Reconfigurable Supercomputing with Scalable Systolic Arrays and In-Stream Control for Wavefront Genomics Processing

C. Pascoe, A. Lawande, H. Lam, A. George
NSF Center for High-Performance Reconfigurable Computing (CHREC)
University of Florida

Y. Sun, W. Farmerie
Interdisciplinary Center for Biotechnology Research (ICBR)
University of Florida

M. Herbordt
Department of Electrical and Computer Engineering
Boston University

I. Introduction

Computational challenges in genomics data mining and analysis are an impending roadblock in modern health-sciences research, where algorithms for DNA sequence processing grow alarmingly in computational needs as datasets from new instruments continue to dramatically expand. Reconfigurable computers featuring customizable processing devices (e.g. FPGAs) hold the key in addressing these challenges with high performance, productivity, and sustainability, where the architecture adapts to match the unique needs of each application instead of vice-versa. In this paper, we present the novel use of a method for incorporating control information into the data stream, limiting wasted cycles and increasing hardware utilization. This method is described in Section II, and then featured in Section III as device-level reconfigurable architectures for in-stream control with scalable systolic arrays to accelerate two leading genomics applications based upon wavefront algorithms, Needleman-Wunsch (NW) and Smith-Waterman (SW) [1], and a third application, Needle-Distance (ND) [2], an augmentation of NW. These architectures are experimentally evaluated on Novo-G, the reconfigurable supercomputer in the NSF CHREC Center at Florida, where results achieve unprecedented levels of sustained performance for these problems. Case-study results are reported in Section IV, followed by Section V with summary and conclusions.

II. Scalable Systolic Arrays with In-Stream Control

Control for conventional systolic-array datapaths usually consists of a separate centralized controller, many small distributed controllers, or combinations of the two. Depending upon the complexity of the underlying algorithm, these conventional control methods can add significant overhead in terms of both chip area (complex state machines, additional control lines, etc.) and execution time (non-computational control states, pipeline stalls, etc.). In the case of DNA sequence alignment, the simplest design of such conventional controllers (i.e. with minimum area overhead) can achieve hardware execution time equal to $N \times$ (setup time + pipeline latency + PE configuration time + time to process streamed sequence + time to record results), assuming N successive alignments need to be calculated. Given that everything except "time to process streamed sequence" is overhead, such a controller is immensely inefficient. By contrast, the best possible performance from a systolic-array architecture is one that overlaps all of the overhead with useful work, achieving hardware execution time equal to pipeline latency + $N \times$ (time to process streamed sequence). This performance is possible using conventional methods but at the cost of a complex controller and other supporting logic, requiring many small state machines per processing element (PE) and many additional counters, registers, and control signals, all consuming a large percentage of PE area.

In the proposed method of in-stream control, we replace complex state machines, routed control signals, and supporting logic with special control data inserted directly into the input data stream. Control words intermixed with application data are used to signal state transitions and trigger complex actions on application data that follows in the stream. In so doing, we can achieve near-optimal time performance from a systolic-array architecture without a complex controller and the corresponding penalty in design complexity and area overhead (i.e. complex-controller performance with simple-controller overhead). Understanding that this method is not beneficial for all applications, this method is proving invaluable for hardware-accelerated implementations of sequence-alignment algorithms that are dominant and vital components of many genomics and bioinformatics applications. For such applications, reduction in controller overhead implies more device resources for other important circuitry, such as additional PEs, which translates into improved performance.

III. Application of In-Stream Control

FPGA-based sequence alignment accelerators are by no means new [3-5] and are commonly implemented as a pipelined, systolic array of PEs, each responsible for computing a single column of the scoring matrix generated from some form of dynamic programming (DP) equations. Figure 1 shows the DP equations for Needleman-Wunsch (other algorithms have similar equations). By loading each PE with a unique character from one sequence (loaded sequence of length X) and streaming another sequence (streamed sequence of length Y) through the pipeline one character at a time, up to X of the required $X \times Y$ scores are calculated simultaneously, reducing the $O(X \times Y)$ time complexity in software to $O(X+Y)$ in hardware. The wavefront aspect of this algorithm comes from data dependencies in the DP equations, where computation propagates like a wave across the scoring matrix as illustrated in Figure 1. Control for these systolic arrays generally requires PE configuration (query character load, on/off, correctly timed reset of initial conditions, etc.), alignment calculation, result recording, and setup for next alignments. In this section, we illustrate the proposed method of in-stream control with scalable systolic arrays to accelerate three important wavefront applications in genomics (NW, SW, ND).

$$S(i,j) = \max \begin{Bmatrix} S(i-1,j-1) + Sub(i,j) \\ S(i-1,j) + eog \\ H(i-1,j) + e \\ S(i,j-1) + eog \\ V(i,j-1) + e \end{Bmatrix}$$

$$for \quad 1 \le i \le i_{max}, 1 \le j \le j_{max}$$

$$H(i,j) = \max \begin{Bmatrix} S(i-1,j) + og \\ H(i-1,j) + eg \end{Bmatrix}$$

$$V(i,j) = \max \begin{Bmatrix} S(i,j-1) + og \\ V(i,j-1) + eg \end{Bmatrix}$$

$$S(0,i) = S(i,0) = og + i \cdot eg$$

$$H(0,i) = V(i,0) = og$$

$$Sub(i,j) = penalty \quad matrix$$

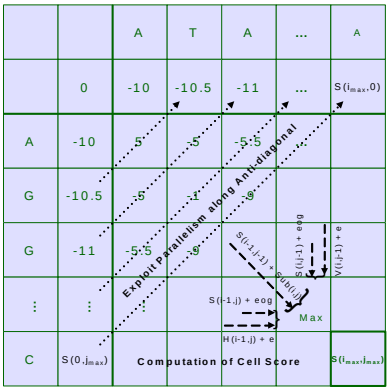


Figure 1: NW characteristic equations and score matrix
(for *og* = -10, *eg* = -0.5, and *Sub(i,j)* = 5 on match, -4 on mismatch).

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant No. EEC-0642422.

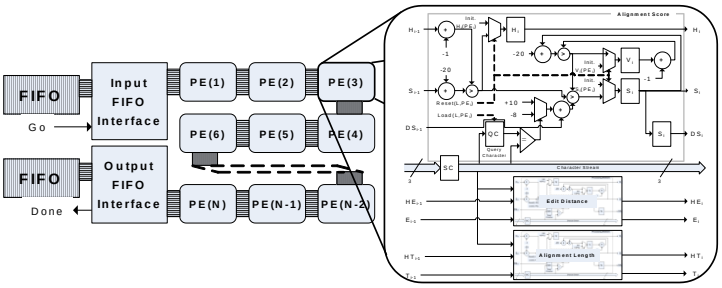


Figure 2: Systolic Array for NW and ND (the difference being design of the PEs, where ND includes distance modules).

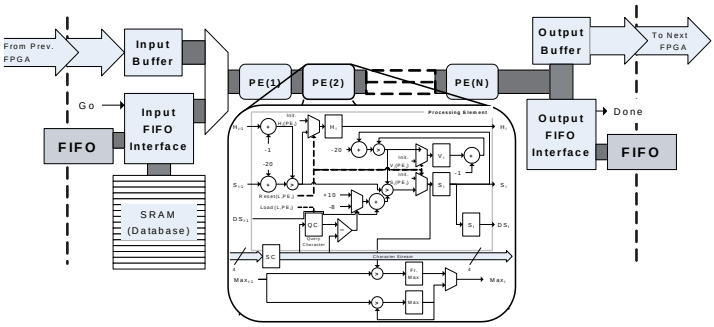


Figure 3: Systolic Array for Smith-Waterman.

A. Needleman-Wunsch (NW)

Using the standard DNA sequence alphabet (A, C, G, T, N), at least three bits are needed for digital encoding, representing five data and three unused characters. Our design of the NW application (Figure 2) adds a set of three special control characters (L, R, P) to the alphabet, replacing the unused values. Without loss of generality, these control characters are encoded directly into the sequence stream before it is transferred to the FPGA's input FIFO and provides control information needed to achieve near-optimal time performance with less area overhead than conventional methods. The design moves towards full hardware automation where meaningful results are obtained as a result of the correct sequence of data words passing through the systolic array, rather than requiring both the correct sequence of data words and the correct sequence of control signals from the controller and CPU.

The first control character 'L' is used to configure each PE with sequence characters. When a particular PE recognizes an 'L' character, it knows that the following sequence in the stream is for loading and not comparison. The character 'R' has multiple roles in the control scheme. Its primary role is to reset each PE to its initial conditions before each new streaming comparison, but is also used to signal the end of a load sequence. Finally, the 'P' character is used to signal the output FIFO interface that the result arriving on the previous clock cycle is a result that must be pushed onto the FIFO and to signal the end of a comparison sequence. The two-character sequence "PN" is used to signal that all comparisons have completed and trigger a completion signal to the CPU.

As an example, consider the following set of short sequences: {ACGT, TTG, ACNG, CNTG}. If it is desired to use NW to perform pairwise alignment between each unique sequence pair in the set (i.e. 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4), the CPU would send "**L**ACGT**R**TTG**PR**ACNG**PR**CNTG**PL**TTG**R**ACNG**PR**CNTG**P L**ACNG**R**CNTG**PN**" to the FPGA with a 3-bit encoding per character. After (50 + pipeline length) FPGA clock cycles, the six results are ready for retrieval when the CPU has free cycles.

From this example it is evident that if the same loaded sequence is to be used more than once in successive comparisons, then it is not necessary to reload PEs after every comparison, and streamed sequences can be piped through the array one after another. One common limitation to NW designs with simple control using conventional methods is that hardware must wait for the tail end of each streamed sequence to propagate through the last PE before each PE can be reset to initial conditions and the next comparison can begin. These wasted clock cycles are devastating to performance and leads to idle PEs nearly 50% of the time. With introduction of control characters into the stream as framing characters for each loaded or streamed sequence, a notion of local control is introduced where later PEs in the pipeline between an 'L' and 'R' can be reconfiguring while simultaneously an earlier group between an 'R' and 'P' can be processing a query while an even earlier group is doing something else. The cost of maximizing accelerator performance by limiting PE idle periods and reducing configuration overhead between runs is the two extra clock cycles per sequence to process the two framing characters, which is more than offset by the additional PEs that can be implemented due to the reduction in area overhead.

B. Smith-Waterman (SW)

The SW design in Figure 3 incorporates the same in-stream control methodology as NW described above but provides many additional functionalities over NW and therefore requires a more complex control scheme (i.e. more than just 'L', 'R', and 'P'). Additional functionalities include the ability to alternate between a FIFO stream from the CPU and a preloaded database stream from local SRAM, the ability to load additional sequences into unused PEs so that multiple queries can be calculated for a single streamed database, the ability to extend a long query across multiple FPGAs, and many more. More details on our SW control scheme will be given in the presentation.

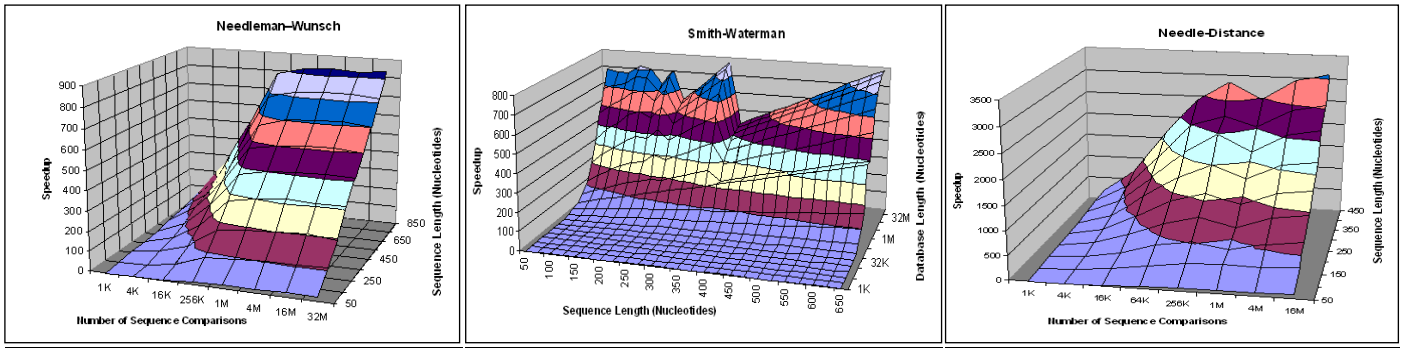
C. Needle-Distance (ND)

The Needle-Distance application has the same systolic architecture shown in Figure 2 and is used as an accelerator in ESPRIT [2], a metagenomics application designed for 16S rRNA sequence data analysis used by dozens of organizations worldwide. ESPRIT is a composite application consisting of five component applications. Among them, ND is the most time-consuming by far and is used for computing optimal pairwise distance between sequences by first calculating global alignment using NW, followed by the *quickdist* algorithm on the alignment. In designing PEs for ND, we recognized that these operations can be performed in parallel by simply adding distance calculation modules to the PEs of NW. Thus, key parts of ND are the same as our NW described earlier, with its in-stream control. More details on our ND design and ESPRIT will be given in the presentation.

IV. Experimental Evaluation on Novo-G

Novo-G, believed to be the most powerful reconfigurable computer ever constructed for academic research, has been operational in the NSF CHREC Center at Florida since July 2009. Novo-G consists of 24 compute nodes, each housing two GiDEL PROCStar-III accelerator boards. Each PROCStar-III contains four Altera Stratix-III E260 FPGAs, totaling 192 FPGAs system-wide, with each FPGA connected to 4.25 GB of dedicated memory. More information on Novo-G can be found at [6].

To evaluate performance of our in-stream control technique and the potential impact that large-scale RC supercomputers like Novo-G can achieve with it, the three previously discussed applications were implemented and tested with large data sets on multiple nodes of Novo-G. Several factors greatly affect PE area, such as equation parameters (Figure 1), dynamic ranges allocated for various signals (such as alignment score), or targeted hardware frequency, and therefore affect the number of PEs mapped per Stratix-III device. To be consistent, the same application-specific parameters required of the ND application for integration into ESPRIT were imposed on all three designs. Each of the three designs was optimized to fit the largest possible number of PEs per FPGA for the chosen configuration, yielding 850 PEs/FPGA for NW, 650 for SW, and 450 for ND, all at 125 MHz. The software baselines used for comparison with NW and SW are optimized, serial codes implemented in C according to standard algorithms



Baseline: $192 \cdot 2^{25}$, length 850 Sequence Comparisons Software Runtime: 11,026 CPU·hours on 2.4GHz Opteron		
# FPGAs	Runtime (sec)	Speedup
1	47,616	833
4	12,014	3,304
96	503	78,914
128	391	101,518
192 (est.)	270	147,013

Baseline: Human X Chromosome v 19200, length 650 Seqs Software Runtime: 5,481 CPU·hours on 2.4GHz Opteron		
# FPGAs	Runtime (sec)	Speedup
1	23,846	827
4	5,966	3,307
96	250	78,926
128	188	104,955
192 (est.)	127	155,366

Baseline: $192 \cdot 2^{24}$, length 450 Distance Calculations Software Runtime: 11,673 CPU·hours on 2.4GHz Opteron		
# FPGAs	Runtime (sec)	Speedup
1	13,522	3,108
4	3,429	12,255
96	144	291,825
128	118	356,125
192 (est.)	77	545,751

Figure 4: Results on Novo-G for NW (left), SW (Center), and ND (Right). Each chart illustrates performance of a single FPGA under varying input conditions. Each table shows scaling performance with varying number of FPGAs under optimal input conditions.

from relevant sections in [1] and each achieves ~100 MCUPS on a 2.4GHz Opteron core running 64-bit Linux, while ND is compared with the needledist.cpp code of ESPRIT at [7]. Output results from all three hardware-accelerated codes were compared with those from the software baselines to confirm correctness.

In Figure 4, for each application a chart illustrates relative design performance for one FPGA under varying input conditions. Corresponding tables show how the designs scale when executed on varying number of FPGAs. For the NW and ND designs, the most influential factors affecting performance are the average sequence length, which determines how many PEs are used for a particular comparison, and total number of comparisons, which determines how much software overhead such as DMA transfers is overlapped with FPGA execution. As expected, increasing hardware utilization by increasing the number of PEs in use (i.e. increasing sequence length) and hiding overhead by overlapping it with useful work (i.e. increasing number of comparisons) leads to best performance – speedups of ~830 per FPGA for NW and ~3100 per FPGA for ND, as shown in the respective contour plots.

In SW, depending on the database size, there are orders of magnitude more computation per comparison when compared to NW and ND, allowing software overhead to become hidden even for a relatively small number of comparisons, and as such database size becomes a more important factor in performance. The effect of average sequence length on performance is visibly different and is attributed to the aforementioned functionality that allows additional queries to be loaded into unused PEs for comparison with a single streamed database. Optimal performance of ~830 speedup per FPGA occurs when multiple queries fit in the array exactly without any unused PEs (i.e. query sizes 650, 325, … for 1 FPGA, or 1300, 650, … streamed across two FPGAs, etc.). Worst performance occurs when multiple queries do not fit in the array exactly and in such a way that maximizes the number of unused PEs (e.g. query size of 326 for 1 FPGA, or 651 streamed across two FPGAs, etc.). When sequence lengths are not held constant for successive comparisons, load-balancing techniques can be employed and this saw-tooth effect is less pronounced.

As for the multi-FPGA studies, these applications are completely independent across runs and as such are completely scalable given a sufficiently large dataset. Using a combination of scripting and MPI for inter-node coordination, the three applications were executed on up to 128 FPGAs (i.e. 16 Novo-G nodes with 8 FPGAs/node) and used to estimate performance on all 192 FPGAs in Novo-G. The tables show optimal performance when there are no unused PEs per run and there is sufficient hardware calculation to hide software overhead.

V. Summary and Conclusions

As shown in Figure 4 with SW, measured speedup peaked at 3,307 (and at 3,304 with NW) on a single Novo-G node using only one board with four FPGAs, as compared to software on a 2.4 GHz Opteron core. Thus, on a conventional HPC machine (even assuming no overhead), 3,307 of these Opteron cores working in parallel would be required to perform the same amount of work in the same period of time as half of a single Novo-G node. For the ND application, experimental results show speedup on a single FPGA in Novo-G in excess of 3,100 and exceeding 356,000 with 128 FPGAs. Projected speedup when employing all 192 FPGAs exceeds 545,000, which is comparable to running the same application on over 545,000 Opteron cores in a conventional supercomputer. To put this into perspective, the two most massive, expensive, and power-hungry machines cited at www.top500.org in May 2010 (Jaguar at ORNL and Roadrunner at LANL) have a combined approximate total of 346,000 cores.

Reconfigurable supercomputing, with scalable RC systems featuring many leading-edge FPGAs configurable specifically for high-intensity data processing for each application, holds the key to address escalating computational demands in genomics data mining and analysis with high performance, productivity, and sustainability. In this paper, we presented novel use of a method for in-stream control with scalable systolic arrays to accelerate a class of genomics applications, limiting wasted cycles and increasing hardware utilization. Combined with the computational power of the Novo-G machine, realization of this method to accelerate important genomics applications was demonstrated with unprecedented levels of sustained performance.

References

[1] W. Pearson and W. Miller, "Dynamic programming algorithms for biological sequence comparison," *Methods Enzymol.* 210: 575-601, 1992.

[2] Y. Sun, Y. Cai, L. Liu, F. Yu, M.L. Farrell, W. McKendree, W. Farmerie, "ESPRIT: Estimating Species Richness Using Large Collections of 16S rRNA Pyrosequences," *Nucleic Acids Research*, vol. 37, no. 10 e76, May 2009. PMCID: PMC2691849.

[3] S. Lloyd and Q. Snell, "Hardware Accelerated Sequence Alignment with Traceback," *Int. Journal of RC*, vol. 2009, Article ID 762362.

[4] O. Storaasli, W. Yu, D. Strenski, J. Maltby, "Performance Evaluation of FPGA-Based Biological Applications," *Proc. Cray Users Group*, Seattle, WA, May 2007.

[5] K. Benkrid, Y. Liu, and A. Benkrid, "A Highly Parameterized and Efficient FPGA-Based Skeleton for Pairwise Biological Sequence Alignment," *IEEE Trans. VLSI Systems*, vol. 17, pp. 561-570, 2009.

[6] Novo-G architecture overview, www.chrec.org/~george/Novo-G.pdf

[7] ESPRIT download, plaza.ufl.edu/sunyijun/ESPRIT.htm