An Adaptive Data Prefetching Scheme for Biosequence Database Search on Reconfigurable Platforms

Xiandong Meng
Department of Electrical and Computer Engineering
Wayne State University
5050 Anthony Wayne Dr., Detroit, MI 48202
(001) 313-577-6592
meng@ece.eng.wayne.edu

Vipin Chaudhary
Computer Science & Engineering
NYS Ctr. Of Excellence in Bioinformatics & Life Sc
University at Buffalo, SUNY
Buffalo, NY 14260
vipin@buffalo.edu

ABSTRACT
Searching on DNA and protein databases using sequence comparison algorithms has become one of the most powerful techniques to better understand the functionality of particular biological sequences. However, the requirements to process the biological data exceed the ability of general-purpose processor. The core of sequence alignment algorithm was implemented as fine-grained parallel architecture that was running on a commercial-off-the-shelf (COTS) FPGA board, where supercomputer performance has been achieved. However, reconfigurable computing platforms have utilized a PCI bus as the communications channel, limiting the communication speed between the host processor and the FPGA. This communication bottleneck often offsets the application speedup enabled by FPGA. In this paper we present an adaptive data prefetching scheme to avoid reconfigurable coprocessor stalls due to data unavailability through profiling techniques and quantitative analysis. Experimental results satisfied time constraints with various query sequences and show that we can effectively eliminate a major portion of data access penalty.

Categories and Subject Descriptors
J.3 [Computer Application]: Life and Medical Sciences – Biology and genetics

General Terms
Algorithms, Design, Performance

Keywords
FPGA, DMA, data prefetching, Smith-Waterman algorithm

1. INTRODUCTION
The amount of biological sequences available in databases has been growing exponentially over the past several years. Biosequence database searches based on FPGA reconfigurable hardware platforms [4] have gained popularity recently and tremendous speedup has been reported as compared to a standard desktop PC. However, the data communication between the PCI-based FPGA board and the host system with the large volume of data involved in this kind of application limits the performance. We have developed an adaptive parallel data prefetching scheme for the execution of Smith-Waterman sequence database searches to alleviate the communication bottleneck thereby leading to substantial performance improvement.

A number of parallel designs [2,3,4] for performing sequence analysis have been developed. Solutions based on PCI-based FPGAs that are flexible to plug into a PC hold great potential to boost communication performance using our proposed adaptive data prefetching scheme.

SAC'07, March 11-15, 2007, Seoul, Korea.

2. DESIGN AND IMPLEMENTATION

2.1 Execution Profile Analysis
The total execution time of FPGA biosequence database search can de decomposed into three primary parts: database sequences loading time on the host machine, Smith-Waterman processing time on FPGA, and other time including FPGA initialization, data setting, communication latencies and printing result. In order to analyze how each part contributes to the total program execution time, we studied the timing profile by comparing various query sequences to a 1 GB of the nr protein database [5], which contains 936,896,903 characters in 2,739,534 sequences. The measurements were taken on a 1.9 GHz Pentium IV processor with 768 MB memory, and an ADP-WRC-II FPGA PCI-board with a Xilinx Virtex II XC2V6000 from [1]. The board contains 119 affine PEs and accepts the query sequence up to 1,420 characters. Based on measurements, we conclude that the database sequence loading time and other time are constant and independent of the query sequence size.

2.2 Data Prefetching Scheme
The data transfer requires low latency and high bandwidth to ensure that the communication does not affect the processing time. Biological sequence search on a PCI-based FPGA board requires database sequences to be stored in an application buffer of the host machine, and then transferred to the FPGA board for processing, because the amount of on-board memory is usually very limited. Loading sequence database, especially the large

volume of biological sequences, in memory of host machine causes significant performance drawbacks.

We designed and implemented a double buffering parallel implementation using DMA on FPGA board and Pthread on host machine. The idea of double buffering strategy is to transfer database sequences into one intermediate application buffer in the background, where the processing can be completed in the other application buffer utilizing the full FPGA power. Once processing is done in a local buffer, the FPGA coprocessor sends results back to the host and receives new sequences from the buffer, which has been filled by host machine, and starts computation immediately with the least idle time. Our implementation takes advantage of DMA transfers, which are performed by the PCI device for large blocks of data, between host CPU and FPGA board. Application software running on the host system can control the DMA engines for the rapid data transfer and processing to and from the FPGA using send/receive threads, at the same time, host software creates additional thread to load database sequences into application buffers to overlap computation with communication. Thus, the transformation overlaps the data communication time with the computation time in parallel loops to effectively hide the latency of the database sequence transfer time.

2.3 Adaptive Data Prefetching

The scheduling solution is able to resolve the unbalance between loading and processing time by providing more flexible prefetching mechanism, such as, the adaptive prefetching buffer size. Since the proportions of database loading and FPGA processing time keep changing with the varied query inputs, the choice of how to select a proper buffer size for data transfer could be critical to performance. A static or improper buffer size may decrease performance and bring additional communication overhead to the system. A data prefetching scheme would be ineffective unless a proper buffer size is selected.

In order to exploit the seamless transformation between loading and processing, we designed and implemented a query-based data prefetching adaptation algorithm based on the length of the query sequence. The idea is that the short query would require small buffers to overcome database loading delay, and the long query sequence would require large buffers. The communication overhead would be eliminated as much as possible when buffer size reaches a certain point.

3. SUMMARY AND CONCLUSIONS

The database buffer loading time by host machine should be fairly close to FPGA processing time. The prefetching buffer size required to compare a query sequence to a database of sequences on a single FPGA board is given as:

$$S_{number} = \lambda \times g(\text{ceil}(m / (N+1))) \times 1000$$

where
S is the prefetching buffer size in the number of database sequences,
λ is the coefficient of host machine speed,
m is the size of query sequence,
N is the number of PEs,
g(x) is a stair function for buffer size based on our experiments,

and is summarized as:

$$g(x) = \begin{cases} 170 & ,x = 1 \\ 180 & ,x = 2 \\ 190 & ,x = 3 \\ 200 & ,4 \le x \le 6 \\ 210 & ,7 \le x \le 12 \end{cases}$$

The worst performances were observed at the boundaries, i.e., buffer sizes of 10,000 and 250,000 respectively, due to additional overhead such as frequent switching between small buffers or filling out a very large buffer. However, these side effects are entirely eliminated by applying our adaptive scheme.

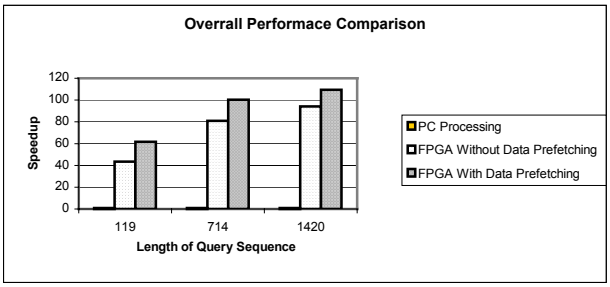

Figure 1. Speedups of Adaptive Data Prefetching

Figure 1 shows the performance gains with various query sequences searched against a 1 GB database. While using the adaptive data prefetching on FPGA, we achieved a 42% overall performance increase on a short query sequence of length 119 over without data perfecting; 21% and 16% improvement for query sizes of 714 and 1420, respectively. For performance comparison, the speedups were 62, 100, and 110 over the PC performance.

Techniques to reduce communication latencies become essential for achieving high FPGA utilization. We have implemented a software data prefetching scheme by exploiting the overlap between computation and communication. Our approach of dynamically determining the size of prefetched data has been shown to be very effective for reducing the communication latency.

4. ACKNOWLEDGMENTS

We thank Progeniq for providing the FPGA software.

5. REFERENCES

[1] Alpha-Data, http://www.alpha-data.com

[2] Meng, X. and Chaudhary, V. Bio-Sequence Analysis with Cradle's 3SoC Software Scalable System on Chip In Proceedings of the ACM SAC SAC'04, March 14-17, Nicosia, Cyprus, 2004

[3] Meng, X. and Chaudhary, V. Exploiting Multi-level Parallelism for Homology Search using General Purpose Processors, in proceedings of the ICPADS, Fukuoka, Japan, 20-22 July 2005

[4] Oliver, T., Schmidt, B. and Maskell, D. Hyper Customized Processors for Bio-Sequence Database Scaning on FPGAs, IEEE Transactions on Circuits and Systems II, Vol. 52, No. 12, pp. 851-855, 2005

[5] Progeniq Pte. Ltd., http://www.progeniq.com