Multi-Semantic, Decision Networks for Massively Distributed Reconfigurable Systems: Goal-Oriented Programming Models and Dynamic Optimization

Alex Doboli
Department of Electrical and Computer Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794-2350
Email: adoboli@ece.sunsysb.edu, Phone: 631-632-1611

1. Introduction

Small size, low cost electronic circuits embedded on a massively large scale into the quotidian world are likely to transform the way humans interact with their natural and social environments [Weiser02]. Such systems can offer unique capabilities, like autonomous response to new needs, identification of trends and correlations in data and events, reaction to unexpected conditions, and many more. These capabilities are expected to produce significant leaps forward in transportation, infrastructure management, environmental protection, and homeland security, just to name a few areas. The complexity of networked computer systems is likely to grow very rapidly due to more and more electronic circuits being networked together each day.

While massively distributed embedded systems have the potential to offer comprehensive data acquisition, huge computing power, and precise actuation, it is very hard to efficiently harvest these capabilities with current theoretical concepts and software methods. Massively distributed embedded systems are hard to develop and maintain, and their performance does not scale well with the amount of used resources.

In our opinion, there are three main challenges that need to be tackled in order to develop applications based on massively distributed, reconfigurable embedded systems: providing (i) reliable behavior in dynamic conditions, (ii) scalable management of heterogeneous, inter-component interactions, and (iii) performance-optimal co-design of hybrid, reconfigurable components.

- *Reliable behavior in dynamic conditions*. Existing decision making methods, including control and resource management, are not flexible enough to cope on a large scale with changing requirements, e.g., real-time constraints and energy budgets. Centralized control is reliable, but does not scale. Local control works well for large systems but with the exception of simple situations, its overall behavior is unreliable.
- *Scalable management of heterogeneous, inter-component interactions*. Components are often tightly related to each other through signals, shared resources, common requirements, etc. of different nature. Some interactions cannot be characterized a-priori, or might change at execution time. This invalidates popular divide-and-conquer or tree-like hierarchical design as both might involve imprecise pruning of interactions.
- *Performance-optimal co-design of hybrid, reconfigurable components*. The availability of reconfigurable hardware (e.g., reconfigurable mixed analog-digital circuits) enables sensing, actuation, processing, and networking of variable performance. Current

methods are arguably not sufficiently well developed for co-designing the hybrid modules in a system, like the analog and digital circuits in an embedded system.

In summary, while massively distributed systems are likely to create unique opportunities for breakthrough applications, there are currently no systemic theories or scalable methods for effectively co-designing systems built out of very large numbers of heterogeneous parts. Developing a suitable programming model and the related programming tools (e.g., specification, compiling and optimization, debugging, simulation, and run-time support) is one of the important research tasks that ought to be tackled first.

2. What programming model is effective for massively distributed embedded systems?

For complex control systems, the algorithmic processing at the embedded node level (e.g., filtering, search, classification, etc.) tends to be the same for all nodes and over time. In contrast, goals, performance requirements, safety criteria, etc., depend on the application, execution platform, specific execution conditions, and might also change in time. Also, many programming languages for distributed systems rely on explicit description of the inter-node interactions, e.g., synchronization and communication. However, this reduces scalability and reusability of the code as it is hard to capture all possible interactions between the components of massively large scale applications.

The programming model must enable performance-efficient co-design of the hybrid, reconfigurable embedded nodes forming the execution network. Typical applications, e.g., monitoring and tracking, define for a variety of situations the global tasks and goals that must be achieved by a massively distributed system through combined operation of its embedded nodes. As a result, two main design challenges emerge: (i) each sensor node has to efficiently sense, process, and network under a wide range of performance requirements while (ii) only scarce hardware, bandwidth, and energy resources are available. Present co-design methods are insufficient for tackling the two challenges. The methods have only limited capability for co-optimizing the sensing, processing, and communication subsystems of embedded nodes. Also, few methods can exploit the flexibility of networked reconfigurable architectures to produce low-cost yet efficient designs for a broad range of requirements.

We suggest that Visual Programming (VP) is an intriguing approach to programming massively distributed embedded systems as it offers high productivity, scalability, and reuse. VP languages are more intuitive, hence easier to learn and use by persons without comprehensive programming background. The concept of Visual Programming (VP) was arguably proposed in the 80s [Johnston04], however, it is only recently that its advantages for embedded applications became more apparent. VP languages have been proposed for applications like managing smart oilfields, vehicle tracking, contour finding, environmental monitoring, etc. The existing VP constructs include successive filtering and functional processing of data pools. The data model is based on continuous data streams sampled from the environment and groups of nodes defined by their specific interests in space and over time, query-based programming, and global behavior.

Similar to [Newton04, Whitehouse05], we suggest a VP programming notation in which the data model is based on data pools and continuous data streams to the modules. However, it differs in that it focuses on optimal decision making in massively distributed environment, and not on algorithmic descriptions. We think that the notation is orthogonal to the existing languages as it concentrates on the interactions between groups of nodes, or nodes and environment, and less on the behavior of the individual nodes. Also, to offer high scalability, the compiler and optimization environment ought to identify the best interaction schemes between components, so that the overall goals as well as the goals of the modules are met. While describing algorithms is arguably done more efficiently by humans, identifying the parameters for optimal execution is cumbersome and should be automated.

The proposed goal-oriented model comprises of separate description modules (DMs) that operate to optimize a well-defined set of goals while the overall goals of the application are also being optimized. Each module executes a set of parameterized behaviors (algorithms) for which the parameters are automatically computed based on the information provided through the goal-oriented descriptions. Similar to other specification languages for sensor networks, the proposed data model is based on data pools associated to regions and groups. Modules sample inputs from and generate outputs to a data pool for region. Regions represent continuous collections of tokens, such as a geographical area. Groups are discrete collections of tokens. Regions and groups can be associated to a specific physical area of the environment, or can be described by their defining properties.

The goal-oriented descriptions are compiled off-line and uploaded onto the embedded nodes. During execution, a scalable execution support (middleware) dynamically optimizes the cooperation of embedded modules and networks. The system software also monitors events, conditions, data, etc., so that related applications can be automatically correlated or uncorrelated to provide accurate and scalable execution.

References

[Weiser02] Weiser, M., The Computer for the 21st Century, IEEE Pervasive Computing, 1(1), pp. 15-25, 2002.

[Johnston 04] Johnston W. et al. Advances in Dataflow Programming Languages, ACM Computing Surveys, Vol. 36, No. 1, (March 2004), 1-34.

[Newton04] Newton, R., Welsh, M. Region Streams: Functional Macroprogramming for Sensor Networks, Proc. Workshop on Data Management for Sensor Networks (2004).

[Whitehouse,05] Whitehouse, K., Zhao, F., Liu, J. Semantic Streams: a Framework for Declarative Queries and Automatic Data Interpretation, Technical Report, Microsoft Research, MSR-TR-2005-45 (2005).

Alex Doboli is an Associate Professor with the Department of Electrical and Computer Engineering, State University of New York at Stony Brook. Dr. Doboli's research is in Computer-Aided Design of mixed-domain embedded systems and networks of systems. He has published over 100 papers in peer-reviewed journals and conference proceedings. He is an Associate Editor for Integration, the VLSI Journal (Elsevier). He is a Senior IEEE Member, and a Member of SigmaXi and Tau Beta Pi.