

SRC Student Symposium 2006 Technical Paper Form

Presentation Title: Automatic Architecture Selection for Custom Processors

Presenter, University: Jelena Trajkovic, University of California, Irvine

Presenter's Expected Graduation Date: 03/31/2007

Author(s), Organization(s): Jelena Trajkovic, University of California, Irvine;

Daniel D. Gajski, Center for Embedded Computer Systems, University of California, Irvine

SRC Research Task ID: 1118.001

Task Leader: Daniel D. Gajski

Abstract: We present a methodology for automatic creation of custom data path for a given application that optimizes performance and cost under given constraints. The inputs to the architecture generator include application source code, operation execution frequency obtained by the profile run and a set of available components. The output is the application specific data path specified as the set of resource instances and their connections.

Motivation: System on Chip (SoC) design has fueled a need for specialized processors for different application domains. Full HW design is too expensive and rigid for tuning the design later for debugging and feature extension while general purpose embedded processors and ASIPs are often too slow and power hungry. Although custom processors are a better fit for SoCs, they still need to be adapted to run the chosen application efficiently. Such adaptation includes the selection of appropriate data path architecture. Manual design of data path can be time consuming and error-prone. Therefore, we propose the automatic generation of the data path architecture based on the profile of the application and the system performance/resource constraints. We follow a design methodology for custom processors that separates the allocation of architectural resources and their connections from the scheduling of operations.

Concept: We propose a custom processor design methodology for the No-Instruction-Set Computer (NISC). NISC removes the decoding stage and stores the control words (that drive the datapath) in the program memory. The NISC compiler compiles the application for a given datapath, creating a set of control signals that drives the components at runtime. By removing the instruction set, the data path can be easily changed, parameterized and reconfigured. Hence, the NISC concept allows separation of scheduling and allocation thereby simplifying the optimization of datapath.

Approach: The allocation algorithm starts with source code of the application and derives component and connection requirements based on the available parallelism. The derived datapath is further refined by adding pipelining, chaining and forwarding. We have developed novel heuristics that are used to match application requirements to available components. The components are instantiated and connected to create a preliminary datapath, which is then input to the estimation and datapath refinement phase, along with operation frequencies obtained by profiler and design constraints. The designer may specify constraints, namely the target execution time, total cost, upper bound on number of component instances etc. We estimate execution overhead and cost while varying the number of instances of every component. Using automatic data path generation allows for fast evaluation of different design decision and therefore fast design space exploration. The library components are annotated with power and area information allowing us to use these metrics as constraints and to incorporate them into the cost function. We also plan to extend our allocation algorithm to support selection of memory hierarchy architecture for both instructions and data memory subsystem.

Results: Based on our allocation algorithm, we have developed a tool to prove that our approach is feasible and removes the need for time consuming and error prone manual datapath design. Several cost functions were applied and used to compute cost for the preliminary and refined architecture. We evaluate our heuristics for matching resource requirements to set of available components for different cost functions. Our preliminary results demonstrate efficiency of our proposed datapath allocation methodology on a variety of applications and optimization criteria.