The Massively Parallel Computing Model GCA

Rolf Hoffmann

Technische Universität Darmstadt, FB Informatik, FG Rechnerarchitektur, Hochschulstraße 10, D-64289 Darmstadt, Germany
{hoffmann}@ra.informatik.tu-darmstadt.de

Abstract. The Global Cellular Automata Model (GCA) is an extension of the Cellular Automata Model (CA). Whereas in the CA model each cell is connected via fixed links to its local neighbors, in the GCA model each cell is connected via data dependent dynamic links to any (global) cell of the whole array. The GCA cell state does not only contain data information but also link information. The cell state is synchronously updated according to a local rule, modifying the data and the link information. Similar to the CA model, only the own cell state is modified. Thereby write conflicts cannot occur. The GCA model is related to the CROW (concurrent read owners write) model and it can be used to describe a large range of applications. GCA algorithms can be described in the language GCA-L which can be compiled into different target platforms: a generated data parallel multi-pipeline architecture, and a NIOS II multi-softcore architecture.

Keywords: Global Cellular Automata, Parallel Programming Model.

1 Introduction

Since the beginning of parallel processing a lot of theoretical and practical work has been done in order to find a parallel programming model (PPM) which fulfills at least the following properties

- **User-friendly:** easy to model and to program
- **System-designer-friendly:** parallel processing target architectures supporting the model are easy to design and to implement, and programs can easily be translated into these architectures
- **Efficient:** The applications can efficiently be executed on the target architecture
- **Platform independent:** The PPM can also be mapped (interpreted, simulated) without much effort onto other standard platforms and can there be executed with a satisfying performance.

In the following sections such a model (Global Cellular Automata) will be described, and how it can be implemented and used. This model was introduced in [1], then further investigated, implemented, and applied to different problems. This paper is based on the results of former publications, mainly [1] [2] [3] [4] [5] [6] [7] [8] [9] [10].
2 The GCA Model (Global Cellular Automata)

The definition of the GCA model was inspired by the CA (Cellular Automata) model. The CA model consists of an array of cells arranged in an $n$-dimensional grid. Each cell (also called the “Center Cell” is connected to its local neighbors belonging to the neighborhood, e.g. to North, East, South, West. The next state of the center cell is defined by a local rule $f$ residing in each cell: $C \leftarrow f(C, N, E, S, W)$. All cells are applying the same rule synchronously and thereby a new generation of cell states is defined. As a cell changes only its own state, no write conflicts can occur which makes the model simple and elegant. Many applications with a local neighborhood can nicely be described as a CA, and CAs can easily be simulated or implemented in hardware.

The idea for the GCA model was (1) to retain the property that a cell can only modify its own state, and (2) to introduce more flexibility. Flexibility was obtained by using (2a) computed dynamic links to the neighbors and (2b) by allowing any cell in the array to be a neighbor (global neighbors). Thus a GCA can informally be described as follows:

A GCA consists of an indexed set of cells (e.g. an $n$-dimensional array). The cells’ states are updated synchronously according to a local rule. Each cell has $k$ global neighbors which can dynamically be changed by the local rule (Fig. 1). Write conflicts cannot occur, therefore the model can easily be supported by hardware for a large number of cells. A GCA is initialized by an initial state for each cell (initial configuration $CFG(t=0)$). The result of the computation is the state of the final configuration (all cell states) at time-step $t_{\text{final}}$. We can also speak of a “GCA algorithm”, meaning the transformation of the initial generation to the final generation.

Three model variants are distinguished, the basic model, the general model and the condensed model. They are closely related to each other and can be transformed into each other. It depends on the application or the implementation which one will be preferred.

**Basic model.** The cell state is a composition $(data, pointer) = (d, p)$ (Fig. 2a). The pointer $p$ is used to access the global neighbor $p = k^*$. The remote state $(d^*, p^*)$ is read from the global cell via the dynamic link. Then the new state components are computed: $d' = e(d, p, d^*, p^*)$ and $p' = g(d, p, d^*, p^*)$. Then

![Fig. 1](image-url) (a) In generation $t$ each cell is connected to $k$ neighbors (e.g. $k = 3$), and it selects $k$ new neighbors. (b) In generation $t + 1$ each cell is connected to its new neighbors.