Journal of Engineering and Applied Sciences 2 (8): 1290-1296, 2007
ISSN: 1816-949X

An Efficient Embedded System Design for Capturing and Storing Analog Data

Omar Elkeelany, Mohammed A.S. Abdallah and Ali Alouani
Department of Electrical and Computer Engineering,
Tennessee Technological University, USA

Abstract: Existing computer based data acquisition systems are cumbersome, expensive and have limited mobility. Part of the problem is the lack of integration of different components of such systems. Recent advances in systems-on-a-chip technology allows for compactness and integration. Computer based data analysis and interpretation requires acquired data in a digital format. To design future handheld data acquisition system, one needs to capture signals and store them into a non-volatile digital memory device. When writing to a Flash memory, existing systems require a personal computer. A prerequisite to the design of future handheld data acquisition system is the development of a Flash memory controller that writes data directly to a Flash memory device, without using a personal computer. The aim of this study is to design an efficient Flash memory controller that writes temporary collected data to a detachable Flash device. To achieve this task without the need of a computer, systems-on-a-chip design technology will be used. We analyze two mechanisms, to store these signals into a secured digital card. The first approach uses single-block write and the second one is based on the multiple-block write. This study shows that the multiple-block write mechanism is more efficient in terms of the time it takes to store the temporary acquired signals into the secure digital card.

Key words: Data acquisition, systems-on-a-chip, block writing analysis, flash memory, FPGA design tools

INTRODUCTION

Computer based multi-sensor data measurement systems are used extensively in many application areas such as quality assurance, data conditioning and automation applications (Chen and Huang, 2005; Gaydecki, 1999; Lindqvist, 2005). Measurements are typically collected through dedicated cables that are attached to the data acquisition card before it reaches the computer. These systems have internal interface slots, which hold various data acquisition cards. Compared to portable handheld devices, these computer based systems are cumbersome and expensive. An example of such systems is shown in Fig. 1. Moreover, these systems are limited since they consume too much power (Haiying *et al*., 2007), require significant operator training and expensive maintenance. Even current portable data acquisition systems suffer from the above limitations.

Systems-on-a-chip design technology provide the computational power to eliminate the need of a computer and interfaces by providing integrated design of processors, on chip memory and peripheral controllers. State of the art advances in integrated circuit technology, especially in the programmable logic devices, initiated the era of Systems-on-Programmable-Chips. Field

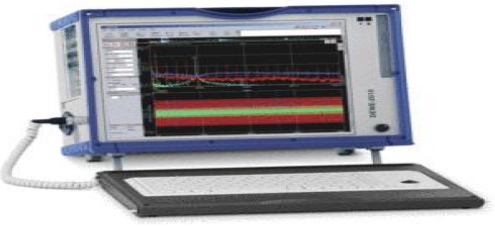

Fig. 1: Computer based data acquisition system

Programmable Gate Arrays (FPGA) are modern, complex programmable logic devices (chips) that are manufactured with a high density of configurable blocks. Typically, a FPGA is made up of Digital Signal Processors (DSPs), Configurable Logic Blocks (CLB), memory cells, input output blocks and microprocessors (Fig. 2). FPGAs were used as DSP to preprocess sampled data signals, such as implementing Discrete Cosine Transform for data compression (Emil *et al*., 1999; Pimentel *et al*., 2001). Typically, power consumption of FPGAs is higher than fully customized integrated chips (Vahid, 2001). However, modern FPGAs such as the Cyclone-II match Application-Specific Integrated Circuit (ASICs) that consumes as little as 12 milli-Watts (Altera, 2007).

Corresponding Author: Omar Elkeelany, Department of Electrical and Computer Engineering, Tennessee Technological University, 115 W 10th Street, Box 5004 Cookeville, TN 38505, USA

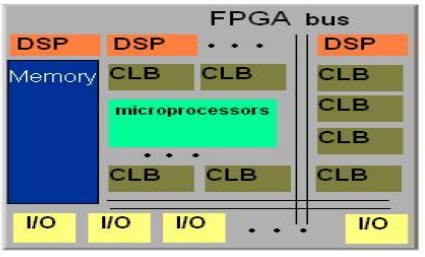

Fig. 2: Various components of modern FPGA

The configuration of a FPGA can be customized to communicate with external peripherals, processing units and memory devices. Using this FPGA technology, the development of new applications becomes very affordable. Examples of advanced commercial FPGA-based low cost battery operated handheld devices are Personal Digital Assistants (PDAs), cell phones and MP3 players.

Nowadays, handheld devices use this low-power FPGA technology, to preserve battery life. Modern FPGAs also allow the integration of processing capability, on-chip memory and other components on one chip. This is a key feature in designing low power affordable and handheld integrated data acquisition and monitoring systems.

In a related work (Wolfgang, 2007), a system was proposed to achieve the overall functionality of a telemedicine application since data can be transmitted over email or FTP servers to the designated location with great reduction in sophistication and set-up costs. The data stored in the memory card could also serve as inputs to other signal processing applications and storage mechanisms like ecgML which stores the patients' record in XML format in a hierarchical database. However, this system did not consider writing data to detachable Flash memory device.

The proposed system will be designed using the latest advances in FPGA chip design. It will eliminate the need of a personal computer and the cumbersome and expensive data acquisition systems currently in use in the field. Moreover, the system will be capable of doing data preprocessing such as compression and encryption. For further data analysis, detachable Flash memory device will be used and a Flash controller will be designed.

FLASH MEMORY

Flash memory was founded by Toshiba in 1980 to allow data storage even when the memory device is not connected to the power source. Flash memories are used whenever shock resistance is a key requirement, as is the case with various consumer electronic devices like in Personal Digital Assistants (PDAs) (Kingston, 2006). With increased capacity of Flash memories, which is now on the order of Giga bytes, a valuable solution for detachable data storage applications is provided.

Table 1: Access time of various non-volatile and volatile memory storage media

Memory	Typical access time	Data retention
after power downOptical #	300 ms	Yes
Magnetic+	2-25 ms	Yes
FLASH#	10-200 μs	Yes
SRAM#	10 ns	No

(Sadik, 2007), + (Western Digital 2007)

One of the commonly used Flash memory devices is the Secured Digital (SD) card. A lot of technical details about the Secured Digital (SD) card are presented in (Sadik, 2006; SanDisk, 2004). The SanDisk SD Card, for example, provides up to 1024 million bytes of memory using Flash memory chips, which were designed especially for use in mass storage applications. In addition to the mass storage specific Flash memory chip, the SD Card includes an on-card intelligent controller which manages interface protocols, security algorithms for copyright protection, data storage and retrieval, as well as error correction code, defect handling and diagnostics, power management and clock control.

The storage Flash chip can be easily transferred to a personal computer whenever needed (e.g., in a secured digital SD card). We use Flash memory because of its compact, fast access, low-power consumption, non-volatile property and multiple write characteristics, as compared to mechanical magnetic or optical media, which does not have all of these desired features. Table 1 summarizes the technical differences across various memory devices.

Flash-memory is non-volatile, shock-resistant and power economic device. With recent technology advancements, Flash memory devices have increased Capacity and reliability and becoming much more affordable than ever. As a result, Flash memory is now among the top choices for storage media in ubiquitous computing. Researchers have been investigating how to utilize Flash memory technology in existing storage systems, especially when new challenges are introduced by the characteristics of Flash memory. In particular, Kawaguchi *et al.* (1998) proposed a Flash memory translation layer to provide a transparent way to access Flash memory through the emulating of a block device. Wu *et al.* (1994) proposed to integrate a virtual memory mechanism with a non-volatile storage system based on Flash memory. Native Flash memory file systems were designed without imposing any disk-aware structures on the management of Flash memory (D. Woodhouse; Aleph

One Company). Douglis *et al.* (1994) evaluated Flash memory storage systems under realistic workloads for energy consumption considerations (Douglis *et al.*, 1994). Chang and Kuo (2002) focused on performance issues for Flash memory storage systems by considering an architectural improvement (Chang and Kuo, 2002), an energy-aware scheduler (Chang and Kuo, 2001) and a deterministic garbage collection mechanism (Chang and Kuo, 2002). Many Flash memory device implementations and specifications were proposed from the industry, such as (Intel; SSFDC Forum; Compact Flash Association; M-Systems). Across the vast number of research efforts on Flash memory devices, much work is done on garbage collection, e.g., (Woodhouse, 2007; Douglis *et al.*, 1994; Anoymous, 2006). Existing implementations of Flash memory devices usually adopt static table-driven schemes. It can be with a fixed-sized granularity of Flash memory management, e.g., 16KB as a block size to manage used and available space of Flash memory (Woodhouse, 2007; Chang and Kuo, 2002; Anonymous, 2006; Intel; SSFDC Forum; Compact Flash Association). The granularity size of Flash memory management is a permanent and unchangeable system parameter. Such traditional design works pretty well for small-scale Flash memory storage systems. With the rapid growth of the Flash memory capacity, severe challenges on the Flash memory management issues will be faced, especially when performance degradation on system start-up and on-line operations would become a serious problem. In addition, current Flash memories are written to using a personal computer, through a permanent interface (i.e., soldered chips on circuit boards).

The objective of this study, is to design a stand alone system that continuously writes data to Flash memory devices without the need of a computer.

PROPOSED SYSTEM DESIGN

Figure 3 shows the typical components of a computer based data acquisition systems and Fig. 4 shows the proposed Handheld Data Acquisition System. As shown in figures, components of the computer based data acquisition system are custom designed in the proposed Handheld Data Acquisition System. There is no need for the computer when using the proposed system.

The Flash memory controller will be designed to copy the collected data temporary stored in the FPGA internal buffers. The Flash memory controller will be designed in the FPGA to write data directly to the Flash memory card, with a minimal interface. It is advantageous to use the FPGA internal memory cells for buffering because their access time is much smaller than that of external memories

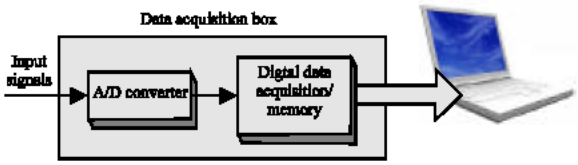

Fig. 3: Computer based data acquisition system

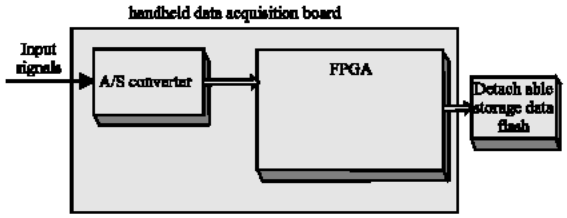

Fig. 4: Future handheld data acquisition system

The Flash memory controller will follow an error detection algorithm in the data writing process to ensure the integrity of the stored data.

Overall, the FPGA will record acquired data after filling its internal storage buffers. In order to optimize the acquire-and-write processes, the Flash memory controller can use single or multiple block writing mechanisms. Single block writing can be done to ensure data integrity, by using an integrity check value (Cyclic redundancy check) at the end of each block. However, if multiple blocks are to be written sequentially to the Flash memory, the total write time can be reduced significantly by use of a more sophisticated write mechanism. In the multiple blocks write mechanism, the total write time decreases. The number of blocks to reach an optimal performance, without degrading data integrity, is subject to research in this paper. This is to avoid excessive waiting times for data storage that will affect the overall performance of the acquire-and-write processes.

In traditional systems, Flash memories are written by a host personal computer, through a permanent interface (i.e. soldered chips on circuit boards). Such use of Flash memory devices is common in embedded systems to store configuration information. One design choice made in the proposed system is to use detachable Flash memory devices. Sampled data must be written to the Flash memory device to allow further remote analysis and interpretation. This introduces technical design challenges. Particularly, Flash memory must be written by a customized hardware not a personal computer. Hence, the proposed Flash memory controller must address the following design requirements.

High data rate: A Detachable Flash memory should have a small and robust physical interface. This limits the

maximum number of data and control pins in the interface. Consequently, this affects the writing speed, as serial data communication must be employed. We use high clock frequency in serial mode to overcome this challenge. Timing and clock signals between the Flash memory controller and the data acquisition units must be properly matched. We use phase locked loop and clock dividers to achieve this matching.

Data integrity: Flash memory must be initialized before the storing process. As Flash memory cards are detachable, card detection is another challenge. The storing process can not be done unless the Flash memory is attached. Continuous monitoring of the Flash memory during both the reading process and the writing process is done to ensure that the data is passed to the Flash memory. We also use Error check code to check the valid arrival and storage of the data into the Flash device.

Data quality: The witting process is limited by the access time of the Flash memory device (Table 1). High quality data should be received fully (without dropped blocks), in order and in time. There is no need to store data which is incomplete or out of order. One solution we used is to use internal buffer. We assume no data compression and fixed data arrival rate. We make sure that the processing rate is faster than the arrival rate. In future research, we will consider variable data arrival rate (e.g. compressed data) and will evaluate the optimal buffer depth, such that no blocks are dropped.

DESIGN METHODOLOGY

Mixed hardware/software architecture is employed using Hardware Description Language (HDL). As a rule of thumb, any time-critical task is implemented in hardware, while other functions are developed in software using embedded C programming language. The FPGA bus signals and the parallel input output lines form a common interface between the hardware and the software components. The software is used to capture the sound data and convert it to appropriate file format for storage. The hardware is used to configure the A/D converter and provide proper timing signals.

The NIOS-II processor core is connected to its components through the Avalon switch interface. The Avalon interface contains logic and the arbitration to manage this connection. The system components are specified in a Graphical User Interface (GUI) and once the system is generated all the necessary logic is automatically created in the form of HDL files (Altera Quartus II, 2006; Altera DE2, 2006). The hardware language description from the user side is limited to building the top-level module where the NIOS-II processor is instantiated. The NIOS-II processor is a 32-bit RISC processor. It has a 4-kB instruction cache and a 2-kB data cache. The processor runs on a clock of 50 MHz. Since the NIOS-II processor follows Harvard architecture the data and the program are stored separately.

The A/D converter used in the proposed system is a WOLFSON WM8731. It can support a sampling frequency of 8 -96 KHz. It has been built in digital filters to improve the sound quality. Sound data is fed to the NIOS-II processor in FPGA serially (Wolfson, 2006).

The proposed system will store input signals (audio sound) in pulse code modulation using the Wave (.wav) file format. The sound will be stored as a 16-bit mono with a sampling frequency of 50 KHz. The data written should not exceed the total length of the wave file (Bashir, 2007).

Capturing the input data and storing it into the detachable Flash memory device has many stages. Each stage has its own time to be completed. But the most important stage is the stage of writing the sampled data into the Flash memory device because it is the slowest operation of all operations preceding it. So, it may cause timing problems. Herein, two approaches are proposed to be able to reduce the witting time into the Flash memory device. The SDRAM in the FPGA is used to store the sound samples in large array blocks. After this array is filled then the write operation is performed for the whole array. This ensures the continuity of sampling. Of course, the size of array of data is limited by the size of the SDRAM.

There are two approaches to achieve our goal. The first one is to use a single-block write approach. A single block data write into the Flash memory device involves the write of 512-bytes of data. The authenticity of the data written into the Flash memory device from the NIOS-II processor is ensured using the Cyclic Redundancy Code (CRC) checksum. A 16-bit CRC is appended to each data block. The Flash memory device controller checks the CRC and if the CRC is the correct CRC then the data is written. But if any failure happens, the data is discarded.

The number of blocks can vary but is limited by the size of the file in which these blocks will be stored. In the single-block write approach, a write command is needed for each block. So, if we need to capture and store x blocks into the detachable Flash card, this command must be called x times. Figure 5 shows the basic steps of read, store and process any number of blocks by the single-block write approach. All details about the Flash card and how to write into it and more specifications about the contents of each command and CRC are available in (Kawaguchi *et al*., 1995).

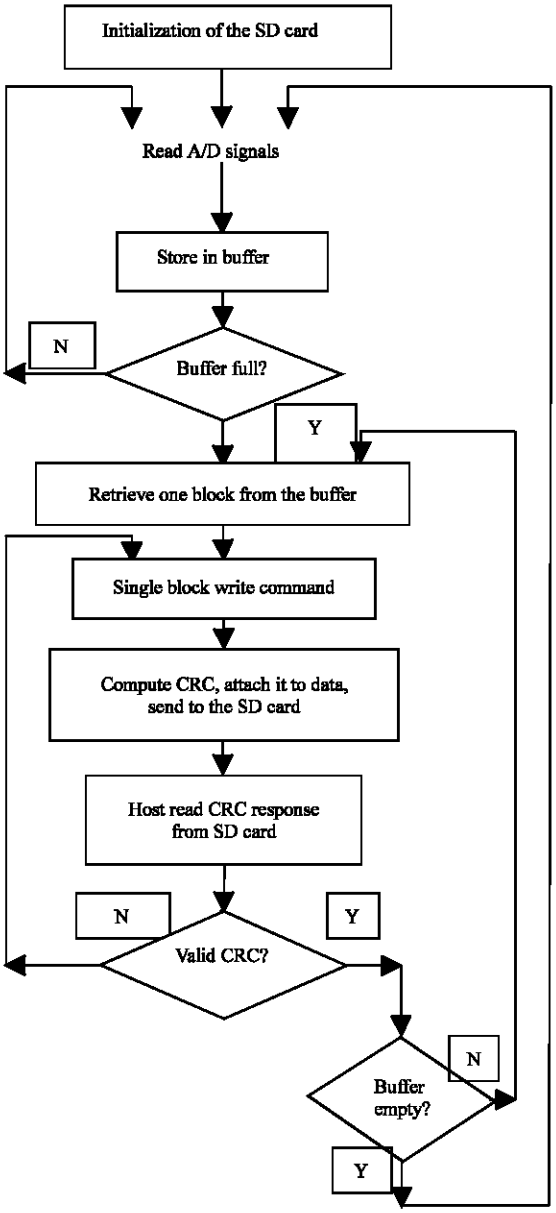

Fig. 5: Flow chart for the single-block write approch

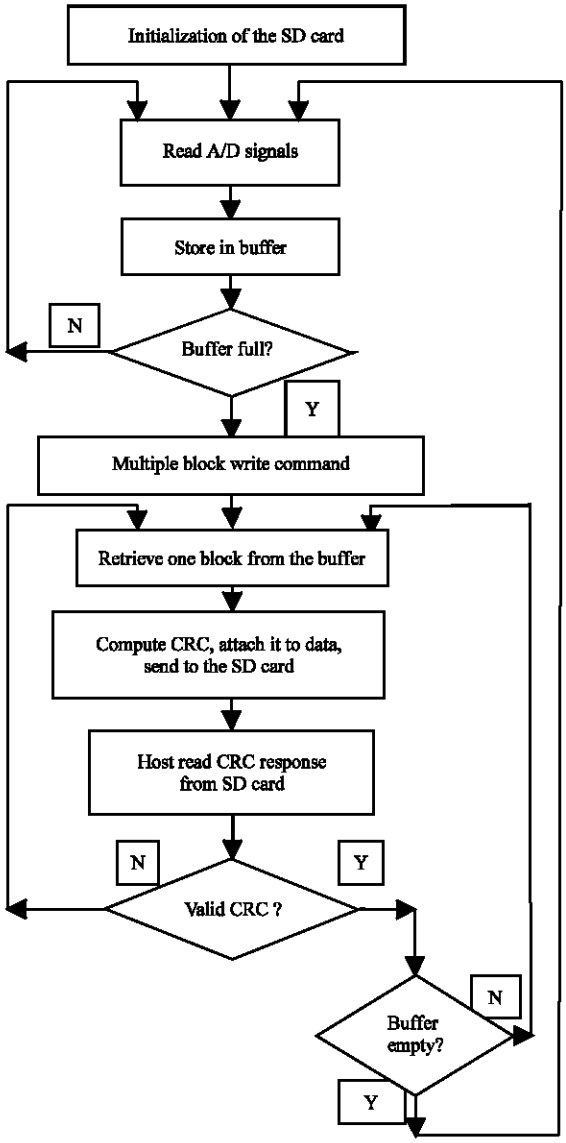

Fig. 6: Flow chart for the multiple-block write approch

The second approach is to use multiple-block write. A write command is called once to write all blocks into the Flash memory device. When this command is called, it is followed by a sequence of the desired blocks. This process is terminated either there is a stop command after all blocks are written or the whole number of blocks must be predetermined in advance. In this proposed approach, the number of blocks is predetermined before running the program. For the purpose of obtaining the comparison study between the single-block write approach and the multiple-block write approach, the experiment is repeated with different numbers of blocks. For each number of blocks, the time required to write the input data into the Flash memory device is calculated.

Figures 6 shows the basic steps of read, store and process any number of blocks by the multiple-block write approach. As shown from this figure, after reading the input signals and storing it in the internal buffer, the writing function is called. In the beginning of this function, the multiple-block write command is called once. Then the 512 bytes for the first block are sent to the Flash memory device with a proper CRC. The block is successfully written into the Flash memory device if the Flash memory controller receives a valid CRC response. Then, this process is repeated for the next blocks until all desired blocks are written.

RESULTS AND DISCUSSION

Our current system implementation on the Altera NIOS-II processor uses only 15% of the total chip capacity, which is 5114 out of the 33,216 Logic Elements on the Altera Cyclone-II EP2C35F672C6 FPGA chip. The worst-case prorogation delay observed for the system is 12.04 ns, which is less than the 50 MHz clock period of 20ns. The Cyclone-II FPGA consumes as low as 12 mW of power, which rivals semi-custom ASIC counterparts of similar cost.

Firstly, the Altera SoPC™ builder is used to configure the NIOS-II processor, peripheral components and memory to the target FPGA. Then compiling and running processes are done in the Altera Quartus environment. Finally, the NIOS-II Integrated Development Environment (IDE) is used to compare both approaches. Both approaches are applied in this IDE via calling two different functions one for each approach. To get the results of each approach, different numbers of blocks are considered as the inputs of the experiment. The time it takes to store the blocks into the Flash memory device is considered as the output of this experiment.

Figure 7 illustrates the performance of both approaches. The number of blocks is the x-axis and the time it takes to store these blocks is the y-axis. From this figure, it is noticed that multiple-block write approach is more efficient than the single-block write approach. It takes less time to store the desired blocks into the Flash memory device.

From Fig. 7, it is shown that the performance of both approaches is approximately linear in shape. As the number of written blocks increases, the time it takes also increases. But the multiple-block write approach's performance is better than the other approach. So, if the number of blocks is n, the time required in to write a block in the single-block write approach is τ_S, then

$$T_S = n \times \tau_S \quad (1)$$

T_S is the total time to write n blocks using single block write approach. On the other hand, if the time it takes to execute a multiple block write command is λ and the actual time it takes to a write a block is

τ_M, then the total time to write n blocks using multiple block write approach is:

$$T_M = \lambda + n \times \tau_M \quad (2)$$

When the two functions are subtracted, an almost linear function is resulted. This difference function represents the saved time of the multiple-block write approach over the single-block write approach and is

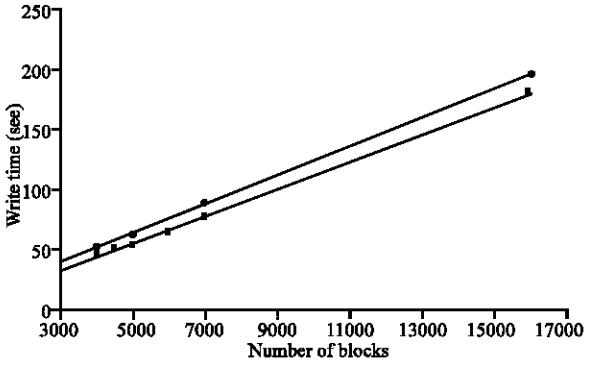

Fig. 7: Comparison between the write time of the single and multiple-block mechanisms

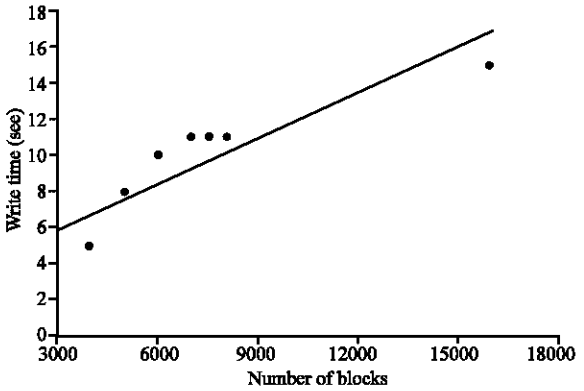

Fig. 8: Saved time by using the multiple block write approach

plotted in Fig. 8. It shows the time difference as related to the total number of the blocks increases, using the two block writing approaches.

CONCLUSION

Capturing and storing signals are required in many fields for the purpose of security, maintenance, control, etc. Low cost design techniques are needed to make data acquisition affordable. The study took advantage of recent development in the area of systems-on-programmable chips to build the bridge toward achieving this goal.

In this study, a flash memory controller has been designed, implemented and successfully tested.

The proposed controller design allows one to store digital data into a Flash memory device without the need of a personal computer and complex interfaces. This is an important step toward designing compact, low cost, handheld data acquisition systems. Two approaches for storing the data into the Flash memory card are presented in this study. A comparative study between these approaches was carried out. The finding is that the

multiple blocks write approach is more efficient than the single block write approach in terms of the time it takes to store the data used.

Future work will use the outcome of this study to design hand held data acquisition systems. The design of a single channel with on board processing capability is currently under way.

ACKNOWLEDGEMENT

This research was partially supported by the ECE Department and the Center for Manufacturing Research (CMR) at Tennessee Tech University.

REFERENCES

Anonymay, 2006. Yet Another Flash Filing System. Apeph one company.

Altera DE2, 2006. Using the SDRAM Memory on Altera's DE2 board with Verilog Design.

Altera Quartus II, 2006. Version 6.0 Handbook, Vol. 4: SOPC Builder.

Altera, 2007. The Cyclone II Power Advantage-1/2 the Power of Competing 90-nm Low-Cost FPGAs: www.altera.com/products/devices/cyclone2/features/power /cy2-power.html

Bashir, A., 2007. Wave File Format. http://technology.niagarac.on.ca/courses/ctec1631/WavFileFormat.html.

Chang, L.P. and T.W. Kuo, 2001. A Dynamic-Voltage-Adjustment Mechanism in Reducing the Power Consumption of Flash Memory for Portable Devices. IEEE Conference on Consumer Electronics.

Chang, L.P. and Kuo T. W., 2002. An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems. The 8th IEEE Real-Time and Embedded Technology Applications Symposium.

Chang, L.P. and T.W. Kuo, 2002. A Real-time Garbage Collection Mechanism for Flash Memory Storage System in Embedded Systems. The 8th International Conference on Real-Time Computing Systems and Applications.

Chen, I. and C. Huang, 2005. A service oriented Agent Architecture to support Telecardiology services on demand. J. Med. Biomed. Eng., 25: 73-79.

Anonymas, 1998. Compact Flash ™ 1.4 Specification.

Dewetron , 2007. DEWETRON PC Instruments. National Instruments, Data Acquisition Card for Laptops and PDAs, http://www.dewetron.com/products/ pc-instruments/dewe-2010/

Douglis, F., R. Caceres, F. Kaashoek, K. Li, B. Marsh and J.A. Tauber, 1994. Storage Alternatives for Mobile Computers. Proceedings of the USENIX Operating System Design and Implementation.

Emil, J., P. Gelabert, R. Adhami, B. Wheelock and R. Adams, 1999. Real Time Holter Monitoring of Biomedical Signals, DSP Technology and Education Conference DSPS, Houston, Texas.

Gaydecki, P., 1999, A real time programmable digital filter for biomedical signal enhancement incorporating a high-level design interface, Department of Instrumentation and Analytical Science, UMIST, Manchester M60 1QD,UK.

Haiying, W., J. Benjamin, A. Francisco and B. Norman, 2007, ecgML: Tools and Technologies for Multimedia ECG Presentation: http://www.idealliance.org /papers/dx_xmle03/papers/ 04-05-02/04-05-02.html

Anonymas, 2006. Understanding the Flash Translation Layer (FTL) Specification.

Kawaguchi A., S. Nishioka and H. Motoda, 1995. A flash-memory based File System, Proceedings of the USENIX Technical Conference.

Kingston, 2006. Flash memory cards, http://www.kingston.com/products/pdf_files/flashMemGuide.pdf .

Lindqvist, P., 2005. Compression and Storage of Medical Data in Pacemakers, Master's Thesis, Royal Institute of Technology Stockholm, Sweden.

Pimentel, B.S., V. de Aliva, R.L. Campos, A.O. Fernandez and C.J. Nunes, 2001. A FPGA implementation of a DCT-based digital electrocardiographic signal compression device, IEEE, 14th Symposium on Integrated Circuits and System Design.

Sadik C. Esener, 2006. Flash memory cards. http://www.wtec.org/loyola/hdmem/01_01.htm#f01_01.

Samsung, 2003. Memory Data Sheet.

SanDisk, 2004. Secure Digital Card, Product Manual, Version 2.2, Document No, 80-13-00169.

Anonymas, 1999. SmartMediaTM Specification.

Vahid, G., 2001, Embedded systems design. A Unified Hardware/Software Introduction. ISBN: 0-471-38678-2.

Western Digital, 2007. Western Digital Caviar RE2 WD4000YR. http://www.storagereview.com/articles/200510/WD4000YR_2.html.

Woodhouse, D., 2007. Red Hat, Inc. JFFS: The Journaling Flash File System.

Wolfgang, R., 2007. Bioscope Features. http://www.reco-medical.de/english/bioscope-c.html.

Wolfson Microelectronics, 2006. WM8731/WM8731L, Production data, Rev., 3.4.

Wu M. and W. Zwaenepoel, 1994. eNVy: A Non-Volatile, Main Memory Storage System, Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems.