Hardware/Software Partitioning of Synchronous Dataflow Graphs in Ptolemy

Final Report
EE382 C – 9 : Embedded Software Systems
May 12, 1999

Heather Hanson
Gayathri Manikutty

Table of Contents

Abstract

We discuss an algorithm known as Global Criticality/Local Phase (GCLP) that partitions a design into hardware and software, and present our implementation of the algorithm in a design environment tool. This algorithm considers multiple resource constraints (for example, physical layout area and time to execute) and iteratively maps sections of the system into hardware and software modules until it reaches a solution that satisfies the design constraints. The design tool consists of a graphical interface that guides the user through the co-design, and the GCLP algorithm as a standalone C program. We tested the tool and methodology with a two-channel filterbank design.

Introduction and Context of Work

Hardware/Software Co-design

Embedded processors for real-time systems must meet stringent requirements in terms of performance and power dissipation while keeping the product cost low and development cycle short. One successful method of designing embedded systems is exploiting the synergism of hardware and software through their concurrent design. Franke and Purvis [1] define co-design as "the system design process that combines the hardware and software perspectives from the earliest stages to exploit flexibility and efficient allocation of function." An important phase in co-design is hardware/software partitioning, which maps each subtask of the application into hardware or software and selects an appropriate implementation. The difficulty is to choose from the many mapping and implementation alternatives available such that the overall design is optimized, and to do so quickly.

Ptolemy and the Adaptive Computing Systems Domain

Ptolemy is a design environment developed at the University of California at Berkeley for designing, modeling, simulating and prototyping heterogeneous systems [2]. Named after an early Greek astronomer, Ptolemy's component names are related to astronomy: stars, galaxies, universes, etc. A star is the most basic block, and galaxies are clusters of stars. A universe is the top level of a design, represented as a block diagram in Ptolemy's graphical interface.

Ptolemy is organized into domains based on different computation models. In addition to domains for simulation, such as synchronous dataflow (SDF), there are also domains for code generation, such as the Code Generation in C (CGC) and VHDL (Very High Speed Integrated Circuits Hardware Description Language) domains.

A recent Ptolemy development is the Adaptive Computing Systems (ACS) domain [3]. Applications modeled in this domain include signal processing and communication systems, such as modems. They are typically implemented in a mixture of digital signal processors, reconfigurable (adaptable) hardware such as field-programmable gate arrays (FPGAs)[4], and software—a mixture well suited for co-design. In the ACS domain, stars have a single interface and multiple implementations, allowing users to select different implementations without modifying the system's block diagram. Currently, a single universe in the ACS domain can support only a single type of implementation (either hardware or software, but not both simultaneously). Future releases will support multiple implementations, enabling users to partition designs into hardware and software sections.

Summary of Previous Work

Partitioning a design into hardware and software has been the focus of ongoing research [5] [6] [7]. Here, we present a brief overview of this partitioning problem and a heuristic to solve it.

The Binary Partitioning Problem

The SDF domain in Ptolemy translates the task-level description of an application into a Directed Acyclic Graph (DAG) where each node represents a computation and the arcs represent the data precedence between the nodes. The partitioning problem is to decide the mapping for each node of the DAG—either into hardware or into software—and arrive at a schedule (the start times for each node) such that the total area of nodes mapped to hardware is minimum. The schedule is subject to latency constraints (the upper bound on the total application execution time) and resource constraints (which includes availability of hardware and software resources, namely program and data memory). The area and latency estimates for hardware and software

mappings of all nodes, as well as the communication and interfaces, must be known for an effective partition.

Solving this partitioning problem is computationally intensive. Integer Linear Programming could be used to solve constrained optimization problems such as the binary partitioning problem but exact solutions are combinatorial in the order of $O(2^N)$ where N is the number of nodes [6]. A heuristic known as the Global Criticality/Local Phase (GCLP) algorithm—with a worst case computational complexity of O(N*A) for N nodes and A arcs—has been proposed to solve these problems [8].

GCLP Algorithm

There are two possible objective functions to be considered while solving the mapping problem: minimize the finish time or minimize the resource consumed by the node. These are contradictory goals. To accommodate both goals, the GCLP algorithm adaptively selects the objective function to be minimized at each step in the form of global criticality and local phase measures. Figure 1 shows a flow chart for the iterative GCLP algorithm.

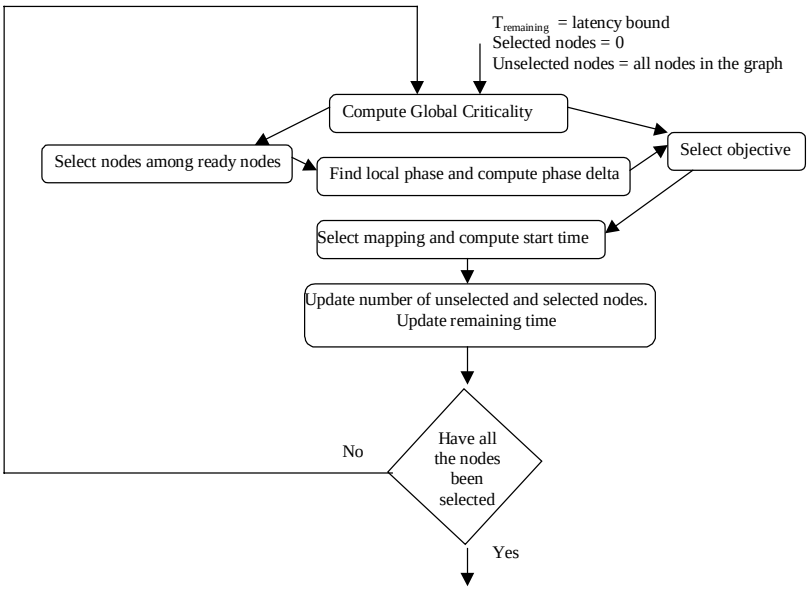

Figure 1: GCLP flow chart [8]

Initially all nodes are assumed to be mapped to software. Since GCLP has not performed the mappings yet, it assumes the nodes to be unmapped. At each time step, the schedule for all the mapped nodes is known. If it is not possible to map all the unmapped nodes into software and complete execution by the latency bound, some of the unmapped nodes are moved to hardware. The fraction of unmapped nodes moved from software to hardware at a given step gives the Global Criticality (GC) at that step. A high GC implies that time is critical and hence the algorithm minimizes the finish time of the nodes.

Local properties of the nodes are taken into consideration by classifying nodes into three types: extremity, repeller and normal. An extremity node is one that would use a large amount of resource—area or time—in a given hardware or software implementation and thus should be mapped to the other implementation. The local preference of such nodes, referred to as the local phase delta, forces the algorithm to choose minimum area as the objective function instead of finish time.

To further swap nodes between hardware and software, the concept of a repeller is introduced. A node is classified as a repeller based on its structure: bit manipulations are better suited for hardware and hence a node performing many bit manipulations would be a software repeller. A node that falls into neither of these groups is a normal node [6]. The algorithm repeats this adaptive selection of objective functions until all nodes are mapped.

Design Objective

Our contribution to hardware/software co-design research is creating a tool to assist designers partition a design into hardware and software modules using the GCLP algorithm. A design assistant using the GCLP algorithm was previously prototyped within the Ptolemy software environment in a specialized co-design domain, [7] but was not released. Our mapping tool is

available for use with Ptolemy's CGC and VHDL domains. The tool guides users through the partitioning process, helping them transform a Ptolemy universe into a C program and VHDL description.

Formal Modeling

The final product of a partitioned design will be a software program and a set of hardware specifications, customized for the application. In order to generate files for software and hardware components, we mixed the CGC domain and the VHDL domain. Both these domains adopt Synchronous Dataflow (SDF) semantics. SDF designs are statically scheduled, meaning that the order of execution for each block is determined at compile time rather than at execution time. Thus, the C code is statically schedulable and data memory is statically allocated. Because the VHDL domain also uses SDF semantics, a galaxy defined in the CGC domain may be used in the VHDL domain—and vice versa—by simply changing the domain parameter in Ptolemy's graphical interface [3]. These properties are beneficial for hardware/software partitioning.

Implementation

The GCLP mapping algorithm has been written as a standalone C program. To integrate partitioning information into Ptolemy, we split the process into two phases:

- providing information from Ptolemy to the GCLP code, and
- bringing the mapping results back into Ptolemy.

Preparing GCLP Input

The user begins the design process with a software-only design in the CGC domain. The first phase consists of invoking two design methodology management (DMM) universes. These are called DMMs because they manage files involved in partitioning. The first universe accepts a

design in the CGC domain as an input and generates a Ptolemy script file in ptcl (Ptolemy tool command language) format which describes the CGC universe in textual format.

The results of the partitioning process depend heavily on the estimates of software and hardware execution times and available resources for each block of the system to be partitioned. Presently, we assume that the user would be able to provide the necessary inputs for resource estimates. The first DMM universe invokes an interactive tcl (tool command language) window to accept the hardware/software resource requirement estimates and the area and timing constraints. The user's input and the graph obtained by parsing the ptcl file are stored in a file which is then processed by the GCLP algorithm. The second DMM universe accepts arguments required for executing the GCLP algorithm and calls the GCLP program. Upon execution, the algorithm produces a file with mapping recommendations.

Partitioning and Co-simulating

In the second phase of implementation, the designer uses the GCLP mapping information to partition the design. The designer modifies the original block diagram by switching recommended blocks from software (in CGC) to equivalent hardware blocks (in VHDL), either by changing galaxies' domains or by replacing blocks. During this step, the user should ensure that design elements in CGC and VHDL correspond exactly. Then, the user selects a hierarchical target—a schedule manager that will schedule the software and hardware components separately and coordinate communication between the two types of resources [9]. Finally, the designer executes the design in Ptolemy, which generates code in C and VHDL, compiles the code, and simulates the design. The C program contains a system call that invokes the VHDL simulator, allowing for co-simulation of the hardware and software components.

Test Application and Results

We modified an existing CGC signal processing demonstration—a two-channel filter bank—to test our partitioning tool. The design consists of signal generators, two sets of filters, and a block that displays the output. Table 1 shows the hardware area, program memory, and execution time (rough estimates only) required for three versions of the filterbank.

Table 1: Comparison of Area, Memory, and Execution Time for Two-Channel Filterbank

	hardware area	memory	execution time
Software-only	-	791	616
User-determined partition	28	503	381
GCLP-recommended partition	44	47	201

In this test, partitioning the design according to GCLP's suggestions produced a design with shorter execution time, less program memory required, and more hardware area required (though still within the area constraint) than the user-determined partition.

Conclusion

Integrating the GCLP algorithm as a partitioning tool in Ptolemy helps designers partition designs into hardware and software within the Ptolemy design framework. The first phase of the partitioning is implemented as a series of menus in Ptolemy's graphical interface. In the second phase, the designer uses mapping information to partition a design into hardware and software components, generate and compile code, and simulate the design. By adding this tool to Ptolemy, we have made the co-design process more accessible. The GCLP algorithm can provide a near-optimal partition that can be further refined by the user.

Future Work

The following points could be considered for future work to automate the partitioning tool:

- The hierarchical scheduler could be modified to consider the GCLP-generated schedule.
- The ptcl description of designs could be expanded to descend into wormholes, so that VHDL sub-blocks could be specified in ptcl's text format. This step would be necessary to ensure compatibility for CGC and VHDL blocks that use customized parameters or that do not have identical functions, such as FIR stars.
- The first phase of the implementation—the DMM universes—could be ported to the ACS domain, and the second phase—partitioning, scheduling, and generating code—could be written into a multiple-implementation ACS target.

References

1. D. W. Franke, M. K. Purvis, "Hardware/Software Co-design: A Perspective," *Proc. ACM Int. Conference on Software Engineering*, pp. 334 –352, Austin, Texas, USA, May 1991.

2. J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems," *Int. J. Computer Simulation*, vol. 4, pp. 155-182, April 1994.

3. E. A. Lee, *et al.*, University of California at Berkeley, *The Almagest*, Volumes 1-3, Regents of the University of California, 1995.

4. J. Villasenor and W. H. Mangione-Smith, "Configurable Computing," *Scientific American*, vol. 276, no. 6, pp. 54-9, June 1997.

5. A. Kalavade and E. A. Lee, "A Hardware/Software Co-design Methodology for DSP Applications," *IEEE Design and Test of Computers*, vol. 10, no.3, pp. 16-28, Sept. 1993.

6. A. Kalavade and E. A. Lee, "The Extended Partitioning Problem: Hardware/Software Mapping, Scheduling, and Implementation-bin Selection," *Journal of Design Automation for Embedded Systems*, vol. 2, no. 2, pp. 126-163, March 1997.

7. A. Kalavade, "System Level Co-design of Mixed Hardware-Software Systems," Technical Report UCB/ERL 95/88, Ph.D. Dissertation, Dept. of EECS, University of California, Berkeley, Sept. 1995.

8. A. Kalavade and E. A. Lee, "A Global Criticality/Local Phase driven Algorithm for the Constrained Hardware/Software Partitioning Problem," *Proc. IEEE Int. Workshop on Hardware/Software Co-design*, pp. 42-48, Sept. 22-24, 1994.

9. J. L. Pino, S. S. Bhattacharyya and E. A. Lee, "A Hierarchical Multiprocessor Scheduling System for DSP Applications," P*roc. IEEE Asilomar Conference on Signals, Systems, and Computers*, vol. 1, pp. 122-6, Oct.30 – Nov.2 1995.