High performance reconfigurable computing with cellular automata

Murtaza, S.

Publication date
2010

Chapter 1

Introduction

The quest to know everything about everything embarks us upon an infinite journey, one, very often dictated by the technological developments of time. With human evolution, our abilities to push the limits of the unknown have also developed, and more recently been fueled by our capability to compute increasingly faster. It may not be incorrect to say that our ability to know more now heavily relies upon innovation in computing.

1.1 The Road to Parallel Computing

Computers, traditionally based on single thread sequential computation model, also called the von Neumann architecture [114], have evolved from the 50's vacuum tubes based technology [110], to the current state of the art CMOS based nanoscale devices. Since the breakthrough in semiconductor devices, the processor technology has ridden the wave called Moore's law. More recently this wave has hit three walls – memory, power and instruction level parallelism – and has thus defined the limits of the sequential computing paradigm. Simultaneously, the boom in internet and its applications, accompanied by the need to understand and solve problems of global impact, demand unprecedented computing infrastructure and resources. To meet these challenges, the industry has been compelled to improve computing through massively parallel computing paradigms.

1.1.1 Cellular Automata and Parallel Computing

Cellular automata (CA), an inherently parallel computing paradigm, also proposed by von Neumann in 1948 [59], has been successfully applied to a range of computational problems to model the behaviour of complex systems from nature. John Conway's Game of Life [109], Cellular Automata Machines by Toffoli and Margolus [102], and the more recent book by Stephen Wolfram, A New Kind of Science [116], are some of the works synonymous to CA. It is not surprising to find why some of the computing pioneers are

fascinated and intrigued by the capabilities of a set of simple cells, arranged on a regular grid, with local connectivity reproducing complex structures from nature like fluid flow.

Long before Moore's law hit the three walls [5], parallel computing paradigms in general have been employed in high performance computing (HPC) [5, 111]. In HPC, the improvement of computations both in terms of speed and accuracy are of prime importance. In general, one of the ways to improve computations is to understand the process of mapping an application to the available hardware. More recently, reconfigurable devices like Field Programmable Gate Arrays (FPGA) have been integrated into HPC systems for application acceleration. Such systems provide grounds where the application can benefit from exploiting the possibilities of using both fine-grain and coarse-grain levels of computations.

The recent emergence of multicore architectures and the shift towards multicore computing, may have triggered the evolution of von Neumann architecture towards a parallel processing paradigm. Today some even consider the possibilities of migrating to different computing approaches like, CA [120]. This more recent switch from a single threaded sequential paradigm to a massively parallel computing paradigm [5, 53] makes CA and its related work more relevant through the breadth of computing. Based on advances in the CMOS technology, chip companies are foreseeing to manufacture hundreds of cores on a single die [53]. However, simply increasing the number of cores is not of much use if we are unable to scale applications to the available cores [58]. So it is no surprise that computing, in general, is seeking solutions that go beyond conventional paradigms based on von Neumann architecture [1, 5]. Based on this framework we attempt to understand, *what can be achieved by implementing massively parallel computing paradigms like CA, using available reconfigurable devices like FPGAs?*

1.2 Research Motivation

The three building blocks of this work are, HPC in general, stencil based computations like CA in particular, and reconfigurable devices like FPGA. Based on this criteria, the challenges are a) to demonstrate performance gain with running HPC applications like fluid simulations, b) to understand and classify CA based computations to exploit computing resources efficiently, and c) to tailor hardware resources like FPGA as per the application requirements. In other words dissecting the system implementation right through the top-level representation to the low-level implementation details to analyse and understand the overall system performance. Simultaneously, the recent shift towards multicore computing demand such dissections to understand and explore the full potential of multicore

computing. This need to look for new computing approaches to meet the demands of future applications led us to research the following questions:

- What can be achieved by implementing massively parallel computing paradigms like CA, using available reconfigurable devices like FPGAs?
- How to map an application from an abstract level to the underlying bit-level implementations in hardware?
- Is hardware reconfigurability a requirement to harness the benefits of the multi-billion transistor era and the overall system speedup?
- Is CA the potential candidate among the parallel processing alternatives in future?

Addressing our research aims requires an understanding of: how to map CA based high performance applications to reconfigurable device enabled systems. This exercise includes a) formulating a model to predict the behaviour and the performance of the overall system design without going into details of the implementation technology, b) based on the computational structure of the application, exploiting and customising the available hardware both at fine-grained and coarse-grained levels for overall performance gain, c) designing a system that can be scaled from a single to a multiple FPGA based implementations and therefore a capability to simulate large simulation sizes, and d) a custom hardware implementation for a CA algorithm using FPGAs. Based on this scientific endeavour we logically formulate the following objectives:

- Formulating a model to predict the behaviour and the performance of the overall CA system implementation using special purpose hardware.
- Develop a detailed system implementation ranging from a single to a multiple FPGA based system to verify our model.
- Analyse our experimental work within the framework of manycore literature review and draw parallels between the two.

1.3 Thesis Roadmap

Chapter two focuses on the research background, presenting concepts and technologies that form the basis of this work. It starts with an overview of CA with some example algorithms. LBM viewed as a generalised CA is also presented. The chapter moves on to the technological side introducing reconfigurable computing and FPGA in particular. It finally concludes with a section devoted to HPC using FPGAs and related works.

Chapter three introduces performance modeling for FPGA-based CA implementation. It presents a simple and a scalable CA computation method along with the performance model. Based on this model, CA algorithms are categorised as compute or IO-bound

computations. Following this categorisation, various CA algorithms are implemented and presented in the following chapters.

Chapter four presents IO-bound computations, like the Game of life and HPP model, and their implementations. It also discusses in detail, the computation strategy based upon pipelining with internal buffers and the optimisation of model parameters for the efficient mapping to application algorithms on to the hardware logic. The chapter concludes with test case results and conclusions.

In chapter five, the thesis enters its core with focus on high performance floating point based computations like LBM. Categorised as compute bound computations, a single FPGA-based implementation along with details of the architecture is presented. The internals of the processing elements implementing the CA collision function and the propagation mechanism using external memory banks are also presented. This single FPGA-based model sets the stage for the following chapters where it is scaled to multiple FPGA-enabled systems. The chapter concludes with test case results and the speedup achieved in comparison to a pure software implementation.

One of the main focuses of this project has been the scalability, that is, to design a system that can be scaled from a single to a multiple FPGA-enabled PC implementation and therefore, a capability to simulate large simulation sizes. Chapter six moves on from single to a multiple FPGA-enabled PC implementation and explains the required modifications brought about in the software and hardware. Application domain decomposition, boundary exchange across multiple FPGAs for every iteration computation, and latency hiding comes into the scene. How efficiently application implementation is able to perform boundary exchange and exploit latency hiding techniques is discussed along with its limits. In conclusion, limitations of a multiple FPGA-enabled PC and results based on the dual FPGA-enabled PC implementation are presented.

Chapter seven presents an FPGA-based LBM implementation scaled to a highly parallel FPGA setup. This chapter demonstrates scalability as one of the main strengths of the mapping strategy. A single FPGA-based implementation is scaled to a 64-FPGA supercomputer with a capability of computing lattice sizes approaching quarter of a billion cells (214.74 million). The chapter concludes with results.

With all the technical details and the implementation models presented in the previous chapters, chapter eight adopts a pencil and a paper approach to predict the performance of desirable system sizes and configurations. The chapter presents how the three dimensional version of LBM would perform based on the multi FPGA-enabled cluster implementation. Based on the implementation demonstrated in previous chapters and the prediction of three dimensional model, the chapter draws some conclusions on using special purpose devices like FPGA for acceleration in HPC.

Mapping an inherently parallel application like CA to a multiple FPGA-enabled system with each FPGA implementing multiple independent processing elements, demonstrates a massively parallel implementation both in hardware and software. Such example implementations [10, 11] have been considered analogous to the manycore system implementation of the future applications. Chapter nine reviews the current state of manycore computing paradigm and its evolution in the future. Based on the current technological push towards manycore paradigm and our experience of implementing multiple FPGA-enabled parallel system, we draw some conclusions about the expectations and the requirements of mapping a CA application to the manycore chips of the future. The chapter concludes with some of the major challenges facing the computing industry and the future impacts.