Portable GPS Baseband Logging

Morgan Quigley, *Stanford University*
Pieter Abbeel, *Stanford University*
Dave De Lorenzo, *Stanford University*
Yi Gu, *Stanford University*
Sara Bolouki, *Stanford University*
Dennis Akos, *University of Colorado*
Andrew Y. Ng, *Stanford University*

BIOGRAPHY

Morgan Quigley and Pieter Abbeel are Ph.D. candidates in the Computer Science Department at Stanford University.

Yi Gu is a Ph.D. candidate in the Electrical Engineering Department at Stanford University.

Sara Bolouki is a graduate student in the Electrical Engineering Department at Stanford University.

Dr. Dave S. De Lorenzo is a Research Associate in the GPS Laboratory at Stanford University, where he is studying space-time adaptive antenna array processing and GNSS software receivers.

Dr. Andrew Y. Ng is an Assistant Professor in the Computer Science Department at Stanford University. His research interests include machine learning, robotic perception and control, and broad-competence artificial intelligence.

Dr. Dennis Akos is an Assistant Professor in the Aerospace Engineering Science Department at the University of Colorado at Boulder.

ABSTRACT

We present the design and implementation of a highly portable multi-antenna datalogging system which can log several minutes of multi-channel raw GPS L1 baseband. In addition, our design interleaves two serial data streams with the baseband data, allowing, e.g., inertial data to remain synchronized with the data stream. The system is FPGA-based and uses two CompactFlash cards for storage. The data is extracted by imaging the CompactFlash cards onto a PC and recovering synchronization codes interleaved with the baseband data. The resulting system is useful for multi- (and single-) antenna receiver development, as it allows for simple data collection. The data files can then be used as inputs to software receivers for algorithm development, testing, and quantitative comparisons. Because the digital logging section is cleanly separated from the RF front-end section, it is possible to substitute any manner of RF front-end (e.g. Galileo) in place of our GPS L1 section, so long as the total data rate stays within the 250 megabit/sec capacity of the logging section.

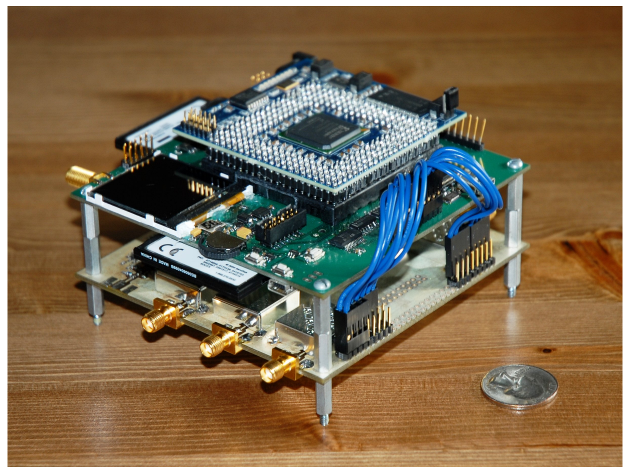

Figure 1. Multi-channel GPS L1 data collection system

1 INTRODUCTION

Traditional GNSS receivers use a bank of special-purpose logic to correlate incoming (downconverted) data samples against locally generated copies of these signals. This correlation is used to steer the code and carrier tracking loops, which in turn produce estimates of code and carrier phase shifts that are used by higher-level navigation processing. In typical real-time implementations, once an incoming sample has been integrated in the correlations, it is discarded on the next clock cycle. This architecture is well suited for a wide variety of applications, as it can be used to design the low-power receiver ASICs that have contributed to the dramatic drop in GPS receiver prices in recent years. However, this architecture does not lend itself well to receiver design and signal analysis, as the raw RF baseband is not available for experimentation.

In this paper, we describe the design and implementation of a portable device, shown in Figure 1, which is well suited to receiver research and algorithm development. This device captures every data sample from a 6-antenna array and saves it to a flash-based storage. The saved samples can be later converted to PC files for receiver algorithm development on real-world data.

1.1 Motivation

In the Stanford Computer Science Department, ongoing research has produced advances in the state of the art of aerobatic autonomous helicopter flight [1]. In prior work, we have presented applications of statistical machine learning techniques to aerobatic R/C helicopter system identification (model building) and controller generation. We have used these techniques to successfully derive controllers capable of autonomous inverted hovering, rolls, flips, funnels, and a variety of other maneuvers.

However, as we have continued to push our airframes and algorithms into ever more agressive maneuvers, we have found that obtaining accurate (centimeter-level) localization is rather difficult. This is particularly true in aerobatic maneuvers that require the helicopter to repeatedly transition through inverted orientations. In our experience, single-antenna GPS receivers, even very good ones, cannot rapidly re-acquire carrier lock in such situations. As a consequence, state estimation would have to rely solely on inertial integration, which in our experiments is MEMS-based and thus state estimation degrades rapidly.

Therefore, we have begun to investigate the feasibility of constructing a multi-antenna GPS receiver which provides omnidirectional reception via antenna switching. As a first step in this undertaking, we describe the construction of a multi-antenna datalogging device which is capable of operating onboard our helicopter platform and collecting the raw GPS baseband streams emitted by a collection of RF front-end chips.

1.2 Application areas

A portable, robust, easy-to-operate baseband data collection system is of great utility in designing GPS receiver algorithms. By capturing the raw output of RF front-end chips, multi-antenna receiver algorithm development can happen offline in software. The datalogging device is particularly useful if experiments, such as helicopter flights, have practical difficulties or are challenging to simulate. Other applications, such as bistatic radar, could also benefit from offline processing of data.

High-fidelity single-antenna simulators have existed for a number of years in the GNSS community. However, obtaining real-world multi-antenna data helps increase the likelihood that receiver algorithms designed in software will work in a wide variety of real-world situations, as some multi-antenna experiments can be difficult to model.

2 ENVIRONMENTAL CONSTRAINTS

The R/C helicopter application imposed numerous restrictions on our design. Because these restrictions were the deciding factor in several design choices, we will provide some details of the R/C helicopter environment.

Figure 2. Aerobatic RC Helicopter

Our helicopters (see Figure 2) run powerful single-cylinder engines. The airframes and small and light (1 meter and 5 kilos, respectively). As a direct consequence, the airframes undergo significant vibrations. Furthermore, the excellent thrust-to-weight ratio allows the helicopters to undergo accelerations in the 3G range (along the axis of the main rotor). Aerobatic tuning of the flight controls allows for rotational rates of 180 degrees per second in the pitch and roll axes and 360 degrees per second in the yaw axis. The dynamic nature of the platform means that any datalogging device must be quite mechanically robust.

As with any aerial platform, the flight weight of the system is severely constrained; heavier payloads reduce the aerobatic capabilities of the vehicle. Furthermore, on such small helicopters, payload size is limited to a few cubic inches, as larger payloads could shift the center of gravity far from its nominal position along the axis of the main rotor, resulting in an airframe that is more difficult to control.

3 DATA STORAGE

There are several options for storing the baseband data. The following sections describe the various options and their respective feasability for the R/C helicopter application.

3.1 DRAM

A large DRAM module could serve as a data store: the hardware could fill the DRAM with the sample data, and later dump this data to a PC for permanent storage. This option has the advantage of being relatively simple, but it is limited by the size of DRAM modules that are readily obtainable (at time of writing, modules larger than 4 GB are rather expensive). Furthermore, a loss of power before the DRAM module has been copied out of the datalogger will result in the loss of all data.

3.2 Mechanical hard drive

Mechanical hard drives have been successfully used in the past [2], [3] to create multi-antenna dataloggers. Because laptop hard drives are readily obtainable, this solution would be inexpensive and have a very large capacity. However, we chose not to pursue this concept due to the severe vibrations and accelerations our platform experiences in normal operation. Because the vibration of our platform is approximately double the vibration rating of most consumer hard drives, we feared that head crashes would destroy a hard drive-based datalogger.

3.3 Solid-state hard drives

Solid state hard drives are gaining momentum in the computer industry as replacements for mechanical hard drives. These drives can use the same ATAPI protocol and form factor as mechanical drives, but they implement data storage using flash memory instead of rotating platters. Although these drives are dropping in price, a large solid-state drive with high throughput costs several thousand dollars at time of writing. This option was not pursued simply because of cost; as prices continue to fall, solid-state drives may become a viable option for portable datalogging.

3.4 USB flash drive

USB flash drives have previously been used to record dual-antenna data [4]. In that work, the logging system was a small form factor (nano-ITX) PC with two USB flash drives for storage. The resultant system was small enough to fly on a balloon for successful experiments. However, as we were seeking to reduce weight and power even further in our logger design, we chose not to follow the embedded PC approach in this work, opting instead for a fully FPGA-based system. Since we did not have a host PC, USB flash drives were not an option in our design.

3.5 Removable Flash Media

There are a variety of removable flash media formats currently used in mass-market digital cameras. Flash cards of each of these formats are readily obtainable. However, access to the technical specifications of many formats require significant licensing fees.

For this project, we chose to use the CompactFlash standard. Its specification is freely available, and the fastest CompactFlash cards currently provide the highest sustained throughput among the various removable flash standards.

CompactFlash cards can run in several operational modes. We chose the “True IDE” mode, which means the card is operated by the parallel ATA (PATA) protocol. Various versions of this protocol have been in use for decades. Thus, documentation and open-source software and hardware implementations are widely available.

4 WRITE STALLS

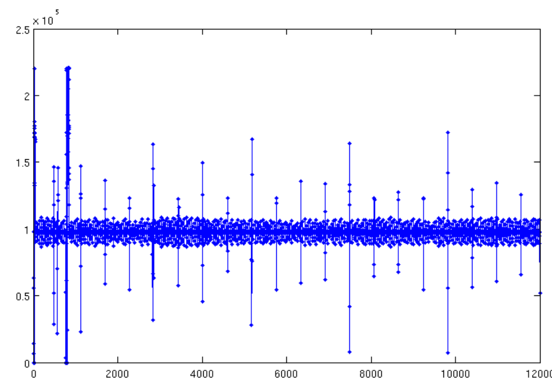

Figure 3. Write speed (bits per second) over time. This plot shows the empirical write throughput during 10 seconds of continuous write requests at a nominal rate of 100 megabits per second.

Like many storage devices, the write speed of CompactFlash cards is not constant over time. As shown in Figure 3, the cards occasionally (especially at startup) stall in the write cycle, resulting in several milliseconds of zero throughput. Since the goal of the GPS baseband logger is to capture the raw output of the RF front-end chips, such gaps in the data are unacceptable. Therefore, we spent significant efforts in designing a buffering scheme which can absorb such write stalls without dropping data samples.

5 FRONT-END BOARD

Our front-end board is based around the SiGe SE4110L. The RF signal from a GPS L1 antenna is fed into the chip, which then outputs a downconverted and digitized version as a 2-bit digital stream at 16.368 MHz.

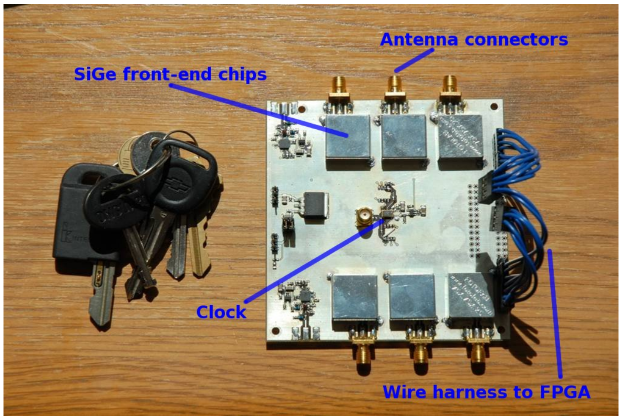

Figure 4. Front-end board

Our prototype front-end board has 8 of these chips, as shown in Figure 4. However, for the experiments described in this paper, only 6 of these chips were used due to limitations on the sustainable CompactFlash write bandwidth. All 6 chips are driven from the same clock reference. Because the SE4110L directly derives its sample clock from the input clock on this frequency plan, the output digital stream is always in phase across all 6 chips. Thus, this front-end board can be seen as producing a single 12-bit digital word at 16.368 MHz, which results in approximately 196 megabits per second. These bits are carried through a 14-wire harness to the digital logger section, which will be discussed in detail in the next section.

6 LOGGER IMPLEMENTATION

The digital logging section of our design is constructed around a mid-size Xilinx Spartan-3E FPGA. The FPGA paradigm was chosen because of the high data rates of the system and the desire for portability. Although an all-software solution might be possible using a high-power CPU and clever use of the memory hierarchy, we instead used custom FPGA logic to implement the system in a small and reasonably low-power fashion.

6.1 Block Diagram

A block diagram of the logger is shown in Figure 5. In the diagram, white blocks are synthesized on the FPGA, and yellow blocks indicate off-chip components. Heavy red lines indicate high-bandwidth data flows, and thin black lines indicate low-bandwidth data flows.

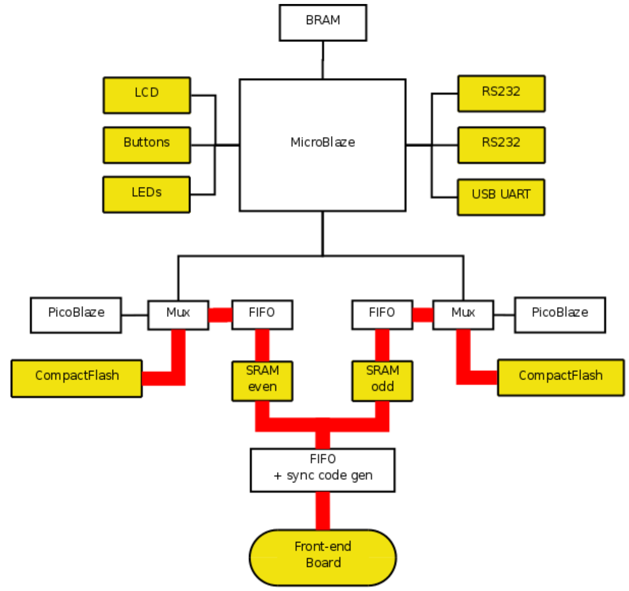

Figure 5. Logger block diagram

The top half of the diagram is the standard picture of FPGA-based embedded systems: a soft-core processor (in this case, the Xilinx MicroBlaze) surrounded by various off-chip I/O peripherals to form a complete computer system.

The bottom half of the diagram, however, was custom-designed for this project. The next few sections will describe this data path in detail.

6.2 Input Stage

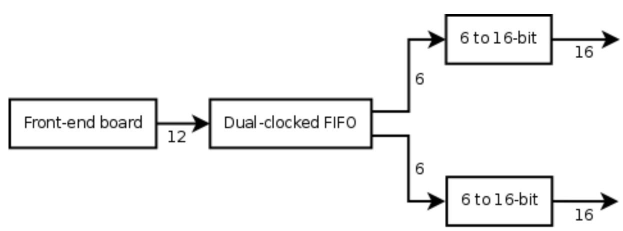

Figure 6. Input stage block diagram

As shown in Figure 6, data words emitted by the front-end board enter the FPGA clock domain through a dual-clocked FIFO. Due to the high data rate of the front-end board, we chose to create two (identical) parallel data paths. Each arm of the data path handles 98 megabit/sec, which is comfortably below the (empirically measured) maximum 128 megabit/sec sustained write speed of CompactFlash cards in "True IDE" programmed I/O (PIO) mode.

After leaving the clock-domain-crossing FIFO, the incoming 12-bit data words are therefore split into two 6-bit data words. The next component in the data path combines these

6-bit data words to form a stream of 16-bit data words. Since the least common multiple of 6 and 16 is 48, the conversion is performed by an 8-state logic block which produces three 16-bit data words for every eight 6-bit inputs.

6.3 Input FIFO, Mux, and Synchronization Codes

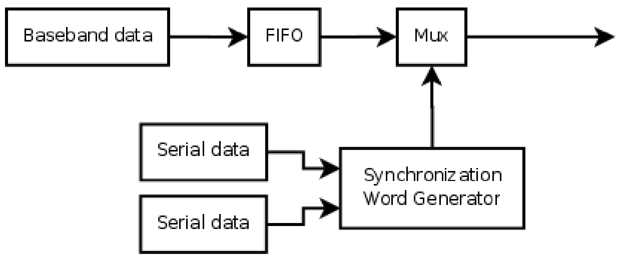

Figure 7. Synchronization stage block diagram

As shown in Figure 7, the 16-bit data words are fed into another FIFO, which is fetched by a component that, every 64 bits, stops the FIFO and inserts a "synchronization word," which consists of an 8-bit counter and two 4-bit shift registers constructed by directly sampling the data lines of two serial receivers (UARTs). This synchronization word thus provides an integrity check on the data as well as allowing two UARTs to be logged alongside the GPS data. Since the UART and baseband data are interleaved, they are well-synchronized in time. In our application, one of the UARTs was driven by an inertial memeasurement unit (IMU), and thus maintaining time synchronization was critical.[1]

After the bit width conversion and synchronization code multiplexing, this stage outputs a stream of 16-bit data words.

6.4 Buffering

The buffering stage, shown in Figure 8 operates on 16-bit data words, and uses off-chip SRAM to provide buffering of 1 million words. This is achieved with a single-port 18-megabit SRAM. In order to provide buffering to both SRAM cards, two circular buffers were implemented on the SRAM: one using the even addresses, and one using the odd addresses. Maintaining read and write addresses for each circular buffer allowed the single SRAM to provide storage for two deep FIFOs.

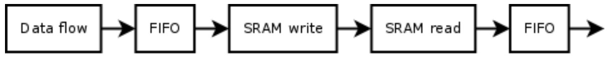

Figure 8. Buffer stage block diagram

To avoid wasting bus cycles while shifting the SRAM between write and read operations, small FIFOs were synthesized on the FPGA to buffer 1000 words in each data path. These FIFOs allow the SRAM to perform large numbers of sequential writes and reads, rather than change the bus direction (and incur wait cycles) for each data word.

At a 200 megabit/sec data rate, the 16-megabit buffer is capable of holding 80 milliseconds of data. This was more than sufficient for our experiments. However, we note that the various brands and models of CompactFlash cards have different stall characteristics and sustained write speeds. Our hardware only works with relatively new high-performance cards. Older or lower-quality cards simply do not have the sustained write speed to keep up (and catch up after write stalls) with the data rate of the front-end board.

6.5 Card Control

Our CompactFlash controller combined two levels of software control with hardware logic.

The top tier of control was written in C and runs on the Xilinx MicroBlaze soft-core CPU which is at the heart of the logger. This high-level control layer resets the cards after power-up, queries their sizes, and places them in a known state.

The middle tier of control was written in assembly for a small 8-bit soft-core processor, the Xilinx PicoBlaze. Because these processors are small and easy to instantiate, we synthesized one for each CompactFlash card. Once the high-speed datapath is started, these processors attempt to keep the card as busy as possible: they initiate the sector writes and poll the cards until they are ready to receive data.

The lowest level of control was written in Verilog. This layer pulls data from the SRAM buffer and, using the CompactFlash PIO Mode 6 timings, shuttles data to the cards. At the end of each sector, this control layer relinquishes control of the card to the middle tier of control.

6.6 User Interface

The user interface of the datalogger is quite simple. A few buttons provide for reset, "start logging," and "stop logging" functionality. A small LCD display shows the number of bytes written as logging progresses. For debugging information, a USB-UART chip is provided, allowing the datalogger to provide a console (via a virtual COM port) when plugged into a host computer's USB bus.

[1]The other UART was driven by a u-blox GPS receiver.

6.7 Runtime

Using two 8-gigabyte flash cards, the system can store approximately 10 minutes of data using the front-end board described previously, which emits data at a rate of 200 megabit/sec. Naturally, this recording time would double if 16-gigabyte flash cards were used, and even higher card densities will likely emerge in the future.

A simple 3.3-volt switching power supply was implemented, allowing the datalogger to be directly driven from any DC power source in the 5-15 volt range. For our experiments, we powered the system from 3-cell lithium-polymer batteries. The nominal power draw of the system was 2 watts. The optional LCD backlight drew another 1 watt when powered on. This means that a 3-cell lithium-polymer battery will power the system for approximately two hours, although this runtime is somewhat irrelevant because the logging capacity of the data store is an order of magnitude less.

6.8 Physical Layout

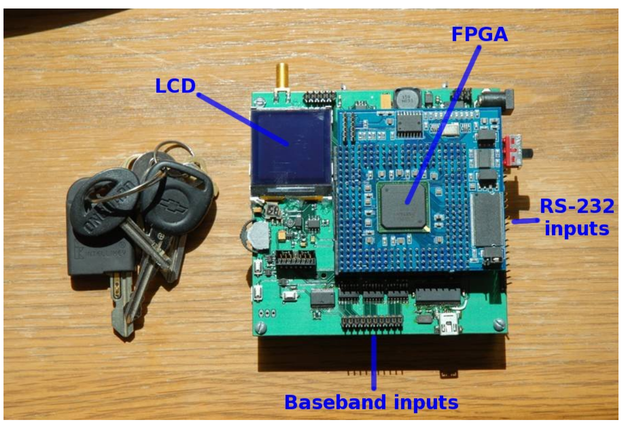

Figure 9. Digital logger board

As shown in Figure 9, the FPGA is carried on a separate circuit board which plugs into a pin-grid array (PGA) socket on a custom-designed carrier board. The FPGA module is commercially available and integrates the FPGA (and its difficult-to-prototype BGA packaging), a PROM to configure the FPGA, and 16 megabytes of DRAM.[2] The carrier board provides two megabytes of SRAM, two CompactFlash sockets, a small LCD display, two RS-232 ports, a few buttons, and a power supply.

Additionally, the carrier board has a u-blox LEA-4S module which is used as a "sanity check" on the data stream. Its NMEA output is interleaved with the GPS baseband data, as discussed previously, to provide a baseline for checking all post-processed navigation solutions.

[2]The Darnaw1 FPGA module is available from Enterpoint Ltd.

6.9 Flexibility

Because the digital logging section is separated from the RF front-end board, it is possible to substitute any RF front-end in the place of our 6-channel GPS L1 board. So long as the data rate stays below 250 megabit/sec, the digital side of the system will operate in the same fashion. For example, one could use different front-end chips to log fewer channels with more bits of resolution, or log another frequency band such as Galileo or COMPASS, or some mixture of frequencies. The digital side of the system simply shuttles bits around and stores them.

7 DATA EXTRACTION

The logger hardware does not attempt to set up a file system on the CompactFlash cards. Rather, it simply treats the card as a (massive) linear array. To extract usable data, the cards are first imaged into a PC's file system. Then, subsequent processing stages decode the disk image and recover the GPS baseband data and the interleaved serial data streams.

7.1 Baseband Data Recovery

To recover the baseband data, the card images are scanned to find the offset of the synchronization word. This offset is easily found, as the synchronization word contains an 8-bit counter which is inserted into the data stream every 128 bytes. By reading the first few sectors of the card image and trying every possible offset, the true offset is readily obtained.

As previously discussed, the 6-bit data words fed to each arm of the data path are combined in a state machine to form streams of 16-bit words. To recover the true starting phase offset out of the 3 possibilities, the average activation of each bit is observed using each of the 3 possibilities. Since the SE4110L chip has automatic gain and bias control, the sign bit should be active 1/2 of the time and the magnitude bit should be active 1/3 of the time. To resolve the ambiguity between antennas, we intentionally reverse the sign and magnitude bits for front-ends 1 and 4. This means that the correct data phase will result in the following bit activation rate:

1/3	1/2	1/2	1/3	1/2	1/3

After decoding the first few sectors of data with each of the possible initial data phases, the correct data phase is readily apparant.

After the initial synchronization word and data phase offsets are found, the data decoding can begin. We decode the sign and magnitude bits to produce binary files which use a single character per data sample. Such files are then easily imported into MATLAB or read by other languages.

7.2 Serial data recovery

To recover the two serial streams, their shift registers are first extracted from the synchronization words. Then, in software, we perform exactly what a UART does in hardware: wait for the start bit, then (using the previously known baud rate) sample the data bits, and finally ensure that a valid stop bit is present. Our software then writes this data stream to a binary file for further processing.

In our experiments, one serial stream was the ASCII NMEA data from a GPS receiver module, and the other stream was 300-Hertz inertial data (3-axis accelerometers, angular rate sensors, and magnetometers). While the NMEA data could be read directly with a text editor, the inertial data required another decoding stage to produce a text file suitable for importing into MATLAB.

8 EXPERIMENTS

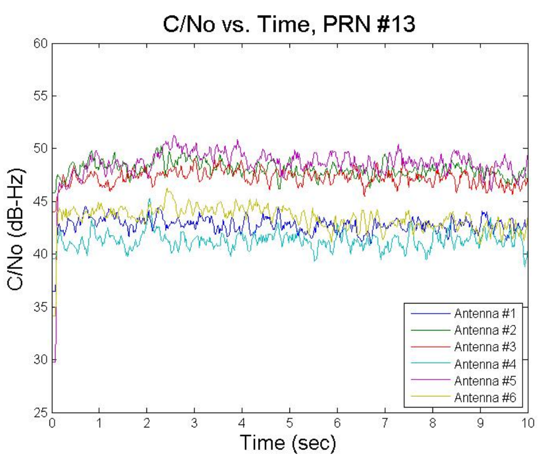

Figure 10. C/No obtained by tracking each of the 6 channels separately. The variations in C/No are perhaps due to antenna placement or physical front-end board layout.

To verify the functionality of the logger, we attached six GPS L1 antennas to the roof of a vehicle and drove for several minutes with the logger running. The resulting data files were first treated as separate data logs and tracked independently. As shown in Figure 10, the antennas have similar noise figures. We are uncertain if the differences in Figure 10 are due to antenna placement on the roof of the vehicle or to the physical layout of the front-end board. Regardless, there is a marked difference, as shown in Figure 11, when the signals are combined. This figure shows the results of tracking the data log using a multi-antenna receiver algorithm [5]. The results verify the functionality of both the datalogger and the tracking algorithm as the signals are adaptively combined to yield a significantly higher C/No ratio.

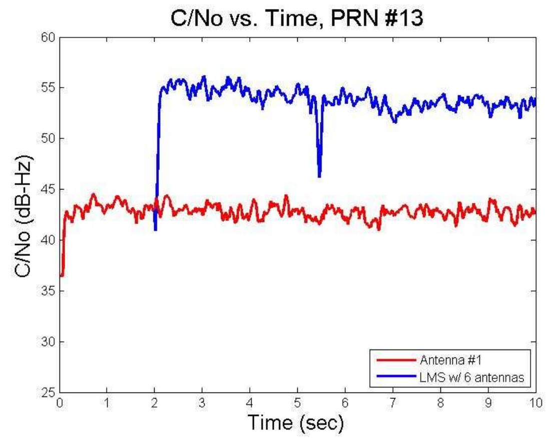

Figure 11. C/No obtained by an adaptive antenna-array processing algorithm. Red shows tracking using only one antenna, and, starting at t=2 seconds, blue shows tracking with all 6 antennas combined adaptively.

Although not shown in this paper, the IMU and reference NMEA outputs were also reasonable. Therefore, we conclude that the datalogging device is successfully storing multi-channel GPS baseband data, interleaved with two serial data streams.

9 CONCLUSIONS

We have presented the design and implementation of a portable data-collection device which can capture data streams of up to 250 megabit/sec. For our own experiments, we constructed a 6-channel GPS L1 front-end board which produces data at 200 megabit/sec. We demonstrated the functionality of the device by logging 6 GPS L1 antennas and an inertial data stream on a moving vehicle's roof and post-processing these streams both independently and using multi-antenna techniques.

REFERENCES

[1] Pieter Abbeel and Adam Coates and Morgan Quigley and Andrew Y. Ng, "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," *Proceedings of Neural Information Processing Systems (NIPS), Volume 19, 2007.*

[2] C. Fernandez Prades, A. Ramirez Gonzalez, P. Closas Gomez, and J.A. Fernadez Rubio, "Antenna Array Receiver for GNSS," *Proc. European Navigation Conference, 2004.*

[3] Staffan Backen and Dennis Akos, "GNSS Antenna Arrays - Hardware Requirements for Algorithm Imple-

mentation," *Royal Institute of Navigation European Navigation Conference & Exhibition (ENC), Manchester, UK, May 2006.*

[4] Stephan Esterhuizen and Dennis Akos, "The Design, Construction, and Testing of a Modular GPS Bistatic Radar Software Receiver for Small Platforms," *GNSSR '06: Workshop on GNSS Reflections, ESTEC, Noordwijk, Netherlands, June 2006.*

[5] David S. De Lorenzo, "Navigation Accuracy and Interference Rejection for GPS Adaptive Antenna Arrays," *PhD Thesis, Department of Aeronautics and Astronautics, Stanford University, 2007.*