Multidimensional Performance Modeling for Advanced, Embedded, Signal Processors

Michael Stebnisky
Lockheed Martin Advanced Technology Laboratories
1 Federal Street
Camden, NJ 08102
mstebnis@atl.lmco.com

Background

Traditional performance modeling approaches for embedded signal processors are unable to address emerging requirements and component technologies. This is a result of an increased awareness and need for dynamically adaptive or reconfigurable systems, particularly in the areas of power dissipation/ performance. Most of the awareness results from research and development programs, such as UC Berkeley's Infopad; Defense Advanced Research Project Agency's Adaptive Computing Systems, Power Aware Computing and Communications, and Polymorphous Computing Systems. The availability of commercial, off-the-shelf processors that scale clock and voltage, like StrongARM and Crusoe, have also provided near-term motivation.

Objectives

With the emerging availability of hardware and software components that provide on-the-fly optimization of power, performance, and other system parameters comes the need to develop decision-support systems that enable designers to take advantage of these characteristics so as to model, simulate, and compare systems against alternative implementations.

Our research seeks to develop algorithms, methods, and rapid-prototyping tools that will enable designers to evaluate system implementations for performance and power optimization as well as other system parameters, such as resolution, latency, and quality of service. Developing new architectures with significantly improved performance, without increased power dissipation, is an important application. We have made considerable progress against our objectives.

Our focus has been to address heterogeneous signal-processing systems that may be rapidly, dynamically reconfigurable. We use data-flow graphs to describe system functionality. Each task in a data-flow graph may have one to many different implementations, encapsulated in a component library, and may differ in one or several of the set of characteristics described above. The objective is to optimally schedule tasks based on the dynamically varying objective function. This involves determining which processors are available and determining which task implementation for which processor best meets the current objective function. Processors may be unavailable, because they are busy, determined faulty, or marked as unavailable.

Design-time and Run-time Elements

To maximize the effectiveness of our approach, the overall system includes online and offline elements, where offline and design-time are roughly synonymous, as are online and run-time. The objectives of the offline portion are to select an optimally minimal set of task implementations as well as capture information about the applications, data-flow graphs, system hardware, etc. that can be used at runtime to expedite the selection of task implementations. This may be performed manually or by using Monte Carlo searching techniques. The library is large, and it may contain many components that are not relevant to a particular system. The first step then is to prune it to a subset of relevant components. Succeeding steps further prune the set of components to that which best services the defined objectives. Given a known set of applications, baseline architecture, and one or more of the indicated libraries, an optimum subset of components can be determined.

Given an appropriate subset of components, the next requirement is to develop methods and algorithms to optimally assign tasks to processors. This assignment is based on processor availability and characteristics of all the task implementations that can execute on the available processors. The assignment method is as an extension of a simple but effective agent-based dynamic scheduler.

Using all available information developed offline, online optimization focuses on dynamic selection of execution components that best match complex objective functions.

We are developing the algorithms and methods as extensions to our CSIM (C Simulator) tool. Information on CSIM is available from our website at: http://www.atl.lmco.com/proj/csim/ We plan further improvements to the algorithms and methods used in this development.

Results To Date

We have focused our development on the online capability, and performed the following:

- Reused a data-flow graph and processor architecture from a recently completed application.
- Modified the baseline architecture, consisting of ten SHARC processors and various interconnect elements (RACE NIC, RACE XBAR, local bus), to represent a system of six PowerPC processors, two FPGA-based reconfigurable coprocessors, and two ASIC coprocessors.
- Defined a library of estimates of several implementations for five different characteristics and mode assignments for each task in the data-flow graph.
- Modified CSIM's Dynamic Scheduler to schedule tasks and processors based on complex objective functions.
- Defined five operating modes with different objective functions: BASE mode, a safe mode where task implementations are fixed assignments; FAST mode, a fixed assignment mode where some tasks have a preferred implementation that executes faster than BASE mode; FASTEST mode, where all task implementations for every available processor are evaluated and the fastest is executed; LOPWR mode, where all task implementations for every available processor are evaluated and the lowest in power dissipation is executed; and DDPSWC mode, where all task implementations for every available processor are evaluated and the task implementation selected for execution maximizes the function 1/(delay*delay*power*size*weight*cost)).
- Added a capability to mark processors as “faulted” or “unavailable” for execution. Processors may be marked or unmarked dynamically.
- Tested and demonstrated the capability, including execution in all five modes with up to nine unavailable processors.
- Overall improvement in max performance/min power ratio of 5:1 achieved for the base case.
- Additional applications are in evaluation.

Conclusions

Emerging system-level requirements and component technologies are demanding new capabilities from rapid-prototyping systems. We are developing capabilities to simply and effectively address these requirements. Results to date include a 5:1 improvement in maximum performance/minimum power ratio for our base case.