

GNSS Solutions:

Host-based processing and choosing inertial sensors

“GNSS Solutions” is a regular column featuring questions and answers about technical aspects of GNSS. Readers are invited to send their questions to the columnists, Professor Gérard Lachapelle and Dr. Mark Petovello, Department of Geomatics Engineering, University of Calgary, who will find experts to answer them. Their e-mail addresses can be found with their biographies at the conclusion of the column.

What is “host-based processing” of GPS signals and how does it compare to traditional systems on a chip and software GPS approaches?

The last few years have seen the emergence of mobile wireless and other devices using a host-based GPS architecture, in which portions of the software traditionally executed within the GPS chip are now performed in the host software.

This important trend is unfamiliar to many even in the GPS industry. For certain customers and certain kinds of devices, the host-based architecture has lower production costs and is much more flexible. In other cases the host-based approach may not work well, and designers would be better off using a traditional GPS architecture for their devices.

Host-Based versus System on a Chip. The architecture of a host-based system is best explained by contrasting it with the traditional system-on-chip (SOC) approach. In SOC architecture, the entire GPS system is integrated within a single device. The SOC contains three major building blocks: an RF tuner block, a baseband processing block, and a CPU subsystem that runs a complete GPS software application.

The output of the SOC is position, velocity, and time (PVT) data. This data is then sent to the host device, and

is commonly formatted as an NMEA message stream. **Figure 1** illustrates a generic SOC system.

One of the key drawbacks of SOC architecture is the complexity of the associated silicon design, which largely determines the size and cost of the chip. The host-based alternative, shown in **Figure 2**, offers a reduced silicon complexity that translates into a smaller, less expensive chip.

In the host-based approach, the on-chip CPU subsystem is eliminated, leaving only the RF tuner and the GPS baseband processor. However, the host-based approach is not simply an SOC with the CPU removed.

In the host-based approach the GPS baseband processor includes control logic functions that would otherwise reside in an on-chip CPU. These control functions enable signal processing tasks to be performed without real-time interaction with the host software.

In the leanest host-based designs, the output of the GPS baseband processor is not pseudorange measurement data, such as you would get from an SOC, but rather raw correlation energy results that are streamed to the host. The software running on the host converts this data to GPS measurements.

The host software application includes a software module that contains a function for computing GPS navigation data. This module is provided as part of the GPS solution. The input to the navigation processing module is the GPS measurement data; the output is PVT data identical to that produced by an SOC.

Figure 3 compares the silicon content of an SOC chip and a host-based GPS chip, drawn approximately to scale.

The three-die combination seen in the SOC portion of **Figure 3** is typical of today’s GPS chip offerings, while a single die is possible with a host-based chip. The SOC requires a separate die for FLASH memory to hold the programming code for the GPS.

Because FLASH cannot be integrated easily with CMOS logic, this will likely remain a separate die for some time to come. The SOC also uses a separate die for the RF section; again, this is typical of GPS chip solutions available today.

Although the CMOS technology used for the baseband/CPU die could support RF integration, the presence of a CPU and external memory may present challenges in the area of RF interference. The host-based GPS chip has no CPU or off-chip memory, which simplifies the task of integrating the RF and baseband in a single die.

To reduce cost, an SOC could be built using mask ROM, which would allow the program storage feature to be integrated within the CPU die. The disadvantage of this approach is that it leaves no way to modify program code after the chip masks are fabricated. Consequently, SOC with mask ROM becomes a viable alternative only for stable, mature applications where the customer is not likely to need changes during the development cycle. A large unit volume is needed to amortize the costs of the custom masks.

Designers evaluating a host-based versus SOC solution should also consider that standard RISC processor IP cores such as those used in GPS SOCs normally are associated with a per-unit royalty that is not trivial for high volume applications, where price pressure has taken the GPS complete bill of material below \$4.

The benefit of having less silicon is obvious when examining the footprint sizes of various commercial GPS chips. One of the smallest SOC products available measures 7×10 millimeters, compared to less than 4×4 millimeters for some host-based devices.

Integration. In choosing a GPS architecture, cost and size are important, but other factors should also be taken into account. Low- and medium-volume projects, or projects with short development timelines, may benefit from an SOC solution that is more

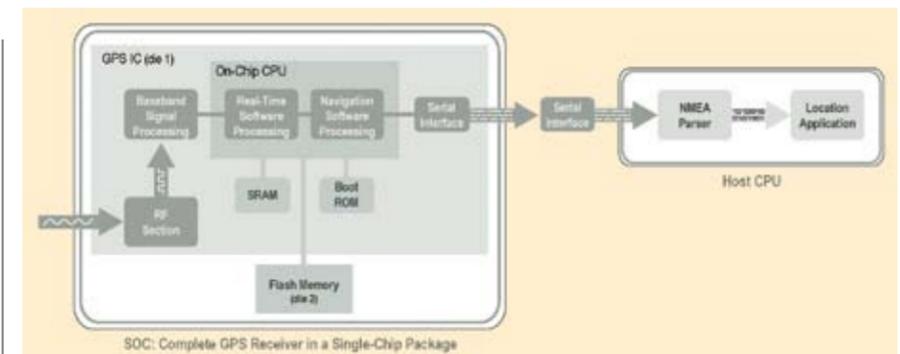


FIGURE 1 System-On-Chip Architecture

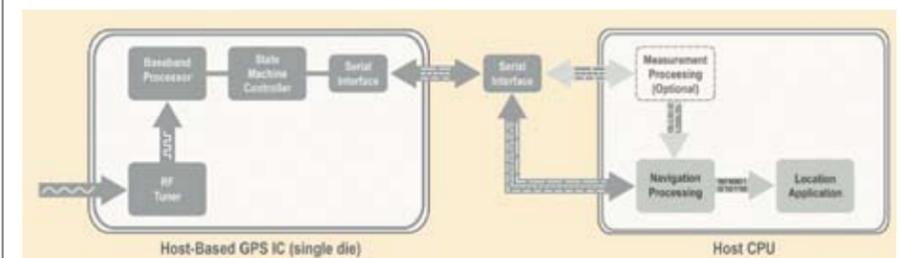


FIGURE 2 Host-Based GPS Architecture

costly, but more straightforward to integrate.

An example of the latter situation would be an autonomous GPS application that will not use network aiding — or assisted GPS. In this case integration of the SOC is elementary: the IC generates PVT information as an output data stream and requires few if any setup commands. After setup, all of the data traffic flows in one direction from the chip to the host CPU.

Figure 4 illustrates the data flow for a host-based GPS system. A number of software layers work together to create the equivalence of an SOC solution driving NMEA data to a COM port.

An assumed requirement of this approach is that the application should be unaware of the host-based architec-

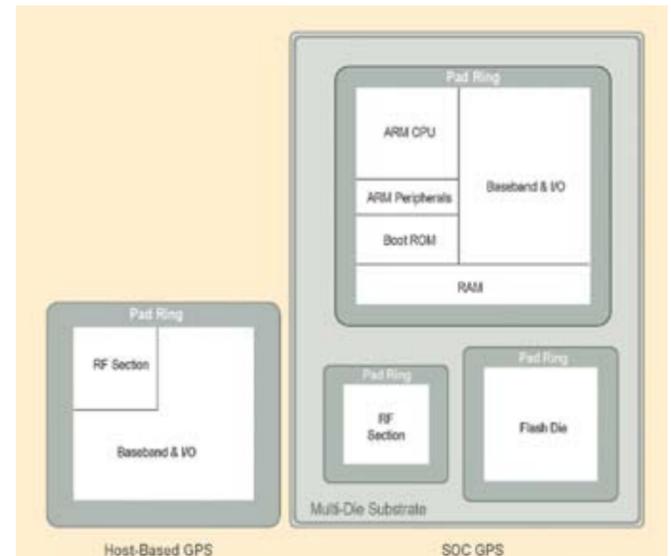


FIGURE 3 Silicon Comparison of Host-Based GPS and SOC GPS

ture. To achieve this, a virtual COM port driver is created. This enables the application to open a COM port as it would for a physical device.

When the GPS port is opened, the virtual driver invokes the GPS application, calling the necessary library functions within the GPS layer. Once the GPS receiver is up and running, the GPS software library delivers NMEA

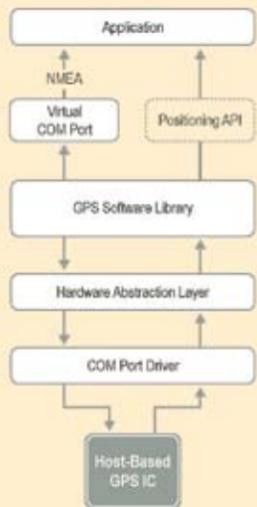


FIGURE 4 Software Architecture for Host-Based GPS

messages to the virtual COM driver, which distributes them to the application. Alternatively, if the application can be modified, it could directly access positioning information from the GPS library through an application programming interface (API). The GPS software must of course communicate with the host-based GPS IC — first to set up and control the IC, and second to receive the correlation results that will be converted to measurements and, ultimately, positions. This requires a duplex communication channel to the IC.

Because the nature of the physical communication will vary from platform to platform, the GPS software is normally created in a generic fashion. A hardware abstraction layer is used to interconnect the GPS functions to the COM driver, which physically relays the data. This is an open layer of software that allows the GPS chip vendor to supply its proprietary software in a compiled library suitable for all platforms.

The integration of the GPS software supplied by the vendor into the customer application includes the following:

- Create the hardware abstraction layer. This is a thin layer of

functions that serve as the “glue” between the software library supplied by the chip vendor and the platform driver that controls the universal asynchronous receiver transmitter (UART, a component that handles asynchronous serial communication).

- Provide the vendor with porting information so that the vendor can create a GPS software library in compiled form. Included in the information would be the make and model of compiler and the preferred build options to obtain a library that successfully links into the existing host software build. Achieving a successful port requires close cooperation between the vendor and the customer.
- If existing applications are to be used, then a virtual COM driver is supplied. Some windows-based operating systems include this capability inherently; for example, the Microsoft Intermediate Driver Environment for Windows Mobile 5. In other cases, the chip vendor may supply a virtual COM driver that can be customized for the platform.
- If a virtual COM driver is not required, an application can interface with the GPS chip through its positioning APIs.

With these steps completed, the system performs in a manner similar to an SOC implementation, and any differences are invisible to the applications running on the host.

Assisted GPS. The integration landscape changes considerably when assisted GPS (AGPS) enters the picture. AGPS applications use a messaging protocol

to communicate to a network server. The communication channel typically is either a wireless IP channel (user plane) or a control channel in a wireless network (control plane).

Many customers prefer that the GPS chip provider supply the necessary protocol software. In the case of a host-based GPS system, this protocol software is part of the functionality provided in the GPS software library. For the user plane option, the software picture illustrated in **Figure 5** resembles that of the autonomous-only case.

The added functionality is supported through the components that control messaging between the GPS software library and the AGPS server. In the user plane these messages flow over TCP/IP. As mentioned earlier, to allow the vendor’s GPS software library to work in any platform, the networking communication functions are generic (that is, applicable to any platform).

A *network abstraction layer* is created by the customer to tie the specific TCP/IP interfaces into the GPS library. A one-to-one correspondence normally exists between the GPS functions and the TCP library; so, the layer is again a “thin” implementation of functionality to open and close TCP sockets and stream data to a network address.

In the case of an SOC implementation, the jump from an autonomous-

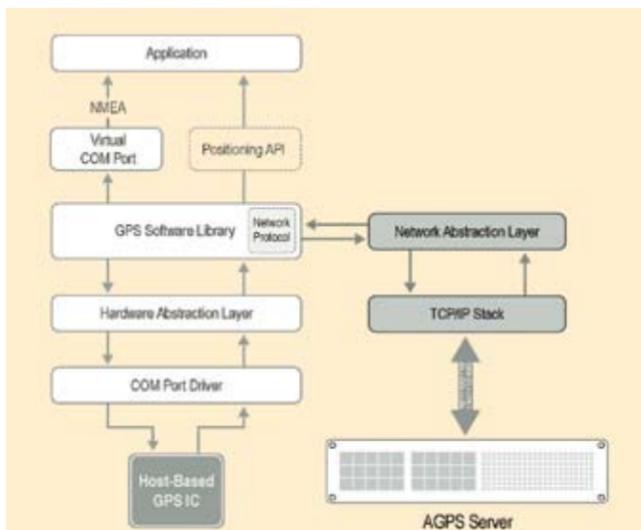


FIGURE 5 Software Architecture for Assisted GPS in a Host-Based System

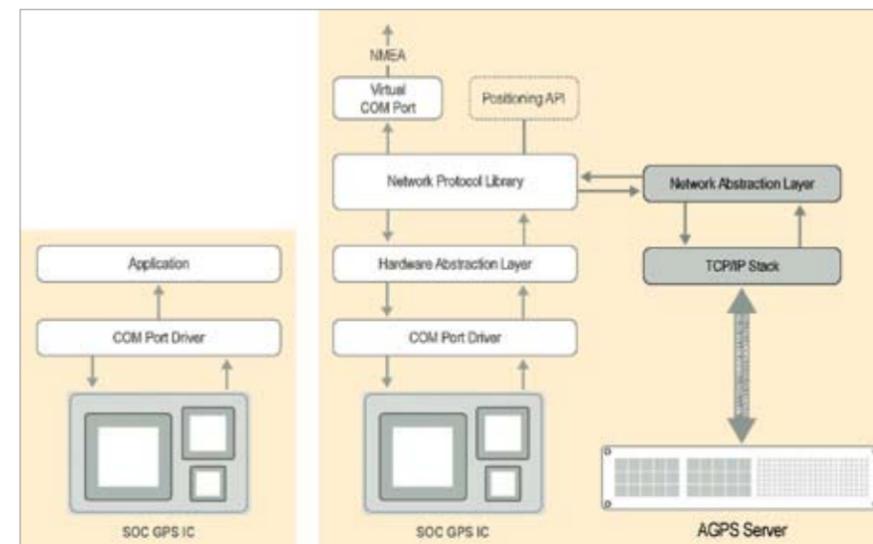


FIGURE 6 Software Architecture for Autonomous and Assisted GPS in an SOC System

only application to an AGPS application is larger because the designer must for the first time introduce vendor-supplied software. **Figure 6** illustrates

this transition: on the left is the SOC architecture for autonomous-only application and on the right the SOC architecture for an AGPS application.

As shown in the illustration, the architecture for the SOC case closely resembles the architecture for the case of host-based GPS. The main difference lies in the vendor’s internal partitioning of functionality between the SOC and the software it supplies to the host. In either case the application and platform interfaces are the same.

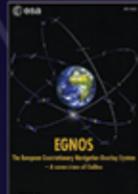
When considering software integration, a designer should determine at the outset whether the application will need to work with a network server, either initially or in subsequent revisions of the product. If the answer is “yes,” the initial integration effort of a host-based approach may offer the advantage of paving the way for easy addition of the assistance data channel in the future. If an application is purely autonomous, the relative simplicity of SOC integration may offer a better approach.

Hot off the Press!

New Books from NavtechGPS



GNSS Aided Navigation and Tracking: Inertially Augmented or Autonomous.
Brand new from James L. Farrell
A NavtechGPS exclusive.
In stock now



EGNOS The European Geostationary Overlay System - A Cornerstone of Galileo
Brand new from ESA. A wide ranging overview of EGNOS and the systems it will augment. In stock now.



GPS System: Signals, Measurements & Performance 2nd ed.
An excellent, cohesive university level textbook by two master teachers. The book provides an overview of key GPS elements. A NavtechGPS exclusive.

navtechgps.com
NavtechGPS (800) 628-0885 or (703) 256-8900

Serial Data Recorder

Logs RS-232 data to CompactFlash™ solid-state storage without a PC

- saves power
- saves hassle
- saves money



DataBridge™ SDR-CF

Great for binary and NMEA GPS !

Acumen Instruments Corporation
2625 N. Loop Drive Suite 2200
Ames, IA 50010
www.acumeninstruments.com
(515) 296-5366

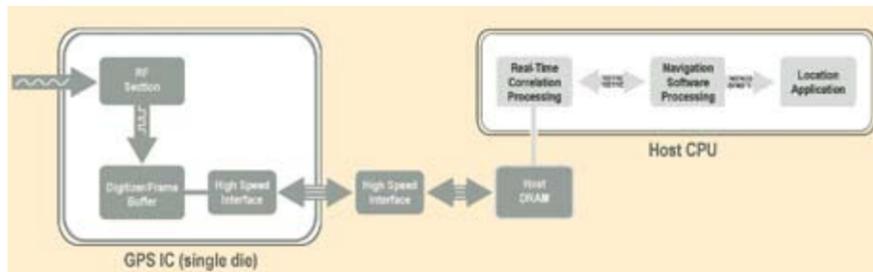


FIGURE 7. Software GPS Architecture

Another thing to keep in mind for AGPS applications is that handset makers normally do not develop protocol stacks themselves, but source them from cellular chip suppliers as part of a complete mobile platform. GPS protocol layers are included in this bundle; so, a growing trend has emerged in which chip suppliers pre-integrate all the GPS software inside their platforms.

This software integration helps to reduce costs and time to market because it frees OEMs from having to conduct qualification tests and interoperability trials for which they would otherwise be responsible. Many chip makers also include a host-based GPS vendor as part of their solutions because the effort of integrating a third-party library is negligible compared to the scope of the complete GPS software inside the handset.

Upgradability. Another important element for integration is the ease or difficulty of upgrading software. Such upgrades may be required during any phase of the development process. In the integration phases, improvements developed by the chip vendor to meet unique customer needs or perceptions drive the upgrades.

In the late stages of development, designers may implement upgrades in order to correct bugs or deficiencies discovered in interoperability or field tests. These issues can arise very late in the process, sometimes on the eve of a product launch. Finally, once a product is deployed, upgrades may be required to fix latent bugs that were unknown at the time the product was launched.

In an SOC solution, FLASH mem-

ory provides an upgrade path. Nevertheless, because the FLASH cannot be removed, the host software must incorporate a FLASH loader feature to support upgrading software in the actual product. This is an important software component, best received from the SOC vendor, then ported to the customer application. Moreover, the loader represents an item of vendor-provided software that will be present even in the purely autonomous case.

Upgrading software in the end user product can be even more complex, as the process of receiving, installing, and upgrading FLASH must be seamless and easily carried out by users. This can be done technically, but it requires exhaustive planning and software development. If a manufacturer must support field upgrades of FLASH, the SOC solution loses much of its simplicity.

In the host-based solution, the software is part of the customer appli-

A growing trend has emerged in which chip suppliers pre-integrate all the GPS software inside their platforms.

cation and is upgraded through the same mechanisms used by customers to upgrade other software packages in the device. GPS improvements and bug fixes can be pulled through during normal product maintenance releases without the complexity introduced by a FLASH loader.

Some SOC vendors offer mask ROM versions of their product in order to eliminate the extra cost of the separate FLASH memory die. In this case the upgrade path is long and expensive.

To effect a software change, the company that fabricates the ICs must create new mask layer(s) for several metal layers to be able to change code. Next, new wafers must be processed and parts packaged. The “compile” time for a code change in this process is several months.

Mask ROM is appropriate for highly stable applications that are governed by mature specifications that seldom change (for example, Bluetooth devices that comply with a mature industry specification). GPS has few universal performance benchmarks, and industry standards are in the early phases, changing frequently.

User needs and expectations for GPS continue to evolve, especially in the case of cellular handsets, where widespread use of location applications is only just beginning. As a result, mask ROM may not be a good choice for these emerging markets. In more mature industries, such as automotive where the GPS software has a track record and has been stable for an extended period, SOC vendors may carry specific mask ROM versions of their chips, targeted for specific customers. This of course leads to added costs, because different customers cannot share silicon wafers.

Software GPS. Host-based GPS is sometimes confused with software GPS. In fact, the two architectures are quite different. As shown in Figure 7, in a software GPS architecture, all of the GPS signal processing is performed in the host in addition to the navigation functions that are performed in the host in a host-based architecture.

This partitioning of the GPS processing results in a radically different design for the GPS IC. The hardware components of the “software only” solution include an RF section, identical to that required in the host-based GPS approach. The output of the tuner, rather than being further processed by signal processing hardware, is digitized and placed into a frame buffer.

Data samples are then transferred to the host over a high-speed interface. Next, these samples are placed into high-speed RAM within the host, where they are made available to the signal processing algorithms. All GPS signal processing tasks, including correlation and matched filter functions, are run in the host.

Several vendors have announced commercially available software GPS solutions, but this architecture has not been broadly adapted. Novel challenges face designers in applying the current software GPS solutions, including the immensity of GPS signal-processing requirements, portability, and data flows.

Hundreds of MIPS are needed just to perform the basic GPS operations that bring the performance of software GPS into parity with traditional GPS performance metrics. Meanwhile, the latest GPS hardware solutions (both

SOC and host-based) are capable of performing large matched filtering operations in real time and without the substantial CPU power required by software GPS.

A second issue is that current software GPS solutions tend to be non-portable from platform to platform, both from hardware and software perspectives. The requirement for a high-speed serial interface, for example, immediately narrows the available interfacing options. A typical interface is SDIO, commonly used in the PC world but rarely available in mobile wireless devices. The ubiquitous and highly requested UART interface is not an option, due to multi-megabit bandwidth requirements.

On the host side, to deal with the incoming data, hardware support is needed for data transfer into the host-side frame buffer. A direct memory access (DMA) controller, if available,

fulfills this need, but adds to the cost of implementation.

From a software perspective, the real time demands of the signal processing tasks require that the solution run in a multi-thread environment in order to guarantee the necessary cycles to the software process to keep up with incoming data. Intuitively one would assume that a software GPS solution is less expensive. However, current semiconductor technology allows a large number of digital processing gates to be packed into a small area. Moreover, cost and size are often as much driven by interfacing requirements (for example, pin count) as by the core processing area.

One advantage of a software GPS solution is the upgradeability of the signal processing components that are “baked” into a host-based or SOC solution. However, a further word of caution is needed here: the migration to

GRANADA
Galileo Receiver ANALYSIS And Design Application

The Reference Galileo Simulation Toolkit
for GNSS Receiver Research and Development

ENVIRONMENT AND NAVIGATION SIMULATOR

- Multisystem navigation analysis (Galileo/GPS)
- Graphical user interface for constellation, environment and receiver configuration
- Raw data and range errors generation

FACTORED CORRELATOR MODEL

- MATLAB/Simulink blockset for simulation of multi-channel GNSS receivers including hybrid solutions
- Swift and flexible receiver model based on analytical formulation of correlator outputs
- Generation of realistic pseudorange measurements for user-defined dynamics

BIT-TRUE GNSS RECEIVER SIMULATOR

- Open design environment for Educational and R&D purposes (Matlab/Simulink)
- Fully configurable GNSS system and receiver architecture
- IF sample-based signal generation and processing

Information and orders: granada@deimos-space.com
Ronda de Poniente 19, 28760 Tres Cantos, Madrid, Spain
tel (+34) 91 806 34 50 - fax (+34) 91 806 34 51 - www.deimos-space.com/granada

deimos
SPACE

Galileo and other GNSS systems is not straightforward. The signal processing requirements for Galileo, for example, are up to 16 times that of GPS due to the longer C/A code, the BOC modulation, and the presence of data and pilot channels. So a software GPS solution is not necessarily upgradable to Galileo, it depends on the capabilities of the hardware on which the software solution is implemented.

Conclusion. System on a Chip (SOC) architecture remains the easiest to implement, but with the most hardware required. Software GPS is useful for R&D but is not currently a viable commercial architecture: it requires multi-megabits/s of bandwidth, hundreds of MIPS, a high degree of platform dependence, and, ironically, often

requires more hardware than host-based GPS.

The choice between host-based GPS and SOC comes down to a company's target implementation and volumes. For autonomous-only implementations SOC architecture is significantly easier to implement, but the difference in integration complexity shrinks for assisted-GPS.

In any event, if a manufacturer is building roughly 100,000 GPS devices per year, SOC GPS is the best choice, because of the ease of integration. The cost savings from host-based GPS will probably not justify the overhead involved in the tight host-based integration.

For production targets around one million devices per year, however,

host-based GPS is easily the best choice — the cost savings in one year will far exceed the overhead involved in the tight integration, with the added benefit of fewer parts, smaller GPS footprint, built-in support for A-GPS, and a more manageable software upgrade path.

CHARLIE ABRAHAM



Charlie Abraham is vice president for engineering at Global Locate, Inc. where he is responsible for the technical architecture of GPS chipsets and software. Previously

Abraham worked at Trimble, Ashtech, Magellan, and Hughes Aircraft. He holds a master of engineering degree from the University of Southern California. Abraham has more than 100 issued or pending patents in the field of GPS.

What are the important considerations when selecting the type and quality of IMU for integration with GNSS?

The key objective in choosing an inertial measurement unit (IMU) for a GNSS-aided inertial navigation system (INS) is to obtain the right trade-off between performance and cost of the integrated system in its intended application.

The type of data and the desired accuracies of those data comprise the performance attributes of a GNSS-aided INS of interest for a particular application. For example, a land-vehicle navigation application may require continuous positioning during periods of poor GNSS coverage, such as in an urban canyon environment. So, here the key performance attribute is position accuracy during multiple GNSS outages.

Another example: an aerial photogrammetry application requires highly accurate position and orientation angles in order to generate the exterior orientation parameters of each image. In this case, GNSS coverage and, hence, position accuracy is usually not a problem. Instead, the key performance attribute is orientation accuracy during normal aircraft dynamics on a survey mission.

Inertial Cost Fundamentals. An IMU contains three gyros and three accelerometers for the purpose of measuring vectors of angular rate and specific force (inertial acceleration and gravity) that an IMU experiences. An INS uses these inertial data plus an initial alignment state to solve Newton's equations of rotational and translational motion on the earth and thereby compute a position and orientation solution.

The cost of an IMU depends on the class of inertial sensors that it contains. IMUs are typically categorized according to their intended applications and gyro quality expressed in terms of gyro bias in units of degrees per hour. A secondary performance measure is the gyro random walk expressed in

degrees per root-hour, which is the integrated gyro random noise. (Accelerometer quality is assumed to be commensurate with gyro quality for its intended application, and is therefore not explicitly mentioned in an IMU categorization.)

A *navigation-grade IMU* is capable of operating in an INS with a free-inertial position drift (that is, the position drift in the absence of any external corrections) on the order of one nautical mile per hour or 0.5 meters per second after a good initial ground alignment. To meet these performance specifications, such an IMU necessarily contains gyros having better than 0.01 degree/hour biases and precise accelerometers having better than 100 micro-g (1 micro-g equals one millionth of gravitational force) biases. Gyro random walk is typically 0.002 degrees/root-hour or better.

The prevailing gyro technology that can deliver this accuracy in a cost-effective manner is the ring-laser gyro (RLG), although some fiber-optic gyros (FOGs) are also capable of competing with RLG performance and cost. The typical cost of a navigation-grade IMU is in the range \$50,000 to \$100,000.

An *avionics-grade IMU* has its main application in attitude and heading reference systems (AHRS) that provide accurate roll, pitch, and heading for commercial and military aircraft. It contains gyros in the 0.1 degree/hour category whose technology can be lower-cost RLG, FOG or spinning mass dry-tuned gyros (DTG). Avionics-grade IMU prices range from \$20,000 to \$50,000, depending on the quality and cost of the inertial sensors.

A *tactical-grade IMU* is typically designed for use in a weapon such as a missile or guided bomb. It needs to navigate the weapon for only a few minutes at most and, consequently, can use less expensive inertial sensors.

This category of IMU is designed to be small, light, and inexpensive, with a typical price in the range \$5,000 to \$20,000. Tactical-grade IMUs use gyros having 1 to 10 degrees/hour bias and accelerometers having around 1 to 5 milli-g biases. Gyro random walk is usually in the range 0.05 to 0.2 degrees/root-hour, depending on the gyro technology.

With the advent of micro-electro-mechanical system (MEMS) inertial sensors — miniature sensors mass-produced out of silicon or quartz using integrated circuit production methods, a new category of IMU called the *low-cost IMU* or *commercial grade IMU* has begun to appear.

The MEMS gyros in these IMUs are designed for large-scale commercial markets such as the automobile industry (for yaw stabilization and skid control) and have biases on the order of 0.1 degrees/second. They tend to be quite noisy, with random walk on the order of several degrees per root-hour.

MEMS accelerometers are likewise mass-produced for large markets such as air bag deployment sensors in automobiles; consequently they are low cost and relatively inaccurate. The typical commercial-grade MEMS IMU price is in the range \$500 to \$2,000.

GNSS/INS Integration. As shown in **Figure 1**, a GNSS-aided INS incorporates the INS (IMU plus inertial

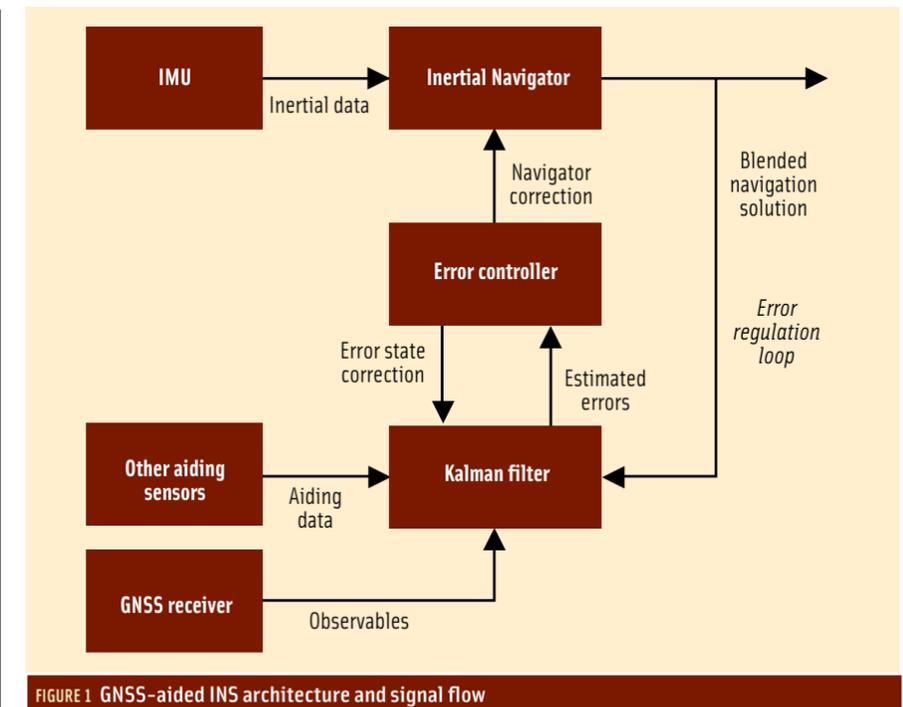


FIGURE 1 GNSS-aided INS architecture and signal flow

navigator mechanization) embedded in a closed error regulation loop that includes a Kalman filter and an error controller. The Kalman filter is designed to estimate the errors in the INS solution and the contributing sources of this error, which include the dominant inertial sensor errors and aiding sensor errors.

The Kalman filter is able to observe these errors in measurements constructed as the differences between elements of the inertial navigation solution and corresponding aiding sensor data, including GNSS. The error controller then converts the estimated INS errors into corrections to the integration processes in the inertial navigator mechanization and applies these corrections periodically. It also adjusts the Kalman filter's estimated errors to reflect the INS correction.

The Kalman filter typically contains models for the gyro and accelerometer biases, and possibly other inertial sensor errors, such as scale factors and misalignments, if these are anticipated to prove significant. Consequently, the GNSS-aided INS provides dynamic calibration of the IMU errors, which allows the GNSS-aided INS to achieve

a better level of performance than an unaided INS would be capable of.

As the result of this integrated design, the IMU in a GNSS-aided INS can be of lower quality and, hence, cost less than the IMU in an unaided INS to achieve a particular level of performance. For example, a navigation-grade IMU having 0.01 degree/hour ring laser gyros in a free-inertial INS following a good ground alignment can provide satisfactory roll-pitch-heading at the 0.05 degree RMS level. The same roll-pitch-heading accuracy can be derived from a GNSS-aided INS containing a tactical-grade IMU with 1 degree/hour gyros and costing one tenth of the navigation grade IMU. The difference is that a free-inertial INS is completely autonomous following a ground alignment, whereas a GNSS-aided INS requires ongoing GNSS aiding with few interruptions to achieve this performance.

In typical survey missions, dependency on GNSS aiding poses no significant hardship, and as a result a GNSS-aided INS may be a better choice for reasons of cost, size, weight, and power consumption. The caveat here, however, concerns the inertial sensor noise,

in particular the gyro random walk, because a Kalman filter cannot calibrate or suppress broadband noise.

Consequently an IMU with fairly high gyro random walk can fail in a GNSS-aided INS application even if its inertial sensor biases are reasonably small. The high broadband inertial sensor noise from commercial-grade MEMS IMUs has to date limited their application to lower accuracy attitude measurement applications.

Choosing an IMU. Selection of an appropriate IMU as part of a design process is in part a value engineering exercise. It requires the skills of a navigation analyst who can understand the relationship between various inertial sensor errors and the resultant inertial navigation errors, as well as the degree to which a Kalman filter aided with GNSS data of a certain quality and frequency can estimate these errors.

In order to establish the maximum IMU errors that an application can tolerate, the IMU selection process

typically involves some simulations of candidate missions that will incorporate a GNSS-aided INS. Modern simulation tools allow the navigation analyst to try out different

Selection of an appropriate IMU as part of a design process is in part a value engineering exercise.

combinations of inertial sensors and Kalman filter designs. Often this includes an exploration of the effect of vehicle dynamics on the resulting performance of a candidate design.

Vehicle accelerations enhance the observability of some INS errors in the Kalman filter of a GNSS-aided INS. In particular, heading error is normally weakly observable in a stationary or benignly dynamic GNSS-aided INS, with the heading error being approximately proportional to the gyro bias. Consequently a fairly expensive IMU with small gyro biases may be needed to achieve an accurate heading.

If the vehicle accelerates periodically in its intended mission, then the Kalman filter's heading error observability improves significantly, and the GNSS-aided INS is able to control the heading error with significantly less dependence on the gyro biases. This allows the use of a lower cost IMU to achieve the same desired heading accuracy.

Let's take aerial photogrammetry as an example. The vehicle is an aircraft carrying a camera or line scanner. A series of parallel flight lines during which the images are captured, connected by 180-degree turns, comprises a typical photo-mission trajectory. The centripetal accelerations during these turns increases the observability of errors, which allows the GNSS-aided INS to achieve excellent roll, pitch, and heading accuracy with a less expensive tactical-grade IMU.

Once a candidate IMU is selected, the next step is to test the IMU in a prototype GNSS-aided INS. This

is usually the last and most costly step in the design process, because it involves real hardware being tested on a real vehicle. This is especially so, for instance, if the vehicle is an airplane and the application is aerial surveying that also involves an expensive sensor such as a large-format camera or LIDAR.

As part of due diligence in the system design process, the candidate IMU should be subjected to statistical analysis to verify its published specifications. The typical tools for this analysis are a thermal chamber, a rate table, and statistical analysis software. The thermal chamber provides accurate control of temperature and temperature rate for these tests. The rate table ensures accurate rotational rates and angular changes.

The thermal chamber and rate table are usually integrated into a single test device. The IMU is subjected to stationary drift tests and rotational tests at various temperatures and possibly different temperature gradients. The resulting data collected from these tests are then reduced to representations of statistical performance such as bias standard deviation, in-run bias variation, and random noise.

If the previous design and simulation analysis was done correctly, and the candidate IMU meets its published specifications, then the vehicle test stage should be a verification of the expected performance.

BRUNO SCHERZINGER



Dr. Bruno Scherzinger

is the chief technology officer at Applanix Corporation in Toronto, Canada, which he co-founded in 1991. He is responsible for advanced navigation technology development and the core navigation technology in the Applanix product line. 

**Mark Petovello** is a Senior Research Engineer in the Department of Geomatics Engineering at the University of Calgary. He has been actively involved in many aspects of positioning and navigation since 1997 including GNSS algorithm development, inertial navigation, sensor integration, and software development.

Email: mpetovello@geomatics.ucalgary.ca

**Professor Gérard Lachapelle** holds a CRC/iCORE Chair in Wireless Location in the Department of Geomatics Engineering at the University of Calgary. He has been involved with GNSS since 1980 and has received numerous awards for his contributions in the area of differential kinematic GPS and indoor location.

Email: lachapel@geomatics.ucalgary.ca