FPGA-Based Multi Protocol Data Acquisition System with High Speed USB Interface

S. Thanee S. Somkuarnpanit and K. Saetang

Abstract—**We have proposed a data acquisition system with high speed USB interface using FPGA chip as the main processing unit. Since the FPGA has a number of modules on chip, which can operate independently, it can be utilized for the data acquisition system with multi-channels for the connection to four ADC signals with four different protocols of Parallel, SPI, I^{2}C and one-wire protocol. The system is controlled by the software written in the visual C^{++}. It allows the user to be able to interface to a PC for data restoration and monitoring. We found that this system can perform data acquisition with high rate data transfer.**

Index Terms— **Data acquisition, high-speed USB, FPGA.**

I. Introduction

The data acquisition system is broadly utilized in a number of automatic test and measuring equipments. They can be used to collect the required data from any peripheral input devices, such as meters, sensors and etc. via controlling software [1]. The measured data could be stored in the PC. Their values can be shown numerically whereas their relationship can be displayed graphically as a curve on the screen.

This paper proposed a design of the data acquisition system using FPGA [2] interfacing to a PC [3]. The system has capability to receive the digital signals from multi-channels sensors with four different ADC protocols.

II. Overall System

The overall system is shown in Figure 1. It presents the connection to the four different ADC (analog to digital converter) sensors with four different protocols: Parallel, SPI, I^{2}C and One-Wire. The FPGA collects the individual data from all ADC sensors. It processes in the individual protocols. After that it produces a stream of data through the output USB port [4], which sends these ADC data to the PC.

We have written a specific application program to control the PC. This program has a function to communicate to the FPGA so that the PC could prepare itself for the data transfer.

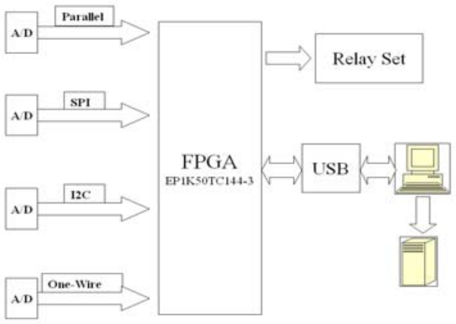


Figure 1 The overall system diagram.

Firstly, the PC will check the FPGA for data availability on the system. After that it will send a set of the instructions to the FPGA for getting these data from USB port. The data will be interpreted into separate data bytes for the individual channels. Finally, the data can be shown to the user, and saved to the main database at the same time.

III. Processing Unit

The processing unit EP1K10TC144-3 from Altera Company is employed for this design. It has 2,880 capacity logic elements or about 50,000 gates, 40,960 internal ram bits and 102 input/output ports for connecting to the external hardware. It supports the power supply at 3 levels, which are 5V, 3.3V and 2.5V. The maximum operating frequency is 180 MHz. This design has used VHDL as the language [7, 8] for writing the code program.

This chip works as the center of the acquisition of the data from all sensors. Its responsibility is to bridge the signals between the ADC inputs to the USB connection, namely, to send/receive the data with the PC. Figure 2 illustrates the internal modules within the FPGA chip [6]. These modules can be described as below:

A. Parallel Protocol

This protocol is the traditional type for most ADC's. It has the advantage of the high speed throughput. This design uses ADC0820 for the peripheral device. Figure 3 presents the simulation of how the FPGA gets data from this ADC. There are two main steps in the conversion process:

- The FPGA sends the start signal to activate the ADC then it will wait for the acknowledge signal.
- After finishing the data converting, the ADC will send the acknowledge signal to the FPGA. Then the FPGA reads the data from the bus. After that the FPGA sends the start signal to activate ADC again for getting the data on next read cycle.

Obviously, this data acquisition is so simple and fast. Thus, this protocol should be employed with the high speed system.

Manuscript received December 27, 2009.

S. Thanee is a graduate student in Master Degree in the Faculty of Engineering at King Mongkut's Institute of Technology, Ladkrabang, Bangkok Thailand.

S. Somkuarnpanit is now with the Department of Electronics, Faculty of Engineering at King Mongkut's Institute of Technology, Ladkrabang, Bangkok Thailand (e-mail: kssuripo@kmitl.ac.th).

K. Saetang is with the Department of Electronics, Faculty of Engineering at King Mongkut's Institute of Technology, Ladkrabang, Bangkok Thailand (e-mail: kskhanit@kmitl.ac.th).

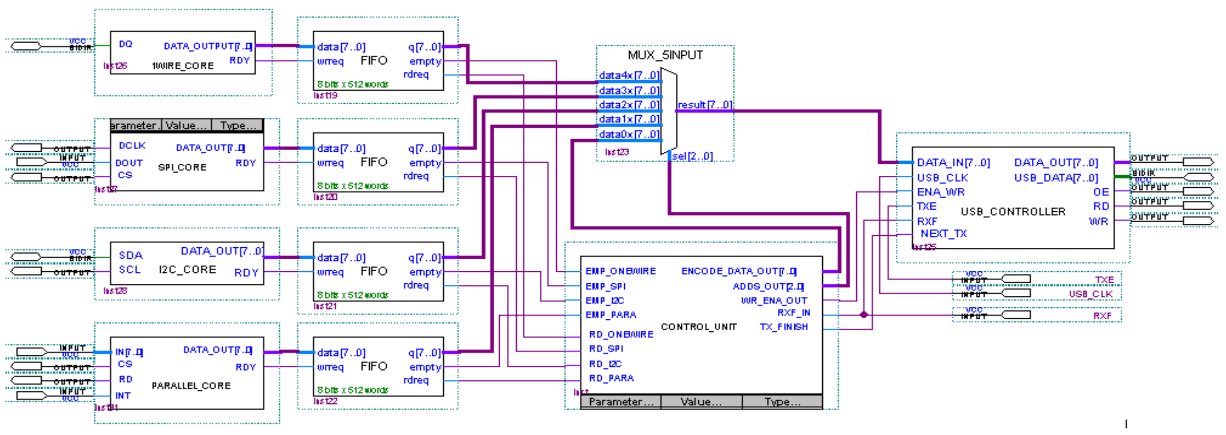


Figure 2 The modules associated in EP1K10TC144-3.

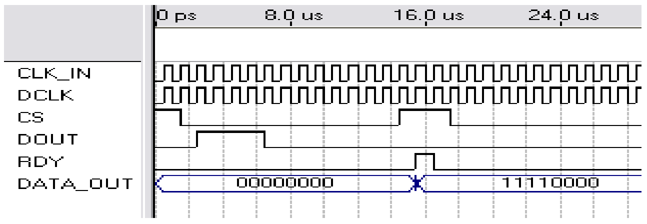


Figure 3 Simulated communication on SPI protocol.

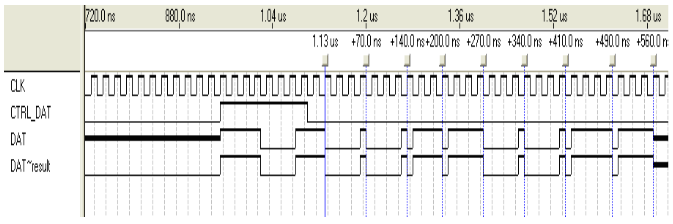


Figure 4 The communications on One-wire protocol.

B. Serial Peripheral Interface (*SPI*) *Protocol*

This protocol was developed by Motorola to accomplish the easy communication, and to reduce the I/O ports. This design utilizes MCP3201 for building the connection with the FPGA. Figure 4 shows the simulation how the FPGA obtains the data from this ADC. The procedures of this data acquisition are similar to those of the previous protocol. The FPGA sends the control signal to activate the ADC, and the ADC will send the acknowledge signal back. After finishing data conversion, the ADC will send a signal to inform the FPGA to read the data from its output. The difference from previous protocol is that the FPGA reads the stream of data in a serial pattern from the MSB to the LSB, instead.

C. Inter-Integrated Circuit (I^{2}C) Protocol

The I^{2}C communication is the protocol, which is designed to reduce the I/O ports. It requires only two signal wires, called SCL and SDA. We use PCF8591 as the ADC with the I^{2}C for this design. The procedures of this data acquisition are quite complicated as following steps:

- The FPGA sends a signal to activate the bus, and to identify the address of the device.
- The FPGA defines the write mode, and sends the command to the ADC that it wants to get the data.
- The FPGA defines the read mode, and identify from which channel it want to read.
- The FPGA reads the data from I^{2}C bus.

After this point, the data has been finished reading in one cycle time. Figure 5 illustrates the example of the data reading by this protocol. We can see that one reading cycle of this protocol takes time more than that of the previous protocol. In the other hand it has a good point of using only two wires in the data communication.

D. One-Wire Protocol

This protocol employs only one signal line in the data communication. The bus is not active unless all data have not been transferred. We used the temperature sensor DS1820 as the input of this One-Wire acquisition data protocol.

The protocol separates the data bit by a *time slot*. It has a length between 60-960 µs, depending on the user assignment and the status of the communication between the master and the slave devices. There are four statuses in the protocol:

- *Reset*: is used to start the communication.
- *Write data* “1” to the slave device.
- *Write data* “0” to the slave device.
- *Read data* from slave.

The first step for this protocol is the reset from the FPGA (the master device). It sends a reset signal to the bus, and waits for the acknowledge signal from the sensor (slave device). After having received the acknowledge signal, the FPGA will send the address command to identify the sensor, and starts the data conversion.

The FPGA waits for the sensors to process the command. Then it sends the reset signal and identifies the device address again. Consequently, it sends the reading command to the sensor to read data from the sensor memory one by one from the LSB to the MSB. Finally, it sends the reset signal to the bus and waits for the acknowledge signal. To obtain the next data, the FPGA will process the same steps again. Figure 7 displays the simulation for the procedures of reset, acknowledge, and writing data of “1100 1100”.

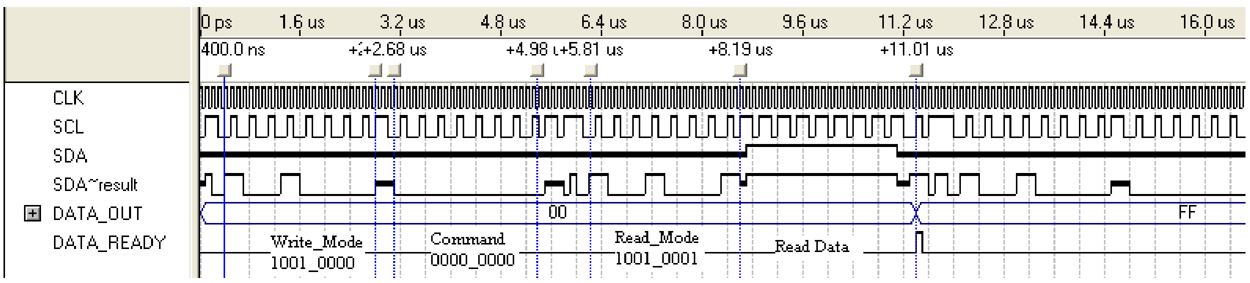


Figure 5 Simulated communication on I^{2}C protocol

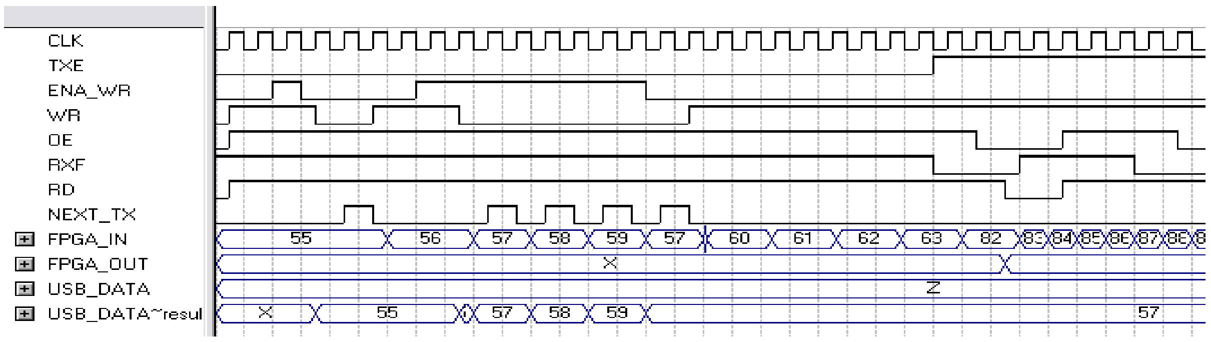


Figure 6 Simulated communication on Control Unit and USB Controller

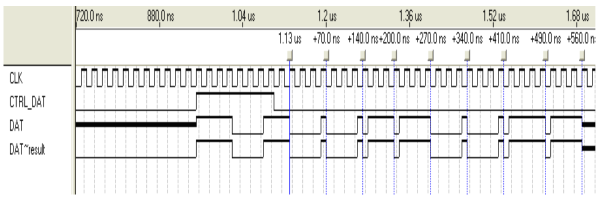


Figure 7 The communications on One-wire protocol.

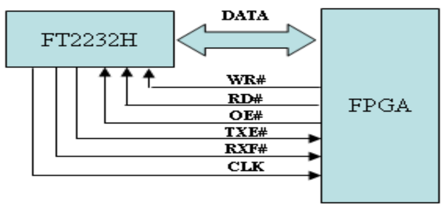


Figure 8 The FT2232H/FPGA interfacing

E. Control Unit and USB Controller

If there is any data appearing on FIFO buffer, the control unit will determine from which channel the data are. Consequently, it will send one-byte data code to the USB port. This first data byte informs the application program on PC that the following data are from which channel. Then it will send a signal to the USB controller again for sending the data in the FIFO to the USB port byte by byte until the FIFO is empty. This means the end of the data sending procedure in the individual channel. If there are another FIFO channel data waiting to be sent out, the FPGA will repeat the same procedures. Figure 6 illustrates the simulation for inter connection between the control unit and the USB controller.

IV. The PC Interfacing

The PC interfacing used in this design is the high-speed USB which can transfer the data at the rate of 480 Mbps. The transfer operation is achieved via *frames*; with the time period of 125 µs. Each frame consists of a number of *Transactions*, which consists of a number of *Packets*. These packets include information about the type of the transaction, the address of the USB device, and the number of the end points required in addition to the data and the CRC packets, and a synchronization packets.

The FT2232H chip from FTDI Corp. [5] has been used to implement this protocol in the design. It is a dual USB to parallel FIFO bi-directional data transfer chip with 4 kilobyte FIFO Tx and Rx buffer, which handles the entire USB protocol on the silicon level. The manufacturer provides the driver D2XX.DLL, which allows full accessibility to all the chip features. This chip provides an interface between the FPGA and the USB port with 8 bit bidirectional data bus, five control signals and one clock signal. Figure 8 illustrates the FT2232H/FPGA interfacing.

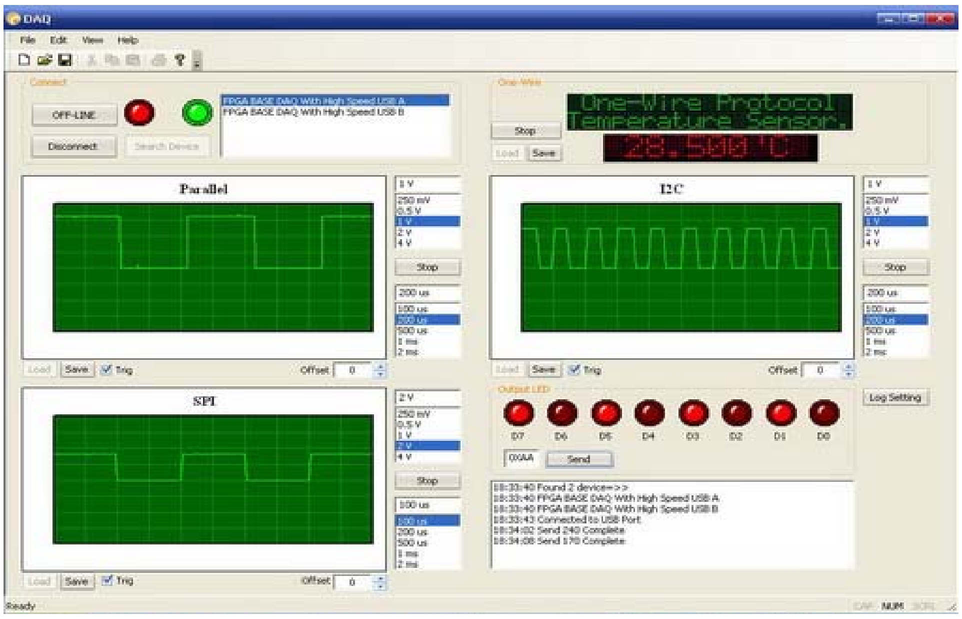
Hardware layer: continuously issues the read request in a "reading worker thread" that will return one or more data, and put the data in a stack from which the data acquisition layer can read it.

Data acquisition layer: When the data requires some kind of interpretation, each sensor will have a thread that collects the data, and save them into the PC. When this thread runs, it moves the data from the queue shared with the hardware layer into its own container, and updates the current value shared with the presentation layer.

This application is responsible to read data from each protocol and presents them to the user in a style of not only a numeric value, but also a graphic relationship. Since the one-wire protocol is used for temperature sensors, it will show the data as the discrete number for the temperature. Figure 9 displays the operation of application program.

V. Conclusions

From our simulations, we can conclude the advantages and disadvantages for the individual protocols as shown in Table1



```
18:33:40 Found 2 device=>>
18:33:40 FPGA BASE DAQ With High Speed USB A
18:33:40 FPGA BASE DAQ With High Speed USB B
18:33:43 Connected to USB Port
18:34:02 Send 240 Complete
18:34:08 Send 170 Complete
```

Figure 9 **Figure 9** The sample displays in the application program

TABLE I. PROPERTY COMPARISON FOR INDIVIDUAL PROTOCOL

Items	**Parallel**	**SPI**	**I^{2}C**	**1-wire**
Sample rate (ksp/s)	667	100	11.1	0.00133
Sample rate (b/s)	5.34×10^6	1.20×10^6	8.88×10^3	21.28
Required Wires	12	3	2	1
Acquired solution	Easy	Easy	Complex	Complex
Power (mW)	75	2	5	7.5

We can see that the parallel protocol is the fastest whereas it used the most connection wires. In the other hand, the I^{2}C or one-wire protocol requires the number of the signal wires for the connection only two and one, respectively, whereas the bit rate is much slower than the parallel ADC. We may use this for our consideration in using the protocol in most applications.

Therefore, we could claim that our data acquisition system is one of the useful solutions for the data acquisition. Having contained the input channels with all possible ADC protocols, our system can interface to the ADC input devices with any protocols to the PC independently. In addition, with an enormous number of the I/O ports in the FPGA, it is feasible to add more channels in the future. Eventually, we could utilize the maximum of 102 I/O ports as the maximum number for this FPGA.

REFERENCES

[1] Ziad Salem, Ismail Al Kamal, Alaa Al Bashar "*A Novel Design of an Industrial Data Acquisition System*", Proc. of Int. Conf. on Inf. and Comm. Tech, ICTTA 2006, pp. 2589-2594.

[2] Jorge Yáñez, David Quintana, Camilo Quintáns, José Fariña, Juan J. Rodríguez-Andina,"*FPGA-based system for the education in data acquisition and signal generation*", Proc. of Ind. Elec. Soc. Conf., IECON 2005. pp. 2168-2173.

[3] A. Sagahyroon ,T. Al-khudairi, "*FPGA Based Acquisition of Sensor Data*" Proc. of Int. Conf. on Ind. Tech., ICIT 2004. pp. 1398-1401.

[4] M. Popa, M. Marcu, A. S. Popa,"*A Microcontroller Based Data Acquisition System with USB Interface*", Proc. of Int. Conf. on Elec., Elec. and Comp. Eng., ICEEC 2004 pp. 206-209.

[5] FTDI Corporation., www.ftdichip.com, Glasgow, UK,

[6] Hamblen J., Furman M., "*Rapid Prototyping of Digital Systems*", Quartus II Edition, Springer Science Business Media, New York, 2006

[7] Douglas L. Perry "*VHDL Programming By Example*", 4 ed., McGraw-Hill, USA, 2002.

[8] Pong P. Chu "*RTL Hardware Design Using VHDL*", 1 ed., John Wiley & Sons, Hoboken, 2006.

[9] George Shepherd, David Kruglinski "*Programming with Microsoft Visual C++ .NET*", 6 ed., Microsoft Press, Washington, 2003.