High performance reconfigurable computing with cellular automata

Murtaza, S.

Publication date
2010

Chapter 10

Summary and Conclusions

Cellular automata (CA) are inherently decentralised spatially extended systems that are capable of modeling the behaviour of complex systems from nature with a high degree of efficiency and robustness. Field Programmable Gate Arrays (FPGA) are inherently parallel commodity chips that are an ideal platform to investigate the hardware realisation of CA systems. More recently, reconfigurable devices like FPGAs have been integrated into high performance computing (HPC) systems for direct hardware execution of computationally intensive algorithms and thus an opportunity to exploit extreme performance potential.

FPGA based CA accelerators are not new to the scientific community, however mapping CA systems from an abstract concept to the hardware logic remains a major challenge. The availability of more advanced FPGAs and the lack of a problem solving environment targeting such systems, makes it necessary to get a better understanding of the underlying mapping process.

FPGA implementations are still compared to yesteryears writing of computer programs in assembly code, which by no means are as simple and straight forward like edit-compile-run-debug cycle of conventional programming. And to make things tough, system design and implementation are carried out using number of independent tools with majority of them being proprietary. In general, with FPGA based implementation, one gets lost with the minute details of the implementation and unable to see the big picture. Finally in the end it's the performance of the overall system and not the fine technical details that count. Though a lot of research is being carried out to develop tools to target FPGAs via conventional high level programming languages, see for example [82] and [24], but as of now there have been no clear breakthroughs. See [105] for an overview of the latest developments with high level language tools targeting FPGAs. Availability of a problem solving environment with pre-cooked engines, glued together at the run time via high level programming constructs, is highly desirable.

Hence the main focus of this work has been to offer a disciplined look into the issues and requirements of mapping CA algorithms to the physical realities of special purpose hardware. There are many trade-offs to consider when mapping such systems that include both the available FPGA resources and the CA algorithm's execution time. The most important aspect is to fully understand the behaviour of the specified CA algorithm in terms of its execution times which are either compute bound or I/O bound. In this work, the strengths and weaknesses of various hardware architectures employed to achieve the speedups ranging from single to a multiple FPGA based implementations are discussed in detail. This includes performance modeling for each of the implementations and determining the optimal values for the various key parameters that control the overall system implementation.

Performance modeling is a highly popular and commonly used technique in HPC [46]. In contrast, there is dearth of published work that demonstrates the application of performance modeling techniques in HPC using special purpose hardware. Instead designers seem to rely on the use of subset or incremental implementations of their application domain as a benchmark for the overall performance. Therefore, if the overall system implementation is going to be a productive process, is determined by these preliminary results. This methodology is by no mean foolproof with the reason being, the lack of understanding of the overall system and its behaviour.

In this thesis we demonstrate how the traditional technique of performance modeling can be applied to application implementations using FPGAs. This methodology provides a tool to efficiently analyse the proposed system in its entirety for the optimal implementations. Starting with single FPGA based CA implementations, a generic model is presented (Chapter 3). One of the key features of this model is its scalability to accommodate huge CA sizes. Based on the generic model, a methodology is presented to categorise a specified CA algorithm as compute bound or I/O bound. The model is validated for both I/O bound (Chapter 4) and compute bound (Chapter 5) two-dimensional CA algorithms ranging from a bit-state based Game of Life to a floating point based Lattice Boltzmann method. It is demonstrated how this methodology helps to predict the performance of running CA algorithms on specific FPGA hardware and how to determine optimal values for the various parameters that control the mapping process. We find that our model predictions are accurate within 7%. For a single FPGA based D2Q9 Lattice Boltzmann method implementation, an overall speedup of 2.3 is achieved as compared to a general purpose FORTRAN implementation.

As per our performance model, as long as the implementation stays compute bound, porting a D2Q9 Lattice Boltzmann method implementation to a multiple FPGA based implementation was feasible. Consequently, progressing from a single to a dual FPGA

based CA implementation; dual FPGA based implementation (Chapter 6) achieved a speedup close to 1.8 as compared to a single FPGA based implementation.

Progressing further, a multiple FPGA enabled PC cluster implementations are presented (Chapter 7). For a floating-point based CA implementation, parallised and distributed over Maxwell – a 64 FPGA Supercomputer, demonstrated a full parallel system implementation with potentially hundreds of processing elements computing a CA lattice. For a 2-million cells 2D LBM accelerator with periodic boundary conditions, performance model showed how the FPGA enabled PC cluster is the preferred multiple FPGA organisation over the multiple FPGA based PC setup. Latency hiding as the premise through out our multiple FPGA based systems, was fully exploited for our PC cluster based system.

Hence, we presented the application of performance modeling to HPC using special purpose hardware. This demonstrated a cost model for the computation and communication time of the overall FPGA based system implementation. Plus we successfully validated our models with FPGA based implementations that ranged from a single to multiple FPGA based computations. With this we achieved the successful completion of our first two objectives as presented in chapter 1.

Finally, we analysed our experimental work within the framework of the manycore literature review, also specified as our third objective in chapter 1. This work contributes to the novel idea of investigating CAs as a potential parallel processing paradigm on multicore architectures. The emergence of multicore architectures and the era of rolling out hundreds of cores per die sometime in near future, might have triggered the evolution of von Neumann architectures towards a parallel processing paradigm. The capability to have hundreds of cores per die is exciting, however, how optimally we are able to utilise such a resource remains a challenge. Again, the challenge is to best map an application to the underlying manycore architecture in order to identify the various key parameters that control the overall system performance.

In chapter 9, a literature review is presented to understand the current status with manycore systems. Parallelism, uniform structure, processor hungry supercomputing application, reconfigurable processing elements and interconnects, and scalability are some of the common elements that stood out when drawing parallels between the manycore review and our FPGA based CA implementations work. Based on these findings we have summarised the various types of manycore chip systems especially in terms of their processing units, memory, and interconnects that are to be expected in the future. For all the various listed desirable features of the machine in future, one thing that is common to most of the elements is the flexibility, not only in terms of software but hardware as well. The processor internal structure, memory layout and the access pattern or the interconnect network, all elements demand to be quite flexible and customisable as per the

application requirements. Without doubt, the ability to configure hardware is going to be an integral part of the manycore systems but in what proportion, that would only evolve with time. Reconfigurable devices like FPGAs making rapid progress with doubling of capacity every 18 months, is narrowing down the concept of having a processor and the memory as separate entities. These trends suggest FPGA devices playing a significant role in the computing device of the future. The review concludes with a set of open questions, ranging from the micro to the macro level of manycore systems and the evolution of manycore processing, that demand significant research and experimentation. As the hunt for answers continues, for future work, we believe investigating the following is of great interest

- Investigate a chip with hundreds of simple and uniform cores, each core flexible enough to perform a function either on its own or as a part of a larger unit, that is, aggregation of cores. This would require cores to be configurable to function as a processing unit, or a memory unit, or as an interconnect conduit.
- The inherent parallelism – a common denominator among CA systems, multicore architectures, and the FPGA chips – demands a need to investigate and explore the possibility of a future computing platform that might be somewhere along the road, where these three independent concepts intersect.