ADVANCES IN TELEMETRY CAPABILITY AS DEMONSTRATED ON AN AFFORDABLE PRECISION MORTAR

Michael L. Don
U.S. Army Research Laboratory
Aberdeen Proving Grounds, MD

ABSTRACT

This paper presents three telemetry techniques demonstrated on an affordable precision mortar that allowed the guidance, navigation, and control (GNC) system to be effectively analyzed. The first is a technique for the real-time integration and extraction of GPS data into a sensor telemetry stream. The second is a method for increasing telemetry bandwidth by saving a short period of high rate data and then broadcasting it over the rest of the flight test. Lastly, I present an on-board data storage implementation using a MicroSD card.

KEY WORDS

Telemetry Processing, FPGA, SD card, GPS, Decom.

INTRODUCTION

The U.S. Army Research Laboratory (ARL) is developing a flight controlled mortar (FCMortar) in conjunction with the Naval Surface Warfare Center. This program aims to develop an affordable guidance and control system, packaged into a fuze kit, that screws onto the fuse threads of a standard 81-mm high explosive projectile. Early flight tests verified the design of fin sets and forward section extensions. The rounds were instrumented with an ARL designed diagnostic telemetry module embedded within the body of the projectile (1). This module, named MIDAS (Multifunctional Instrumentation and Data Acquisition System), encodes analog sensor data into a pulse code modulated (PCM) stream and transmits it to a ground station. Although the original PCM encoder design was sufficient for early phases of the program, the introduction of a GPS receiver and GNC subsections in later tests created new telemetry challenges.

This paper presents three of the techniques developed to overcome these challenges. The first is a technique for the real-time integration and extraction of GPS data into a sensor telemetry stream. The second is a method for increasing telemetry bandwidth by saving a short period of high rate data and then broadcasting it over the rest of the flight test. Lastly, I present an on-board data storage implementation using a MicroSD card.

GPS INTEGRATION

Although ARL has extensive experience with munitions telemetry, GPS receivers are a recent addition to the ARL instrumentation suite. The FCMortar project was the first ARL exposure to a GPS receiver produced by Mayflower Communications. Due to the risk associated with integrating a new GPS receiver into a mortar, all of the GPS data was required to be available during flight experiments in real-time as well as recorded for post-processing. A method was needed to integrate the GPS data into the existing sensor telemetry stream and extract it at the ground station. First I will present the original PCM encoder design, and then I will describe the method I developed to manage the new GPS data.

Figure 1 shows a functional block diagram of the MIDAS PCM encoder before the GPS integration logic was added. Analog sensor data is multiplexed into an analog to digital converter (A/D). The FPGA serially shifts the digital data into an input shift register. A commutator inserts the sensor data into the telemetry frames which are saved into an off chip RAM. The RAM acts as a buffer which delays the data about 100 ms before it is read into an output shift register and serially shifted out to the transmitter. This delay allows data sampled within the mortar gun tube to be transmitted later after exiting the tube, preventing the transmission interference of the gun tube from corrupting the in-bore data. Telemetry data was formatted into 48, 16 bit words and transmitted at a rate of 4 Mbit/s.

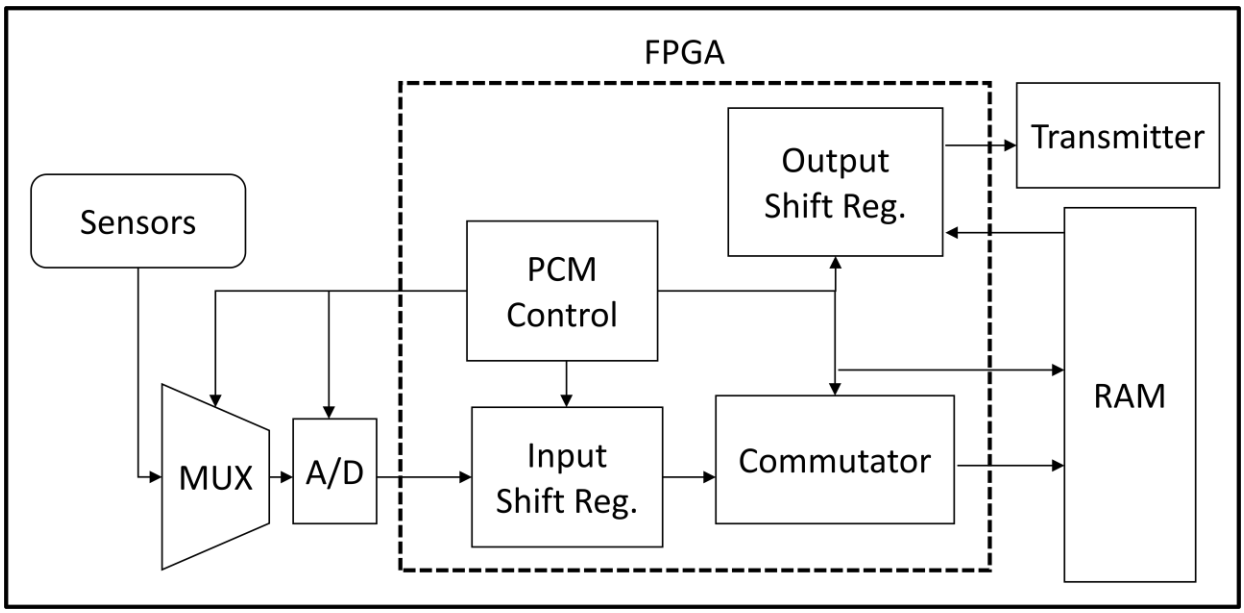

Figure 1: Original MIDAS PCM encoder block diagram

There were two general options for integrating the GPS data into the telemetry frames. One possible method would have been to decode all of the GPS messages in MIDAS and integrate the decoded data values into the telemetry frames. This was unpractical using the current MIDAS FPGA since there were 15 different messages, some of which were 424 bytes long containing 150 different data values. A simpler method was to insert the raw universal asynchronous receiver/transmitter (UART) bytes directly into the telemetry stream. Figure 2 shows a block diagram of the new MIDAS configuration used to integrate the GPS data. The GPS receiver

transmits data messages using a UART at a rate of 115200 baud. I developed a custom UART core to receive the incoming data while minimizing FPGA resources. In order to transfer the GPS UART data from the UART clock domain to the PCM stream clock domain, a first in, first out memory (FIFO) scheme was employed (2). UART words received from the GPS are written to the FIFO using the UART system clock. The PCM control logic then reads this data from the FIFO using the PCM clock and inserts the UART words into the telemetry frames. The 8 bit UART bytes are converted into 16 bit telemetry words by adding zeros to the most significant bits. AAAA hex is used as a placeholder when there is no valid data available.

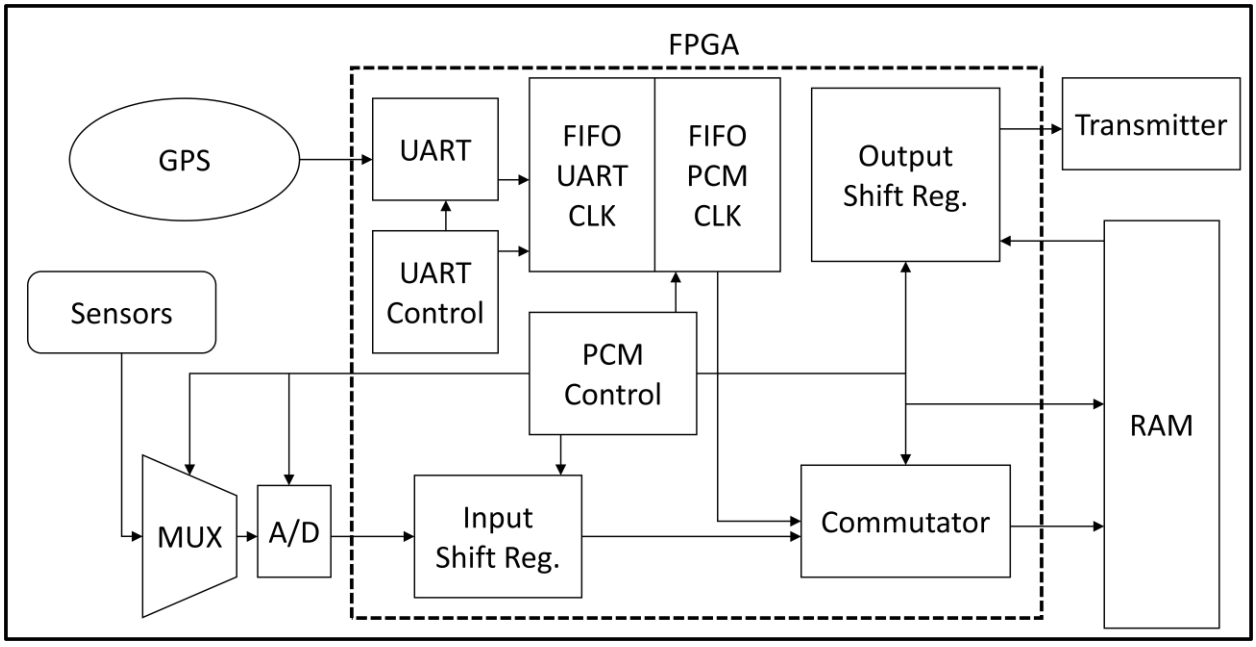

Figure 2: New MIDAS PCM encoder block diagram

Although inserting the raw UART data into the telemetry stream made the MIDAS processing easier, it posed a problem for monitoring the GPS data at the ground station. The telemetry software we employed was unable to decode and display the complex GPS messages in real time. In order to solve this problem, I created a custom decom box. This decom box extracted the UART data from the PCM stream and sent it to a PC for further processing. Figure 3 shows a functional block diagram of the decom box. Using data and clock signals supplied by the ground station, the data is shifted into an input shift register. Control logic identifies synchronization words to determine frame boundaries. The UART words can then be located and checked for valid data. Valid data is determined by checking the most significant bits which are zero if valid, or non-zero if invalid as described above. Valid data is then written to a FIFO. The custom UART core designed for the MIDAS FPGA is then re-used here to send the data in the FIFO to a COTS UART to USB converter.

This scheme not only allowed all of the GPS data to be recorded, but also allowed it to be available to a PC COM port in real-time, enabling us to use Mayflower's own parser program to decode and view the GPS data. The ability to view this data in real-time was essential during the development and debugging of the rounds, as well as the flight tests.

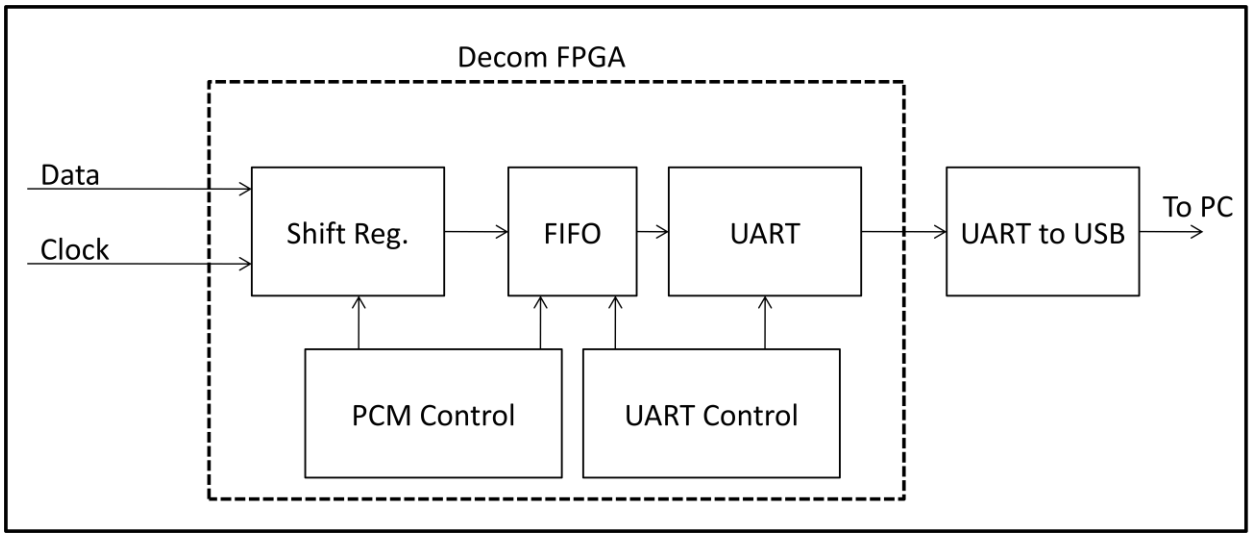

Figure 3: Decom Box block diagram

TELEMETRY BANDWIDTH CONSERVATION

As FCMortar progressed, more of the finalized design was added to the rounds. A DSP was added to the design to implement GNC algorithms. Telemetry frames now needed to include GNC information. It was decided that half of the frames would contain MIDAS sensor data, and half GNC data. This effectively cut the MIDAS data rate in half creating a severe data bandwidth shortage. Unique test conditions presented a solution to this problem. Much of the telemetry bandwidth was taken up by high speed sensors that were only needed in-bore at the beginning of the flight. By saving this high speed data for a short duration at the beginning of flight and then playing it back over the rest of the test, the available telemetry bandwidth could be significantly increased.

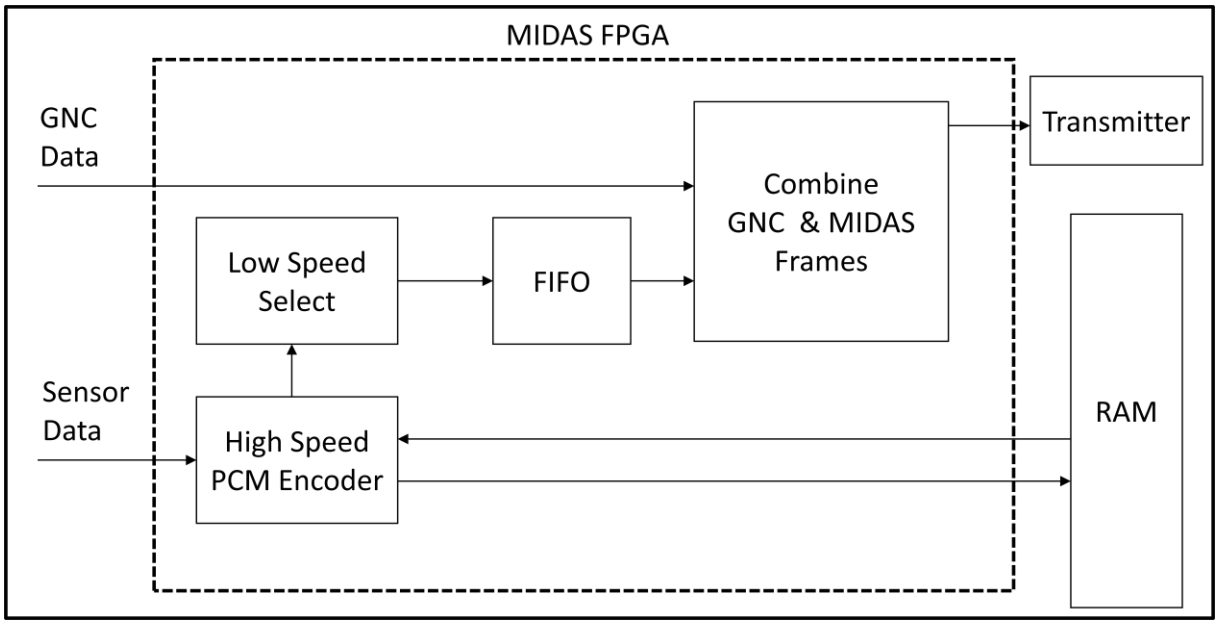

Figure 4: Telemetry Bandwidth Conservation block diagram

Figure 4 shows a functional block diagram of the telemetry bandwidth conservation scheme I employed. The high speed PCM encoder block implements the functions of the PCM encoder shown before in Figure 2. This runs at a rate of 4 Mbit/s with frames made up of 96, 16 bit words. These frames are saved in a circular buffer of 32768 words using the off chip RAM. 48 of the words are selected from the 96 word high speed frames to create 2 Mbit/s low speed frames. These words are saved into a FIFO and combined with the 2 Mbit/s GNC data to form a 4 Mbit/s combined PCM stream. Once a g-switch indicator triggers, the PCM encoder waits 100 ms and then switches from saving data to reading from RAM. 32768 words provide about 131 ms of record time, thus sensor data 31 ms before the g-switch and 100 ms after the g-switch is saved in RAM. This saved in-bore high speed data is inserted into one word per frame of the low speed MIDAS frames. At one word per frame at 2 Mbit/s, it takes about 12.5 seconds to transmit all of the saved high speed data. This allows the data to be transmitted more than twice for a typical flight test that lasts about 30 seconds.

Table 1 shows the statistics of the saved in-bore high speed 96 word frames. The sensors are grouped by sampling rate. The "Number of sensors" column indicates the number of sensors that were sampled at the given sampling rate. The "Rate sub-total" column is the sampling rate multiplied by the number of sensors. In all, there were a total of 30 sensors, with 10 of those sensors sampled at high rates, mainly to capture short in-bore events. The total data rate of all of the sensors was 224 K samples/s. Table 2 shows the statistics for the lower speed MIDAS frames that were transmitted throughout the flight test. Here only one sensor was required to be sampled at a high rate, allowing all of the necessary data to be captured in 48 word frames at a total sampling rate of 117 K samples/s.

This shows the dramatic increase in telemetry bandwidth gained by this scheme. If all the MIDAS sensor data was sampled at the same rate throughout the whole flight, roughly twice of the bandwidth would have been required. By only saving the high rate data for a short period at the beginning of flight where it was crucial, and then integrating this data into the rest of the telemetry frames over the course of the flight test, the MIDAS telemetry rate could be cut in half.

Words/Frame	Data rate (K Words/s)	Number of sensors	Rate sub-total (K Words/s)
1	2.60	20	52.1
2	5.21	5	26.0
4	10.4	2	20.8
16	41.7	3	125
		Total:	**Total:**
		30	224

Table 1: High speed frame statistics

Words/Frame	Data rate (K Words/s)	Number of sensors	Rate sub-total (K Words/s)
1	2.60	29	75.5
16	41.7	1	41.7
		Total:	**Total:**
		30	117

Table 2: Low speed frame statistics

ON-BOARD DATA STORAGE USING A MICROSD CARD

Once again problems arose in later flight tests that resulted in the development of new telemetry techniques. Interference problems between the telemetry transmission and GPS reception initiated a search for alternative means of telemetry data recovery. Since the test rounds were recoverable, on-board storage was a viable option that would solve the GPS interference problem. I developed a storage scheme using MicroSD cards which are small, cheap, and contain abundant data storage.

SD cards are equipped with a standard serial peripheral interface (SPI) supported by many microprocessors (3). The high speed PCM data, however, could not be processed by an inexpensive microprocessor. It follows that the ideal implementation was to use an FPGA to format the incoming PCM data and to implement the SD card interface using a soft processor instantiated into the FPGA. A CoreABC soft processor from Actel was chosen which allowed for easy programming in assembly language within Actel's Libero design environment and provided bus interfaces to Actel SPI and UART cores (4). Due to occasional delays during the SD card writing, an external RAM was added to the design for additional data buffering. Figure 5 shows a simplified block diagram of the SD card recording implementation. During data recording, the incoming PCM stream is processed and written into a FIFO. When the FIFO is almost full, data from the FIFO is written to RAM using the FIFO address counter for the address. After 512 bytes have been written to RAM, the RAM data is read back using the SD address counter and written to the SD card though the SPI interface. To read from the SD card, no buffering is necessary. Data is read through the SPI interface and sent out of a UART peripheral.

The design was verified to write up to 4 Mbit/s, with reading speed limited by the UART receiver. To date this design has not been used in a flight test, but it has been employed in a wind tunnel test where the test environment made RF transmission impossible.

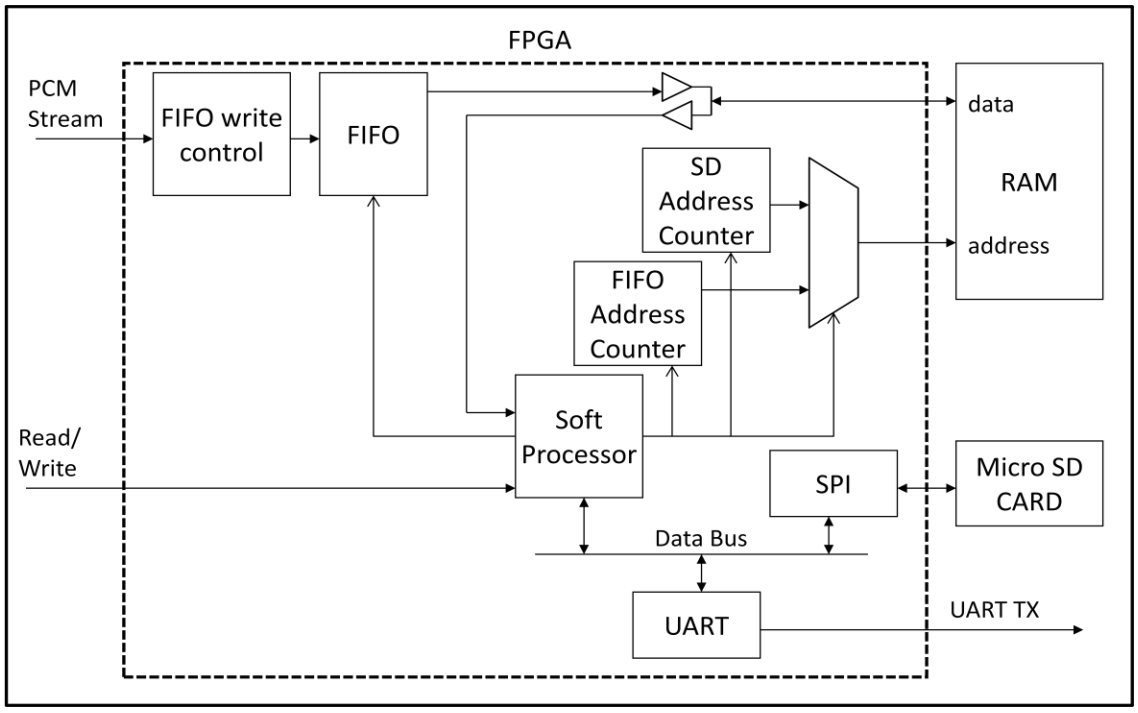

Figure 5: SD card control block diagram

CONCLUSION

The FCMortar program presented several problems which called for creative solutions. Custom decommutation, on-board data storage, and telemetry bandwidth conservation techniques were all developed to overcome unique diagnostic and telemetry challenges. In addition to advancing the FCMortar program, these solutions will extend ARL's telemetry capabilities for future programs.

NOMENCLATURE

A/D – analog to digital converter
ARL – Army Research Laboratory
CLK – clock
COM – communication
COTS – commercial off-the-shelf
Decom - decommutator
DSP - digital signal processor
SD card – secure digital card
FCMotar – flight controlled mortar
FIFO – first in, first out (memory)
FPGA – field programmable gate array
GNC – guidance, navigation, and control
GPS – global positioning system
MIDAS - Multifunctional Instrumentation and Data Acquisition System
PC – personal computer

PCM – pulse code modulated
RAM – random access memory
Reg. – register
SPI – serial peripheral interface
UART – universal asynchronous receiver/transmitter
USB – universal serial bus

ACKNOWLEDGEMENTS

I would like to thank two of my co-workers at ARL for their contributions:

- Pete Muller for designing the MIDAS hardware and original FPGA PCM encoder.
- Mark Ilg for the initial ideas to save high speed in-bore data and to use an SD card for onboard storage.

REFERENCES

1. Condon, John A. et al, "Design and Experimental Results of a Developmental 81mm Flight-Controlled Mortar," Aberdeen Proving Grounds, MD, U.S. Army Research Laboratory, 2011.

2. Chu, Pong P., RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability, Hoboken, NJ, John Wiley & Sons, Inc., 2006. p. 652.

3. Technical Committee SD Card Association, "SD Specifications Part 1 Physical Layer Simplified Specification Version 2.00," San Ramon, CA, SD Card Association, 2006.

4. Actel Corporation, "CoreABC v3.1 Handbook," Mountain View, CA, Actel Corporation, 2010.