FPGA Based Micro controller for Voice Message Synthesis

Ivan Kanev

Abstract: *The paper discuses the basic results from design and analysis of Field Programmable Gate Array (FPGA) based microcontroller for Voice Message (VM) system. The conditions that the system must answer, its architecture and instruction set are defined. The whole project, called VMCore, is implemented in VHDL and is tested on Altera ACEX 1K FPGA device.*

Key words*:* ***FPGA, VHDL, VM, Microcontroller.***

INTRODUCTION

Computer voice communications is a modern, dynamically developing field with versatile applications. There is a special interest in independent systems for Voice Messages (VM) that can be used for different applications. In this paper are shown the basic results of the design and investigation of a Field Programmable Gate Array (FPGA) based microcontroller for VM synthesis.

CONCEPTUAL MODEL

Three fundamental problems have to be solved in designing Voice Message systems:

- Choosing a method for synthesizing;
- Choosing a compression method for voice primitives;
- Choosing a hardware platform for the system implementation;

The system should consume a part of the hardware resources. In that way, the unused resources could be used for system upgrade such as:

- Other applications (intelligent testers or measurement devices, Security, Information or Interactive Voice Response (IVR) systems);
- Other interfaces (RS 485, I^2C Bus, SPI, USB, TCP/IP etc.);
- The system should allow reconfiguring with a new or optimized configuration in the field.

Figure 1 shows the conceptual model of a microcontroller for synthesis of voice messages, called Voice Message Core (VMCore).

A hardware platform based on FPGA has been picked for the implementation of VMCore project. This approach allows for satisfying scalability and reconfigurability requirements.

A method based on words, phrases and sentences has been chosen for voice message synthesis. In comparison with other known methods for voice message synthesis (phoneme, diphthongs, etc.) [2], this one is limiting the dictionary of the system and is consuming a considerable part of system resources for storage of the sample voice primitives, but nevertheless it is the only method that guarantees high quality of the sent message. Management of the system, as it is in its basic version is carried out by an ASCII coded text file, through a Serial Port.

All cardinal numbers in their verbal variety [6] are synthesized by VMCore's system software.

Modified Adaptive Delta Modulation (MADM) algorithms have been chosen for compressing the voice primitives used for VM synthesis [5]. Compressed voice primitives are stored in System Flash Rom SFR). MADM algorithms and their modifications are considered to have significant errors in decompressing. The error correction is based on

algorithms [4] that use statistically defined coefficients. The initialization of the FPGA is performed by a special configuration CPLD, which after reset extracts the configuration file from the SFR and loads it into the configuration port of the FPGA. The configuration file and the decompression error-correction coefficients are stored in the SFR. The conversion of the digital voice signals into analog ones is done by an integrated double-buffered serial DAC [7]. For testing of interactive applications, a bidirectional, 8-bit parallel PORT A is also included in the base version of VMCore.

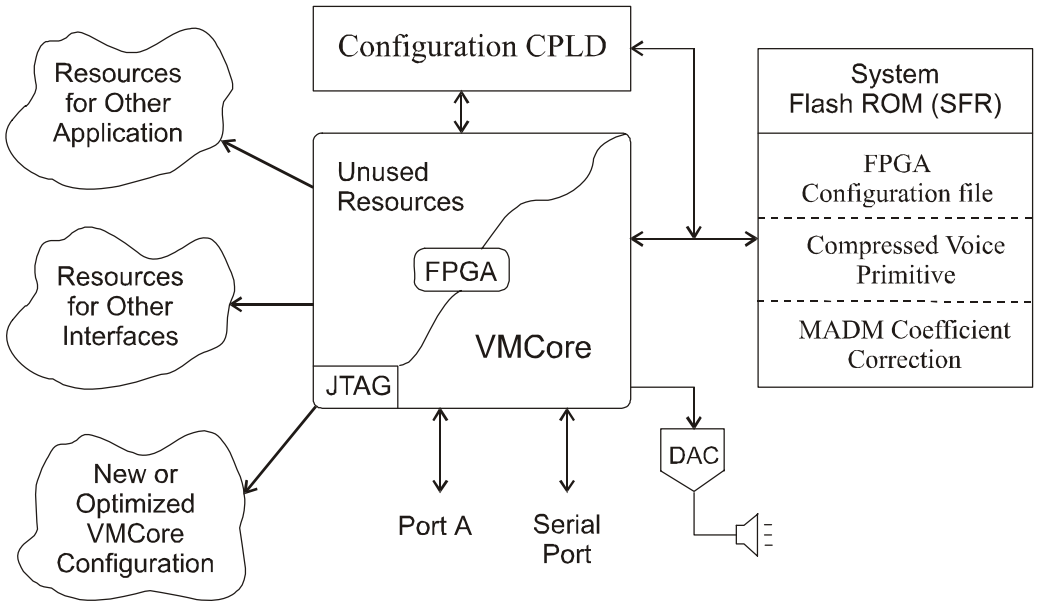

Fig 1.VMCore. Conceptual Model

ARCHITECTURAL OVERVIEW

The Architectural model of VMCore (Fig.2) is defined after analysis of resources needed for the synthesis of VM. It is based on Harvard RISC Like architecture of accumulator type [3], [8], [10], [11].

File Register (FR). All of the system, I/O and general purpose registers are combined in a single File Register (fig. 3). The system and I/O registers are synthesized from logic elements and occupy the first 16 bytes of the address space of FR. The general purpose registers are RAM-based and occupy 240 bytes of a single [1] Embedded Array Block (EAB). The remaining 256 bytes of the EAB are occupied by the System RAM. In the VMCore project, System RAM is used for:

- Storing the strings used for management of the VM synthesis.
- Synthesizing a sequence of phonetic primitives (primitive begin address, length) that will form the current message.
- Organizing stacks for storage of system registers during interrupt proceeding.

The outputs of FR registers are multiplexed by Dout_mux and they form Data OUTput (DOUT) bus.

ALU. Arithmetic Logic Unit is synthesized as a parallel combinatory logic circuit. Two registers, A and B store the input operands during one Machine Cycle (MC). The output of ALU forms Data INput bus (DIN). In the VMCore project, The ALU performs the following operations on the operands A and B:

DIN = B (pass); DIN = 0; DIN = A + B; DIN = A + B + Carry; DIN = A - B; DIN = A - B - Carry; DIN = A + 1; DIN = A – 1; DIN = A and B; DIN = A or B; DIN = A xor B; DIN = not B; DIN = rol B; DIN = ror B; DIN = A[bit I] set B[bit i]; DIN = A[bit I] clear B[bit i].

A special Skip Flag (SF): SF = 1 if B[bit i = 1] else SF = 0, has been synthesized for the implementation of the Skip operation in testing bit operands in ALU.

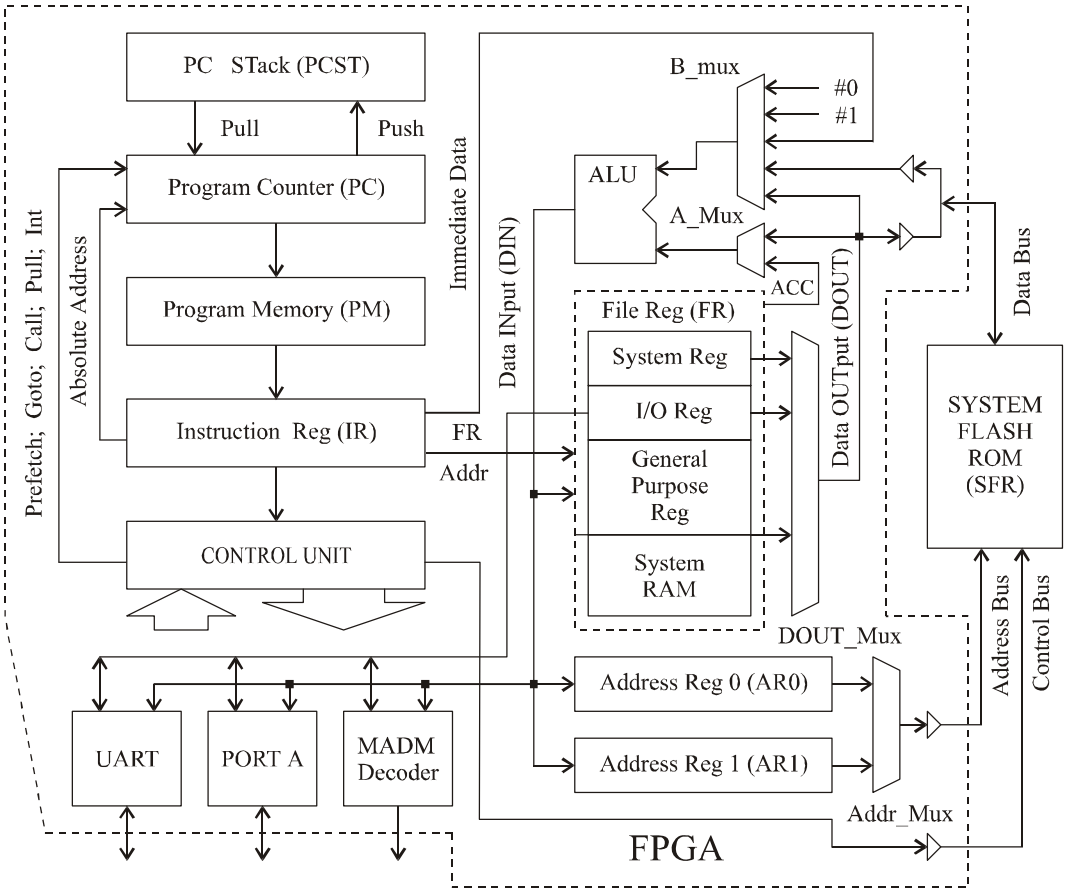

Fig 2. VMCore Block Diagram

Address Register. Two address registers: AR0 (28 bit) and AR1 (16bit) are used for addressing of SFR. AR0 is synthesized as a counter with parallel initialization and it is used for indirect access to data stored in SFR as well as for SFR's programming. AR1 is synthesized as a right shift register with parallel initialization. In the VMCore project AR1 is used [4] for addressing the SFR area, which holds the decompression error-correction coefficient. The outputs of AR0 and AR1 are multiplexed by Addr_Mux and they form (External) Address Bus.

Program Counter (PC), PC STack (PCST) and Program Memory (PM). The basic version of VMCore uses a 9 bit PC to extract the instructions from the PM. All instructions are extracted in advance. PC uses a separate stack (PCST) to store its current value when executing instructions for subprogram call (Call) or interrupt. In the base version, PCST is synthesized by means of four registers organized as a FIFO structure. This allows the implementation of up to two nested interrupts and two nested calls to subprograms. Program memory is synthesized from EAB blocks in 16 bit data ROM configuration. The size of PC can be incremented up to 14 bits to support PM extension.

Instruction Register (IR). IR contains the operation code (OPC) and operands of currently executing instruction. It is 16 bits register formed by two 8 bit registers: **IR_H и IR_L**. **IR_H** (fig. 5) contains one of the following: OPC or OPC and operands of instructions from "Bit" and "Control" groups (Call or Goto). **IR_L** holds the address of a register **(FR_Addr)** or Immediate Data. When a data transfer from FR to address registers occur, **IR_L** goes to into

counting mode to address several consecutive FR's registers, which content will be assumed to corresponding AR.

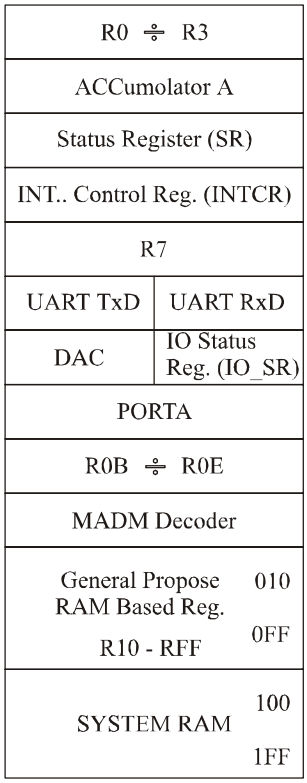

Fig. 3. File Reg Map

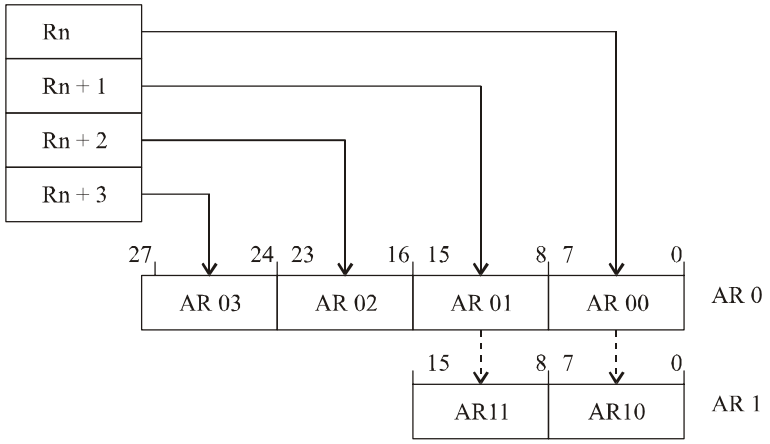

Fig. 4. Address Register MOVE:
MOVE ARO,Rn (MOVE AR1,Rn)

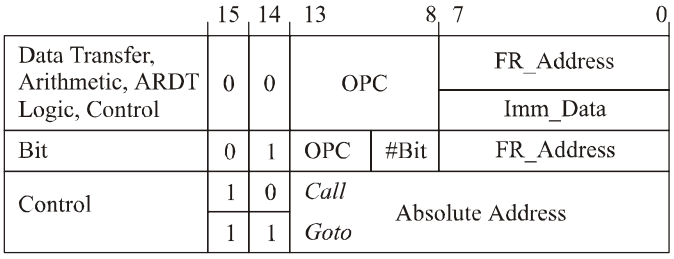

Fig. 5. Instruction Format

Control unit (CU). In the VMCore project, the CU has several functions: instruction decoding; forming the logical conditions for controlling the multiplexers (A_mux, B_mux, etc.); generating the signals for control and synchronization of the processes in the microcontroller. When addressing the SFR, in the CU are built the access control signals (Control Bus).

I/O Controller. There are three synthesized peripheral controllers in VMCore project:

- UART. A controller used for Host system communication. Among its tasks are managing voice synthesis, programming SFR and testing of the system.
- Port A. One of the basic applications of microcontroller for VM synthesis is for Interactive Voice Response systems. Because of that, an 8 bit parallel PORT A is also included in the VMCore project for testing of IVR applications.
- MADM Decoder. A controller used for decompression and error correction of voice primitives, which are subject of transmission. It also incorporates a serial DAC controller.

INSTRUCTION SET SUMMARY

The project's instruction set (Table 1) consists of 54 instructions arranged in six groups: Data transfer, Arithmetic, Logic, Bit, Control and Address Register Data Transfer. All instructions have a fixed length of 16 Bit and are coded (fig. 5) in four different formats. The execution of 49 of the instructions takes one system clock.

Three instructions (*DSZ Rn, TBSC Rn,bi, TBSS Rn,bi*) are executed for different count of system clocks (one or two depending on SKIP operation). The execution of initialization

address register instructions take respectively four system clocks for *MOVE AR0,Rn* and two system clocks for *MOVE AR1,Rn*.

TABLE 1. VMCore Instruction Set

Mnemonic	**Function**	**Mnemonic**	**Function**
Data Transfer		**Logic**	
MOV A,Rn	(A) ← (Rn)	AND A,Rn	(A) ← (A) AND (Rn)
MOV Rn,A	(Rn) ← (A)	AND A,#Data	(A) ← (A) AND #Data
MOV A,#Data	(A) ← #Data	OR A,Rn	(A) ← (A) OR (Rn)
MOV R0,#Data	(R0) ← #Data	OR A,#Data	(A) ← (A) OR #Data
MOV Rn,(R0)	(Rn) ← ((R0)); FR Scope	XOR A,Rn	(A) ← (A) XOR (Rn)
MOV Rn,(R0),RAM	(Rn) ← ((R0)); RAM Scope	XOR A,#Data	(A) ← (A) XOR #Data
MOV (R0),Rn	((R0)) ← (Rn) ; FR Scope	NOT Rn	(Rn) ← not (Rn)
MOV (R0),Rn,RAM	((R0)) ← (Rn); RAM Scope	ROL Rn	Rotate Left through Cary
MOV Rn,(R1+A)	(Rn) ← ((R1) + (A)); FR Scope	ROR Rn	Rotate Right through Cary
MOV Rn,(R2)	(Rn) ← ((R2)); RAM Scope	Bit	
MOV (R2),Rn	((R2)) ← (Rn) ; RAM Scope	SETB Rn,bi	(Rn[bi]) ← 1
CLR Rn	(Rn) ← 00	CLRB Rn,bi	(Rn[bi]) ← 0
Arithmetic		TBSS Rn,bi	Skip if (Rn[bi]) = 1
ADD A,Rn	(A) ← (A) + (Rn)	TBSC Rn,bi	Skip if (Rn[bi]) = 0
ADD A,Rn,C	(A) ← (A) + (Rn) + C	LDBF Rn,bi	(BF) ← (Rn[bi])
ADD A,#Data	(A) ← (A) + #Data	Control	
ADD Rn,A	(Rn) ← (A) + (Rn)	RET	(PC) ← (PCST)
ADD Rn,A,C	(Rn) ← (A) + (Rn) + C	RETI	(PC) ← (PCST) ; (IEN) ← 1
SUB A,Rn,	(A) ← (A) - (Rn)	CWDT	(WDT) ← 0
SUB A,Rn,C	(A) ← (A) - (Rn) - C	NOP	(PC) ← (PC) + 1
SUB A,#Data	(A) ← (A) - #Data	CALL Abs	(PCST) ← (PC); (PC) ← Abs
SUB Rn,A	(Rn) ← (A) - (Rn)	GOTO Abs	(PC) ← Abs
SUB Rn,A,C	(Rn) ← (A) - (Rn) - C	**Address Register Data Transfer (ARDT)**	
CMP A,Rn	(SR[C,Z]) ← (A) – (Rn)	MOV Rn,(AR0)	(Rn) ← ((AR0))
CMP A,#Data	(SR[C,Z]) ← (A) – #Data	MOV Rn,(AR0 + 1)	(Rn) ← ((AR0 + 1))
INC Rn	(Rn) ← (Rn) + 1	MOV Rn,(AR1<<BF)	(Rn) ← ((AR1 << BF))
DEC Rn	(Rn) ← (Rn) - 1	MOV AR0,Rn	(AR0i) ← (Rn+i); i = 0..3
DSZ Rn	(Rn) ← (Rn) – 1; skip if(Rn)=0	MOV AR1,Rn	(AR1i) ← (Rn+i); i = 0..1
		MOV (AR0),Rn	((AR0)) ← (Rn)
		MOV (AR0+1),Rn	((AR0+1)) ← (Rn)

Some of the instructions are unique for the VMCore project. These are all the instructions from "Address Register Data Transfer" group. They are used for indirect access to SFR's data. Some of the instructions in "Data Transfer" group are used for processing arrays stored in File register (*MOV Rn,(R1+A)*) or File register and System RAM (*MOV Rn,(R0); MOV Rn,(R0),Ram; MOV (R0),Rn; MOV (R0),Rn,Ram*). Two instructions (MOV Rn,(R2) and MOV (R2), Rn) address System RAM only. They are used to store system registers in case of interrupt proceeding, because they do not change Status Register's flags.

RESULTS

The VMCore project is designed using VHDL and compiled in Altera's QUARTUS II design & synthesis software environment. The separate components are simulated on Mentor Graphic's MODELSIM simulation software. The project is built for three FPGA chips from the Altera *ACEX 1k* series [1]. The allocation of the occupied resources in percentages is shown on figure 6.

The microcontroller is tested at 20 Mhz system clock and execution time of 250ns for one machine cycle. The actual tests are made on Altera *EP1K50 - 3 Speed Grade* module and are observed with Fluke's ScopeMeter 99B current clamping oscilloscope.

The whole project uses 1249 Logic Elements and 3 Embedded Array Blocks from the *EP1K50* device. Figure 7 shows the allocation of separate components in percentages.

The quality of the synthesized messages is expertly measured by the Mean Opinion Scores (MOS) [9] method. Four experiments are carried out with phonetic primitives based on:

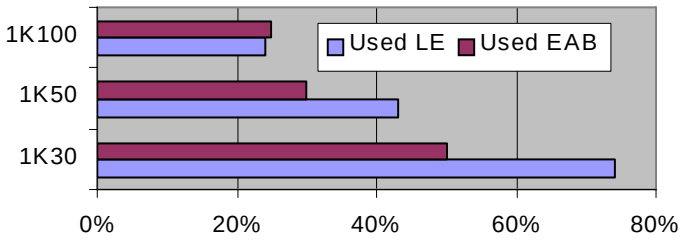

Fig.6. Used FPGA resourses

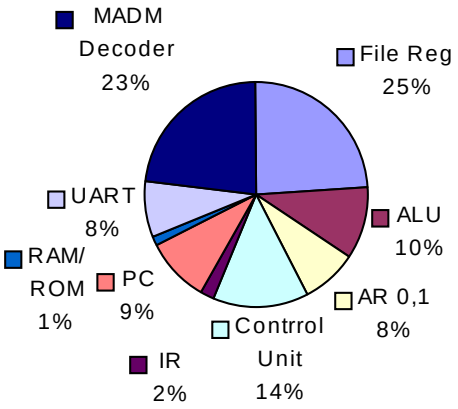

Fig. 7. VMCore - Component Allocation

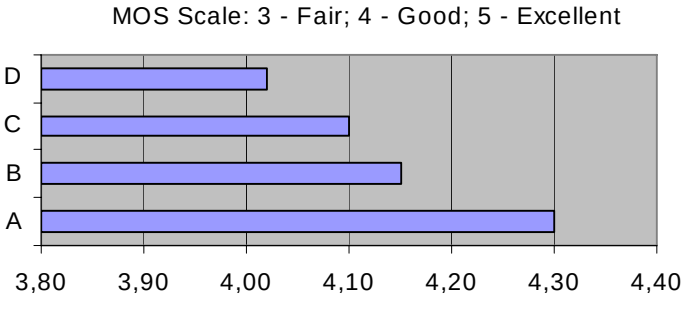

Fig. 8. Quality of the voice message synthesis

- A. Sentences
- B. Words and phrases
- C. Synthesized isolated numbers
- D. Words, phrases and synthesized numbers

The test messages are sampled at frequency of 11025 Hz and are converted in analog speech signals using serial DAC *LTC 1451* in 9 bit mode [7].

The expert evaluation of the quality of the synthesized messages is shown in fig. 8 and is classified as "good quality, only very slight impairments" [9].

As a result of the carried investigations, the following conclusions can be made:

- The designed microcontroller can be implemented on low cost FPGA platforms and considerable amount of resources (57% LE and 70% EAB for *EP1K50* device) remain free for developing additional applications and interfaces (fig.6). Through the embedded JTAG port the system offer the possibility to be partially or fully reconfigurable, which enables its elaboration after it is put into operation.

- The chosen method for synthesis ensures high quality of the sent voice messages. Regardless of the limited dictionary that is typical for systems of such class, they have wide practical application in the cases when the quality has to dominate.

REFERENCES

[1] Altera, *ACEX 1K Programmable Logic Device Family* , Data Sheet, 2002.

[2] Dutoit T., *An Introduction to Text - to - Speech Synthesis,* Academic Publisher, 1996.

[3] Hennessy J, Patterson D., *Computer Architecture a Quantitative Approach*, Morgan Kaufman Publisher, 2000.

[4] Kanev I., Error Correction at Decompressing with MADM Algorithms, J.T.U., Fundamental Sciences and Application, Vol. 11, 2004.

[5] Kanev I., *Implementation of MADM Algorithms on FPGA Based Platform*, Proc. of CompSysTech'04, Int. Conf. on Computer Systems and Technologies, I.7.1-9, 2004.

[6] Kanev I., *Voice Synthesis of Cardinal Numbers,* Proc. of the National Scientific Conf. "10 Years Department of Computer Systems at the Technical University – Plovdiv ", Plovdiv, Bulgaria, 2003.

[7] Linear Technology, *LTC 1451 - 12 Bit Rail - to - Rail Micropower DAC*, Data Sheet, 2003.

[8] Manoilov P., Kuzmanov G., Stefanov T., Momchilova V., Popov A., *Two Approaches in One for Quick and Efficient Design of Low Area Custom Microprocessor Cores. Developments if the Ims8NI Core via VHDL and Logic Synthesis*, Proc. of the Seventh International Conference Electronics '98, September 23 –25, 1998, Sozopol, Bulgaria, book 2, pp. 57-64.

[9] Rabiner L., *Applications of Voice Processing to Telecommunications,* Proc. of the IEEE vol. 82 No. 2, February 1994.

[10] Tabak D., *RISC Systems*, Research Studies Press Ltd.: Taunton, Somerset, England TA1 1HD, 1990.

[11] www.opencores.org/projects/riscmcu, Yap Zi He, Building A RISC Microcontroller in an FPGA.

ABOUT THE AUTOR

Ivan Kanev, Department of Computer Systems, Technical University Sofia – Branch Plovdiv, Phone +359 32 659 704, E-mail: ikanev@it-academy.bg.