMD is now present in almost all modern processors. Later, Intel introduced more new instruction sets called Advanced Vector Extensions (AVX) featuring, 256-bit registers, new instructions and a new coding scheme. In 2013 AVX-2 expanded most integer commands to 256 bits, by 2016 the introduction of AVX-512 provided 512-bit extensions. SIMD technology is also present in embedded systems: NEON technology is a 128-bit SIMD architecture extension for the ARMv7 Cortex-A series processors, providing 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide), and AArch64 NEON for ARMv8 processors, which provides 32 128-bit registers. In many cases, well written code will be automatically implemented as some combination of these SIMD `intrinsics'. In other cases, they can be coded explicitly.

4.2 Hardware Acceleration

Another possibility for accelerating signal processing is to offload computation-intensive portions of the workload to a device external to the main GPP executing the software. This is the case of graphics processing units (GPUs). Such processor architecture follows another parallel programming model, called Single Instruction, Multiple Threads (SIMT). While in SIMD elements of short vectors are processed in parallel, and in SMT instructions of several threads are run in parallel, SIMT is a hybrid between vector processing and hardware threading. Currently, OpenCL is the most popular open GPU computing language that supports devices from several manufacturers, while CUDA is the dominant proprietary framework specific for NVIDIA GPUs. The key idea is to exploit the computation power of both GPP cores and GPU execution units in tandem for better utilization of available computing power. The main constraint in using GPUs is memory bandwidth. If not

programmed carefully, most of the time will be spent on transferring data back and forth between the GPP and the GPU, instead of in the actual processing. A possible solution to this is an approach known as "zero copy" operations, which consists of a unified address space for the GPP and the GPU that facilitates the passing of pointers between them, thus reducing the memory bandwidth requirements.

Similar benefits can be had by offloading correlation to reconfigurable hardware such as FPGAs. The correlation duties can be offloaded to FPGA and the loop-closure and navigation engine can remain in the GPP. The FPGA is particularly well suited to the GNSS correlation tasks and can implement dedicated low-resolution (e.g. 1-4 bit) multiply-and-accumulate blocks, where the equivalent 8, 16, or 32 bit operations on a GPP would be excessive or inefficient. Early approaches involved an FPGA connected as a peripheral device via Ethernet, PCIe or a similar bus. However, similar to the GPU, the data transfer quickly becomes a bottle-neck. This challenge is addressed by integrating the GPP-FPGA packages. Early examples of this being the Intel Atom E6x5C package hosting an Altera FPGA. More recent examples are the Xilinx's Zynq 7000 family integrating ARM and FPGA processors in a single encapsulation. These SoCs allow the direct injection of signal samples from the RF front-end into the FPGA, greatly reducing the amount of information to be interchanged with the GPP. This approach provides flexibility with regards to how tracking and correlation resources are allocated, allowing configurable architectures according to the targeted signals of interest and application at hand, and enabling the execution of full-featured software-defined receivers in small form factor devices.

5 The Cloud

This ability to manage resources as logical entities instead of as physical, hardwired units dedicated to a given application has materialized in business models such as Software as a Service (SaaS); Platform as a Service (PaaS); and Infrastructures as a Service (IaaS). A network of software-defined GNSS receivers executed in the cloud, appears to be the next natural step in this technology trend, in which the GNSS receiver is no longer a physical device but a virtualized function provided as a service.

A virtualized software application is a program that can be executed regardless of the underlying computer platform. This can be achieved by packaging the application and all its software requirements (the operating system, supporting libraries and programs) in a single, self-contained software entity, that can be then run on any platform. An instance of a software-defined GNSS receiver executed in a virtual environment can then be called a *virtualized GNSS receiver*.

Early virtualization was in the form of full or machine virtualization (virtual machine, VM) which is a software application that emulates the hardware environment and functionality of a physical computer. With VMs, a software component called a hypervisor interfaces between the VM environment and the underlying hardware (CPU), providing the necessary layer of abstraction. A VM can run a full operating system, so conventional software applications (*e.g.*, a software-defined GNSS receiver) can run within a VM without any required change.

Recently, the use of operating system virtualization, or software containers, has become more popular, as they are often faster and more lightweight. Instead of a hypervisor, software containers use a daemon that supplements the host kernel, and can therefore be more efficient in making use of the underlying hardware. Examples of these software containers are Docker and Ubuntu Snaps. An example of an open source software-defined GNSS receiver packed as a Docker container is available at https://github.com/carlesfernandez/docker-gnsssdr.

Virtualized GNSS receivers bring important benefits in two fields: business-wise, as a technology enabler for new GNSS-based services; and also the use of SDR GNSS as scientific tools, to ensure reproducibility.

As a service enabler, virtualized GNSS receivers allow for: 1) automatic and elastic creation, execution and destruction of application instances as required, and, 2) intelligent spread of the running instances across computing resources, regardless of processor architecture, host operating system or physical location. Several solutions are reported in the technical literature, many based on the GNSS snapshot-receiver, in which a short batch of data is sent to the software for PVT computation. Notable examples of such approach are Microsoft's energy-efficient GPS sensing with cloud offloading [Liu12, Liu16] and the system running on Amazon Web Services presented in [Luc16a, Luc16b]. These approaches allow extremely low power consumption to the user equipment, at the expense of limited accuracy (ranging from 10 to 100 m of error) and high latency. In [Fer17], authors proposed a system architecture based on optical networks and automated orchestration tools to deliver continuous service with high-accuracy performance to users with high-bandwidth connectivity. Commercially, Trimble offers Catalyst, a subscription-based GNSS receiver cloud-based service in which the user is charged according to the provided accuracy level, although the exact details are not yet public.

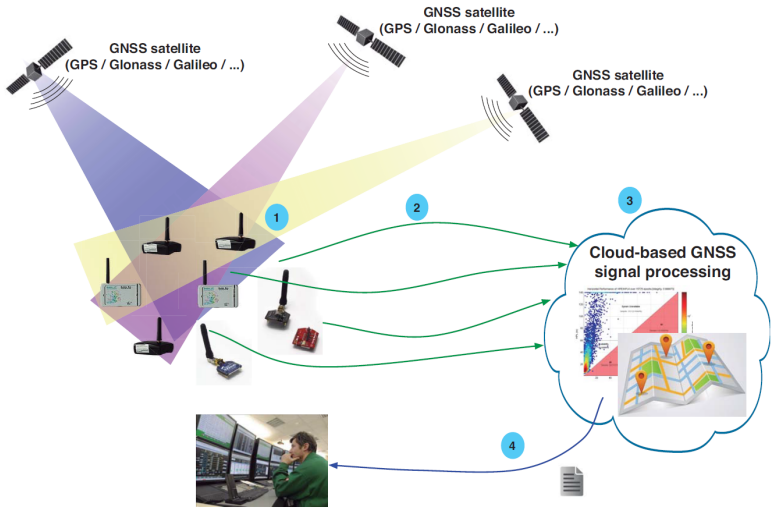

Figure 4: Illustration of cloud-based GNSS signal processing paradigm (Courtesy of SPCOMNAV, Universitat Autònoma de Barcelona)

Virtualization technologies also offer a convenient solution for security-related applications (e.g. GPS M code, Galileo PRS), since the encryption module remains on the service provider's premises, and there is no need for a security module in the receiver equipment.

This approach may enable the widespread use of restricted/authorized signals by the civilian population [NSL].

Finally, virtualization also offers important benefits for science. The flexibility of SDR receivers makes them an ideal tool for scientific experiments, since an implementation released under an open source license would allow the scientist to share a complete description of the processing from raw signal samples to the final research results. However, a key aspect in order to obtain meaningful results is *reproducibility*: the ability to reproduce an entire experiment, either by the researcher or by someone else working independently. Often software-defined GNSS receivers do not rely exclusively on their source code, and require features provided by the underlying operating system as well as a set of supporting libraries. In this context, software virtualization appears as an excellent solution for ensuring reproducibility, since it allows sharing the full software stack, data and all the scripts required to reproduce the plots from the original paper.

6 Standardization Efforts

GNSS signals are generally introduced to the front-end through a standard interface, perhaps an SMA, MCX, or U.FL connector, and the digitized signals depart through another standard interface, perhaps USB, PCIe, or RJ45. However for GNSS SDR, this is where the standardization ends. As discussed above, it is clear that there is a wide range of possibilities when capturing and digitizing GNSS spectrum. Before processing this stream of digitized samples, details such as sample-rate, center-frequency, sample resolution and format/packing, and a variety of other parameters must be established. This is particularly important in a variety of scenarios: such as when sharing/post-processing archived datasets in scientific applications; when offloading computational burden to a cloud-computer.; or when interfacing different data-capture devices with different receivers. Ad-hoc methods of digitized data formats do not encourage interoperability and instead cultivates potential for technology segmentation.

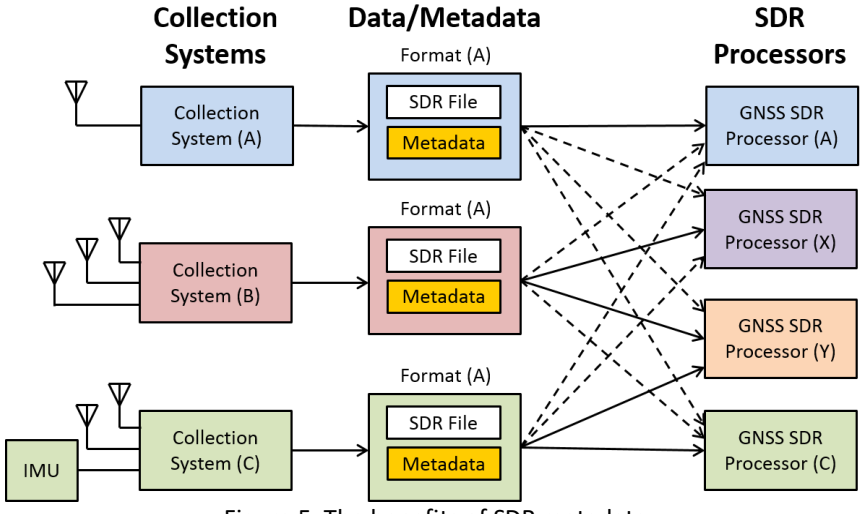

Figure 5. The benefits of SDR metadata

To address this challenge, the Institute of Navigation has lead an effort to develop a specification for a standardized metadata which would accurately and unambiguously describe the digitised data. Adoption of this metadata standard both by the data collection hardware, and the software-defined radio receiver, can promote interoperability, and can reduce the potential for error. Similarly, an SDR processor's utility is extended when it is capable of supporting many file formats from multiple sources seamlessly. An illustration of how this metadata might be processed in conjunction with the digitized samples is shown in Figure 5. For more detail on the initiative, readers are encouraged to visit sdr.ion.org.

7 New Frontiers: GNSS SDR in Space

In space, GNSS receivers need to operate in scenarios which are quite different from those of ground-based receivers: higher (albeit predictable) dynamics conditions, low signal-to-noise-density ratio and poor poor positioning geometry. It is then an excellent scenario for SDR, since it requires non-standard features from the receiver.

However space is a harsh environment for semiconductor devices. Charged particles and gamma rays create ionization, which can alter device parameters. In addition to permanently damaging to CMOS ICs, radiation may cause single-event effects, which are caused by ionizing radiation strikes that discharge the charge in storage elements, such as configuration memory cells, user memory, and registers. When those effects happen, the system is usually recoverable with a power reset or a memory rewrite, but they also may destroy the device.

Until recently, radiation-hardened solutions were limited to ASICs and one-time-programmable solutions, however recently there has been an increase in the availability of space-grade FPGA and memory devices. As examples, we can mention Xilinx's Virtex-5QV, Microsemi's RTG4 and Atmel's ATF80 FPGA processors, and commercial SDR platforms such as GOMspace's GOMX-3. Those devices allow the implementation of space-qualified GNSS receivers fully defined by software.

SDR receivers offer both reprogrammability (or upgradeability) and self-healing (or auto-remediation) capabilities. Examples could be the possibility to upload algorithms yet-to-be-invented at the receiver's launch time, or the ability to recover from single-event effect by remotely rewriting damaged functionalities, reducing the need of onboard redundancy.

8 The Economics of SDR

Flexibility has a cost - and more flexibility costs more. This is why an FPGA implementation of a complex system can never compete with the unit-cost with a fixed function ASIC.

An example of a virtuous overlap might be seen in the Maxim 2120 and 2112 line of DVB-S2 TV receiver ICs, which have been successfully co-opted for GNSS SDR front-ends due to their features (configurable mixers, gains, filters, operating power range etc.) happen to be a 'good enough' match for the GNSS domain. On initial inspection, this allows for flexibility between the two applications spaces and provides an ideal platform for SDRs supporting both TV decoding or GNSS on the same hardware radio module, but soon problems appear.

The MAX21xx series are designed for TV applications, and TV applications tend to use 75 ohm impedances while GNSS has standardized on 50 ohms. Certainly one could add a 'software defined' impedance selector block to the design, but we are now spending real hardware resources to accommodate SDR options. Adding an application which requires RX and TX such as wifi adds an entire signal chain to the design, as well as a large increase in the required dynamic range of the system. Adding an application which exploits MIMO multiplies the hardware resources needed.

The flexibility of SDR makes it an indispensable research, development, validation and hobbyist tool, but system design is about target selection and trade-offs. To quote one of the most successful engineers of the current era and Eckert-Mauchly award winner Dr. Robert P. Colwell "*Pick your [technical] targets judiciously. ... Pick your vision and then chase it. You can't pick everything as your vision, that's a recipe for mediocrity. If you can't pick your target you're not going to hit any of them.*" For SDR based systems, this would seem to mean that we should focus on applications where the flexibility afforded offsets the inevitable platform cost push, or where it allows 'targets of opportunity' that require a subset of the capabilities of the platform already being used.

At the same time, our earlier definition of SDR as ''a reconfigurable radio system whose characteristics are partially or fully defined via software or firmware'' means that SDR is already everywhere around us on some level. Cellular phones provide an example of devices that connect a large number of hardware radios to a dizzying array of applications that process, consume, modify and sometimes retransmit the received data, while consumer devices such as wireless routers can often add support for protocol changes or tweaks via firmware. While the economics might prevent radio systems from being universal on all dimensions, there are very few radio devices now sold that don't expose at least a few parameters via software.

9 References:

[Liu12] J. Liu, B. Priyantha, T. Hart, H. S. Ramos, A. A. F. Loureiro, and Q. Wang, "Snapshot positioning for unaided GPS software receivers," in *Proc. of the 10th ACM Conference on Embedded Networked Sensor Systems*, Toronto, ON, Nov. 2012, pp. 1–14.

[Liu16] J. Liu, B. Priyantha, T. Hart, Y. Jin, W. Lee, V. Raghunathan, H. S. Ramos, and Q. Wang, "CO-GPS: Energy efficient GPS sensing with cloud offloading," *IEEE Transactions on Mobile Computing*, vol. 15, no. 6, pp. 1348–1361, June 2016, DOI: 10.1109/TMC.2015.2446461.

[Luc16a] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Demonstration of cloud GNSS signal processing," in *Proc. of the 29th Intl. Technical Meeting of the Satellite Division of The Institute of Navigation*, Portland, OR, Sep. 2016, pp. 34–43.

[Luc16b] V. Lucas-Sabola, G. Seco-Granados, J. A. López-Salcedo, J. A. García-Molina, and M. Crisci, "Computational performance of a cloud GNSS receiver using multi-thread

parallelization," in *Proc. of the 8th Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing*, Noordwijk, The Netherlands, Dec. 2016, DOI: 10.1109/NAVITEC.2016.7849357.

[Fer17] C. Fernández-Prades, C. Pomar, J. Arribas, J.M. Fàbrega, J. Vilà-Valls, M. Svaluto Moreolo, R. Casellas, R. Martínez, M. Navarro, F.J. Vílchez, R. Muñoz, R. Vilalta, L. Nadal and A. Mayoral, "A Cloud Optical Access Network for Virtualized GNSS Receivers," in Proceedings of the 30th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2017), Portland, OR, Sept. 2017, pp. 3796-3815.

[Pen11] R. D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, Dec. 2011. DOI: 10.1126/science.1213847.

[BG17] B. K. Beaulieu-Jones and C. S. Greene. Reproducibility of computational workflows is automated using continuous analysis. *Nature Biotechnology*, 35(4):342–346, Apr. 2017. DOI: 10.1038/nbt.3780.

[Swift] Swift Navigation, "Piksi Multi GNSS Module", https://www.swiftnav.com/piksi-multi

[Trimble] Trimble Inc., "Catalyst: Subscription-based, software GNSS" https://catalyst.trimble.com/howitworks.htm , 2017

[VazXilinx] B. Vaz *et al*., "16.1 A 13b 4GS/s digitally assisted dynamic 3-stage asynchronous pipelined-SAR ADC," *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, 2017, pp. 276-277.

[Xilinx] Xilinx, "RFSoC: Radio Frequency System on Chip", https://www.xilinx.com/products/technology/rfsoc.html

[NSL], NSL, "TRUSTED NAVIGATION: AUTHENTICATED GNSS RECEIVERS", http://nsl.eu.com/nsl-jcms/advanced-gnns-hw-sw/authenticated-gnss-receivers, October 2016

Biographies

James T. Curran received Ph.D. in Electrical Engineering in 2010, from the Department of Electrical Engineering, University College Cork, Ireland. He has worked as a research engineer with the PLAN Group in the University of Calgary, and as a grant-holder at the Joint Research Centre of the EC in Italy. He is currently a radio-navigation engineer at the European Space Agency, in the Netherlands. His research interests include signal processing, information theory, cryptography and software defined radio for GNSS

Aiden Morrison received his PhD degree in 2010 from the University of Calgary, where he worked on ionospheric phase scintillation characterization using multi frequency civil GNSS signals. Currently, he works as a research scientist at SINTEF Digital. His main research interests are in the areas of GNSS and multi-user collaborative navigation systems.

Carles Fernández-Prades received the M.Sc. and Ph.D. degrees in electrical engineering from the Universitat Politecnica de Catalunya (UPC), Barcelona, Spain,in 2001 and 2006, respectively. In 2001, he joined the Department of Signal Theory and Communication at UPC as a Research Assistant, getting involved in European and National research projects both with technical and managerial duties. He also was Teaching Assistant in the field of Analog and Digital Communications (UPC 2001-2005). In 2006 he joined CTTC, where he currently holds a position as Senior Researcher and serves as Head of the Communications Systems Division. His primary areas of interest include signal processing, estimation theory and nonlinear Bayesian filtering, with applications in positioning, communications systems, software radio and the design of RF front-ends. He proudly served as Thesis advisor of works acknowledged by the European Association for Signal Processing with the Best Ph.D. Thesis Award in 2014 and 201

Michele Bavaro received his master degree in computer science from the University of Pisa, Italy. Shortly afterwards he started his work on Software Defined Radio technologies applied to navigation, first in Italy, then in The Netherlands at the European Space Agency. In the United Kingdom he worked for NSL on several projects being directly involved with the design, manufacture, integration, and test of radionavigation satellite system (RDSS) equipment and supporting customers in the development of their applications. After working on his own consulting firm, mostly on SDR and low cost precision positioning, he was appointed as technical officer at the Joint Research Center (JRC) of the European Commission. Michele works now at Swift Navigation in California in the Measurement Engine team.