Seamless Navigation through GPS Outages

A Low-Cost GPS/INS Solution

YONG LI, PETER MUMFORD, AND CHRIS RIZOS
UNIVERSITY OF NEW SOUTH WALES

View from the car looking up George Street

The search for seamless, continuous navigation capability frequently looks to integration of inertial technology to help bridge outages of GPS positioning when satellite signals are blocked. A team of Australian researchers describe the progress of their efforts to create an FPGA-based sensor-fusion unit built on an embedded processor.

GPS signal blockage can be experienced in many environments in which a vehicle may operate: a forested road, the urban jungle, a tunnel, road "spaghetti" — you get the picture.

Inertial sensors complement GPS well, potentially bridging the gaps in GPS coverage and providing vehicle attitude information as a bonus. But there is a catch: the data streams from a GPS receiver and inertial sensors are independent and must be fused together to generate a useful stream of navigation (position, velocity, and attitude) data.

In this article the hardware and software algorithms that have been developed at the University of New South Wales (UNSW) to do this fusing operation are presented along with an example of the capabilities of the system. The article concludes with a look at some applications presently being explored and ideas for future work.

The complimentary nature of GPS and an inertial navigation system (INS) is well known. GPS provides absolute XYZ coordinates at a low data rate; INS essentially provides changes in XYZ (and attitude) at a high data rate. While GPS can suffer loss of satellite signals that can lead to no XYZ at all, INS just keeps on going but will drift as errors compound.

So, it's a happy marriage. GPS can bound the INS errors, and INS can fill the gaps between GPS fixes.

Broadly speaking our GPS/INS system has four parts: the sensors, the synchronization component, the integration/fusion component, and data output. The system can operate in real-time or post-processing mode — the data fusion algorithms for both modes are essentially the same. In the real-time mode the algorithms run on an embedded processor on the prototype hardware device. In the post-process mode, the algorithms run on a PC.

The hardware and the system components will be detailed shortly, but first let's see what the system is capable of.

In the City

We took our car on a drive across Sydney, Australia, with a GPS receiver that incorporates differential corrections

InsideGNSS

from a commercial satellite-based augmentation system (SBAS) and a tactical-grade inertial sensor attached. (Editor's note: Details on all commercial products used in the UNSW project can be found in the Manufacturers section at the end of this article.) Our custom hardware synchronized and logged data from these sensors as we drove through a variety of tough environments for GPS. The data was processed on a PC and the results analysed.

In the course of this journey GPS signal loss occurred on many occasions. Most surprising was a long outage that took place as we crossed the Sydney Harbour Bridge.

The drive took us across the Harbour Bridge and through the center of the city. The accompanying photo provides a view from the car of the environment in George Street, the main thoroughfare of Sydney – tall buildings on both sides, moderate traffic, and lots of multipath. **Figure 1** shows the trajectory where GPS was available. **Figure 2** shows the trajectory with the gaps in GPS filled by INS data.

The area around George Street represents a fairly typical "downtown" scenario; so, let's take a closer look at the performance of the system here. **Figure 3** shows the integrated GPS/INS solution overlaid in dark green on a raster map, left, and light green on a georeferenced aerial image-based map. GPS outages of up to 107 seconds occurred, with short (and possibly fairly inaccurate) GPS fixes in between.

Traveling up George Street was slow due to traffic and traffic lights, but the integrated solution follows the streets and corners reasonably well, eventually emerging into an area of good GPS coverage on the other side of the city on Oxford Street.

The constant offset to the south that can be seen on the raster map does not appear on the other map, a phenomenon that may be caused by using different coordinate systems. Although a slight drift away from the road in the final segment of the trajectory can be seen on both maps, the solution is still encouraging considering the seriousness of the GPS outages.

GPS/INS System Design

The components of the test system consist of the INS, the SBAS-capable GPS receiver, and the sensor-fusing hardware and software. Several GPS receivers were used in the initial phase of the development, all of which can work with the system. The SBAS-capable unit was chosen for the final system because it is used by our project partner, the NSW Department of Lands.

Figure 4 provides a schematic overview of the system components and interconnections. The sensor fusion device (henceforth "the device") receives data streams from the INS and GPS receiver as well as the one pulse-per-second (PPS) signal from the GPS receiver.

The device can be configured to output a solution consisting of position as well as attitude that can be logged to a compact flash (CF) card or be fed directly into a laptop running, for example, geographic information system (GIS) software. Alternatively, the device can simply log time-synchronized data for subsequent post-processing on a PC. This second option is convenient for algorithm development.

The INS is manufactured using micromachining technology. It can provide inertial measurement outputs including delta velocity and delta theta (differences in speed and angle/orientation). The inertial sensor assembly consists of six single-axis sensors, three quartz rate sensors, three vibrating quartz accelerometers, the drive electronics, preamplifier circuitry for the sensor outputs and the digital conversion electronics.

The INS has internal time-synchronization functionality, which provides a reference for evaluating the timing result from the field programmable gate array (FPGA) system. The timing performance of the system has previously been reported. (See the paper by P. Mumford et alia 2006, in the Additional Resources section at the end of this article).

A commercial SBAS differential service is used to provide a GPS solution accurate to one meter or better. However, it's important to note that, like GPS signals, an SBAS correction signal can be blocked by obstructions.

Hardware details

The prototype device hardware consists of a general-purpose embedded RISC

InsideGNSS

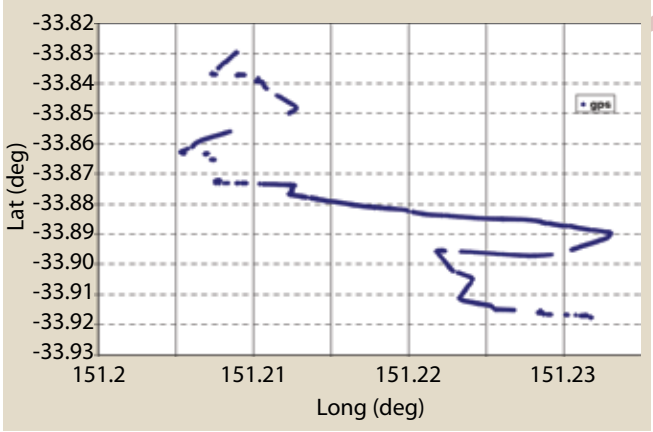

FIGURE 1 Trajectory of successful GPS fixes

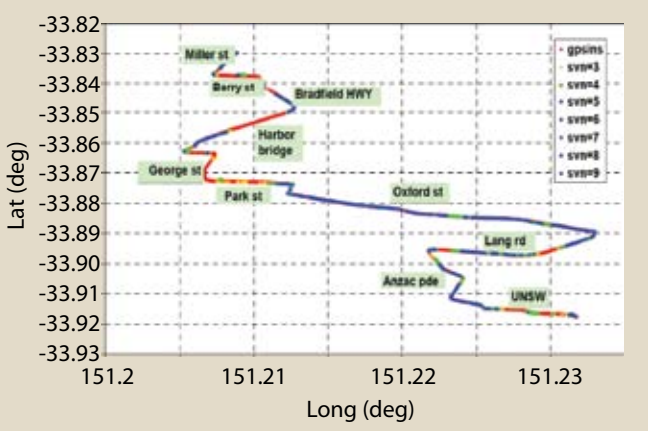

FIGURE 2 Trajectory of integrated GPS/INS solution

soft-core processor development board and additional circuits for serial ports and input pulse conditioning. The primary feature of the development board is the FPGA chip that hosts the custom logic and soft-core processor that does most of the work.

We exploit the flexibility of FPGA technology to achieve the required functionality on an off-the-shelf piece of hardware. **Figure 5** provides a schematic overview of the logic design in the device. The sensors feed the time-synchronization logic block that in turn feeds the time-stamped data to software running on the processor core.

The data streams from the sensors must be time-tagged for the subsequent time-synchronization algorithm. This process uses a custom serial port design and the PPS signal from the GPS receiver. Accurate time-synchronization of the data streams from the sensors is fundamental in achieving good data fusing results.

Typically, low-cost INS sensors do not provide a timing pulse; so, the data stream must be time-tagged when it arrives at the device. To minimize the effect of latencies in serial port hardware, custom logic in the FPGA has been developed to pick up the start bit of a data byte and attach a time tag to it.

To allow the time tags to be tied to an absolute reference, the PPS is also time-tagged. The PPS is aligned to the GPS second, and this is typically accurate to within 100 nanoseconds or better.

A GPS message following the PPS contains the absolute coordinated universal time (UTC) at the instant of the pulse. This allows the time of the pulse to be known and — via interpolation — the time of all time tags and, hence, the time of a data byte.

The alert reader may be thinking at this stage, if the time-synchronization function relies on the PPS, and the PPS comes from the GPS receiver, what happens when the GPS receiver has an outage? The time-synchronization function does rely on the PPS, and the PPS is guided by the GPS receiver, solving for its clock frequency (TCXO) error.

GPS receivers can handle the PPS in various ways. The receiver can turn off the PPS when there is no position/velocity/time (PVT) solution — but this is what we don't want to happen. Another way is to model the TCXO drift terms while the receiver has a good PVT solution. The receiver can then use this model to coast through an outage and keep the PPS coming — this is what we do want.

The actual logic design is quite complex, as it includes first-in-first-out (FIFO) buffers for each serial data

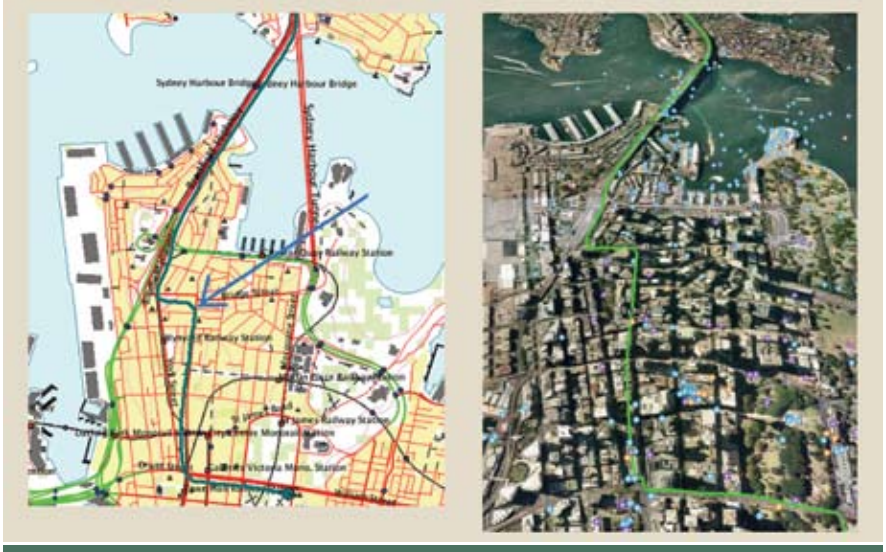

FIGURE 3 Trajectory of car through the Sydney city center

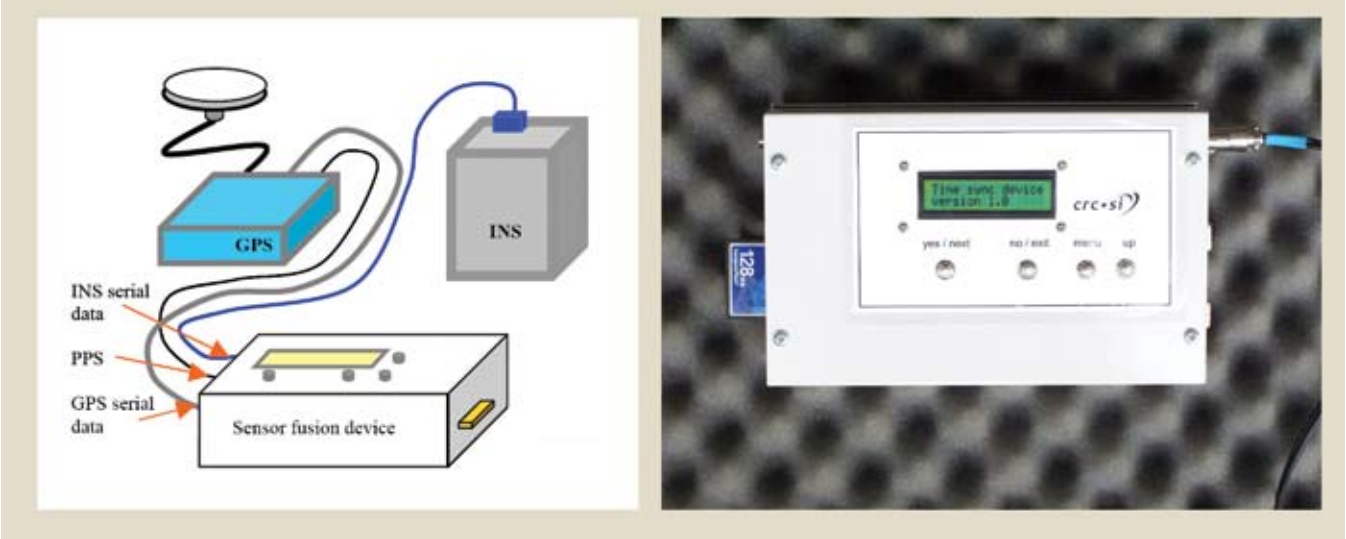

FIGURE 4 Test system

stream to cope with high data rates without overloading the processor with too many interrupts. **Figure 6** reveals the inner workings of the logic.

A free-running counter is latched on the arrival of a start bit and attached to the following data byte going into a FIFO. When the FIFO gets full, it generates an interrupt to the soft-core processor, signaling the software to fetch and process the data further. For serial data, each byte of received data has a time tag appended to it to form a 32-bit value.

With this setup we can obtain the time-of-arrival for each byte; however, generally only the header of the serial data message requires time to be associated with it. Software running on the processor core can parse the data stream to find message headers and attach timing information for further processing.

The embedded processor on the FPGA hosts the real-time software. This software uses the Embedded Configurable Operating System (eCos) for library and multi-thread support (more about eCos in the book by E. J. Massa listed in Additional Resources). In addition to performing the integration computation, the processor must respond to interrupts from the FIFOs to receive the incoming (time-tagged) data streams.

All these processes put significant demands on the processor, and in practice this has limited the update rate. The soft-core processor is custom-designed using the "SOPC Builder," a tool that is part of the FPGA design software package.

The processor core can be big and fast or small and . . . not so fast, depending on system needs. Here we need the biggest, fattest processor we can build, with large data and address caches and every hardware accelerator we can get. Unfortunately, although a single precision floating-point unit is easily available via the SOPC Builder, we need a double point unit.

This is where the flexibility of the FPGA platform can really help out, and a range of options can be pursued to solve performance issues. Increasing the processor clock frequency is one option; implementing the double precision unit is another.

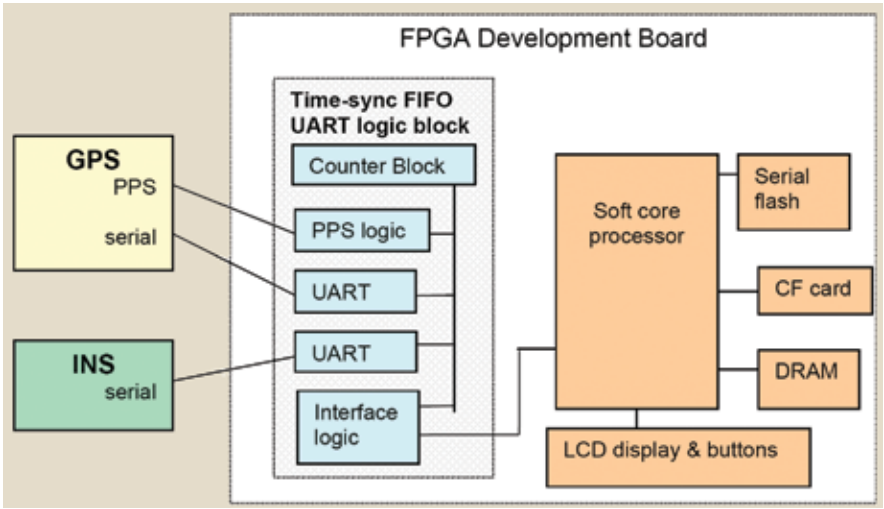

FIGURE 5 Logic design overview of the sensor fusion device

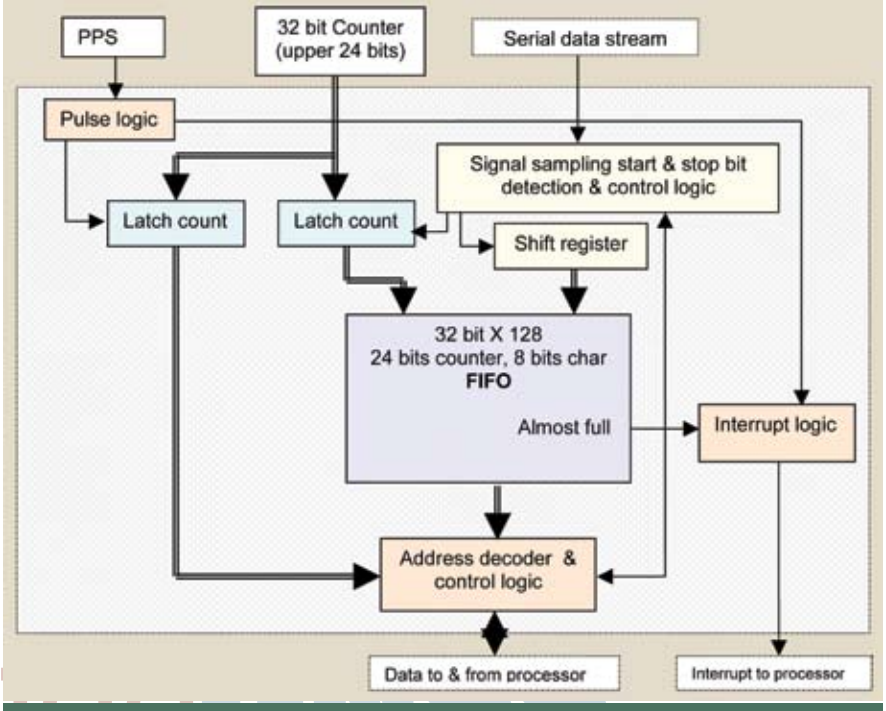

FIGURE 6 Time-tagging logic details

A useful feature of the processor development tool set provides a further option: off-loading software code loops to FPGA logic as custom instructions. So, when the processor starts running out of steam, the software can be analyzed to see if any intensively used loops can be converted to logic circuits and called as a one-cycle custom instruction instead. This can result in a huge acceleration of code and reduce code bottlenecks.

A further option is to split the software tasks over several processor cores. A small core could perform the preprocessing task of receiving serial stream data and time-synchronization, while a large fast core with hardware accelerators could perform the strap-down INS, Kalman filter, and output tasks. The large core could then be free from high rate interrupts.

We hope that, by applying some of these options, we can squeeze enough performance out of the silicon to really get the device flying.

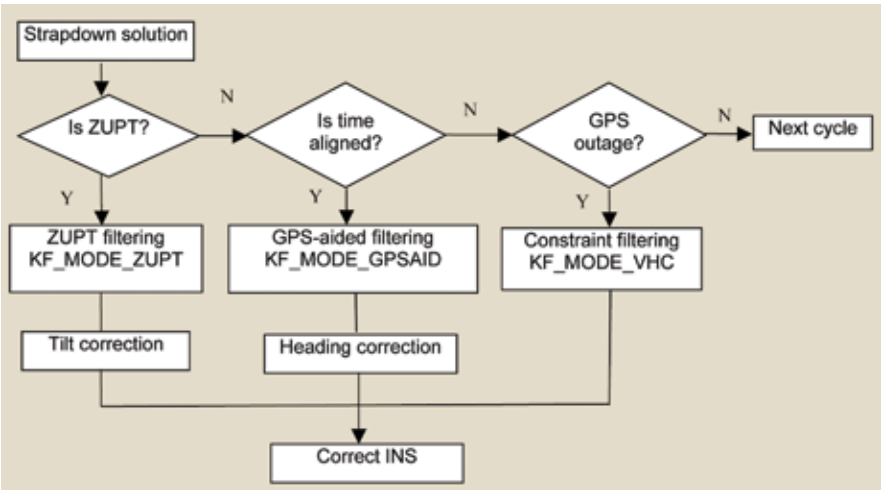

FIGURE 7 Flowchart of the GPS/INS integration algorithm

Software and Algorithm Details

The software has the overall task of taking the sensor input data and providing an integrated solution, while also providing the user with status information and facilitating user control. The software performs a loosely coupled integration, taking the PVT data from the GPS receiver.

Tightly coupled integration is also possible, using satellite range data from the GPS receiver. The tightly coupled approach has some advantage over loosely coupled, as range data from the GPS receiver can still be used to assist the INS when a PVT solution is not available. However, range data is not included in any standard NMEA message, and not all receivers include a proprietary range data message.

The software has four main tasks: the user interface (UI), time-synchronization and message decoding (TS), strapdown INS (SINS), and Kalman filtering (KF). The low update rate GPS and PPS data are collected in the UI task. The time-synchronization procedure is active when the GPS, PPS, and INS are available. The SINS task runs once the INS data are time-synchronized. The KF task runs when both the GPS and INS data are time-synchronized.

The high update rate INS data are collected in the TS task. This task aligns the INS data with the GPS time frame so that comparison of the INS and GPS data are possible. The strapdown inertial computation implements the INS mechanism equations in the navigation coordinate system (n-frame). In other words, the strapdown computation produces the position, velocity, and attitude of the host platform from the specific force sensed by the accelerometers and the angular rate sensed by the gyroscopes.

The Kalman filter uses the GPS data to estimate the errors of the inertial solution. The error estimates are then used to correct the inertial solution and improve the result. The embedded system implements a Kalman filter of 15 states including position, velocity, and angular errors, as well as the inertial sensors errors. The sensor bias and scale factor error are taken as a net sum to reduce the real-time computation load. The INS errors are described in the n-frame.

The integration algorithm combines the alignment and the integrated navigation. The alignment period is set to 120 seconds. Therefore, the IMU requires a static period of no less than 120 seconds (= two minutes) in order to perform coarse alignment. **Figure 7** presents a flowchart of the integration algorithm.

The GPS/INS system works in one of three modes, ZUPT (zero velocity update), GPSAID, or VHC. ZUPT mode operates when either of two conditions is satisfied: the system is in the alignment phase or the INS is stationary. The system works in the GPSAID mode if the GPS and INS data are time-synchronized. The VHC mode means "velocity and height constraint" and is described further in the article by S. Godha and M. E. Cannon cited in Additional Resources. The system uses VHC mode if no GPS data are available.

The tilt correction is disabled when the system is in the alignment phase. It is enabled when the INS is stationary. The INS has a gyro bias of 30 deg/hr, a magnitude almost twice the earth rotation rate. Consequently, the INS-derived heading is not useful in the alignment phase, and external heading data are needed to aid the system. For GPS/INS integration, the heading derived from the GPS velocity is a natural choice. Heading correction is used after alignment is completed whenever GPS is available.

From testing, the following parameters were found to have effects on the performance of the GPS/INS system: (1) the length of alignment period, (2) Gauss-Markov time constants of the gyro and accelerometer biases, (3) heading correction, (4) tilt correction, and (5) the process noise covariance (Q) and measurement noise covariance (R) matrices of the integration Kalman filter.

Finally, the corrected INS solution is logged into a file on the CF card or sent to the outside world via a serial data port. The data can then be used for real-time mapping, navigation or geo-registration of images.

Demonstrator Projects

The project team is currently working towards two demonstrator projects. One is an airborne oceanographic application to provide positioning data for aerial Lidar image geo-registration on a joint project among several departments at UNSW. The platform is an aircraft owned and operated by the Department of Aviation at UNSW, which is pictured in the accompanying photograph.

The second demonstrator project is with the NSW Department of Lands to develop an inexpensive and robust vehicle-based positioning system. This application requires an inexpensive INS that, along with a GPS receiver and our sensor fusion device, can provide a real-

InsideGNSS

time position data stream to feed into a GIS application running on a laptop computer.

The land vehicular system must cope with signal blockages due to steep terrain and the tree canopy (see accompanying photo). Some test results can be found in the article by Y. Li et alia (2008), listed in Additional Resources.

Future developments

Many challenges lie ahead for the project team. Gaining experience with the new generation of inexpensive MEMS INS sensors to replace the venerable tactical-grade IMU is likely to keep the team very busy. The goal is to find a sensor that costs less than $1,000 and can provide acceptable performance in our system (e.g., bridge a GPS outage of more than tens of seconds).

Algorithms for both loosely coupled and tightly coupled integration have been developed and tested, but the performance advantages of the tightly coupled system have not been properly quantified in a real-time system. We also need to address performance issues involving the sensor-fusing device — reliability must be improved and we need to perform more testing over a wide variety of environments.

The performance of the system can be improved by implementing some of the strategies mentioned earlier, but it is likely that the design will grow too big for the processor development board that we have been using to date. We also use another development board from the same manufacturer that has a larger capacity FPGA on it, but the FPGA does not have the speed performance of the chips in the latest version of our processor.

Fortunately, a large range of development boards available from numerous FPGA vendors that can be used for our application, and the newer ones tend to have larger and faster FPGA chips on them.

Finally. . . .

GPS outages are a fact of life, but navigation must go on. The prototype devices developed at UNSW can effectively integrate GPS and INS in real-time to bridge the outage gaps. Great potential exists to include this functionality in car navigation units, once the cost and performance of MEMS-based units comes down. In the meantime, specialised applications can benefit from this low-cost GPS/INS system developed at UNSW.

(above) Aircraft used in UNSW airborne oceanographic demonstrator project
(below) The GPS/INS system used in NSW Department of Lands fire-trail surveying demonstrator project must cope with steep terrain and tree canopy.

InsideGNSS

Manufacturers

The GPS/INS system incorporates an 8200-HP receiver from **OmniStar Pty Ltd.**, West Perth, Australia. Several other GPS receivers were used in the initial phase of the development including the Motorola M12, the MG5001 from **Signav Pty Ltd.**, Chatswood, NSW, Australia, and the CMC Allstar (now offered by **NovAtel, Inc.**, Calgary, Alberta, Canada). The tactical-grade inertial measurement unit was the C-MIGITS II from **Boeing Defense and Space Group**, Anaheim, California. The raster map was from NSW Department of Lands topographic 1:25,000 "Parramatta river" displayed on the **OziExplorer mapping software**, Brisbane Australia. The photo-based map was generated by Google Earth, **Google Inc.**, Mountain View, California, USA.

Seamless Nav *continued on page 53*

Seamless Nav *continued from page 44*

The embedded processor development board used to design the sensor-fusion device was the NiosII (Stratix edition) from **Altera Corporation**, San Jose, California, USA. We also use a Cyclone II edition NiosII development board for some design activities. "Quartus" FPGA design software. Under consideration is the new NiosII Development kit (Stratix II edition), which has an FPGA chip with six times the capacity that can run at higher clock frequencies than the Stratix edition.

Additional Resources

[1] Altera, *Nios development board – reference manual*, Stratix edition, http://www.altera.com, 2003

[2] Altera, *Nios II software developer's handbook*, <http://www.altera.com>, 2007

[3] Boeing North American Inc., *User's manual of C-MIGITS II*, 1997

[4] Mumford, P., and Y. Li, J. Wang, C. Rizos, and W. Ding, "A time-synchronisation device for tightly coupled GPS/INS integration," *Proceedings of IGNSS Symposium 2006*, Holiday Inn Surfers Paradise, Australia, July 17-21, 2006

[5] Godha, S., and M. E. Cannon, "GPS/MEMS INS integrated system for navigation in urban areas," *GPS Solutions*, Vol. 11, pp. 193-203, 2007

[6] Li, Y., and P. Mumford and C. Rizos, "A real-time GPS/INS integrated system," *Coordinates*, vol. IV, no. 3, pp. 12-17, March 2008

[7] Massa, A. J., *Embedded software development with eCos*, Prentice Hall Professional Technical Reference, New Jersey, 2002

Authors

Yong Li, Ph.D., is a senior research fellow at the Satellite Navigation and Positioning (SNAP) Lab within the School of Surveying & Spatial Information Systems, the University of New South Wales (UNSW), Sydney, Australia. He was involved in development of a spaceborne GPS attitude determination receiver and a MEMS inertial sensors/GPS system for a sport application. His current interests include integration of GPS, INS, and pseudolite (Locata), attitude determination, GPS receiver technique, FPGA technology, and its application to navigation, and optimal estimation/filtering theory and applications.

Peter Mumford is a research assistant in the School of Surveying & Spatial Information Systems at UNSW. He is part of the SNAP Lab, specializing in FPGA and embedded software design for GNSS applications. He has an engineering degree in surveying (UNSW) and a science degree in mathematics (Sydney University).

Professor Chris Rizos is a graduate of UNSW, obtaining a Ph.D. in satellite geodesy. He is currently the head of the School of Surveying & Spatial Information Systems at UNSW. Rizos has been researching the technology and applications of GPS since 1985 and over a decade ago established the Satellite Navigation and Positioning group at UNSW, today the largest and best known academic GPS and wireless location technology R&D laboratory in Australia. Rizos is the vice-president of the International Association of Geodesy (IAG), a member of the governing board of the International GNSS Service, and a member of the IAG's Global Geodetic Observing System Steering Committee. He is a Fellow of the IAG and of the Australian Institute of Navigation. IG

InsideGNSS