Chapter 5
Model-Based Techniques and Tools for Programming Embedded Multicore Platforms

Konstantin Nedovodeev
St. Petersburg State University of Aerospace Instrumentation, Russia

Boris Sedov
St. Petersburg State University of Aerospace Instrumentation, Russia

Yuriy Sheynin
St. Petersburg State University of Aerospace Instrumentation, Russia

Vera Ivanova
St. Petersburg State University of Aerospace Instrumentation, Russia

Alexey Syschikov
St. Petersburg State University of Aerospace Instrumentation, Russia

Sergey Pakharev
St. Petersburg State University of Aerospace Instrumentation, Russia

ABSTRACT

The chapter considers VIPE development environment with the main emphasis on its formal ground. The detailed description of a formal VIPE model of computation (MoC) and the semantics of language constructs let the reader reason about the behavior of the constructs in question. The authors propose a rigorous description of program transformations applied to the program while it is compiled. The program after all the transformations is a correct one from the view of the host MoC. Its behavior meets the programmer's expectations even when it includes fragments, which belong to a guest MoC. Techniques for translation of the guest MoC (OpenVX) constructs into the host MoC (VIPE) constructs were proposed. The approach described here leads to the end program that is fully conformant to the host MoC. In addition, the whole toolset is at the programmer's disposal, namely visual editor, compiler, runtime, and analysis tools. They stay applicable to the program, some parts of which are now guest MoC constructs.

DOI: 10.4018/978-1-7998-1974-5.ch005

INTRODUCTION

Development of parallel programs, which should be efficiently executed on heterogeneous manycore platforms, is a hard challenge for embedded system developers. Such platforms are targeted to the domains like ADAS, cryptography, video surveillance, aerospace etc. Even today, there are many heterogeneous manycore platforms on the market from NVidia, Qualcomm, Imagination, AllWinner, Samsung, Mediatek, ELVEES and other vendors. Tomorrow most of embedded systems will be heterogeneous (Joshi, 2016).

Nowadays developer teams create complex computing embedded systems (Evans, 2004); (Balandin & Gillet, 2010). Teams often include many experts from various domains. For an efficient problem solving such teams desperately need a common language for their project. According to many researchers, a visual graph notation is a natural representation of an operations sequence (Mellor, Balcer, & Jacoboson, 2002). Each member of a developer team explicitly or implicitly uses some kind of a graphical flow chart in his project design. It is better to have a single "big picture" of the whole project, to which all the members have simultaneous access.

In addition, developers face extremely complex and contradictory requirements; for example, they need to produce a high-quality embedded solution for some task within a tight time frame. Meanwhile, the volume of code vastly increases. When companies describe existing situation they compare it to the shift from writing programs solely in an assembly language to writing them in a high-level language. It is more comprehensible and productive and let teams cope with large projects. Modern projects are so huge and sophisticated that the high-level text-based language fall into a state of an assembler. It is not a coincidence that source-to-source compilers are used there (Puschel, 2005); (Ayguadé, 2009).

The VIPE IDE (Syschikov, Sheynin, Sedov, & Ivanova, 2014) is the model-based visual integrated environment for software design – from parallel algorithms to portable parallel programs for multicore heterogeneous platforms. The chapter presents its adaptation to a specific domain (Syschikov, Sedov, Nedovodeev, & Pakharev, 2017) by integrating a DSL into VIPE, tacking OpenVX as an example.

BACKGROUND

A visual approach has a long history and has wide support by large players in software development tools, such as Mathworks (Simulink), National Instruments (LabVIEW), Esterel technologies (SCADE) etc. These systems are essentially similar in the main reason for using visual programming approach: make it easier for experts to develop high-quality software in limited time with reasonable resources.

However, these systems have key drawbacks for embedded software programming. Parallel programs which are not model-based ones lack quality and correctness. The aforementioned model-based development environments (MDE) are limited (Simulink) or unable (others) to provide software performance analysis, which is a significant part of embedded systems development. They are closed systems and give no convenient ways for creation of domain-specific languages or libraries. SCADE is the MDE most adapted to the embedded software design. The final stage of the development process is the generation of a C-code, which is not hardware-specific.

The next fundamental approach in visual and domain-specific development is UML. UML is a general-purpose modelling language. It lacks quantifiable notions of time, as well as an ability to express non-functional properties and constraints, take into consideration execution platforms features and characteristics that is vital in embedded software design. All these aspects are particular concerns