FPGA-Based Flash Memory Controller for BZK.SAU.FPGA10.1 Microcomputer Architecture Design as an Educational Tool

H. Oztekin, F. Temurtas, E. Olmez, and A. Gulbag, *Member, IACSIT*

***Abstract*—Operating systems and computer architecture and organization course are fundamental topics underlying many disciplines, including computer, electrical and electronics engineering departments. These courses involve both a theoretical and practical part for the effective learning process. A FPGA-based micro computer architecture design named BZK.SAU.FPGA10.1 was proposed to reinforce computer architecture and organization fundamentals in 2011. Also we developed an educational operating system named BZK.SAUOS from scratch on BZK.SAU.FPGA10.1 architecture. The program written using the user interface of this operating system was saved to main memory of this microcomputer architecture since it does not have a nonvolatile memory unit. So the process of editing on a file previously written is not impossible. This paper addresses an effective learning approach that permits one to work on real computer architecture and original operating system while supporting easily to save and edit their program using flash memory instead of main memory. The flash memory controller in this work have been designed completely hardware. So this controller allows the students to examine the internal structure of this controller while improving the motivation in the control of storage units like flash memory, hard disk etc. Overall, this work helps the students to improve controller design experience of storage units in low-level in addition to an increase in their learning process in the computer architecture and operating systems topics.**

***Index Terms*—BZK.SAU.FPGA, educational tool, memory controller, microcomputer architecture design, operating system.**

I. Introduction

Operating systems and Computer architecture and organization courses are some of the main courses in the computer engineering and computer science. The effective learning process in these courses involves both a theoretical and practical part. There are several of designs and simulators in the open literature to improve the motivation in these courses. We also developed two FPGA-based microcomputer architecture designs named BZK.SAU.FPGA10.0 and BZK.SAU.FPGA10.1 in addition to literature. Then we designed an educational operating system named BZK.SAUOS on our microcomputer architecture from scratch. To save a program written using the user interface of our operating system was used the main memory of our microcomputer architecture since it does not have a nonvolatile memory. In other words it does not have a controller of flash memory with 8MB on Altera De2-70 FPGA development board that we used. So the editing process on a file previously written was impossible. We developed a flash memory controller to save a file and edit the saved file for this reason. This controller can be only done 3 processes on flash memory for now: reading, sector-based writing and chip erasing. It is also completely designed at logic gate level in order to examine all units of this controller in detail since our microcomputer architecture and operating system designs have an educational nature.

This controller can be realized using FPGA (field programmable gate array) development board, which is the popular technology widely, with a low cost and high flexibility. FPGAs offer a design platform that allows students to implement more meaningful projects with tens of thousands of gates on actual hardware [1-2].

The main goal in designing of this controller is to facilitate saving a program written using the user interface of BZK.SAUOS operating system on nonvolatile storage environment and hence the ability to allow for editing the saved file. The other goal is to improve controller design experience of storage at logic gate level.

This paper introduces a flash memory controller design for BZK.SAU.FPGA10.1 microcomputer architecture and BZK.SAUOS operating system. The rest of this paper is organized as followed. Section 2 presents the general information about BZK.SAU.FPGA10.1 and BZK.SAUOS system designs. Section 3 introduces the proposed approach to flash memory controller design while section 5 concludes some achievement of this work.

II. Related Works

Tang Lei and Zhou Xuan [3] designed a Nand Flash controller for fpga application. Controller has its own instruction set to manage a flash memory chip. Users can operate the controller without caring about the strict timing sequences of the memory chip. Lin and Dung [4] designed a NAND flash memory controller for SD/MMC flash memory card. They designed a t-EC w-bit parallel BCH ECC code for correcting the random bit errors of the flash memory chip. And they presented a code-banking mechanism to support various kinds of NAND flash memory. Jin Hyuk Yoon [5]

Manuscript received April 30, 2012; revised June 10, 2012. This work was supported in The Scientific and Technological Research Council of Turkey (TUBITAK) project no. 110E069.

Oztekin Halit and Temurtas Feyzullah are with the Department of Electrical and Electronics at Bozok University, Turkey (e-mail: oztekinhalit@ gmail.com, temurtas@gmail.com).

Olmez Emre is with the Department of Mechatronical Engineering at Bozok University, Turkey (e-mail: olmezemre@gmail.com).

Gulbag Ali is with the Department of Computer Engineering at Sakarya University, Turkey (e-mail: agulbag@sakarya.edu.tr).

proposed a high-performance Flash/FRAM hybrid SSD architecture that cures the inefficiency of flash memory in handling small random writes by using a small amount of FRAM. Seong [6] presented a new flash memory SSD architecture called Hydra that exploits multichip parallelism effectively. And the other controller O3 [7] executes multiple flash operations out of order.

III. BZK.SAU.FPGA10.1 and BZK.SAUOS System Architecture

If The BZK.SAU.FPGA10.1 is the new version of BZK.SAU.FPGA10.0 [8]. BZK and SAU words are the acronym of Bozok and Sakarya University. We have adopted the modular approach to the second FPGA version of the BZK.SAU [9] named BZK.SAU.FPGA10.1 [10]. [8] and [9] have the same architecture. The only difference between them the development environments. While the development environment of [8] is FPGA, other is an emulator program.

Modular design approach is an important factor for the educational training of topics in computer architecture and organization course since this approach allows to students to be able to integrate seamlessly into existing system. In this approach, the learning process takes place in stages since system is divided into parts. We took the approach of modularization in order to avoid having students be overwhelmed by the complexity of a complete computer system design. So the students do not have to develop a microcomputer architecture design from scratch. It is quite simple to include their own designs like adding unit, subtraction unit etc. to our system in this approach. So they can see the operation of their own designs on our system without affecting the operation of our system. In other words, our modular approach is plug and see. Therefore, teachers can teach more productive course by applying this approach to their teaching methods since our approach aims to learn piece by piece rather than complete system. All units in BZK.SAU.FPGA10.1 microcomputer architecture are our own specific designs and we built these units using only schematic design at logic level. A detailed block diagram that shows the components of the BZK.SAU.FPGA10.1 design is shown in Fig. 1.

The common features of both versions are listed as the following:

1) Support six addressing modes: immediate, direct, indirect, index, relative and inherent mode.
2) They have eight general and special registers: Address Register(AR), Data Register(DR), Accumulator(AC), Program Counter(PC), Stack Pointer(SP), Index Register(IX) and Temporary Register(TR) are 16-bit width while Instruction Register(IR), Output Register(OUTR) and Input Register(INPR) are 8-bit width.

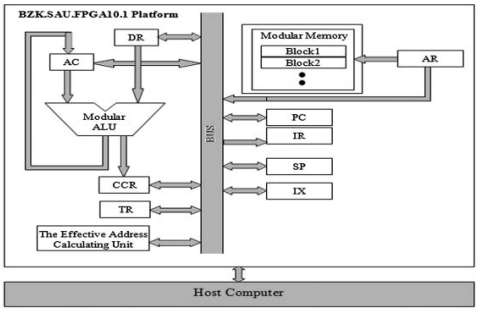

Fig. 1. The block diagram of BZK.SAU.FPGA10.1[10].

3) 16-bit bus for data and address information.
4) The effective address calculating unit for instructions which have relative and index addressing modes
5) Their instruction set consists of 51 instructions: memory and accumulator-based 21 instructions, 8 instructions on index and stack registers, 22 instructions to change the flow direct of program execution.
6) 64 KB main memory for running programs and 8 MB flash memory for storing files.
7) The output pixel size of display screen on VGA monitor is 320x384 pixels and screen area has 40 columns x 24 rows since every character is 8x16 pixels.[11]

Instructions occupy one or two bytes according to addressing modes. While instructions with inherent addressing mode occupy one byte in the main memory, other instructions occupy two bytes. In order to provide readers with the detailed features of instructions and instruction structures, more data are given [8-10].

BZK.SAUOS is an educational operating system for BZK.SAU.FPGA10.1 microcomputer architecture design. It has the two operation mode, system and user modes, and "Esc" key in the keyboard is used to switch between them. OS is initially at system mode and the user interface like MS-DOS command prompt is active. This user interface is shown in Fig. 2. BZK.SAUOS implements four commands at command line. These are given in Table I.

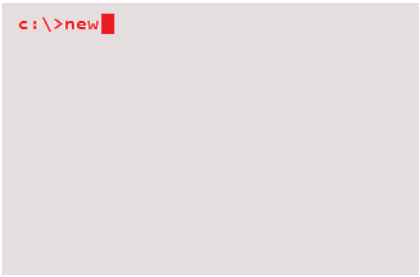

Fig. 2. The command prompt of BZK.SAUOS.

TABLE I: The Command List In BZK.SAUOS.

Name	Description	Writing format
new	Creates a file	new
save	Save the current file	save filename.ext
edit	Open to edit a file	edit filename.ext
run	Compileandrun the current file	run

IV. The Proposed Approach: Flash Memory Controller

Flash memory is a non-volatile memory that can be electrically erased and reprogrammed. It was developed from EEPROM (electrically erasable programmable read-only memory). It is used in mobile devices and other digital products because of low power consumption, fast random access and high shock resistance.

Flash memory must be erased in large sectors before these can be rewritten data. The difference between EEPROM and FLASH Memory is EEPROM erasable in small blocks, typically bytes, but the flash memory is erasable in large block sizes. The large block sizes give FLASH Memory a significant speed advantage over old-style EEPROM when writing large amounts of data.

Flash memory now costs far less than byte-programmable EEPROM and has become the dominant memory type wherever a significant amount of non-volatile, solid state storage is needed.

The most popular Flash types are NOR and NAND types. The high density NAND type must also be programmed and read in (smaller) blocks, or pages, while the NOR type allows a single machine word (byte) to be written or read independently. NAND Flash memory is used for storage applications. NOR Flash has long erase and write times, but provides full address and data buses and allowing random access to any memory location. This makes it particularly suitable for code storage which requires high-speed random access.

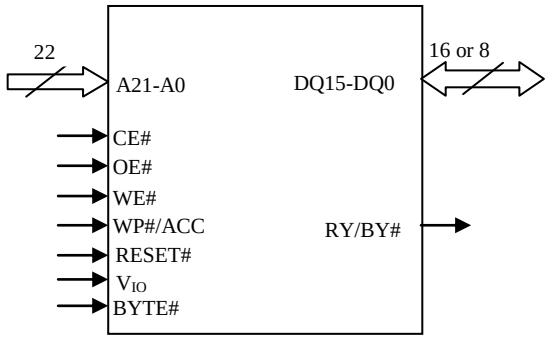

Fig. 3. The logic symbol of flash memory on Altera DE2-70 FPGA development board [12].

TABLE II: The PIN Descriptions of Flash Memory

Pin	Description
A21-A0	22 Address Input
DQ15-DQ0	15 Data Input/Output
CE#	Chip Enable Input
OE#	Output Enable Input
WE#	Write Enable Input
WP#/ACC	Hardware Write Protect/Programming Acceleration Input
ACC	Acceleration Input
WP#	Hardware Write Protect Input
RESET#	Hardware Reset Pin Input
RY/BY#	Ready/Busy Output
BYTE#	Select 8-bit or 16-bit mode
V_{IO}	Output Buffer Power

In this study we use SPANSION S29GL064N NOR flash, that is 8 Mb on Altera DE2-70 FPGA development board. The instruction set of flash memory consists of 21 commands. But we used only 3 commands of it: reading, programming and chip erasing for now. Flash Memory has an internal command register. The command register itself does not occupy any addressable memory location. The register is a latch used to store the commands, along with the address and data information needed to execute the command. Flash memory has 22 address pins and 16 data pins. Pin descriptions and the logic symbol of its shown in Fig. 3 and Table II respectively.

A. Configuration and Programming of Flash Memory

The BYTE# pin controls whether the device data I/O pins operate in the byte or word configuration. If the BYTE# pin is set at logic 1, the device is in word configuration, DQ0–DQ15 is active and controlled by CE# and OE#. If the BYTE# pin is set at logic 0, the device is in byte configuration, and only data I/O pins DQ0–DQ7 are active and controlled by CE# and OE#. The data I/O pins DQ8–DQ14 are tri-stated, and the DQ15 pin is used as an input for the LSB (A-1) address function.

The device is automatically set to reading data after device power-up. No commands are required to retrieve data. To read data from the outputs, the system must drive the CE# and OE# pins to Logic Low. CE# is the power control and selects the device. OE# is the output control and gates array data to the output pins. WE# should remain at Logic High. Table III summarizes this situation. After setting the situation shown in table for read, we can instantly read data that corresponds to the address we entered from address inputs.

Writing specific data to specific address by defined sequences into the command register initiates device operations. Table IV defines the valid register command sequences of read, program and chip erase commands. Read command is one-cycle command, but the other commands we use program and chip erase commands are multi cycle commands.

TABLE III: The Commands Used in This Study[12].

Operation	CE#	OE#	WE#	RESET#
Read	L	L	H	H
Write(Program/Erase)	L	H	L	H
Reset	X	X	X	L

For writing data to flash memory we should process program command consists of four cycle shown in Table IV. After processing the first three cycle the device is ready for writing data. In the fourth cycle we write data that we want to corresponding address. The other command we use, chip erase is six-cycle command. For erasing the whole memory we should implement the required command sequence of chip erase command. After the sixth cycle chip erase operation begins and it takes approximately a few minutes. The main information we need for processing a command is how to process a cycle. Processing a cycle operation is as follows; firstly corresponding address and data information attach to the data and address pins of the flash memory. Secondly latch the address and data. All addresses are latched on the falling edge of WE# or CE#, whichever happens later. All data is latched on the rising edge of WE# or CE#, whichever happens first. In our design we keep CE# always at logic 0. We latch the address and data by switching the WE#. So the addresses are latched on the falling edge of WE# and data is latched on

the rising edge of WE#.

Writing incorrect address and data values or writing them in the improper sequence may place the device in an unknown state. A reset command is then required to return the device to reading array data.

TABLE IV: THE REQUIRED COMMANDS SEQUENCES[12].

Command Sequence	#Cycle	First		Second		Third		Fourth		Fifth		Sixth	
		Adress	Data	Adress	Data	Adress	Data	Adress	Data	Adress	Data	Adress	Data
Read	1	RA	RD										
Program	4	555	AA	2AA	55	555	A0	PA	PD				
Chip Erase	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10

PA: Program Adress, PD: Pogram Data

B. The Controller Structure of Flash Memory

This controller fundamentally consists of Mod-2 counters. The signals come from BZK.SAU.FPGA10.1 microcomputer enable these counters in this controller for the required process. The structure is shown in Fig. 4. This controller implements three commands on flash memory. These are programming(writing), chip erasing and reading. Program command consists of four cycles works as follows:

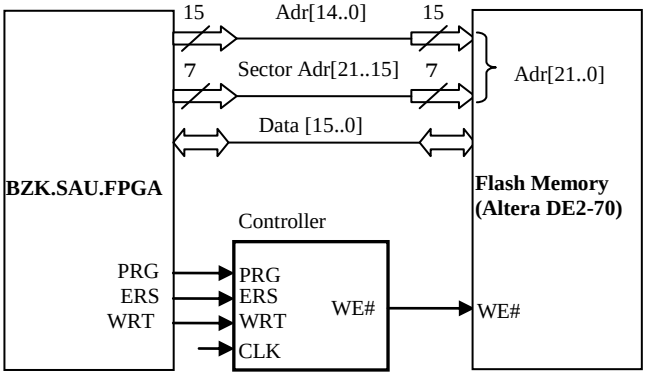

Fig. 4. The block diagram of controller between BZK.SAU.FPGA and flash memory.

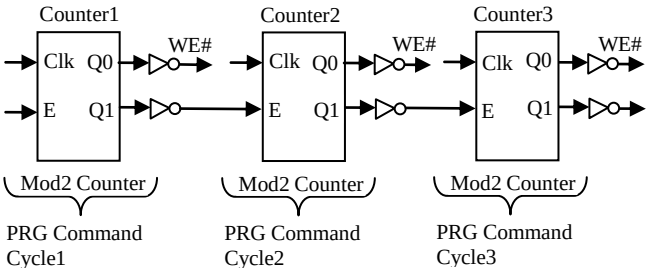

Fig. 5. The cycles of program command on flash memory.

With the falling edge of PRG input signal from BZK.SAU Counter1 starts to count from 00 to 10 (00-01-10). So the first cycle of the program command is on the process. In the first cycle 555h address information attached to the address pins and AAh data information attached to the data pins of flash memory. The valid command sequences are shown in Table IV. The least significant bit of the each counter is connected to the WE pin of the flash memory. WE pin latches the address and data which we attached to the address and data pins of the flash memory. Falling edge of WE (LSB of Counter1) latches the address and the rising edge of WE latches the data. With the rising edge of WE Counter1 reaches at 10 so first cycle of the program command is finished and the most significant bit of the counter1 triggers the counter2 so the second cycle of the program command is on the process. The second and third cycle work same as the first cycle. When the third counter finished to count flash memory is ready for writing data. In the fourth cycle we write the data into the flash memory. The WRT signal comes from BZK.SAU performs the fourth cycle of program command.. After attaching the address and data information with the falling edge of WRT signal from BZK.SAU.FPGA corresponding data is written to the corresponding address.

Counters provide the flow in the proposed design and the Least Significant Bits of the counters trigger the WE pin of Flash memory for attaching address and data. Each counter is used to implement for each command cycle. PRG command signal from BZK.SAU.FPGA triggers the counter1 then it starts to count from 00 to 10. When the counter1 reaches at 10 it stays at this value and the MSB of the counter1 triggers the counter2. Counter2 and the next counters work at the same way when the counter2 reaches at 10 it stays at this value and the MSB of the counter2 triggers the counter3. Program command process is finished with the counter3 of the program command reaches at 10 and counter3 resets all counters in the circuit, so the device is ready for another command input.

The second command is chip erasing. Chip erase command operates like as program command. Chip erasing command lasts six cycles. So we used six counters for designing chip erase command. Chip erase command starts with the falling edge of ERS signal which comes from BZK.SAU.FPGA microcomputer architecture. After the sixth cycle of the command, flash memory starts to erase the whole memory. After chip erase command, all cells in the flash memory become logic 1. The command sequence of chip erase is shown in Table IV.

V. CONCLUSION

This article presented a controller design of flash memory on Altera DE2-70 FPGA development board to save a file written using the user interface of BZK.SAUOS operating system. Thus it allows the students improving the motivation on our educational operating system since the editing process on the saved file is available. This controller is entirely realized using schematic structure at logic gate level to be able to examine the internal structure of this controller while improving the motivation in the control of storage units like flash memory, hard disk etc. Overall, this work helps the students to improve controller design experience of storage units in low-level in addition to an increase in their learning process in the computer architecture and operating systems topics. In the future work, we will improve the other commands of this controller in addition to three commands; reading, writing and chip erasing designed in this work.

REFERENCES

[1] M. A. Soderstrand, "Role of FPGA's undergraduate project courses," Proc. *IEEE Conf. Microelectronics System Education(MSE 97)*, July 1997, pp. 109-110, doi: 10.1109/MSE.1997.612570

[2] M. S. Nixon, "On a programmable approach to introducing digital design," *IEEE Trans. Educ.*, vol. 40, pp. 195-206, 1997.
[3] T. Lei, Z. Xuan, W. Yao, and L. Jincheng "Flash controller design for FPGA application," *ICEEE 2010 International conference*, pp. 1-4, Nov. 2010.
[4] C. S. Lin and L. R. Dung, "A NAND Flash memory controller for SD/MMC flash memory card," *IEEE Trans. On Magnetics*, vol. 43, no. 2, Feb. 2007.
[5] J. H. Yoon, E. H. Nam, Y. J. Seong, H. Kim, B. S. Kim, S. L. Min, and Y. Cho, "Chameleon: A High Performance Flash/FRAM Hybrid Solid State Disk Architecture," *IEEE Computer Architecture Letters*, vol. 7, pp. 17, Jan. 2008.
[6] Y. J. Seong, E. H. Nam, J. H. Yoon, H. Kim, J. Y. Choi, S. Lee, Y. H. Bae, J. Lee, Y. Cho, and S. L. Min, "Hydra: a block-mapped parallel flash memory solid-state disk architecture," *IEEE Trans. On Computers*, vol. 59, no. 7, pp. 905-921, 2010.
[7] E. H. Nam, B. S. J. Kim, H. Eom, and S. L. Min, "Ozone (O3): An Out-of-Order Flash Memory Controller Architeture," *IEEE Trans. on Computers*, vol. 60, no. 5 , pp. 653-666, May 2011.
[8] H. Oztekin, F. Temurtas, and A. Gulbag, "BZK.SAU.FPGA10.0: Microprocessor architecture design on reconfigurable hardware as an educational tool," *Int. Conf. ComputersandInformatics (ISCI 11)*, pp. 385-389, doi: 10.1109/ISCI.2011.5958946), March 2011.
[9] H. Oztekin, F. Temurtas, and A. Gulbag, "BZK.SAU: Implementing a hardware and software-based Computer Architecture simulator for educational purpose," *Int. Conf. Computer Design and Applications (ICCDA 10)*, vol. 4, pp. V4-90-V4-97, doi: 10.1109/ICCDA.2010.5541476), June 2010
[10] H. Oztekin, F. Temurtas, and A. Gulbag, "BZK.SAU.FPGA10.1: A modular approach to FPGA-based micro computer architecture design for educational purpose," *Comput. Appl. Eng. Educ.* doi: 10.1002/cae.20553, in press.
[11] H. Oztekin, F. Temurtas, and A. Gulbag, "A modular approach to VGA Monitor Controller for BZK.SAU.FPGA10.1 microcomputer architecture design," *Int. Proc. Computer Science and Information Technology (ICICA 12)*, vol. 24, pp. 27-31, February 2012.
[12] S29GL-N MirrorBit® Flash Family. [Online]. Available: http://www.spansion.com/Support/Datasheets/s29gl-n_01.pdf

Halit Oztekin received his B.Sc., M.Sc. and Ph.D degrees in Computer Engineering at Sakarya University. He has been working as a lecturer in Department of Electrical and Electronics at Bozok University. His research interests are in computer architecture, microprocessors, FPGAs, operating systems.

Feyzullah Temurtas received his B.Sc. degree in Electrical and Electronic Engineering at Middle East Technical University, M.Sc. degree in Electrical and Electronic Engineering at Dumlupinar University and Ph.D. degree in Electrical and Electronic Engineering at Sakarya University. He has substantial experience of data processing techniques for sensor arrays used in electronic nose formats, computer architecture, neural network and fuzzy logic discriminators.

Ali Gulbag received his B.Sc., M.Sc. and Ph.D. degrees in Electrical and Electronic Engineering at Sakarya University. He has been working as an assistant professor in Department of Computer Engineering at Sakarya University. His research interests are in artificial intelligence and pattern recognition, computer architecture.

Emre Olmez received his B.Sc., degree in Electrical and Electronic Engineering at Kırıkkale University. He has been studying as an M.Sc. student in department of Mechatronical Engineering at Bozok University. His research interests are microprocessors, FPGAs and automation
.