
HARDWARE ACCELERATED SOLUTIONS FOR SEQUENCE ALIGNMENT

ACCELERATING GLOBAL ALIGNMENT WITH
GPU-BASED AND FPGA-BASED SOLUTIONS

SAM MARTIN VARGAS GIAGNOCAVO, AU703393

MASTER'S THESIS

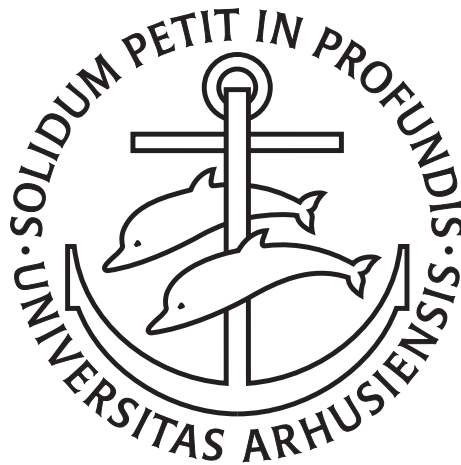
June 2023

Advisor: Christian Storm Pedersen

HARDWARE ACCELERATED SOLUTIONS FOR SEQUENCE ALIGNMENT

Accelerating global alignment with GPU-based and FPGA-based solutions

SAM



Master's Thesis

Department of Computer Science
Faculty of Natural Sciences
Aarhus University

June 2023

ABSTRACT

This thesis presents a GPU-based and FPGA-based implementation of a global sequence alignment algorithm. The proposed approach is based on the Needleman-Wunsch algorithm, which is widely used in bioinformatics for pairwise sequence alignment. Two implementations have been developed using Apple's Metal API for a GPU implementation, and Verilog for an FPGA implementation. The results show that both approaches achieve a notable speedup compared to equivalent non-accelerated implementations.

CONTENTS

1	INTRODUCTION	1
2	THE SEQUENCE ALIGNMENT PROBLEM	5
2.1	Pairwise Alignment	5
2.2	Multiple Sequence Alignment	8
3	RELATED WORK	11
3.1	Background	11
3.2	State of the art for GPU accelerated sequence alignment workloads	14
3.3	State of the art for FPGA accelerated sequence alignment workloads	18
4	SWIFTSEQAL: THE SWIFT SEQUENCE ALIGNMENT LIBRARY	23
4.1	Developing a Bioinformatics-specific library	24
4.2	Threads, Warps and Threadgroups	25
4.3	Submitting Work to the GPU with Metal	26
4.4	Accelerating the alignment using SIMD-group Functions	27
4.5	A note on the <i>makeBuffer</i> function	30
4.6	Assessing performance during development: XCTest and XCode's Instruments	32
4.7	GUI	34
5	VERILATOR AND THE XILINX SPARTAN 7	37
5.1	The Verilator testbench	37
5.2	Implementing the PU from a Systolic Array	39
5.3	Transitioning to Hardware	42
6	EVALUATION	45
6.1	Testing the "bytesNoCopy" option	45
6.2	CPU vs GPU	46
6.3	Testing the FPGA simulation	48
7	CONCLUSION	49
7.1	Future Work	50
I	APPENDIX	53
A	HASEAL	55
B	FIELD PROGRAMMABLE GATE ARRAYS	57
B.1	Synthesis	57
B.2	Verilator	57
	BIBLIOGRAPHY	59

LIST OF FIGURES

Figure 1	How mutations affect the resulting sequence. Graphic from the University of Leicester.	1	
Figure 2	The traceback step on both alignment algorithms.		8
Figure 3	Optimal alignment visualization for three sequences.	10	
Figure 4	Inner layout of a configurable logic block.	14	
Figure 5	Latency comparison for computing the product of an array.	15	
Figure 6	Keeping values using shared memory (left) and local registers (right) for aligning two sequences.		16
Figure 7	Reduced solution space in a dynamic programming implementation.	19	
Figure 8	Processing diagonals in the dynamic programming table.	28	
Figure 9	Graphical representation of Listing 5 . Example extracted from Oxford University Mathematical Institute, CUDA Programming, Lecture 4.	29	
Figure 10	The SIMD group implementation of the algorithm.	29	
Figure 11	Using XCode's Instruments to compare performance metrics for different iterations of the code.	34	
Figure 12	Opening a FASTA file in HASEAL.	35	
Figure 13	Circuit design overview of a processing element from a systolic array.	40	
Figure 14	Comparison of Metal buffers created with and without the <i>bytesNoCopy</i> option.	46	
Figure 15	GPU allocation, deallocation and compute events recorded during testing.	47	
Figure 16	Verilator simulation results. Cycles required to read from memory are not included in the simulation.	48	
Figure 17	Warp 1 executes the first subset of values. Warp 1 and warp 2 continue computing the table from the values they were originally computed by warp 1.	50	
Figure 18	Startup screen for the application. At the right pane, a list with the latest files used with the program.	55	

Figure 19	Sequence view of the selected FASTA file. At the bottom, specific sequence info. 55
Figure 20	Raw sequence view. Users can view the raw FASTA file that was selected. 56
Figure 21	Integration with macOS allows displaying recent files in the mission control view. 56
Figure 22	Output of the synthesis process exported as an SVG file. File generated by the <i>Yosys</i> framework. 57
Figure 23	Value changes as recorded in the “waveform.wcd” file. 58

LIST OF TABLES

Table 1	Performance comparison obtained from the forward pass of the Needleman-Wunsch algorithm. Each row represents a different sequence length (bp, base pairs). Runtimes are represented in seconds. 47
---------	--

LISTINGS

Listing 1	Gap openings in different sequences. Formatted according to the FASTA format. 6
Listing 2	A simple Verilog module containing a conditional assignment. 12
Listing 3	Computing the linear index from a multidimensional index. 24
Listing 4	An example of <i>conditionals</i> to avoid in a compute shader. 25
Listing 5	Sample code extracted from Oxford University Mathematical Institute, CUDA Programming, Lecture 4. 28
Listing 6	Creating a page buffer from a page-aligned address. 31
Listing 7	Main function from a sample Verilator testbench. 38
Listing 8	Connectivity from different modules within a slide. 41

ACRONYMS

UMA	Unified Memory Architecture
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
SoC	System-on-Chip
SIMD	Single Instruction/Multiple Data
FGMT	Fine-Grained Multithreading
HDL	Hardware Description Language
CLB	Configurable Logic Block
GCUPS	Giga Cell Updates Per Second
HVL	Hardware Verification Language
PU	Processing Unit

*Man only likes to count his troubles;
he doesn't calculate his happiness.*

— Fyodor Dostoevsky

ACKNOWLEDGMENTS

Special thanks to my supervisor, Prof. Christian Storm Pedersen, and to all the wonderful people I have met at the Bioinformatics Research Center.