UDC 004.231.3 (004.056.55) **doi:** https://doi.org/10.20998/2522-9052.2023.2.13

Dmytro Salnikov, Dmytro Karaman, Viktoriia Krylova

National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine

HIGHLY RECONFIGURABLE SOFT-CPU BASED PERIPHERAL MODULES DESIGN

Abstract. Research motivation. When developing microcontrollers, manufacturers try to include as many different types of peripherals as possible in order to increase the marketing attractiveness of their products. On the one hand, with a large assortment of various peripheral modules, it is very difficult to implement several devices of the same type in the microcontroller: manufacturers are mainly limited to 1-2 instances, in rare cases 4 modules of the same type are included. On the other hand, most software projects do not use all the peripherals of modern microcontrollers and many devices are left unused, while there may be a shortage of other types of modules. Another problem that has become especially noticeable for microcontrollers used in the field of IoT is the cryptographic protection of data that is transmitted through built-in information exchange interfaces. The main efforts of researchers and developers of cryptographic data protection methods were aimed at reducing energy-intensive operations, memory access iterations and speeding up encryption processes while maintaining a high level of cryptographic protection and enabling efficient data distribution within IoT devices networks. **Research results.** This paper presents an alternative approach to the manufacture of peripheral modules as part of microcontrollers. The authors propose to use a configurable software processor module based on the MIPS architecture with a reduced instruction set and limited capabilities. **Conclusions.** This approach would make it possible to dynamically change the functionality of peripheral modules in accordance with the requirements of the developed software solution, which in turn will increase the efficiency of the microcontroller chips capabilities utilization. In addition, the transfer of data stream encryption functions to the reconfigurable core of the peripheral module will provide fast and transparent cryptographic protection, as well as allow offloading the microcontroller core and increasing the energy efficiency of chips while reducing their production cost.

Keywords: peripheral module; soft-CPU; RISC; MIPS architecture; FPGA; internet of things; encryption, lightweight cryptography; AEAD; Ascon.

Introduction

Modern microcontrollers must have the widest possible capabilities so that devices based on them can meet the requirements of users. To do this, microcontroller manufacturers are constantly improving the computing cores in microcontrollers, as well as increasing the number and expanding the functionality of peripheral modules. However, this approach leads to a narrowing of promising areas of application of microcontrollers, especially from electronic devices with intelligent capabilities for everyday life, as part of smart home systems, as well as distributed systems with stringent requirements for energy efficiency: wireless sensor networks, devices as part of the Internet of Things (IoT) etc.

At the moment, the problem of the efficiency of the use of energy resources is one of the most important on the European continent and even in the world. At the same time, humanity uses millions of electrical devices that use microcontrollers to deliver specific features to their users.

Internet of things devices are widely used in most modern homes in the form of climate control systems, safety or security systems, voice-controlled assistants etc. The number of such devices continues to grow. Thus, each device's efficiency impacts energy usage.

Despite the relatively low level of energy consumption of each such device, the total costs for millions of sold devices are significant, and optimizing the energy consumption of each of them is an important task of modern science and technology.

Another feature that has become relevant recently for distributed embedded systems, especially for IoT, is ensuring the security, integrity and confidentiality of transmitted information, as well as providing protection from unauthorized access. Implementations of traditional encryption, integrity and confidentiality methods are very resource intensive. During their development, first of all, the issues of reliability and resistance to cryptanalytic attacks were considered, rather than the possibility of use on platforms with limited computing capabilities and limited power supplies. That is why recently there have been many attempts to adapt existing solutions to the operating conditions of IoT devices, as well as to search for new algorithms that initially take into account the limitations of the IoT platform elements and provide a level of protection no less than traditional cryptographic methods.

Relevance of the topic and related works

The design of electronic devices is directly related to the choice of optimal microprocessor used to implement all planned functional capabilities of the device. Modern microcontrollers use different architectures of the processor core and have a different set of peripheral units that can be configured to perform various tasks. Selection of an appropriate architecture and a set of such peripheral blocks affects a possibility to expand and/or improve the functional capabilities of the device. With this in mind, electronics and software engineers often use overpowered devices to avoid problems in the future and let the project grow without significant printed circuit boards and software changes.

Most of the changes in the software can be related to the lack of peripheral modules of the required type or the inability to configure them to work in the required mode.

Problems of this kind can be solved by including CPLD or FPGA configurable blocks in the circuit [1, 2, 3]. In this case, the programmer can include the implementation of the necessary peripheral modules in the Verilog or VHDL hardware description languages as part

of the project. Which requires additional programming skills and, more importantly, significantly increases the cost of such solutions and the process of working with them [4]. Solutions with embedded CPU and FPGA in a single chip are available from most semiconductor manufacturers like Intel FPGA, AMD Xilinx, Analog Devices [5, 6], etc. Another option is highly configurable hybrid ICs like Cypress microcontrollers [7].

Flexible combination of the embedded microprocessor functions core (CPU) and the programmable part (FPGA) allows developers, if necessary, to expand the functionality of the designed solution by transferring complex and intensive operations for the CPU to be executed in a separate specialized module in the FPGA part. In the same way, you can expand the pool of available peripherals, since the FPGA part in most cases has access to the lines and I/O ports of the chip.

The issue of creating flexible auxiliary peripheral modules has been considered for the past decade. For example, [8] studies the possibility of expanding the capabilities of the PIC microcontroller by connecting external programmable peripherals to general purpose I/O ports. This solution allows you to add the missing peripheral module, but occupies one of the few input/output ports of the microcontroller. In addition, this approach requires the involvement of external additional hardware resources.

The use of hardware emulation through software is proposed in [9] that allows you to recreate the behavior of the I2C serial bus controller through the GPIO ports in a microcontroller based on the RISC-V core. Thus, the expansion of the functionality of the microcontroller without the involvement of additional equipment is provided. However, in this case, the kernel receives an additional load, since it must spend additional processor time emulating the I2C transceiver. In addition, questions arise regarding the performance parameters of such a solution. Definitely, such a module will not be able to work with devices on the bus at high data exchange rates.

A specific way to use custom peripheral modules is discussed in [10]. The authors create digital hardware twins of peripheral modules using FPGA, which make it easier for students to learn embedded systems in distance education. Access to the modules is provided via the Internet using a client-server architecture.

Another example of a remote peripheral module is discussed in [11]. A method for remote control of microcontroller input/output interfaces using WebUSB technology is proposed.

Thus, the problem of the efficient, easy to use and low-cost architecture for microcontroller peripheral unit implementation is a relevant topic to research and development nowadays.

The issues of security and data protection for elements of the Internet of Things (IoT) system have become especially acute since the concept was adopted by leading electronics manufacturers and began to be massively introduced into various fields of human activity. The methods and technologies that initially relied on that concept of devices networking did not assume that devices would be available to a wide range of users, among which there could be many intruders.

Numerous attempts to adapt existing methods of cryptographic protection and authentication have shown that in order to implement these methods in systems with limited resources, it is necessary either to take resources from other functions in the IoT system, or to curtail the capabilities or even modify the main algorithms [12], which can lead to a decrease in cryptographic resilience or the emergence of vulnerabilities.

This state of affairs led to the need to develop a separate class of cryptographic protection methods – lightweight cryptography [13]. These methods had to take into account the features of distributed embedded systems: reduced power supply and low level of performance, and at the same time provide a level of protection that is not inferior to traditional methods.

Authenticated Encryption with Attached Data (AEAD) [14] occupies a special place among the methods of lightweight cryptography. Their peculiarity lies in the fact that not the entire message is encrypted, but only a part of it with the most sensitive data, while the entire message is authenticated. This allows you to ensure data protection and message integrity, while the data necessary for successful and efficient routing remains open.

In February 2023, the US National Institute of Standards and Technology (NIST) announced the results of a competition to select a lightweight cryptography algorithm [15] that will form the basis of the corresponding standard. As a result of many years of thorough selection among 57 applicants, the family of cryptographic algorithms Ascon [16] was chosen. The implementation of the block cipher algorithm, which is part of this family, is discussed in this article.

Soft-CPU-based peripheral module

From the authors' point of view, the complexity of implementing peripheral modules in hardware significantly reduces the possibilities of their usage, and most companies prefer usage-ready solutions that do not require additional development process.

At the same time, most programmers involved in the creation of microcontroller-based systems have an understanding of the processes and operations which take place in peripheral units and can create their programmatical description using C/C++ or assembler language.

Worthless to say that availability of mature programming development tools, in particular, a C language compiler and/or an assembly parser, plays an even more important role in the life cycle of any software project.

Thus, it is promising to use general purpose CPU architectures without additional license conditions, mature development tools and well-known instruction sets, such as MIPS, RISC-V etc.

At the current stage, a software processor that uses MIPS architecture has been chosen to construct the experimental architecture. Issues of the efficiency of MIPS/RISC-V software processors and development for such systems on chip are widely considered in the literature, particularly in [17].

Basic architecture of MIPS soft processor was modified for this experiment. Modules for working with data memory were removed. Registers $a0-$a3 were used to load data from the queue. Registers $s0-$s7 – for setting the output GPIO lines to the required state. Other registers are available to the programmer for intermediate calculations.

For such a system, basic UART module can be executed in the form of the following program assembly language (Listing 1).

```
_start:
   li        $t1, 10          # set bit counter st+8d+sp
   move      $t0, $a0    # load data from queue to tx buffer
   andi      $t0, $t0, 0xFF   # cut data to 1 byte with mask
   sll       $t0, $t0, 1      # prepare start bit
   or        $t0, $t0, 0x01
loops:
   move      $s0, $t0   # copy to s0.0 bit data to tx from buf
   jal baud_delay
   srl       $t0, $t0, 1      # prepare next bit to tx
   subu      $t1, $t1, 1      # count bits transmitted
   bgtz      $t1, loops       # test if tx complete
   # rest of code
   nop
   j _start
baud_delay:
   # delay routine according to baudrate
   jr $ra
```

Listing 1

The process of designing and manufacturing ICs is overcomplicated and involves significant workforce requirements with licensing expenses. It is not possible for small IT companies to plan a set of required peripherals and produce a chip or ASIC which matches needs of a project exactly. It increases time-to-market of the device, expenses and complexity of any device. Moreover, there is no way to deal with changes in requirements. It is not possible to reconfigure such chips in future. A possible way of solving this problem presented in [8], while our proposal is more generic and configurable.

Implementation of some peripheral modules consume significant hardware resources. This affects chip cost and energy consumption. Thus, widely adopted practice is to use cheaper devices and implement required peripheral in software using general purpose input/output pins. An example of such a study is presented in [9]. It provides software implementation of I2C peripheral.

Proposed in this work, MIPS-based block, is 3 times smaller in terms of consumed FPGA resources.

Another field where highly reconfigurable peripherals are needed is a study laboratory. Any university in the world now has remote laboratory classes and requires students to perform experiments and laboratory assignments from home. A solution that uses FPGA logic to mimic some type of hardware is proposed in [10, 11, 19]. In addition, hardware emulated using FPGA can be used to improve testing methods [18], and to measure and characterize network architectures [19]. Our approach of realization of peripheral blocks may simplify implementation of generic execution units suitable for remote/concurrent usage.

We can't analyze hardware expenses to create specific peripheral blocks in ASIC. It depends on the technical process of the manufacturer and a lot of design choices of the engineer. We assume that ASIC resource consumption is compatible with FPGA resources consumed to implement such blocks. Mentioned blocks and their configuration can be seen in [20].

In Table 1 one can see resource consumption for widely used Intel FPGA peripheral IP blocks. Note that some of them use additional memory which is not used to implement register memory maps, queues or FIFO buffers. Thus, additional memory is required to implement an interface to control them.

To verify the solutions proposed in the article, we used a 5CSEBA6U23I7 device of Intel Cyclone V SoC with 2-core ARM Cortex A9 FPGA.

Table 1 – **FPGA Resources consumption for different types of IP cores**

	Regis-ters	ALM blocks	Block me-mory bits
Modified MIPS core	74	72	0
Intel SPI IP Core	56	23	0
Intel 16550 UART IP Core	156	114	0
Intel UART IP Core	0	1	0
Intel I2C Host IP Core	215	143	40

In addition to peripheral blocks, we provide resources consumed to synthesize a modified MIPS processor suitable for software implementation of the listed peripherals. The structure of the synthesized CPU is show on Fig. 1.

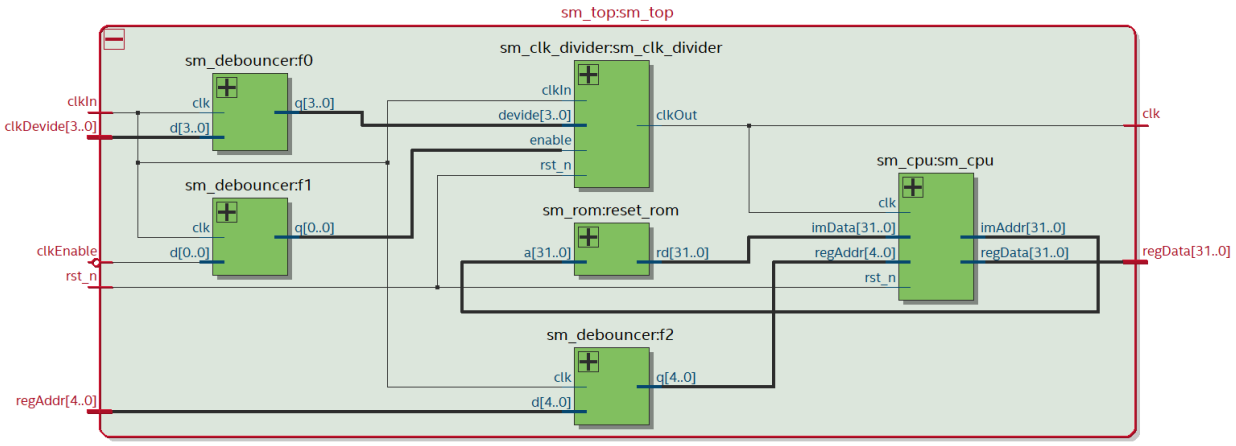

Fig. 1. Structure of the synthesized MIPS core

Modern CPU implementations use pipelining to allow computing cores to work faster by using higher frequencies. The problem is that on high frequencies timing delay on logic used to implement CPU becomes significant. Such delay limits frequency and, as a result, CPU performance. Thus, adding a pipeline (e.g., save intermediate states of signals to registers) to the design allows to improve capabilities of the CPU.

Moreover, it is a common practice to include hazard units to CPU implementations to allow pipelining of the CPU and achieve higher performance. In contradiction to common CPUs, reconfigurable peripheral modules do not have a requirement to run with frequencies higher than 30-40MHz. Such relaxation of the requirements allows to drop these modules and significantly reduce resources required to implement peripheral module.

As most peripheral blocks of modern microcontrollers the suggested implementation requires additional FIFO buffers and input/output multiplexers to achieve efficient signal transmitting/receiving. Such logic can be connected to the registers of the peripheral module as shown on Fig. 2.

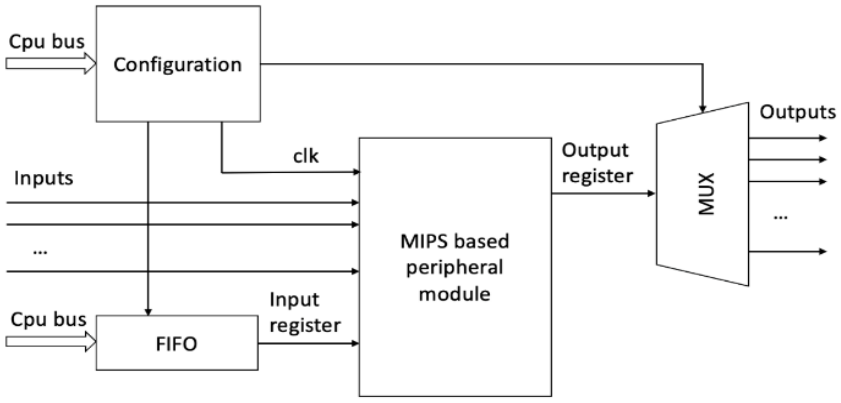

Fig. 2. Additional logic needed to used proposed modules

In this article we don't compute resources that required to implement such buffers and interconnect. In most cases resources spent to synthesize such logic is equivalent to the resources needed for functioning of regular peripheral block.

Implementing Encryption Functions in the Peripheral Module

One of the modern requirements for devices that are oriented to work as part of the IoT is to support a certain stack of cryptographic data protection methods: block or stream encryption, various ciphertext block chaining modes, hash sum calculation, message authentication mechanism (MAC) and even digital signature generation. In most cases, the necessary protection mechanisms are implemented at the software level and are executed by the main computing core of the microcontroller. Peripheral modules are not involved in the process in any way and perform only the functions of receiving, transmitting and transport integrity control of data packets.

Much less common are devices in which cryptographic functions or their composite operations are implemented at the level of the computing core (hardware support for cryptographic operations). However, despite the excellent speed of the encryption and authentication processes, the use of such tools faces significant limitations: high price and export control by government agencies.

Since the core of the MIPS soft processor was used in the development of the reconfigurable peripheral module, the possibility of transferring the execution of encryption functions from the main core of the microcontroller to the soft processor core of the peripheral module was considered.

Such a solution will allow to unload the main core of the microcontroller and making the encryption process transparent: in the main program, it will only be enough to send data for transmission and read the received data, the peripheral module will automatically perform encryption during transmission and decryption during reception. In the main program code, it is only necessary to provide for the process of initial initialization of encryption functions in the peripheral module (loading keys, initialization vectors, mode selection), which is performed when the system is initialized after switching on or by special request.

As an example, the family of lightweight algorithms Ascon [16] was considered. It includes an authenticated encryption algorithm and a hash function calculation algorithm. Both algorithms use a transformation called the sponge function. The authors of the algorithm set offer a wide range of software implementations for various microprocessor architectures with different degrees of optimization in public repository on GitHub.

The implementation of the entire set of algorithms by means of a peripheral module does not seem appropriate, since after compilation, the entire code, taking into account optimizations, occupies at least 6456 bytes in memory. Therefore, it is proposed to perform all operations for initialization and preparation of the encryption process in the main core of the microcontroller, and to transfer the initial state for the encryption or decryption functions to the peripheral module.

An estimate of the size of the executable code is given in Table 2. For estimation, a general implementation version of the algorithm was compiled from the authors' repository on GitHub, ascon128v12 version for 32-bit processors with instructions on integer

operands (bi32). The compilation was done on mips gcc ver. 12.2.0, emulation and debugging were performed on the MIPS32r5 generic kernel simulator.

Table 2 – **Estimation of the code size of the encryption system**

Implemented function	Code size, bytes
Full implementation of the algorithm stack	6456
Implementation of encryption and decryption (auxiliary code)	5344 (3928)
Encryption function only (including auxiliary code)	4544
Decryption function only (including auxiliary code)	4720

An example of the previously discussed program for the operation of the UART module with the implementation of preliminary encryption of the transmitted data is shown below (Listing 2).

```
_start:
    move        $t0, $a0    # load plaintext from queue to buffer
    xor $t0, $t0, $t4       # plain text being xored with state block to get ciphertext
    jal ascon_encrypt       # call encryption function - permutations of state block
    li          $t1, 10     # set bit counter st+8d+sp
    andi        $t0, $t0, 0xFF  # cut data to 1 byte with mask
    sll         $t0, $t0, 1     # prepare start bit
    or          $t0, $t0, 0x01
loops:
    move        $s0, $t0    # copy to s0.0 bit data to tx from buf
    jal baud_delay
    srl         $t0, $t0, 1     # prepare next bit to tx
    subu        $t1, $t1, 1     # count bits transmitted
    bgtz        $t1, loops      # test if tx complete
    # rest of code
    nop
    j _start
baud_delay:
    # delay routine according to baudrate
    jr $ra
```

Listing 2

Conclusions

The use of processor architectures with reduced command and pipeline functionality is seen as a profitable replacement for traditional configurable peripheral modules that are widely used at the moment.

The FPGA implementation of software-based MIPS core, modified to be lightweight from the recourse usage perspective, shows that such design consumes comparable, with most peripheral blocks, among FPGA resources.

Thus, microcontrollers with 10 to 20 such modules may be a competitive replacement to widely used solutions.

The transfer of encryption functions for transmitted/received data from the main core program of the microcontroller to the core of the soft-processor allows to significantly reduce load of the main core, increase performance, and also make the processes of ensuring the protection and authentication of data transmitted by IoT devices transparent to user software. As the results of the implementation showed, the transfer of all encryption functions to the microcode that is executed in the peripheral module can lead to a high consumption of a very limited amount of soft processor memory and reduce the performance of the peripheral module. In this regard, the issue of implementing hardware support for encryption at the level of assembler commands of a soft-processor is considered.

REFERENCES

1. Liu, C., Liu Q. and Cheng L. (2011), "CPLD based MCU coprocessor design and experiment platform", *2011 Int. Conf. on Electronics, Communications and Control*, Ningbo, China, 2011, pp. 1365-1368, doi: https://doi.org/10.1109/ICECC.2011.6066408.
2. Schiavone, P.D., Rossi, D., Mauro, A. Di, Gürkaynak, F.K., Saxe, T., Wang, M., Yap, K,C. and Benini, L. (2021), "Arnold: An eFPGA-Augmented RISC-V SoC for Flexible and Low-Power IoT End Nodes", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 677-690, April 2021, doi: https://doi.org/10.1109/TVLSI.2021.3058162.
3. Matsumura, T., Okada, N., Kawamura, Y., Nii, K., Arimoto, K., H. Makino and Y. Matsuda (2014), "The LSI implementation of a memory based field programmable device for MCU peripherals", *17th Int. Symposium on Design and Diagnostics of Electronic Circuits & Systems*, Warsaw, Poland, pp. 183-188, doi: https://doi.org/10.1109/DDECS.2014.6868787.
4. Amano, H., Abe, S., Hasegawa, Y., Deguchi, K. and Suzuki, M. (2005), "Performance and cost analysis of time-multiplexed execution on the dynamically reconfigurable processor", *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 315-316, doi: https://doi.org/10.1109/FCCM.2005.52.
5. (2023), *Intel® Products / Intel® FPGAs and SoC FPGAs*, available at: https://www.intel.com/content/www/us/en/products/details/fpga.html.
6. (2023), *AMD Xilinx Products / Adaptive SoCs: ZYNQ and Versal*, available at: https://www.xilinx.com/products/silicon-devices/soc.html.
7. (2023), *Cypress PSoC® 6 Microcontrollers Purpose-Built for the Internet of Things*, available at: https://www.infineon.com/dgdl/Infineon-PSoC_6_MCU_The_New_Standard_for_the_Internet_of_Things-ProductBrochure-v05_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0f64f95450c7.
8. Penteado, C.G. and Moreno, E.D. (2009), "A Specialized Processor for Emulating Peripherals of the PIC Microcontroller", *IEEE Latin America Transactions*, vol. 7, no. 2, pp. 133-140, June 2009, doi: https://doi.org/10.1109/TLA.2009.5256820.
9. Molina-Robles, R., García-Ramírez, R., Chacón-Rodríguez, A., Rimolo-Donadio, R. and Arnaud, A. (2021), "Low-level algorithm for a software-emulated I^2C I/O module in general purpose RISC-V based microcontrollers", *2021 IEEE URUCON*, Montevideo, Uruguay, pp. 90-94, doi: https://doi.org/10.1109/URUCON53396.2021.9647309.
10. Buysse, L., Van den Broucke, Q., Verslype, S., Peuteman, J., Boydens, J. and Pissoort, D. (2021), "FPGA-based digital twins of microcontroller peripherals for verification of embedded software in a distance learning environment", *2021 XXX Int. Scientific Conference Electronics* (*ET*), Sozopol, Bulgaria, pp. 1-4, doi: https://doi.org/10.1109/ET52713.2021.9579770.
11. Huang, L. and Shu, Y. (2022), "Design and Research of Microcontroller I/O Control Technology", *2022 IEEE 4th International Conference on Power, Intelligent Computing and Systems (ICPICS)*, Shenyang, China, 2022, pp. 263-266, doi: https://doi.org/10.1109/ICPICS55264.2022.9873583.

12. Kane, L. E., Chen, J. J., Thomas, R., Liu, V. and Mckague, M. (2020), “Security and Performance in IoT: A Balancing Act”, *IEEE Access*, vol. 8, pp. 1219.69-1219.86, doi: https://doi.org/10.1109/ACCESS.2020.3007536.
13. Biryukov, A. and Perrin, L. (2017), “State of the art in lightweight symmetric cryptography”, *Cryptology ePrint Archive*, Nov. 2017, available at: https://eprint.iacr.org/2017/511.pdf.
14. Jutla, C. S. (2008), “Encryption Modes with Almost Free Message Integrity”, *Journal of Cryptology*, vol. 21, pp. 547–578, doi: https://doi.org/10.1007/s00145-008-9024-z.
15. (2023), *NIST Selects ‘Lightweight Cryptography’ Algorithms to Protect Small Devices*, National Institute of Standards and Technology (NIST) Website, February 07, 2023, available at: https://www.nist.gov/news-events/news/2023/02/nist-selects-lightweight-cryptography-algorithms-protect-small-devices.
16. Dobraunig, C., Eichlseder, M., Mendel, F. and Schläffer, M. (2023), *Ascon: Lightweight Authenticated Encryption & Hashing*, available at: https://ascon.iaik.tugraz.at/index.html.
17. Dewangan, G. K., Prasad, G. and Mandi, B.C. (2021), “Design and Implementation of 32 bit MIPS based RISC Processor”, *2021 8th International Conference on Signal Processing and Integrated Networks* (*SPIN*), Noida, India, 2021, pp. 998-1002, doi: https://doi.org/10.1109/SPIN52536.2021.9566007.
18. Li, T. and Liu, Q. (2016), “Cost effective partial scan for hardware emulation", *2016 IEEE 24th Annual Int. Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 131-134, doi: https://doi.org/10.1109/FCCM.2016.39.
19. Khamis, M., El-Ashry, S., Shalaby, A., AbdElsalam M. and El-Kharashi M. W. (2018), “A configurable risc-v for noc-based mpsocs: A framework for hardware emulation”, *2018 11th International Workshop on Network on Chip Architectures (NoCArc)*, pp. 1-6, doi: https://doi.org/10.1109/NOCARC.2018.8541158.
20. (2021), *Embedded Peripherals IP User Guide for Quartus Prime 21.4*, Intel, UG-01085, available at: https://www.intel.com/content/www/us/en/docs/programmable/683130/21-4/introduction.html.

Received (Надійшла) 24.02.2023
Accepted for publication (Прийнята до друку) 10.05.2023

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Сальніков Дмитро Валентинович – кандидат технічних наук, старший викладач кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Dmytro Salnikov – Candidate of Technical Sciences, Senior Lecturer of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: dmytro.salnikov@khpi.edu.ua; ORCID ID: http://orcid.org/0000-0007-6201-5370.

Караман Дмитро Григорович – старший викладач кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Dmytro Karaman –Senior Lecturer of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: dmytro.karaman@khpi.edu.ua; ORCID ID: http://orcid.org/0000-0002-7252-3172.

Крилова Вікторія Анатоліївна – кандидат технічних наук, доцент кафедри автоматики та управління в технічних системах, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Viktoriia Krylova – Candidate of Technical Sciences, Associate Professor of the Department of automation and control in technical systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine.
e-mail: vika.hpi@outlook.com; ORCID ID: http://orcid.org/0000-0002-4540-8670.

Архітектура гнучко конфігурованих периферійних модулів на базі програмованих процесорних ядер

Д. В. Сальніков, Д. Г. Караман, В. А. Крилова

Анотація. **Мотивація дослідження.** При розробці мікроконтролерів виробники намагаються включити якнайбільше різних видів периферійних пристроїв, щоб підвищити маркетингову привабливість своєї продукції. З одного боку, при великому асортименті різних периферійних модулів дуже складно включити до складу мікроконтролера велику кількість пристроїв одного типу: виробники переважно обмежуються 1-2 екземплярами, дуже рідко зустрічаються 4 модулі однакового типу. З іншого боку, більшість програмних проектів не використовують всю периферію сучасних мікроконтролерів і багато пристроїв залишаються незадіяними, тоді як може відчуватися нестача модулів іншого типу. Ще однією проблемою, яка стала особливо відчутною для мікроконтролерів, що застосовуються у сфері IT, це криптографічний захист даних, які передаються через вбудовані інтерфейси обміну інформацією. Основні зусилля дослідників та розробників криптографічних методів захисту даних були спрямовані на зниження енерговитратних операцій, звернень до пам'яті та прискорення процесів шифрування за одночасного збереження високого рівня криптографічного захисту та забезпечення можливості ефективного поширення даних у мережах пристроїв IoT. **Результати дослідження.** У роботі подано альтернативний підхід до виготовлення периферійних модулів у складі мікроконтролерів. Пропонується використовувати конфігурований модуль програмного процесора на базі архітектури MIPS з укороченим набором команд та обмеженими можливостями. **Висновки.** Такий підхід дозволяє динамічно змінювати функціонал периферійних модулів відповідно до вимог програмного рішення, що розробляється, що в свою чергу дозволить підвищити ефективність використання можливостей мікросхем мікроконтролерів. Крім того, перенесення функцій шифрування потоку даних в ядро периферійного модуля, що реконфігурується, дозволить забезпечити швидкий і прозорий криптографічний захист, а також дозволить розвантажити ядро мікроконтролера і підвищити енергоефективність мікросхем при одночасному зниженні собівартості їх виробництва.

Ключові слова: периферійний модуль; софт-процесор; RISC; архітектура MIPS; ПЛІС; інтернет речей; шифрування; легка криптографія; AEAD-режим, Ascon.