Design and Implement DARPAR Integer Sorting Benchmark on the Reconfigurable Computer with High Level Synthesis

Gang Quan Allen Michalski Duncan A. Buell James P. Davisand Sreesa Akella
Department of Computer Science & Engineering
University of South Carolina
Columbia, SC 29208, USA
{gquan,michalsk,buell,jimdavis,akella@cse.nd.edu}

Abstract

Although reconfigurable computers can potentially offer several orders of speedup for applications, high level synthesis techniques must be applied during the design process to ensure this potential be effectively explored. While there have been many high level synthesis techniques proposed in the literature, not much empirical work has been reported by designing practical applications on the practical platforms. In this paper, we study the problem of designing and implementing the DARPA integer sorting benchmark on the commercial high performance reconfigurable computer. Based on our empirical results and hands-on experience, we evaluate the effectiveness and discuss the deficiency of existing high level synthesis techniques.

1 Introduction

Reconfigurable computers (RC) are the customizable computing machines built on the reconfigurable hardware, such as FPGA. RC platforms combine the speed advantage from the custom ASIC design and the flexibility from the general computer and, therefore, have attracted much attention from the electronic design automation research community. Due to its high potential in the high performance computing, many commercial RC platforms such as SRC [2], SBS [1], have been developed.

The high level synthesis (HLS) [6] technology and RC platform are perfectly matched in designing high performance computing systems as high level synthesis provides the fast design cycle and the synthesized results can be rapidly implemented in FPGAs. To fully take the advantage of computing capability of RC platforms, HLS tools and methods must be incorporated into the application designs. Some previous work has shown that, for certain applications [3], the significant speedup can be achieved by implementing these applications alone in reconfigurable hardware. However, such a strategy is not always possible due to the limited FPGA resources, especially for large scale or complex applications. In addition, other architecture related constraints must also be accounted for in the design phases to develop a feasible and efficient solution for the system. On the other hand, by implementing and verifying the synthesized design with the RC platform, the deviation of the theoretical model with the practical scenarios can be readily identified, which will further improve the synthesis results.

In this paper, we study the problem of designing and implementing an application, *i.e*, the high bit-width integer sorting (DARPA benchmark four), on the commercial RC platform, i.e., SRC-6e computer, aided by the HLS technology. HLS consists of a number of highly coupled sub-problems, such as allocation, partitioning, scheduling, and mapping. While there have been extensive techniques proposed for each sub-problems, these techniques must be incorporated under a systematic framework such that these sub problems can be solved in a cooperative manner and, thus, the system performance can be optimized. Moreover, when designing a practical application on commercial RC platform, architecture issues and implementation details, such as memory organization, data path and data movement cost, slice usage, clock rate, signal wiring delay, which may not be formulated in the theoretical models must be taken into considerations to produce an optimized and feasible solution. These distinguish our work from others. Finally, through our work, we evaluate the effectiveness of the HLS techniques proposed in the recent research, and discuss the deficiency in these techniques.

2 The application and RC platform

The application adopted in this paper comes from one of the benchmarks which recently released by DARPAR to measure the capability of high performance computing systems [4]. The fourth DARPAR benchmark, i.e.*integer sort*, requires to sort a stream of N ($N \sim 5 \times 10^7$) unsigned 64 bit integers. Sorting a data set in such a large scale simply cannot be performed by pouring all the data in and out of the reconfigurable hardware once and for all. HLS techniques are desired in developing effective designs for this problem.

The RC platform used in this paper is the SRC-6e RC platform [2] that combines the flexibility of traditional microprocessors with the computing power FPGA. It consists of two processor boards and one Multi-Adaptive Processor (MAP) board. MAP board has two user logic FPGAs (Virtex II XC2V 6000TM). Processor boards are connected to MAP board through SNAP cards with 800MB/s transfer data rate. SNAP card plugs into DIMM Slot on microprocessor motherboard and provides interconnect of MAP to microprocessor. In the SRC-6e RC platform, the computation can be partitioned as: *the software component*, i.e., using instructions of microprocessors, or *the hardware component*, i.e., using capabilities of modern FPGAs. Different hardware processing elements (PEs) can be realized for the required hardware computation.

3 General Approach

Our general approach can be largely divided into following three phases: *parameter collection, trade-off analysis*, and *design implementation*.

During the *parameter collection* phase, the characteristics of the basic PE developed on the reconfigurable hardware need to be identified. At this time, the research team has successfully developed two types of principle PEs based on the classic pop sort and merge sort. A thorough analysis for these PEs regarding to their computation efficiency and resource usage is critical for the later design phases. In addition, the parameters for the targeted architecture such as the communication cost and configuration overhead must also be captured and collected for the same purpose. The *trade-off analysis* involves formulating our design as an HLS problem and performing the corresponding theoretical trade-off analysis, based on the information collected in the first phase. Techniques for different aspects of HLS [5, 7, 9, 10] are combined together to generate the theoretically optimized solution for our problem. Finally, in *design implement* phase, the HLS result is implemented on the practical RC platform and compared with those by other approaches, such as the *pure* software implementation.

4 Conclusions

In this paper, we propose to study the problem of designing and implementing the real application on the commercial high performance RC platform, based on the HLS techniques. At the present, our implementation has not completed yet. Through our study, we expect to demonstrate that the RC platform, combined with HLS techniques, can effectively achieve the projected goal of such platform, i.e, several orders of speedup, compared with the software implementation alone. We also expect to evaluate the effectiveness and identifying the deficiency of existing high level synthesis techniques based on our hands-on experience and empirical results.

References

[1] *Star Bridge High Performance Computer, http://www.starbridgesystems.com*. Star Bridge, 2002.

[2] *SRC Computers,http://www.srccomp.com/Products.htm*. SRC, 2003.

[3] N. Bergmann and J. Mudge. Comparing the performance of fpga-based custom computers with general-purpose computers for dsp applications. In *Proceedings of FCCM*, pages 165–171, 1994.

[4] DARPA. *High Productivity Computing Systems Discrete Mathematics Benchmarks*. DARPA, 2003.

[5] A. Doboli and R. Vemuri. Integrated high-level synthesis and power-net routing for digital design under switching noise constraints. In *Design Automation Conference*, pages 629–634, 2001.

[6] D. Gajski, N. Dutt, C. Wu, and Y. Lin. *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, Boston, MA, 1991.

[7] S. Gupta, T. Kam, M. Kishinevsky, S. Rotem, N. Savoiu, N. Dutt, R. Gupta, and A. Nicolau. Coordinated transformations for high-level synthesis of high performance microprocessor blocks. In *DAC*, 2002.

[8] J. Henkel and R. Ernst. A hardware/software partitioner using a dynamically determined granularity. In *Proceedings of DAC*, pages 691–696, 1997.

[9] G. Quan, X. Hu, and G. Greenwood. Performance-driven hierarchical hardware/software partitioning. In *Proceedings of ICCD*, pages 652–657, 1999.

[10] R.P.Dick and J.K.Jha. Mogac:a multiobjective genetic algorithm for the co-synthesis of hardware-software embedded systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 17(10):920–935, October 1998.