Problems of Abstract Representation of Embedded Systems at High-level Stages Design

Alexey Platunov[1,2] and Pavel Kustarev[1,2]

[1] Saint-Petersburg State University of Information Technologies Mechanics and Optics (SPbSU ITMO), Computer Science Department
49 Kronverksky avenue, Saint-Petersburg, 197101, Russia
[2] LMT Ltd., 16 Birzhevaya Line, Saint-Petersburg, 199034, Russia

Abstract. Conditions and nature of embedded systems design demand today serious revision of techniques and technologies of their creation. To a considerable degree artificial dividing the computer system into hardware (HW) and software (SW) components and, moreover, allocation of the SW industry in a separate sphere only aggravates the situation. One of the ways of this problem decision can be usage of architectural abstractions system directed on the creation of an integrated design space of the computational process organization in which for the computer system a representation of traditional HW and SW implementations becomes unified, there are obviously functional and nonfunctional aspects of designing and the tool component is integrated.

1 Introduction

Impetuous growing requirements for embedded systems (ES) forces developers to improve actively design methods and tools. The ES share with a complex internal organization actively grows that becomes apparent in the multiprocessor heterogeneous architecture, the distributed character of calculations, a wide range of computational resources potentially accessible to the developer. The significant part of the most complex ES has the distributed organization and refers to a category of network embedded control systems (NECS).

It is necessary to consider as the key feature of the ES creation the necessity of the complex approach to the designing, covering practically all levels of the computer system (CS) organization. The developer really faces with the necessity to analyze the large number of various potentially-possible variants of the architectural organization both a target system created, and a project infrastructure as a whole (it is a matter, first of all, of design technologies and tools). An experience of development of microprocessor systems convinces of the necessity of joint (parallel) designing HW and SW parts of the system that in practice within the limits of traditional technologies is realized extremely seldom.

From a space of the limited set of canonical structures and circuit decisions designers can today already move in a space of practically free architectural design.

Platunov A. and Kustarev P. (2009).
Problems of Abstract Representation of Embedded Systems at High-level Stages Design.
In *Proceedings of the International Workshop on Networked embedded and control system technologies: European and Russian R&D cooperation*, pages 100-107

Change of the situation on a background of growing needs in ES and requirements to them demands overcoming crisis of designing, which becomes apparent in insufficient speed of creation and bad quality of a product. The disturbing state in the field of ES design is marked by all leading experts of this sector of informatics and computer engineering [6, 13, 12].

Top priority importance in overcoming the crisis has the further development of methods and means of high-level (architectural) ES design. Still there are opened questions of creation effective CAD of complex (through all stages) design of ES.

2 Problems of Embedded Systems Design

Custom Microprocessor Systems Design. The hands-on experience of ES design bases on the fine-tuned technological methods and tools and consists in a choice of one of canonical computational platforms on the basis of which due to a program superstructure the applied task is solved. Other variant is also applied: computer platform is being chosen and along with upwards completion updating downwards modification is carried out. Such way is used less often because of high labour-intensiveness.

The first problem, which ES developers face with, consists in the following:

- existing programming languages assume the description of a task for idealized virtual (language) machine;
- the compiler maps this language machine on the real machine, bringing the certain restrictions and leaving out many important technical features of the executive machine realization;
- there is no uniform system of the language machine description, the compiler and the executive machine.

The second problem - the large number of tasks, especially in the field of control systems of physical objects, which badly keep within the scheme of implementation on the basis of canonical computer platform with a program realized superstructure. Such tasks demand specialized computer platform, for example, with a high degree of parallelism and specialization of operational blocks, and in this case efficiency of the traditional scheme of designing can appear to be inadmissibly low.

The third problem is connected with constantly growing volume of necessary ES designing. Particularly the disunity of descriptions noted above, prevents from a reuse of the developed components (hardware blocks, programs, realizations of algorithms).

State and Perspectives of High-level ES Design. The basic directions of researches for overcoming the listed problems lay in the field of perspective techniques of high-level (system) ES design (HLD). For these directions the following design levels are determined by specialists:

- *mission (or operational, or specification) level* - development of the system behavior script in the form of «the executed system specification/model»; modeling of an external system environment.

architecture (or macroarchitecture, or performance) level - formation of system architecture, irrespectively to a way of realization: the analysis of architecture for conformity with functional and nonfunctional requirements/restrictions.

– microarchitecture (or functional) level or (especially with reference to development digital VLSI and SoC) - electronic system level (ESL) - a choice of a way of realization of architectural model components is carried out, algorithms, interfaces are being developed, specifications and the environment of verification are prepared for the implementation stage of the system.

General problems of accessible methodologies, tools and environments of designing (frameworks), inducing to the certain choice of directions and development of research works in the field of system designing for ES are fixed below:

– particular (especially nonfunctional) aspects of designing are considered separately within specialized entire system models: power consumption, reliability, information protection, etc. It is necessary to develop such structure and the form of architectural components description, which obviously will offer aspect estimations (weights) that are reflected in the concept of the architectural aggregate (AA), which is explained further in this article.

– explicit priority of the functional aspect at an architectural level is held true. The nonfunctional aspects either play a role of auxiliary (secondary) criteria of the project, or many of them are not taken into consideration at all. But in general case for ES functionality not always appears the most higher-priority requirement. These issues should be taken into consideration in the tool development process of the system architectural description.

– a set of variants of the microarchitecture realization is limited by HW/SW of the runtime phase. Search of project decisions at a variation of Design/Run Time ratio, consideration of various levels of architecture virtualization, integration of a tool component in most cases within the limits of the system designing is not carried out.

The decision of the specified problems is probable through integral perception of target system space project, and project infrastructure. This assumes integrated space creation of ES design in terms of computational abstractions (computational mechanisms, functional converters, virtual machines, architectural aggregates), and, accordingly, a new model of the design process. The necessary conditions of the creation of such space are:

– unification of hardware/software treatments;

– *ES representation within the limits of possibly a lot of design steps (from the beginning) without dividing on HW/SW realization;*

– using «aspect technologies», which allows to track explicitly transformations of key ES components in the design process (functionality, tool, reliability aspect, and others), and to consider during designing factors, which explicitly are not present in specification requirement, but seriously influence on results (tool aspect, the possibilities of the team, accessible technologies, taking into account a groundwork and team interests, and others).

3 Integrated Design Space of Embedded Systems

Computational Process Organization. The concept of computational process in ES occupies the central position. We consider, that *process of the computer system design* as the object solving a specific target of the user, it is reasonable to consider as *the organization of a computational process in space and time in the restrictions specified by the technical requirements*.

Tools of the realization of computational process are:

- design concepts (platform-oriented, model-oriented, actor-oriented, and others);
- computational models (models of computation - MoC);
- virtual machines;
- platforms (computational, tool, protocols of interaction, and so on);
- mechanisms (as technical decisions from various areas-aspects in an abstract representation);
- element base (as a set of mechanisms realizations).

The carrier of a run-time component of computational process is ES architecture - ES representation with a minimum refinement level, showing all its unique technical decisions in relation to the computational process organization. Depending on the one to whom given ES architectural representation is addressed, the set of "known" elements can essentially differs.

Embedded Systems Programmability. Modern ES is accepted to name SW-dominated systems, which are created on the basis of «highly programmable platforms» [1, 13] that is caused by inseparable connection of a computational platform and an applied level.

The most part of modern element base is programmed or configured. In whole it expands the developer potentialities, simultaneously significantly increasing risk of an error and a labor intensiveness of low-level design. The range of system organization levels, which the developer is forced to cover for the qualitative hardware control, is extremely wide. Attempts to minimize the low-level design volume in the ES field didn't have any success, as imposing on the developer in this case the limited number of configuration patterns significantly worsens design quality.

Models of Computations. There are various MoC definitions, which can be divided into two directions. One of them [5, 7] is based on the desire to be limited by use of simple (by «a principle of action») models of one (abstract) level (model of data-flows, FSM models, models of synchronous processes, and others). Another direction assumes a formalization of computer devices representation as virtual machines irrespective of their complexity and a belonging to the computational hierarchy level.

In a context of integrated design space MoC is interpreted as *mathematical model showing to the user computational possibilities and rules of the computer device use*.

Virtual Machines. *The virtual machine is understood as the computer device, for which rules of behavior are certain* (for example, command system, conditions of command and data input, reception of the result, a rule of process synchronization), *allowing unambiguously to describe algorithm of the task decision*. The virtual

machine description shows only external properties of the computer device and the rule of its use, not concerning of its organization. A principle of extraction of virtual machines is powerful technological instrument, allowing to structure the design process, to provide portability and a reuse of groundwork, to get scale project decisions.

Computational Platforms, Mechanisms, Element Base. A *computational platform* formalism takes the important place in computer engineering. In the context of the integrated design space the *platform can be considered as unity of "external" and "internal" representation [any] of a functional-completed and a functional-significant object in ES structure.*

The computational mechanism as an abstraction solving a particular task plays a role of the «building block» in the computational process organization. The range of functional complexity of computational mechanisms is limited only by developer imagination that creates serious problems for formalization of their representation and operations on them [8, 10].

The element base is considered as a set of mechanisms realizations. It is necessary to expand this concept by inclusion in it a set of objects (physical, informational, technological, and others), which belong to all aspects of the design space.

Design/Run Time Space is effective means of ES analysis. In many respects a template type of nowadays ES designing uses in the majority of cases empirical decisions. Chains of computational process realization include such elements, as compilers, interpreters, virtual machines, hardware programmed processors, special functional hardware blocks, and many other things. The developer, often not consciously, distributes elements of the computational process inside of tool (design-time) and performing (run-time) phases of the project. The analysis and the realized choice of decisions in both phases of ES existence allow to increase significantly quality of designing.

4 Aspect Technology Bases of Embedded Systems Design

Today Aspect-Oriented Software Development was formed as an independent direction in programming [14]. Abstractions of aspect representations have actively started to be used in ES design [2], however, this direction is in a formation stage.
The authors demonstrate below the use of the aspect ideology in a combination with the system of architectural abstractions in ES design.

On the top level of aspect technology the developer deals *with project architecture* and *product architecture* (ES designed), which contain all components respectively of design process and product created. We shall name such components as *aspects of designing* (or simply aspects).

Let's define *ES architecture* as a *set of ES conceptual aspects of some refinement level, completely representing designed system for the given level of consideration*. In the list of conceptual ES aspects, besides traditional structural and functional elements reliable, constructive-technological, power assumption, conditions of maintenance,

tools, reuse, organizational-economical, documental, and others elements are included.

Certain elements of architectural representation we shall name *aggregates*. Thus, *the architectural aggregate* (AA) represents a base element of the system design process.

AA, describing not all aspects, is named an *abstract.* Complete AA (describing all project aspects), which cannot be realized now by a concrete team in an element base accessible to it, we name *virtual*. It mast be said, that AA virtuality can be determined not only by objective limitation of the element base, but also by subjective restrictions, such as requirements specification, predilections, and interests of the team. Accordingly, complete AA for which there are all necessary conditions of realization, we shall consider realizable.

The system model expressed in AA terms, we shall name *architectural model* or *A-model* of the system. A-model can be abstract, virtual and realizable.

Abstract A-model contains at least one abstract AA. The model essentially unrealizable and demands further completion. But such models have their way of application, namely, such models should be considered as *platforms*. Abstract A-models, passed in the category of the standard platforms is possible to consider standard interfaces, protocols, computational cores, operational systems, and many others. Some abstract A-models are convenient as a reusable platform within the team bounds which can fix the certain successful decision, designate a direction of development, to provide continuity, having determined abstract A-model of the system and developing it in variety of concrete applications.

Virtual A-model. In such model abstract AA's are not present, but there is at least one virtual AA. Such model cannot be realized because of the virtual AA's, but it is already completely determined and, in principle, can be realized if to expand accessible element base. For successful realization of the model it is necessary to get rid of virtual AA's. It is achieved in two ways: by changing the model or changing external factors. In the first case developers change model (carry out design process) until virtual AA's don't remain in its structure. In the second case the model remains constant, and external factors vary (requirements specification accurate definition along the lines of restrictions changes; team education; transition to other computational cores; use of new technologies for team).

Realizable A-model consists only of realizable AA. Such model can be realized by the team in an element base and a technology accessible to it.

During designing of target system at the initial stages there is a concrete definition and verification of A-model. The result of that becomes "golden" model. *The "golden" model is the verified and fixed architectural model of the system, which is not limiting ways of the implementation.* An important task of the "golden" model becomes a creation of initial specifications for developers who are engaged in final realization of components and units of the system. Being A-model the "golden" model specifies the whole set of aspects of ES being realized.

An architectural platform acts as the reuse tool of the conceptual decisions within the bounds of aspect ideology. *The* architectural platform should be considered as association of following elements of design:

- aspect space of design process (the list of design aspects);
- model (models) of computation (MoC, behavioral aspect);

- external factors setting admissible different aspects ratio (design criterions);
- list of the fixed patterns of reuse;
- element base.

Architectural platform reconfigurability is defined as an ability to change MoC "embodied" at realization. The superstructure on the architectural platform, created with the purpose to change MoC, is named *as an operational environment*.

Tool aspect of ES designing on importance ranks with functionality [11] that assumes its taking into consideration not only "outside" on a course of the project, but also its inclusion at an early stage in the initial specification of the project/system.

5 Conclusions

Formalization and automatization of design processes always demands enormous efforts. Field of custom ES design - an example, where such efforts are necessary to make. A serious obstacle on the way of design technologies implementation on the basis of computational architecture abstractions is the problem of specialists education in the field of computer engineering and informatics possessing HLD-ideology is represented.

It is necessary to recognize, that a large number of ES developers teams nowadays insufficiently high estimate a role and labor-intensiveness of high-level design stages. They do not have adequate technical language for communication at this level, mutual understanding and a correctness of differentiation of developers' responsibility zones suffers, and there is no due integration.

The understanding of responsibility zones by HLD-specialists in a context of offered integrated ES design space determines of the design efficiency. The scope of ES architectural abstraction should extend by means of system engineering from the bottom border of computer system organization up to the top border of the project, which the requirements specification containing computational functionality of the project in aggregate with nonfunctional components is. As the bottom border the computational mechanisms distinctly showing the organization of computational process in analyzed structure (today it is RTL level) acts.

The system of architectural abstraction allows creating effective communication language in the field of ES design, qualitatively to advance technologies and tools, dramatically to improve the engineering documentation. For this purpose the coordinated efforts of operating developers of computer engineering and specialists of universities are necessary.

In St. Petersburg State University ITMO [15] within the framework of direction ES/NECS works on development aspect technology of design are carried out. Objective-event models of computations OEMoC, DOMoC [9, 11] have been offered and developed, researches in the field of virtual machines with a dynamic set of instructions [4] are carried out, models of actualization of ES computational process [3] are developed and investigated.

References

1. Ferrari, A., Sangiovanni-Vincentelli, A.: System Design: Traditional Concepts and New Paradigms. Proceedings of the 1999 Int. Conf. On Comp. Des., Austin (1999).
2. Jackson, E., Sztipanovits, J.: Using Separation of Concerns for Embedded Systems Design. Conference paper EMSOFT'05, Jersey City, New Jersey, USA (2005).
3. Kovyazin, R.: Information-control systems design as task target actualization. Scientific-technical bulletin SPbSU ITMO, Vol.45, Saint-Petersburg, Russia (2007), 79-85 (in Russian).
4. Kovyazin, R., Postnikov, N.: Local regulators creation on the basis of virtual machine with a dynamic instruction set. Scientific-technical bulletin SPbSU ITMO, Vol.32, Saint-Petersburg, Russia (2006), 55-62 (in Russian).
5. Lavagno, L., Sangiovanni-Vincentelli, A., Sentovich, E.: Models of Computation for Embedded System Design. - 1998 NATO ASI Proceedings on System Synthesis, Il Ciocco (1998), 57.
6. Lee, E.: Embedded Software. Technical Memorandum UCB/ERL M01/26. University of California, Berkeley (2001).
7. Lee, E., Neuendorffer, S., Wirthlin, M.: Actor-oriented design of embedded hardware and software systems. Journal of Circuits, Systems, and Computers, Vol. 12, No. 3 (2003) 231-260.
8. Lukichev, A.: Computational mechanisms as the tool of embedded systems design. Scientific-technical bulletin SPbSU ITMO, Vol.45, Saint-Petersburg, Russia (2007), 58-64 (in Russian).
9. Lukichev, A.: Denotative-objective model of computation for the embedded systems, Author's abstract of dissertation. SPbSU ITMO, Saint-Petersburg, Russia (2008) (in Russian).
10. Platunov, A., Postnikov, N.: Formalization perspectives of embedded systems design methods. Electronic components, No.1, Moscow, Russia (2005), 24-29 (in Russian).
11. Postnikov, N.: Behavioral and tool aspects of embedded computer systems design, Author's abstract of dissertation. SPbSU ITMO, Saint-Petersburg, Russia (2004) (in Russian).
12. Sangiovanni-Vincentelli, A.: "Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design". Proceedings of the IEEE, 95(3) (2007) 467-506.
13. Sangiovanni-Vincentelli, A., Martin, G.: A Vision for Embedded Software. Proceedings of CASES 2001. Atlanta, Georgia, USA (2001).
14. Aspect-Oriented Software Development conference WEB-page [online]. Available: http://www.aosd.net.
15. SPbSU ITMO Computer science department, embedded computer system sector WEB-page[online]. Available: http://embedded.ifmo.ru (in Russian).