



**International Global Navigation Satellite Systems Society
IGNSS Symposium 2007**

The University of New South Wales, Sydney, Australia
4 – 6 December, 2007

A low-cost field re-configurable real-time GPS/INS integrated system – design and implementation

Yong Li, Peter Mumford, Jinling Wang, and Chris Rizos

University of New South Wales, Australia

Tel: 61 2 9385 4173, Fax: 61 2 9313 7493, Email: yong.li@unsw.edu.au

ABSTRACT

This paper describes the development of a real-time GPS/INS integrated system based on a field programmable gate array (FPGA) platform. The objective is to develop a generic hardware/software platform for positioning and imaging sensor integration. Compared with the application-specific integrated circuit (ASIC) approach, the FPGA approach can shorten the research and development (R&D) cycle. Its reprogrammable hardware provides a system design methodology of lower risk. It also allows maximum flexibility, being able to integrate a wide range of GPS and INS sensor packages.

The hardware design is built on the Altera's Nios II development kit. A time-sync universal asynchronous receiver/transmitter (UART) is implemented, which accesses a free-running clock counter and reads the count at the instant of start bit of a serial transmission. The count is then used to align the time of the INS data with the GPS time frame. The embedded software is implemented in a multi-thread configuration based on the Embedded Configurable Operating System (eCos). Four threads perform tasks that include; real-time decoding GPS and INS binary streams, time synchronisation, strapdown INS computation, and integration Kalman filtering. The system records the raw data and the solution onto a compact flash card for replay or post-processing purposes. The real-time solution is also sent out through a serial port to a mobile device which can access the internet. As a result, the real-time solution can be visualised on GoogleEarth as well

as monitored from a command centre.

Preliminary tests demonstrate the feasibility of this type of system on the FPGA platform, and the functionality of the system including; the stability and accuracy of the time synchronisation mechanism, the performance of the hardware and software architecture, and the workability of the algorithm.

KEYWORDS: GPS; INS; FPGA; multisensor integration; embedded system.

1. INTRODUCTION

Although the Global Navigation Satellite Systems (GNSS) technology is developing quickly, the major disadvantage of GNSS will still exist even when the European Galileo system is fully operational, that is, signal blockage due to obstructions and the low power of the signals. The combination of GNSS with a self-contained inertial navigation system (INS) provides an ideal solution, which can not only address the weakness of GNSS, but also bound the INS error that grows with the time when the INS works alone. The integrated system can provide a continuous position, velocity, and attitude solution at a high output rate. These advantages drive GNSS/INS integration in military and civil applications, and the rapid progress in the development of microelectromechanical sensors (MEMS) makes the integration of MEMS inertial sensors and GNSS especially attractive because of the former's small size, low power consumption, and low cost (Liccardo and Rios 2006).

Realising the market opportunities of using MEMS integrated systems brings challenges, e.g. short R&D cycle, low risk, and efficiently fixing the hardware and software bugs introduced by implementing new sensors and algorithms. In addressing these challenges, a multisensor integration platform based on a field programmable gate array (FPGA) is being developed at the Satellite Navigation and Positioning Laboratory (SNAPlab), University of New South Wales. The objective is to develop a generic hardware and software platform for positioning and imaging sensor integration to support a range of applications. The biggest advantage of the FPGA-based system is that all the hardware and software components of the system are field re-programmable without any hardware changes, with even the processor of the system itself is "soft". A "hardcopy" FPGA can be made after the system has been sufficiently tested.

An FPGA is an integrated circuit capable of implementing digital circuits by means of a configuration process. FPGAs are made up of three principal components: configurable logic blocks (CLBs), input-output blocks (IOBs) and connection blocks (Hidalgo *et al* 2003). These elements are distributed in the form of a matrix in the FPGA, and the designer can use CLBs to implement the specified logic circuits which are routed together using the connection blocks and interconnection lines. The IOBs connect the circuits with the external world for data exchange. Two major hardware description languages (HDLs) are popularly used for the FPGA hard design today, namely Verilog and VHDL, both of which are IEEE standards. The US west coast and Asia area prefer Verilog, whilst the US east coast and Europe more frequently use VHDL (Meyer-Baese 2001). The authors have used the Altera HDL "AHDL" for most of the design.

This paper describes the design and implementation of the FPGA-based GPS/INS integration system developed in the SNAPlab. The development comprised three phases: (1) algorithm and software development for tight and loose integration; (2) time-sync data logging device

development; and (3) real-time integrated system on the FPGA. The project is now near the end of the third phase. The time-sync device completed in the second phase has been modified and is used as the hardware platform for the real-time system. The software developed in the first phase has been ported to the FPGA and runs with the support of eCos. Tests are currently underway to verify the design and functionality of the real-time system. Further refinement of the hardware logic design and software optimisation is discussed along with the performance of the system.

2. OVERALL SYSTEM DESCRIPTION

Building an embedded GPS/INS system in an FPGA involves system requirements analysis, hardware design, software design, and algorithm development. Figure 1 illustrates the system architecture of a typical setup of a real-time GPS/INS integrated system.

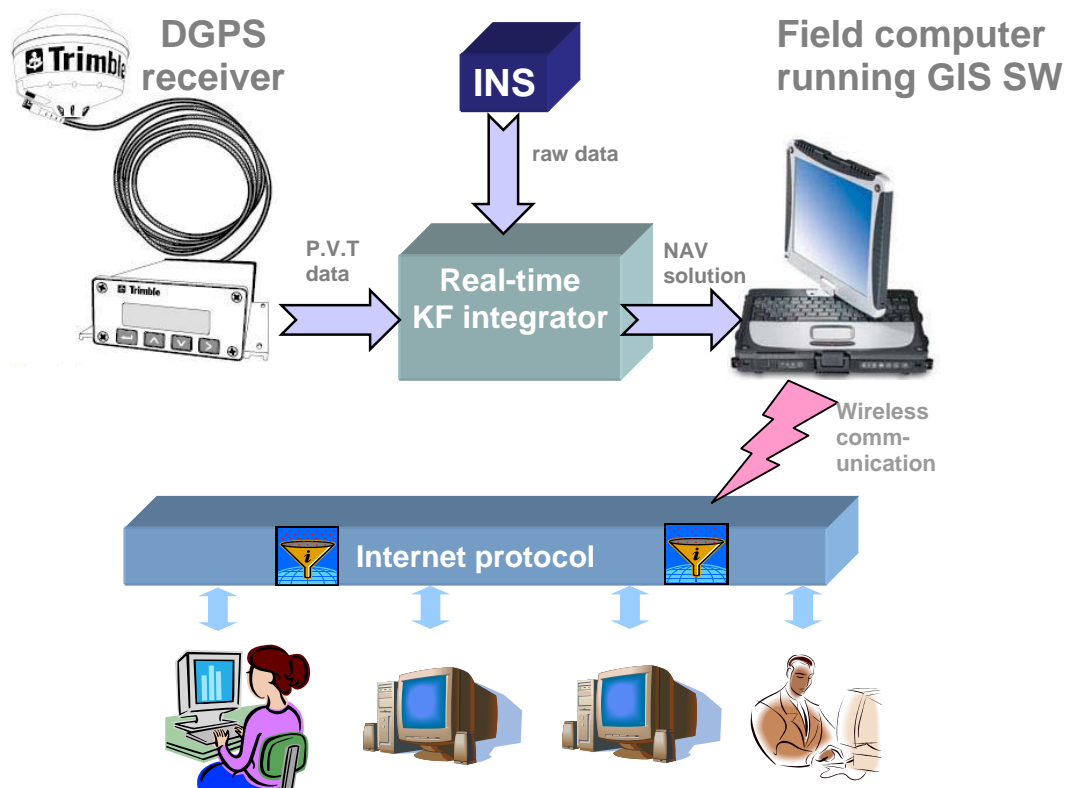


Figure 1. Illustration of the real-time GPS/INS integrated system

As shown in Figure 1, the GPS and INS data are fed into the FPGA system where the real-time Kalman filter estimates the INS errors that are then used to correct the INS solution. The corrected solution is sent to a field computer on which a geographic information system (GIS) runs. The INS solution is plotted onto a map and visualised on the GIS platform. The solution and data collected in the field is sent to a command or monitoring centre via wireless communication, i.e. wireless internet, or the mobile phone network. High-level commands or decisions can be made by the centre on the basis of the real-time information that is received. In comparison with post-processing systems, real-time systems can respond to urgent events promptly, with minimum delay. This is vital in, for example, emergency service applications.

The Altera Stratix EP1S10 FPGA development board is used in the prototype system. The real-time system is built around the Nios II soft-core on a Stratix EP1s10 device, which is configured with 1MB static RAM (512KB × 2), 16MB SDRAM, and 8MB flash memory. The Stratix EP1S10 features 10,570 logic elements and 920KB of on-chip memory (Altera 2003). The real-time device is currently configured with four UARTs, two of them for INS and GPS input, and another two for integration result output. The GPS pulse-per-second (PPS) signal is required for the time synchronisation process and is connected to the prototype device via a BNC socket. One output UART is connected to the computer running the GIS, and another is connected to a handheld device that can access the internet. The device has an LCD screen for menu and status information display and four buttons for option selection and operation control.

The embedded software is developed using the Nios II integrated development environment (IDE). A special version of eCos – ‘the eCos for Nios II’ provides support of the FAT32 file I/O in Compact Flash (CF) card, multi-task programming, LCD display, and interrupts from UARTs and buttons. The main specifications of the real-time system are listed in Table 1.

FPGA	Altera’s Stratix EP1S10
Processor	Nios II
Oscillator	50MHz
SRAM	512KB x 2
SDRAM	16MB
Flash	8MB
Embedded OS	eCos for Nios II, ver5.1.
Interface	one LCD; four Buttons; one CF card slot; four UARTs; one BNC.
GPS	OmniStar-HP8200
INS	C-MIGITS II

Table 1. Summary of the real-time system

The INS used is the Boeing's C-MIGITS II, which is a so-called tactical grade inertial measurement unit (IMU) that provides raw inertial data (Boeing 1997). The C-MIGITS II has internal time synchronisation functionality which provides a reference for evaluating the timing result from the FPGA-based system. An OmniStar-HP8200 GPS receiver is used to provide the PPS signal and the GPS navigation solution. The OmniStar differential service is always turned on while it is available.

The device can self-boot from on-board flash memory for stand-alone operation, or can be configured to run from RAM for development and debugging. The C-MIGITS II and the OmniStar-HP8200 are automatically initialised to send out the requested data through their UARTs. The threads in the embedded firmware run to perform scheduled tasks.

3. HARDWARE

The Stratix EP1S10 FPGA development board includes an FPGA chip, associated support chips, a range of I/O options, CF card socket and power supply conditioning. Custom designed logic has been developed for the FPGA to provide count stamping on the incoming serial data streams. A NiosII soft-core processor residing in the FPGA chip hosts

the application software that interfaces with the user and controls the custom logic and CF card operations.

The logic design was developed in the Altera Hardware Description Language (AHDL), using the Altera synthesis and fit tool - 'Quartus'. Quartus also includes the System On a Programmable Chip (SOPC) builder that is used to generate a NiosII soft-core processor (Mumford *et al* 2006). A Nios II processor core is a hardware design that implements the Nios II instruction set and supports the functional units described in the Nios II processor handbook (Altera 2005). Figure 2 depicts the hardware box of the real-time FPGA-based GPS/INS system.

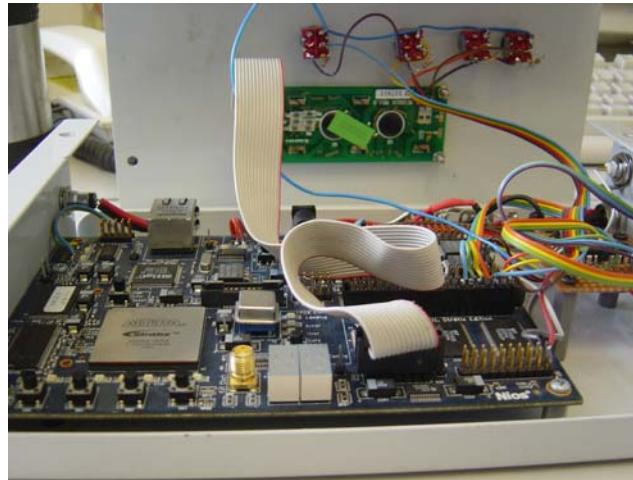


Figure 2. Hardware of the real-time FPGA-based GPS/INS system

The Nios II development kit provides the means to download the hardware design file and the firmware to the flash memory (AMD AM29LV065D). When power is applied to the board, a configuration controller device attempts to configure the FPGA with hardware configuration data stored in flash memory.

Greatly benefiting from the ready-to-use Nios II soft-core, the design of the embedded GPS/INS system can focus on the user-specified hardware circuits, in this case the UART design central for the time synchronisation of the GPS and INS data. This "time-sync UART" logic is attached to the processor as a memory-mapped peripheral with one interrupt line. The connection of the time-sync UART and the Nios II is depicted in Figure 3. The UART must detect transmission, receive the data in serial format, strip off the start- and stop-bits, and store the data word in a parallel format.

The UART accesses a free-running counter that is latched at the start bit of a serial transmission. This count is appended to the incoming byte and placed in a first-in-first-out (FIFO) buffer. When the FIFO buffer is nearly full, an interrupt is generated to notify the system to collect the data for further processing. The biggest difference between the real-time system and the data logging system is the setup of the FIFO buffers. In the data logging system, the nearly full point can be set close to the top of the buffer, without concern for the delay of receiving the data messages. However in the real-time system the nearly full point must be low to ensure the data message of the current epoch can be captured in time and processed in the present computation cycle. This means that there are more interrupts in the

real-time system.

The PPS signal along with GPS time data are used in an interpolation algorithm to calculate the time-of-arrival of serial data from the INS. As a result, the INS data is time-tagged with GPS time and therefore the INS data is available to compare with the GPS data in the GPS time frame.

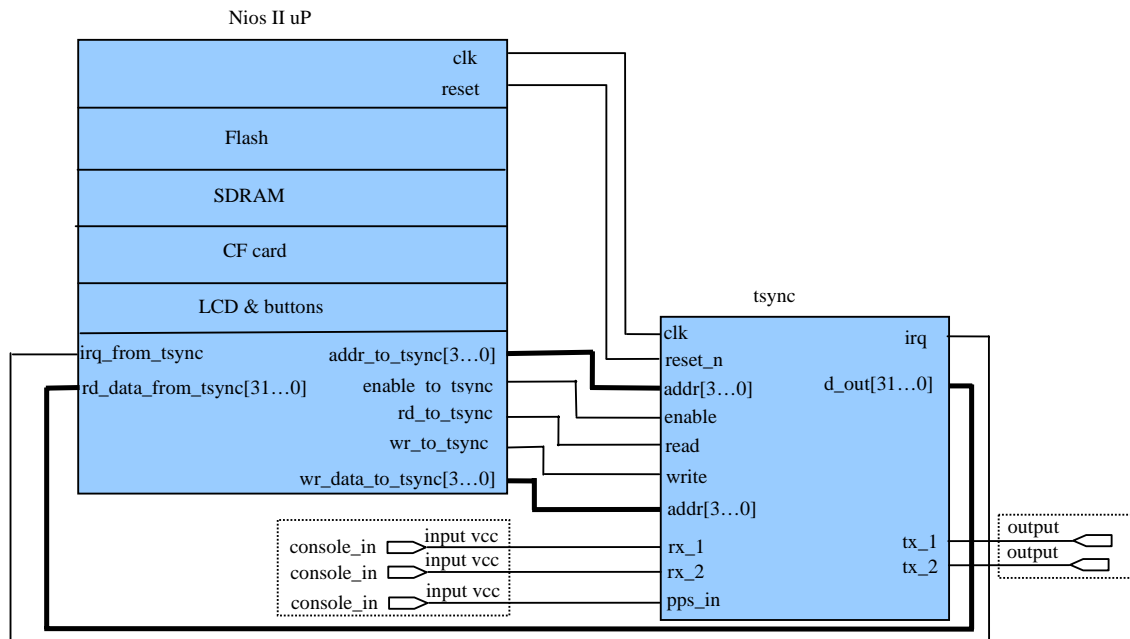


Figure 3. Connection of the time-sync UART and Nios II

4. SOFTWARE

The firmware running on the Nios processor has the primary task of collecting data from the UART and processing it. The eCos is used to provide FAT32 file I/O support to the CF card as well as multi-tasking support. The eCos is an open source operating system developed specifically for use in the embedded environment. The eCos system includes all the tools necessary to develop embedded applications, including eCos software configuration and build tools, and GNU-based compilers, assemblers, linkers, debuggers, and simulators. It provides all the functionality required for general embedded application support including device drivers, memory management, exception handling and math libraries (Nios Community Forum 2005).

The software is designed in a structure consisting of four threads, which are the user interface (UI), time synchronisation (TS), strapdown INS (SDINS), and Kalman filtering (KF). The low update rate GPS and PPS data are collected in the UI thread, and receiving the high update rate IMU data is assigned to TS thread. The time-sync procedure is active when the GPS, PPS, and IMU are available. The time synchronised data are stored in circular buffers and the SDINS thread picks up the data from the buffers to compute the strapdown inertial solution. The KF thread runs once the GPS and INS data matches in the time. Each thread contains its own context or workspace to perform its operations and a priority level to

execute. It is the job of the scheduler to determine which thread is entitled to run at any given time. The kernel contains API functions for controlling threads within an application (Massa 2002).

Although single thread structure could be chosen for a real-time GPS/INS system, there are drawbacks. In the single thread program, in that the tasks like data accumulation, numerical computation, and CF I/O are performed in sequence. The time consuming tasks like SDINS and KF may block the data accumulation for a while and therefore cause loss of data. A multi-threaded program can avoid this disadvantage by assigning threads different priority levels. Whenever the data from the UARTs are ready to be drained, the time-consuming threads will suspend and allow the system to execute the data accumulation threads.

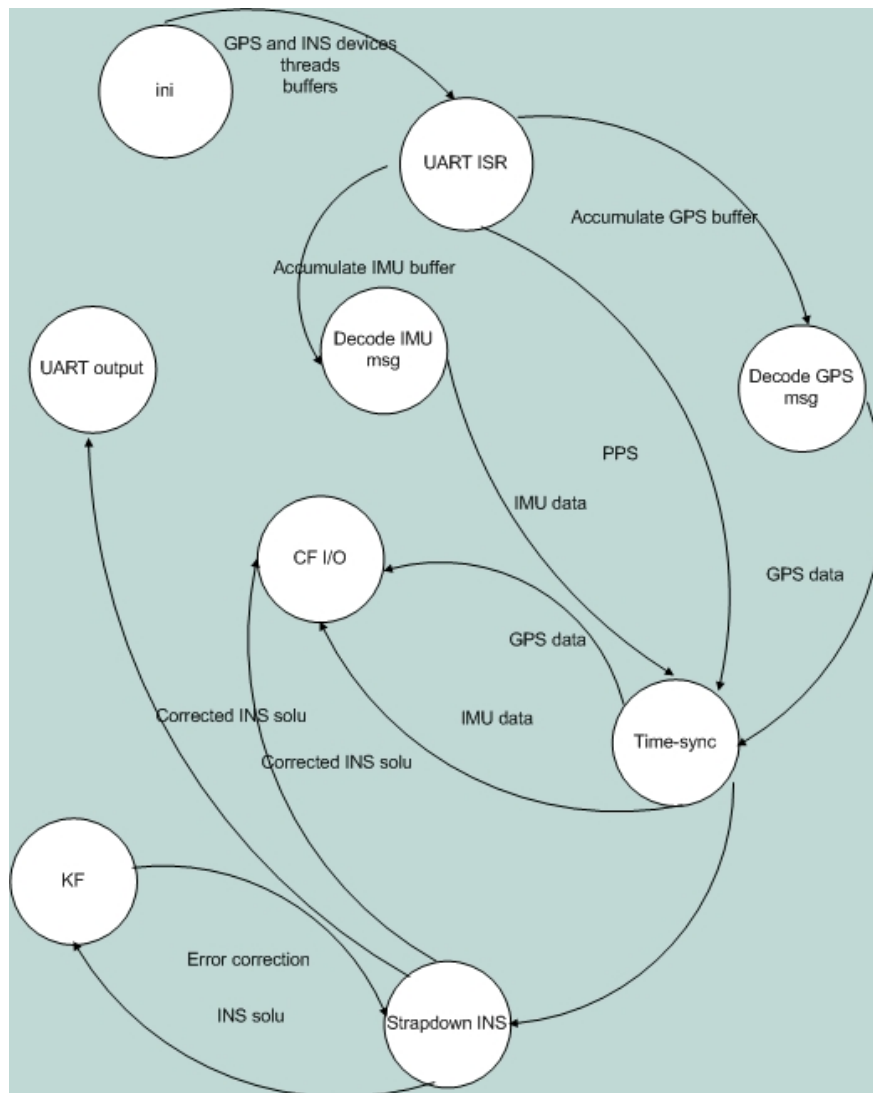


Figure 4. Operation diagram of the embedded software

FIFO buffers are used in the UARTs for receiving the data from GPS and INS. This is essential to reduce the number of interrupts which would otherwise be generated every time a new byte is available in the UART. These frequent calls to the Interrupt Service Routine (ISR) would place a heavy burden on the processor due to the overheads associated with the

ISR call. Interrupts are also generated by pressing the buttons. Button events are used to control the operation of the program, i.e. starting/stopping data logging to CF files, and changing the display information on the LCD.

The operation procedure of the program is depicted in Figure 4. The system first initialises the threads, assigning them priority levels of execution, and the stack sizes. It also allocates the memory for the circular buffers, and commands the GPS and INS devices to output the requested messages. The ISR of UART event notifies the UI or TS thread to drain the data from the UART FIFO buffer and performs the decoding procedure to convert the binary stream to the data messages. The TS procedure aligns the IMU data with the GPS time frame so that comparison of the IMU and GPS data is possible. The strapdown navigation solution is calculated from the time-synced IMU data. Using the GPS data, the Kalman filter further estimates the errors residing in the inertial solution and the inertial sensors. The error estimates are used to correct the inertial solution and improve the result. The corrected INS solution is sent to the external world via a UART port. Meanwhile the system stores the time-synced IMU data and GPS data together with the corrected inertial solution onto the CF card for replay or post-processing.

The corrected navigation solution is sent out in a pre-defined format to an external device, e.g. a handheld computer, where a program runs to receive the solution and send it to GoogleEarth via the internet. The ".kml" file describes the server address on the internet to which the client sends the data. By linking to the address, the position of the host platform can be monitored on any PC in the command/monitor center using GoogleEarth. Figure 5 depicts a screen copy of a test at the top of building of the SNAPlab. Lead by the real-time GPS/INS solution, the GoogleEarth automatically zooms in to the area around the SNAPlab building.

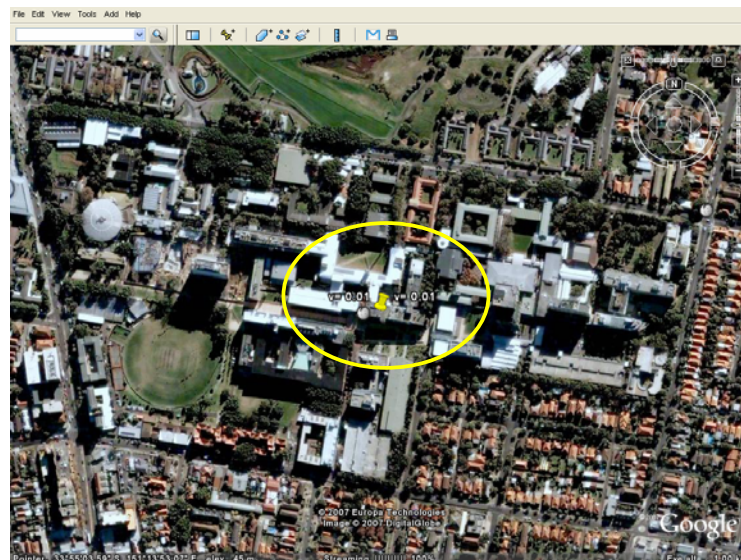


Figure 5. Real-time GPS/INS solution on the GoogleEarth

5. ALGORITHMS

Strapdown inertial computation can be performed in either a navigation coordinate system (n-

frame) or an earth-centred-earth-fixed coordinate system (e-frame). The INS errors can be mathematically described in either one of the frames. The most-commonly used error equations are expressed in the n-frame, being the so-called psi-angle model (Bar-Itzhack and Berman 1988). This is the model used in the GPS/INS integration Kalman filter. The embedded system implements a Kalman filter of 15-states include position, velocity, and angular errors, as well as the inertial sensors errors. The sensor bias and scale factor error is taken as a net sum to reduce the real-time computation load. More states could describe the system behaviour better, and can be implemented in the post-processing software.

The C-MIGITS II is initialised to output 10Hz raw IMU data (Δv and $\Delta \theta$). The GPS receiver is initialised to output at a 1Hz rate. Therefore, the IMU data is ten times faster than the GPS data. The system runs in the SDINS mode between every GPS output. The integration Kalman filter runs when the GPS data arrives.

The data processing can be divided into three operation modes: (1) the coarse alignment; (2) the fine alignment; and (3) the strapdown INS and integration Kalman filtering. During the coarse alignment, the platform remains static while the tilt angles are computed from the accelerometer data. In addition, the sensors noise levels can be estimated during the coarse alignment (Kennedy 2006). The heading angle can be roughly estimated from gyro-compassing the gyroscope data. However the C-MIGITS II has a gyro bias of 30deg/hr (Boeing 1997), a magnitude almost twice the earth rotation rate and hence the heading result derived from the C-MIGITS II is not very meaningful. Because the GPS heading becomes observable only when the host platform begins to move, a heading correction can be obtained from the GPS velocity when the platform is moving. During the fine alignment, the Kalman filter estimates the tilt errors and the sensor biases. Due to the weak observability of heading angle, the fine alignment cannot prevent the heading from gradually drifting. Therefore the fine alignment period should not be too long to avoid accumulated error introduced by the heading error.

6. TESTS

With the 50MHz system design, the timing resolution of the counter is 5.12 μ s. To compare the time derived by the FPGA device with the time of the INS output, the FPGA-based system has demonstrated time-sync accuracy of better than 0.3ms. The system has potentially higher accuracy because it can reveal the C-MIGITS II's 10 μ s/sec clock drift, as analysed in (Li *et al* 2006).

Long-term tests of the real-time performance of the system have been conducted in the laboratory. The focus of testing included stability of the multi-threaded firmware, real-time decoding of the GPS and INS data messages, real-time time synchronisation, multiple-stage circular buffering, float-pointing calculation, stability of the Kalman filtering, button interrupts and response, CF file I/O, data output through additional UARTs, and interfacing with GoogleEarth.

Dynamic tests are planned in the coming months to further evaluate the real-time performance of the embedded system under dynamic conditions. The algorithm was developed and tested independently using the data collected by the FPGA-based data logging system (Li *et al* 2006; Mumford *et al* 2006). The result of the algorithm test is presented below. This well-tested algorithm will be migrated to the embedded system at a later stage.

6.1 Static test

An important requirement of the system is to estimate the attitude angles and the inertial sensor biases. Static data is used to evaluate this capability because the tilt angles are invariant during the test. Without correction from the Kalman filtering estimation, the strapdown attitude solution gradually drifts with the time. In the test, a 160-second alignment period is pre-defined. The zero-velocity is used to update the attitude and sensor errors in the alignment. Once the alignment process has been completed, the system automatically changes to the navigation mode where the GPS position data is used to update the estimate. Figure 6 depicts the tilt solution from a 1360-second test. It is evident that the solution converges gradually during the alignment phase, and remains stable within a small range of $\pm 0.05\text{deg}$ during the navigation phase.

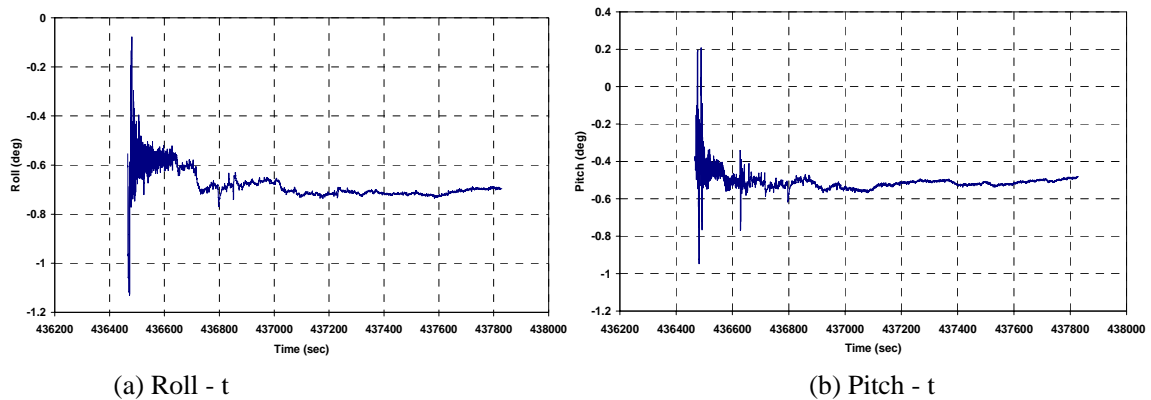


Figure 6. Tilt angle solution in the static test

6.2 Van test

Kinematic data was collected along roads in a variety of environments, including a race track with significant attitude manoeuvres, a highway, forested mountain areas with GPS signal blockage, and also through a tunnel. The vehicle and device setup is shown in Figure 7.



(a) Vehicle



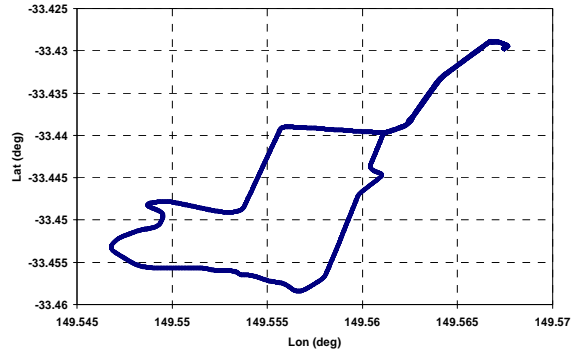
(b) FPGA-based GPS/INS system

Figure 7. Vehicle set up for kinematic testing

The ground trajectory depicted in Figure 8 shows the result from the test carried out around the Mount Panorama racing circuit, Bathurst, NSW. The velocity and attitude solution is depicted in Figure 9. Without a reference to evaluate the accuracy of the result, analysis concentrates on the stability of the solution, and the ability to capture the attitude manoeuvres.

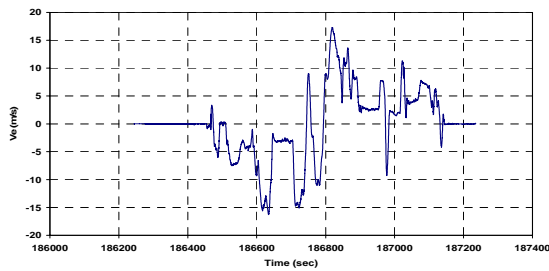


(a) Display on the map running on tablet PC

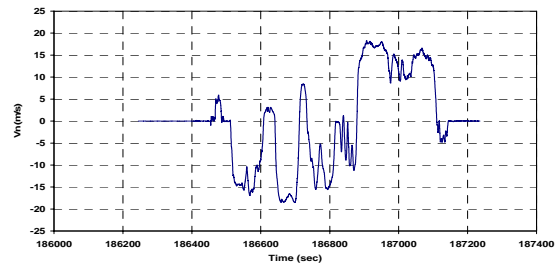


(b) GPS/INS trajectory

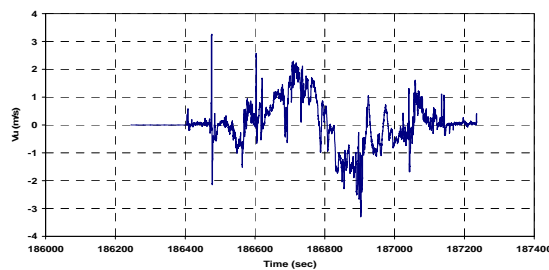
Figure 8. Trajectory from the GPS/INS solution



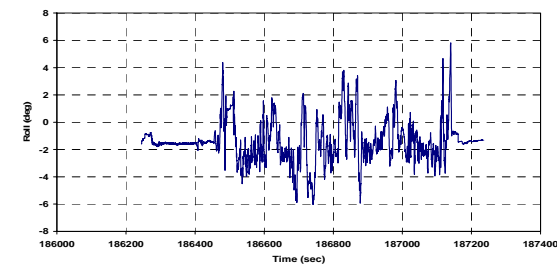
(a) $V_e - t$



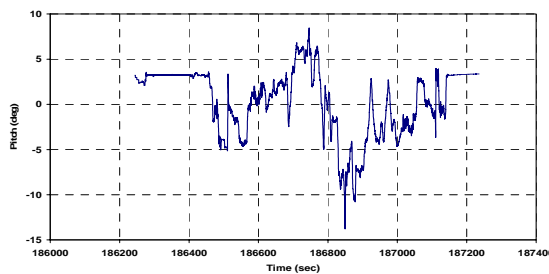
(b) $V_n - t$



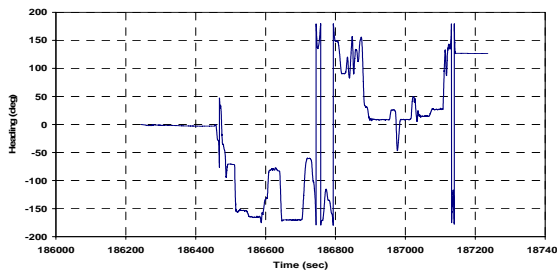
(c) $V_u - t$



(d) Roll - t



(e) Pitch - t



(f) Heading - t

Figure 9. Velocity and attitude solution from the GPS/INS integration

In comparison with the INS-only solution, the integrated velocity is stable and correctly reflects the movements of the vehicle – backward and forward when the vehicle is driven from the parking site, stops to wait for the traffic light, and speeding up. The integrated attitude solution is also stable, and properly reflects the angular movement of the vehicle – especially on pitch (9e) and heading (9f). The tilt angles (roll in 9d and pitch in 9e) converge when the vehicle is static, however the heading (9f) remains on the initial value with slight drift during the static period. The heading becomes observable when the vehicle moves and the GPS-derived heading is then used to initialise the heading. From Figure 9f, it is easy to see that the heading quickly changes from the initial value to the correct value when the vehicle starts to move. In the last section of the heading curve in Figure 9f, the heading has a jump of about 126deg in comparison with the initial value. In comparison with the compass data collected in the test, the angle at the end reflects the correct heading direction. Because the vehicle starts and stops at the same site and heads in the same direction the correct heading angle at the end demonstrates that the integration Kalman filtering works properly. Figure 10 compares the GPS solution with the GPS/INS integrated solution in a 10-second period. It is clear that the 10Hz corrected INS solution bridges the 1Hz GPS position smoothly in this application.

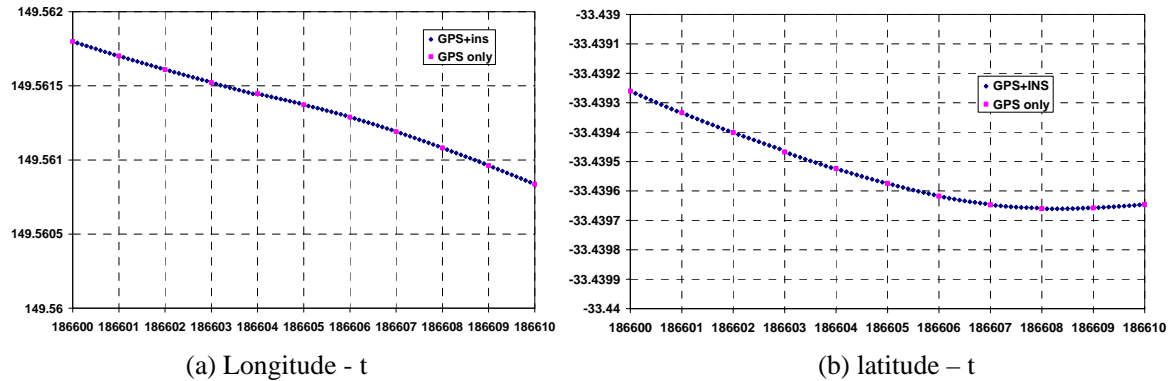


Figure 10. Comparison of solutions (the GPS/INS integration vs GPS-only)

6.3 Performance in a tunnel

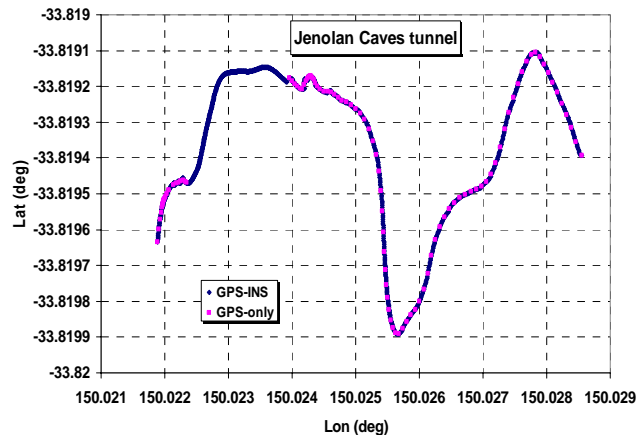
One test was performed in the Jenolan Caves area. The road goes through a tunnel near the Jenolan Caves township. Table 2 lists the last GPS data point before entering the tunnel and the first GPS data point after leaving the tunnel. The tunnel has a length of 197.7m in east, and 32.8m in north. It took 45sec to drive through the tunnel, including the time waiting due to traffic. Figure 11b depicts the trajectory from GPS and GPS/INS solutions. During the 45-second GPS outage in the tunnel, the INS solution correctly outlines the shape of the tunnel. This result demonstrates that the integration system is working properly – once an accurate navigation solution and the inertial sensor biases have been estimated before the vehicle enters the tunnel.

	t (sec)	Lon (deg)	Lat (deg)
entrance	193563	150.0223443	-33.8194693
exit	193608	150.0239595	-33.8191757
Interval/distance	45sec	179.7 m	32.8m

Table 2 The positions of the ends of the tunnel



(a) Entrance of the tunnel



(b) GPS/INS solution in the tunnel

Figure 11. Comparison of solutions in the tunnel

7. CONCLUSIONS

An FPGA-based real-time GPS/INS integrated system has been developed. The hardware is built around the Nios II soft-core. A time-sync UART is designed to connect with the Nios II processor system to enable communication between the Nios II and the GPS and INS devices, as well as time-synchronise the GPS and INS data streams with a resolution of $5.12\mu\text{s}$.

The embedded software has been developed using eCos – an open source embedded operating system. The software is programmed in four threads to implement multiple tasks; decoding the GPS and INS data streams, time synchronisation, strapdown inertial computation, and the integration Kalman filtering. With eCos support, the software implements the FAT32 filing system for CF card I/O, operation status display on the LCD, and button controls. The real-time solution is sent out via two additional UARTs and can be displayed on GoogleEarth. Long-term tests have demonstrated the functionality and the operational robustness of the embedded software.

The GPS/INS integrated algorithm has been developed and tested in the laboratory and in the field. The results have primarily demonstrated that the integration Kalman filter estimates the inertial errors correctly, to compensate for the drift in the inertial solution. The results of the test in the Jenolan Caves tunnel have shown that the corrected INS can bridge the solution during the GPS outage in the tunnel with a reasonable accuracy.

ACKNOWLEDGEMENTS

The project is funded under the Australian Cooperative Research Centre for Spatial Information (CRC-SI). The authors would like to acknowledge the support from the CRC-SI. Thanks for Henry Hu for writing the C++ code for interfacing with GoogleEarth.

REFERENCES

- Altera (2003) *Nios development board – reference manual*, Stratix edition, <http://www.altera.com>.
- Altera (2005) *Nios II software developer's handbook*, <http://www.altera.com>.
- Bar-Itzhack IY and Berman N (1988) Control theoretical approach to inertial navigation systems, *Journal of Guidance, Control and Dynamics*, 11 (3): 237-245.
- Boeing North American Inc (1997) *User's manual of C-MIGITS II*.
- Buck TM, Wilmot J, and Cook MJ (2006) A high G, MEMS based, deeply integrated, INS/GPS, guidance, navigation and control flight management unit, *Proceedings of IEEE/ION PLANS 2006*, San Diego, California, 25-27 April.
- Kennedy S, Hamilton J, Martell H (2006) Architecture and system performance of SPAN – NovAtel's GPS/INS solution, *Proceedings of IEEE/ION PLANS 2006*, San Diego, California, 25-27 April.
- Li Y, Mumford P, Wang J, and Rizos C (2006) Development of a GPS/INS integrated system on the field programmable gate array platform, *Proceedings of ION GNSS 2006*, Forth Worth, Texas, 26-30 September, 2222-2231.
- Liccardo D and Rios J (2006) Robust sensor fusion for GPS inertial measurement units in diverse flight environments, *GPS World*, July 2006, 33-39.
- Hidalgo JI, Fernandez F, Lanchares J, Sanchez JM, Hermida R, Tomassini M, Baraglia R, Perego R, and Garnica O (2003) Multi-FPGA systems synthesis by means of evolutionary computation, *GECCO 2003, LNCS 2724*, E. Cantu-Paz et al (Eds.), Springer-Verlag, Berlin, 2109-2120.
- Massa AJ (2002) *Embedded software development with eCos*, Prentice Hall Professional Technical Reference, New Jersey.
- Meyer-Baese U (2001) *Digital signal processing with field programmable gate arrays*, Springer-Verlag Berlin.
- Mumford P, Li Y, Wang J, Rizos C, and Ding W (2006) A time-synchronisation device for tightly coupled GPS/INS integration, *Proceedings of IGNSS Symposium 2006*, Holiday Inn Surfers Paradise, Australia, 17-21 July.
- Nios Community Forum (2005), eCos for Nios II, <http://www.niosforum.com/>.