A Methodical Approach to Designing Secure Adaptive Embedded Systems Implemented Via Reconfiguring their Structures

Serhii Shtanenko [1], Yurii Samokhvalov [2] and Oleksiy Silko [3]

[1] *Military Institute of Telecommunication and Information Technologies named after the Heroes of Kruty, Moskovska str. 45/1, Kyiv, 01011, Ukraine*
[2] *Taras Shevchenko National University of Kyiv, Volodymyrs'ka str. 64/13, Kyiv, 01601, Ukraine*

Abstract

The article proposes a methodical approach to designing secure adaptive embedded systems which is based on their ability to restore their proper functionality by way of reconfiguring hardware components, this process being based on the results of self-control. This approach includes two stages. The first stage involves the initial allocation of tasks to a system's hierarchy levels and nodes followed by their reallocation caused by the system failure in the wake of adverse exposure. The second stage sees the reconfiguration of the system implemented to restore its proper functionality by means of automatic hardware reprogramming.

Keywords[1]

Adaptation, survivability, embedded system, programmable logic devices.

1. Introduction

Embedded Systems are specialised microprocessor-based systems whose development concept is based on their interaction with a controlled object and which are immediately built into the device controlled by them [1].

At present embedded systems are widely used in various fields of activity such as mechanical engineering, machine-tool construction, aviation, automobile production, nuclear energy, banking, military-industrial complex. As well as that, they are used as computer-aided systems and means of regulatory control of technological processes.

It is to be noted that early embedded systems were developed as dedicated digital devices with the use of integral circuits having low- and high-degree integration. However, since the advent of microcontroller- and microprocessor-based hardware, as well as programmable logic devices that came later, the concept of embedded system has undergone radical change. If early embedded systems were dedicated structures that contained a central processing unit, separate integral circuits for peripheral controllers, digital data storage media, present-day embedded systems make use of system-on-chip technologies (SoC) [2].

A system-on-a-chip or SoC is a computing system whose architecture is specifically developed to solve an application task (or a class of problems) and implemented as a complex of algorithm-specific hardware and software which is based on the configurable microelectronic platform [3].

Apart from that, it is to be noted that the designing of modern embeddable systems which are based on System-on-Chip technologies makes use of hi-tech computer-aided design software for digital devices, this requiring developers to have deep knowledge of digital circuit engineering, computer architectures, methods of synthesising dedicated microprogram-control devices, as well as knowledge of high-level languages and controllable synthesis methods. Considering the above, we

Information Technology and Implementation (IT&I-2022), November 30 - December 02, 2022, Kyiv, Ukraine
EMAIL: sh_sergei@ukr.net (A. 1); yu1953@ukr.net (A. 2); oleksiy.silko@viti.edu.ua (A. 3)
ORCID: 0000-0001-9776-4653(A. 1); 0000-0001-5123-1288 (A. 2); 0000-0001-6605-6945 (A. 3)

may draw a conclusion that the development of modern embedded systems is a process of designing proper components and using standard digital components of intellectual property that serve not only as circuit design description, but are in effect full form documents for functional and parametric modelling, verification and production with modern technologies being used [4].

At present the questions relating to the designing of dedicated microprocessor systems receives a lot of scientific attention. So, paper [5] examines the issue of improving the performance of microprocessors used in access control systems. The paper outlines the requirements and proposes a set of commands required for quality microprocessor designs based on residue number systems and used for access control. Paper [6] provides a review of facilities for designing embedded microprocessor systems relying on programmable logic devices and examines software debugging tools for microprocessor systems based on such core families as Pico Blaze, Micro Blaze and Power PC. Paper [7] reviews the issues of regularising embedded microprocessor systems as well as the synthesis of hardware componentof embedded systems by means of variable logic with the program-controlled automaton model being used. The paper [8] examines the need to ensure prompt (crisis) response to cyber incidents within a limited time frame and determines the improvement of the information decision-making model.

However, the analysis carried out shows that as of today, the issues relating to health assessment of embedded systems have not been dealt with exhaustingly as regards their proper functionality in the event of adverse exposure (e.g. one producing ionising and electromagnetic effects, cyber-attacks by hardware and software Trojan horses, network worms, DDoS attacks etc.), the same applying to issues of automatic online restore of the system in question which is based on self-checking results.

Considering the above, the purpose of this article is to develop a methodological approach to designing adaptive embedded systems which are able to work when adversely exposed and which are based on integral circuits with soft structures.

2. Adaptation as a Means of Enhancing Survivability of Embedded Systems

Adaptation is understood as a change of parameters and structure of the system and, possibly, of management action based on processing data which is implemented with the purpose of achieving a certain optimal system status with initial uncertainty under changing working conditions [9].

The use of the adaptivity principles allows:

the designing of the system in compliance with the requirement as to its quality when the initial (a priori) information about controlled objects and exposure is limited;

correspondence of control quality to the specified requirements when the system in the controlled object is working and receiving exposure, this being done via rational adaptation of the system to changing conditions [10].

Figure 1 shows the generalised structure of the adaptive system, which can be represented as a two-level construction.

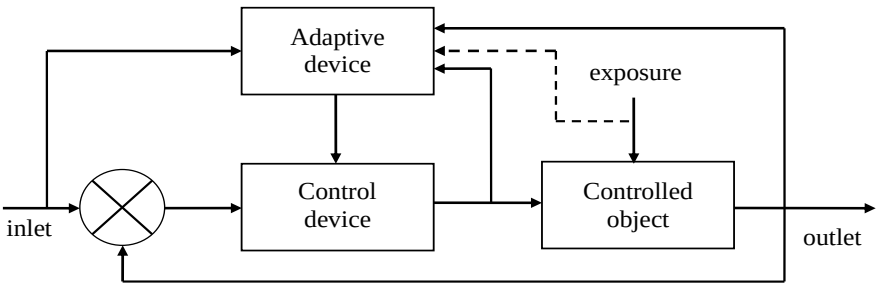

Figure 1: Flow diagram of the adaptive system

The lower level represents the master control system made up of the control object and control device. The upper level is the combination of devices used to modify the properties of the master control systemas required by the stated adaptation aims. The adaptation device that constitutes the upper level of the construction and receives data about exposure, system statusand monitoring

signalsfrom the control object's inlet. On analysing this data, it establishes how much the quality criterionfor the basic system corresponds to the requirements applicable toit. If the quality criterion for these requirements is not met, the adaptation device generates commands that change the structure and parameters or only parameters of the device in such a way that the quality criterion is harmonised with the existing requirements.

If one examines the adaptation of embedded systems to adverse exposure, one can see that is directly associated with such a property of complex systems as survivability. The property that is activated (by a properly organised structure and via its operation) to withstand adverse exposure and implement its functions in specified exposure conditionsis referred to as the system survivability [11]. Owing to this property, the failure of any subsystem or section of the system results in its reduced performance rather than the failure of the whole system. When evaluating the survivability of hierarchy structures, we assume the existence of a certain degree of functional, structural and information redundancy. According to [12], the maintenance and enhancement of survivability of complex hierarchic technical systems is provided by advanced mechanisms for recognition, counteraction and restore, as well as by special means of adaptation, reconstruction, reconfiguration and reorganisation.

Papers [4, 13] propose the use of crash-proof programmable microprocessor systems as hardware components, which belong to the class of reconfiguration devices and are able to employ system-on-chip technologies. Apart from that, paper [14] proposes the use of the service processor for diagnosing and automatic restorationof microprocessor systems. Here the main function of the service processor is to diagnose a microprocessor system and make decisions based on troubleshooting informationfor the reconfiguration of the system via reprogramming integrated-circuit logic, this beingcarried out to restore proper functionality. Thus, the service processor combined with the embedded system effectively serves as a closed cycle of the adaptive microprocessor.

It is to be noted that the latest time has seen the arrival of a new type of control systems, namely distributed microprocessor-based control systems. In numerous cases, the architecture of these systems does have a central processing unit as all control functions are implemented on-device by local control cells based on microprocessors (often built into core process equipment). Frequently, in structural terms distributed (spatially) microprocessor-based control systems are hierarchic control systems. Notably, the upper-level control functions can be implemented both by a personal computer and microprocessor.

In designing such systems, a major task is to synthesise the structure which defines the internal organisation of and relatively stable dependency between the elements of the system. When solving this problem, we will use the fundamental principles of the decomposition-aggregation approach [15]. According to this approach, the synthesis of the distributed microprocessor-based control system is understood as a successive solution to problems relating to the synthesis of the main elements and parts of the system which includes:

determining the number of a system's hierarchy levels and nodes;

task allocation (reallocation) to a system's hierarchy levels and nodes;

a choice of computing systems capable of reconfiguring the underlying hardware structure.

3. Determining the Number of a System's Hierarchy Levels and Nodes

An embedded system is a microprocessor-based control system whose elements are spatially distributed between separate components of the complex controlled technological process. For this reason, the organisational design of the system is to be created in conformity with the principle of compatibility between its design and the organisational structure of the controlled technological process. In other words, based on the necessity of some or other equipment having a microprocessor system, one needs to build the map $\aleph : N \times M \to N_1 \times M_1$, where N, M is the number of hierarchy levels of the design of controlling elements and the number of these elements whereas N_1, M_1 is the number of hierarchy levels and nodes of the embedded system.

4. Task Allocation (Reallocation) to a System's Hierarchy Levels and Nodes

Task allocation (reallocation) to a system's hierarchy levels and nodes consists of the following subtasks: ones involving initial allocation and ones involving reallocation in the wake of adverse exposure.

Task allocation to a system's hierarchy levels and nodes is a typical task for designing complex technical systems. To minimise the time spent on the exchange of information between the system levels, the standard practice is to allocate to each level the tasks that have maximum interdependency when the system is at work [16].

Now we will pose an allocation problem. Let there be given a set of problems $F = \{f_i\}, \overline{1, n}$ of the embedded system which has m levels. Any of tasks $f_i \in F \times F^*$ can be performed only on one level, with F^* being a set of tasks that have already been allocated.

We will introduce the allocation parameter x_{ik}:

$$x_{ik} = \begin{cases} 1, \text{if task } i\text{ - th task is implemented at the k - th level, } k = \overline{1, m}, \\ 0, \text{alternatively} \end{cases}.$$

The fulfilment of condition $\sum_{i=1}^{n} x_{ik} = 1, \quad k = \overline{1, m}, i = \overline{1, n}$ is mandatory, this meaning that every task can be allocated only to one level.

Then we will use a_{ij} to represent the algorithmic connection of task i-th with task j-th (relative frequency of the executing task f_i on completing task f_j), B_i is the memory space necessary for executing task i, $B^{(k)}$ is the available memory space of the computing tools at level k.

With the designations given above, task allocation can be represented as the linear integer programming problem coming below: to find

$$\min \sum_{k=1}^{m} \sum_{i=1}^{n} a_{ij} x_{ij}, \tag{1}$$

with constraints

$$\sum_{i=1}^{n} B_i x_{ik} \le B^{(k)}, \quad k = \overline{1, m}, \sum_{i=1}^{n} x_{ik} = 1, \quad x_{ik} = 0,1. \tag{2}$$

The problem solution will be a cluster of vectors $x_1^* = (x_{11}, ..., x_{i1}, ..., x_{n1})$; $x_2^* = (x_{12}, ..., x_{n2})$; …; $x_m^* = (x_{1m}, ..., x_{nm})$; with the corresponding ones $x_{ik} = 0,1$, which ensure the implementation of (1).

This problem can be interpreted as that of partition of the apex-directed weighted graph $G =< Y, V >$, where the memory space occupied by task i-th B_i is placed in correspondence with apexes Y and the algorithmic connection of task i-th with j-th a_{ij} is made congruent with the set of circular-arc graphs V. The solution to the problem lies in partitioning graph G into $k = m$ of sub graph $< G_k >$ that meet conditions $Y = \bigcup_{k=1}^{m} Y_k$; $\bigcap_{k=1}^{m} Y_k = \varnothing$ and the requirements of the minimum objective function (1) with constraints imposed on other parameters (2).

The application of one or another method to solving this problem largely depends on the number of apexes of the graph. The graph dimension is conventionally defined as small ($n \le 6$ number of apexes); middle ($6 \le n \le 30$) and high ($n > 30$) [17]. As for small-dimension graphs, it is expedient that exhaustive algorithms are applied to them. The branch-and-bounds method works best with middle-dimension graphs. As regards higher-dimension graphs, these methods if applied to them take much time. On account of this, the methods that are most used for partitioning middle-dimensionand, all the more, higher-dimension graphs (this corresponding to embedded system graphs) are heuristic algorithms proposed in [18].

The second subtask is the reallocation of tasks a system's hierarchy levels and nodes in the wake of their failure. As of today, there are two approaches applied to organising task reallocation in a system in the event of node failures. The first approach, a static one, assumes that a subset $S = \{s_\nu\}$ of system states for which task reallocation is necessary is known and it is required to find a set $\Gamma = \{G_\nu\}$ of task allocations where each allocation G_ν, which corresponds to state s_ν, satisfies the requirements for the chosen indicators. In this case, the optimal allocations G_ν for all states $s_\nu \in S$ are available when designing the system, and each node when building the system is provided with resources necessary to perform tasks both in state s_0 and in each of states $s_\nu \in S$ as pursuant to allocation G_ν. When the system is changed to state $s_\nu \in S$ and a failure is detected in all the operable nodes of the system, the tasks corresponding to allocation G_ν are getting started.

The second approach, which is referred to as a dynamic approach, is based on the fact that the task of finding the optimal allocation G_ν for state s_ν is performed each time (with the help of a special solver, i.e. service processor) when the system enters state s_ν and its result may depend on the previous states of the system. In general, in both static and dynamic approaches, obtaining an optimal allocation may require task reallocation of both failed and operable nodes.

Let us consider a static approach to organising task reallocation, assuming that only failed nodes are reallocated when the system enters state s_ν [19].

For each state $s_\nu \in S$, find the allocation G_ν satisfying the conditions:

$$\Phi_\nu = \max\ ;\ C_\nu \le C_{\nu\ addit}\ ,\ T_i^\nu \le T_{i\ addit}\ ,\ i = 1, ..., g_\nu\ ,$$

where C_ν is the consumption required for the transfer of tasks carried out by failed nodes to operable ones state s_ν; T_i^ν is the average time for handling the declaration in node M_i which is in state s_ν; $C_{\nu\ addit}$, $T_{i\ addit}^\nu$ are permissible values of the specified indicators for state s_ν.

We shall designate task $\Re_j$ which is to be executed at the initial allocation to node M_i, as $\Re_{ji}$. Let the initial allocations G_0 be assigned by matrix $\|\beta_{ji}\|$, $j = 1, ..., L$, $i = 1, ..., n$. Element β_{ji} is the weight of the task $\Re_j$, performed with the initial allocation given to node M_i, defined as $\beta_{ji} = \sum_{\varphi_{ji}^s \in F_{ji}} \rho_{ji}^s$, where $F_{ji} = \{\varphi_{ji}^s\}$ is a set of production processes which requires task $\Re_{ji}$ to be performed; ρ_{ji}^s is the weight of the production process φ_{ji}^s, $s = 1, ..., h_{ji}$, $h_{ji} = |F_{ji}|$. Let $D_\nu^f = \{M_l\}$, $D_\nu^r = \{M_i\}$ be sets of failed and operable nodes to be applied to state s_ν; A_ν^f is a set of all tasks performed node $M_l \in D_\nu^f$ in state s^0.

Then the equation used to determine the system efficiency in state s_ν, which is defined by the performance of the operable nodes in state $\Re_{ji} \in A_\nu^f$, can be expressed as

$$\Phi_\nu^f = \sum_{\Re_{jl} \in A_\nu^f} b_{jl} \beta_{il}\ ,$$

where $b_{jl} = 1$, if task $\Re_{jl}$ is otherwise performed by one of the operable nodes $b_{jl} = 0$ when being in state s_ν.

Let $C_{(jl)i}$ be the consumption required to implement the transfer of task $\Re_{jl}$ ($\Re_{jl} \in A_\nu^f$) to some node M_i, where $M_i \in D_\nu^r$ and values $C_{(jl)i}$, are independent of l and allocated by matrix $\|C_{ji}\|$.

Then the consumption required to implement the transfer of a fixed task $\Re_{jl}$ to any of the operable nodes in state s_ν is

$$C_{\nu(jl)} = \sum_{M_i \in D_\nu^r} d_{jl}^i C_{(jl)i},$$

where $d_{jl}^i = 1$, if task $\Re_{jl}$ is otherwise transferred to node M_i, $d_{jl}^i = 0$.

In the event of failure of node M_l, its task $\Re_{jl}$ can either be discarded, i.e. not transferred to any of the operable nodes, or assigned to only one of the nodes of set D_ν^r. In the former case $d_{jl}^i = 0$ for all $M_i \in D_\nu^r$ and $b_{jl} = 0$, in the second case only one of the values $d_{jl}^i = 1$ and hence $b_{jl} = 1$.

Therefore,

$$b_{jl} = \sum_{i=1}^{g_\nu} d_{jl}^i,$$

where $d_{jl}^i \in \{0,1\}$ and $b_{jl} \in \{0,1\}$ are valid.

The total consumption required to implement the reallocation of all tasks $\Re_{jl} \in A_\nu^f$, is expressed as

$$C_\nu = \sum_{\Re_{jl} \in A_\nu^f} C_{\nu(jl)} = \sum_{\Re_{jl} \in A_\nu^f} \sum_{M_i \in D_\nu^r} d_{jl}^i C_{(jl)i}.$$

The average time for a handling declaration in node M_i which is in state s_ν is determined by the expression

$$T_i^\nu = T_i^0 + \sum_{\Re_{jl} \in A_\nu^f} d_{jl}^i \Delta T_{(jl)i}$$

where $\Delta T_{(jl)i}$ is the increment of the average time for handling declaration in the operable node M_i with a task $\Re_{jl}$ being transferred to it, where $\Re_{jl} \in A_\nu^f$; values $\Delta T_{(jl)i}$ being independent of l are assigned by matrix $\|\Delta T_{jl}\|$.

Thus, finding the optimal allocation of tasks C_ν in the system for a fixed state s_ν boils down to solving an integer linear programming problem, which consists in finding a set of values of variables d_{jl}^i, where $d_{jl}^i \in \{0,1\}$, at which the maximum value of the functions is achieved

$$\Phi_\nu^f = \sum_{i=1}^{g_\nu} \sum_{\Re_{jl} \in A_\nu^f} d_{jl}^i \beta_{jl}$$

with the following constraints:

$$\left.\begin{aligned} C_\nu = \sum_{i=1}^{g_\nu} \sum_{\Re_{jl} \in A_\nu^f} d_{jl}^i C_{(jl)i} \le C_{\nu\, addit}; \\ T_i^\nu = T_i^o + \sum_{\Re_{jl} \in A_\nu^f} d_{jl}^i \Delta T_{(jl)i} \le T_{i\, addit}^\nu,\ i = 1, 2, ..., g_\nu \end{aligned}\right\}.$$

This problem can be solved by any of the known methods of integer linear programming.

Once the set $\Gamma = \{G_\nu\}$ of optimal task allocation for the assigned subset $S = \{s_\nu\}$ of system states is found, the set of tasks to be performed in state s^0 and in all states $s_\nu \in S$ is to be found for each node of the system. Also, it is necessary to compute the total consumption required for implementing

the resulting allocation set $\Gamma = \{G_\nu\}$, as well as the average time for handling a declaration in each node and the system performance for each state of the assigned set $S = \{s_\nu\}$.

We shall describe the optimal allocation of tasks determined for each state $s_\nu \in S$ $(\nu = 1, ..., r)$ by using matrix $\left\| A_{jl}^\nu \right\|$, whose lines correspond to tasks $\Re_j$ $(j = 1, ..., L)$, and columns to nodes M_i $(i = 1, ..., n)$, A_{ji}^ν represents some subsets of tasks determined below. As regards the columns corresponding to operable nodes M_i in state s_ν, $A_{ji}^\nu = \Re_{jl} \cup \{\Re_{jl}\}^{\nu i}$, where $\Re_{ji}$ is the task performed in M_i in state s^0, here $\{\Re_{jl}\}^{\nu i}$ is the set $\Re_{jl}$ transferred to M_i in state s_ν.

As for the columns corresponding to failures M_i, $A_{ji}^\nu = \text{Ø}$, let us construct for them the resulting matrix $\left\| A_{ji} \right\|$, where $A_{ji} = \bigcup_{\nu=1}^{r} A_{ji}^\nu$.

The matrix $\left\| A_{ji} \right\|$ defines for each node M_i, $(i = 1, ..., n)$ a set of tasks for whose completion it has to have the necessary resources in order that each of the corresponding system states might achieve the maximum value of its efficiency with the constraints imposed on the node performance and additional consumption. This matrix can be used to calculate the above-mentioned system performance.

Thus, the application of the proposed approaches towards reallocating tasks to a system's hierarchy levels and nodes allows both the embedded system and the whole system of controlling compound elementsand complex processes to remain functional when a failure stems from adverse exposure.

5. The Choice of Computing Systems Capable of Reconfiguring the Underlying Structure of Hardware Components

The choice of computing systems is meant to single out from the options for organising computing systems those which hold the most potential for the future and are capable of reconfiguring the underlying structure in the wake of adverse exposure. It is to be noted that the issue of choosing a computing system occurs is to be solved at the beginning stage of designing. This means that it is solved before the development of functional subsystems of the system for controlling compound elements and complex technological processes has been completed. Therefore some requirements posed to the system intrinsically lack clarity and lucidity. Fuzzy input calls for adequate decision-making methods based on imprecise expert evaluation as regards the tasks relating to input data and result presentation.

Now let us introduce the composite linguistic variable *computing system* and narrow down the task of choosing a computing system to determining the compliance (of the meaning) of this variable with technical requirements posed to it [20]. The constraints imposed by the technical requirements represent clear or unclear relationshipsassigned by a universal set of values of the composite linguistic variable *computing system*. This set consists of the values making up the linguistic variable, each of them corresponding to a particular hardware-software module.

The statement *computing system must have a high-performance processor* is to be interpreted in the following way. The name *computing system* is regarded as a name of the composite linguistic variable whose components are linguistic variables such as *processor*, *random-access memory*, *external storage*and others. The statement in question assigns the value *high-performance processor* to the linguistic variable *processor*.

The value *high-performance processor* is to be interpreted as a name of some fuzzy constraintimposed on the basic variable*processor*, with the meaning of this constraint being defined by its membership function. For example, as pursuant to the value meaning, the linguistic value *high-performance processor* leaves only four processors as to generating computing system variants: Intel Core i5-12600K, Intel Core i9-12900K, AMD Ryzen 5 5600X и AMD Ryzen 9 5900X. In the same

way, one can evaluate other linguistic variables. Now we shall give a formal description of the approach in question.

Suppose the linguistic variable value computing system is assigned through a tuple of the constituent terms X_i, each of them naming a particular functional characteristic of the computing system. The values of the corresponding terms $X_i (i = \overline{1, m})$ of the linguistic variable are taken from the corresponding subsets T_i of the term set of the linguistic variable, whose elements are assigned by way of enumerating all values accepted by the given characteristic of the computing system. With the concept of the linguistic variable employed, requirements are to be interpreted as an equation for allocating the linguistic variable *computing system* of some linguistic value from an allocated term set. In this case, the linguistic variable *computing system* is assigned a value which is the constituent term t representing the tuple of the constituent terms: *computing system* $= t$, where $t =< X_1, X_2, ..., X_m >, t \in T$.

The meaning of the concrete value *t* of the linguistic variable *computing system* is the fuzzy set U_t withthe membership function $v_t = (v_{1t}, v_{2t}, ..., v_{nt})$ allocated to the universal set U which includes all accessible computing systems. The value v_{it} indicates the computing system's compliance j-tn with a set of characteristics formulated by term t. The semantical rule for determining the meaning of each value of the linguistic variable is to be represented as follows:

$$\nu_{jt} = \prod_{i=1}^{m} \mu_{ijt}^{\alpha_i}, \quad \left(j = \overline{1, n}\right), \quad \left(i = \overline{1, m}\right), \quad t \in T, \tag{3},$$

where α_i is the importance factor i-tn of the functional characteristic; μ_{ijt} is the compliance of the computing system j-tn with the functional characteristic i-tn (the degree of its implementation, with $0 \le \mu_{ijt} \le 1$). Values α_i и μ_{ijt} are to be given expert evaluation as regards each functional characteristic and each computing system in question by way of averaging evaluations given by each expert group.

In this case,

$$opt\,(\text{Computing system}) = \max_i v_{jt} \tag{4}.$$

The process of choosing a computing system can be represented by the following algorithm (Fig. 2).

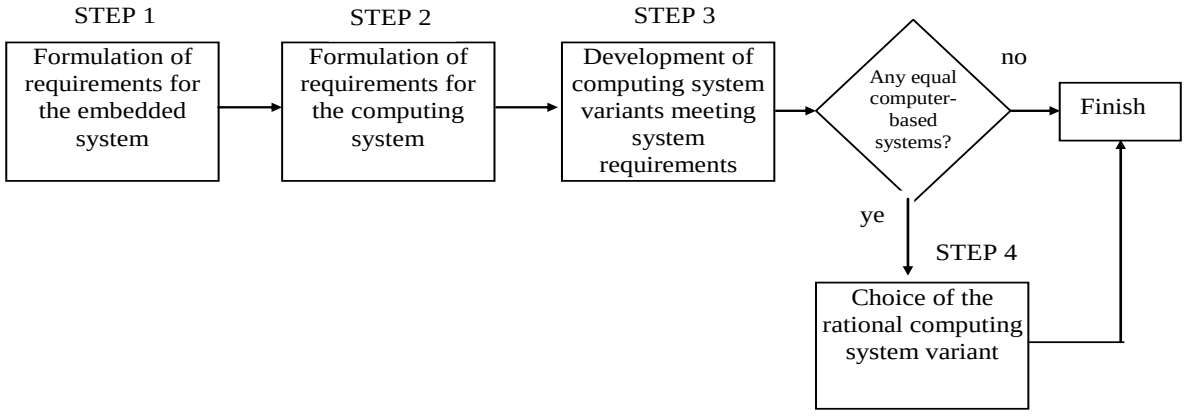

Figure 2: Algorithm of choosing computing tools

Steps 1 and 2, which represent the algorithm for choosing a computing system, map the designing of a set of requirements for an embedded system formulated in the technical requirements for developing this system for the set of valid values of the computing system's output characteristics. The mapping of requirements for an embedded system for the set of valid values of the computing system's output characteristics is implemented via the tasks assigned to the system and represents the main goal as to the development of embedded systems: enhancing the survivability of the system for controlling compound elements and technological processes. These are system requirements posed to

a computing system which include requirements for productivity, reliability, adjustability, costs, proper functionality etc.

As for Step 3, it is about determining the importance factor α_i, compliance μ_{ijt}, priority vector v_t and choosing a rational computing system as pursuant to (4).

If vector v_t has one maximum value, a corresponding computing system will be one that meets the assigned requirements in the best way. Otherwise, when taking Step 4, of all computing systems that have the highest values v_{it}, with local characteristics of functional subsystems taken account of, one is to choose a rational computing system by way of using a group method for handling the superiority of alternatives [21, 22].

As of today, computing systems that have the properties for changing algorithms of control instructions and architectures include microcontrollers (PIC, AVR, MSP430, STM32, Cortex-M, TSP32 and others), programmable logic devices (CPLD, FPGA) and single-board computers Raspberry Pi. A distinctive feature of the computing systems in question is that microcontrollers and single-board computers do not allow the change of internal links between primitive elements in contrast to programmable logic devices where programming provides a basis for administering links between logical elements in order that a necessary architecture may be obtained. As well as that, microcontrollers and Raspberry Pi computers are not able to process data externally because of fixed-block architectures and fixed instruction set.

Thus, it can be concluded that programmable logic devices are computing systems that hold the most potential for the future and can reconfigure the underlying structure of the embedded system hardware components via programming.

6. Reconfiguration of the Embedded System

As is mentioned above, that programmable logic devices are computing systems that hold the most potential for the future and can reconfigure the underlying structure of the embedded system via changing internal links between its logical elements. However, it is to be taken into account that reconfiguration of the system structure requires substantial redundancy as each logical element or IP software module (intellectual property) in programmable logic devices (hereinafter referred to as the module) is to ensure the implementation of any function with the corresponding system signal being supplied [23]. Therefore, when implementing one function φ_i, each module is to have j-fold redundancy (j is the number of modules implementing a particular function. Figure 3 features structure redundancy of the device having j-fold redundancy. As a result, a small part of the general module structure is employed. This redundancy is not always acceptable. It more expedient to employ the primary module structure which is capable of keying out its certain segments in the event of their failure.

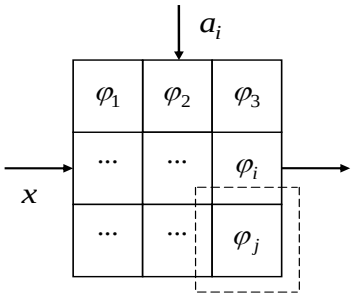

Figure 3: Structure of the redundant device with j – fold redundancy

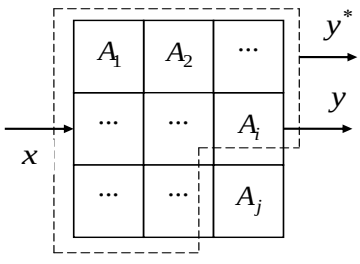

Figure 4: Flowgraph of the non-redundant device with non-uniform modules

Figure 4 features a flowgraph of the non-redundant device with non-uniform modules. If a module containing nodes $A_1 - A_j$ implements function *y*, it will implement the disturbed function y^* in the event of failure. In this case there arises a task of restoring the assigned function *y*. This task can be fulfilled by way of replacing device A_j with a backup device, which in fact is parallel reservation.

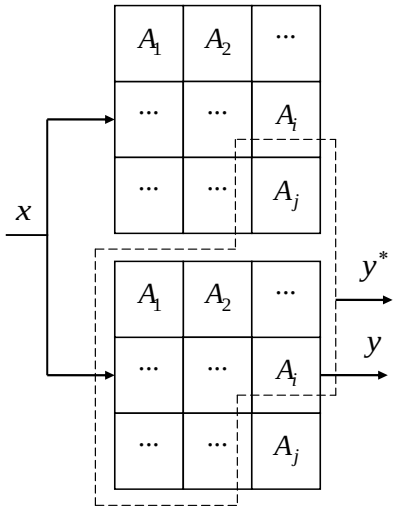

Figure 5: Structure of the redundant device with non-uniform module

Figure 5 shows the structure of the redundant device with non-uniform modules. If a failure in node A_j of module A recurs, it renders the whole module inoperative. To repair a recurring failure, it is necessary to provide n-fold module reservation.

Now let us define the problem in the following way: the keying out of any faulty node can be compensated without intervening in the underlying module structure (in devices with inaccessible structures). One of the options for solving this problem is to finda redundant structure B_i which restores the functional characteristics of module *A* when connected to the inlet of module *A* (with the faulty node A_j switched off) (Fig. 6).

Now we shall consider module *A* which is made up of a set of discrete components a_i. The module implements the function $Y_{n-1} = \varphi_0(X_n, Y_n)$, (5),

where $X(x_1, x_2, ..., x_n)$ is an input word, $Y(y_1, y_2, ..., y_n)$ is an output word and n is time steps.

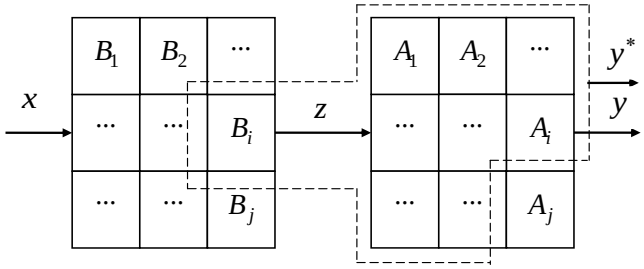

Figure 6: Structure of the restored module $\mathcal{A}$

It is necessary to synthesise some system $S \supseteq A$ which implements function $\varphi_0(X_n, Y_n)$ in the event of failure of any subsystem A_j of system $A(a_i \in A)$ with the assigned decomposition complexity C_j. System A may be represented as matrix M_A of non-overlapping subsystems A_j. The extraction of any subsystem of matrix M_A results in the formation of the error-correcting matrix $M\{A_{0j}\}, A_{0j} = (A_0 \setminus A_i)$, which generates a set of new functions $M\{\varphi_{0j}(X_n, Y_n)\}$.

In order to restore the functional characteristics of the system (implementation of function *Y*), it is necessary to create the restorative matrix $\{M_{B_j}\}$. In this case, each subsystem B_j of matrix $\{M_{B_j}\}$ is to be connected to the inlet of subsystem error-correctingmatrix $M\{\varphi_{0j}(X_n, Y_n)\}$. Normally, all B_i have the structural intersection

$$\bigcap_i B_i = B_1 B_2 \ldots B_j \ ,$$

whose functional characteristics are shared by all $B_1, B_2 ..., B_j$ or by most of them. As well as that, there may be some structures that do not contain intersections. In this case, for every failure and switch-off system the redundant structure B_i is created by way of connecting corresponding input $\{X\}$ and output $\{Z\}$ signals of this module on the basis of the generic unit $\bigcap_i B_i$ (Fig. 7).

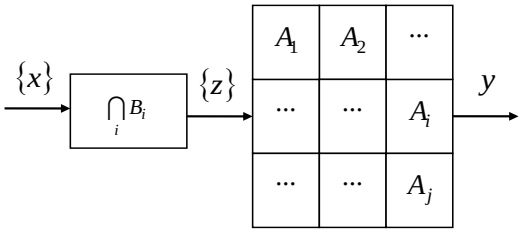

Figure 7: Restoration of module $\mathcal{A}$

The essence of the synthesis of system A_0 which is considered lies in defining rule Ψ for describing subsystem B_j of matrix $M\{B_j\}$.when extracting any subsystem A_j of matrix M_A.

Thus, rule Ψ is to induce the restorative subsystems B_j on A_{0j}:

$$B_j = \Psi A_{0j}, \tag{6}$$

as well as the restorative matrix $M\{B_j\}$ of the error-correcting submatrix:

$$M\{B_j\} = \Psi\{M_{Aj}\}. \tag{7}$$

Now let us consider the problem for formalizing the induction rule Ψ. Subsystem A_{0j} of the error-correcting matrix $M\{A_{0j}\}$ forms a segment of system A_0 and is described by function (6). To convert the error-correcting function φ_{0j} into function φ_0, one needs a new subsystem B_j whose composition coupled with subsystem A_{0j} forms the system

$$A_0 = \{B_j \Theta_j A_{oj}\} \text{ in ratio } \theta_j. \tag{8}$$

Ratio θ_j is assigned by the connector for systems A_{0j} и B_j or by the relation of equivalence between the output subset $Z \subseteq X_{oj}$ of subsystem B_j and subset X_{oj} of subsystem A_{0j}. The connector assigns the functional relationship between indices *i* and outlets of subsystem B_j with indices k for outlets of subsystem A_{0j}:

$$i = \Theta_j(\alpha, \beta, k), \tag{9}$$

where α is a parameter characterising the number of inlets A_{0j} employed by subsystem B_j; β is a parameter assigning the relationship between inlets and outlets according to α.

As stated above, the outlet $Z = B_j X_n$ of the restorative subsystem B_j implements the function $Z = f_j(X_n, Y_n)$.

In order to determine B_j or f_j according to (7), we obtain theexpression

$$\varphi_{0j}(X, Y, Z) = \varphi_0(X_n, Y_n),$$

which functionally reflects the right part of ratio (8).

Then the equation

$$\varphi_{0j}(\Theta, X, Y, Z) = \varphi_0(X_n, Y_n) \tag{10}$$

will determine Z, i.e. it will describe the subsystem

$$Z = \Psi(X_n, Y_n, \Theta, j). \quad (11)$$

Thus, by means ratio (11) function Ψ assigns the rule for inducing the restorative subsystems B_j for all $j \in J$.

Hereinafter the paper propose that that the suggested approach to designing embedded adaptive systems should be implemented with CAD software produced by Intel PSG (Altera) for designing the SOPC (System-On-a-Programmable-Chip) on the basis of programmable logic devices and NIOS II processor core [24, 25]. This core contains enclosing packages Quartus II (PLD), SOPC Builder (a configurable processor core), NIOS II IDE (software) and other extensions (DSP Builder, C-to-Hardware Compiler and others).The equivalent powerful instrumental complexes are produced by such companies as Xilinx, Cypress (PSoC Designer), Actel (Smart Design) and some others.

To sum up what is stated above, we may say that unlike multiple reservationof system elements, the proposed reconfiguration of the embedded system not only allows the compensation of recurring failures and fault conditions, but also provides the necessary levels of survivability as regards the system for controlling compound objects and complex processes.

7. Conclusion

The article proposes a methodical approach to designing secure adaptive embedded systems based on the ability of the microprocessor-based system to restore the system's proper functionality affected by adverse exposure by implementing automatic reconfiguration of the underlying structure, the procedure being based on self-checking results. What underlies this approach is the completion of systematically-related tasks of synthesising key elements and parts of distributed microprocessor control systems, which includes the determination of the number of a system's hierarchy levels and nodes, allocation (reallocation) of tasks to a system's hierarchy levels and nodes, as well as the choice of computing systems capable of reconfiguring underlying structures of hardware components. All this makes it possible to enhance survivability of the embedded system affected by adverse exposure, as well as the survivability of the whole system for controlling compound elements and technological processes.

8. References

[1] A. Crespo, P. Albertos, J. Simó, Embedded control systems: from design to implementation, Ifac Proceedings Volumes, Volume 40, Issue 1, 2007, Pages 25-32, ISSN: 1474-6670, ISBN: 9783902661210, https://doi.org/10.3182/20070213-3-CU-2913.00006.

[2] H. De Man, System-on-Chip design: impact on education and research, in *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 11 – 19, July - Sept. 1999, doi: 10.1109/54.785820.

[3] Chen, Yen-Kuang & Kung, Sun-Yuan. (2005). Trends and challenges with System-on-Chip technology for multimedia system design. 2005. 4 pp.. 10.1109/EITC.2005.1544365.

[4] Ma L, Xia F, Peng Z. Integrated Design and Implementation of Embedded Control Systems with Scilab. Sensors (Basel). 2008 Sep 5;8(9):5501-5515. doi: 10.3390/s8095501. PMID: 27873827; PMCID: PMC3705517.

[5] Khan FH, Pasha MA, Masud S. Advancements in Microprocessor Architecture for Ubiquitous AI-An Overview on History, Evolution, and Upcoming Challenges in AI Implementation. Micromachines (Basel). 2021 Jun 6;12(6):665. doi: 10.3390/mi12060665. PMID: 34204065; PMCID: PMC8227299.

[6] Furber S. Microprocessors: the engines of the digital age. Proc Math Phys Eng Sci. 2017 Mar;473(2199):20160893. doi: 10.1098/rspa.2016.0893. Epub 2017 Mar 15. Erratum in: Proc Math Phys Eng Sci. 2017 May;473(2201):20170304. PMID: 28413353; PMCID: PMC5378251.

[7] Wehrmeister, Marco & Becker, Leandro & Pereira, Carlos. (2005). An approach for designing real-time embedded systems from RT-UML specifications. 16. 1060-1060.

[8] P. Khusainov, S. Toliupa, V. Bakanov and S. Shtanenko, "Substantial formulation of the task of improving the information model of decision-making in the prompt (crisis) response to cyber

incidents," *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 287-290, doi: 10.1109/TCSET55632.2022.9766994.
[9] Horbenko, V., Korshetc, O., Korolyuk, N., &Nevgad, S. (2020). Method for designed and adaptation of complex organization and technical systems. Journal of Scientific Papers «Social Development and Security», 10(1), 47-55. https://doi.org/10.33445/sds.2020.10.1.6
[10] Hnatiienko, H., Kiktev, N., Babenko, T., Desiatko, A., Myrutenko, L. Prioritizing Cybersecurity Measures with Decision Support Methods Using Incomplete Data. *CEUR Workshop Proceedings*, 2021, 3241, pp. 169-180
[11] Dodonov A.G. and Lande, D.W. (2011), Information Systems Survivability, Nauk. dumka, Kyiv, 256 p.
[12] Tkachov, V., Kovalenko, A., Kuchuk, H., & Ni, I. (2021). Method of ensuring the survivability of highly mobile computer networks. Advanced Information Systems, 5(2), 159-165. https://doi.org/10.20998/2522-9052.2021.2.24
[13] M.A. Aguirre, J.N. Tombs, V. Baena-Lecuyer, J.L. Mora, J.M. Carrasco, A. Torralba, L.G. Franquelo, Microprocessor and FPGA interfaces for in-system co-debugging in field programmable hybrid systems, Microprocessors and Microsystems, Volume 29, Issues 2–3, 2005, Pages 75-85, ISSN 0141-9331, https://doi.org/10.1016/j.micpro.2004.06.009.
[14] Shtanenko, S., Samokhvalov, Y., Toliupa, S., Silko, O. (2023). The Approach to Assessment of Technical Condition of Microprocessor Systems that Are Implemented on Integrated Circuits with a Programmable Structure. In: Klymash, M., Luntovskyy, A., Beshley, M., Melnyk, I., Schill, A. (eds) Emerging Networking in the Digital Transformation Age. TCSET 2022. Lecture Notes in Electrical Engineering, vol 965. Springer, Cham. https://doi.org/10.1007/978-3-031-24963-1_28
[15] D.L. Parnas, P.C. Clements and D.M. Weiss, "The Modular Structure of Complex Systems," in *IEEE Transactions on Software Engineering*, vol. SE-11, no. 3, pp. 259-266, March 1985, doi: 10.1109/TSE.1985.232209.
[16] Kimla, R.; Czerwinski, R. A Methodical Approach to Functional Exploratory Testing for Embedded Systems. Appl. Sci. 2022, 12, 10016. https://doi.org/10.3390/app121910016
[17] Sergienko, I.V., et al. "Solving the conditional optimization problem for a fractional linear objective function on a set of arrangements by the branch and bound method." Cybernetics and Systems Analysis, vol. 48, no. 6, Nov. 2012, pp. 832.
[18] Zilla Sinuany-Stern, Foundations of operations research: From linear programming to data envelopment analysis, European Journal of Operational Research, 2022, ISSN 0377-2217, https://doi.org/10.1016/j.ejor.2022.10.046.
[19] Jian-Yu, Chen & Tai, Kuo-Cheng & Chen, Guo-Chin. (2017). Application of Programmable Logic Controller to Build-up an Intelligent Industry 4.0 Platform. Procedia CIRP. 63. 150-155. 10.1016/j.procir.2017.03.116.
[20] D.P. Atherton, The Role of CAD in Education and Research, IFAC Proceedings Volumes, Volume 14, Issue 2, 1981, Pages 3395-3400, ISSN 1474-6670, https://doi.org/10.1016/S1474-6670(17)63975-2.
[21] Samokhvalov, Y.Y. Group Accounting of Relative Alternative Superiority in Decision-Making Problems. Cybernetics and Systems Analysis 39, 897-900 (2003). https://doi.org/10.1023/B:CASA.0000020231.09571.33
[22] Thomas L. Saaty. Decision making with the analytic hierarchy process, International Journal of Services Sciences, Vol. 1, No. 1, 2008, pp.83-98. ISSN: 1753-1446, ISSN: 1753-1454.
[23] V.Ye. Obukhov and V.V. Pavlov, Synthesis of Redundant Discrete Systems with Structure Reconfiguration, Naukova Dumka, Kiev (1979), 156 p.
[24] Y. Samokhvalov, S. Toliupa, S. Buchyk and S. Shtanenko, Design of Robotic Systems in the Basis of CAD Intel Quartus Prime, *2021 IEEE 3rd International Conference on Advanced Trends in Information Theory (ATIT)*, 2021, pp. 179-183, doi: 10.1109/ATIT54053.2021.9678857.
[25] Z. Zou, T. Long, L. Wang, B. Zou and J. Chen, "Design of digital system development platform based on FPGA," *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 2021, pp. 801-804, doi: 10.1109/IAEAC50856.2021.9391086.