Method and Framework for IoT Nodes Self-Configuring with Cloud-Based Automatic Design and Synthesis Tools

Viktor Melnyk[1,2,*,†] and Anatoliy Melnyk[3,†]

[1] *Lviv Polytechnic National University, Bandery st. 12, 79013, Lviv, Ukraine*
[2] *John Paul II Catholic University of Lublin, Al. Racławickie 14, 20–950, Lublin, Poland*
[3] *IT Step University, Zamarstynivska st. 83A, 79019, Lviv, Ukraine*

Abstract

The paper is dedicated to the latest trends in the development of architecture of IoT systems. Special attention is paid to edge and fog computing paradigms which are becoming increasingly common today. The benefits of data processing in IoT nodes are shown and their processing components are overviewed. The expediency of using a reconfigurable hardware in IoT nodes is justified. The challenges of FPGA-based nodes application in IoT systems related to their compliance with the requirements of effective installation, changing functionality, upgrading/optimizing a network without changing functionality, creating configurations for built-in FPGA are shown in the paper. To address these challenges, a method and a framework for nodes self-configuring in IoT systems are proposed. The basic idea of them is the creation and application of a cloud-based service with specialized processor automatic design and synthesis tools. The advantages of the proposed method and framework as well as ways of addressing challenges of FPGA-based nodes application in IoT systems are highlighted.

Keywords

Internet of Things, self-configuration, cloud computing, edge computing, fog computing, reconfigurable computing, specialized processors, automatic design and synthesis tools

1. Introduction

The concepts of the Internet of Things (IoT) and cyber-physical systems have emerged relatively recently, at a time when the computer, information and communication technologies have reached high levels of their development. Although the methodological and technological foundations for the implementation of these concepts were formed earlier, today their development consists not only in the further expand of the application range and the emergence of the newest instances of IoT and cyber-physical systems, from local and simple to large-scale and complex, but also in the further improvements of methods of their design and operation, and the invention of new architectural models and approaches to information processing. If

ICyberPhyS-2024: 1st International Workshop on Intelligent & CyberPhysical Systems, June 28, 2024, Khmelnytskyi, Ukraine

* Corresponding author.
† These authors contributed equally.

viktor.melnyk@kul.pl (V. Melnyk); melnyk.anatoliy@itstep.org (A. Melnyk)
0000-0002-5046-8002 (V. Melnyk); 0000-0002-8981-0530 (A. Melnyk)

cloud computing had been used in the earliest IoT systems, today edge and fog computing IoT systems were becoming more common, in which moderate and even intensive computations are shifted from cloud to IoT nodes. Along with high computing capabilities, a number of requirements is imposed to the latest IoT systems in terms of mobility, serviceability, autonomy, adaptability, self-configuration, etc. [1] - [5].
A significant number of modern IoT systems are equipped with nodes based on reconfigurable hardware platforms, which are implemented as Field-Programmable Gate Arrays (FPGAs). Their use makes it possible to reconfigure the structure of the hardware components implemented in them, thus changing their functional purpose in the system. Furthermore, the specialized processors (SPs), implemented in FPGA-based IoT nodes, are characterized by 2-3 orders of magnitude higher efficiency on set of algorithms they execute as compared to universal processors, resulting in higher performance, lower power consumption and equipment volume [6]. Due to the need of data processing in IoT nodes in edge and fog computing architectures, the SPs have become their conventional components [7].

However, significant design time, high production costs and complex configuring process are obstacles to implementation of SPs in IoT nodes. That is why the configuration and maintenance of such IoT nodes in comparison with fully programmable ones, based on universal processors, is more complex, namely the tasks of setting up and maintaining IoT systems based on such nodes. It requires further development of methods of operation and computational processes organization in the IoT systems built on FPGA-based nodes. This paper proposes a method and a framework for IoT nodes self-configuring, that provide IoT systems built with FPGA-based nodes with essentially new properties: self-configurability, self-optimization, and functional self-adjustment. These also automate the processes of initial configuration, technical maintenance, re-profiling and upgrading, shifting the responsibility of their performance from a human to a system. In addition, these open new directions for the further research and development in the field of IoT.

The paper is structured as follows. In Sect. 2, we summarize the relevant literature with the aim of evaluating different IoT architectural paradigms. In Sect. 3, we show approaches to the implementation of computing components in IoT nodes, the advantages and disadvantages of using universal and specialized processors for this purpose. In Sect. 4, characteristic features of FPGA-based IoT nodes are revealed, the benefits and complexities of using reconfigurable hardware in IoT nodes are shown. In Sect. 5, the challenges of FPGA-based node application in IoT systems are analyzed, which have a huge impact on setup, manageability, and functional re-profiling of IoT systems. To address these challenges, the foundations for self-configuring of FPGA-based IoT nodes, which are based on specialized processors automatic design and synthesis tools, are discussed in Sect. 6, and in Sect. 7 the method and framework for self-configuring of FPGA-based nodes in IoT systems are proposed. In Sect. 8, we formulate the benefits of self-configuring of IoT Nodes and show the ways of addressing the challenges of FPGA-based nodes application in IoT systems. And finally, in Sect. 9 we draw our conclusions.

2. Related Research

Today, there are several views on the architecture of IoT systems, which differ in structuring of its levels. In scientific works [5], [8], three layers of architecture of such systems are described:

- *Perception*: sensors, actuators and edge devices that interact with the environment;
- *Network*: connects devices in a network in coordination with the application layer;
- *Application*: data processing and storage within special services and functionality for users.

In some other scientific papers, in particular, [9], edge devices are taken out into a separate layer, a layer of *edge computing IT systems*. Thus, a four-layer model of IoT systems is formed which consists of sensors and actuators, Internet gateway layer and data acquisition systems, edge IT and cloud analytics.

Edge computing systems are used to perform preliminary analytics and data pre-processing. Some additional processing may also be done here, prior to entering the cloud data server. The goal of edge computing is to filter the data and reduce the amount of massive data sets by economizing resources.

The five-layer IoT architecture [5], [8] enhances the previous one with a *business Layer* which defines the management for the entire IoT system and its functionality, business logic and top-level requirements that need to be coordinated for a sustainable architecture that will be of some consistent value to various business and end users. All three types of IoT architecture, and how the next one enhances previous, are shown in Figure 1.

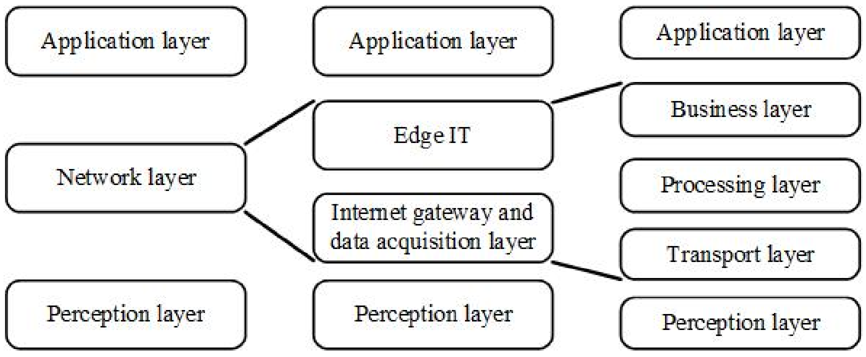

Figure 1: Three types of IoT architecture.

Another tendency in the development of IoT architecture is a *fog computing*. It involves moving individual IoT services closer to the elements of the sensory level, at the edge computing level, in order to speed up decision-making and increase the level of automation [10], [11].

In fog computing IoT architecture the corresponding *fog layer* is introduced which includes several sub-layers: *security*, *storage*, *preprocessing* and *monitoring* [5]. It is placed between the perception and transport layers. According to [12], the following essential characteristics of fog computing help to distinguish it from other computing paradigms:

- Contextual location awareness and low latency;
- Geographical distribution of nodes and support of services to moving vehicles;
- Heterogeneity in terms of collecting and processing of data of different forms acquired through various types of networks;
- Interoperability of the fog computing components;

- Real-time interactions;
- Scalability and agility of fog-node clusters. Fog computing is stated to be adaptive in nature in order to enable network condition variations and changes, resource pooling and data-load changes.

Two additional characteristics associated with fog computing are the predominance of wireless access of the nodes to the network and the mobility support. The following fog node attributes, defined in [12], are distinguished: the heterogeneity for enabling deployment in different environments, manageability and programmability for easy and automated services and changes of the functions performed by compute and storage resources of nodes.

What differs fog computing from the edge computing is that the first one allows dynamic reconfiguration for different applications while performing intelligent computing and transmission services, while the edge computing uses specific applications in a fixed logic location and provides a direct transmission service.

Papers [13], [14] discuss new challenges in IoT systems which require new architectural and technological solutions. Several examples of IoT application in systems that require such solutions are given there.

Among them, resource-constrained IoT devices – sensors, data collectors, actuators, controllers, cars, drones, medical devices embedded in patients and others. Such devices, having very limited resources, must often perform quite computationally intensive tasks while performing various operations, but their direct interaction with the cloud is costly and resource-intensive [15], therefore, impractical.

Considering cyber-physical systems, the five-layer structure proposed in [16] can also be enhanced with the edge computing layer and fog computing layer, taking into account above-mentioned consideration, thus forming the structure shown in Figure 2.

In complex cyber-physical systems, including industrial control systems, smart cities and connected cars and trains, uninterrupted and safe operation is a top priority task. If there is a need, for instance, to install software updates on cars, they have to be brought to technical stations for some time which can result in costs losses. Thus, updating the hardware and software in such cyber-physical systems is severely limited and must be planned in advance.

There is also a number of security issues in IoT systems, requiring their nodes performing cryptographic algorithms and protocols, to be updated in order to withstand new security threads and keep security credentials and resources up to date on large number of nodes.

The edge and fog computing IoT architectures are aimed to address these issues, while performing control, storage, computing and networking functions closer to end user devices.

Along with that, for the further development of IoT systems in order to increase their efficiency and provide them with new quality characteristics, it is necessary to create a theoretical and methodological foundations for building their infrastructure which will enable fast and automatic change of their functionality, perform optimization and updating, adapt to environmental conditions.

Nowadays, significant efforts are put into solving the issue of self-organization and self-adaptation of IoT systems [17] - [19], but they are mostly focused on systems with nodes based on all-purpose programmable processors or microcontrollers. In contrast, in FPGA-based systems, these issues are still investigated insufficiently. Our approach to address these challenges is given out in the following sections of the paper.

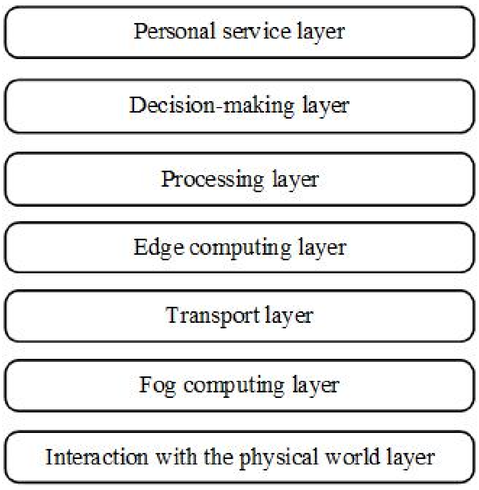

Figure 2: Cyber-physical system multilayer architecture.

3. Data Processing in IoT Node: Resources and Troubles

Depending on the functions carried by IoT node, its structure may vary – it may contain sensors or / and actuators, possess different computational and storage capacity, embedded power supply, etc. In general, a structure of an IoT node contains primary sensors to collect data from physical environment, in which they operate, a data acquisition system (usually containing an analog-digital converter), a processing unit to perform data processing and communication, a memory to store information related to data acquisition and further processing along with a software, specifying the processing algorithms and the operating system software.

To process data in IoT nodes the all-purpose processors and specialized processors are used. Obviously, the use of high-performance all-purpose processors like *Xeon*, *Pentium* or *Core i7* produced by *Intel*, or *Ryzen*, *Athlon* produced by *AMD*, cannot be applied reasonably due to high power consumption. On the other hand, computing capabilities of low-power energy efficient processors, e.g., *G-Series* produced by *AMD* or *ARM Cortex* are not always sufficient for many IoT applications.

The SP architecture is optimized to run only the specified algorithms. Due to the task characteristics consideration and the hardware implementation of algorithms, they achieve higher performance, low energy consumption and low equipment volume as compared to universal processors. This explains why SPs are used today in many important applications, including deep learning and mining cryptocurrencies. SPs of different types and functions are used on different layers of an IoT architecture [16].

There are two major troubles of SP application that IoT node designers are facilitated with. Firstly, it is the high cost of its production as custom integrated circuits. Secondly, a significant increase in its design time due to the overall increase in the complexity of the SP architecture which results from the increased complexity of the algorithms performed and the need to achieve higher performance as well as due to the complicated SP design process because it must ensure the optimum energy consumption / performance ratio.

The following solutions can be applied to resolve these troubles: 1) the use of reconfigurable computing in IoT nodes and 2) the implementation of SPs in the FPGA and the use of high-level software tools (e.g., C2HDL, HDL denotes a hardware description language) for the automatic design and synthesis of SPs. Both solutions will be considered in detail further on.

4. FPGA-Based IoT Nodes Features

In recent years, the share of reconfigurable computing that is mostly FPGA-based has increased. In FPGA, being reconfigurable hardware devices, the components can be arranged in different ways to implement various computing devices [20]. The use of FPGAs is absolutely feasible in IoT nodes as they contain the reconfigurable logic arrays, arithmetic devices and even digital signal processor units for specialized computations, embedded memory for data and program storage, standard interface controllers. In addition, many modern FPGAs contain all-purpose processors that could be used for specialized computational control and decision making. These processors can be integrated into the FPGA chips as full custom regions or delivered in a form of soft cores and synthesized in FPGA together with other specialized computational means. The full-custom microprocessor implementation is, for example, *ARM Cortex* in FPGA and SoC devices from *AMD Xilinx* and *Intel Altera.* The widely known microprocessor soft cores are the VHDL-models of the 32-bit *LEON3* processor with the *SPARC V8* architecture produced by *Aeroflex Gaisler*, the 32-bit embedded processors *Nios II* from *Intel Altera* and *MicroBlaze* from *AMD Xilinx* with the architecture adapted to their own FPGAs.

FPGA-based IoT nodes are quite common today. In nodes, the FPGAs are used to deploy specialized computing and decision-making means. In terms of computing, they can be used to implement specialized processors to execute various algorithms, for example, compression [21], sound processing [22], image processing [23] and cryptographic transformations [24].

Although implemented in reconfigurable hardware SPs possess a number of benefits listed above, their design flow is rather complex and challenging. The first challenge is the need of designing SP IP cores to be implemented in FPGA that require specialists and software packages for IP Cores design, testing, synthesis and implementation, causing material and time expenses, intellectual efforts etc. The second challenge is inability to develop such specialized processors IP Cores, which will be effective on classes of problems. This challenge imposes a critical limitation of FPGA-based IoT nodes: they are effective only for certain classes of problems, for which their SPs IP cores were developed.

Nevertheless, the benefits of FPGA-based IoT nodes made them efficient for a wide range of application domains, where nodes with fixed storage and computing hardware resources fain in their capabilities.

5. Challenges of FPGA-Based Nodes Application in IoT Systems

There is a number of challenges associated with FPGA-based IoT nodes application that are typical for IoT systems, especially with the edge nodes and fog nodes:

- *Challenge 1*: installation of IoT nodes in the network.
  - The installation of IoT nodes and the organization of their operation in the network are carried out in such a way that the topology of the network is initially developed, the types and functions of its elements are determined, then the elements of the network are prepared, mounted in the proper places, and the interaction is set up

and debugged. Each network node must be connected to the computer in order to configure its built-in FPGA.

- *Challenge 2*: changes of IoT node functions in the network.
    - This need may arise as a result of changing the coordinates of the IoT node (the fog node, in particular) and / or the parameters of the environment in which it operates, as a consequence of the IoT node interaction with the environment, other IoT nodes or elements of the network, or the need to change the node functions with time. The changing of the node functions is performed by changing the logic (program) of its operation while keeping the sensor and power supply system as well as the executive system untouched. Example 1: Performing the operation of cryptographic data protection. Example 2: Performing the operation of a sensor / data fusion function, which produces a new metric parameter.
- *Challenge 3*: upgrading / optimizing a network or its individual nodes without changing functionality.
    - The upgrading and optimizing can be performed in order to improve the parameters of the IoT node operation – to reduce the time of function execution and power consumption etc.
- *Challenge 4*: creating configurations for the IoT node built-in FPGA and its configuring, i.e., realization of processors for specialized computations, interface controllers and other components of built-in FPGAs.
    - Firstly, implementing in FPGA requires specialized processors IP cores designing (or getting ready-made solutions from the third parties). Designing process is laborious and requires significant financial and time costs as it involves architectural designing of the specialized processors, IP cores development and debugging by using the hardware description language and their logic synthesis in the target FPGA. As a result of the logic synthesis, the FPGA configuration code is obtained.
    - Secondly, after the specialized processor designing, a problem of loading the configuration code into the IoT node built-in FPGA arises. For its solution, it is necessary to connect the above-mentioned node to a computer with the relevant software installed in order to perform the configuration procedure. Taking into account the scale of the network (that may involve hundreds or thousands geographically distant nodes), one may realize how longstanding and complicated this process can be. Moreover, changing functions or even adjusting operating parameters of the IoT node may require FPGA reconfiguration which makes the network rigid, inert and hardly suitable for modernization or re-profiling.

6. Automatic Design and Synthesis Tools application for Self-Configuring of FPGA-Based IoT Nodes

To enable a rapid and automatic change of IoT nodes functionality, perform optimization and updating, adapt to environmental conditions, we need to create an approach for high-level hardware-independent specification of the operating algorithms which the nodes have to execute in the IoT system. Our task is also to develop a complex of software tools that would receive this hardware-independent specification and create universal microprocessors executable files and FPGA configurations for FPGA-based IoT nodes automatically.

The core of such tools is a software that allows generating the HDL-models of SPs automatically, by inputting the description of the algorithm to be performed in a high-level programming language. This software is primarily aimed to solve above mentioned issues of SPs conventional design flow. In our research the *Chameleon* C2HDL design tool developed by *Intron* was used [25].

In order to generate SP HDL models with the *Chameleon*, the following should be included into the specification:

- The algorithm, which a SP has to execute, represented as ANSI C language software module;
- Test vectors for functional verification (if needed);
- The SP interface: the number of input/output signals and their width;
- Data format (width, signed/unsigned, unfixed/fixed point);
- Constrains for the algorithm execution time (the time between the moment of the last input data item submission to the SP and the moment of the last result coming out from the SP).

The preparation and submission of the listed above is incomparably simpler and faster than a manual design of a SP according to the conventional design process. Therefore, in our research, the *Chameleon* tool (or other similar tools) can perform the automatic design of SPs for FPGA-based IoT nodes.

Our next step, in order to obtain FPGA configurations for FPGA-based IoT nodes, is to compile the design with logic synthesis tools. Such tools are able to perform this task automatically, by inputting the HDL description and design specification – FPGA type and package, pin assignment information, area and performance constrains, etc. *Xilinx ISE*, *Intel Quartus II* are widely used tools for logic synthesis. Such automatic design flow is shown in Figure 3 on the right, while the conventional design flow is depicted on the left.

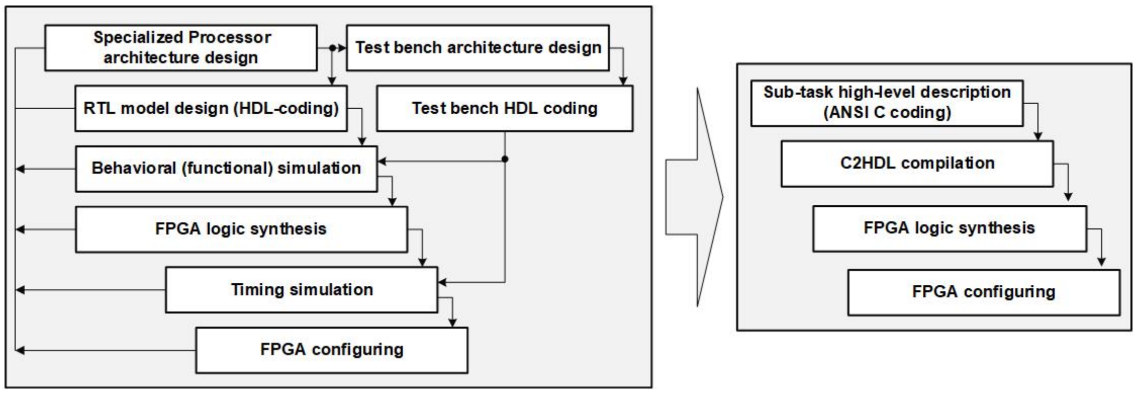
Specialized Processor architecture design
Test bench architecture design
RTL model design (HDL-coding)
Test bench HDL coding
Behavioral (functional) simulation
FPGA logic synthesis
Timing simulation
FPGA configuring
Sub-task high-level description (ANSI C coding)
C2HDL compilation
FPGA logic synthesis
FPGA configuring

Figure 3: Benefits of C2HDL Design Tools application: conventional FPGA-based SP design flow (left) and design flow with C2HDL Design Tools (right).

Finally, there is a need for the high-level hardware-independent specification of the operating algorithms to be split between the IoT node embedded processing units – the universal processor and FPGA. This task can be done with computational load balancing systems, examples of which are given in [26], [27], [28], [29]. Such systems split input source program onto two parts – the first one, comprising mostly of the control flow, should be executed on the universal processor, and the second one, with the computationally intensive data processing parts, should be executed by high-performance computing units. We suggest this second part to be further implemented in FPGA, after automatic C2HDL conversion and SP design and automatic logic synthesis.

Thus, a total chain of SP design and synthesis with the mentioned tools can be performed automatically. Moreover, since these processes should pass in IoT system without human assistance, and imply creating and acquiring of the hardware structure (on the level of FPGA), we can talk about the self-configuring of FPGA-based nodes in IoT system. This process is illustrated in Figure 4.

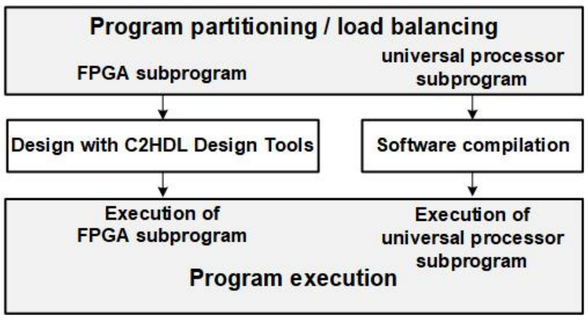
Program partitioning / load balancing
FPGA subprogram
universal processor subprogram
Design with C2HDL Design Tools
Software compilation
Execution of FPGA subprogram
Execution of universal processor subprogram
Program execution

Figure 4: Diagram of an FPGA-based IoT node self-configuring.

7. Method and Framework for Self-Configuring of FPGA-based Nodes in IoT Systems

Suppose the operation of nodes in IoT system is defined by the software modules written on high-level programming language, for example C. These software modules can be refreshed or updated periodically when the IoT system optimization or changes of the IoT node functions in the network take place. The set of software modules comprises a full range of operational algorithms the nodes can execute while functioning in IoT system, by preserving the abilities to change the functionality, being mobile, manageable and highly productive. The function of each node in an IoT system is defined according to its *coordinates*, *state* and *environment parameters*.

We propose placing these software modules into a repository (a library) at the IoT processing layer and indexing them according to functionality they specify.

The IoT node sends its coordinates, state and environment parameters to the IoT processing layer, together with the parameters of its embedded universal microprocessor and FPGA, in the following cases:

- at the time of the IoT system installation and its initial configuration,
- during the IoT system reprofiling or modernization,

- when its coordinates, state or environment parameters are changed (such behavior should be defined additionally according to IoT system operation characteristics).

At the IoT processing layer, the index of the software module P_{HLL}, which specifies the function of the node in the IoT system, is determined according to the node coordinates, state and environment parameters: $index = f(C,S,EP)$; $P_{HLL} = SWML[index]$, where C is IoT node coordinates, S – its state, and EP – its environment parameters in the IoT system, $SWML$ – software modules library. This module is then taken from the repository and sent, together with the parameters of the IoT node embedded universal microprocessor (*UPP*) and reconfigurable hardware platform, i.e., FPGA (*RHPP*), to the cloud-based automatic design and synthesis tools, the complex of which is hereinafter referred to as the *IoT Node Configuration Generation System*. When the IoT Node Configuration Generation System receives the above software module and parameters, it performs the following operations:

- automatically selects the most computationally intensive fragments in the software module and splits it onto two subprograms – the first one, SP_{UP}, for the universal processor, and the second one, SP_{RHP}, which is constructed from the selected fragments, for the reconfigurable hardware platform (i.e., FPGA): $\{SP_{UP}\&SP_{RHP}\} = CLB(P_{HLL})$, where *CLB* denotes the computational load balancing;
- performs the compilation of the first subprogram into the executable file *obj* suitable for the universal processor architecture and creates from the second subprogram the FPGA configuration file *conf* corresponding to FPGA parameters. The sequence of transformations performed during the SP_{RHP} subprogram compilation includes the automatic generation of the SP HDL source code and its further logic synthesis with the corresponding design tools: $conf = LS(HDLG(SP_{RHP}))$, where *LS* denotes the logic synthesis, *HDLG* – SP HDL source code generation;
- returns two files, *obj* and *conf*, to the IoT node.

The IoT node acquires the configuration automatically after it receives the above-mentioned files.

The framework for IoT nodes self-configuring in the network is illustrated in Figure 5.

To carry out the above-mentioned actions, the IoT Node Configuration Generation System will contain a complex of software tools, namely:

- Computational load balancing tools for splitting the input program module into two subprograms, for the universal processor and FPGA respectively.
- Compiler(s) for the universal processor subprograms compilation from the input language they are written, into the object codes that may be directly executed by the universal processor of the IoT node (the processor parameters are considered).
- Tools for the automatic generation of the specialized processors HDL models from the RHP subprograms (the FPGA parameters are considered).
- Logic synthesis tools for specialized processors HDL models synthesis for the target FPGAs (the FPGA parameters are considered).

There are two sets of the FPGA parameters. The first set consists of the FPGA type and series, the package and the number of pins. It is the input information for the logic synthesis tools. The second set of parameters are as follows: the amount and organization of the embedded memory units, the arithmetic and logic devices, the basic logic elements and the input/output blocks amount, the operating clock frequency and the energy consumption constrains. These parameters are necessary for the specialized processor HDL model's generation tools to determine the specialized processor characteristics, particularly, the number of parallel computational units, the maximum instruction memory size, the interface capacity, etc.

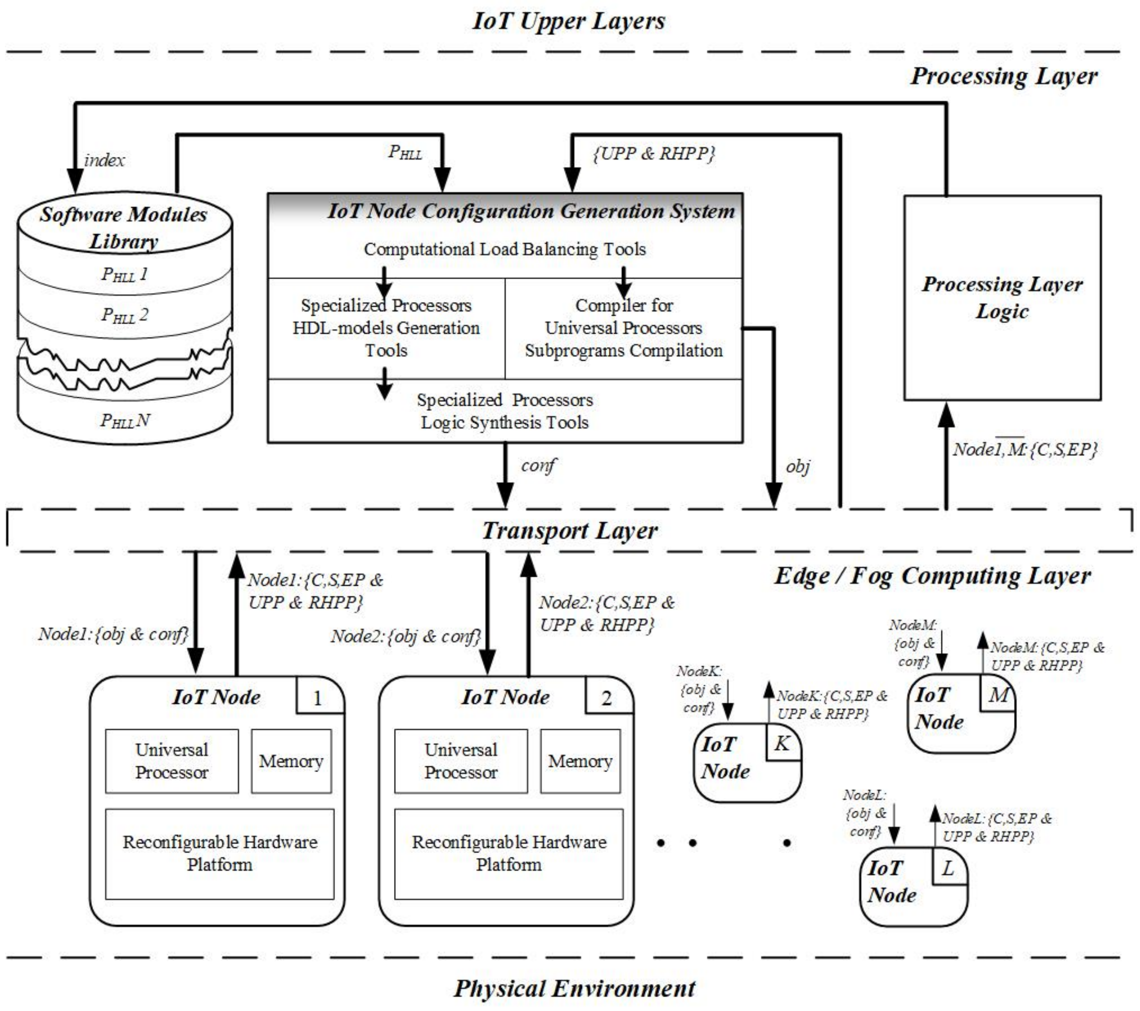
IoT Upper Layers
Processing Layer
index
P_HLL
{UPP & RHPP}
Software Modules Library
P_HLL 1
P_HLL 2
P_HLL N
IoT Node Configuration Generation System
Computational Load Balancing Tools
Specialized Processors HDL-models Generation Tools
Compiler for Universal Processors Subprograms Compilation
Specialized Processors Logic Synthesis Tools
Processing Layer Logic
conf
obj
Node1,M:{C,S,EP}
Transport Layer
Edge / Fog Computing Layer
Node1:{obj & conf}
Node1:{C,S,EP & UPP & RHPP}
Node2:{obj & conf}
Node2:{C,S,EP & UPP & RHPP}
IoT Node 1
Universal Processor
Memory
Reconfigurable Hardware Platform
IoT Node 2
NodeK: {obj & conf}
NodeK:{C,S,EP & UPP & RHPP}
IoT Node K
NodeM: {obj & conf}
NodeM:{C,S,EP & UPP & RHPP}
IoT Node M
NodeL: {obj & conf}
NodeL:{C,S,EP & UPP & RHPP}
IoT Node L
Physical Environment

Figure 5: The proposed framework for self-configuring of FPGA-based nodes in IoT system with the automatic design and synthesis tools.

8. The Benefits of IoT Nodes Self-Configuring

The basic methodological and technological approaches which are used in the framework for IoT nodes self-configuring are the following:

- the method of self-configuring of the computer system with reconfigurable logic, which enables the automation of the specialized processor HDL models design and synthesis processes [30];
- the "Software as a Service" model for software delivering via a network, which enables the above-mentioned processes to be performed as a service for the IoT nodes;
- the IoT technology itself, which is aimed to ensure that the configuration creation is initiated and transferred by the IoT node without human assistance.

The proposed method and the framework for IoT nodes self-configuring imply the dynamic reconfiguration of FPGA in the IoT nodes using the cloud-based automatic design and synthesis service, which allows to provide IoT systems with advanced capabilities of automatic adaptation, function changing, upgrading and optimizing. It is aimed to bring solutions to the challenges specified in Sect. 5 of the paper, as follows:

- *Solution to the challenge 1* (installation of the IoT nodes in the network):
  - In order to get rid of the need to create the configurations of all IoT nodes in advance, and then mount them in the envisaged locations, it is possible to pre-place them at the locations in the network, and then each node, by providing its coordinates and environment parameters, will automatically receive the necessary configuration of the FPGA in order to perform the functions corresponding with these coordinates and the parameters of the environment.
- *Solution to the challenge 2* (changes of the IoT node functions in the network):
  - The changing of the IoT node functions is traditionally performed by the way of demounting each of them from a network location, connecting to a computer to change the configuration of its built-in FPGA, and re-mounting back to a location, or simply by replacing the node with a new one with the desired configuration. The volume of work is extremely large if the network consists of many territorially distanced nodes, and in the worst case is close to re-installation of the network.
  - In order to eliminate the need for manual demounting of the IoT node, its reconfiguration and re-installation, the node can provide the relevant information to the cloud and obtain a new FPGA configuration automatically. The FPGA reconfiguration is also possible on the initiative of the network control nodes to change network functions.
- *Solution to the challenge 3* (upgrading / optimizing of a network or its individual nodes without changing functionality):
  - The upgrading of the IoT node is performed in the same way as the changing of functions, as described above. In order not to reconfigure the required IoT node manually, it is possible to do this remotely and automatically, for example, by replacing programs that specify the functions of the nodes and the corresponding change in the configuration of their built-in FPGAs.
- *Solution to the challenge 4* (creating the configurations for the IoT node built-in FPGA and its configuring):
  - The proposed framework for IoT nodes self-configuring enables to automate the processes of configuration, technical support, re-profiling and upgrading (modernization) of the networks which involve FPGA-based IoT nodes and replace

the responsibility of their execution from the human to the system that applies cloud-based service with the automatic design and synthesis tools.

In addition to solving the challenges described above, the proposed framework brings essentially new properties to IoT systems with FPGA-based nodes – flexibility, functional self-adjustment and adaptability. The latter property can be implemented at the level of hardware (the concept of adaptive hardware: the optimization of the FPGA configuration) and at the level of software (the concept of self-adaptive software: the optimization of the IoT node operation algorithm, which is performed autonomously). Finally, the proposed technology opens up the opportunity for creating smart networks, that will be, in essence, a new generation of networks for IoT and cyber-physical systems.

9. Conclusions

The proposed method and framework for FPGA-based node self-configuring in IoT systems use the method of self-configuring of the computer system with the reconfigurable logic, the "Software as a Service" model, the Internet of Things technology and implies the automatic generation of the specialized processors IP cores, their compilation and logic synthesis allowing the specialized processors automatic design and synthesis to be performed as a service for the FPGA-based IoT nodes. The framework and method are intended to solve several challenges that are typical for their application, namely: (a) installation of the IoT nodes in the network, (b) changes of the IoT node functions in the network, (c) upgrading / optimizing a network or its individual nodes without changing functionality, and (d) creating configurations for the IoT node built-in FPGA and its configuring.

The framework and a method for FPGA-based IoT node self-configuring enable the automation of the processes of initial configuring, technical support, re-profiling and upgrading (modernization) of the IoT systems which involve FPGA-based nodes and replaces the responsibility of their execution from the human to the system that applies cloud-based service with the automatic design and synthesis tools. Additionally to flexibility and adaptability, they provide IoT systems built with FPGA-based nodes with essentially new properties – self-configurability, self-optimization, functional self-adjustment and re-profiling. It is a step to creating *smart networks*, a new generation of networks for IoT and cyber-physical systems.

References

[1] S. Naveen and M. R. Kounte, Key Technologies and challenges in IoT Edge Computing, 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, pp. 61-65, doi: 10.1109/I-SMAC47947.2019.9032541.

[2] R. P. Kumar and S. Smys, A novel report on architecture, protocols and applications in Internet of Things (IoT), 2018 2nd International Conference on Inventive Systems and Control (ICISC), 2018, pp. 1156- 1161, doi: 10.1109/ICISC.2018.8398986.

[3] S. A. Goswami, B. P. Padhya and K. D. Patel, Internet of Things: Applications, Challenges and Research Issues, 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019, pp. 47-50, doi: 10.1109/I-SMAC47947.2019.9032474.

[4] N. Lata and R. Kumar, Internet of Things: A Review of Architecture and Protocols, 2020 International Conference on Decision Aid Sciences and Application (DASA), 2020, pp. 1027-1031, doi: 10.1109/DASA51403.2020.9317091.

[5] A. Calihman, Architectures in the IoT Civilization, January 30, 2019. Available online: https://www.netburner.com/learn/architectural-frameworks-in-the-iot-civilization/ (accessed on 19 May 2024).

[6] N. Thompson, S. Spanuth, The Decline of Computers as a General Purpose Technology: Why Deep Learning and the End of Moore's Law are Fragmenting Computing (November 20, 2018). Available at SSRN: https://ssrn.com/abstract=3287769 (accessed on 19 May 2024).

[7] I. Sultan, M. T. Banday, "A Study of the Design Architectures of Configurable Processors for the Internet of Things," 2018 3rd International Conference on Contemporary Computing and Informatics (IC3I), 2018, pp. 320-325, doi: 10.1109/IC3I44769.2018.9007256.

[8] H. Rahimi, A. Zibaeenejad, A. A. Safavi, A Novel IoT Architecture based on 5G-IoT and Next Generation Technologies, to be presented at IEEE IEMCON conference, Vancouver, BC, Canada, Nov. 2018. Available online: https://arxiv.org/ftp/arxiv/papers/1807/1807.03065.pdf (accessed on 19 May 2024).

[9] S. Vanitha, P. Balasubramanie, K.S. Arvind, V.R. Niveditha, Saravanan Elumalai, "Internet of Things: a Review on Their Requisite in The Digital Era," International Journal of Advanced Science and Technology Vol. 29, No. 9s, (2020), pp. 591-602.

[10] Pallavi Sethi, Smruti R. Sarangi, Internet of Things: Architectures, Protocols, and Applications. Journal of Electrical and Computer Engineering / 2017 (1):1-25.

[11] M. Yannuzzi, R. Milito, R. Serral-Gracia, D. Montero, and M. Nemirovsky, "Key ingredients in an IoT recipe: fog computing, cloud computing, and more fog computing," in Proceedings of the IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD '14), pp. 325–329, Athens, Greece, December 2014.

[12] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, and C. Mahmoudi, "Fog computing conceptual model," 2018. NIST Special Publication 500-325.

[13] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," IEEE Internet Things J., vol. 3, no. 6, pp. 854-864, Dec. 2016, doi: 10.1109/JIOT.2016.2584538.

[14] P. Pieta, S. Deniziak, R. Belka, Miroslaw Plaza, Malgorzata Plaza, Multi-domain model for simulating smart IoT-based theme parks, in Proc. SPIE 10808, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018, 108082T (1 October 2018); doi: 10.1117/12.2501659.

[15] S. Lysenko, O. Bondaruk, Advanced Methods for Maintaining and Managing the Life Cycle of Cloud Environments: Survey. Computer Systems and Information Technologies, (1), 2024, p. 39–45. https://doi.org/10.31891/csit-2024-1-5.

[16] A. Melnyk, Cyber-physical systems multilayer platform and research framework, Advances in cyber-physical systems, 1 (1), 2016, pp.1- 6.

[17] A. P. Athreya, B. DeBruhl and P. Tague, Designing for self-configuration and self-adaptation in the Internet of Things, 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, Austin, TX, 2013, pp. 585-592.

[18] M. T. Moghaddam, E. Rutten, P. Lalanda, G. Giraud, "IAS: an IoT Architectural Self-Adaptation Framework," 14th European Conference on Software Architecture (ECSA), Sep. 2020, L'Aquila, Italy. ffhal-02900674.

[19] I. Chatzigiannakis et al., True self-configuration for the IoT, 2012 3rd IEEE International Conference on the Internet of Things, Wuxi, 2012, pp. 9-15.

[20] R. Tessier, K. Pocek, A. DeHon, Reconfigurable Computing Architectures, in Proceedings of the IEEE, vol. 103, no. 3, pp. 332-354, March 2015, doi: 10.1109/JPROC.2014.2386883.

[21] J. Chen, M. Daverveldt, Z. Al-Ars, FPGA Acceleration of Zstd Compression Algorithm, 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2021, pp. 188- 191, doi: 10.1109/IPDPSW52791.2021.00035.

[22] B. da Silva, L. Segers, Y. Rasschaert, Q. Quevy, A. Braeken, A. Touhafi, A Multimode SoC FPGA-Based Acoustic Camera for Wireless Sensor Networks, 2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2018, pp. 1-8, doi: 10.1109/ReCoSoC.2018.8449381.

[23] R. Cheour, S. Khriji, D. E. Houssaini, M. Baklouti, M. Abid, O. Kanoun, "Recent Trends of FPGA Used for Low-Power Wireless Sensor Network," in IEEE Aerospace and Electronic Systems Magazine, vol. 34, no. 10, pp. 28-38, 1 Oct. 2019, doi: 10.1109/MAES.2019.2901134.

[24] H. T. Huynh, T. K. Tran, T. P. Dang, T. T. Bui, "Security Enhancement for IoT Systems Based on SoC FPGA Platforms," 2020 4th International Conference on Recent Advances in Signal Processing, Telecommunications Computing (SigTelCom), 2020, pp. 35-39, doi: 10.1109/SigTelCom49868.2020.9199016.

[25] A. Melnyk, V. Melnyk, Specialized Processors Automatic Design Tools-the Basis of Self-Configurable Computer and Cyber-Physical Systems. 2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019 - Proceedings, pp. 326-335. DOI:10.1109/ATIT49449.2019.9030481.

[26] V. Melnyk, V. Stepanov, Z. Saraireh, Computational Load Balancing System Between the Host-Computer and Self-Configurable Accelerator, Scientific Bulletin of Chernivtsi National University "Computer systems and components". 3 Issue 1 (2012) pp. 6-16.

[27] S. Amiri, S. Abdi, S. Sharifzadeh, "Simultaneous Multiprocessing on FPGA-CPU Heterogeneous Chips," 2021 22nd IEEE International Conference on Industrial Technology (ICIT), Valencia, Spain, 2021, pp. 805-809, doi: 10.1109/ICIT46573.2021.9453638.

[28] S. Amiri, M. Hosseinabady, A. Rodriguez, R. Asenjo, A. Navarro and J. Nunez-Yanez, "Workload Partitioning Strategy for Improved Parallelism on FPGA-CPU Heterogeneous Chips," 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, Ireland, 2018, pp. 376-3764, doi: 10.1109/FPL.2018.00071.

[29] M. Khatti, X. Tian, Y. Chi, L. Guo, J. Cong, Z. Fang, "PASTA: Programming and Automation Support for Scalable Task-Parallel HLS Programs on Modern Multi-Die FPGAs," 2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Marina Del Rey, CA, USA, 2023, pp. 12-22, doi: 10.1109/FCCM57271.2023.00011.

[30] A. Melnyk, V. Melnyk. Self-Configurable FPGA-Based Computer Systems, Advances in Electrical and Computer Engineering, vol. 13, no. 2, 2013, pp. 33-38.