Fast Boolean Calculations Using the GPU

Bernd Steinbach and Matthias Werner

Freiberg University of Mining and Technology, Institute of Computer Science,
D-09596 Freiberg, Germany

Abstract. The growing number of Boolean variables requires very efficient approaches to solve the given tasks. We explore the utilization of the GPU for fast parallel Boolean calculations in this paper. Hundreds of processor cores of the GPU offer a significant potential for improvements. Constraints in their application may restrict the reachable speedup. This paper summarizes alternative approaches for utilizing the GPU in the Boolean domain and improvements of several orders of magnitudes.

1 Introduction

The technological progress in micro- and nano-electronics leads to both a strong extension of applications and growing requirements for the design of digital systems. Boolean functions are the main instrument for their description.

An important source for improvements to solve Boolean tasks is the utilization of many computer cores for parallel computations. Today's processors contain a small number of cores in the *Central Processing Unit* (CPU) which can be used by means of the *Message Passing Interface* (MPI) based on the paradigm *Single Program Multiple Data* (SPMD). Significantly more cores are available on the *Graphics Processing Unit* (GPU). The main paradigm for the GPU is *Single Instruction Multiple Data* (SIMD).

There are several basic approaches for utilizing the GPU in the Boolean domain. First, we can distinguish between parallel algorithms for special applications and general parallel algorithms for a wide field of Boolean tasks. Second, we can select between the *Compute Unified Device Architecture* (CUDA) to utilize all features of Nvidia GPUs or OpenCL for a wide field of vendors. We studied all these approaches and present our practical results in this paper.

2 Solving the Unate Covering Problem Using CUDA

The unate covering problem [1] is an important special problem in the Boolean domain. It must be solved to find minimal sets of prime conjunctions which completely cover the required Boolean function of a circuit. The architecture of the GPU is built for very efficient execution of matrix multiplication. In [2] the unate covering problem was mapped to a matrix multiplication and executed on the GPU using CUDA. Unfortunately this approach reached in the best case only a 3.5 times shorter calculation time using the GPU in comparison to the

Central Processing Unit (CPU). The reachable speedup does not only depend on the number of used processor cores, but also on the implemented algorithm. We developed an algorithm [5] that reaches an speedup of $8 * 10^8$ using a single CPU. As result of a very deep analysis we implemented the modified algorithm of ordered restricted vector evaluation [4, 8] which reached on GPU Tesla C2070 using CUDA an speedup of $1.2 * 10^{11}$.

3 Solving the Unate Covering Problem Using OpenCL

The found powerful algorithm of [4, 8] was implemented and evaluated in [3] using OpenCL. The strong benefit of this OpenCL implementation is that it can be used for different GPUs and even for multi-core CPUs. A more efficient data management lead for small benchmarks to shorter run-time in comparison to the CUDA implementation of [4, 8]. However, for the largest benchmark of 32 Boolean variables and 1024 clauses in the unate covering problem the OpenCL implementation needs approximately twice the time of the CUDA implementation using the same GPU Tesla C2070.

4 Implementation of XBOOLE Using CUDA

XBOOLE is a library of more than 100 functions which can be used to solve a wide field of Boolean problems [6, 7]. All time-consuming operations of XBOOLE were implemented in [9] in the compatible library XBOOLE-CUDA using CUDA. In this way all recent and future application benefit from the reached speedup of XBOOLE-CUDA. The reached speedup depends strongly on the executed operation and the size of the data. In best case a speedup of more than $13 * 10^3$ was realized using a special brute force algorithm.

5 Conclusions

The large number of cores of a GPU is an important source to reduce the time for hard Boolean problems. There are two ways for utilizing the GPU:

1. the direct implementation using a given API;
2. utilizing a domain specific basic software, like XBOOLE.

The first way provides unrestricted possibilities utilizing the properties of the GPU, but requires a strong effort for the implementation. We showed that the results of this way significantly depends on the selected algorithm. In case of the unate covering problem we get the large range of improvements from 3.5 to more than 10^{11}. There are two alternative APIs for direct implementation: CUDA and OpenCL. From our experiments we get the expected results: the special API CUDA for Nvidia GPUs found a special solution approximately two times faster than the more general API OpenCL for several GPU types.

The second way allows us to utilize the power of the GPU without exploring all details of GPU-programming. A much simpler program that uses XBOOLE on a single CPU reduces the speedup for the Petrick function of 32 variables and 1024 clauses from $1.2 * 10^{11}$ of the special CUDA implementation to $3.2 * 10^9$. The substitution of the XBOOLE library by XBOOLE-CUDA library improves this speedup to $1.2 * 10^{10}$. Hence, the very quick and simple implementation using XBOOLE-CUDA allows us to utilize the power of the GPU without the huge effort of CUDA programming only losing a small speedup factor. The improvement of XBOOLE-CUDA in comparison to XBOOLE depends on the used operations and the sizes of the data (TVL); the typical speedup is in the range from 10 to 100.

References

1. Cordone, R., Ferrandi, F., Sciuto, D. and Wolfler Calvo, R. *An Efficient Heuristic Approach to Solve the Unate Covering Problem.* Proceedings of the Conference on Design, Automation and Test in Europe, Paris, France, 2000, pp. 364–371.
2. Paul, E., Steinbach, B. and Perkowski, M.: *Application of CUDA in the Boolean Domain for the Unate Covering Problem.* Boolean Problems, Proceedings of the 9th International Workshops on Boolean Problems, Freiberg, 2010, pp. 133–142.
3. Grehl, S.: *Vergleich von Implementierungen des "Unate Covering Problems" mit OpenCL und CUDA*, Project-Thesis, Freiberg University of Mining and Technology, 2013.
4. Steinbach, B. and Posthoff, Ch.: *Fast Calculation of Exact Minimal Unate Coverings on Both the CPU and the GPU.* in: Roberto Moreno-Díaz, Franz Pichler und Alexis Quesada-Arencibia: Computer Aided Systems Theory - EUROCAST 2013, 14th International Conference, Las Palmas de Gran Canaria, Spain, February 2013, Revised Selected Papers, Part II, Lecture Notes in Computer Science Volume 6928, Springer, ISBN: 978-1-612-08292-9, DOI: 10.1007/978-3-642-53862-9_30, 2013, pp. 234–241.
5. Steinbach, B. and Posthoff, Ch.: *Improvements of the Construction of Exact Minimal Covers of Boolean Functions.* in: Roberto Moreno-Díaz, Franz Pichler und Alexis Quesada-Arencibia: Computer Aided Systems Theory - EUROCAST 2011, 13th International Conference, Las Palmas de Gran Canaria, Spain, February 6-11, 2011, Revised Selected Papers, Part II, Lecture Notes in Computer Science Volume 6928, Springer, ISBN: 978-3-642-27578-4, DOI: 10.1007/978-3-642-27579-1_35, 2012, pp. 272–279.
6. Posthoff, Ch., Steinbach, B.: Logic Functions and Equations - Binary Models for Computer Science. Springer, Dordrecht, The Netherlands, 2004.
7. Steinbach, B. and Posthoff, Ch. Logic Functions and Equations - Examples and Exercises. Springer Science + Business Media B.V., 2009.
8. Steinbach, B. and Posthoff, Ch.: *Sources and Obstacles for Parallelization - a Comprehensive Exploration of the Unate Covering Problem Using Both CPU and GPU* Astola, J.; Kameyama, M.; Lukac M.; Stankovi R. S. (Eds.): GPU Computing with Applications in Digital Logic. Tampere International Center for Signal Processing. TICSP series # 62, Tampere 2012, ISBN 978-952-15-2920-7, ISSN 1456-2774, pp. 63–96.
9. Werner, M.: *Parallelisierung von XBOOLE-Operationen mit CUDA*, Master-Thesis, Freiberg University of Mining and Technology, 2014.