Implementação em FPGAs dos Algoritmos Global (Needleman-Wunsch) e Local (Smith-Waterman) de Seqüenciamento de Gens

Cristiane da Silva Morony, Kalinka RLJ Castelo Branco, Edward David Moreno

Ciência da Computação – Centro Universitário Eurípides de Marília (UNIVEM) –
Av Hygino Muzzi Filho 520, CEP 17525-901 – Marília – SP – Brasil
crismorony@yahoo.com.br, {kalinka, edmoreno}@fundanet.br

Resumo. *O artigo tem como objetivo implementar em hardware os algoritmos que são considerados padrão para a comparação e o alinhamento global e local das seqüências genéticas de DNA utilizando a técnica de programação dinâmica. Esses algoritmos foram implementados e analisados em software (linguagem C) e em hardware (sendo descritos na linguagem VHDL e prototipados usando a tecnologia FPGA), visando conseguir um bom desempenho. Foram analisados os tempos de execução dos resultados alcançados tanto em software e hardware e, comparados com outros sistemas específicos para o seqüenciamento genético, mostrando bons resultados de nossa implementação em FPGAs.*

Palavras-chaves: *Sequenciamento de Gens, Algoritmos Global e Local, Programação Dinâmica, Desempenho de Hardware e Software.*

1. Introdução

Com os avanços da engenharia genética, na tentativa de estudar e mapear o genoma dos seres vivos e, em especial, o do homem, dados de seqüências genéticas são gerados em taxas cada vez maiores. Profissionais como biólogos deparam-se com uma enorme vazão de dados que gostariam de classificar e comparar com aqueles armazenados nos diversos bancos de dados.

Estes profissionais necessitam da utilização de instrumentos computacionais como *software* e *hardware especializados* para realizar de maneira mais eficiente e em menor tempo as análises dos algoritmos de alinhamento de seqüências, visto que, em grande parte destas aplicações o tempo e o resultado são variáveis fundamentais [Oliveira 2002].

Neste sentido, a técnica de programação dinâmica aplicada nos algoritmos de alinhamento global, conhecido como algoritmo de *Needleman-Wunsch* [Needleman and Wunsch 1970] tem como finalidade a similaridade global entre duas seqüências de DNA, onde todas as bases são alinhadas umas com as outras ou com *gaps*. Isto é, os valores globais requerem que o alinhamento comece no início e se estenda até o final de todo o comprimento da seqüência [Ward, Galban and Ruiz 2004].

O algoritmo de alinhamento local, como o algoritmo de *Smith-Waterman* [Smith and Waterman 1981], possui similaridade local entre duas seqüências, não necessita alinhar todas as bases em todas as seqüências, ou seja, os valores locais requerem identificação na região mais similar entre duas seqüências e também a penalidade por falta atribuída ou não [Ward, Galban and Ruiz 2004].

Com uma pequena mudança da técnica de programação dinâmica aplicada no alinhamento global torna-se local. Assim, é visível a importância dos algoritmos de alinhamento de seqüências na bioinformática, pois ao encontrar o alinhamento máximo será possível detectar altas similaridades que existem entre as seqüências de maneira a permitir encontrar similaridades funcionais e conseqüentemente estruturais, contribuindo desta forma para os avanços nas pesquisas dessas áreas.

Atualmente há vários programas que se encontram disponíveis para fazer a comparação de seqüências biológicas, exemplo o BLAST [Altschul 1990]. *Softwares* como o BLAST ou FASTA são extensivelmente usados para bancos de dados biológicos e têm sido projetados para executar em computadores padrões, isto é, máquinas *Von Neuman*, e incluem técnicas para acelerar os processos.

Este *software* somente implementa o algoritmo para alinhamento local das seqüências e não utiliza a programação dinâmica e sim heurística. Por este motivo, não têm garantia de que vai encontrar o melhor alinhamento [Altschul 1990]. Uma vantagem de sua utilização é a utilização em plataformas com baixo custo computacional.

Por esse motivo, escolheu-se esse assunto, pois mesmo existindo algoritmos de programação dinâmica para o problema da comparação de seqüências genéticas, esses algoritmos são amplamente usados pela comunidade acadêmica da área biológica na tentativa de encontrar maiores soluções para os desafios que estão sempre surgindo nesta área. Não obstante

o tempo para fazer a varredura nos bancos de dados na tentativa de encontrar rápidas soluções ainda é muito alto, uma vez que os mesmos estão crescendo rapidamente.
Motivados pelos avanços tecnológicos e as novas possibilidades de se projetar *hardware*, pensou-se na possibilidade de implementação dos algoritmos de alinhamento global e local usando a tecnologia FPGA, com intuito de analisar mais detalhadamente as partes mais críticas desses algoritmos e poder amenizar os problemas de desempenho dos mesmos, com versões específicas desses algoritmos em *hardware*.
Assim, o objetivo principal deste artigo é a implementação em *hardware*, dispondo da tecnologia FPGA, o algoritmo global (*Needleman-Wunsch*) e local (*Smith-Waterman*) que são considerados ou adotados como padrão para a comparação e o alinhamento das seqüências genéticas de DNA utilizando a técnica de programação dinâmica. Foram realizadas considerações sobre o projeto e o desempenho das respectivas implementações em FPGAs, assim como otimizações em nível arquitetural e de programação em VHDL, visando obter bons resultados. Também foram realizadas comparações e respectivas análises com outras soluções já existentes em *hardware*, em especial, aquelas baseadas em FPGAs.
O artigo está organizado em 5 seções. A seção 2 apresenta algumas soluções específicas em *hardware*, a seção 3 apresenta resultados de nossas implementações usando linguagem C. A seção 4 apresenta dados de nossa implementação em *hardware*, destacando os recursos espaciais e temporais dos protótipos em FPGAs. Finalmente a seção 5 apresenta as conclusões.

2. *Hardware* para Comparações de Seqüências

A versatilidade, a velocidade e o aumento do poder computacional dos PCs (Computadores Pessoais) em redes locais, têm guiado os cientistas no trabalho com modelagem molecular e alinhamento de múltiplas seqüências de DNA (evolução molecular) independentemente da existência de supercomputadores [Lavenier 1996].
A maioria das máquinas (*hardwares*) que tem sido projetadas para acelerar a comparação de seqüências biológicas são baseadas em estrutura de arranjos (*arrays*) lineares. Entretanto, essas estruturas diferem na flexibilidade de programação. Dentre essas podem ser destacadas três categorias [Lavenier 1996]:

- ***Arrays* Dedicados VLSI (*Very Large Scale Integration*)**: essas máquinas podem conseguir a mais alta performance sob um único algoritmo que foi adaptado dentro do silício e não pode ser modificado. Alguns exemplos dessas máquinas são: **BioSCAN** (*Biological Sequence Comparative Analysis Node)* que acelera a identificação de seqüências similares de DNA ou de proteínas sem permissão de *gaps* e o **SAMBA** (*Systolic Accelerator for Molecular Biological Application*) que é um *array* sistólico dedicado para analisar os algoritmos envolvidos na comparação de seqüências biológicas;
- ***Arrays FPGAs (Field Programmable Gate Arrays)*:** eles incluem sistema com *hardware* reconfigurável (FPGA). Eles tendem a ser mais lentos e têm densidade muito mais baixa do que os *arrays (*arranjos*)* VLSI. A criação e modificação de algoritmos para esses sistemas são possíveis. Exemplos*:* ***Splash-2****:* sua arquitetura pode facilmente suportar aplicações paralelas, tais como sistólica ou computações de dados paralelos. O *Splash -2* é um melhoramento da primeira versão do Splash -1 [Gokhale 1990], a qual consistiu de um tamanho fixo de *array* linear de *chips Xilinx* 3090 FPGA e o *Hscan:* é um filtro processador dedicado para pesquisa em bancos de dados de DNA. Desenvolvido pela IRISA (*Institut de Recherque en Informatique et Systèmes Aléatories*) [Guerdoux-Jamet and Lavenier 1995][Lavenier 1996]; e
- **Arrays Programáveis VLSI**: essa última categoria de máquinas empenha-se para a flexibilidade algorítmica de sistemas reconfiguráveis e a velocidade e a densidade de máquinas VLSI de propósito único. Logo, a principal vantagem é a facilidade de programação. Exemplo: B-SYS (*Brown SYStolic Array*) [Hughey and Lopresti 1991] foi projetado principalmente para o propósito de comparação de seqüências, ainda que a flexibilidade de programação possibilite muitas outras aplicações. Esta máquina foi fabricada pela *Brown University* e testada sobre um protótipo de 10 *chips*, conduzindo a uma quantidade total de 470 processadores (47 processadores por *chip*) [Hughey 1991].

A figura 1 ilustra as diferenças referentes aos itens mencionados. Os *arrays* dedicados VLSI são vinte vezes mais rápidos do que os FPGAs ou os *arrays* programáveis VLSI. Já, os FPGAs e os *arrays* programáveis VLSI têm performances comparáveis.
Sabe-se que as pesquisas e os estudos biológicos caminham numa rápida evolução, logo, esta área precisa de tecnologia que realmente consiga acompanhar seu enorme desenvolvimento. Diante desse fato, possivelmente, pesquisadores, cientistas biológicos e projetistas de *hardware* foram impulsionados a estudar e averiguar as funcionalidades da arquitetura reconfigurável utilizando a tecnologia FPGA nesta área.

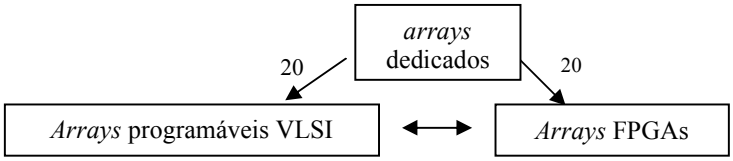

Figura 1 Performances dos sistemas VLSI e FPGA [Lavenier 1996]

A presente seção retrata a abordagem de dois computadores maciçamente paralelos que utilizam arquiteturas reconfiguráveis: NGEN e POLYP. Destaca-se o alinhamento de seqüências sobre a arquitetura *Cray* MTA-2 e uma abordagem em *hardware* para algoritmos de comparação de seqüências.

2.1. NGEN (eNGiNE) e POLYP (Paralle OnLine PolYmer Processing)

NGEN é uma plataforma de *hardware* para computação maciçamente paralela, permitindo sua configuração, em baixo nível, pelo usuário. Baseado na tecnologia FPGA é reconfigurável para cada problema nos níveis elementares de processamento digital e de comunicação.
A arquitetura NGEN sobressai-se pelo uso de acessos maciçamente paralelos à memória, em conjunto com variadas conexões entre FPGAs e a capacidade de configuração minuciosa da topologia global de interconexão.
A sigla POLYP refere-se ao processamento paralelo o*n-line* de polímeros. POLYP é a segunda geração de computadores maciçamente paralelos reconfiguráveis. Sua construção foi baseada em FPGAs microrreconfiguráveis e alta densidade de memória distribuída adicional. O projeto do computador POLYP difere em vários aspectos do projeto do NGEN, permitindo a microrreconfiguração dinâmica nos seus FPGAs [Tangen and Mccaskill 1998].

2.2. Alinhamento de Seqüências sobre o Cray MTA-2

Esta seção apresenta algumas características da arquitetura *Cray* MTA (*Multithreaded Architecture*), segundo Vianna [Vianna 2003] e como elas influenciam no algoritmo padrão de alinhamento de seqüências de DNA.
Essas características foram relacionadas com as características da arquitetura híbrida SIMD (*Single Instruction Single Data*) e MIMD (*Multiple Instruction Multiple Data*) para amenizar a deficiência do limite do tamanho das seqüências de consulta, na arquitetura híbrida, com a utilização das placas *Systola*1024 e *Fuzion*150 para a componente SIMD, enquanto que a componente MIMD é estruturada em um *Cluster* de 16 PCs.
Algumas variações do algoritmo de programação dinâmica foram implementadas no *Cray*, com relação a diferentes ordens de atualização da matriz, devendo, portanto ter concentração na implementação que realiza atualização da matriz na ordem antidiagonal.
Esta especialização foi escolhida devido à facilidade de relacionamento com a técnica empregada pelas arquiteturas *Systola* e *Fuzion* [Vianna 2003], as quais executam a computação das células da matriz da mesma forma. As arquiteturas *Systola* e *Fuzion* são melhor descritas em [Lang 1986].

2.3. Estrutura Sistólica para Algoritmos de Comparação de Seqüências

Um trabalho recente é o de Carvalho (2003). Carvalho apresentou uma visão geral de uma estrutura sistólica para algoritmos de comparação de seqüências genéticas baseados em programação dinâmica. Foi utilizada a placa APEX PCI *Development Board* do fabricante *Altera*, contendo o FPGA APEX EP20K400EFC672. Essa placa foi instalada em um computador da *Dell* com um processador Pentium IV. A ferramenta de síntese adotada foi o *Quartus* II da própria *Altera* [Carvalho 2003].
A implementação dessa estrutura sistólica em *hardware* foi baseada no algoritmo de comparação local de duas seqüências proposto por *Smith-Waterman* [Smith and Waterman 1981] e se deu apenas em nível de síntese, desta maneira o resultado apresentado em [Carvalho, 2003] foram obtidos a partir de simulações feitas na ferramenta de desenvolvimento do fabricante ALTERA que fornece resultados bastante precisos, não só em termos de funcionalidade do sistema, como também em termos de velocidade final do sistema indicando caminhos críticos e atrasos dos sinais.
Os testes foram feitos com seqüências fictícias cuja matriz de similaridade era conhecida ou poderia ser construída facilmente. Para o teste de desempenho, vetores de diferentes tamanhos foram sintetizados e suas freqüências máximas de operação foram obtidas a partir de informações fornecidas pela ferramenta de síntese, ver tabela 1.

Tabela 1.Desempenho obtido da ferramenta de síntese [Carvalho 2003].

Quantidade de células no vetor	Quantidade de elementos lógicos	Freqüência máxima de	Média de elementos utilizados por célula
5	146	122,49 MHz	29,2
10	507	77,54 MHz	50,7
15	900	70,35 MHz	60
20	1292	64,94 MHz	64,6
25	1758	62,63 MHz	70,32
30	2222	56,45 MHz	74,06
40	3204	56,27 MHz	80,1
50	4403	56,1 MHz	88,06

Como esperado, pode-se perceber que quanto maior a quantidade de células do vetor, maior a utilização dos recursos espaciais oferecidos pelos FPGAs e menor sua freqüência máxima de operação. Este projeto mostrou como um *hardware* dedicado, baseado em uma estrutura sistólica, é capaz de linearizar a complexidade temporal e reduzir o tempo de processamento desses algoritmos.
As arquiteturas especiais brevemente comentadas neste artigo (NGEN, POLYP, *Cray* MTA-2 e estruturas sistólicas para algoritmos de comparação de seqüências baseados em programação dinâmica) mostraram a utilização de FPGAs na computação maciçamente paralela. Esses projetos permitem antever o potencial de utilização de dispositivos programáveis como importante bloco de construção de sistemas de computação das mais variadas espécies.

3. Algoritmos de Alinhamentos Implementados na Linguagem de Programação C

Esta seção apresenta dados de desempenho da nossa implementação dos algoritmos de alinhamento global e local, implementados em C, aqui denominados respectivamente de AGM e ALM.
O *hardware* utilizado para realizar a execução desses algoritmos foi um computador Pentium IV 1.66 GHz, AMD Athlon (tm) XP 2000+, 240 MB de memória RAM, 256 K de *cache*, com sistema operacional *Microsoft Windows* XP. Foi possível variar o tamanho das seqüências utilizadas na execução dos algoritmos AGM e ALM, mudando de 10 até 1000 caracteres. A tabela 2 mostra o tempo de execução (em segundos) de seqüenciamento, para vetores de vários tamanhos.

Tabela 2 Performances dos algoritmos AGM e ALM em C

Tamanho das Seqüências	Tempo em Segundos		
	AGM	ALM	Diferença %
10	0.220 (***milissegundos***)	0.160 (***milissegundos***)	27
20	1	1	0
30	2	2	0
40	4	4	0
50	9	9	0
100	57	60	5.2
150	227	222	2.2
300	2664	2979	11.8
400	7999	10259	28.2
500	19354	21187	9.5
600	39704	43281	9
1000	300184 = **83 horas**	328280 = **91 horas**	12

Observando os dados da tabela 2 é possível verificar que o algoritmo global executa mais rápido que o algoritmo local, resultado esperado, pois o algoritmo local além de fazer as mesmas operações que o global também precisa de uma operação de obter o maior seqüenciamento. Interessante observar também que esses tempos são quase iguais para seqüências de até 50 caracteres e se tornam maiores para seqüências de tamanhos maiores a 300 caracteres (diferença acima de 5%, e pode chegar a 28%).

Conforme esperado, também é possível observar que o tempo para realizar o alinhamento aumenta conforme o tamanho da seqüência aumenta, e se torna crítica para as seqüências maiores de 300 caracteres. Para seqüências de tamanhos 1000, esse tempo é da ordem de dezenas de horas. Isso lembra a importância de ter plataformas e/ou algoritmos específicos que possam diminuir o tempo de execução dessas operações de alinhamento (global e local), em especial, para grandes tamanhos de seqüências.
A seguir se apresenta uma comparação dos dados obtidos com a nossa implementação em C (executando em um Pentium IV 1.66 GHz) com aqueles obtidos na plataforma SAMBA. A arquitetura SAMBA (*Systolic Accelerator for Molecular Biological Application*) [Lavenier, 1996] [Mattos, 1994] é um *array* sistólico dedicado para a comparação de seqüências biológicas. Esse *hardware* acelera uma versão parametrizada do algoritmo de *Smith-Waterman* [Arnold, 1992] permitindo a computação de alinhamentos globais e locais com ou sem *gap penalty*.
Pelo fato da arquitetura SAMBA utilizar o algoritmo de *Smith-Waterman*, é possível realizar a comparação dos dados de performances desta arquitetura com as performances obtidas nas nossas implementações dos alinhamentos Global e Local implementados em C. Na comparação usamos as seqüências definidas na versão 31 do banco SWISS-PROT[1] [Lang, 1986], banco de dados de proteínas composto por um conjunto de 815 proteínas. A versão 31 do banco SWISS-PROT contém exatamente 43.470 seqüências distribuídas sob 15.335.248 aminoácidos. As 815 seqüências representam um total de 307.400 aminoácidos. A implementação do SAMBA, além de permitir plataformas de *hardware* maciçamente paralelas, usa também diferentes matrizes de substituição[2] e *gap penalties*.
A arquitetura SAMBA contém uma *workstation*, um *array* sistólico composto de 128 processadores de 12 *bits*, e uma interface baseada em FGPA (figura 2). A interface FPGA é a placa PerLe-1 desenvolvida por *Bertin* (1992) e funcionava como um *driver* programável para o *array* sistólico.

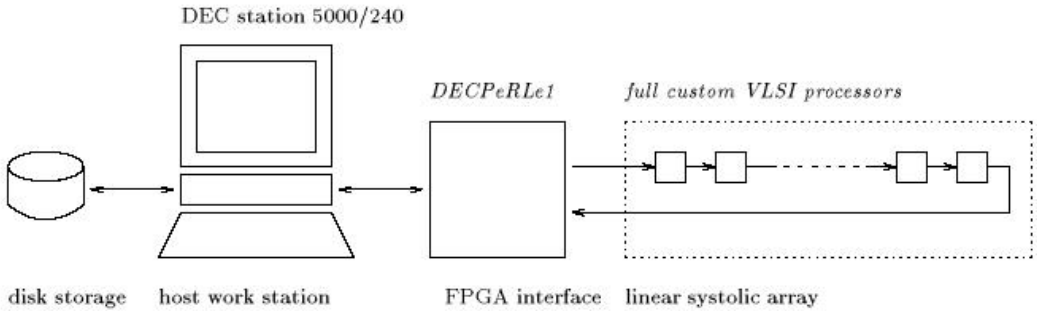

Figura 2 Sistema completo do *hardware* SAMBA [Lavenier, 1996].

SAMBA pode ser comparado com dois outros sistemas de *hardware*: BISP [Chow, 1991] e BioSCAN [White, 1991]. O BISP (*Biological Information Signal Processor*) fornece uma alta velocidade e um sistema linearmente extensível que pode localizar estaticamente subseqüências similares de duas seqüências de DNA ou de proteínas. Este implementa uma versão modificada do algoritmo do *Smith Waterman* e permite que vários parâmetros sejam estabelecidos.
O *hardware* foi projetado no Instituto de Tecnologia da Califórnia. Um *chip* contém 16 processadores e um protótipo da máquina de 16 *chips* foi validado, tornando possível a computação de tamanho de seqüências ilimitadas. Essa máquina não tem uma versão comercial.
Os protótipos do BISP e BioSCAN contém, respectivamente, 256 processadores de 16 *bits* e 12.000 processadores de 1 *bit*. Porém, a arquitetura BISP e SAMBA são similares. Elas diferem da seguinte maneira: o *array* BISP é acionado com um processador programável (Motorola 68020) enquanto o SAMBA usa tecnologia FPGA.
Para efeitos de comparação também se usaram os dados obtidos por Lavenier (1996), obtidos a partir da execução do *software* SSEARCH (que encontra alinhamentos locais de acordo com o algoritmo de *Smith* e *Waterman*, o qual é fornecido com o pacote de *softwares* FASTA), e executam sob uma *workstation* DEC-Alpha 21064 de 150Mhz.
Os dados da tabela 3 indicam o tempo (em segundos) de pesquisa usando o *software* SSEARCH sob diferentes *workstations*, que seguem a arquitetura SAMBA executando em diferentes processadores. Na tabela 3.2, esses dados são denominados conforme a máquina usada, assim por exemplo, **SAMBA – DEC - Alpha 150MHz**, define a execução do SAMBA com processadores DCE Alpha 150. As outras descrições são: **SAMBA - Sun SPARC 5 110MHz, SAMBA - DEC 5000/250 40MHz**.

[1] SWISS-PROT: *Protein Knowledgebase*. Banco de dados de seqüências de proteínas.
[2] Matrizes de Substituição: desenvolvida na década de 1980 e são as mais usadas no alinhamento de seqüências protéicas [Raimbault, 1993].

Tabela 3 Performances do tempo consumido em segundos [Lavenier, 1996].

Tamanho da Seqüência	Tempo em Segundos					
	AGM	ALM	SAMBA	SAMBA DEC-Alpha 150MHz	SAMBA - Sun SPARC 5 110MHz	SAMBA - DEC 5000/250 40MHz
10	0.220 milissegundos	0.160 milissegundos	25	57	95	182
30	2	2	25	120	239	548
100	57	60	26	350	746	1407
300	2664	2979	30	1041	2215	4054
1000	300184	328280	40	3468	7300	12920

O dados da tabela 3 comparam a nossa implementação (seqüencial) com aquelas obtidas para o SAMBA com diferentes plataformas, que usam vários processadores funcionando em paralelo. Para uma melhor visualização, a figura 3 ilustra os diferentes desempenhos (tempo de execução em segundos) dos alinhamentos global (AGM) e local (ALM) em C juntamente com aqueles da arquitetura SAMBA sob diferentes *workstations*.
Importante ressaltar que a arquitetura SAMBA contém uma *workstation*, um *array* sistólico linear composto de 32 *chips full custom* distribuídos sobre 2 (duas) placas. Um *chip* integra 4 (quatro) processadores, conduzindo a um *array* sistólico de 128 processadores e uma interface baseada em FPGA.

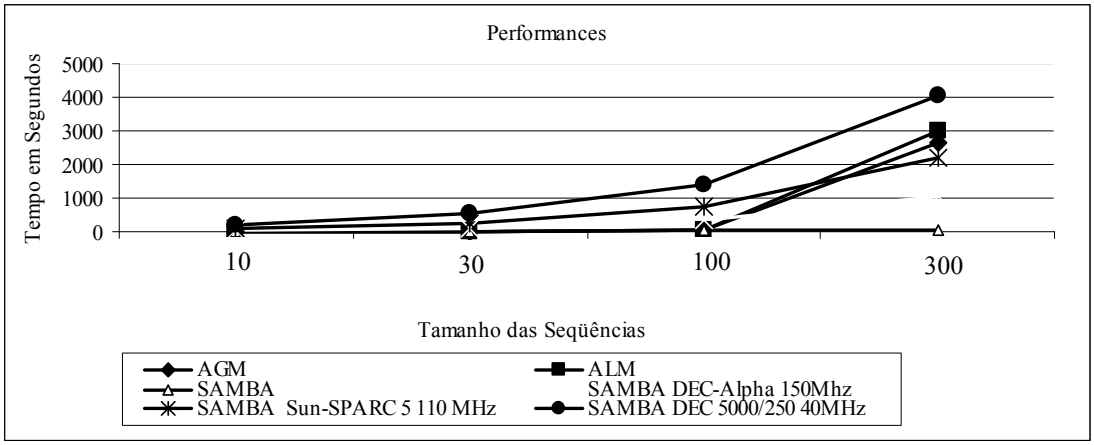

Figura 3 Performances dos alinhamentos com tamanhos de seqüências até 300 caracteres

Esses dados do SAMBA foram obtidos em 1996 e com múltiplos processadores. Não obstante, a nossa implementação possui bom desempenho (menor tempo de execução) para seqüências menores a 300 caracteres. Para seqüências maiores a 300 caracteres, a nossa versão seqüencial é muito lenta e merece atenção o uso de máquinas paralelas.
Considerando que nos últimos anos não se vê um desenvolvimento de máquinas paralelas específicas para problemas de bioinformática, e nem a utilização de FPGAs modernas, e levando em consideração os avanços tecnológicos nesta área e a necessidade crescente de termos soluções eficientes para esses problemas, decidimos fazer uma implementação específica desses algoritmos em FPGAs.

4. Implementação em *Hardware* dos Algoritmos de Seqüenciamento

Na implementação dos algoritmos de alinhamento de seqüências, global e local em VHDL usamos uma máquina de estado finito (*Finite State Machine* – FSM) [Ordonez 2003]. Esta máquina de estado permite descrever os estados da máquina e as suas respectivas transições de maneira síncrona. Com a máquina de estado é possível: i) ficar em um estado sempre após ocorrer um evento; ii) mudar de estado conforme mudança de um valor; iii) mudança de estado em seqüência lógica; e iv) mudança de estado sem seqüência.

Tabela 4 Descrição dos estados do algoritmo Global em VHDL.

Estado	Descrição dos Estados
0	Inicializa a máquina de estados e zera todas as posições da matriz.
1	Preenche a primeira linha e primeira coluna do inicio ao fim.
2	Inicialização dos caracteres das seqüências 1 e 2 que serão comparadas e apresentação dos mesmos nos respectivos vetores da linha e da coluna.
3	Cálculo para realizar o preenchimento do restante da matriz e atribuição de valores a sinais e variáveis.
4	Cálculo para realizar o procedimento *traceback*, o qual retorna o melhor alinhamento entre as seqüências analisadas e comparadas.

Tabela 5 Descrição dos estados do algoritmo Local em VHDL.

Estado	Descrição dos Estados
0	Inicializa a máquina de estados e zera todas as posições da matriz.
1	Preenche a primeira linha e primeira coluna do inicio ao fim.
2	Inicialização dos caracteres das seqüências 1 e 2 que serão comparadas e apresentação dos mesmos nos respectivos vetores da linha e da coluna.
3	Cálculo para realizar o preenchimento do restante da matriz
4	Busca e retorna o maior valor dentro da matriz.
5	Cálculo para realizar o procedimento *traceback*, o qual retorna o melhor alinhamento entre as seqüências analisadas e comparadas.

A tabela 4 apresenta o número de estados e suas respectivas descrições no algoritmo global em VHDL. O algoritmo global precisou de 5 estados, enquanto o algoritmo local de 6 estados (um adicional ao algoritmo global, que permite calcular o maior valor dentro da matriz). Os estados do algoritmo local são mais bem definidos na tabela 5.
Desta forma, os algoritmos global e local foram descritos em VHDL e simulados usando o recurso de síntese do FPGA e obtiveram-se dados de desempenho em *hardware*. Após esta descrição fez-se o uso do recurso da implementação dos mesmos, usando apenas duas famílias da ferramenta *Xilinx Foundation Series* [Xilinx, 2006]. É importante ressaltar que *arrays* (arranjos) multidimensionais geralmente não são suportados pela ferramenta de síntese (*synthesis*) e por este motivo as famílias da ferramenta *Xilinx* que conseguiram suportar a implementação dos *arrays* usados nos algoritmos foram apenas:

- **Família** : VIRTEX, **Dispositivo** V800FG680, **Velocidade**: -6; e
- **Família**: VIRTEXE, **Dispositivo**: V812EFG900, **Velocidade**: -8.

Os tempos gerados pela ferramenta *Xilinx* representam o tempo necessário que o circuito que implementa todas as operações que executam os algoritmos global e local. Esses dados estão relacionados aos atrasos para gerar a saída desejada de um determinado circuito. Este tempo é o resultado da soma do tempo de lógico mais o de roteamento em FPGA do circuito digital. Importante ressaltar que a tabela 6 exibe as estatísticas temporais em nanosegundos, para uma seqüência de tamanho 10 caracteres.

Tabela 6 Temporização dos algoritmos global e local em VHDL com as famílias VIRTEXE e VIRTEXE.

	Algoritmos	Tempo de Lógica	Tempo de Roteamento	Atraso Total	Freqüência de Operação
VIRTEX V800FG680	Global	10.878 ns 22.9%	36.590 ns 77.1%	47.468 ns	21.067 MHz
	Local	10.900 ns 20.8%	41.518 ns 72.9%	52.418 ns	19.077 MHz
VIRTEXE V812EFG900	Global	7.161 ns 16.0%	37.674 ns 84.0%	44.835 ns	22.304 MHz
	Local	7.089 ns 14.9%	40.542 ns 85.1%	47.631 ns	20.995 MHz

Observa-se um tempo de lógica menor para a família VIRTEXE em relação à família VIRTEX. Conforme visto na seção 3, percebe-se novamente que o algoritmo local requer mais tempo de execução em relação ao algoritmo global. Neste caso, as diferenças são respectivamente de 10% e 6.2% para as placas Virtex e Virtex E. Lembrar que essa diferença, para um tamanho de 10 caracteres, foi de 27% para a implementação em C. Isto quer dizer que em *hardware* as diferenças são menores.

Pode-se observar então que as estatísticas temporais não sofrem grandes diferenças quando se mapeiam esses algoritmos em diferentes FPGAs, uma vez que, os algoritmos implementados têm pequenas diferenças, ou seja, nota-se isso nos estados da tabela 4 e 5.

Também foi possível mudar o tamanho das seqüências, considerando 8, 10 e 15 caracteres. Na tabela 7 se observa o tempo (em nanosegundos) da implementação dos algoritmos de alinhamento global e local em *software*, (usando a linguagem de programação C e execução em umMPentium IV 1.66 GHz). Já a tabela 8 apresenta os tempos (também em nanosegundos) da implementação em *hardware* (FPGAs), obtidos usando a ferramenta *Xilinx*. Para uma melhor visualização, a figura 4 mostra o tempo de execução das nossas implementações, dos algoritmos global e local tanto em C quanto em VHDL, utilizando a família VIRTEX V800FG680.

Tabela 7 Performance da implementação em *software* (em C).

Tamanho da Seqüência	**Global**	**Local**
8 caracteres	216.000 ns	156.000 ns
10 caracteres	220.000 ns	160.000 ns
15 caracteres	750.000 ns	744.000 ns

Tabela 8 Performance da implementação em *hardware* (em FPGA).

	VIRTEX V800FG680		**VIRTEX E V812EFG900**	
Tamanho da Seqüência	**Global**	**Local**	**Global**	**Local**
8 caracteres	47.468 ns	52.418 ns	44.835 ns	47.631 ns
10 caracteres	45.902 ns	55.065 ns	44.748 ns	51.117 ns
15 caracteres	57.445 ns	67.476 ns	50.887 ns	62.626 ns

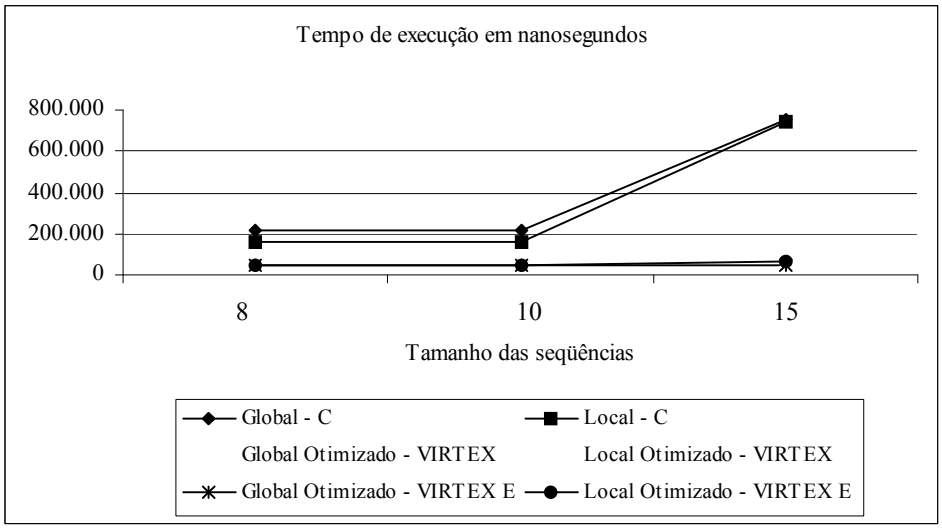

Figura 4 Comparação das implementações em *software* e *hardware*

Nesta figura 4 é possível observar que a execução em *hardware* (usando FPGAs) é mais rápida do que a implementação em *software* em C executando em um computador moderno Pentium IV 1.66 GHz. A nossa implementação em *hardware* somente foi possível para seqüências de tamanho 8, 10 e 15 caracteres, resultado similar ao obtido por Lozano (2006), que fez um processador sistólico em FPGAS da Altera para seqüências de tamanho de até 7 caracteres, obtendo um atraso também similar, de 43 ns.

Após ter realizado as implementações dos algoritmos de alinhamento global e local, das seqüências genéticas de DNA, utilizando a técnica de programação dinâmica, tanto em nível de *software* (em C) quanto em nível de *hardware* (VHDL e prototipados em FPGA) foi possível comparar os nossos dados com aqueles obtidos para outras plataformas. A tabela 9 oferece informações referentes aos caracteres processados por segundo (CPS – *Characters Processed per Second*) para a implementação do algoritmo BioSCAN (também implementado em C) em plataformas paralelas.

Tabela 9 CPS - Caracteres Processados por Segundo [Hoffman 1993].

Sistemas	Tamanho das seqüências					
	10	**50**	**100**	**500**	**1000**	**5000**
BioSCAN	1.979.583	1.979.583	1.979.583	1.979.583	1.979.583	1.979.583
Sun 690	5.776.729	1.235.597	625.385	125.359	62.629	12.489
Convex 240	6.423.184	2.953.318	1.733.355	371.772	189.344	39.535
MasPar-1	1.162.529	1.160.915	1.160.850	1.161.847	1.161.107	1.154.186

Nós utilizamos a seguinte consideração: Se para uma seqüência de 10 caracteres, o circuito precisou um tempo medido em nanosegundos, então, calculamos quantos caracteres ele consegue processar em um segundo. No algoritmo global, usando a família VIRTEX V800FG680, obteve-se um tempo de 45.902 ns.

Então: (**1 seg * 10 caracteres / 45.902 ns) = 10/45.902*10^{-9} = 217.864.923 CPS.**

Comparados com os sistemas da tabela 4.6, onde aparece o valor de CPS para as respectivas implementações desses algoritmos em linguagem de alto nível e plataformas especiais de *hardware*, e observando o mesmo tamanho de seqüência de 10 caracteres, podemos dizer que a nossa implementação em *hardware* mostrou-se mais rápida do que os sistemas apresentados na tabela 10.

Tabela 10 Comparação com Sistemas Conhecidos.

Sistemas	CPS	Quantidade de vezes mais lento
Morony	**217.864.923**	1
BioSCAN	1.979.583	110
Sun 690	5.776.729	37.7
Convex 240	6.423.184	33.9
MasPar-1	1.162.529	187.4
Sistólico [Lozano, 2006]	217.864.923	1

Finalmente, ao comparar os dados em *software* e *hardware* neste artigo, nota-se a diferença significativa nos tempos de execução, visto que as performances em *hardware* são melhores do que as versões em *software*. Não obstante, os bons resultados obtidos, seria interessante analisar seqüências com tamanhos variados e maiores em *hardware*, já que não se conseguiu que a ferramenta *Xilinx* suportasse a implementação para tamanhos de seqüências maiores a 15 caracteres.

5. Conclusões

Neste trabalho foram implementados, tanto em *software* usando a linguagem de programação C quanto em *hardware* (usando VHDL e FPGA), os algoritmos de alinhamento global e local, ou melhor, conhecidos como *Needleman-Wunsch* e *Smith-Waterman*, respectivamente. Com os dados de desempenho obtidos (usando o tempo de execução) desses algoritmos, a versão em nível de *hardware* usando VHDL e FPGAs e sintetizando o circuito com a ferramenta *Xilinx Series Fundation* é mais rápida do que uma implementação em nível de *software* utilizando a linguagem de programação C e computadores pessoais modernos, tais como Pentium IV 1.66 GHz.

Em *software*, foram comparadas seqüências com tamanhos de até 1000 caracteres e em FPGA o *hardware* suportou apenas tamanho com 15 caracteres. Para seqüências de até 300 caracteres, a nossa implementação em *software* obteve desempenho razoável (tempo de execução comprável aos obtidos em máquinas paralelas). Já para tamanhos maiores, as outras implementações paralelas são bem melhores do que a nossa seqüencial, motivando o uso de plataformas e soluções paralelas para tamanhos de grandes seqüências.

Foi possível implementar os algoritmos acima descritos utilizando tecnologia FPGA da Xilinx, porém somente as placas VIRTEX e VIRTEXE conseguiram suportar esses algoritmos, onde as seqüências de DNA analisadas e comparadas conseguiram suportar seqüências com tamanho máximo de 15 caracteres. A utilização de recursos espaciais do FPGA é pequena, o que sugere a possibilidade de usar a implementação para seqüências de tamanho maior.

Observando os resultados obtidos, percebe-se que a implementação em *hardware* apresentou tempo de execução mais rápido que alguns sistemas amplamente usados, tais como: BioSCAN, Sun 690, Convex 240 e o MasPar-1 que implementam os algoritmos em linguagem de alto nível e os executam em *hardware* com maior capacidade de processamento.
Portanto, FPGAs parecem ser uma alternativa para resolver o problema dos algoritmos destinados ao alinhamento e a comparação de seqüências genéticas, uma vez que esta tecnologia, FPGA, permite baixos custos e ganho em tempo de execução.
Como trabalhos futuros, sugere-se realizar mais otimizações, paralelizar os cálculos, com o objetivo de diminuir os recursos espaciais do FPGA e aumentar a velocidade de processamento dos algoritmos de seqüenciamento genético. Seria interessante propor novos algoritmos e/ou implementações, a fim de contribuir para solucionar o problema do seqüenciamento de gens, implementações como por exemplo: mais placas FPGAs funcionando em paralelo. Também propor outras técnicas para otimizar a velocidade de processamento como *pipeline*, técnicas de paralelismo e o uso de outras tecnologias como por exemplo SoC - *System on Chip*.
Seria interessante também propor novas arquiteturas, tais como: sistólica, MIMD e realizar as respectivas implementações em FPGAs, assim como fazer comparações e a respectiva análise de desempenho.

6. Referências

ALTSCHUL, S. F. et al. (1990) "**Basic local alignment search tool**", Journal of Molecular Biology, 215:403–410.

ARNOLD, J. M. BUELL, D.A. and DAVIS, E.G. (1992) "**Splash-2**", In Proc. 4th Annual ACM Symposium on Parallel Algorithms and Architectures, p. 316-322.

BERTIN, P., RONCIN, D. and VUILLEMIN, J. (1992) **Programmable active memories: a performance assessment**. In F. Meyer, B. Monien, and A.L. Rosenberg, editors, Parallel Architectures and their efficient use, p. 119-130. LNCS, October.

CARVALHO, L. G. A. (2003) "U**ma abordagem em hardware para algoritmos de comparação de seqüências baseados em programação dinâmica**", Dissertação de Mestrado, Departamento de Ciência da Computação, Universidade de Brasília – DF, Brasil, Dezembro.

CHOW, E. et al. (1991) **Biological information signal processor**. In: ASAP, (Valero, M. et al., eds) p.144-160, Los Alamitos, CA: IEEE Computer Society.

GOKHALE, M. B. et al. (1990) "**SPLASH: A reconfigurable linear logic arrays**", In Proceedings of the International Conference on Parallel Processing, p.526-532, Agosto.

GUERDOUX-JAMET, P. and LAVENIER, D. (1995) "**Systolic filter for fast dna similarity search**", In ASAP'95, Strasbourg.

HOFFMAN, D. L. (1993) "**A Comparison of the BioSCAN Algorithm on Multiple Architectures**", Department of Computer Science. University of North Carolina at Chapel Hill. 27599-3175.

HUGHEY, R. and LOPRESTI, D. P. (1991) "**B-SYS: A 470-processor programmable systolic array**", Proceedings of the International Conference Parallel Processing, p.580-583.

HUGHEY, R. (1991) "**Programmable Systolic Arrays**", PhD thesis, Departament of Computer Science, Brown University, Providence, RI.

JEANMOUGIN, F. et al. (1998) "**Multiple sequence alignment with CLUSTAL X**", Trends Biochem Sci, 23:403–405, Outubro de 1998. Disponível em: < http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=PubMed&list_uids=9810230&dopt=Abstract >, Dezembro de 2004.

LANG, H. (1986) "**The instruction systolic array a parallel architecture for vlsi**", Journal of VLSI , 1:65–74.

LAVENIER, D. (1996) "**Dedicated Hardware for Biological Sequence Comparison**", Journal of Universal Computer Science, 2(2):77-86.

LAVENIER, D. (1996) "**SAMBA: Systolic Accelerators for Molecular Biological Applications**", Technical Report 988 IRISA 35042 Rennes Cedex, França.

LOZANO, Martin A.; MEDINA, Jaime Velasco (2006). "**Diseño de un procesador para el alineamiento global de secuencias de DNA**". Iberchip Workshop, Costa Rica, p. 1-5, Marzo.

MATTOS, N. P. e RASKIN, S. F. (1994) "**Aspectos da Arquitetura de Processadores Risc. Um comparativo RISC X PC**", Disponível em: <http://www.pr.gov.br/batebyte/edicoes/1994/bb35/aspectos.htm>, Novembro de 2004.

NEEDLEMAN S. B. and WUNSCH, C. D. (1970) "**A general method applicable to the search for similarities in the amino acid sequence of two proteins**", Journal Molecular Biology, 48, p. 443-453.

OLIVEIRA, D. C. (2002) "**Alinhamento de Seqüências**". Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação. Lavras - Minas Gerais – Brasil.

ORDONEZ, E. D. M. et al. (2003) "**Projeto, Desempenho e Aplicações de Sistemas Digitais em Circuitos Programáveis (FPGAs)**", Pompéia: Bless, 300p.

RAIMBAULT, F. et al. (1993) "**Fine grain parallelism on a MIMD machine using FPGA's**", Research Repport INRIA N.

SMITH, T. and WATERMAN, M. (1981) "**Identification of common molecular subsequences**", Journal of Moleular Biology, 147:195–197.

TANGEN, U., r MCCASKILL, J. S. (1998) "**Hardware Evolution with a Massively Parallel Dynamically Reconfigurable Computer: POLYP**", In: International Conference On Evolvable Systems: From Biology To Hardware, 2., 1998, Lausanne. Proceedings... Berlin: Springer-Verlag (Lecture Notes in Computer Science v. 1478), p.364-371.

VIANNA, C. J. M. (2003) "**Arquiteturas de Computadores para alinhamento de seqüências biológicas**", Relatório Técnico da Disciplina de Arquiteturas Avançadas. Departamento de Computação e Estatística – Centro de Ciências Exatas e Tecnologia. Universidade Federal de Mato Grosso do Sul, Julho.

WARD, R. J., GALBAN, V. D. and RUIZ, J. C. (2004) "**Tópicos em Bioinformática. Introdução à Análise de Seqüências**", Faculdade de Medicina de Riberão Preto – FRMP – USP. Departamento de Bioquímica e Imonologia. Disponível em: <http://rbq.fmrp.usp.br/bioinfo/bioinfo1.htm>, Julho.

WHITE, C. T. et al. (1991) **BioSCAN: A VLSI-basead system for biosequence analysis**. Proceedings of the International Conference on Computer Design: VLSI in Computers & Processors, p. 504-509.

XILINX. (2006) **The Programmable Gate Array Data Book**. 2100 Logic Drive, San Jose, CA 95124 USA.