A Survey of Embedded Systems Tools

Embedded systems must operate under special constraints beyond those which usually apply to general-purpose computers. Such constraints might limit aspects of the hardware architecture, including its cost, power consumption, processor speed, and memory size. Other constraints might require the embedded software to meet real-time deadlines or avoid unsafe system states. A number of free downloadable tools have been developed by research institutions to help system designers face the daunting task of creating embedded systems which can comply with these types of restrictions. These tools allow system designers to specify, verify, analyze, and simulate complex real-time, distributed, and embedded systems. This article provides a brief survey of a few such tools, including the Generic Modeling Environment, Giotto, HyTech, Metropolis, Mocha, POLIS, and Ptolemy II. For more information or to download one of the tools, visit the appropriate provided URL.

The ESCHER institute, a consortium of university and industrial participants in embedded systems research, also maintains a repository of peer-reviewed, quality-controlled tools and software for embedded systems. This repository includes several of the tools reviewed by this article as well as a number of additional tools and software. More information about ESCHER may be found at www.escherinstitute.org.

GME: www.isis.vanderbilt.edu/Projects/gme/

The Generic Modeling Environment (GME) is a configurable tool framework for creating domain-specific modeling, model analysis, and model transformation environments. The configuration is accomplished by using formal metamodels to specify the abstract and concrete syntax of a modeling paradigm, or domain-specific modeling language (DSML). The modeling paradigm contains all the concrete and abstract syntax information of the domain, including which concepts will be used to construct models, what relationships may exist among those concepts, how the concepts may be organized and viewed by the modeler, and what well-formedness rules govern the construction of models. GME also provides high-level C++ and Java interfaces for writing plug-in components to traverse, manipulate, and interpret models.

GME is widely used in embedded systems research. Example projects include the Automatic Integration of Reusable Embedded Software toolkit (AIRES) developed at the University of Michigan and the EAST Architecture Definition language (EAST ADL) built for the EAST-EEA project.

Giotto: http://www-cad.eecs.berkeley.edu/~tah/giotto/

Control applications typically consist of periodic software tasks grouped with a mode switching behavior for enabling and disabling specific tasks. Giotto is a tool-supported methodology for the development of distributed real-time embedded control systems using time-triggered task invocations and mode switches. The Giotto system consists of a time-triggered programming language with a formal semantics, a compiler, and a runtime system. Giotto's programming language provides a strict separation between

platform-specific scheduling and communication issues and platform-independent functionality and timing concerns such as sensor readings, task invocations, actuator updates, and mode switches. This separation of concerns permits flexibility in the choice of the control platform and facilitates significant automation in the validation and synthesis of control software. Because the time-triggered nature of Giotto achieves timing predictability, the Giotto system is especially suitable for designing safety-critical applications.

HyTech: www-cad.eecs.berkeley.edu/~tah/hytech/

HyTech, a symbolic model checker for linear hybrid automata, is an automatic tool for embedded systems analysis. HyTech provides model-checking facilities which do more than simply determine if a system satisfies its correctness constraints – they also provide relevant diagnostic information to facilitate design and debugging. If a linear hybrid system description contains design parameters with unspecified values, HyTech can automatically compute the conditions under which the system satisfies its temporal requirements. If system verification fails, then HyTech generates a diagnostic error trace.

Metropolis: www.gigascale.org/metropolis/

Metropolis is an environment for designing complex embedded systems. It allows users to define various levels of abstraction to formally represent and refine the system from design through implementation. Metropolis also allows users to formulate the problems to be addressed at and across these different abstraction levels. The Metropolis design environment includes a modeling language, infrastructure, libraries and tools. The core of the Metropolis infrastructure is a metamodel of computation, which allows users to model various communication and computation semantics in a uniform manner. It also includes a tool interface for integrating new tools (e.g., verification) with the Metropolis environment, a simulator for verifying temporal properties with the SPIN model checker, and libraries providing typical application services such as FIFO communication semantics. Metropolis provides multiple tools for model simulation, analysis and verification, and synthesis.

Mocha: www-cad.eecs.berkeley.edu/~tah/mocha/

Mocha is an interactive software environment for the specification and verification of embedded systems. It serves as a tool for developing new verification algorithms and approaches. Mocha allows the specification of heterogeneous synchronous, asynchronous, and real-time systems using a language of hierarchical components. It provides a temporal logical language for the formal specification of design constraints as well as verification through model checking. Mocha also enables user-guided and/or automatic system execution. jMocha, a Java implementation of Mocha, provides a graphical proof manager and a new scripting language for rapid, structured symbolic algorithm development.

POLIS: www-cad.eecs.berkeley.edu/Respep/Research/hsc

POLIS, a tool for hardware/software co-design, centers on a concurrent finite state machine-like representation called Co-design Finite State Machines (CFSM). Designers

may implement their specifications in a higher-level language such as ESTEREL and directly convert them into CFSMs. POLIS incorporates a formal verification methodology which uses certain CFSM-specific abstractions and assumptions to verify large, complex designs. POLIS also includes a translator from the language of CFSMs FSMs which can interface with external FSM-based verification systems. Designers can simulate systems using POLIS based on design decisions such as choice of CPU and scheduler. POLIS supports hardware synthesis, application-specific real-time operating system synthesis, and software synthesis in the form of C code.

Ptolemy II: ptolemy.eecs.berkeley.edu/ptolemyII/

Ptolemy II is a Java-based infrastructure for experimenting with concurrent real-time embedded system design techniques. It supports the heterogeneous modeling, simulation, and design of systems through the assembly of concurrent components. Users may specify a heterogeneous mixture of different models of computation to govern the concurrency model and inter-component communication mechanisms. Ptolemy II includes an extensible library of ready-made components, and new components may be specified using Java, Python, MATLAB, or a component definition language called Cal. It also includes an extensible library of interoperable models of computation. The Ptolemy II graphical user interface is extensible too, allowing custom visualizations of components, component interactions, and model data.