Using Constraint Logic Programming in Memory Synthesis for General Purpose Computers

Renate Beckmann* and Jürgen Herrmann**
University of Dortmund,Department of Computer Science XII* and I**
{beckmann@ls12,herrmann@ls1}.informatik.uni-dortmund.de

Abstract

In modern computer systems the performance is dominated by the memory performance. Currently, there is neither a systematic design methodology nor a tool for the design of memory systems for general purpose computers.We present a first approach to CAD support for this crucial subtask of system level design. Dependencies between influencing factors and design decisions are explicitly represented by constraints, and constraint logic programming is used to make the design decisions. The memory design is optimized with respect to several objectives by iterating the (re)design cycle. Event driven simulation is used for evaluation of the intermediate results. The system is organized as an interactive design assistant.

Introduction

A memory system nowadays consists of a hierarchy of components ranging from small, fast and expensive ones, placed near the CPU (i.e., register, buffer, first level cache), to large, slower and cheaper ones (i.e., second level cache, main memory, secondary memory). The main component under design is the cache. To speed up a memory reference, requested data has to be available in a fast memory component. This requires a good organization of the memory system. Parameters of a cache are e.g., number of cache levels, size, and associativity. “Good” values for these parameters depend on the time and order of the different data requests. These, in turn, depend on two factors: application programs (i.e. locality of the references) and underlying computer architecture (i.e. pipelining). The task can be described as designing an optimal memory system for a given general purpose computer architecture and a class of application programs with respect to given criteria (objectives) like access time, chip area and cost.

Memory synthesis can be formalized as a “*parameter selection problem*”. Constraint logic programming (CLP) is an adequate programming paradigm for this application

SPEISE’s Design Cycle

After analyzing the application program features the design cycle starts with restricting the search space by constraints expressing the relations between input data and design decisions. Then decision making is done by parameter labelling with respect to the objectives. Memory components are mapped to existing on-/off-chip modules described in the technology database. Evaluation is based on the performance, computed by event driven simulation. If the expected performance is met (re)design stops. Otherwise a consistent set of possible revisions of design decisions are determined (‘deficiencies analysis’ and ‘redesign planning’). They are expressed in form of constraints, that are imposed at the beginning of the next redesign cycle.
SPEISE is implemented in the CLP language ECLIPSE. In most cases, when a cache and TLB hierarchy for several computer architectures was designed, few (5 - 10) redesign cycles where sufficient to meet the expected performance.

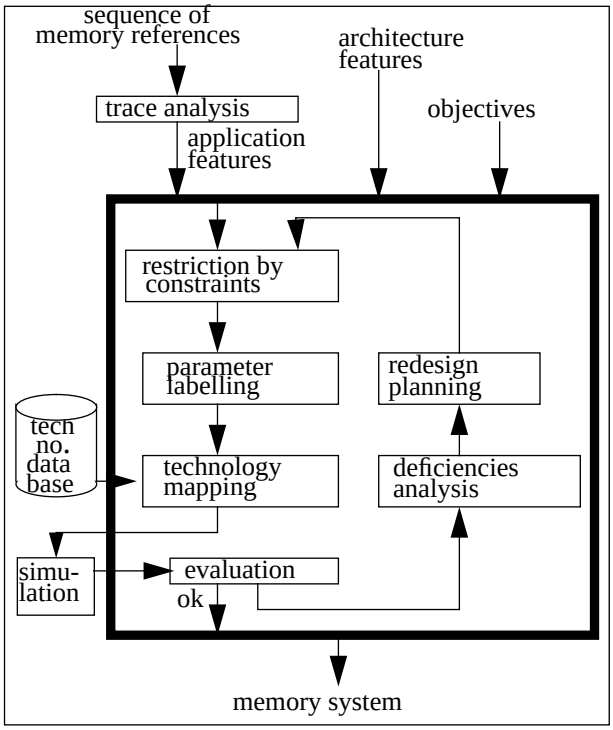

Fig. 3: Components of SPEISE

References:

Renate Beckmann, Ulrich Bieder, Ingolf Markhof, “Application of Constraint Logic Programming for VLSI CAD Tools”, International Conference of Constraints in Computational Logics, München, September 1994.

ED&TC ’97 on CD-ROM