Using FPGA-Based Hybrid Computers for Bioinformatics Applications

Seamless integration of FPGAs into grid computing infrastructures is key to the adoption of FPGAs by bioinformaticians.

Ravi Kiran Karanam
Research Assistant
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
rkkarana@uncc.edu

Arun Ravindran
Assistant Professor
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
aravindr@uncc.edu

Arindam Mukherjee
Assistant Professor
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
amukherj@uncc.edu

Cynthia Gibas
Associate Professor
Department of Computer Science
University of North Carolina at Charlotte
cgibas@uncc.edu

Anthony Barry Wilkinson
Professor
Department of Computer Science
University of North Carolina at Charlotte
abw@uncc.edu

The past decade has witnessed an explosive growth of data from fields in the biology domain, including genome sequencing and expression projects, proteomics, protein structure determination, and cellular regulatory mechanisms, as well as biomedical specialties that focus on the digitization and integration of patient information, test, and image records. Bioinformatics refers to the storage, analysis, and simulation of biological information and the prediction of experimental outcomes.

To address the computing and data management needs in bioinformatics, the traditional approach has been to use clusters of low-cost workstations capable of delivering gigaflops of computing power. However, microprocessors have general-purpose computing architectures, and are not necessarily well suited to deliver the teraflops of high-performance capability required for compute or data-intensive applications. The recent availability of off-the-shelf high-performance FPGAs – such as Xilinx® Virtex™-II Pro devices with on-board high capacity memory banks – has changed the computing paradigm by enabling high-speed processing and high-bandwidth memory access.

Nallatech offers multi-FPGA computing cards containing between one and seven Virtex FPGAs per card. Several such cards are plugged into the PCI slots of a desktop workstation and networked using Nallatech's DIMEtalk networking software, greatly increasing the available computing capability. This is our concept of a hybrid computing platform. The hybrid platform comprises several workstations in a cluster that you can integrate with Nallatech multi-FPGA cards to construct a high-performance computing system.

Grid Computing

Grid computing is a form of distributed computing that employs geographically distributed and interconnected computing sites for high-performance computing and resource sharing. It promotes the establishment of so-called virtual organizations – teams of people from different organizations working together on a common goal, sharing computing resources and possibly experiment equipment.

In recent years, grid computing has become increasingly popular for tackling difficult bioinformatics problems. The rise of "bio-grids" (such as the NIH Cancer Biomedical Informatics Grid and Swiss Biogrid) is driven by the increasingly enormous datasets and computational complexity of the algorithms involved. Computational grids allow researchers to develop a bioinformatics workflow locally and then use the grid to identify and execute tasks seamlessly across diverse computing platforms.

To integrate the computing power of FPGA-based hybrid computing platforms with the existing computing infrastructure used by the bioinformatics community, we have grid-enabled the hybrid platform using the open-source Globus Toolkit. Thus, the hybrid platform and the associated software are available as a grid resource so that bioinformatics applications can be run over the grid. The hybrid platform partitions the tasks to be run on processors and FPGAs and uses application program interfaces (APIs) to transfer data to and from the FPGAs for accelerated computation. Bioinformaticians can now take advantage of the computing power of our FPGA hybrid computing platform in a transparent fashion.

Hybrid Computing Platform

The different FPGA related components in the hybrid platform are:

- High-capacity motherboard – BenNUEY
  The Nallatech BenNUEY motherboard features a Xilinx Virtex-II Pro FPGA and module sites for as many as six additional FPGAs. The PCI/control and low-level drivers abstract the PCI interfacing, resulting in a simplified design process for designs/applications.
- Virtex-II expansion module – BenBlue-II
  The Nallatech BenBlue-II DIME-II module provides a substantial logic resource ideal for implementing applications that have a large number of processing elements. Through support for as many as two on-board Xilinx Virtex-II Pro FPGAs (XC2VP100 devices), the BenBlue-II can provide more than 200,000 logic cells on a single module.
- Multi-FPGA management – DIMEtalk
  To manage the large silicon resource pool provided by the hardware, the Nallatech DIMEtalk tool accelerates the design flow for creating a reconfigurable data network by providing a communications channel between FPGAs and the host user environment.
- FUSE Tcl/Tk control and C++ APIs
  Nallatech's FUSE is a reconfigurable operating system that allows flexible and scalable control of the FPGA network directly from applications using the C++ development API, which is complemented by a Tcl/Tk toolset for scripting base control.

DNA Microarray Design – A Case Study

Our goal was to accelerate the Smith-Waterman implementation in the EMBOSS suite of publicly available bioinformatics code. The Smith-Waterman algorithm is widely used to screen gene databases for sequence similarity, with many different applications in bioinformatics research. Smith-Waterman is specifically used in situations where faster heuristic methods fail to detect some potentially meaningful sequence hits.

Dr. Cynthia Gibas of the Bioinformatics Center at UNC Charlotte currently uses water.c, the Smith-Waterman implementation in the open-source EMBOSS software, as a part of a DNA microarray design work flow. The biology goal is to select the optimal probe sequences to be printed on a DNA microarray, which will then be used in the lab to detect individual gene transcripts in a target mixture with high specificity.

Hardware/Software Partitioning

The EMBOSS implementation of the Smith-Waterman (water.c) is an extensive C program comprising more than 400 functions. A partitioning strategy is required to identify the functions that need to be implemented on the FPGA and those that remain in software (and run on the processor). The partition is done by profil-

Each sample counts as 0.01 seconds.						
Percentage Time	Cumulative Time (sec)	Self Time (sec)	Function Calls	ms/call (self)	ms/call (self)	Name of Function
83.32	178.94	178.94	32768	5.46	5.65	embAlignPathCalcSW
5.12	189.94	11.00	32768	0.34	0.35	embAlignWalkSWMatrix
4.62	199.87	9.93	32768	0.30	0.30	embAlignScoreSWMatrix
2.92	206.13	6.26	2719612928	0.00	0.00	ajSeqCvtK
1.86	210.13	4.00	53936723	0.00	0.00	match

Table 1 – Profiling results of the EMBOSS Smith-Waterman implementation

ing the execution characteristics of the code. First, the legacy C code in the water.c file is profiled using the Linux gprof tool. Profiling tells us where a given code spends time during execution, as well as different functional dependencies.

Table 1 shows the results of profiling in terms of the execution times of the top five functions executed by the code, listed in decreasing order of execution time. Note that one function, embAlignPathCalcSW, accounts for 83% of the total amount of program execution time. The embAlignPathCalcSW function uses the Smith-Waterman-based local alignment algorithm to create a "path matrix" containing local alignment scores of comparing probe and database sequences at different matrix locations, and a compass variable to show which partial result is used to compute the score at a certain location.

Once the code profiling is done, the computationally intense embAlignPathCalcSW call is mapped to the FPGA network using VHDL, while the rest of the code is run on the processor. Calls to the computationally intense embAlignPathCalcSW function in the C code of the water.c file are then replaced with corresponding application program interface (API) calls to the FPGA network. These APIs transfer data between the FPGA network and the processor, such that the calculation of the scores in the path matrix is done inside the FPGA network. All other parts of the code, including backtracking, are executed on the processor in software. A flow diagram of these steps is shown in Figure 1.

Hardware Implementation

When comparing a target sequence (T) from a database with a probe sequence (P), the scores and the compass values of the path matrix in the embAlignPathCalcSW function are calculated using a systolic array of basic processing elements (PEs). Figure 2 shows the systolic array implementation of the path matrix algorithm for Smith-Waterman with three PEs. Each PE passes the score and compass values it calculates to the successive PE, which in turn uses these values to calculate its path and compass values. At each clock cycle the path and the compass values are stored in the block RAM. At the end of the computation, each block RAM has the corresponding row of the path matrix stored in it.

The output of the computationally intensive function involves huge amounts of data transfer (the order of probe length times the target length). As a result, the improvements achieved in computing performance are compromised by communication latency. To decrease communication latency, two additional functions (embAlignScoreCalcSW and embAlignWalkSWMatrix) were moved to the FPGA from software. These functions do the backtracking and calculate the score and alignment sequence for the given probe and target. The functions operate on the path matrix and calculate the maximum path value. Then, starting at the location of the maximum path value, the functions back-

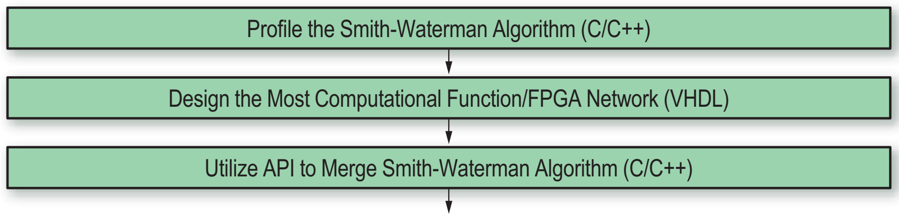
Profile the Smith-Waterman Algorithm (C/C++)
Design the Most Computational Function/FPGA Network (VHDL)
Utilize API to Merge Smith-Waterman Algorithm (C/C++)

Figure 1 – Design flow for accelerating water.c on the FPGA-processor hybrid computer

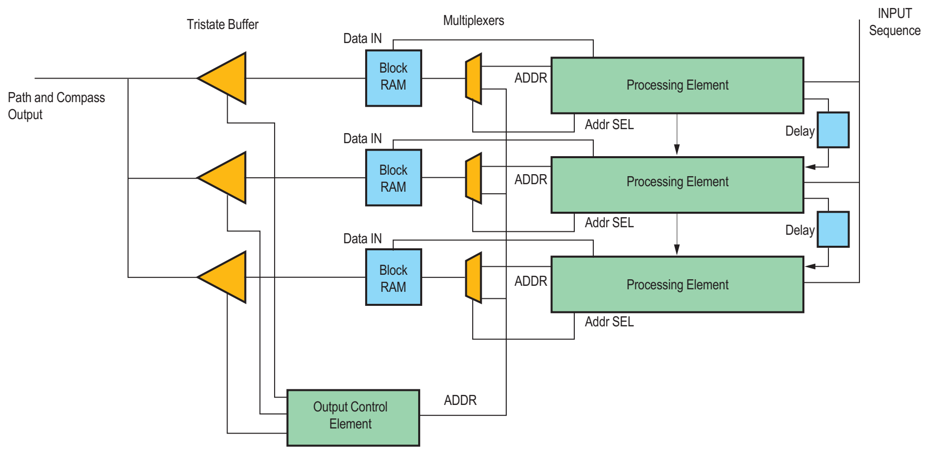

Figure 2 – Systolic array of three PEs for the path matrix calculation

track through the path values in the path matrix based on the compass values to determine the best alignment. The output of the FPGA is now the score and the alignment sequence, which is only about the size of the probe sequence, thus greatly reducing communication latency.

Grid-Enabling the Hybrid Computing Platform

The first step to grid-enable our resource was to install the components of the grid software utility package called Globus Toolkit 4.0 (GT4). The GT4 is an open-source reference implementation of key grid protocols. The toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability.

The core of GT4, the open grid services architecture (OGSA), is an integration of grid technologies and Web service technologies. Above the OGSA, the job submission and control is performed by the grid resource allocation and management (GRAM). GRAM provides a single interface for requesting and using remote system resources for the execution of "jobs." The Globus security interface enables the authentication and authorization of users and groups in the grid through the exchange of signed certificates. Certificates are created and maintained by a tool called SimpleCA in GT4.

Installation and Integration with the UNC-Charlotte VisualGrid

To test our installation in a real grid, we took advantage of a new grid project called VisualGrid, a collaborative project between UNC-Charlotte, UNC-Asheville, and the Environmental Protection Agency. The FPGA-based hybrid computing platform is added as a client to VisualGrid through the master CA server.

Results

We implemented a prototype acceleration task comparing a 40-nucleotide probe sequence with target database sizes of as many as 1,048,576 targets of an approximate length of 850 nucleotides both in software (processor) and an FPGA hybrid computing platform. The probe sequence and the target database reside in the main memory of the host computer (dual Opteron-based Sun Java W1100z workstation).

For each call to the embAlignPathCalcSW, embAlignScoreCalcSW, and embAlignWalkSWMatrix functions from water.c, a 32-bit probe and target combination is sent over the PCI interface to the PCI FIFO on the Nallatech BenNUEY motherboard at a rate of 33 MHz. From the PCI FIFO, the data is transferred to a 32-bit, 512-word-long input FIFO on the Virtex-II Pro FPGA of the BenNUEY motherboard. The systolic array reads the data from this FIFO. The 40-processing-element-deep systolic array operates at 40 MHz.

After the systolic array completes processing on a single string, the output values from the block RAM are used to calculate the alignment. The alignment results are then written back to a different block RAM. Finally, the host processor reads the output alignment string from the block RAM over the PCI interface.

The hybrid computing platform was accessed through the VisualGrid with jobs submitted through the UNC-Charlotte VisualGrid Portal. The EMBOSS water.c program ran in software on the workstation and separately on the FPGA-based hybrid computing platform. Figure 3 shows the comparison between the run times of the two implementations. For small databases, the processing times of the processor and the hybrid computing platforms are comparable. However, as databases get larger, the processor computing time rises exponentially, while the hybrid computing platform shows a linear increase. For a database size of 1,048,576 strings, the hybrid computing platform is 44 times faster than execution on a processor. Such large databases are common in bioinformatics applications.

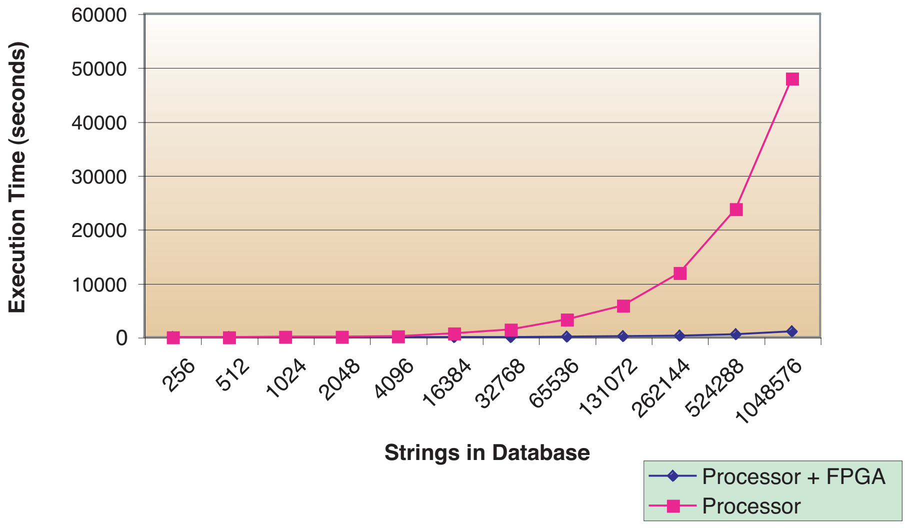

Figure 3 – Execution time for water.c for different sizes of sequence database strings running on a processor and a processor + FPGA hybrid system

Conclusion

We have demonstrated the computing potential of a grid-enabled hybrid computing platform for sequence-matching applications in bioinformatics. A careful partitioning of the computing task between the processor and the FPGAs on the hybrid platform circumvented the potential I/O bottleneck associated with large bioinformatics databases. The ability to submit jobs over the grids enables ready integration and use of the hybrid computing platforms in bioinformatics grids. We are currently involved in integrating the hybrid computing platform on the large inter-university SURAGrid (*www.sura.org/SURAgrid*).

This research was partly funded by NSF Award Number 0453916, and by Nallatech Ltd.