FPGA configuration on Xilinx ML506 Development Board Through the USB port in C/HDL

Kushal Sreedhar
Department of Electronics and Communication Engineering
RNS Institute of Technology, Channasandra, Bangalore 560061
kushalsathreya@gmail.com

***Abstract*- This paper gives a novel yet convenient technique of configuring a Virtex-5 FPGA device through the Universal Serial Bus(USB) port. The monotonous parallel port configuration using JTAG connectors is overcome by USB-port programming of the FPGA either in C or HDL (Hardware Description Language). A Xilinx Development platform (ML506 Evaluation platform is considered) consisting of the Field Programmable Gate Array (FPGA) populated with an SXT device is initially configured to henceforth being reconfigurable through the USB on reset/power up. The encrypted configuration bit stream arriving at the USB port is first accessed by the on-chip USB Controller operating preferably on a standalone mode. The data is then loaded on to the Type-I Compact Flash (CF) storage device (expandable to 8GB) through the System ACE controller. The System ACE MPU port is connected to the FPGA which allows the System ACE Controller to access the Compact Flash Card as a generic FAT File system. The FPGA is finally configured either in Serial/Select MAP modes through the dedicated pins. The reason why USB method is more beneficial is that it is more versatile, and doesn't require JTAG connectors which are scanty. Also, in areas where FPGA programming is done more frequently, USB method eliminates the process of disconnecting and reconnecting the subsequent FPGA boards, since the USB cable can be permanently connected with the respective FPGAs. It is faster to program FPGAs in bulk, also cheaper as the connectors are more costly than the USB cable.**

***Keywords*- Programmable Gate Array, Compact Flash, System ACE controller, USB host controller, Configurable Logic Block, JTAG connector, EEPROM.**

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are widely used for rapid prototyping and manufacturing digital systems using ASIC design [1, 2]. FPGAs are available commercially and generally exist in two types: SRAM-based and segmented channel FPGAs [3, 4]. Research in FPGAs has encompassed many aspects such as technology mapping, routing and placement [2, 8].

Figure 1. shows the FPGA structure which is built up of individual blocks like memory, logic, input/output blocks etc., woven in a mesh of interconnecting wires. The various blocks are connected to the wire mesh through switches. Different wires in the mesh are also connected by switches. In modern FPGAs, SRAMS or antifuses are used for making the switches as they are based on the current CMOS technology.

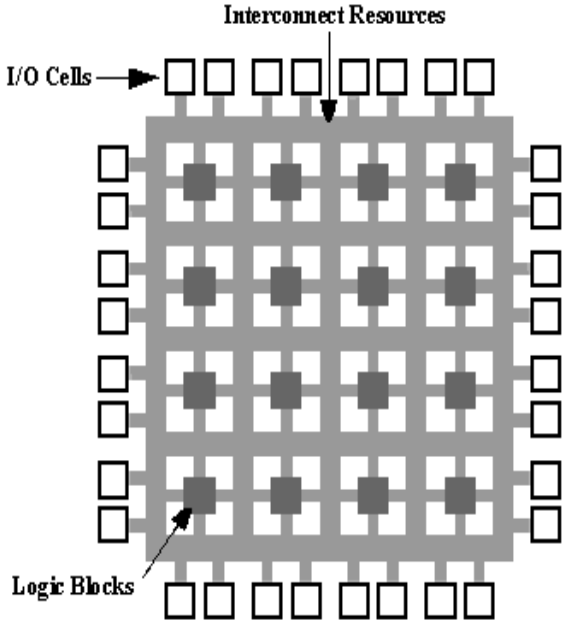

Figure 1. Basic FPGA Structure

In the early 1990's, interest in yield and fault tolerance by reconfiguration has been repeated in technical literature [5, 6, 7, 11, 12].

Further, the FPGA is usually configured using a parallel port connection which incorporates a JTAG connector which is already provided by the manufacturer. These JTAG connectors are quite scanty and their usage can be avoided in applications where the JTAG port is not accessible for providing external connections.

The objective of this paper is to propose a new methodology for configuring and reconfiguring the FPGA. Configuration of the FPGA is very essential to rendering it functional. This paper adopts a methodology which is based on utilizing the embedded USB host controller CY7C67300 provided by CYPRESS [13]. The FPGA used is a Virtex-5 LXT device. It is mounted on the ML506 evaluation platform populated by an SXT device [15, 16, 17].

This paper is organized as follows. Section II introduces the Virtex-5 FPGA and the ML506 development board in detail. Section III describes the different modes of configuration made available by the manufacturer on the development board of interest. Section IV deals with the analysis and it is followed by Section V which throws light on the configuration methodology. Conclusions inclusive of the future enhancement are presented in the last section.

II. OVERVIEW AND BACKGROUND

A. Virtex-5 FPGA

Virtex-5 devices are user-programmable gate arrays with various configurable elements and embedded cores optimized for high-performance high-density system designs. The basic Virtex-5 logic element, illustrated in Fig. 2, is composed of a 6-input look-up table (LUT), a configurable flip-flop/latch, and multiplexers to control the combinational logic output and the registered output (flip-flop/latch input). Additional dedicated fast carry logic is included to perform special logic and arithmetic functions. In some slices, the LUT can be configured as a small RAM, called a distributed RAM or LUT RAM, or as a shift register [9]. Four such basic logic elements are grouped to form a slice, and two slices are grouped to form a complete CLB. has shown in Fig. 3 [9]. Each CLB is connected by a switch matrix to local and global programmable routing resources. The Block Ram Modules provide 36 Kbit true dual port RAM. These modules are cascadable to form larger memory blocks. CMT (Clock Management Tiles) provide flexible clocking resources for the FPGA. These blocks contain two DCM (Digital Clock Manager) and one PLL block for clock distribution, delay compensation,clock multiplication/division, coarse/fine clock phase shifting, input clock jitter filtering [12].

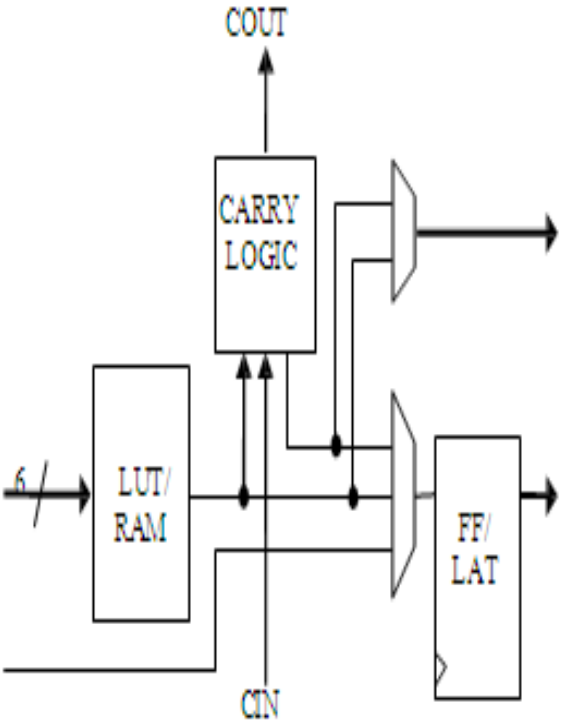

Figure 2. Simplified basic logic element [10]

B. ML506 Development Board

The Xilinx ML506 development Board [15] comes with the Virtex-5 SXT FPGA which contains extra DSP4E slices, and can provide high transfer rates/speed-performance on the Ethernet line. There are numerous ways of configuring the device, some of which have not been fully developed yet. One

such way is using the USB port for transfer of configuration file to th FPGA. On ML506 board, this seems possible if we carefully look at the way in which the System ACE controller and the USB controller are interfaced. This has been described in the following text.

B.1. System ACE Controller

This chip was developed by Xilinx for convenience in configuring their FPGAs. It is also a space-efficient, pre-engineered configuration solution for multiple FPGAs. It uses Type-I Compact Flash storage device (expandable to 8 GB) or a Microprocessor Interface (MPU) for gathering configuration data. It contains a built-in JTAG controller and a JTAG test scanner for configuring the FPGA through JTAG interface. Figure 3 shows the System ACE chip block diagram. On ML506 board, a Compact Flash port is provided for configuration. Additionally, the System ACE controller also shares data and address lines with the Cypress USB controller CY7C67300 [13].

B.2. USB Controller

This chip is present on ML506, and is used to control the peripheral USB ports [18]. It is manufactured by Cypress [13] for standalone USB devices, with or without a separate microcontroller, which have recently flooded the electronics goods market. These devices include USB pen drives, mice and keyboards, external hard drives, bluetooth devices, wireless devices, USB modems etc.

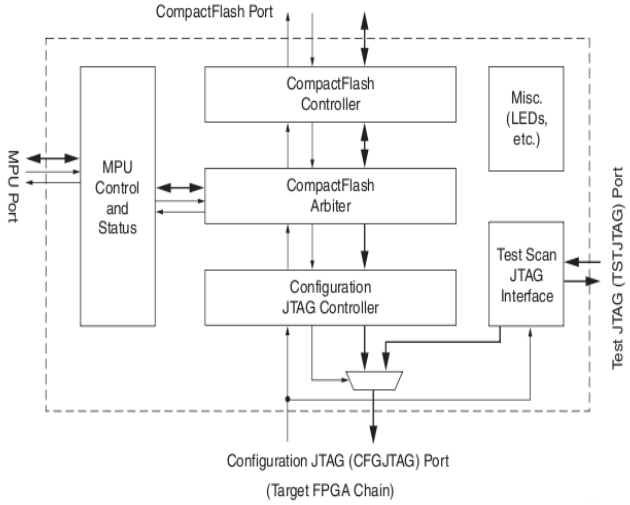

Figure 3. System ACE Block Diagram

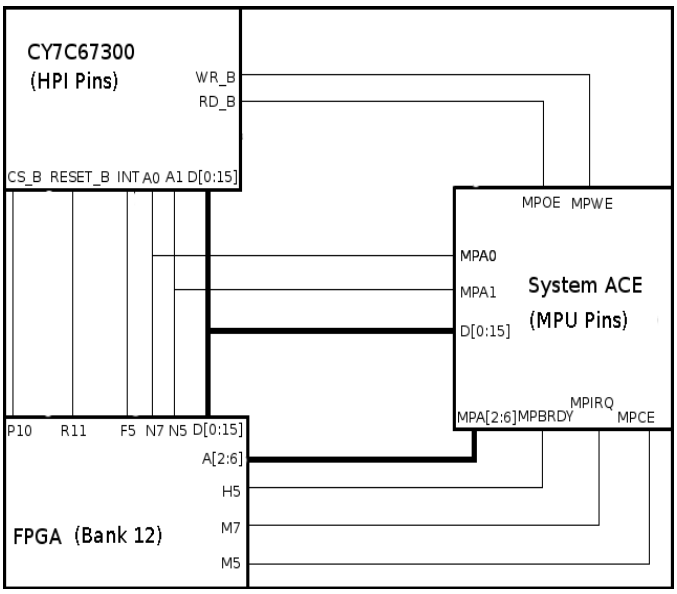

Figure 4. System Interconnects

It contains a RISC CPU inside it for standalone operation, or to reduce load on the external CPU.

It can be operated in two prominent modes: Coprocessor mode and the Standalone mode. In the former, an external CPU gives high level commands to the internal RISC CPU. The interfaces can be a 16-bit parallel Host Port Interface (HPI), a High Speed Serial Interface (HSSI) with about 2M baud rate or a Serial Peripheral Interface (SPI), slave mode (yielding upto 2 Mbps transfer rate). In the latter, a firmware has to be developed for the RISC CPU. The firmware is stored in a 8/16 Kb I^2C EEPROM, and the EEPROM is connected to GPIO [30:31] pins [14] of the controller. In the following Analysis, it is evident that USB port on the ML506 can be used for the configuration of the Virtex-5 FPGA.

III. CONFIGURATION MODES

The Virtex 5 FPGA is configured by loading the cofiguration file (the bit stream) into the internal memory of the FPGA. The internal memory is volatile, thus it has to be configured each time on power up. The memory is in turn connected to the antifuses inside the FPGA, and thus stores the connection information for the various wires, logic and memory blocks, etc. Configuration modes [16] available are:

1. Serial Configuration mode

When FPGA is in master mode, the clock is provided by the FPGA on CCLK pin. This mode can be used for configuration using Platform Flash PROM.

The slave mode is generally used while configuring multiple FPGAs serially on a daisy chain. The clock is provided externally.

2. Select MAP Configuration
This interface provides a bidirectional 8/16/32 bit data bus to the configuration logic. The data bus can be used for configuration and verification of the bit stream loaded to the configuration logic. In master mode, the FPGA CCLK pin provides the clock, while in slave mode the CCLK pin acts as an input pin for the external clock. This interface can be used for configuration of a single or many devices together (parallel daisy chain).
3. JTAG/Boundary Scan Configuration
The JTAG standard provides a means to ensure the board-level integrity of the components and the interconnections between them. It was developed by 'Joint Test Action Group' and is named after the committee. This standard provides means of transferring data and an integrity check called boundary scan.
4. Serial Peripheral Interface (SPI) Flash Configuration
In this mode, the FPGA configures itself from an industry-standard attached SPI serial flash PROM.
5. *Byte Peripheral Interface (BPI-up/down) Configuration*
This mode is used for configuration through an industry-standard parallel NOR flash PROM. The FPGA drives 26 address lines to acquire data from the parallel flash.

IV. ANALYSIS

The schematics provided by Xilinx for ML506 board are given in this section. Figure 4. shows how the System ACE, USB Controller and the FPGA are connected to each other.
CY7C67300 USB controller is perhaps configured to work in the HPI mode, which can be deduced from the way the pins are connected. The WR_B and RD_B signals are also connected to the FPGA pins R9 and N8 respectively. In HPI mode, the controller will send an interrupt on the INT pin as soon as it gets any data from the USB. The CY7C67300 data sheet [13] can be referred for read cycle timings and other details of HPI mode.

The System ACE controller can be used in 2 modes consecutively in two steps to successfully transfer the data from the USB port to the FPGA. In the first step, it is put in 'MPU to Compact Flash' mode, wherein the data is first stored in the Compact Flash. This will help, because the data will be available every time the FPGA is restarted, provided the configuring is done through System ACE. In the second step, it is put in 'Compact Flash to JTAG' mode and reset. This will automatically transfer the data from Compact Flash to FPGA.

The control register in System ACE can be changed through the MPU port. The 3 bits in this register that decide which ace file is to be loaded are CFGAD-DRBIT[0:2]. Refer System ACE data sheet for more details [14].
A program should be running on the PC connected through USB to ML506, listening to identification packets. This program should be started before giving the configuration trigger to the FPGA. The program for configuring the FPGA can be written either using C or a suitable Hardware Description Language (HDL).

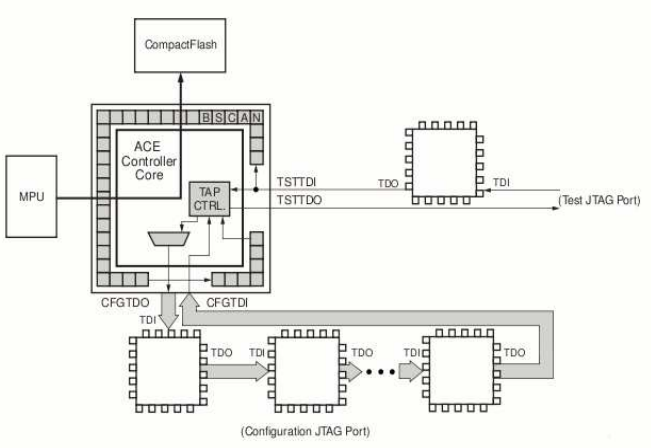

Figure 5. USB to Compact Flash

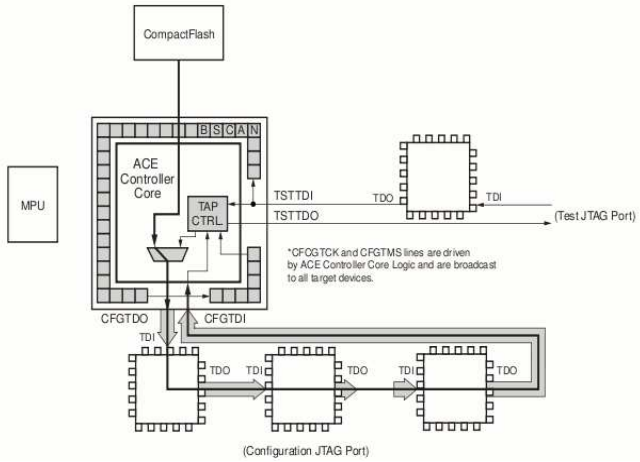

Figure 6. Compact Flash to JTAG chain

V. CONFIGURATION METHODOLOGY

After an in-depth analysis, we can now arrive at a step-by-step procedure to successfully configure the FPGA through the USB port. They are as given under:
1. A trigger is generated, which tells the program already running on the FPGA to load the configuration program stored in Compact Flash. The MPU control register on the System ACE is set so that the second ace file in CF is loaded (which is the configuring program). Then a soft reset is given to the System ACE, which automatically configures the FPGA with the configuring program.
2. The USB controller chip is enabled, and some identifying packets are sent through the USB port (data preceded by a start packet). These packets are caught by the program running on the PC, and it in turn starts sending the configuration bit stream through word-long packets.
3. When a word-long packet is received at the USB port, the controller generates an interrupt, and puts the packet on the

HPI data port.

4. The FPGA checks for the interrupt by USB controller, and generates the remaining five address bits needed by the System ACE MPU buffer. This buffer is located at the addresses 0x20 - 0x3F and can be accessed through the register DATABUFREG [0:15], 16 bits at a time. Each time the buffer gets full, MPBRDY pin on System ACE goes high. When this happens, WR_B pin on the USB controller should be set high by the FPGA to stop data acquisition. When the buffer on System ACE is ready, MPBRDY again goes low, and WR_B also has to be set low to start data acquisition. This process is described in Figure 6 [14].

5. Steps 3-4 are repeated till all the data is stored. When this happens, the stop packet will be received to the USB port, sent by the PC. This should be followed by setting of the control register on System ACE [14].

6. A soft reset should be given to System ACE again for loading the newly acquired program. When this is done the FPGA is automatically configured through JTAG chain as seen in Figure 6.

The entire configuration process is depicted in the flow chart given in the Figure 7.

VI. CONCLUSION

In this paper, an innovative approach for configuring the Virtex-5 FPGA is discussed. Here, the USB port is utilized effectively in loading the encrypted data bit stream into the Compact Flash card which configures the FPGA on reset/power on. This technique can be beneficial in areas where the FPGAs are required to be reprogrammable more frequently.

One of the most significant advantages of opting for such a technique is its ability to support bulk programming. As more USB ports are available when compared to the parallel ports on the microprocessor computer system, we have the option of programming multiple FPGA boards at the same time. For example, if there is a Fourier Spectrometer which consists of several identical sections, each containing an FPGA required to perform similar operation, then there are avenues open to program all the FPGAs simultaneously. This is indeed a cost effective solution in terms of time. It is faster to program the FPGAs in bulk, also cheaper as the connectors are more costlier than the USB cables. Also, it eliminates the cumbersome process of disconnecting and reconnecting the successive FPGA boards, since the USB cable can be permanently connected to the respective FPGA.

According to the current trend, there will be an increased support for the USB controllers on future FPGA development boards as these ports are versatile and faster. Therefore this methodology should be adopted in configuring the FPGA which also envisages a realistic enhancement in the performance of the future development boards.

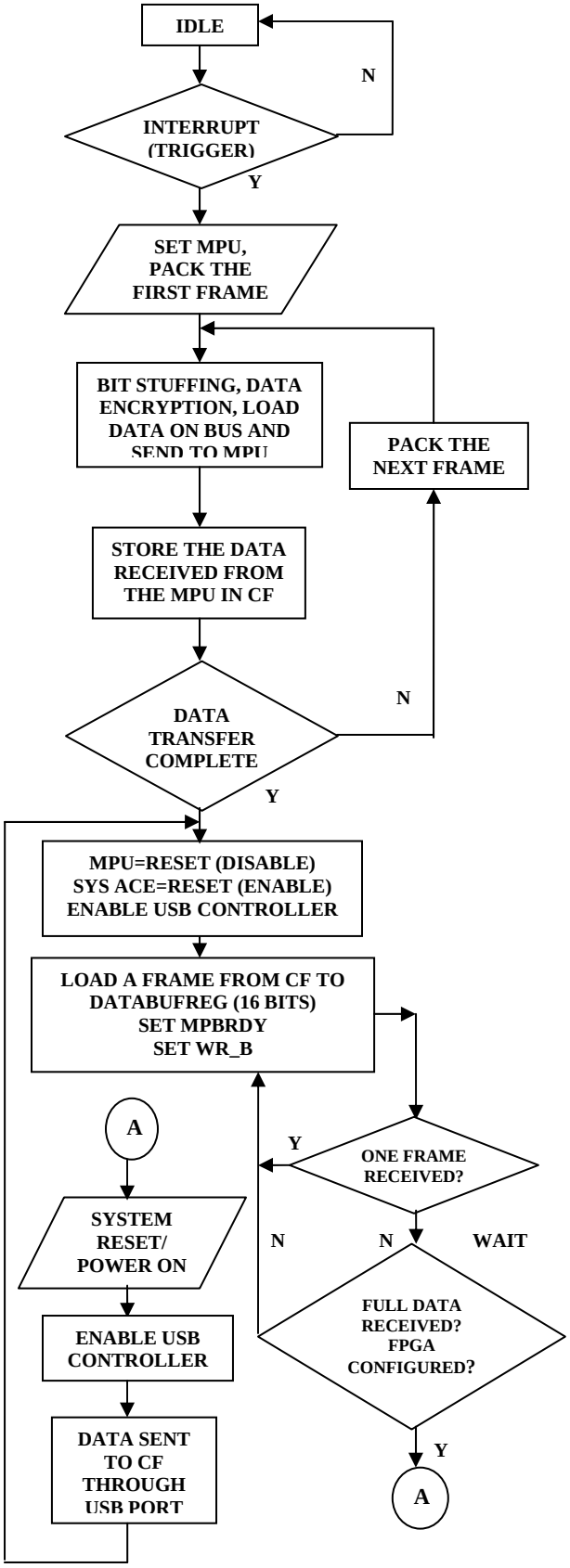

Figure 8 Flow Chart

ACKNOWLEDGMENT

The author would like to thank the reviewers for their insightful suggestions to make this paper better. In particular, many thanks to Harshad Sahasrabudhe of BITS, Pilani for his valuable inputs regarding the USB port programming and Dr. A A Deshpande of Raman Research Institute for his guidance and support.

REFERENCES

[1] Chan, P.K. and M.D.F. Schlag, "Architectural Tradeoffs in FPGA-based Conputing Systems," *Proc. IEEE Workshop for Custon Comp. Machines*, pp. 152-161, 1993.

[2] Brown S., R.J. Francis, J. Rose and Z.G. Vranesic, "Field Programmable Gate Arrays," Kluwer Academic Publishers, Boston, Mass. 1992.

[3] Actel Corporation, FPGA Data Book and Design Guide, Sunnyvale, 1994.

[4] Xilinx Inc., Programmable Gate Array Data Book, San Jose, 1991.

[5] Hatpori, F., et all, "Introducing Redundancy in FPGAs," *Proc. IEEE CICC*, pp. 7.1, 1993.

[6] Kelly, J.L and P.A. Ivey, "A Novel Approach to Defect tolerant Design for SRAM based FPGAs," *Proc. ACM 2nd Int. Work. On FPGAs*, Berkeley, 1994.

[7] Narasimhan, J., K. Nakajima, C.S Rim and A.T Dahbura, "Yield Enhancement of Programmable ASIC arrays by Reconfiguration of Circuit Placements," *IEEE Trans on CAD of ICAS, Vol. CAD13, No. 8*, pp. 976-986, 1994.

[8] Green, J., V. Rowchowdury, K. Kaptanoglu and A. El Gamal, "Segmented Channel Routing," *Proc 27th IEEE/ACM DAC*, pp. 567-572, 1990.

[9] Bradley F. Dutton and Charles E. Stroud, "Built-In Self-Test of Configurable Logic Blocks in Virtex-5 FPGAs," *Proc. 41st IEEE SSST*, University of Tennessee, 2009.

[10] *Virtex-5 FPGA User Guide*, UG190 (v 4.2), Xilinx Inc., San Jose, CA, May 2008. Avaliable: *www.xilinx.com.*

[11] Fawcett, B.K., "Taking Advantage of Reconfigurable Logic," *Proc. ACM 2nd Int. Work. On FPGAs*, Berkeley, 1994.

[12] Durand, S. and C. Piguet, "FPGA with Self-repair Capabilities," *Proc. ACM 2nd Int. Work. On FPGAs*, Berkeley, 1994.

[13] Cypress, CY7C67300 Data Sheet, *www.cypress.com.*

[14] Xilinx Inc., System ACE CompactFlash Solution Data Sheet v1.4, *www.xilinx.com.*

[15] Xilinx Inc., UG437, ML505/ML506/ML507 Evalaution Platform, User Guide.

[16] Xilinx Inc., UG191, Virtex-5 Configuration User Guide.

[17] Xilinx Inc., DS100, Virtex-5 Family Overview.

[18] Cherng-Ying Ing and Tzao-Lin Lee, "USB Device Sharing Server for Office Environment," *Proc. IEEE APSCC*, 2008.