1. Introduction

Ptolemy II is a software framework that supports modelling and simulation of real-time embedded systems, particularly emphasizing the concurrency among the components of such systems. Together with its graphical user interface Vergil, Ptolemy II provides a visual language that allows the creation of system models that can be validated through simulation and refined towards a final implementation using code generation techniques. It was developed at UC Berkeley by the group of Prof. Edward A. Lee, which periodically releases new versions of the framework (version 7.0.1 at the time of writing).

2. Description

Unlike other block-based modelling and simulation languages such as LabVIEW and Simulink, Ptolemy II includes the definition of the execution semantics as a part of the model rather than of the underlying simulation engine. This means that it is up to the programmer to decide how do the system components – the actors – interact with each other over time. Such decision is usually restricted to a set of well defined models of concurrent computation (such as synchronous dataflow or communicating sequential processes) which are provided as a library, but the framework is flexible enough to support extensions that implement user-defined execution semantics. Figure 1 shows a snapshot of Vergil displaying a number of interconnected actors and a director (green rectangle on the upper left corner, which representing a model of concurrent computation that in turn defines the execution semantics of that model).
Actors can be hierarchically composed, and different directors can determine the execution semantics of each hierarchical layer. With this feature, Ptolemy II simplifies the modelling and validation of systems with heterogeneous behaviour regarding time and concurrency, as each of the subsystems can be described at a different hierarchical level using a different model of concurrent computation. Examples of such heterogeneous systems include analogue/digital circuits, hardware/software systems or electrical/mechanical devices. However, not all actors can be executed under different models of concurrent computation. Those presenting such capability are referred as \textit{domain polymorphic}.

Ptolemy II models can’t be directly compiled or synthesized towards an implementation, but there is the possibility to automatically generate programming language code out of them. The Copernicus code generator, which was part of the Ptolemy framework up to its 4.0 version, generates Java bytecode out of non-hierarchical models based exclusively on the synchronous dataflow director. Since Ptolemy II libraries are composed of actors written in Java, the major functionality of Copernicus is to rearrange the code so that it doesn’t depend on the infrastructure of the Ptolemy simulator.

Codegen, which was included in the version 6.0 of the framework, aims to generate C source code from models that use synchronous dataflow, heterochronous dataflow or finite state machine directors. It is based on code templates that must be added \textit{a priori} to the actor libraries. Actor libraries that were not extended with code templates can’t be used within models that are used as input to Codegen. The code generation process doesn’t include sophisticated optimisations and simply integrates the output of the multiple templates.

3. Links and References


Ptolemy website:  http://ptolemy.berkeley.edu