Chapter 9
Merging and Splitting Petri Net Models within Distributed Embedded Controller Design

Luis Gomes
Universidade Nova de Lisboa, Portugal
& UNINOVA-CTS, Portugal

João Paulo Barros
Instituto Politécnico de Beja, Portugal
& UNINOVA-CTS, Portugal

Anikó Costa
Universidade Nova de Lisboa, Portugal
& UNINOVA-CTS, Portugal

Filipe Moutinho
Universidade Nova de Lisboa, Portugal
& UNINOVA-CTS, Portugal

Fernando Pereira
Instituto Politécnico de Lisboa, Portugal
& UNINOVA-CTS, Portugal

ABSTRACT

Design of distributed embedded controllers can benefit from the adoption of a model-based development attitude, where Petri nets modeling can provide support for a comprehensive specification and documentation of the system together with verification capabilities and automatic deployment into implementation platforms. This chapter presents a Petri nets-based development flow based on composition and decomposition of Petri net models, using Input-Output Place-Transition Petri nets (IOPT nets) as the underlying formalism, allowing reusability of models in new situations through a net addition operation, as well as partitioning of the model into components using a net splitting operation. Distributed embedded controllers are addressed adding the concept of time domains to IOPT nets. Finally, a tool chain framework is presented supporting the whole development process, from specification to implementation, including property verification, simulation, and automatic code generation for deployment into implementation platforms (considering hardware-based implementation and VHDL coding or software-oriented implementation and C coding).

DOI: 10.4018/978-1-4666-3922-5.ch009

1. INTRODUCTION

According to what is usually referred as Moore's law, stated in 1965, on-chip density (and resources) roughly doubles every 18 months. Unfortunately, this sustained increase of resources available at chip level has not been followed with identical increase of designer's productivity. The lack of methods and tools that adequately support designer's productivity led to what is normally called the *productivity gap*. Several attempts have been carried on by different communities, companies, and consortiums to provide designers with adequate methodologies and engineering tools to reduce this gap. Also, the increase on resources availability allows the development of more complex system, and as the complexity of the design increases, the lack of methods and tools to verify and to assure specific system's properties becomes more visible, leading to what is sometimes called the *verification gap*.

Having an adequate set of selected formalisms and computation models, complemented by a set of the "right" tools, are key aspects to fight against those *productivity and verification gaps*.

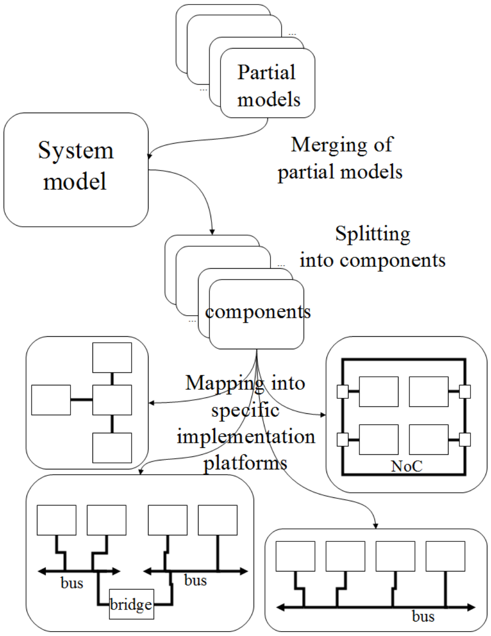
In this sense, along the last decades, the role of Model-Based Development has increased its importance within system design methodologies, as far as it can adequately support designer's needs to integrate the reusability of already available components in the development flow and the use of design automation tools. This should be amenable to support automatic code generation and deployment into implementation platforms and an easy mapping to new or updated platforms.

Complementing the referred concerns, as far as the complexity of the system increases, new challenges for distributed embedded controller design have arrived. These challenges have led to face the need to decompose the system into different components, which could be physically distributed, as well to consider the use of embedded system co-design techniques allowing the concurrent design of hardware and software parts for some components.

Overall, the balanced usage of the referred attitudes have been contributing for improving several metrics on distributed embedded controller design, namely cost-performance relation, time-to-market, system robustness, and power consumption, among others.

Figure 1 summarizes a model-based development flow for distributed embedded controllers, where composition and decomposition of models play a central role. The system model is obtained from the composition of partial models and can adequately support splitting into components to be deployed into implementation platforms, as different as System-on-Chip solutions (including or not network-on-chip communication support), networks of distributed industrial controllers (including multi-bus architectures), or solutions

Figure 1. From partial models to deployment into implementation platforms (adapted from Gomes & Costa, 2006)


Partial models
System model
Merging of partial models
Splitting into components
components
Mapping into specific implementation platforms
NoC
bus
bus
bridge
bus