Design Environment for Dynamically Reconfigurable Embedded Systems

Ted Bapty, Sandeep Neema, Jason Scott
Institute for Software Integrated Systems, Vanderbilt University
(bapty@vuse.vanderbilt.edu)

Modern high-performance embedded systems, such as Automatic Target Recognition for Missiles or Dynamic Protocols Mobile Communications devices, face many challenges. Power and volume constraints limit hardware size. Accurate, high-performance algorithms involve massive computations. Systems must respond to demanding real-time specifications. In the past, custom application-specific architectures have been used to satisfy these demands. This implementation approach, while effective, is expensive and relatively inflexible. Hardwired application-specific architectures fail to meet requirements and become expensive to evolve and maintain. A fixed, application specific architecture will require significant redesign to assimilate new algorithms and new hardware components.

Flexible systems must function in rapidly changing environments, resulting in multiple modes of operation. On the other hand, efficient hardware architectures must match algorithms to maximize performance and minimize resources. Structurally adaptive, reconfigurable architectures can meet both these needs, achieving high performance with changing algorithms. Reconfigurable computing devices, such as Field Programmable Gate Arrays allow the implementation of architectures that change in response to the changing environment.

Efficient system architectures must encompass a heterogeneous mix of the best technologies. The target systems are built on a heterogeneous computing platform: including configurable hardware, ASIC and general-purpose processors and DSPs. The primary difficulty in a heterogeneous, reconfigurable approach lies in system design. A designer must now maintain a set of diverse system architectures, which exist at different times in the system's lifetime, and map these architectures onto the same group of resources. The designers must manage the behavior of the system, determining the operational modes of the system, the rules for transitioning between operational modes, and the functional properties within each operational mode. In addition, the system must make efficient use of the resources, enabling the designer to minimize the envelope of hardware required to support the union of all operational modes. Current system design tools are insufficient to manage this complexity.

High-level design tools are being developed to capture designs and to generate functional systems as part of the DARPA Adaptive Computing Systems Program. This presentation describes a *model-integrated* approach to be used in the development of reconfigurable systems. There are many significant issues in the development process. The approach described here divides these issues into several categories: (1) Representation and Capture of design information in terms of *Models*; (2) Analysis of the models for design/requirements/resource trade-off studies; (3) Synthesis of architectures and executable systems directly from the models; and (4) Runtime support environments to support efficient execution of the synthesized reconfigurable systems.

The Adaptive Computing Systems (ACS) environment divides the design process into four major categories:

1. **Behavioral Modeling:** the operational adaptive behavior is defined. The designer can specify the operating modes of the system, the legal transitions between modes (and the conditions for transition), and the specifications for system operation while in each operating mode.
2. **Algorithm/Structural Modeling**: potential algorithms are described. The algorithms define signal flow specifications to compute required system outputs.
3. **Resource Modeling**: The resource models describe the hardware available for construction of the system. This consists of physical processors, devices, and the interconnection topology.
4. **Constraint Specification**: These modeling categories are augmented and linked together with a Constraint framework. The Constraints allow user-defined interactions to be specified, establishing linkages between properties in one category and objects in the same or another category.

These modeling aspects are used to capture design specifications and implementation alternatives. Given this information, the designer must analyze and evaluate system designs. Systems must be evaluated from an algorithmic standpoint, to ensure that numerical precision is sufficient to achieve sufficient accuracy. For these activities, the development system supports a link to Matlab. The models can generate a Matlab code, giving access to a wide variety of mathematical and visualization tools.

The next step in developing a system involves evaluating system design alternatives. OBDD techniques are used to evaluate very large design spaces without requiring exhaustive enumeration. Evaluation criteria include timing, power, and size. The result of the analysis is a set of feasible designs.

The feasible designs are simulated to evaluate timing performance. Simulations can be executed at various levels of detail to allow either a large number of alternatives to be evaluated at a gross level of detail or a few designs to be evaluated with a high fidelity. The result of this step is a small set of design solutions with high confidence of success.

These design solutions are used to synthesize the target, dynamically adaptive, system. Software synthesis consists of creating real-time schedules and communications maps. The hardware components are created with a synthesis of structural VHDL code containing the proper interfaces and “glue” logic.

The entire system executes upon a software real-time kernel/hardware virtual kernel with a closely linked Reconfiguration Manager. The kernels implement a dataflow architecture, with asynchronous interacting nodes. The reconfiguration manager contains all necessary modal information and all hardware/software configuration information, and executes the behavior specified in the models.

This design environment is being demonstrated in the development of a dynamically adaptive missile Automatic Target Recognition application for the US ARMY. The application requires 10 GOPS, given less than 40 watts, in less than ½ sq. ft. A missile lifespan encompasses many distinct modes, with various performance constraints and disparate functional requirements. The application is an effective demonstration of the complexity inherent in designing and synthesizing real-world reconfigurable systems.