



Introduction

Modern high-throughput genome sequencing technologies have presented researchers with the challenge of assembling sequence matches from immense quantities of raw data. As the cost of sequencing falls, the volume of data is increasing. Add to the increasing volume of raw data to be analyzed, the fact that more and more genome sequences for known diseases are discovered, providing exponentially increasing volumes of sequences to be matched against the raw data.

Being able to match known disease-causing sequences against an individual's genome has the potential benefit of lowering health care costs by providing valuable information about an individual's susceptibility to certain diseases, and therefore taking preventable measures where appropriate. Accuracy is the key to an individual's optimum lifestyle and lowering the cost of healthcare.

Common sequence analysis algorithms, such as Smith-Waterman, are extremely demanding of time and computation resources when implemented using conventional computing resources. However the nature of such algorithms causes them to lend themselves easily to parallel computing solutions. The DRC Accelium Coprocessor can be operated in a cloud computing environment to provide massively parallel computing resources for sequence analysis.

The Smith-Waterman algorithm with affine gap model delivers the highest accuracy of any analysis algorithm, but generally requires so much computation resources to perform that throughput is compromised.

The DRC solution brings high-performance gene sequence analysis to standard Intel-based servers installed with DRC Accelium Coprocessors, running Microsoft Windows HPC Server 2008 R2.

Smith-Waterman

The Smith-Waterman algorithm is a well-known dynamic programming algorithm for determining whether similar regions exist in two DNA or protein sequences. When implemented with the affine gap model, it produces more accurate results than the BLAST algorithm.

The Smith-Waterman algorithm determines whether similar regions exist between the two DNA sequences by comparing sequences of all possible lengths and optimizing the similarity scores. By doing so, it is guaranteed to find optimal matches.

A score matrix is created that has the query sequence across the top and the reference sequence down the side. The table cells are filled by calculating a score that shows the quality of the match between the row and column characters. The best score in the table indicates the best match between the two sequences.

The value of the score in each cell is dependant upon the match between the two sequences and upon the score of each adjoining cell above and to the left. Filling the table becomes a matter of calculating the score for each cell in the score matrix.

Since the problem becomes one of calculating the score for each cell, it lends itself easily to parallel computing techniques. As the scores for each row or column are calculated, you can begin calculating the scores for the adjacent row or column in parallel.

In the example below, a query sequence of “TTCCGTAG” and a reference sequence of “CATTAGGC” are shown. The first row and column of the table are initialized to zero. The scores are then calculated started at the upper left corner and proceeding outward.

$$V(i,j) = \text{MAX}(0, E(i,j), F(i,j), V(i-1,j-1) + \text{SUBST}(S[i], T[j]))$$

$$E(i,j) = \text{MAX}(V(i,j-1) - \text{GAP_OPEN}, E(i,j-1) - \text{GAP_EXT})$$

$$F(i,j) = \text{MAX}(V(i-1,j) - \text{GAP_OPEN}, F(i-1,j) - \text{GAP_EXT})$$

	T	T	C	C	G	T	A	G
C	0	0	5	5	0	0	0	0
A	0	0	0	1	1	0	5	0
T	5	5	0	0	0	6	0	1
T	5	10	1	0	0	5	2	0
A	0	1	6	0	0	0	10	0
G	0	0	0	2	5	0	0	15
G	0	0	0	0	7	1	0	5
C	0	0	5	5	0	3	0	0

Smith-Waterman with affine gap model

DRC Implementation

The DRC implementation of the Smith-Waterman algorithm with affine gap model is optimized for the latest DRC Accelium coprocessors to provide the highest possible performance. Early DRC investigation showed that while the Smith-Waterman algorithm offers the potential of high performance, the algorithm with affine gap model is more useful for real-world analyses.

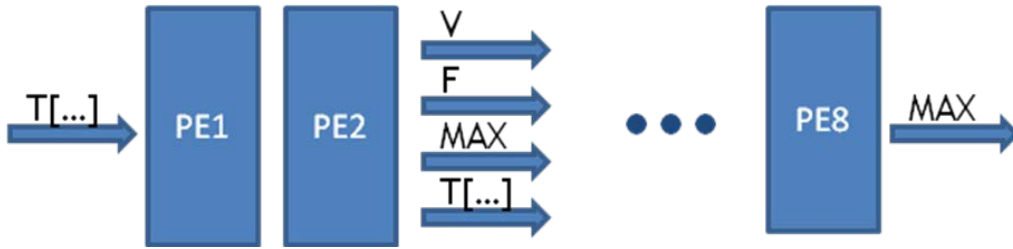
After making the decision to base the implementation on the Smith-Waterman algorithm with affine gap model, a design that used 1024 base-pair query strings was created to accelerate the SSEARCH35 tool. Performance evaluations showed that approximately 200 BCUPS was achieved in a single AC2030 Coprocessor versus approximately 4 BCUPS on a single CPU core. Additional investigation determined that the SSEARCH35 tool is not optimized for searching with multiple query strings in parallel.

Substantial performance improvements are the result of many factors including optimizing the query sequence to match the most common usages. Because common sequencers output short reads, typically between 25 and 100 base pairs, the query sequence length of 190 characters was chosen, allowing the design to be optimized for highest throughput. This choice allows the maximum number of processing elements to be created in each FPGA-based coprocessor, while providing room for growth as sequencer outputs become longer, thus maximizing the utilization of the FPGA. These design choices allow data paths and processing element arrays to be “right-sized” to match the query string length and match scores.

Processing elements are arranged in a systolic array such that each processing element works on two string characters at the same time. Each DRC AC3611 Coprocessor provides 6 arrays of processing elements and each DRC AC3621 provides 14 arrays. The processing elements operate at an internal clock rate of 200 MHz, and produce a result on each clock cycle.

	T	T	C	C	G	T	A	G
C	0	0	5	5	0	0	0	0
A	0	0	0	1	1	0	5	0
T	5	5	0	0	0	6	0	1
T	5	10	1	0	0	5	2	0
A	0	1	6	0	0	0	10	0
G	0	0	0	2	5	0	0	15
G	0	0	0	0	7	1	0	5
C	0	0	5	5	0	3	0	0
	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8

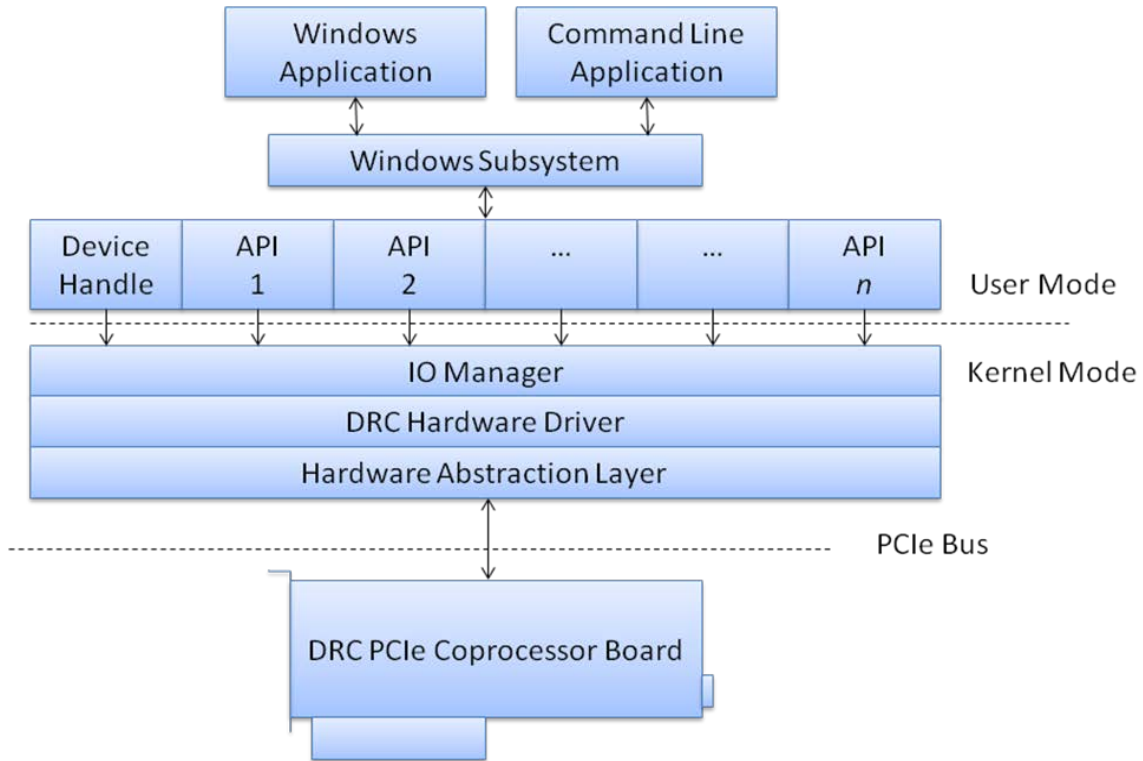
Each PE computes one column, so number of PEs = length of query string



E(i,j) depends only on V and E of the cell above, which are available locally

DRC’s Milano Integration Environment makes system integration easy under Linux or Windows operating systems. This modularity and ease of integration increases scalability, allowing a modest start and increasing performance as resources become available. It also achieves a high price / performance ratio because standard servers can be used.

The following diagram describes the software interface layers that connect the user application with the DRC Accelium PCIe Coprocessors. This interface provides easy integration of DRC’s high-performance coprocessors in the development environment.



Software Interface Layers

DRC Accelium PCIe Coprocessors are based on Altera™ Stratix™ EP4SGX530 (AC3621) and EP4SGX230 (AC3611) FPGAs for high performance and low power consumption (typically 25 watts). DRC Accelium Coprocessors provide an internal memory bandwidth in excess of 1 TB/s with DDR-3 64-bit wide memory support, and feature full PCIe Gen 2 support at 80 Gbps.

Benchmark Results

DRC's implementation of the Smith-Waterman algorithm on the massively parallel DRC processor achieved a performance of 9,400 billion cell updates per second (BCUPS) using synthetic data representing 13.4 million reads of 190 base pairs against a 958.6 million base pair reference (representing coverage of about 2.7 X).

On a single DRC Accelium AC3621 Coprocessor, DRC achieved 525 BCUPS.

The benchmark ran on standard servers incorporating multiple DRC Accelium AC3611 and AC3621 PCIe Coprocessors in a heterogeneous computing environment communicating through the MPI under Microsoft Windows HPC Server 2008 R2.

Conclusion

By bringing a new performance standard to genome sequence analysis, DRC has reduced the computing cost, power, and infrastructure requirements by a factor of 20.

The combination of high-performance on standard, cost-effective servers running Microsoft Windows HPC Server 2008 R2, improves the cost/performance ratio of performing complex DNA sequence analysis.

Because of the inherent parallelism of DRC's reconfigurable coprocessors, these solutions are extremely scalable and adaptable to modern cloud computing environments where computing resources can be shared across multiple users and applications.

Reconfigurable computing allows the same computation resources to be used for multiple applications, allowing system builders flexibility in design and economy through the elimination of duplicate resources.