1 Motivation

The flow of a liquid around an obstacle can be described by a set of partial differential equations, known as the Navier–Stokes equations. It requires very complex calculations to solve these equations for a given problem. Moreover it is very tedious to describe the boundary conditions for non–trivial problems. Instead of using numerical methods to compute the liquid flow one can also describe the local behaviour of the particles on a molecular level. This model is called a lattice gas and most often implemented by a cellular automaton. The description of the particle behaviour is called the local rule of the cellular automaton. A two dimensional model that shows almost all the properties of the Navier–stokes equations is the FHP–gas[1]. Field sizes of 1 million cells and more are common to get precise results. The application of the local rule to all cells of the field is called a generation. Between 1000 and 10000 generations are required in lattice gas models to reach a stable state. Thus the rule must be applied to the cells $10^9$ to $10^{10}$ times. This takes several hours in a software simulation and thus special hardware has been developed to speed up this simulation.

2 Overview

The main topic of this paper is an uncommon high–level synthesis approach. Combined with a special post processing step it is capable of producing circuits that can run at very high speed in FPGAs.

At first we will give a very short introduction to cellular processing and the programming language CDL[2], since this is the area to which we apply hardware synthesis. Then we will give an overview on the special hardware (CEPRA–1X [3]) used for the computation. This hardware uses a table–lookup FPGA to compute the next state of the cells and is capable of computing up to 33Mio. cells/s. A new structural program can be loaded into the FPGA for each new problem. Thus the machine can be called a custom computing machine. A synthesis system is required to transform the high–level description of the algorithm in CDL into configuration data for the FPGA. This synthesis system and an accompanying post processing step are presented in detail.

---

$^1$This limit is set primarily through the maximum bandwidth of the PCI bus.
3 High–Level Synthesis of CDL

CDL was introduced as a new cellular programming language for several reasons.

- The description of the rule becomes much more comprehensive and compact than in traditional programming languages.
- The description of the rule becomes target architecture independent. Thus the user does not require any special knowledge about the special hardware.
- Rapid prototyping becomes available through software simulation.

For historical reasons we decided to generate a pure combinational circuit to evaluate the rule given in CDL. All loops are unrolled at compile time allowing us to apply constant propagation and dead code elimination extensively. In the full paper we will give a detailed description of this synthesis process.

4 Post–Synthesis Pipeline Generation

The resulting circuit of the synthesis process has an overall delay of up to 300 to 400ns. Yet the executing machine can only gain optimum performance if the delay does not exceed 30ns. Thus we introduce a special post processing step, that automatically generates as much pipeline stages, as are required to fulfill this condition. We insert these pipeline stages in the circuit after technology–mapping, because the timing of the stages can be estimated much more precisely than before technology mapping. The inserted pipeline stages don’t even produce much hardware overhead, since each lookup–table in the FPGA is followed by an optional flip–flop. In the full paper we will present this processing step in detail and give some experimental results.

The presented combination of stateless synthesis and post synthesis pipeline generation is not only applicable to cellular processing and the CEPRA–1X, but can also be used for other problems where much data is streamed through a computational unit.

References

