Performance of a low-cost field re-configurable real-time GPS/INS integrated system in urban navigation

Yong Li, Peter Mumford, and Chris Rizos
University of New South Wales, Sydney 2052, Australia

***Abstract*- This paper describes a low-cost real-time GPS/INS integrated system based on a field programmable gate array (FPGA) platform and its performance in an urban environment. The system is built on the Altera's Nios II soft-core processor. The embedded software is implemented on the Embedded Configurable Operating System (eCos). The FPGA approach can shorten the research and development (R&D) period. Its reprogrammable hardware configuration represents 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.**

Many tests have been conducted in various environments, including suburban, a car parking building, highways, country roads, forested mountain area, in tunnels, under bridges, and in urban canyons. The results have demonstrated the functionality of the system including, the stability and accuracy of the time synchronisation mechanism, the performance of the hardware and software architecture, as well as the accuracy of the algorithm. This paper presents the results of the test in Sydney city in real traffic conditions. Severe GPS outages frequently occurred during the test. The results show that the system can bridge all the GPS outages reasonably well even in the heart of the city where GPS signals were completed blocked by the high buildings.

I. Introduction

Although Global Navigation Satellite Systems (GNSS) technology is developing quickly, the major disadvantage of GNSS will still exist even when the Russian Glonass system, the European Galileo system, or the Chinese Compass system are 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 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 formers' small size, low power consumption, and low cost [1].

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 multisensor 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 being "soft". A "hardcopy" FPGA can be made after the system has been sufficiently tested to reduce system costs.

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. 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 prefer Verilog, whilst the US east coast and Europe more frequently use VHDL [2]. The authors have used the Altera HDL "AHDL" for most of the design.

In addition to aspects on the hardware, the software, and the algorithm, this paper presents the results of the test in the Sydney city, particularly focusing on its performance during frequent GPS outages.

II. Overall System Decription

The Altera Stratix EP1S10 FPGA development board is used in the prototype system (the 'device'). The real-time system is built around the Nios II soft-core processor 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 [3]. The real-time device is currently configured with four UARTs, two of them for INS and GPS input, and another two for the 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 IDE. A special version of eCos – 'the eCos for Nios II' provides support of the FAT32 file I/O for the 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 I.

TABLE I
SUMMARY OF THE REAL-TIME SYSTEM

FPGA	Altera's Stratix EP1S10
Processor	Nios II
System frequency	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 II)

The INS used is the Boeing's C-MIGITS II, which is manufactured using the micromachining technology. It can provide the inertial measurement outputs including delta velocity and delta theta. The inertial sensor assembly consists of six single-axis sensors, three quartz rate sensors (QRS), three vibrating quartz accelerometers (VQA), the drive electronics, preamplifier circuitry for the sensor outputs, and the digital conversion electronics. The main specifications of the C-MIGITS II are listed in Table II. 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.

Table II
ERROR SOURCES OF C-MIGITS II [4]

Coefficient	Gyro (1σ)	Accelerometer (1σ)
Bias	30 deg/hr	4.0 mg
Bias - in-run stability from turn on	5 deg/hr	500 μg
Scale factor stability	350 ppm	800 ppm
Random walk	0.035 deg/sqrt(hr)	60 μg/sqrt(Hz)
Noise	Rate: 360 deg/h (max); Δθ: 4 μrad	Acceleration: 400 μg (max); Velocity: 0.006 ft/sec
Non-orthogonality	0.5 mrad	0.5 mrad

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.

III. 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 counter stamping on the incoming serial data streams. A Nios II 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.

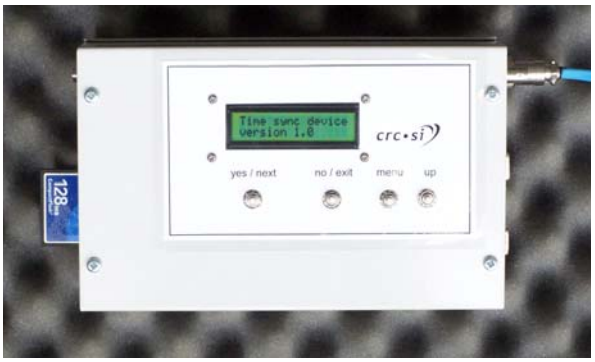

Fig. 1. Hardware of the real-time FPGA-based GPS/INS system

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 Nios II soft-core processor [5]. 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 [6]. Fig. 1 depicts the hardware box of the real-time FPGA-based GPS/INS system.

The hardware design file and the firmware are downloaded 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 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 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.

IV. 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 [7, 8].

The software is designed in a structure consisting of four tasks: the user interface (UI), time synchronisation (TS), strapdown INS (SINS), and Kalman filtering (KF). The low update rate GPS and PPS data are collected in the UI task, and receiving the high update rate IMU data is assigned to TS task. The time-sync procedure is active when the GPS, PPS, and IMU are available. The SINS is followed once the IMU data is time-synced. The KF task runs once the GPS and INS data matches in the time.

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 system first 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 covert 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 affecting 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.

V. ALGORITHMS

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 is taken as a net sum to reduce the real-time computation load. The INS errors are described in a navigation coordinate system (n-frame).

The data processing can be divided into three operation modes: (1) the coarse alignment; (2) the fine alignment; and (3) the integration Kalman filtering. During the coarse alignment the platform remains static while the tilt angles and the sensors noise levels are estimated from the accelerometer data. The C-MIGITS II has a gyro bias of 30deg/hr [4], a magnitude almost twice the earth rotation rate and hence the heading result derived from gyro-compassing is not very meaningful. 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. It quickly converges to the correct heading with the aid of GPS when the platform starts to move.

A. Strapdown INS

Strapdown INS navigation computation equations in the n-frame are implemented for INS position, velocity and attitude updating:

$$\dot{\mathbf{r}}^{\mathrm{n}} = \mathbf{v}^{\mathrm{n}} - \boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}} \times \mathbf{r}^{\mathrm{n}} \tag{1}$$

$$\dot{\mathbf{v}}^{n} = -\left(2\boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{en}^{n}\right) \times \mathbf{v}^{n} + \mathbf{f}^{n} + \mathbf{g}^{n} \tag{2}$$

$$\dot{\mathbf{C}}_{n}^{b} = -\Omega\left(\boldsymbol{\omega}_{nb}^{b}\right)\mathbf{C}_{n}^{b} \tag{3}$$

where $\mathbf{C}_{\mathrm{n}}^{\mathrm{b}}$ is the direction cosine matrix (DCM) which transfers a vector from the n-frame to the body-frame coordinate system (b-frame). The attitude updating can be also implemented through the quaternion:

$$\dot{\mathbf{q}} = -\frac{1}{2}\mathbf{Q}(\boldsymbol{\omega}_{nb}^{b})\mathbf{q} \tag{4}$$

where the 4 by 4 matrix $\mathbf{Q}$ is:

$$\mathbf{Q}(\boldsymbol{\omega}_{nb}^{b}) = \begin{bmatrix} 0 & \boldsymbol{\omega}_{nb}^{b} \cdot \\ \boldsymbol{\omega}_{nb}^{b} & \boldsymbol{\omega}_{nb}^{b} \times \end{bmatrix} \tag{5}$$

Both approaches achieve the same solution and are implemented in the software.

B. INS error model

The commonly used psi-angle model is used as the INS error model in the integration Kalman filter [9]:

$$\delta\dot{\mathbf{r}}^{\mathrm{n}} = -\omega_{\mathrm{en}}^{\mathrm{n}} \times \delta\mathbf{r}^{\mathrm{n}} + \delta\mathbf{v}^{\mathrm{n}} \tag{6}$$

$$\delta\dot{\mathbf{v}}^n = -(\omega_{ie}^n + \omega_{in}^n)\times\delta\mathbf{v}^n + \mathbf{f}^n\times\psi + \nabla^n + \Delta\mathbf{g}^n \quad (7)$$

$$\dot{\psi} = -\omega_{in}^n\times\psi + \boldsymbol{\varepsilon}^n \quad (8)$$

where $\delta\mathbf{r}$, $\delta\mathbf{v}$, and ψ are the error vectors of position, velocity and angle respectively. ∇ and $\boldsymbol{\varepsilon}$ are errors of the accelerometers and gyroscopes respectively. $\Delta\mathbf{g}$ is the error of the calculated gravity.

C. Integration Kalman filter

The system equation of the integration Kalman filter can be written as:

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{w} \quad (9)$$

where the state vector is:

$$\mathbf{x} = \begin{bmatrix}\delta\mathbf{r}^n & \delta\mathbf{v}^n & \psi & \nabla^n & \boldsymbol{\varepsilon}^n\end{bmatrix}^T \quad (10)$$

The matrix F is:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{pp} & \mathbf{F}_{pv} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{F}_{vp} & \mathbf{F}_{vv} & \mathbf{F}_{v\psi} & \mathbf{C}_b^n & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{\psi\psi} & \mathbf{0} & \mathbf{C}_b^n \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (11)$$

The sub-block matrices can be derived from the psi-angle model through Eqs (6) to (8). The sensor errors are modelled as first-order Gauss-Markov processes. The sub-block $\mathbf{F}_{vp}$ comes from $\Delta\mathbf{g}$.

The measurement equation of the integration Kalman filter can be written as:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \boldsymbol{\xi} \quad (12)$$

For loose integration using position and velocity, the measurement matrix $\mathbf{H}$ can be written as:

$$\mathbf{H} = \begin{bmatrix}\mathbf{I}_p & \mathbf{I}_v & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix} \quad (13)$$

where $\mathbf{I}$ is the 3 by 3 identity matrix, and $\mathbf{0}$ is the 3 by 3 zero matrix.

VI. TESTS

In order to evaluate the functionality of the system, in-house and on-the-road tests have been conducted. The tests include the stability and accuracy of the time synchronisation mechanism, the performance of the hardware and software architecture, as well as the accuracy of the algorithm.

The focus of tests in the laboratory include testing the stability of the multi-task firmware, time synchronisation, multiple stages of circular buffering, float-pointing calculation, stability of the Kalman filter, button interrupts and responses, CF file I/O, data output through additional UARTs, and interfacing with Google Earth.

All the functionalities above have demonstrated their workability and stability through the whole system was working properly and stably in the long-term tests. With the 50MHz system design, the timing resolution of the counter is 5.12μs. 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 10μs/sec clock drift, as analysed in [10].

The integrated system has demonstrated its functionality and the accuracy of the algorithm in many on-the-road tests in a variety of environments, including suburban, parking building, a race track with significant attitude manoeuvres, a highway, country roads, and forested mountain areas with GPS signal blockage, as well as through tunnels. The results of the tests have also demonstrated the stability of the integrated solution and the ability to capture the attitude manoeuvres as well as the capability of bridging the GPS outages. [11,12].

The performance in urban canyon is a big challenge for a GPS/INS integrated system. An urban canyon test was carried out in October 2007.

For the test the GPS/INS system was installed on a car. The test started at Miller Street in North Sydney, travelling across the Sydney Harbour Bridge, through George Street – in the heart of the city, and ended at the UNSW campus. The test period length was 46 minutes. Fig. 2 shows the trajectory on Google Earth.

Fig. 2. Trajectory shown on the Google Earth

The GPS signals were blocked out in areas such as avenues heavily masked by trees, the Harbour Bridge, and narrow roads between high buildings in the heart of the city. Figs. 3a-d show typical environments suffering from GPS signal blockage.

Fig. 3a. Avenue – Miller Street

Fig. 3b. Sydney Harbour Bridge

Fig. 3c. Sydney city - George Street 1

Fig. 3d. Sydney city - George Street 2

The OmniStar GPS receiver works in 2D or 3D, automatically changing mode to provide position and velocity data for the integration Kalman filter. It operates in the 3D mode when there are more than three visible GPS satellites, and automatically changes to the 2D mode when the satellites fall to three. There is no output from GPS when the number of visible satellites is less than three, and a GPS outage occurs. Fig. 4 depicts the number of the visible satellites over time.

In addition to the number of visible satellites, the GPS geometry also becomes worse in urban areas, which implies larger values of PDOP. Fig. 5 depicts the PDOP value during the test against time.

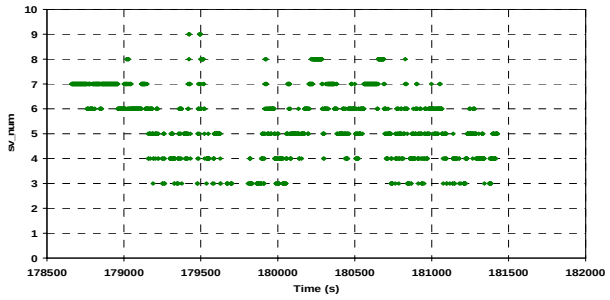

Fig. 4. Number of GPS visible satellites

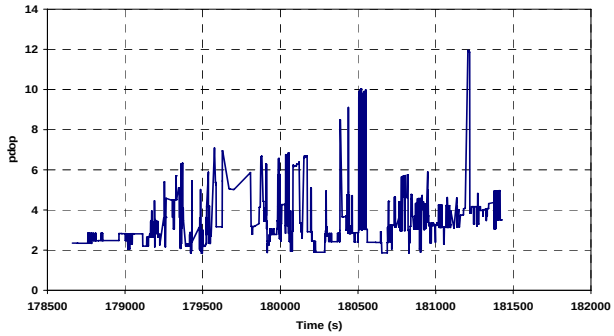

Fig. 5. PDOP vs. time

During the period of the test there were 37 instances of GPS outages of less than 10s as listed in Table III, and 21 outages of over 10s as listed in Table IV.

Fig. 6 depicts the GPS outages against time, in which the outages over 25s are marked by names of places where the outages happened. Apart from a 33s outage on the UNSW campus, the others occurred in the city area.

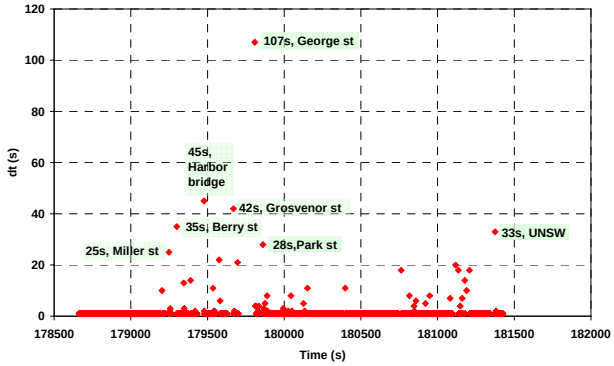

Fig. 6. GPS outages vs. time

TABLE III
LIST OF GPS OUTAGES (<10S)

Outages (s)	Times
2	17
3	4
4	5
5	3
6	2
7	2
8	4
9	0

TABLE IV
LIST OF GPS OUTAGES (≥10S)

Outage (s)	Start time (s)	End time (s)	place	Distance (m)
10	179193	179203	Miller St	62
10	181180	181190	UNSW	43
11	179526	179536	Bradfield HWY	67
11	180141	180152	Oxford St	4
11	180387	180398	Oxford St	46
13	179332	179345	Berry St	146
14	179376	179390	Bradfield HWY	277
14	181164	181178	UNSW	95
18	180745	180763	Lang Rd	265
18	181118	181136	UNSW	45
18	181190	181208	UNSW	27
20	181098	181118	High St	166
21	179675	179696	George St	6
22	179554	179576	Grosvenor St	144
25	179224	179249	Miller St	208
28	179834	179862	Park St	57
33	181330	181363	UNSW	47
35	179265	179300	Berry St	180
42	179628	179670	Grosvenor St	183
45	179432	179477	Harbour Bridge	836
107	179701	179808	George St	781

With these outages, the ground trajectory from the GPS is depicted in Fig. 7, from which it is easy to find the severe outages cause difficulty in estimating the actual path, especially in the city area.

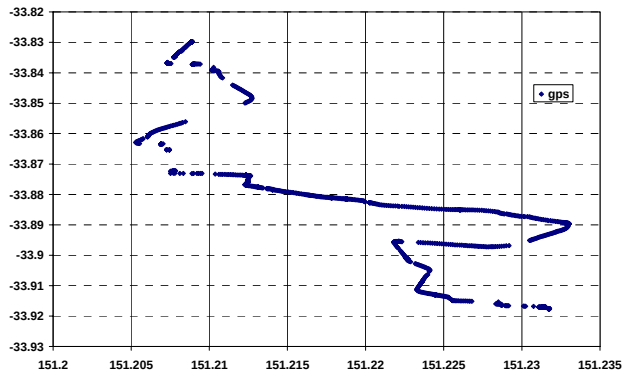

Fig. 7. Ground trajectory from GPS

Although these GPS outages are serious, the GPS/INS solution has successfully bridged them as depicted in Fig. 8. The GPS trajectory is also depicted in Fig. 8, which shows the solutions derived from different numbers of satellites in different colours - yellow for three satellites, green for four and blue for more than four. It can be observed that the GPS visibility is very poor just before and after an outage – only 3 or 4 satellites are available. Moreover, the solutions at these epochs were most likely affected by severe multipath and therefore they have a poor accuracy and cannot be expected to be able to correct the INS solution due to the drift after the outage.

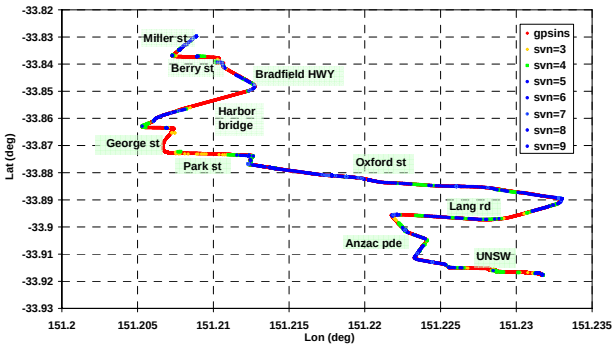

Fig. 8. Ground trajectory from GPS/INS

Fig. 8 confirms that the severe outages happened in the city area – Berry Street, Harbour Bridge, Grosvenor Street, George Street, and Park Street. For a clear illustration of how the GPS/INS solution bridged these outages, zoom-in trajectories in these areas are depicted in Figs. 9-11.

In Fig. 9, there is a 25s outage on Miller Street. The car turned left into Berry Street and the GPS receiver lost all satellites during the turn, which caused a 35s GPS outage on Berry Street. Shortly later, a 13s outage occurred during the right turn from Berry Street to Bradfield Highway. Fig. 9 clearly shows that these GPS outages have been smoothly bridged by the GPS/INS solution, even in two turns.

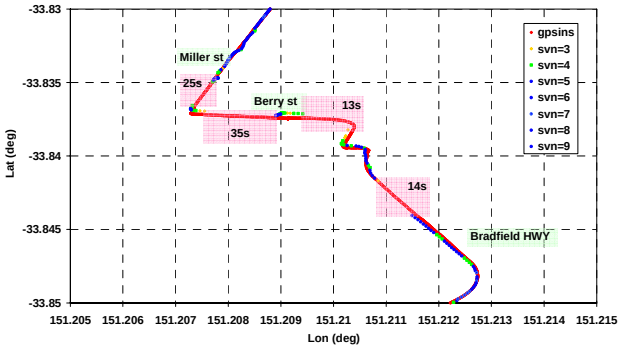

Fig. 9. Ground trajectory from GPS/INS on Berry Street

It is surprising that the GPS receiver completely lost its solution on the Sydney Harbour Bridge as shown in Fig. 10. Although the bridge has a steel structure, there is a open-sky view on the bridge as shown in Fig. 3b. A possible reason is that the severe multipath caused by the steel frames of the

bridge caused the GPS receive to lose track of the signals or the solution was of too poor a quality to use. There is a 45s GPS outage cross the bridge over a distance of 836m as shown in Table IV. Fig. 10 shows that the GPS/INS solution smoothly bridges this outage. It is obvious that the GPS/INS solution does not suddenly jump to the GPS solution after the outage, instead it gradually comes along with the GPS solution when the GPS solution becomes reliable with more than four satellites.

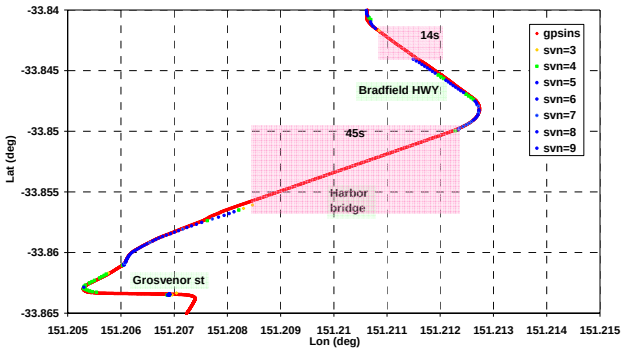

Fig. 10. Ground trajectory from GPS/INS on Harbour Bridge

The worst case is the outage on George Street – in the heart of the city. From Fig. 11, there is a series of outages - a 22s outage on Grosvenor Street first, then some GPS data available for a short period at a traffic light, and the GPS quickly lost again to make a 42s outage including the right turn from Grosvenor Street into George Street, then a 2D position available but followed by a 21s outage at once, a 2D position available again just before the succeeding 107s outage, several 2D positions available after the left turn from George Street into Park Street, but a 28s outage followed immediately. It is evident that only 2D positions (yellow points) or complete outages (red points) occur during most of time on Park Street. All of these poor situations together provide a serious challenge to the GPS/INS system.

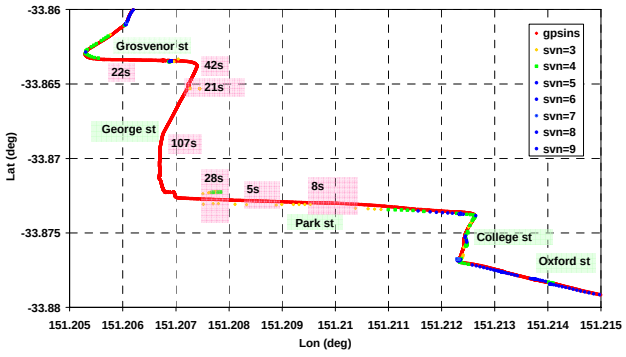

Fig. 11. Ground trajectory from GPS/INS on George Street

From Fig. 11 it is evident that the GPS/INS solution has successfully bridged all of these outages. In the 42s outage from Grosvenor Street to George Street, the GPS/INS solution has smoothly tracked the big right turn. During the 21s outage on George Street, just between the two 2D positions, the traffic was very heavy and the car moved forward about 6m within 21s, see Table IV. Even in such a serious situation, the GPS/INS solution still keeps it accuracy and no obvious drift is observed.

The traffic was still heavy during the left travel on George Street. From Table IV, the travel distance of 781m took 107s. The last section of this outage is a left turn into Park Street. Fig. 11 clearly shows that the GPS/INS maintains its accuracy to track the turn even when experiencing such a long GPS outage.

VII. Conclusions

An FPGA-based real-time GPS/INS integrated system has been developed. 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 to time-synchronise the GPS and INS data streams.

The embedded software implements multiple tasks; decoding the GPS and INS data streams, time synchronisation, strapdown inertial computation, and the integration Kalman filtering. With eCos support, the software supports 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.

Tests have been conducted in various environments. The results demonstrate the functionality of the system including, the stability and accuracy of the time synchronisation mechanism, the performance of the hardware and software, as well as the accuracy of the algorithm.

A test in Sydney city in traffic has been conducted. Severe GPS outages were frequently suffered in the test. The results show that the integrated solution of the system can bridge all the GPS outages reasonably well even in the city centre where GPS signals were completely blocked by the high buildings. It demonstrates the capability of the system for a variety of land applications.

References

[1] D. Liccardo and J. Rios, (2006)"Robust sensor fusion for GPS inertial measurement units in diverse flight environments", *GPS World*, pp. 33-39, July 2006.

[2] U. Meyer-Baese, *Digital signal processing with field programmable gate arrays*, Springer-Verlag Berlin, 2001.

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

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

[5] P. Mumford, 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, 17-21July 2006.

[6] Altera (2005) *Nios II software developer's handbook*, http://www.altera.com.

[7] Nios Community Forum (2005), eCos for Nios II, http://www.niosforum.com/.

[8] Massa AJ (2002) *Embedded software development with eCos*, Prentice Hall Professional Technical Reference, New Jersey.

[9] I. Y. Bar-Itzhack, "Navigation Computation in Terrestrial Strapdown Inertial Navigation System", *IEEE Trans. On AES*, vol. AES-13, no. 6, pp. 679-689, 1977.

[10] Y. Li, P. Mumford, J. Wang, and C. Rizos, "Development of a GPS/INS integrated system on the field programmable gate array platform", *Proceedings of ION GNSS 2006*, Forth Worth, Texas, pp. 2222-2231, 26-30 September 2006.

[11] Y. Li, P. Mumford, J. Wang, and C. Rizos, "A low-cost field re-configurable real-time GPS/INS integrated system – design and implementation", *Proceedings of IGNSS 2007*, Sydney, Australia, , paper #42, 4-6 December 2007.

[12] Y. Li, P. Mumford, and C. Rizos, "A low-cost real-time GPS/INS integrated system with hardware and software field re-configurability", *Coordinates,* vol. IV, no. 3, pp. 12-17, March 2008.