Journal of Edge Computing, 2025, Vol. **4**, Iss. 1, pp. 57–72 https://doi.org/10.55056/jec.717

Managing energy consumption in FPGA-based edge computing systems with soft-core CPUs

Oleksandr V. Hryshchuk, Sergiy P. Zagorodnyuk

Taras Shevchenko National University of Kyiv, 64/13 Volodymyrska Str., Kyiv, 01033, Ukraine

Abstract. Edge computing, characterized by processing data closer to its source, has emerged as a promising paradigm to address the challenges of latency, bandwidth, and privacy in the Internet of Things (IoT) era. At the same time, Field-Programmable Gate Arrays (FPGAs) have gained significant attention in edge computing due to their ability to reconfigure design, low power consumption, and high performance. However, the energy consumption of FPGA-based edge computing systems remains a critical concern, particularly in resource-constrained environments where power efficiency is crucial. This paper presents an energy-efficient edge computing system focusing on job scheduling and power management optimization. We review existing techniques and methodologies for optimizing energy consumption in computing systems, including FPGA-based edge devices, identify key challenges and opportunities for future enhancement and propose a flexible, low-power system design with soft-core CPUs.

Keywords: FPGA, power monitoring, soft-core CPU, RTOS, edge computing

1. Introduction

Edge computing systems are usually defined as remote systems based on domains of embedded systems, telecommunication, and cloud systems [19]. In the real world, the edge can be used in the following approaches: “fog” computing, multi-access edge (MEC), and cloudlets (for example, personal network access storage) [19]. Initially, computing edge devices (also called programming logical controllers or PLCs) were based on microcontrollers with hard-core central processing units (CPU), like the RISC-based Microchip AVR ATtiny85 [4], with limited resources and flexibility. But nowadays, field-programmable gate arrays have gained significant attraction in the scope of computing devices for edge systems. FPGA can offer several benefits – customizable hardware acceleration allows to optimize performance for case-specific applications; FPGA can process data with extremely low latency, which can be crucial in autonomous vehicles [1], industrial automation, aerospace engineering [30] and other fields where timely decision-making is essential for safety and efficiency. FPGA chip allows reconfiguring a device without requiring physical hardware changes, enabling edge devices to stay relevant and efficient over time. At the same time, FPGAs are highly power-efficient compared to general-purpose processors like CPUs and graphics processing units (GPU) when performing specific tasks [23].

Increasing the variety and amount of computing systems causes significant energy consumption growth. For example, nowadays, commercial data centres consume 200 TWh yearly, while an entire set of computing devices contributes 2% of the total carbon emissions in the world [16]. According to estimates, in 2030, IT resources will utilize 8% of the whole power supply in the world (or up to 51% in the forecast for the worst-case scenario) [3]. The prospect of heightened power consumption and, subsequently, increased computing costs prompts researchers and engineers to

0009-0007-9926-4231 (O. V. Hryshchuk); 0000-0003-3415-7746 (S. P. Zagorodnyuk)
oleksandr_hryshchuk@knu.ua (O. V. Hryshchuk); szagorodniuk@gmail.com (S. P. Zagorodnyuk)

explore and devise novel methods and strategies for optimizing power management in various computing systems, including edge devices.

There is a set of approaches to increase efficiency in FPGA applications in various domains – from low-end FPGAs for edge computing to high-end FPGAs in ultra-scale high-performance computing (HPC) systems [7, 10, 22, 23, 32]. However, the central part of these works is case-specific. Applications are often based on toolkits like OpenCL [29] or oneAPI [2], which provide general and efficient solutions for high-end (like Intel Stratix 10) devices in HPC systems, but maybe not supported for low-end devices, used in edge computing (in example Max 10 FPGA), and vice versa there are works on low-level implementation in hardware-description languages like Verilog or VHDL (an example implementation of k-means algorithm on FPGA [6]).

Therefore, this paper aims to develop and evaluate the architecture of general-purpose low-end FPGA-based edge computing systems with extended power consumption management capabilities.

The paper is structured as follows: section 2 reviews state-of-art power management solutions in computing systems with FPGA accelerators and defines the problem of energy-efficient scheduling to provide generalized power consumption optimization. Section 3 describes the architecture of the proposed system and the experimental setup for its evaluations. Section 4 discusses obtained results, including system limitations and future work. Section 5 concludes this paper.

2. Theoretical background

2.1. Power management in FPGA-based computing systems

In FPGA chips, separate nodes distribute power consumption, which is divided into the following FPGA sub-element categories: logic cells, interconnect network, clock buffer, and embedded memory [9].

Internal elements such as logical cells and programmable wires consume over 80% of the energy supplied to the chip. These resources can be used in different ways, such as optimizing the reconfiguration process and minimizing route paths, to manage and save power.

Moreover, case-specific or generalized FPGA-based edge computing systems can be applied to typical power management methods in computing systems. Classification of the mentioned techniques was proposed in [11] and depicted in figure 1. Two main classes define all methods – static power management (SPM) and dynamic power management (DPM). From the FPGA point of view, designers apply static power management methods during the device reconfiguration stage. For example, the Intel Quartus Prime software suite provides an Early Power Estimator and Power Analyzer (available only in the post-fit stage) [15], which helps to achieve various thermal or power optimization goals.

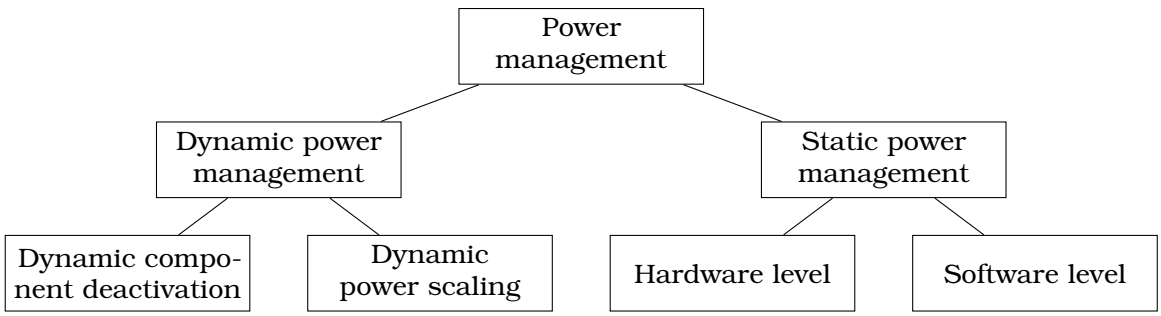


Figure 1: General classification of power management methods in computer systems.

Dynamic methods are divided into two main groups — dynamic component deactivation (DCD), which are based on predictive or heuristic approaches, and dynamic power scaling (DPS), for example, resource throttling and dynamic voltage and frequency scaling (DVFS) [13, 14]. The mentioned method serves as a base for more sophisticated optimization techniques, such as task scheduling based on DVFS [26] or the DCD heuristics solution, which is utilized in the HPC domain [17].

The approaches presented in this section can be applied to different hardware platforms. For example, it can be homogeneous or heterogeneous (with GPU, TPU, FPGA) HPC systems or IoT soft-core and hard-core hardware solutions, including edge computing devices. Soft-core CPUs allow developers to use them as the foundation for software solutions without overhead in hardware implementation. At the same time, Micro-C RTOS provides a simple and easy-to-use task scheduler with opportunities to optimize it. In this paper, we propose to use an FPGA-based edge computing system with soft-core CPU NIOS II and Micro-C RTOS.

2.2. Task scheduling in RTOS

Edge computing devices are often part of real-time systems, so we chose Micro-C RTOS as an operating system for the proposed FPGA-based edge computing system. Successful task completion is crucial and mandatory in real-time systems because of safety-critical requirements. To fulfil this necessity, RTOS utilize simple but reliable scheduling policies, like rate-monotonic scheduling (RMS), which is used in Micro-C RTOS [18].

Rate-monotonic scheduling has the following requirements:

1. Tasks do not use any synchronization or shared resources (like hardware resources of a semaphore blocker)
2. Tasks are periodic
3. Only static prioritization is used and assigned in compliance with rate-monotonic conventions

Initially, all deadlines in rate-monotonic scheduling must satisfy the following criteria:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1) \tag{1}$$

where n is the number of tasks (process) to be scheduled, C_i and T_i are computation time and release period for task i, and U corresponds to total CPU utilization by scheduled tasks [18, 20]. This condition provides CPU utilization from 83% for two tasks and to ≈69% when the task number goes to infinity ($U \leq \ln 2$ for $n \to \infty$). Scheduling criteria and policy can be extended and optimized for multi-core scenarios [21] or include energy consumption criteria, described in the following section.

By default Micro-C/OS-II provides the following options for scheduling [18]:

1. Up to 64 tasks (4 tasks with the highest priority and 4 with the lowest are reserved), 56 of them available for application
2. The lower value of the priority field means the highest priority of the task
3. Task priority number used as task identifier

2.3. Problem definition for energy-efficient scheduling

Power consumption optimization techniques can be classified as hardware and software-based. Hardware-based methods are case-specific for different hardware variations, such as the CPU. Software-based techniques can be generalized and provide a solution for various hardware devices with common characteristics, such as

homogeneous or heterogeneous HPC systems with GPU and TPU accelerators [24]. In the scope of this paper, we focused on software solutions that lead to energy-efficient task-scheduling methods.

Optimization problem energy consumption via task scheduling can be defined as finding a set of start times $\{s_1, s_2, \ldots, s_{|J|}\}$ for jobs from a finite set of tasks J, allocated to resources $\{a_1, a_2, \ldots, a_{|J|}\}$ using a finite set of resources R in conditions where:

$$\forall s_x : \nexists s_y : s_x \leq s_y + \text{time}\,(y, A_y) \wedge s_y \leq s_x + \text{time}\,(x, A_x) \wedge a_x = a_y, \forall a_x : x \in R \tag{2}$$

where $time(j, r)$ is the function for calculating job execution duration.

Optimization criteria for the mentioned problem will be finding the maximum or minimum (depending on a function that uses simple metrics such as execution time, consumed energy, resource usage, etc.) [17].

$$\min / \max \left(\text{ OptimizationCriteria } \left(\left\{ s_1, s_2, \ldots, s_{|J|} \right\}, \left\{ a_1, a_2, \ldots, a_{|J|} \right\} \right) \right) \tag{3}$$

The described problem definition has several disadvantages – an assumption that one resource can take only one task at a time and the fact that the count of available resources is always equal to or higher than the number of jobs to complete and does not include the impact of communication between tasks on nodes or computing elements. Micro-C/OS-II RTOS provides inter-process communication (IPC) functionality. Therefore, it is essential to mitigate these issues in embedded computing systems. Resolution of these issues and adapted and upgraded model was suggested [17] – for two tasks x and y from a set of jobs pairs D, P_j is a set of devices, which can be assigned for job $j \in J$, time of communication between jobs obtained from function $comm(x, y, a_x, a_y)$, then the solution is a set of assignments A_j and start times $\{s_1, s_2, \ldots, s_{|J|}\}$ for each job as it described in equations 4-7:

$$\forall x \in A_j : x \in P_j \tag{4}$$

$$\forall s_x : \nexists s_y : s_x \leq s_y + \text{time}\,(y, A_y) \wedge s_y \leq s_x + \text{time}\,(x, A_x) \wedge A_x \cap A_y = \emptyset \tag{5}$$

$$\forall \{x, y\} \in D : s_x + \text{time}\,(x, A_x) + \text{comm}\,(x, y, A_x, A_y) \leq s_y \tag{6}$$

With optimization condition:

$$\min / \max \left(\text{ OptimizationCriteria } \left(\left\{ s_1, s_2, \ldots, s_{|J|} \right\}, A_1, \ldots, A_{|J|}, D \right) \right) \tag{7}$$

Enumeration over the entire set of jobs for the entire set of available resources means that solution time can not be found in reasonable polynomial time, which was for energy-efficient active time as NP-complete scheduling [5]. To find the solution for this problem in a reasonable time before starting (in the case of static scheduling) or during runtime, precalculated configurations or heuristic methods, for example, genetic algorithms [8], can be used.

Another essential point in this definition is the selection of optimization criteria for specific edge-computing system cases. Existing solutions use energy consumption metric (EC) or can take under consideration other properties, such as execution time [17]. Energy consumption can be described via energy itself (in joules/watts) or depicted via more complicated parameters like instruction per joule or power per Watt [25]. For example, this approach is used in the Green500 rating as the FLOPS per Watt metric [27]. Improved or adapted parameters can be used as a combination of metrics such as energy consumption, execution time, utilization, average weighted time (AWT), wait time, power, Pareto front, AST, AFT, clock frequency, task(job) per energy, reliability, electricity cost, temperature, EDP, EDF, number of cores, probability of execution, branch transition rate, cache efficiency, issue width [17]. Due to implementation transparency, we used energy (consumed by the testbed board) as the key metric in our proof of concept system.

3. Edge computing system architecture

The suggested edge computing system consists of several key components – user and task scheduler services (physically located on a single management server), edge agent, power monitor, and FPGA per each testbed setup. User service acts as an interface between end-users and the computing system and is responsible for handling user requests and managing job submissions. The scheduler service is responsible for resource allocation and workload distribution optimization (using power monitor agent output). Edge Agent facilitates communication between the scheduler and target edge hardware and performs FGPA reconfiguration when required. The FPGA board is designated for hardware acceleration for computationally intensive operations and is used as this paper's primary target for power profiling. The block diagram of the suggested architecture is shown in figure 2.

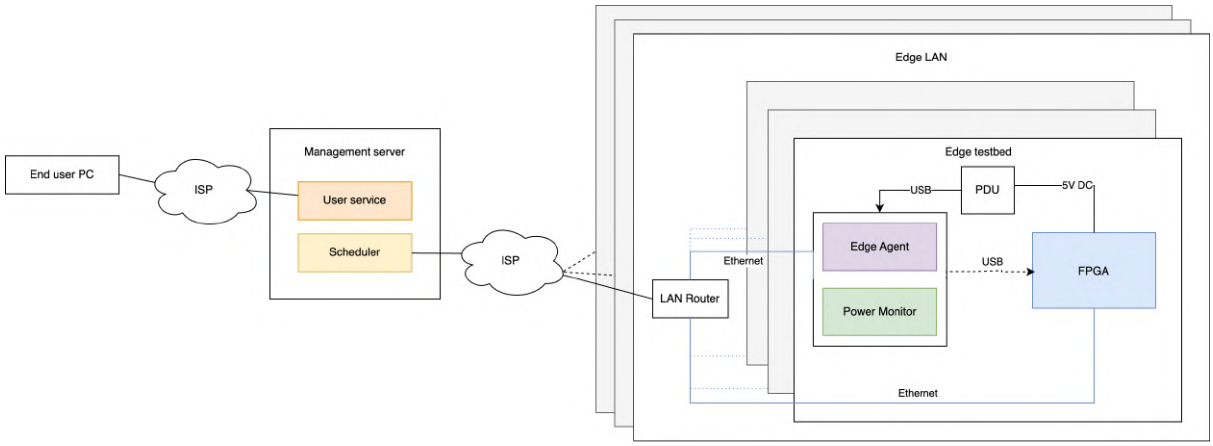


Figure 2: Edge computing system architecture.

After the "power on" event, the user service requests the scheduler to begin the initialization process and bring up the required services. The suggested design can support soft-core IP blocks (like NIOS II CPU); in this case, the edge agent will provision the soft-core SoC (system-on-chip) design in the alternative stage of initialization flow (detailed process depicted in figure 3).

When the system is ready, the end user can submit the job for execution to the task scheduler service. The scheduler acknowledges job requests and performs FPGA reconfiguration when required. When the FPGA chip is ready, the scheduler will send a job initialization request to the FPGA and request that the power monitor start the power consumption measurement. Following the acknowledgement response, the scheduler will begin data exchange with the target with FPGA (due to the limitation of onboard memory, frequent data exchange is mandatory for extensive jobs). It will read measurements from the power monitor. The scheduler will forward the results to the end user when the job is done. The complete task execution flow is shown in figure 4. The described process requires edge agent involvement only for FPGA reconfiguration, not for computing load.

3.1. Experimental setup

Hardware implementation is based on the Intel Max 10 FPGA chip on the Terasic DE10 Lite board. To create server-platform architecture (where the host PC is used for FPGA programming and sending tasks to the FPGA computing node), we used a Wiznet W5100 chip on an Ethernet shield from Keyestudio. An edge computing system was implemented using Intel Quartus Platform Designer and Nios II soft-core CPU. A created computer consists of onboard clock source for 50 MHz clock (clk_50), ALTPLL

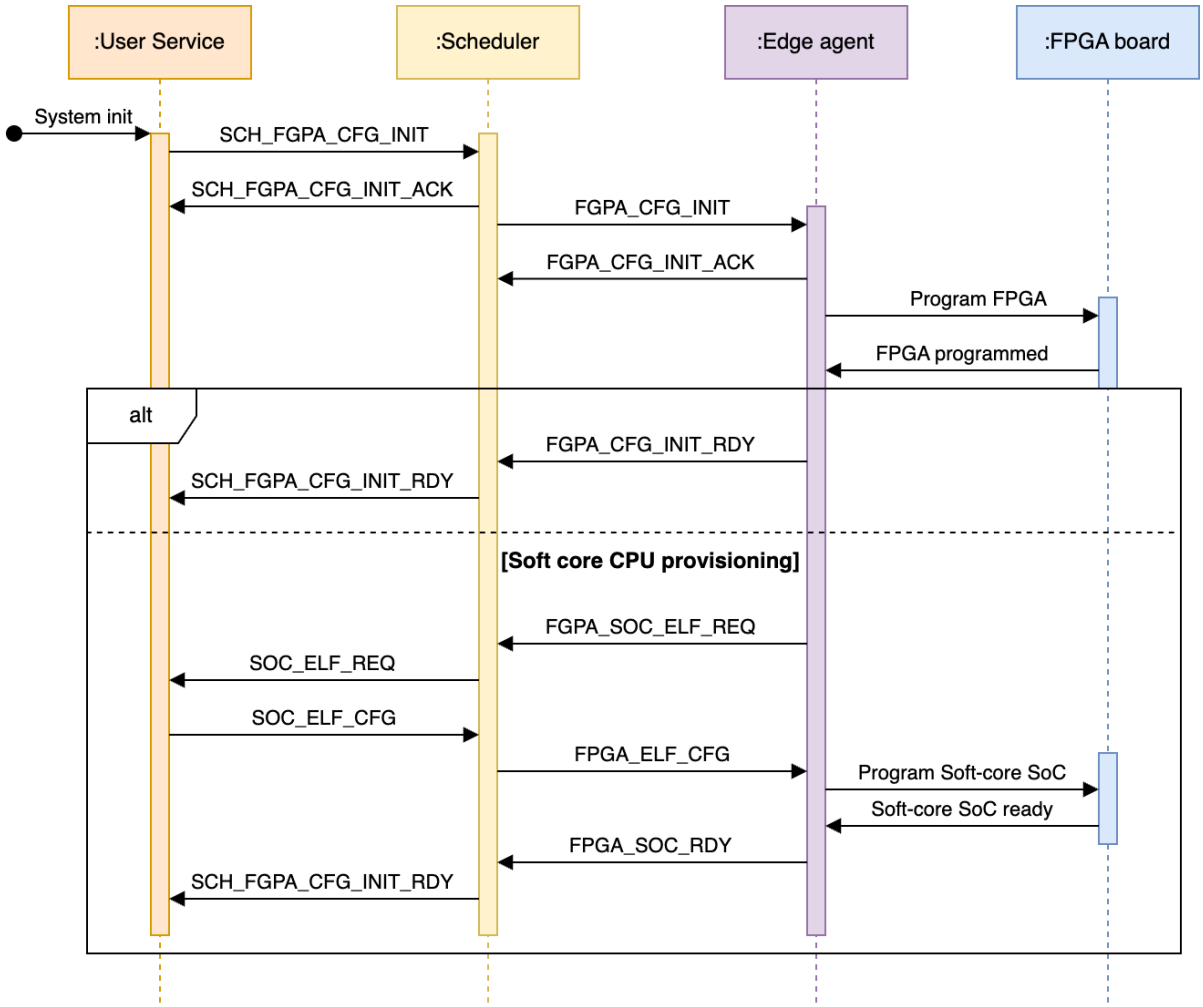


Figure 3: FPGA computing agent initialization flow.

Intel FPGA IP-block (altpll) used to multiply input clock to be able to work on higher frequency, system ID, interval timer IP block for interrupt sender, SDRAM controller IP block for RAM, PIO for push button on board (key), RISC-based Nios II processor and existing IP block for W5100 controller [12], created architecture depicted on figure 5 and figure 6.

Both the host and FPGA were connected to the same local network, and static IP addresses were assigned for both devices. TCP sockets were used to enable communication between the host and board and send computing tasks. The power-to-edge device was supplied via a USB interface, which was also used for programming and debugging via the JTAG interface (initialization output from Nios console shown in figure 7 using an onboard USB blaster).

A FNB 58 USB tester was used for energy consumption monitoring, capture from the tester shown in figure 8, the tested device, and the tester itself shown in figure 9. An overview of the implemented testbed system architecture (including key components and connection interfaces) is shown in figure 10.

The suggested architecture was configured using Platform Designer from Intel Quartus Prime Lite and shown in figure 6. All parts of this computing system were connected via Avalon Bus for system interconnect and were connected to `nios2_qsys.data_master` and `nios2_qsys.instruction_master`. The address map for the current configuration is displayed in table 1.

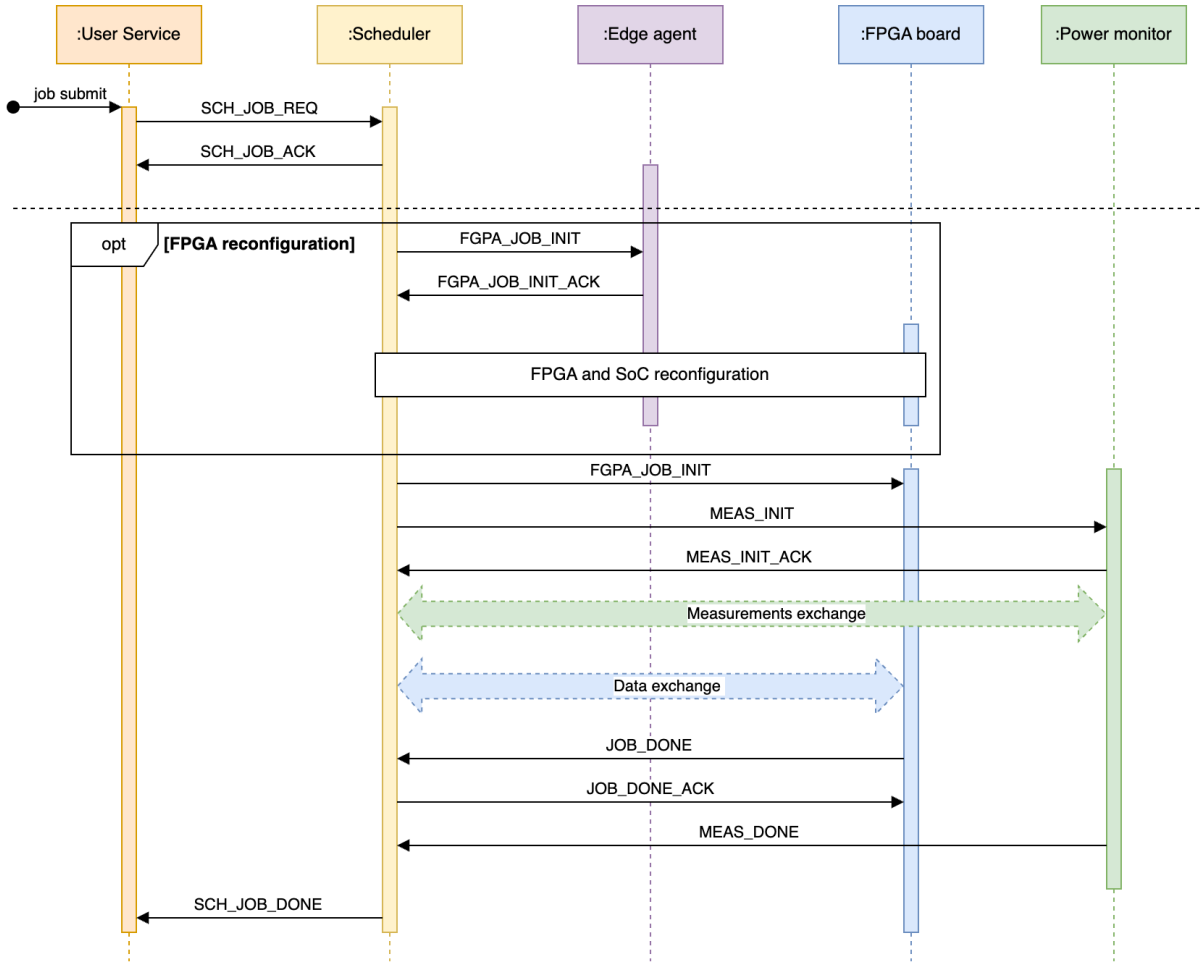


Figure 4: Job execution flow.

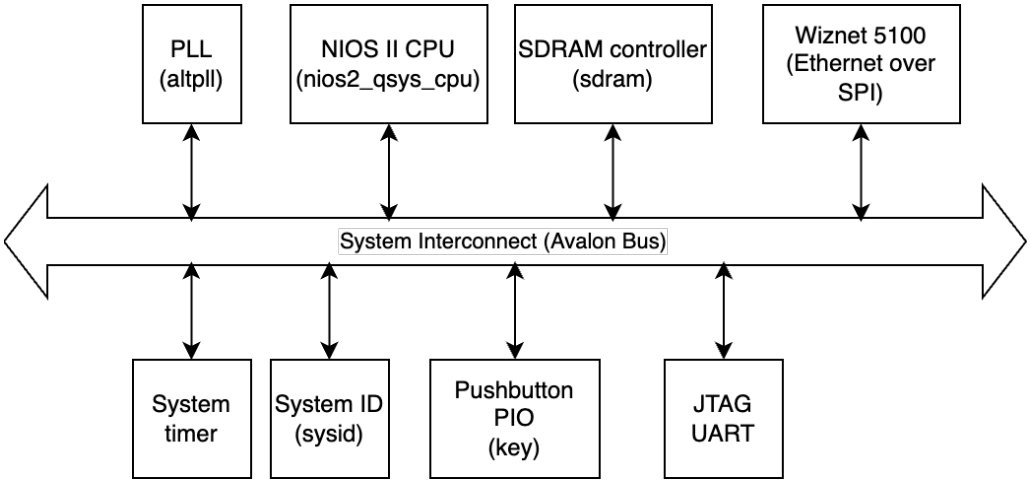


Figure 5: Suggested architecture of edge computing system.

Register transfer level view of the top entity with a connection to board input and output pins (including GPIO) shown in figure 11. A default factory reset configuration for DE10_Lite was used to simplify HDL development. For this reason, some unused pins exist, like `ADC_CLK_10`, `GSENSOR_INT`, `MAX10_CLK_10`, etc. These pins may be used to connect peripheral devices and sensors. As a result, the current setup utilized

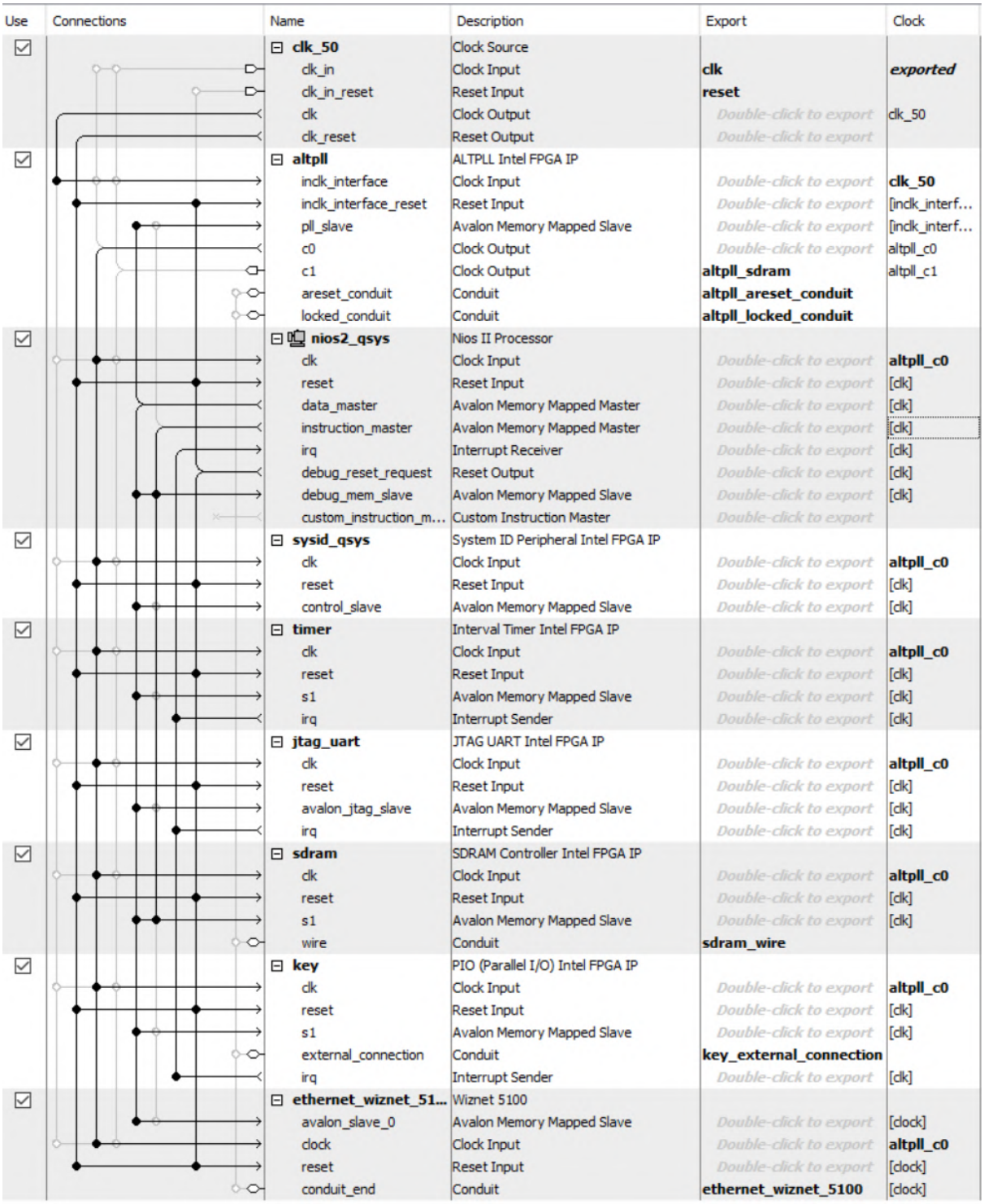


Figure 6: Platform Designer configuration for suggested architecture.

only 10% of the available logical elements inside the FPGA chip (detailed report from compilation in table 2). The rest of the logical resources can be used to create a multi-core setup or evaluate other soft-core CPUs.

3.2. Advantages and disadvantages of the suggested architecture

The proposed setup provides several benefits for users and researchers. As advantages, the following points can be emphasized:

```
Wiznet5100 Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
Hello from UCOSII
-- Wiznet-Ethernet SPI Succeeded
STATIC IP:192.168.0.17
read RMSR 55
read TMSR 55
Wiznet W5100 initialized!
```

Figure 7: Device initialization output.

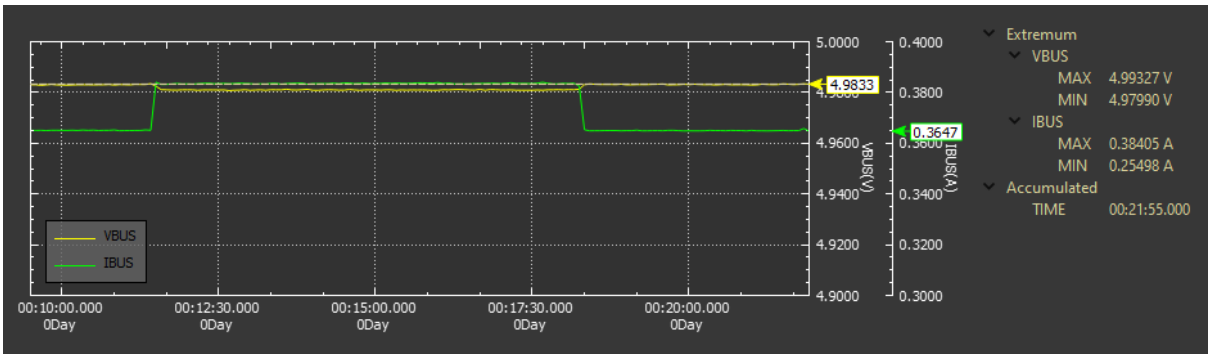
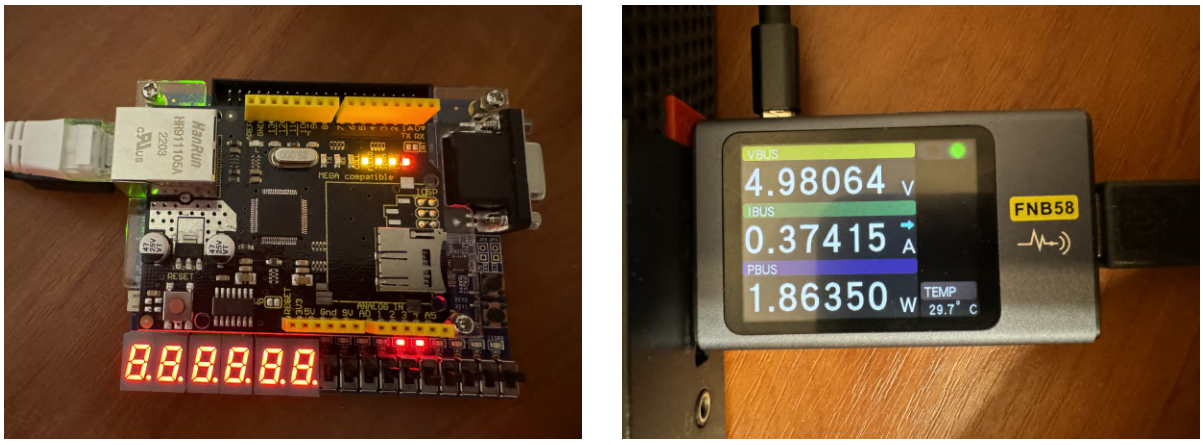


Figure 8: Supplied current and voltage levels, measured on USB for 21 minutes.



(a) (b)

Figure 9: Photo of connected edge device (a), Fnirsi FNB 58 USB tester.

- The usage of FPGA provides incredible flexibility for various tasks in edge computing.
- Using a soft-core CPU alongside RTOS allows using a single HDL configuration for numerous applications with an easy-to-use Micro-C/OS-II advanced binary interface (ABI).
- Easy-to-use energy consumption monitoring via an external USB tester.
- RTOS provides a simple scheduler that can be optimized or replaced with advanced techniques, focusing on energy saving.

At the same time, the developed system has several disadvantages that can be mitigated:

- Wiznet Ethernet to SPI controller limits data transmission speed to 10 Mb/s, which may be fine for reading and handling information from low-speed peripheral

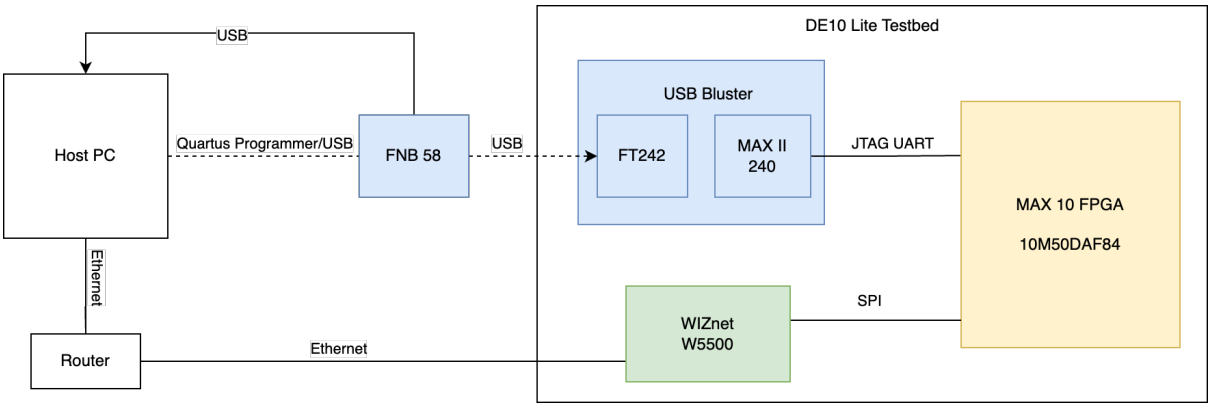


Figure 10: General architecture of implemented testbed system.

Table 1
Address map in implemented setup.

	nios2_qsys.data_master
sdram.s1	0x0200_0000 – 0x0300_0fff
nios2_qsys.debug_mem_slave	0x0400_0800 – 0x0400_0fff
timer.s1	0x0400_1000 – 0x0400_101f
ethernet_wiznet_5100.avalon_slave_0	0x0400_1020 – 0x0400_103f
key.s1	0x0400_1040 – 0x0400_104f
altpll.pll_slave	0x0400_1050 – 0x0400_105f
sysid_qsys.control_slave	0x0400_1060 – 0x0400_1067
jtag_uart.avalon_jtag_slave	0x0400_1068 – 0x0400_106f

	nios2_qsys.instruction_master
sdram.s1	0x0200_0000 – 0x03ff_0fff
nios2_qsys.debug_mem_slave	0x0400_0800 – 0x0400_0fff

Table 2
Parameters of compiled design.

Parameter name	**Parameter value**
Total logical elements	5036/49760 (10%)
Total PLL	1/4
Total pins	180/360 (51%)
Minimum core junction temperature	0 °C
Maximum core junction temperature	85 °C
Device I/O Standard	2.5 V
F_{max} (1200 mV 85 °C Slow model)	130.17 MHz
F_{max} (1200 mV 0 °C Slow model)	141,2 MHz

devices, such as temperature sensors, keyboards, GPIO, etc. Still, it will not suit a high-load fast application. Complex Ethernet components with proprietary Qsys IP blocks can be onboarded, providing a communication speed of up to 1 Gb/s for Max 10 devices to mitigate the low-speed issue.

- The power supply via USB cable is acceptable for the current prototype but can be replaced with an external 5V power supply connected to the 5V/GND header.
- The current implementation uses Wiznet 5100 on the Arduino header, which leads to additional energy dissipation on LEDs for status indicators on board. In

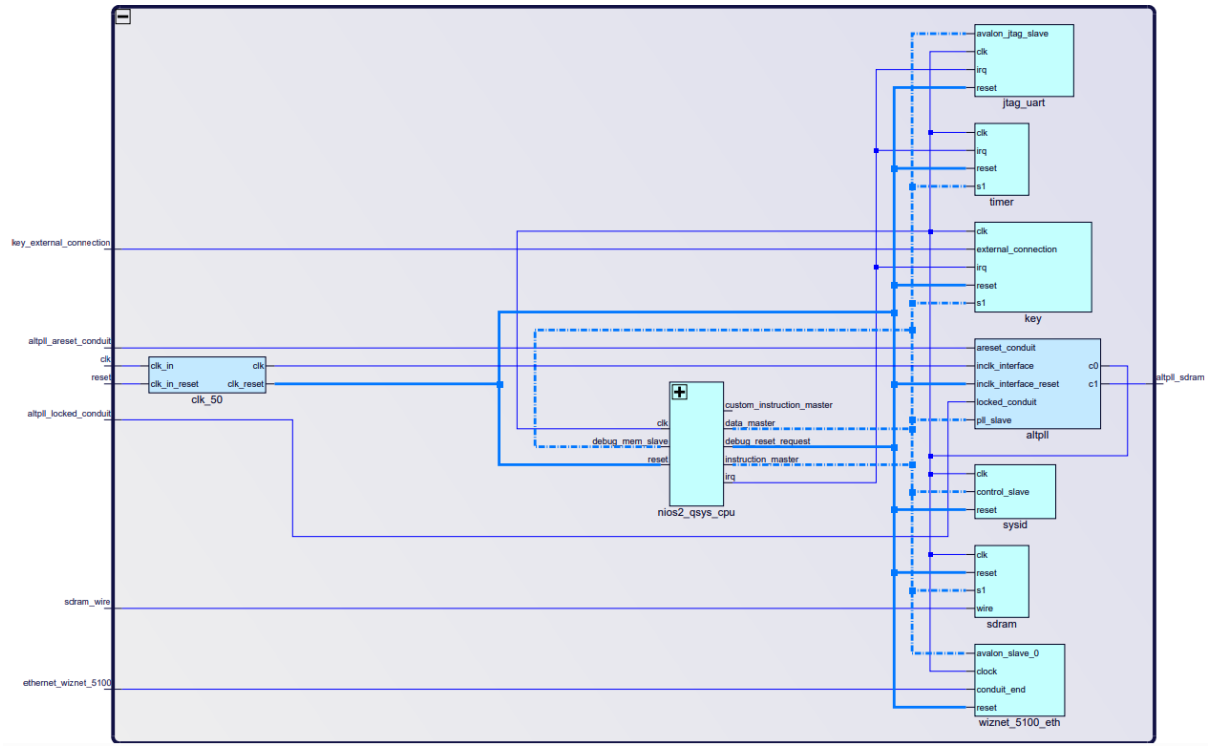


Figure 11: Schematic for implemented edge computing device.

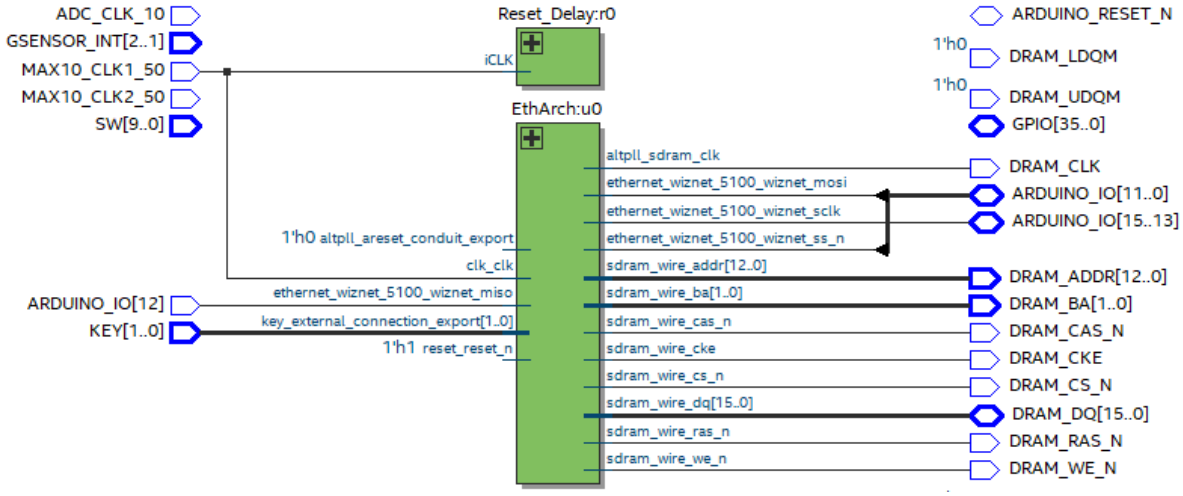


Figure 12: RTL view of configured FPGA.

the future, it can be replaced with a simplified Ethernet module.

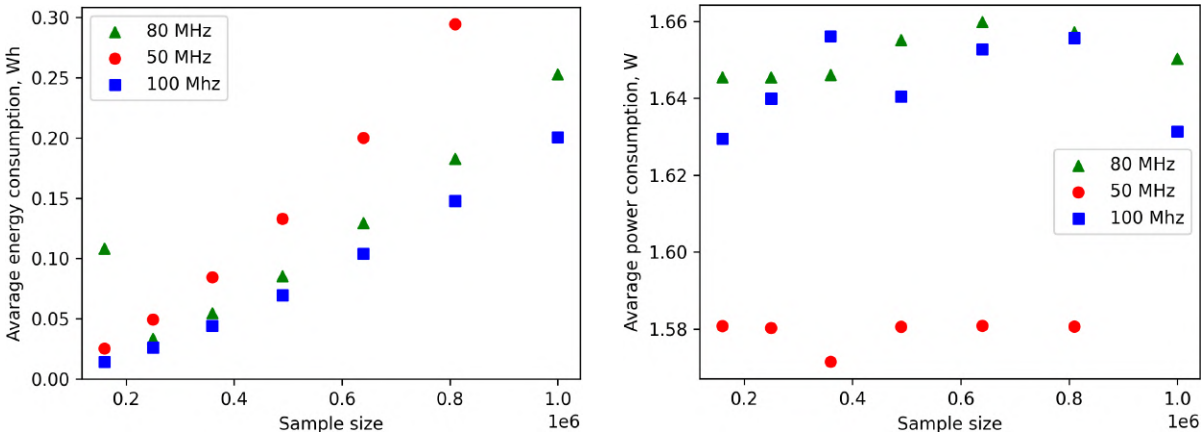
We suggested use of simple matrix multiplication as load task for edge device, which can be used as a base for more complex functions like neural networks, image processing, etc. We used the C++ program for this task, which was compiled for Micro-C/OS-II RTOS and executed on the Nios II processor.

4. Results and discussion

Several experiments were conducted to evaluate energy consumption in the proposed system. The task was executed 20 times with different matrix sizes and clock frequencies. The size of measured matrices varies from $1.6 \cdot 10^5$ (5.12 MB) to 10^6

(30 MB) elements. The frequency was set to 50, 80, and 100 MHz. As expected, energy consumption grows with the increase of matrix size and frequency decrease. For power consumption, we observed the lowest consumption for 50 MHz and the highest for 80 and 100 MHz. However, the difference between 80 and 100 MHz is not significant (in the range of 0.03 W). Measurements for higher frequencies were not conducted due to the limitation of PLL configuration and F_{max} of 141.2 MHz for the current design. The received data does not include energy consumption on the host PC and networking equipment. Collected data is shown in the table 3. An example of measured data of energy and power consumption of 700x700 matrix multiplication on a 100 MHz clock is shown in figure 14.

Obtained results can be compared with related works, such as power and energy consumption modelling on NIOS II in [28] on Arria II GX and Cyclone III LS. The average power measured in our experiments is 90% higher than shown in [28], which is expected due to two different measurement methodologies and load tasks. Measurements described in this paper include not only FPGA chip power consumption but the entire board (including the connected W5100 chip, which can dissipate up to 0.3 watts). Another example is a case of FPGA-based edge computing, described in [31], with a computer vision application confirming significant power consumption optimization with a 15% power save compared to regular CPU load.



(a) (b)

Figure 13: Energy (a) and power (b) consumption for different frequencies.

4.1. Limitations

From our obtained results, comparison with related works, and methodology definition, we identified the following limitations:

- Low-speed (10 Mb/s) Ethernet connection with high power dissipation on SPI to Ethernet chip is a bottleneck for high-load applications.
- The suggested design supports CPU frequency only up to 100 MHz
- Lack of isolated measurements – only whole board power consumption is monitored.
- Single-core CPU design utilizes only 10% of available logical elements (table 2).
- Default preemptive scheduling has limited flexibility.

4.2. Future work

Limitations, described in the previous subsection, define possible areas of improvement and future work. It will include the replacement of the SPI to Ethernet controller

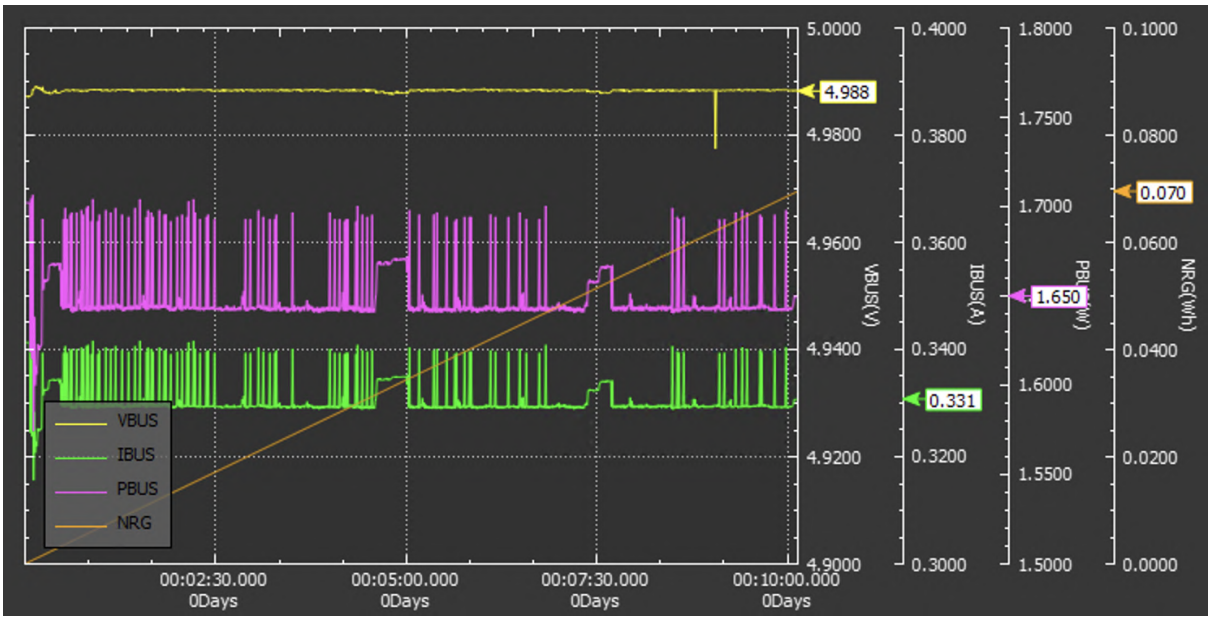


Figure 14: Current, voltage, and power consumption of 700x700 integer matrix multiplication on 100 MHz clock.

Table 3
Energy and power consumption for different frequencies and sample sizes (the size of each element in the sample is 32 bits).

Frequency, MHz	Sample size	Time, s	Power min, W	Power max, W	Power avg, W	Energy, Wh
50.00	160000	225.60	1.56343	1.59797	1.5807	0.02522
	250000	433.70	1.56255	1.59785	1.5802	0.04927
	360000	752.00	1.542	1.60083	1.571415	0.08435
	490000	1184.00	1.56225	1.59878	1.580515	0.13283
	640000	1804.00	1.56213	1.59939	1.58076	0.2
	810000	2548.00	1.56216	1.59897	1.580565	0.29432
80.00	160000	141.10	1.62501	1.66578	1.645395	0.10808
	250000	271.40	1.62362	1.66714	1.64538	0.0333
	360000	470.30	1.62406	1.66797	1.646015	0.05457
	490000	740.80	1.64005	1.67011	1.65508	0.08521
	640000	1128.90	1.64414	1.67543	1.659785	0.12931
	810000	1594.70	1.64129	1.67297	1.65713	0.18262
	1000000	2173.70	1.62696	1.67351	1.650235	0.25262
100.00	160000	112.60	1.56074	1.6981	1.62942	0.01394
	250000	216.40	1.57307	1.70668	1.639875	0.02592
	360000	375.20	1.6054	1.70673	1.656065	0.04387
	490000	590.60	1.57427	1.70655	1.64041	0.06954
	640000	900.30	1.60133	1.70407	1.6527	0.10394
	810000	1271.20	1.607	1.70423	1.655615	0.14769
	1000000	1733.40	1.55591	1.70671	1.63131	0.20045

with the high-speed interface and FPGA resource utilization expansion (for example, with a multi-core CPU design). Integrating advanced scheduling algorithms. The important point is expanding the test base with tasks like the Dhrystone benchmark used in [28]. Also, the edge network will be expanded with various computing devices of different CPU architectures.

5. Conclusions

This paper describes modern energy consumption management and optimization methods in FPGA-based edge computing systems, focusing on job scheduling.

The architecture of general-purpose low-end FPGA-based edge computing systems with extended power consumption management capabilities was proposed and evaluated. The developed system, described in this paper, provides flexibility for configuration by utilizing soft-core CPUs and custom case-specific IP blocks. Programs designed for the mentioned processors can be easily ported to another device or device family or vendor. At the same time, the developed configuration shows low energy consumption (for example, 1.86 Watt, shown in figure 9 (b)), which satisfies requirements for edge devices. Therefore, this system can be used as a foundation for more complex but energy-efficient edge computing solutions in various domains, including hard and soft real-time systems, artificial intelligence applications, and others.

Matrix multiplication conducted for 50, 80, and 100 MHz clock frequencies for sample sizes from 1 MB to 30 MB is suitable as a benchmark for suggested architecture evaluations; however, additional extended test methods, such as the Dhrystone benchmark, are required for better comparison with existing solutions. Energy consumption measurements showed that the proposed system could be used as a testbed for research on energy-efficient edge systems.

Declaration on generative AI: The authors have not employed any generative AI tools.

References

[1] Aishwarya, G., Patil, B., Joshi, P.V., Sudarsham, K.M., Vaidyanathan, K., Parandkar, P. and Dsouza, A., 2022. A Survey on use of FPGA in Automotive System. *2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*. pp.51–56. Available from: https://doi.org/10.1109/DISCOVER55800.2022.9974941.

[2] Alcaraz, S.R., Laso, R., Lorenzo, O.G., Vilariño, D.L., Pena, T.F. and Rivera, F.F., 2024. Assessing Intel OneAPI capabilities and cloud-performance for heterogeneous computing. *The Journal of Supercomputing*, 80(9), p.13295–13316. Available from: https://doi.org/10.1007/s11227-024-05958-5.

[3] Andrae, A.S.G. and Edler, T., 2015. On global electricity usage of communication technology: Trends to 2030. *Challenges*, 6(1), pp.117–157. Available from: https://doi.org/10.3390/challe6010117.

[4] *ATtiny85*, 2024. Available from: https://www.microchip.com/en-us/product/attiny85.

[5] Chang, J., Gabow, H.N. and Khuller, S., 2012. A Model for Minimizing Active Processor Time. In: L. Epstein and P. Ferragina, eds. *Algorithms – ESA 2012*. Springer, *Lecture Notes in Computer Science*, vol. 7501, pp.289–300. Available from: https://doi.org/10.1007/978-3-642-33090-2_26.

[6] Dias, L.A., Ferreira, J.C. and Fernandes, M.A.C., 2020. Parallel Implementation of K-Means Algorithm on FPGA. *IEEE Access*, 8, pp.41071–41084. Available from: https://doi.org/10.1109/ACCESS.2020.2976900.

[7] Favaro, F., Dufrechou, E., Ezzatti, P. and Oliver, J.P., 2021. Energy-Efficient Algebra Kernels in FPGA for High Performance Computing. *Journal of Computer Science and Technology*, 21(2), pp.80–92. Available from: https://doi.org/10.24215/16666038.21.e09.

[8] Fernández, A. Cocaña, Ranilla, J. and Sánchez, L., 2015. Energy-efficient allocation of computing node slots in HPC clusters through parameter learning and hybrid genetic fuzzy system modeling. *Journal of Supercomputing*, 71(3), pp.1163–1174. Available from: https://doi.org/10.1007/s11227-014-1320-9.

[9] Garcia, A.D.G., Perez, L.F.G. and Acuna, R.F., 2005. Power consumption management on FPGA. *15th International Conference on Electronics, Communications and Computers (CONIELECOMP'05)*. pp.240–245. Available from: https://doi.org/10.1109/CONIEL.2005.60.

[10] Goel, A., Goel, A.K. and Kumar, A., 2023. Performance analysis of multiple input single layer neural network hardware chip. *Multimedia Tools and Applications*, 82(18), pp.28213–28234. Available from: https://doi.org/10.1007/s11042-023-14627-3.

[11] Haj-Yahya, J., Mendelson, A., Asher, Y.B. and Chattopadhyay, A., 2018. *Energy Efficient High Performance Processors: Recent Approaches for Designing Green High Performance Computing*. Springer. Available from: https://doi.org/10.1007/978-981-10-8554-3.

[12] Holguer, A., 2024. *W5100 Qsys Component*. Available from: https://www.fpgalover.com/index.php/component/content/article/30-cores/39-wiznet-5100-core?Itemid=101.

[13] Ibro, M. and Marinova, G., 2020. DVFS Technique on a Zynq SoC-based System for Low Power Consumption. *2020 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*. pp.1–5. Available from: https://doi.org/10.1109/CoBCom49975.2020.9174061.

[14] Ibro, M. and Marinova, G., 2021. Review on Low-Power Consumption Techniques for FPGA-based designs in IoT technology. *2021 16th International Conference on Telecommunications (ConTEL)*. pp.110–114. Available from: https://doi.org/10.23919/ConTEL52528.2021.9495970.

[15] *Intel® Quartus® Prime Standard Edition User Guide*, 2018. Intel. Available from: https://www.intel.com/content/www/us/en/docs/programmable/683230/18-1/user-guides.html.

[16] Jones, N., 2018. How to stop data centres from gobbling up the world's electricity. *Nature*, 561, pp.163–166. Available from: https://doi.org/10.1038/d41586-018-06610-y.

[17] Kocot, B., Czarnul, P. and Proficz, J., 2023. Energy-Aware Scheduling for High-Performance Computing Systems: A Survey. *Energies*, 16(2), p.890. Available from: https://doi.org/10.3390/en16020890.

[18] Labrosse, J., 2002. *MicroC/OS-II: The Real Time Kernel*. CRC Press.

[19] Lea, P., 2020. *IoT and Edge Computing for Architects*. 2nd ed. Packt Publishing Ltd. Available from: https://books.google.com.ua/books?id=bZJ8zQEACAAJ.

[20] Liu, C.L. and Layland, J.W., 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1), p.46–61. Available from: https://doi.org/10.1145/321738.321743.

[21] Min-Allah, N., Hussain, H., Khan, S.U. and Zomaya, A.Y., 2012. Power efficient rate monotonic scheduling for multi-core systems. *Journal of Parallel and Distributed Computing*, 72(1), pp.48–57. Available from: https://doi.org/10.1016/j.jpdc.2011.07.005.

[22] Minhas, U.I., Woods, R., Nikolopoulos, D.S. and Karakonstantis, G., 2022. Efficient, Dynamic Multi-Task Execution on FPGA-Based Computing Systems. *IEEE Transactions on Parallel and Distributed Systems*, 33(3), pp.710–722. Available from: https://doi.org/10.1109/TPDS.2021.3101153.

[23] Qasaimeh, M., Denolf, K., Lo, J., Vissers, K., Zambreno, J. and Jones, P.H., 2019. Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels. *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. pp.1–8. Available from: https://doi.org/10.1109/ICESS.2019.8782524.

[24] Raca, V., Umboh, S.W., Mehofer, E. and Scholz, B., 2022. Runtime and energy constrained work scheduling for heterogeneous systems. *Journal of Super-*

computing, 78(15), pp.17150–17177. Available from: https://doi.org/10.1007/s11227-022-04556-7.

[25] Safari, M. and Khorsand, R., 2018. Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. *Simulation Modelling Practice and Theory*, 87, pp.311–326. Available from: https://doi.org/10.1016/j.simpat.2018.07.006.

[26] Saha, S. and Purohit, M., 2021. NP-completeness of the Active Time Scheduling Problem. 2112.03255[cs], Available from: http://arxiv.org/abs/2112.03255.

[27] Scogland, T., Azose, J., Rohr, D., Rivoire, S., Bates, N. and Hackenberg, D., 2015. Node variability in large-scale power measurements: perspectives from the Green500, Top500 and EEHPCWG. *SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. pp.1–11. Available from: https://doi.org/10.1145/2807591.2807653.

[28] Senn, L., Senn, E. and Samoyeau, C., 2012. Modelling the Power and Energy Consumption of NIOS II Softcores on FPGA. *2012 IEEE International Conference on Cluster Computing Workshops*. pp.179–183. Available from: https://doi.org/10.1109/ClusterW.2012.41.

[29] The Khronos Group, 2013. *OpenCL - The Open Standard for Parallel Programming of Heterogeneous Systems*. Available from: https://www.khronos.org/opencl/.

[30] Vipin, K. and Fahmy, S.A., 2018. FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures, Methods, and Applications. *ACM Computing Surveys*, 51(4), p.72. Available from: https://doi.org/10.1145/3193827.

[31] Xu, C., Jiang, S., Luo, G., Sun, G., An, N., Huang, G. and Liu, X., 2022. The Case for FPGA-Based Edge Computing. *IEEE Transactions on Mobile Computing*, 21(7), pp.2610–2619. Available from: https://doi.org/10.1109/TMC.2020.3041781.

[32] Young, A.R., Miniskar, N.R., Liu, F., Blokland, W. and Vetter, J.S., 2022. Adrastea: An Efficient FPGA Design Environment for Heterogeneous Scientific Computing and Machine Learning. In: K. Doug, G. Al, S. Pophale, H. Liu and S. Parete-Koon, eds. *Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation*. Cham: Springer Nature Switzerland, *Communications in Computer and Information Science*, vol. 1690, pp.227–243. Available from: https://doi.org/10.1007/978-3-031-23606-8_14.