Design and Modeling of a Dataflow Avionics System

Prof. Peter B. Panfilov and Sergey M. Salibekyan
National Research University – The Higher School of Ecomonics (HSE), Moscow, Russia

Abstract

The major challenges faced by designers of modern avionics systems (ASs) based on advanced parallel and distributed computing technologies (including multicore and many-core processor architectures for embedded systems) are discussed. While providing many benefits, those technologies bring along design problems, such as difficulty in providing one-to-one correspondence between the AS simulation model and actual AS characteristics, great waste of time and resources on the AS modeling and simulation in the AS life cycle (while the models are in most of cases abandoned after the AS completion), and the necessity of conducting extensive testing of complex aerospace hardware and software system in testbeds with actual AS equipment (which itself is heterogeneous by its nature). In this paper we present new strategies aiming to cope with those challenges at different design levels as well as some recent research results that may lead to the solution of the problems, taking into account their impact on area and system performance.

A recently proposed mitigation technique is reviewed that suggests integration of actual AS and its simulation model in a single system so that modeling and simulation time is used not only for system performance evaluation but for actual AS development and test as well. The related research work is focused on the development of a novel distributed computer architecture of the AS (hardware and software system) which is called the object-attribute (OA) architecture. The OA-architecture follows the dataflow execution paradigm and tackles the problems with the novel organization principles of computing process and software mobility (portability among hardware platforms) that allow for complete one-to-one correspondence between actual AS and its simulation model and for the reusability of simulation model features in actual AS.

First, the main principles of the OA-architecture of distributed dataflow computer system are introduced with two core elements in the foundation of the architecture, namely, the information pair (IP) and the functional unit (FU). The *information pair* (attributed data) is a set of a data (or references to data) and a tag/attribute (a unique identifier) of a description of the (computational) load. The *functional unit* is virtual (implemented in software) or real (hardware) data processing unit. The FU can be considered as a set of a context (a set of internal registers storing intermediate data for computation) and an algorithm of the FU performance (realized in hardware or in software). Active elements of the computing process in the OA-architecture are FUs. The performance control of FU is done with milli-commands (mOps) which are essentially identifiers (tags) of data transmitted in the computing system. Unlike the von-Neumann (controlflow) architecture of a computer system, the commands for the FU in the OA-architecture are not external ones, and a decision on the beginning of computation is made by the FU itself based on arriving data. An act of information receipt in the FU is described with the help of following four-tuple (an algorithm, a context, a mOp, a reference to data), where the mOp and the reference to data constitute the IP. The IPs are transmitted via data-attribute bus (DABus) that connects all FUs of the computer system. The FU is activated upon the receipt of all needed data for the operation which corresponds to the dataflow execution paradigm in computer systems.

Then we discuss some important features of the OA-architecture as it applies to the design of distributed AS. It is shown that application OA-computer system can be equally effectively realized in hardware or in software. Software FUs are realized with the help of

context which is described as a data structure or a record (in high-level programming language), and an algorithm of FU functional logic realization – in the form of a procedure with standard interface. Millicommand system of the OA-architecture design allows for easy hardware/software realization of n-operand operations (in contrast to the only zero-operand, single-operand or two-operand commands used in "traditional" von-Neumann-like computer architectures). OA-architecture (like an object-oriented programming or OOP) is capable of data and software code abstraction where the capsule (a set of IPs) can be used for description of an object, and the use of references to different information constructions as loads in IPs extends abstraction capabilities even more. Mechanisms of capsules and references can be used to form a graph of abstractions or abstractions tree which describes objects of arbitrary complexity. Abstraction property facilitates significantly the design and programming, debugging and test of an application computer system. Mechanism of abstractions tree allows for easy implementation of a distributed AS: for this the FUs and the capsules are distributed among different computing nodes which are interconnected to a communication links that provide information capsule transfer from DABus of the one node to the DABus of the other node.

The distributed AS in OA-basis consists of two major components: 1) the OA-platform (software that realizes FUs of the system) and 2) image of the OA-system (the description of system operation algorithm as a set of IP exchanges among FUs). Only first part of the OA-system is hardware dependent (and forms a middleware of the OA-architecture), while the OA-image of the system is hardware independent (the unified interface of the FU hides all architectural peculiarities of the computing node). Thus the OA-system's "portation" to the other hardware platform would only require the OA-platform re-programming in programming language for the specific computer hardware while the OA-image of the system (which only works with the FUs) would run seamlessly on the new hardware platform without the need of reprogramming.

The OA-system portability in combination with the IP-mechanism allows for autonomous (offline) debugging of the AS. Thus due to the portability feature, the OA-image can run both on actual (heterogeneous) computer system of the actual AS and on a commodity PC or workstation. The IP-mechanism permits easy substitution of the source of input information, that represents the stream of IPs arriving at DABus of the OA-system, while the FU process incoming data based on tag (attribute) attached to the data. For example, in an actual AS, the sensors that provide data of a control object parameters attach corresponding attributes to that data and then send IPs to the DABus. After the completion of an OA-image debugging evaluation and test, an actual AS can be realized by simply downloading OA-image of a system to actual computing nodes that execute the OA-platform. Then FUs connected to the DABus receive data stream and start abstraction synthesis and control signal generation process or an output data stream generation process.

Finally, an application of the OA-approach to the design and simulation of an avionics system of a modern aircraft is demonstrated. Major steps include the creation of an OA-image of the flight control system and an aircraft simulation model. Then the OA-image of the onboard AS is debugged and tested offline (on a commodity PC or workstation without the use of actual AS equipment) in the OA-programming/simulation environment. At this stage, the OA-image of the onboard AS exchange IPs (sensor data and aircraft data) with the flight simulator of an aircraft executing simulated flight of the modeled mission. Installation of the debugged and tested OA-image of the AS into onboard computer system consists of the OA-platform "tailoring" into microprocessors of the onboard computer system, and the OA-image download (under control of OA-platform) to computing nodes of the distributed AS.