BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES,
DOI:10.24425/bpasts.2026.158308

Performance analysis of RISC-V based cores in a dual-core programmable logic controller

Bartlomiej Jekot, Blazej Korzuch, Fabian Oles, Pawel Lempa, Oskar Stajer, Rafal Przechowski, Robert Czerwinski*, Miroslaw Chmiel**

Silesian University of Technology, Department of Digital Systems, Adademicka Str. 16, 44-100 Gliwice

Abstract. The article presents the concept of dual-core bit.WORD programmable logic controller central processing unit. The idea is based on the use of two microprocessor cores, one fast and low-power for bit operations and one computationally efficient for word operations. The main contribution is a comparison of efficiency of the bit core and the word core. The paper presents results for logic utilization, core clock frequency and power consumption in order to undertake further implementation work related to the development of the bit.WORD unit.

Key words: Programmable Logic Controllers (PLC); Field Programmable Gate Arrays (FPGA); Central Processing Units (CPU); bit.WORD unit; RISC-V

1. INTRODUCTION

Programmable Logic Controllers (PLCs) occupy an undisputed position in modern industry. They represent well-thought-out designs that are safe and meet industry expectations to date. However, the fourth industrial revolution, Industry 4.0 [1], sets completely new goals and requirements for PLCs. And Industry 5.0 is already just about the corner. Changes must be made in this well-established area. On the one hand, programmable logic controllers are being embraced as a standard, while on the other hand, a flexible approach and new designs that keep up with the new requirements are needed.

It takes a lot of effort to design a dedicated microprocessor. It is much simpler to design a PLC using a standard microprocessor or microcontroller. In fact, the resources of a standard microprocessor/microcontroller are not aligned with the IEC 61131-3 standard—the Instruction List (IL) of a standard microprocessor is seriously mismatched with what the standard defines. By developing a specialized microprocessor that is compatible at the machine level with, for example, the IL language defined in IEC 61131-3, these drawbacks can be eliminated [2, 3]. For example, the instruction pipeline can be introduced for better response time [4]. Also, a multiprocessor programmable controller can be designed [5], so that multiple processors runs control tasks in parallel.

Another solution is presented in [6]. This CPU is capable of performing logic operations up to 32-bit Boolean data types. The developed processor is very flexible thanks to its architecture: data memory can be addressed as bit/byte/word/dword. Additionally, various devices can be connected to the processor using the APB AMBA bus. A similar approach was adopted by the authors of [7]. The presented in [7] IL processor is capable of handling 1-bit, 8-bit, 16-bit, and 32-bit data types, but also single precision floating-point operation. It also directly performs instructions with a parenthesis modifier for all supported data types and facilitates the fast execution of a code containing Boolean expressions.

Tremendous opportunities for implementing digital systems are offered by Field Programmable Gate Arrays (FPGAs). FPGAs are customer-configurable integrated circuits. The main structure of an FPGA contains programmable logic and programmable interconnections, allowing flexible implementation of designed functions. The logic structure allows the implementation of true concurrency, which is a major advantage over microprocessors. In addition, microprocessors can be implemented in FPGAs as hard- or soft-cores, so a system-on-a-chip can be built.

Moreover, the FPGAs enable direct synthesis of reconfigurable logic controllers [8], where the program is compiled to hardware structure with a massive parallel processing using a method that automatically allocates resources and operations.

FPGA-based methods for acceleration and implementation are also used for various industrial control blocks, such as PID controllers, modulators, etc. The purpose of [9, 10] is to propose a new method for implementation of the PID algorithm. Tuning rules for PID and PI-PI servo controllers, developed using a pole placement approach with a multiple pole are shown in [11]. An idea for a hardware realization of the space vector modulation in a direct matrix converter is proposed in [12]. It proposes a novel hardware algorithm for a matrix converter switch control. Contrary to the existing hardware methods, the presented technique does not require neither specialized modeling tools, nor advanced converters to achieve the hardware representation. [13] presents a fast direct pulsewidth modulation (PWM) algorithm for the conventional matrix converters. All PWM duty cycle calculations are performed in one cycle by an atomic operation.

FPGA circuits are often used as devices in which concurrent calculations are implemented, but they are also ideal for all kinds of accelerators, nowdays especially in the field of artificial intelligence. This was facilitated not only by the possibility of implementing concurrent processing, but also by the ex-

*e-mail: rczerwinski@polsl.pl
**e-mail: mchmiel@polsl.pl

traordinary ease of creating a hardware-software systems with a soft microprocessors. In [14] a high accuracy hardware implementation of the hyperbolic tangent and sigmoid activation functions is presented. The high level synthesis methodology is used.

Methods of supporting artificial intelligence are often combined with the development of dedicated microprocessors and the use of the open RISC-V specification. Hyperdimensional computing, based on the novel 16-bit fixed-point HDC model enabling training at the Edge is presented in [15]. The paper [16] aims to bridge the power-performance gap using the energy efficiency-centric RISC-V ecosystem, while the [17] aims to integrate the simplicity of structured sparsity into vector execution to speed up the corresponding matrix multiplications. Initially, the implementation of structured-sparse matrix multiplication using the current RISC-V instruction set vector extension was comprehensively explored. Recent progress on Post-Training Quantization and Quantization-Aware Training has shown that the key to high energy efficiency lies in executing deep learning models with low- (8- to 5-bit) or ultra-low- (4- to 2-bit) precision. A hardware-software co-designed architecture that enables RISC-V processors to efficiently compute arbitrary mixed-precision DNN kernels, supporting all data size combinations from 8- to 2-bit is presented in [18].

The idea of dual-core units is not new. Work on such units began as early as the 1980s [19, 20, 21]. This idea was also developed in form of microprocessing units and co-processors instead of ASIC-based PLC [21]. Nevertheless, the topic of developing multi-core structures is still active. Industry 4.0 and 5.0 are placing ever-increasing demands on CPUs. One of these demnads is increased system security [22]. A dual-core unit can therefore be used for cros-detection and redundancy [23]. Nowadays, the increasing scale of complexity of systems results in very fast changing input signals to the PLC. This makes the response time of PLC an important issue [24, 25]. Concurrency can also be used to design hardware-accelerated PLC modules [26]. Industry 4.0/5.0 enforces also the issue of robust communication [27].

Thanks to the proliferation of FPGAs, building a dedicated central processing unit is now not a problem. However, the speed that modern microprocessor systems offer means that they are also readily used to construct PLCs [28, 29] or even to build framework for using general purpose systems [30, 31]. It is also possible to combine these two ideas (to use general purpose microprocessor and FPGA) and to construct a microprocessor to be used as a soft-processor central processing unit in an FPGA. One of the more dynamically developed architectures today is the RISC-V. The idea of RISC-V is compared to the Linux operating system but in the hardware world. Linux became one of the most popular systems available on the market. The reason for this was the open source license, allowing free use and modification of the source code. Nowadays, virtually all electronic devices have an embedded microprocessor. Therefore, the new microprocessor architecture based on such license is an attractive alternative to closed structures created by companies specializing in microprocessor system design. It should be clearly emphasized that RISC-V ISA is a specification that must be interpreted as the software-visible interface to a wide variety of implementations.

The article presents the concept of a dual-core bit.WORD programmable logic controller central processing unit. However, the main contribution is a comparison of the bit and the WORD cores based on RISC-V ISA. The paper presents results for logic utilization, core clock frequency and power consumption. The experimental results presented justify further work on core integration and PLC implementation according to the bit.WORD concept.

2. TWO-CORE PLC

2.1. Basics

PLCs process both binary and numerical signals. The boundary between processing both types of data can be very precisely defined. The possibilities of cooperation of two units will be presented in the following example.

In the PLC system/circuit presented in Fig. 1 pressing the Start button with inactive states on the other binary inputs, together with the presence of an appropriate pressure, should switch the Start_Up signal on, and after a certain time, actived the operation of the actuator.

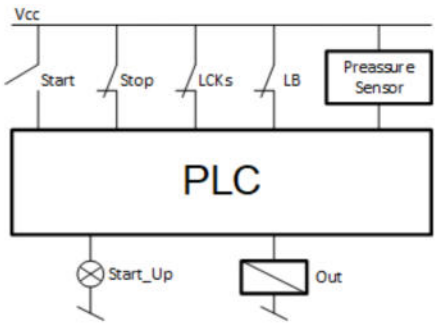

Fig. 1. A PLC system with binary and numeical inputs

The basic version of the control program, written for a standard PLC CPU, written in the LD language looks like the one shown in Fig. 2, while its IL version is shown in Listing 1. IL language is preferable in this case, as it ensures clear division between bitwise and numerical operations is quite clear. Therefore, in the following examples in the IL form will be used.

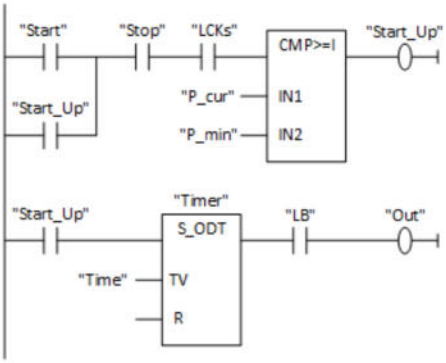

Fig. 2. LD control program

Listing 1: Example program in IL—serial version.

```
LD      "Start" //bit
OR      "Start_Up"
AND     "Stop"
AND     "LCKs"
LD      "P_cur" //WORD
LD      "P_min"
AND GE  //instruction to avoid jumping;
        //it multiplies the bit CR
        //by the result of the comp action
ST  "Start_Up"
LD  "Time"
ST  "Timer".PT
LD  "Timer".Q
AND "LB"
ST  "Out"
```

The „AND GE" instruction here is to prepare data for two-core PLC. It will be utilized in the next example.

Next, the program may be adapted for a two-processor CPU unit with the possibility of concurrent operation of individual microprocessors. The program should be equipped with additional commands that will allow the transfer of information between processors. Such information will always be a two-state variable. program serially executed by two cores (bit and word) with additional commands to transfer information between microprocessors is shown in Listing 2. The program for concurrent execution in both microprocessors cores is presented in Listing 3.

Listing 2: The program with additional commands to transfer information between microprocessors.

```
LD      "Start"
OR      "Start_Up"
AND     "Stop"
AND     "LCKs"
LD      "P_cur"
LD      "P_min"
GE
ST      F_Bb1   //from WORD to bit
AND     F_Bb1
ST  "Start_Up"
ST      F_bB1   //from bit to WORD
LD      F_bB1
ST      "Timer".IN
LD  "Time"
ST  "Timer".PT
CAL     "Timer"
LD  "Timer".Q
ST      F_Bb2
LD      F_Bb2
AND "LB"
ST  "Out"
```

Listing 3: Program with concurrent program execution by two microprocessors.

```
//bit microprocessor
LD      "Start"
OR      "Start_Up"
AND     "Stop"
AND     "LCKs"
AND     F_Bb1
ST  "Start_Up"
ST      F_bB1
LD      F_Bb2
AND "LB"
ST  "Out"

//WORD microprocessor
LD      "P_cur"
LD      "P_min"
GE
ST      F_Bb1
LD      F_bB1
ST      "Timer".IN
LD  "Time"
ST  "Timer".PT
CAL     "Timer"
LD  "Timer".Q
ST      F_Bb2
```

The program in Listing 3 includes additional commands, yet its execution time on each processor is shorter than that of the program in Listing. 1. While this approach yields better performance even when executed serially, it requires both processors to run constantly, which increases energy consumption.

2.2. The bit.WORD PLC architecture

The example given in Section 2 shows that splitting the control program into two units may be possible and make sense. This led to arose the idea of the bit.WORD central processing unit in which one of the processors has a single-bit and the other a WORD architecture. The generalized structure of the proposed solution is shown in Fig. 3.

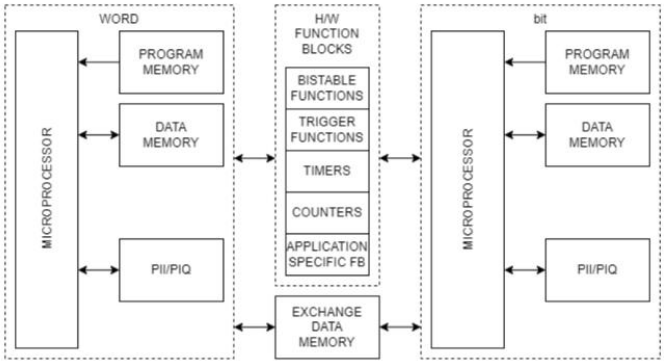

Fig. 3. The idea of the bit.WORD programmable logic controller

Splitting the tasks between two microprocessors makes sense because the design of a bit microprocessor is small in relation to multi-bit processors, potentially much faster, as confirmed by preliminary test results and the literature, and the power dissipation will be less than in high-performance multi-bit microprocessors. The use of the proposed design will make it possible to exploit the potential of each unit.

The components of the designed unit will be a full 64-bit processor, the basic feature of which is high versatility, and its depleted version, which is optimized for processing single-bit

variables. As shown in the papers [32, 33], such processors can be equipped with appropriate memories.

Each microprocessor is able to autonomously execute commands intended for it. It also has the ability to communicate with its peripherals. An additional important component is the exchange memory, through which the processors are able to exchange information necessary for control among themselves. This memory is single-bit memory because the information exchanged between processors is always of this nature. A WORD processor sometimes requires bit information from a bit microprocessor, but a bit CPU will never need word information from WORD unit.

The realization of the structure presented in this section is allowed by the possibility of configuring different types of memory in FPGAs—including the required by this solution true dual-port RAM memory [34] accessible by each processor [from its own side]/[through its dedicated port]. The concurrent operation of modules in the FPGA system also enables completely independent operation of functional blocks, which perfectly support the operation of the entire PLC controller. Research on the architectures of individual functional modules has been presented, among others, in [35, 36, 37, 38].

As noted in the introduction, the development of a microprocessor dedicated to the IL language can yield tangible results in terms of a good match between the actual operations performed and the IL language operations. Another approach is to use a standard microprocessor. In this case, a microprocessor conforming to the RISC-V Instruction Set Architecture (ISA) is proposed. The creation of a bit and WORD structure of RISC-V based microprocessors will make it possible to test the possibilities offered by the idea of building a dual-processor PLC CPU. In the following section, the core designs and the results of the comparison of their performance will be presented.

3. RISC-V IMPPLEMENTATIONS

RISC-V is an open instruction set architecture based on Reduced Instruction Set Computing (RISC). The RISC-V ISA basically consists of two parts:

- unprivileged specification [39]—it is defined in such a way such as to remove any dependence on specific features of the microarchitecture, such as cache line size, or on details of the privileged architecture, such as page translation; this is intended to both simplify and allow maximum flexibility for alternative microarchitectures or alternative privileged architectures,
- privileged specification [40]—covers all aspects as privileged instructions as well as additional functionality required for running operating systems and attaching external devices.

The ISA specification for the 64-bit version of the RISC-V microprocessor (RV64I) has been ratified. The situation is completely different for microprocessors below 32-bit wide. The 32-bit specification is basic, and it is possible to include the specification marked as C, developed as a compression scheme for the 16-bit version. Single-bit applications remain completely outside the scope of the organisation's analysis for obvious reasons.

A 1-bit microprocessor can be implemented in several ways: as a completely separate unit, as a modification of the C extension of the RV64I, or as an independent extension development. The latter two approaches are not of interest to us, as they do not allow for independent clocking of units.

After analysing the solutions, the cores were implemented as independent units capable of performing tasks in parallel. The cores have separate address buses as well as control and data buses. Parallel operation allows the entire bit.WORD model to operate most effectively by utilising the maximum capabilities offered by a fast bit processor and a universal WORD processor. The bit microprocessor is based on the RV32I/64I specification.

3.1. Pipelining

Both microprocessors, bit and WORD, were designed according to the same five-stage pipeline processing scheme:

1. the Instruction Fetch (IF) module,
2. the Instruction Decoding (ID) module,
3. the Instruction Execution (EX) module,
4. the Memory Control Module (MEM),
5. the Write Back (WB) module.

The listed main modules form a typical pipeline architecture/-solution [41]. Each block performs its function in one clock cycle signal. Thus, a standard instruction is executed in five clock cycles. The block diagram of the designed core is shown in Fig. 4. First, the instruction is fetched from program memory (by IF), then it is decoded (in ID), the operands are fetched and the result is calculated (by EX), and finally the result is stored in the appropriate location (MEM or WB).

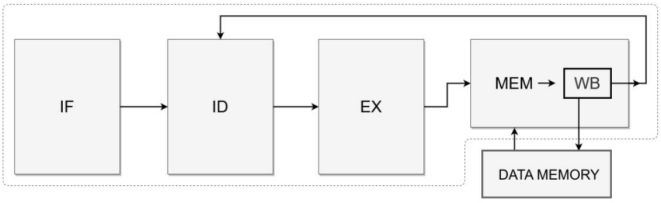

Fig. 4. The block diagram of the designed core

As is well known, the solution with pipeline can be problematic, because the processor may have not enough time to complete the operation whose result is used in the next what may give invalid results. Avoiding this problem is possible through the use of modules that will detect potential error conditions (hazards) and offset them through data transfer and appropriate data control in the pipeline.

The following subsections will discuss the 5-stage pipeline modules implemented in a WORD processor, and then the differences between bit and WORD implementations will be presented.

3.2. Instruction Fetch

Instruction Fetch module is responsible for loading instructions written in machine code (Fig. 5). In the case of the RISC-

V specification, the 32-bit instrucion encoding is defined. The Program Counter (PC) is responsible for the addressing. It indicates the address of the program memory, where the instruction to be executed is located. After fetching the instruction, the address is incremented so that it points to the next instruction.

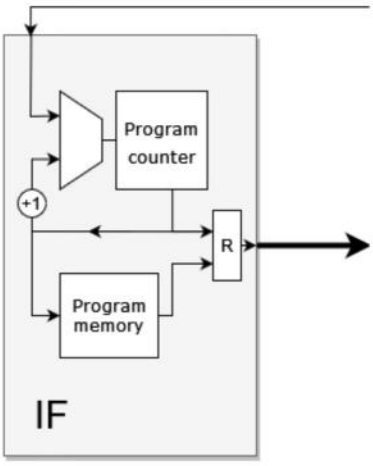

Fig. 5. The instruction fetch module

By default, the address is incremented after each instruction is read. However, the program may contain branch instructions that directly affect the PC address by changing its contents as required. The choice between the incremented and the directly set values is made using a multiplexer. The current address of the instruction counter and the machine code of the instruction read at this address are passed to the output of the IF module.

3.3. Instruction Decode

The main task of the instruction decoding module ID is to properly control the core so as to create a correct path for the data. In addition, the general purpose registers store temporary values on which arithmetic or logical operations are often performed. Depending on the instruction, two values can be read from the registers, which are passed to the module's output alongside signals from the control unit. An important component is also the constant value immediate generator, which extracts the values encoded in the instruction's machine code (Fig. 6).

The Control Unit is responsible for the correct setting of all control signals that control the progress of the currently executed instruction. For the sake of preserving the correctness of pipelined processing, these signals are propagated to subsequent blocks depending on their destination, through the parallel registers placed in the modules (pipeline registers). The signals are mainly set based on the seven least significant bits of the instruction, which constitute the opcode [41, 39].

The Immediate Generator deals with the restoration of the data contained in the instruction's machine code. When decoding an immediate argument, the module expands it to 64-bit size taking into account the most significant bit, which is

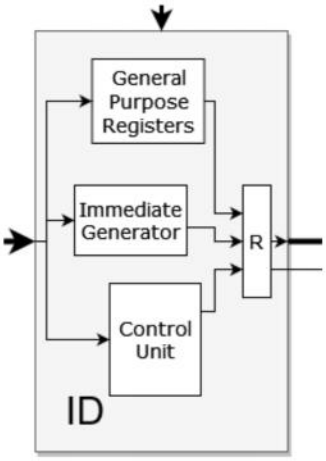

Fig. 6. The instruction decode module

the sign bit. Based on the information from the control unit, the type of instruction is known, and the corresponding result is passed to the output.

In the RISC-V idea, general purpose registers are the primary place to store temporary data i.e. data on which operations are mainly performed. There are 32 64-bit registers created in the core. There are 32 registers with a 64-bit width created in the core. Each of them is accessed by pointing to a 5-bit address. Values can be read from up to two addresses in parallel, which is used in some instructions, which is used in some instructions. After reading, the values are passed to the input of the execution module EX. By pointing to the corresponding target register address, the result can also be transferred from the memory controller module to the register block and written. Writing to and the reading from registers are performed synchronously.

3.4. EXecution

The Execution module (Fig. 7) is the block where the main calculations involved in the execution of instructions are carried out. Its primary element is the Arithmetic-Logic Unit ALU. In addition, the block contains a unit responsible for controlling the execution of jumps/branches. The results of the operations and all the information related to the archiving of the result, such as the target register address, memory address, data width, read/write signals, are transferred to the pipelined registers.

The ALU, is the main place in the processor where arithmetic and logical operations are performed. The data can be taken from various sources: from general purpose registers, from the immediate generator, or can be taken as fixed values, e.g. 0 for a comparison instruction, or 4 for jump instructions. Based on the signals from the control unit, the data on which the operation is to be performed as well as the operation itself are selected. Along with the output data, two signals are produced by the ALU—flags providing information about the currently generated result. The carry flag is a flag for exceeding the range of the data, and the zero flag indicates the zero result of the operation performed.

The Branch Control module is the module responsible for controlling the core for any jump operations. Two types of

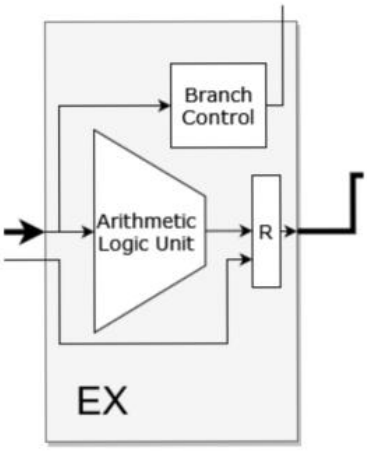

Fig. 7. The execution module

jumps are distinguished: conditional (branch) and unconditional (jump) [39]. In addition, unconditional jumps are divided into: jump with and without link register. The type of jump is performed depends on the corresponding configuration of the input signals. For an unconditional jump, the operation is performed immediately, while for an uconditional jump the fulfillment of the jump conditions is checked. Whenever a jump (unconditional or conditional with condition met) is to be performed, the module transmits the address of the next instruction to be executed and the signal controlling the multiplexer to the IF module. This data is passed between modules bypassing the pipeline registers, so execution of the jump is possible even in the next active clock edge.

3.5. MEMory and Write Back

Memory controller is a module responsible for managing data memory. Data written to or read from memory can be of different widths: byte, word, double and quadruple word (8, 16, 32, 64 bits, respectively). Based on the signal from the control unit, the size of the data is determined and a read/write mask is created. The result obtained from the arithmetic-logic unit indicates the memory address, and an 8-bit mask is derived from the three least significant bits of this address. The stored value comes from the working register, and once the mask is defined, the data is shifted so that the bits coincide with the target area. Table 1 shows this idea.

The module contains a muliplexer responsible for selecting the data to be written to the write-back register: it can be data from memory or the result from the arithmetic-logic unit. This is the final stage of pipelined processing.

3.6. Anti-hazard module

The modules presented in the previous sections are the basic elements of the RISC-V microprocessor core developed. To support its proper operation, additional elements have been developed: control hazard detection module and forwarding unit.

The task of the control hazards detection module is to detect risks associated with dependencies between instructions and eliminate their effects. The designed CPU implements pipelined processing in five phases. There may be situations in which the pipeline is filled with the wrong instruction (e.g. when conditional branch is to be realized). The implementation is based on 'stalling' and 'flushing' the pipeline. The method of 'stalling' the pipeline consist-in stopping the entry of new data in successive clock edges to the designated pipeline register. The idea behind this method is to prevent the execution of an instruction that would overwrite a register or memory cell or, in the case of a jump instruction or overwrite the program counter. 'Stalling' the possibility of overwriting one pipeline register will result in the re-execution of operations on the same data in the next clock cycle. For this purpose, the pipeline 'flushing' method allows resetting all control signals in the pipeline register, so that the result of the executed operation will be ignored by the rest of the modules in the microprocessor. The control hazards detection module is not directly involved in pipeline processing. Its task is limited to the control of the pipeline registers i.e. the control hazards module's outputs signals are responsible for 'stalling' and 'flushing' the pipeline.

The forwarding unit is responsible for redirecting data within a pipeline. It is a hardware solution designed to resolve data conflicts (data hazards) in the microprocessor pipeline. Its main purpose is to ensure that the right data gets to the arithmetic-logic unit in the execution module. The data redirection module will cancel the delays caused by holding up (e.g. NOP instructions) the pipeline in the microprocessor when a data conflict occurs. The unit was designed to allow the arithmetic-logic unit to access the result data before it is written to the general purpose registers by the WB module. The data redirection module also processes information about the occurrence of branch instructions related to control hazards.

3.7. Single-bit RISC-V based microprocessor

The aim of creating a bit unit is not only to reduce the high complexity of the microprocessor, but also to explore new efficiency and optimisation paths made possible by using an architecture based solely on single-bit operations. The hazard detection or data forwarding modules have not been implemented in the bit unit. Implementing these modules could further improve the performance of the single-bit core, but would complicate its design, which is intended to be as simple as possible. Only the simplest solution, the so-called 'stalling', has been used. The pipeline is paused until the previous instruction releases the required resources or performs the required read/write from/to memory.

The instruction list has been adapted to those arithmetic and logic instructions that make sense for single-bit operations (AND, OR, XOR, ADD and SUB; also with immediate data). Branch instructions and data transfers have also been implemented. A word of explanation is required for the ALU, which in this case is single-bit. It makes the PC-relative branch from determining the address correctly. Therefore, an extra 32-bit adder has been added, to allow jumps in the full address space.

Table 1. The permissible configurations of the memory cell mask

	Mask			
	Byte	**Half-word**	**Word**	**Double-word**
0b000	0b00000001	0b00000011	0b00001111	0b11111111
0b001	0b00000010			
0b010	0b00000100	0b00001100		
0b011	0b00001000			
0b100	0b00010000	0b00110000	0b11110000	
0b101	0b00100000			
0b110	0b01000000	0b11000000		
0b111	0b10000000			

3.8. Data exchange module

The WORD and bit cores do not have dedicated commands enabling direct signal exchange between them, therefore any data exchange takes place exclusively via memory. The data exchange unit with memory supports four modules (Figure 8):

- the direct memory access (DMA) mechanism which supports peripheral devices,
- the interrupt controller which is responsible for direct data exchange during programme interrupts,
- bit CPU,
- word CPU.

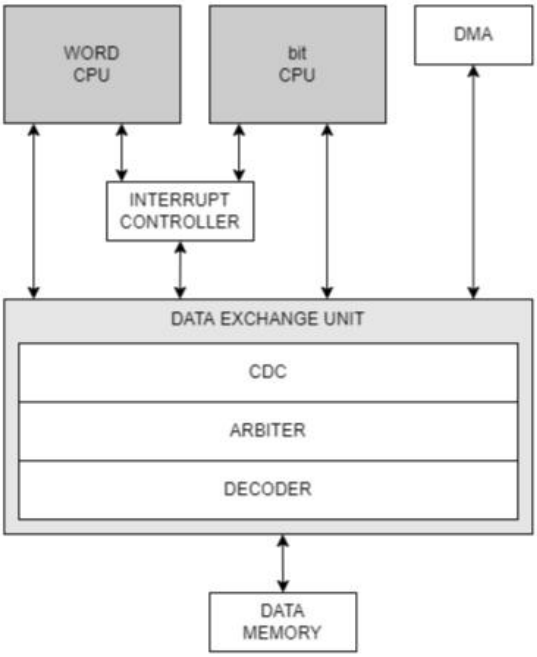

Fig. 8. The data exchange moduleconnectivity

Each of the four modules (as a master) can read and write data from and to the RAM, which acts as a shared slave module in the system. In order to increase throughput and reduce the waiting time for data access, a dual-port RAM has been implemented.

The main element of the data exchange unit is the arbiter, whose task is to assign priorities to individual requests from external modules, ensuring fair access to the bus and minimising the risk of collisions, which increases the efficiency and stability of communication in the system. The main role of the decoder in this system is to manage access to memory ports. Dual-Port RAM allows the use of two independent inputs, enabling simultaneous operations. The decoder must correctly interpret the address of each transaction and direct it to the appropriate slave module, ensuring correct operation and efficient data exchange between system components. Due to the fact that the system has modules, each of which can operate in a separate time domain, it is necessary to implement Clock Domain Crossing (CDC) modules. This synchronisation is crucial to ensure correct data exchange between modules operating with different clocks and to prevent metastability phenomena.

4. RISC-V IMPLEMENTATION RESULTS

The developed microprocessor cores developed in Verilog HDL [42] were implemented in the FPGA structure of Xilinx's XC7A100T-1CSG324C Artix-7 family FPGA [43], characterized by low power consumption at a high maximum clock frequency.

Verilog plays a key role in the design of dedicated digital circuits, enabling formal description of the structure and behaviour of hardware at various levels of abstraction. Verilog is commonly used for modelling combinational and sequential logic, creating RTL modules, and integrating system components, making it the foundation of modern ASIC and FPGA design flows. Its unambiguous hardware semantics allow for direct mapping of code to logic resources, which is particularly important in projects with high time, power and area requirements. SystemVerilog [44] extends the capabilities of Verilog by introducing advanced language constructs that support the design, verification and maintenance of complex digital systems. In the context of dedicated digital circuits, SystemVerilog enables a consistent combination of architecture description and advanced functional verification. The SystemVerilog language was used in the implementation of the developed cores, both for design and functional verification. The description of the circuit remained at the RTL level and closely reflects the structure of the designed cores. A subset of the SystemVerilog language was used, which includes synthesisable elements

of the structural description (convergent with the Verilog language). All features of the SystemVerilog language (e.g. interfaces, enumerated types, structures, classes and assertions) were used in the verification.

The implementation and analysis of the resources consumed by each module were carried out in VIVADO by Xilinx. The results are shown in Figs. 9-11.

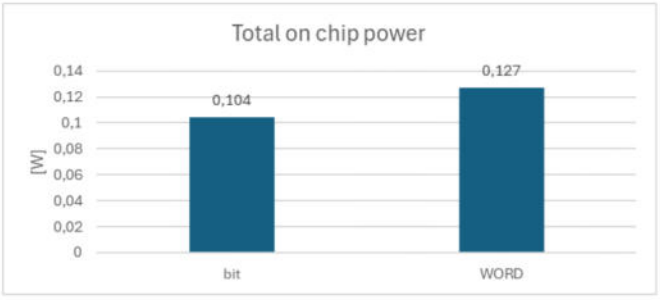

Fig. 9. Power analysis of the bit and WORD RISC-V cores

Because the bit processor processes only one bit variables, it requires far fewer resources than the word processor, as clearly shown in Fig. 10. As expected, the largest difference, reaching up to thirty times, occurs for the use of LUTRAM. The use of LUTs differs by more than ten times, while the use of flip-flops by almost seven times less for the structure of the bit processor.

This, of course, translates into energy consumption. The bit unit consumes 82% of the energy required by the word processor, as shown in Fig. 9.

However, what is most significant from the perspective of speeding up a dual-processor unit compared to a classic single-processor unit is the maximum clock frequency of the resulting structure. And we see in Fig. 11 that the bit processor can operate at more than three times the frequency of the word processor, with the maximum frequency in the FPGA used reaching more than 300 MHz.

5. ANALYSIS

The instruction list of RISC-V microprocessor is not a native list of an IL-based PLC compliant with the Standard. This may result in a certain excess of instructions developed by the unit in relation to the IL code.

In PLCs, atomic instructions are usually commands that enables loading data into CR or memory. By atomic, we mean those that execute in the shortest time. This is the case, for example, in [45, 46, 47, 48, 49] controllers. The concept of the controller presented here is similar, because the LD and ST instructions have their direct translation in RISC-V instructions.

Access to memory in RISC-V is performed using a base address that must be prepared, so at least once you need to use an additional instruction to load a given immediate value („lui"). Finaly, the LD and ST are covered by two RISC-V instructions, as shown in the example in the Listing 4 and take two clock cycles fo each.

Listing 4: Example load and store instructions in IL vs. RISC-V.

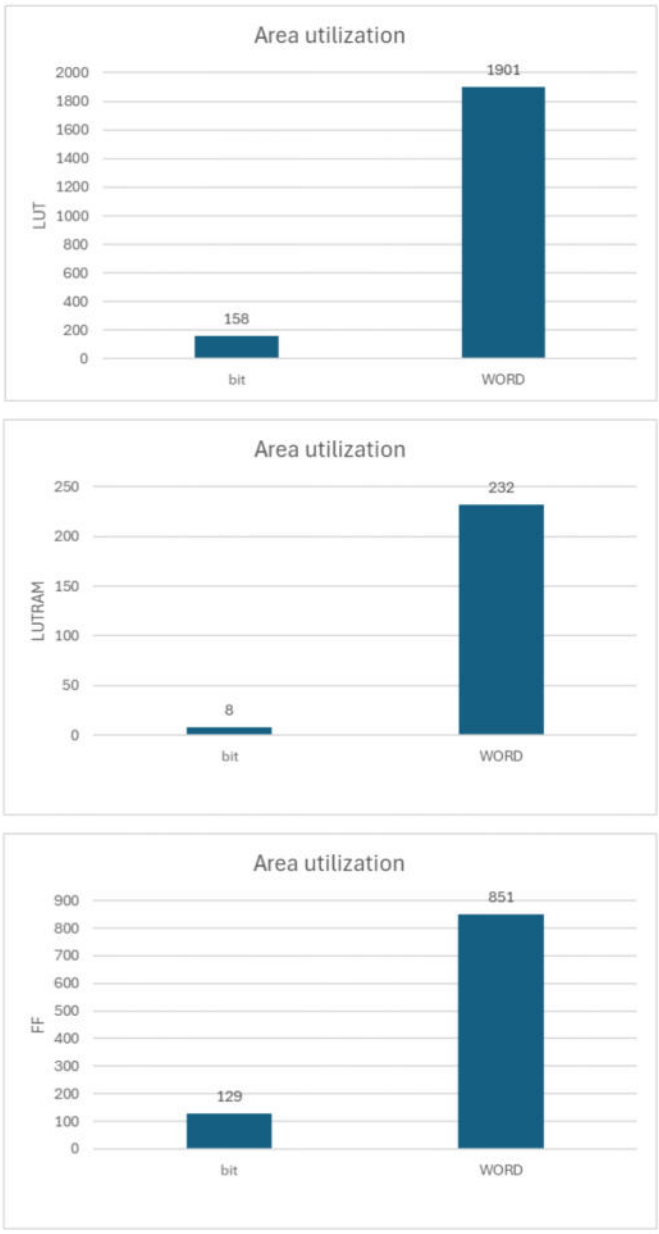

Fig. 10. Area analysis of the bit and WORD RISC-V cores

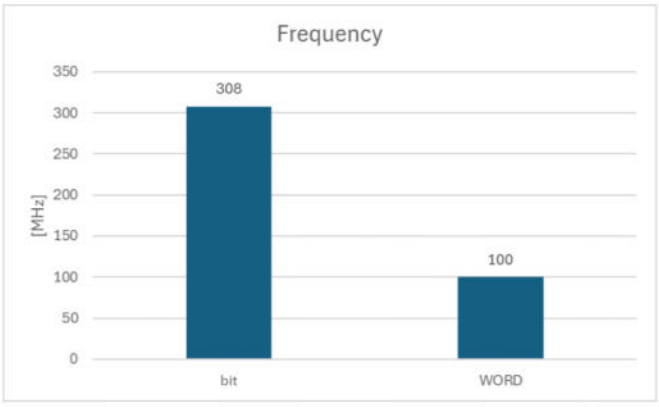

Fig. 11. Max frequency of bit and WORD RISC-V cores

```
;IEC 61-131-3
LD MEM

;RISC-V
lui rd,imm
ld rd,offset(rs1)

;IEC 61-131-3
ST MEM

;RISC-V
lui rd,imm
sd rs2,offset(rs1)
```

A word of commentary is required. An integral part of a classic central processing unit (based on the IEC Standard) is the CR register. In the case of the RISC-V microprocessor, there is no dedicated accumulator, and operations can use any general purpose register. This, in turn, can bring significant benefits because there will be no need to perform multiple copy operations to a single register, as is the case when CR is used. Nevertheless, RISC-V microprocessors (usually in RISC-class microprocessors) perform arithmetic operations on registers, and memory accesses are only performed by the instructions shown in the Listing 4.

Listing 5: Example load and store instrukctions in IL vs. RISC-V.

```
;(MEM1)+(MEM2)->(Mem3)

;IEC 61-131-3
LD MEM1
ADD MEM2
ST MEM3

;RISC-V
lui x10, 0x80001 ;base addr
ld  x11, 0(x10)
ld  x12, 8(x10)
add x13, x11, x12
sd  x13, 16(x10)
```

In the case of Listing 5, operations performed on memory markers show the advantage of IL, but when more arithmetic commands need to be performed on the same data, the advantage of a three-argument machine becomes obvious.

It must be admitted that comparing the developed bot.WORD CPU with other units (including those natively supporting IL compliant with the IEC Standard) is a complex matter. On the one hand, we have a bit processor that performs operations three times faster than a WORD processor, but it requires an excess of data preparation operations in registers. On the other hand, in relation to the Standard, there is no need to continuously copy data using the CR register, where it is temporary. A complex benchmarking process is required.

6. CONCLUSIONS

This paper presents the concept of a bit.WORD dual-processor unit based on the RISC-V microprocessor specification. One part of the microprocessor performs operations only on BOOL-type variables, while the other handles WORD-type variables. Such a concept allows some parallelism of operations, e.g. bit and word data could be written to a function block concurrently.

The article presents a custom implementation of the concept using RISC-V microprocessor and the single-bit microprocessor built as an adaptation of the ISA specification for the RISC-V microprocessor. The developed components were described in the Verilog hardware description language and were implemented in an FPGA using Xilinx VIVADO.

The most important contribution of the paper is the comparison of the two cores in terms of logic occupancy, clock speed and power dissipation. Because, as shown, adding a dedicated bit processor costs few resources and energy, such a concept is worth further integration and development.

In the context of increasing requirements for Industry 4.0/5.0, the use of a RISC-V microprocessor as the core in a PLC controller is justified. The microprocessor achieves high performance, as shown by the results of various studies. It is also widely used in many solutions. This paper shows that it also works well in the context of PLC controller design. However, it should be clearly stated that the instruction set is not native, which is a certain disadvantage. Secondly, the design of the RISC-V microprocessor is typical for a general-purpose unit. This constitutes a certain implementation overhead, which is not beneficial in the normal operation of the controller as a control system. The desire to utilise the full capabilities of the RISC-V processor in a dual-core bit.WORD unit also requires overhead related to the implementation of data exchange mechanisms, maintaining coherence, and developing mechanisms for accessing shared resources.

However, building a system based on RISC-V for the operating system behind the basic operation (classical control) already offers significant benefits. The authors believe that, the use of RSC-V in the core of a large system is justified, while in the case of a small system, it makes more sense to develop a dedicated dual-core CPU that natively supports the IL language.

ACKNOWLEDGEMENTS

This work was supported by the Polish Ministry of Science and Higher Education funding for statutory activities.

REFERENCES

[1] S. Klaus, *The Fourth Industrial Revolution*. Portfolio Penguin, 2017.

[2] S. Carrillo, A. Polo, and M. Esmeral, “Design and implementation of an embedded microprocessor compatible with IL language in accordance to the norm IEC 61131-3,” in *Proceedings of the IEEE International Conference on Reconfigurable Computing and FPGAs. ReConFig 2005*, 28–30 September 2011, pp. 28–30. DOI: 10.1109/RECONFIG.2005.14

[3] M. Okabe, “Development of processor directly executing IEC 61131-3 language,” in *SICE Annual Conference, The University of Electro-Communications, Tokyo,*

Japan, 20–22 August 2008, pp. 2215–2218. DOI: 10.1109/SICE.2008.4655032

[4] S. Rudrawar and M. Patil, "Design and implementation of FPGA based high performance instruction list (IL) processor," *IOSR Journal of Electronics and Communications Engineering (IOSRJECE)*, vol. 1, no. 4, pp. 38–45, 2012. DOI: 10.9790/2834-0143845

[5] Z. Hajduk, B. Trybus, and J. Sadolewski, "Architecture of FPGA embedded multiprocessor programmable controller," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 2952–2961, 2015. DOI: 10.1109/TIE.2014.2362888

[6] P. Mazur, R. Czerwinski, and M. Chmiel, "PLC implementation in the form of a System-on-a-Chip," *Bulletin Of The Polish Academy Of Sciences - Technical Sciences*, vol. 68, no. 6, pp. 1263–1273, 2020. DOI: 10.24425/bpasts.2020.135386

[7] Z. Hajduk, "IEC61131-3 instruction list language processor for FPGAs," *Electronics*, vol. 12, no. 19, 2023.

[8] A. Milik and E. Hrynkiewicz, "Synthesis and implementation of reconfigurable PLC on FPGA platform," *International Journal of Electronics and Telecommunications*, vol. vol. 58, no. No 1, 2012. DOI: http://journals.pan.pl/Content/87098/PDF/12.pdf

[9] J. Kulisz and M. Chmiel, "Implementation of a PID controller in hardware using floating point arithmetic," *Przegląd Elektrotechniczny*, vol. 101, no. 12, p. 8, 2025.

[10] J. Kulisz and F. Jokiel, "A hardware implementation of the PID algorithm using floating-point arithmetic," *Electronics*, vol. 13, no. 8, 2024. DOI: https://www.mdpi.com/2079-9292/13/8/1598

[11] A. Bożek and L. Trybus, "Tuning PID and PI-PI servo controllers by multiple pole placement," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 70, no. 1, p. e139957, 2022. DOI: http://journals.pan.pl/Content/121938/PDF-MASTER/2369_corr.pdf

[12] R. Wiśniewski, G. Bazydło, and P. Szcześniak, "Low-cost FPGA hardware implementation of matrix converter switch control," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 7, pp. 1177–1181, 2019.

[13] P. Szczepankowski, W. Sleszynski, and T. Bajdecki, "A direct modulation for matrix converters based on the one-cycle atomic operation developed in Verilog HDL," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 4, pp. 3303–3312, 2022.

[14] Z. Hajduk, "Hardware implementation of hyperbolic tangent and sigmoid activation functions," *Bulletin of the Polish Academy of Sciences Technical Sciences*, vol. 66, no. No 5, pp. 563–577, 2018. DOI: http://journals.pan.pl/Content/108428/PDF/563-578_00557_Bpast.No.66-5_31.10.18_K1.pdf

[15] S. A. Wasif, M. Wael, P. R. Genssler, E. Azab, M. Mashaly, M. A. A. E. Ghany, and H. Amrouch, "Domain-specific hyperdimensional RISC-V processor for Edge-AI training," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 10, pp. 5825–5838, 2025.

[16] O. Khan, "A data prefetcher-based 1000-core RISC-V processor for efficient processing of graph neural networks," *IEEE Computer Architecture Letters*, vol. 24, no. 1, pp. 73–76, 2025.

[17] V. Titopoulos, K. Alexandridis, C. Peltekis, C. Nicopoulos, and G. Dimitrakopoulos, "Optimizing structured-sparse matrix multiplication in RISC-V vector processors," *IEEE Transactions on Computers*, vol. 74, no. 4, pp. 1446–1460, 2025.

[18] J. Fornt, E. Reggiani, P. Fontova-Musté, N. Rodas, A. Pappalardo, O. S. Unsal, A. Cristal Kestelman, J. Altet, F. Moll, and J. Abella, "Mix-GEMM: Extending RISC-V CPUs for energy-efficient mixed-precision DNN inference using binary segmentation," *IEEE Transactions on Computers*, vol. 74, no. 2, pp. 582–596, 2025.

[19] Z. Getko, "Programmable systems of binary control," *Elektronizacja*, 1983, in Polish.

[20] J. Donandt, "Improving response time of programmable logic controllers by use of a boolean coprocessor," in *Proceedings. VLSI and Computer Peripherals. COMPEURO 89*, 1989, pp. 4/167–4/169. DOI: 10.1109/CMPEUR.1989.93466

[21] N. Aramaki, Y. Shimokawa, S. Kuno, T. Saitoh, and H. Hashimoto, "A new architecture for high-performance programmable logic controller," in *Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control, and Instrumentation (Cat. No.97CH36066)*, vol. 1, 1997, pp. 187–190 vol.1. DOI: 10.1109/IECON.1997.671044

[22] T. Wilkening, J. Holtz, and J. O. Krah, "Mixed-critical control architecture for Industry 5.0," in *2024 4th International Conference on Smart Grid and Renewable Energy (SGRE)*, 2024, pp. 1–6. DOI: 10.1109/SGRE59715.2024.10428840

[23] Z. Du, G. Cheng, M. Wang, R. Sun, and K. Liu, "Research on safety programmable controller based on Dual-CPU architecture," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 2034–2038. DOI: 10.1109/ITNEC.2019.8729401

[24] S. K. Rudrawar and D. Sakhare, "High performance instruction list processor on FPGA platform," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 145–152. DOI: 10.1109/ICCMC.2019.8819846

[25] L. Liangliang and S. Zhengyu, "The design of dual-core PLC SoC and application on launch control system," in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, vol. 1, 2014, pp. 375–379. DOI: 10.1109/InfoSEEE.2014.6948135

[26] A. Milik, M. Kubica, and D. Kania, "Reconfigurable logic controller—direct FPGA synthesis approach," *Applied Sciences*, vol. 11, no. 18, 2021.

[27] M. Niedermaier, D. Merli, and G. Sigl, "A secure Dual-MCU architecture for robust communication of IIoT devices," in *2019 8th Mediterranean Conference*

on Embedded Computing (MECO), 2019, pp. 1–5. DOI: 10.1109/MECO.2019.8760188

[28] C. Su, H. Pang, Y. Liu, and J. Ye, "Design of Multi-CPU automatic sorting system based on machine vision," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2019, pp. 209–212. DOI: 10.1109/ICEIEC.2019.8784567

[29] M. Hubacz and B. Trybus, "Dual-core PLC for cooperating projects with software implementation," *Electronics*, vol. 12, no. 23, 2023.

[30] M. Becker, K. Sandström, M. Behnam, and T. Nolte, "A many-core based execution framework for IEC 61131-3," 11 2015. DOI: 10.1109/IECON.2015.7392805

[31] S. Mubeen, M. Becker, X. Zhao, L. Gan, M. Behnam, and T. Nolte, "Towards automated deployment of IEC 61131-3 applications on multi-core systems," 05 2016. DOI: 10.1109/WFCS.2016.7496531

[32] M. Chmiel, W. Kloska, J. Mocha, and D. Polok, "FPGA-based two-processor CPU for PLC," in *International Conference on Signals and Electronic Systems (ICSES16)*, 2016, pp. 247–252. DOI: 10.1109/ICSES.2016.7593860

[33] M. Chmiel, J. Mocha, and A. Lech, "Implementation of a two-processor CPU for a programmable logic controller designed on FPGA chip," in *International Conference on Signals and Electronic Systems (ICSES18)*, 2018, pp. 13–18.

[34] AMD-Xilinx, *UltraScale Architecture Memory Resources*, 2021, user Guide, UG573 (v1.13).

[35] M. Chmiel, "FPGA-based implementation of bistable function blocks defined in the IEC 61131," *Microprocessors and Microsystems*, vol. 65, pp. 37–46, 2019. DOI: 10.1016/j.micpro.2018.11.006

[36] R. Czerwinski and M. Chmiel, "Hardware-based single-clock-cycle edge detector for a PLC central processing unit," *Electronics (MDPI)*, vol. 8, 2019, art. no 1529. DOI: 10.1016/j.micpro.2018.11.006

[37] M. Chmiel, R. Czerwinski, and A. Malcher, "FPGA implementation of IEC-61131-3-based hardware aided counters for PLC," *Applied Sciences*, vol. 11, no. 21, 2021. DOI: 10.3390/app112110183

[38] M. Chmiel, R. Czerwinski, and A. Malcher, "FPGA implementation of IEC 61131-3-based hardware-aided timers for programmable logic controllers," *Electronics*, vol. 12, no. 20, 2023.

[39] A. Waterman (ed.) and K. Asanovic (ed.), "The RISC-V instruction set manual, volume I: User-level ISA, document version 20191213," RISC-V Foundation, Tech. Rep., December 2019.

[40] A. Waterman (ed.), K. Asanovic (ed.), and J. Hauser (ed.), "The RISC-V instruction set manual, volume II: Privileged architecture, document version 20211203," RISC-V Foundation, Tech. Rep., December 2021.

[41] J. L. Hennessy and D. A. Patterson, *Computer Organization and Design, The Hardware/Software Interface: RISC-V Edition*. Morgan Kaufmann Publishers, 2017.

[42] "IEEE standard for Verilog hardware description language," *IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001)*, pp. 1–590, 2006.

[43] AMD-Xilinx, *7 Series FPGAs Data Sheet: Overview*, 2020, dS180 (v2.6.1), Product Specification.

[44] "IEEE standard for SystemVerilog–unified hardware design, specification, and verification language," *IEEE Std 1800-2023 (Revision of IEEE Std 1800-2017)*, pp. 1–1354, 2024.

[45] Siemens AG, *Simatic S7-300 Instruction List, CPU 31xC, CPU 31x, IM 151-7 CPU, IM 151-8 CPU, IM 154-8 CPU, BM 147-1 CPU, BM 147-2 CPU*, 2008.

[46] Rockwell Automation, "Logix5000 controllers IEC 61131-3 compliance," Rockwell Automation Publication 1756-PM018C-EN-P, Tech. Rep., 2003.

[47] Schneider Automation, *Modicon Quantum automation platform, Hot standby system Unity Pro*, 2013.

[48] VIPA GmbH, *VIPA System 300S SPEED7 – CPU 314-6CF02*, 2014.

[49] General Electric Company, *Intelligent Platforms, Programmable Control Products, PACSystems, RX7i and RX3i CPU Reference Manual, GFK-2222W*, 2015.