UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL

LEONARDO GOMES TAVARES

ABORDAGEM BASEADA EM LÓGICA RECONFIGURÁVEL POR *HARDWARE* PARA O PROBLEMA DA DETECÇÃO DE GENES

DISSERTAÇÃO

CURITIBA

2010

LEONARDO GOMES TAVARES

ABORDAGEM BASEADA EM LÓGICA RECONFIGURÁVEL POR *HARDWARE* PARA O PROBLEMA DA DETECÇÃO DE GENES

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Ciências" – Área de Concentração: Informática Industrial.

Orientador: Prof. Dr. Carlos R. Erig Lima

Co-orientador: Prof. Dr. Heitor Silvério Lopes

CURITIBA

2010

Dados Internacionais de Catalogação na Publicação

T231d Tavares, Leonardo Gomes
Abordagem baseada em lógica reconfigurável por *hardware* para o problema da detecção de genes / Leonardo Gomes Tavares. – 2010.
101 p. : il. ; 30 cm

Orientador: Carlos R. Erig Lima
Co-orientador: Heitor Silvério Lopes
Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2010.
Bibliografia: p. 90-94

1. Bioinformática. 2. Genes – Detecção. 3. Computação reconfigurável. 4. Engenharia elétrica - Dissertações. I. Lima, Carlos R. Erig, orient. II. Lopes, Heitor Silvério, co-orient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD 621.3

Biblioteca Central da UTFPR

UTFPR
CAMPUS CURITIBA

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Curitiba

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL
CPGEI 22 anos

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

Título da Dissertação Nº 527:

"Abordagem baseada em lógica reconfigurável por hardware para o problema de detecção de genes"

por

Leonardo Gomes Tavares

Esta dissertação foi apresentada, às 14h do dia 23 de abril de 2010, como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Informática Industrial, ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR – Campus Curitiba. O trabalho foi aprovado pela Banca Examinadora.

Prof. Carlos Raimundo Erig Lima, Dr.
(Presidente – UTFPR)

Profª. Nádia Nedjah, Drª.
(UERJ)

Prof. Jean Marcelo Simão, Dr.
(UTFPR)

Visto da coordenação:

Prof. Humberto Remígio Gamba, Dr.
(Coordenador do CPGEI)

Dedico à minha família.

AGRADECIMENTOS

Agradeço primeiramente a Deus pela vida, pela criação, pela salvação e por promover em nós tanto o querer como o realizar.

Aos meus pais, Oziel e Ivone, pela educação, pelo cuidado e apoio dado não somente agora, mas em todas as vezes que fomos buscar a realização dos nossos sonhos.

À minha esposa Carollina, pela compreensão nos momentos de ausência, pelo amor, carinho e apoio incondicionais demonstrados inúmeras vezes no período do desenvolvimento deste trabalho.

À minha filha Aline, que chegou há um ano e trouxe muita alegria à nossa casa.

Ao meu orientador, o Prof. Dr. Carlos Erig Lima, pela oportunidade, confiança e pelas palavras de incentivo nas horas certas.

Ao meu co-orientador, o Prof. Dr. Heitor Silvério Lopes, pelo suporte, direcionamento e inúmeras correções.

Aos prezados professores membros da banca, por aceitarem o convite e pelas preciosas contribuições.

A todos os colegas do laboratório de Bioinformática, pelas idéias, sugestões e críticas sempre bem colocadas.

Finalmente, agradeço a todos os que de alguma forma, contribuiram direta ou indiretamente para que este trabalho pudesse ser concluído.

“Se você pensa que pode ou se você pensa que não pode, de qualquer forma, você tem toda a razão.” (Henry Ford)

RESUMO

TAVARES, L. G. Abordagem baseada em lógica reconfigurável por *hardware* para o problema da detecção de genes. 101 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2010.

O volume de dados gerados pelos vários projetos de sequenciamento genômico tem aumentado drasticamente nos últimos anos. Para processar, adequadamente, toda essa massa de dados são necessárias ferramentas eficientes a fim de se extrair informações genéticas úteis. Algumas das informações mais importantes neste processo são conhecidas como genes. A identificação computacional de genes de organismos eucariotos ainda é um problema em aberto, e é uma das principais tarefas em bioinformática atualmente. O presente trabalho investiga as principais técnicas utilizadas na tarefa de identificação de genes e propõe a aplicação de algumas delas, em uma arquitetura híbrida, baseada em *software* embarcado e *hardware* reconfigurável utilizando FPGA. O principal objetivo é desenvolver uma ferramenta que possa reduzir o tempo de execução do processo da identificação de genes em sequências de DNA explorando o paralelismo inerente dos modelos adotados e descrevendo-os em uma linguagem de descrição de *hardware*. Dentre as principais contribuições do presente trabalho estão a implementação de sensores de sinais em *hardware* reconfigurável e a proposta de um modelo para gerenciamento dos sensores. Os resultados obtidos comprovam a adequação do modelo proposto ao problema em questão e abre novas possibilidades ao uso da computação reconfigurável e dos sistemas embarcados na solução dos problemas de bioinformática.

Palavras-chave: Bioinformática, Genes, DNA, Computação Reconfigurável, FPGA.

ABSTRACT

TAVARES, L. G. Reconfigurable hardware-based approach for the gene prediction problem. 101 f. Dissertação – Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná. Curitiba, 2010.

The amount of data produced by the several genomic sequencing projects has increased dramatically in recent years. To process all this data, effective tools to extract useful genetic information are needed. One of the most important information in this process are known as genes. The computational identification of genes of eukaryotic organisms is still an open problem, and it is a major task in bioinformatics today. This study investigates the main techniques used in the task of identifying genes, and proposes the use of some of them, in a hybrid architecture, based on embedded software and reconfigurable hardware using FPGA. The main goal is to develop a tool that can reduce the runtime of the process of identifying genes in DNA sequences, exploring the parallelism inherent in the models adopted, and describing them in a hardware description language. Among the main contributions of this work are the implementation of sensor signals in reconfigurable hardware and the proposal of a model for managing the sensors. The results confirm the suitability of the proposed model to the problem and open new possibilities to the use of reconfigurable computing and embedded systems for solving bioinformatics problems.

Keywords: Bioinformatics, Genes, DNA, Reconfigurable Computing, FPGA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

DNA	*DeoxyriboNucleic Acid*
RNA	*RiboNucleic Acid*
ORF	*Open Reading Frame*
WMM	*Weight Matrix Model*
RNA	Rede Neural Artificial
HMM	*Hidden Markov Model*
TBP	*Three-Base Periodicity*
CHMM	*Class Hidden Markov Model*
MORGAN	*Multi-frame Optimal Rule-based Gene Analyzer*
MZEF	*Michael Zhangs Exon Finder*
ROC	*Receiver Operating Characteristics*
ASIC	*Application-Specific Integrated Circuit*
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language*
VHDL	*Very High Speed Integrated Circuit Hardware Description Language*
IEEE	*Institute of Electrical and Electronic Engineer*
IMM	*Interpolated Markov Model*
HRE	*Hormone Response Element*
FPGA	*Field Programmable Gate Array*
GPU	*Graphics Processing Unit*
ROM	*Read Only Memory*
RAM	*Random Access Memory*
UART	*Universal Asynchronous Receiver Transmitter*
PC	*Personal Computer*

RISC	*Reduced Instruction Set Computer*
CPU	*Central Processing Unit*
IDE	*Integrated Development Environment*
DMIPS	*Dhrystone Millions of Instruction per Second*
JTAG	*Joint Test Action Group*
LUT	*Look up Table*
SDRAM	*Synchronous Dynamic Random Access Memory*
SRAM	*Synchronous Random Access Memory*
PLL	*Phase Locked Loop*
MDD	*Maximal Dependence Decomposition*
WWAM	*Windowed Weight Array Model*
HS3D	*Homo Sapiens Splice Sites Data Set*

SUMÁRIO

1 INTRODUÇÃO

O objetivo deste capítulo é fornecer uma introdução sucinta sobre as motivações, os objetivos e a forma de organização deste trabalho.

Na seção 1.1 são descritos os fatores motivadores que justificam a proposta do presente trabalho. A seção 1.2 expõe os objetivos pretendidos e na seção 1.3 é apresentada a organização dos capítulos que compõem esta dissertação de mestrado.

1.1 MOTIVAÇÕES

Os diversos projetos de sequenciamento genômico, principalmente do genoma humano, tem contribuído nos últimos anos para um incremento considerável no volume dos dados dos repositórios de sequências biológicas. O histórico do GenBank (BENSON et al., 2008), um dos mais importantes repositórios atualmente, mostra que o número de sequências armazenadas cresce em ritmo exponencial, conforme mostrado na Figura 1.

As sequências armazenadas na sua forma bruta, entretanto, não possuem sentido imediato e, por isso, precisam ser processadas adequadamente de forma a se obter informações genéticas úteis. Este processo inclui várias fases. Uma das mais importantes corresponde a uma análise das sequências de DNA na busca por trechos responsáveis pela síntese de proteínas, conhecidos como genes.

A identificação de genes em sequências de DNA é uma tarefa muito custosa quando realizada por meios laboratoriais. Uma alternativa a esse processo é a utilização de técnicas computacionais aplicadas às sequências de DNA previamente amostradas e armazenadas em bancos de dados. Este processo computacional, conhecido como detecção de genes, consiste no desenvolvimento de um algoritmo que receba como entrada uma sequência de DNA e produza como saída uma tabela contendo a localização e a estrutura de todos os genes presentes nesta sequência (FICKETT; TUNG, 1992).

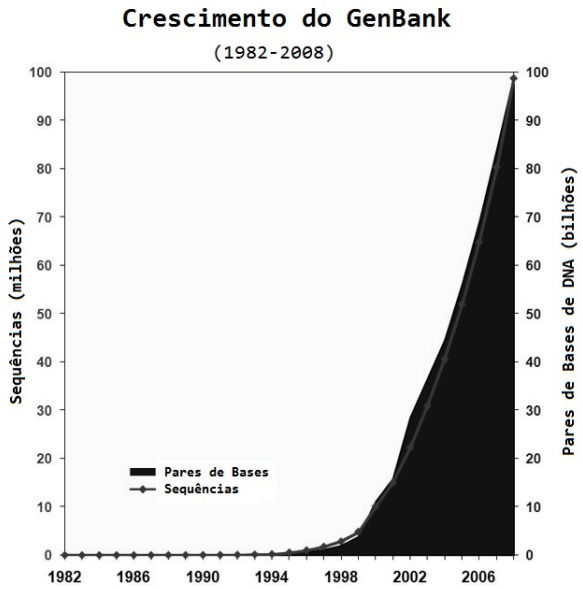


Figura 1: Crescimento do GenBank (BENSON et al., 2008)

Mesmo utilizando as mais recentes ferramentas computacionais, o processo de detecção de genes tende a ser demorado, principalmente quando são analisadas grandes quantidades de dados. Estas ferramentas são executadas em arquiteturas de computadores de propósito geral e normalmente não utilizam aceleradores ou co-processadores para reduzir o tempo de análise.

Uma das abordagens da detecção de genes, chamada de busca por sinais, possui uma tarefa dentro do seu processo, que é a varredura da sequência de teste por diferentes sensores. Esta tarefa possui um potencial paralelismo que não é aproveitado nas ferramentas tradicionais de *software*.

A detecção de genes é ainda um problema em aberto e de grande importância no estudo da evolução das espécies, síndromes, doenças genéticas e outros.

1.2 OBJETIVOS

O objetivo geral deste trabalho é propor uma arquitetura, baseada na lógica reconfigurável, que seja capaz de tirar proveito do potencial paralelismo da tarefa de varredura, reduzindo consequentemente o tempo de análise das sequências de DNA na detecção de genes.

Os objetivos específicos do presente trabalho são:

- descrição de sensores de sinais, baseados em matrizes de pesos, em *hardware* reconfigurável;

- proposta de uma arquitetura paralela, com *pipeline*, destinada à execução dos sensores de sinais, em *hardware* reconfigurável;
- validação da arquitetura proposta por meio de comparações com outras ferramentas.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação está organizada em cinco capítulos.

O capítulo 2 apresenta uma revisão dos conceitos biológicos envolvidos, as principais técnicas e ferramentas utilizadas na detecção de genes e conceitos de computação reconfigurável.

O capítulo 3 descreve em detalhes a escolha da abordagem, a proposta e implementação da arquitetura baseada em *hardware* reconfigurável.

O capítulo 4 expõe como foram realizados os experimentos, apresenta os resultados obtidos e a validação da arquitetura proposta.

O capítulo 5 apresenta a discussão dos resultados alcançados, as conclusões do trabalho e registra as principais oportunidades para continuidade deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados conceitos biológicos, estratégias, métodos, ferramentas e métricas relacionadas à tarefa de detecção de genes e a computação reconfigurável, os quais se apresentam como referencial teórico para o desenvolvimento do presente trabalho.

2.1 CONCEITOS BIOLÓGICOS

Nesta seção são apresentados os principais conceitos da biologia molecular que têm relação com a tarefa de detecção de genes. São abordados o DNA, o RNA, o dogma central da biologia molecular e os genes.

2.1.1 DNA

Todo organismo vivo armazena sua informação biológica na forma de ácidos nucleicos, que são formados por unidades básicas chamadas de nucleotídeos. Cada nucleotídeo é constituído de uma molécula de açúcar (desoxirribose ou ribose), um grupo fosfato e uma base nitrogenada. Podem ser de 5 tipos: Adenina (‘A’), Timina (‘T’), Citosina (‘C’), Guanina (‘G’) e Uracila (‘U’).

O DNA, ou Ácido Desoxirribonucleico (do inglês, *DeoxyriboNucleic Acid*), é uma longa molécula, formada por um polímero de nucleotídeos, que contém informações necessárias ao desenvolvimento e a manutenção da vida dos organismos, e armazena as informações genéticas codificadas que são transmitidas de geração a geração.

De maneira simplificada, o DNA pode ser representado como uma fita dupla complementar e antiparalela (Figura 2), sendo por meio dos nucleotídeos que as duas fitas componentes permanecem ligadas. O DNA possui o esquema de emparelhamento onde o nucleotídeo ‘A’ (Adenina) sempre se liga ao nucleotídeo ‘T’ (Timina) e o nucleotídeo ‘C’ (Citosina) sempre se liga ao nucleotídeo ‘G’ (Guanina). A base ‘U’ (Uracila) não compõe o DNA.

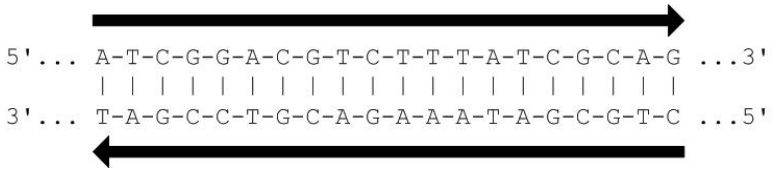


Figura 2: Representação simplificada do DNA

Por convenção, uma molécula de DNA começa no lado 5' e termina no lado 3', onde 5' e 3' correspondem aos átomos de carbono livres da molécula de açúcar presentes em cada nucleotídeo.

2.1.2 RNA

As moléculas de RNA, ou Ácido Ribonucleico (do inglês, *RiboNucleic Acid*), possuem composição estrutural e química muito similares às moléculas de DNA. A principal diferença química reside no fato de que o nucleotídeo 'T' (Timina) do DNA é substituído por 'U' (Uracila) no RNA. As demais bases nitrogenadas são as mesmas que compõem o DNA: 'A' (Adenina), 'C' (Citosina) e 'G' (Guanina).

Estruturalmente, os RNAs normalmente são moléculas de fita única e bem menores que o DNA. Os RNAs podem ser de três tipos: o RNAm (mensageiro) que é aquele RNA que contém a sequência que codifica uma proteína; o RNAt (transportador) que carrega os aminoácidos até os ribossomos, e o RNAr (ribossômico) que faz parte da estrutura dos ribossomos.

A molécula de RNA se comporta como um intermediário no fluxo de informações dentro da célula, do DNA às proteínas.

2.1.3 Dogma Central da Biologia Molecular

O DNA e as proteínas são macromoléculas que desempenham um papel essencial à vida. A informação genética, que está armazenada no DNA, é simplificadamente, uma sequência constituída pelos quatro tipos de nucleotídeos ('A', 'C', 'G' e 'T'), e o seu mecanismo de transmissão da informação se chama replicação. Por outro lado, são as proteínas que realizam as funções vitais da célula. Sendo assim, é necessário que os quatros tipos de nucleotídeos sejam traduzidos para os 20 tipos de aminoácidos conhecidos. Isto é feito basicamente através de dois processos: a transcrição que converte a informação do DNA em uma forma mais acessível (uma fita de RNA complementar), e a tradução que converte a informação contida no RNA em proteínas (Figura 3).

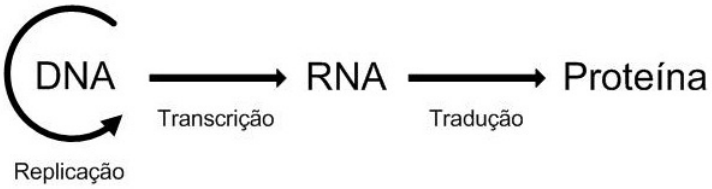


Figura 3: Dogma central da Biologia Molecular

O fluxo de informação para gerar o RNA a partir do DNA, e uma proteína a partir do RNA, juntamente com o processo de replicação do DNA, formam o dogma central da biologia molecular (KANEHISA, 2000).

Replicação

A replicação do DNA é o processo de auto-duplicação do seu material genético que possibilita a transmissão das características hereditárias de um indivíduo para gerações futuras.

O início do processo de replicação do DNA se dá com o desenrolamento das fitas seguido pela separação das mesmas, por ação de uma enzima denominada DNA helicase.

À medida que as duas fitas do DNA se separam, forma-se uma região de replicação denominada zona de replicação ou forquilha de replicação e que se move em ambas as direções ao logo da molécula de DNA. Iniciam-se as replicações em ambas as fitas. Em uma fita a replicação acontece de forma contínua e na outra de forma descontínua.

O processo de replicação do DNA envolve a participação de várias enzimas. O reconhecimento da região de origem, o mecanismo de abertura das fitas e a cópia do molde sintetizando uma nova fita complementar são alguns exemplos de atividades realizadas por diferentes enzimas.

Transcrição

A Transcrição é o processo de formação de uma fita de RNA complementar a uma região do DNA, como pode ser observado resumidamente na Figura 4.

A enzima responsável pela transcrição é a RNA polimerase. Para iniciar a trancrição, a RNA polimerase deve reconhecer um local específico onde começará a síntese. Esse local chama-se região promotora, ou simplesmente promotor. Os promotores contêm sequências localizadas antes do início da transcrição, a distâncias específicas.

Nos promotores de organismos procariotos (organismos mais simples que não possuem

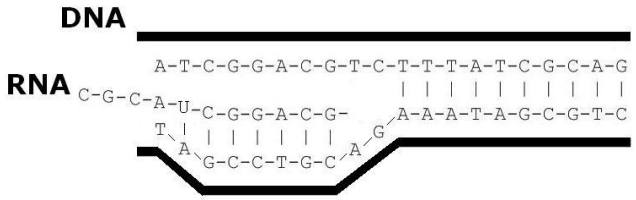


Figura 4: Processo de transcrição

envoltório nuclear, como bactérias, e têm seu material genético disperso na célula) encontram-se sequências comuns geralmente localizadas nas regiões de aproximadamente 10 e 35 pares de bases de distância do ponto onde ocorre o início da transcrição.

Nos eucariotos (organismos complexos, como os humanos, que têm uma membrana que envolve o núcleo de suas células, e seu material genético está delimitado dentro desse núcleo) o processo de iniciação e regulação da transcrição é muito mais complexo e envolve um número maior de sequências promotoras e de fatores de transcrição.

A RNA polimerase liga-se ao promotor e inicia a síntese da fita de RNA complementar até parar em uma região chamada terminador. Em procariotos, a transcrição ocorre no mesmo lugar onde ocorre a tradução. Dessa forma, tão logo o RNA comece a ser formado, a tradução já se inicia. Por isso, diz-se que a transcrição e a tradução nos procariotos é acoplada. Nos eucariotos a transcrição ocorre no núcleo e a tradução ocorre no citoplasma. O RNA recém-sintetizado nos eucariotos ainda passa por várias modificações antes de estar efetivamente concluído o procedimento.

Tradução

A tradução converte a informação na forma de trincas de nucleotídeos (chamados de códons) em aminoácidos, que quando ligados formarão uma proteína.

Os aminoácidos são moléculas orgânicas que contém átomos de carbono, hidrogênio, oxigênio e nitrogênio em sua composição. São 20 os mais comumente encontrados na natureza e universalmente aceitos como os utilizados para a síntese de proteínas. A Tabela 1 mostra a lista destes aminoácidos. A primeira coluna desta tabela mostra o nome dos aminoácidos, a segunda coluna a abreviatura e a terceira os possíveis códons que são traduzidos nos respectivos aminoácidos.

O processo de tradução se inicia quando o ribossomo (Figura 5) encontra, no RNAm, o códon de início de tradução AUG.

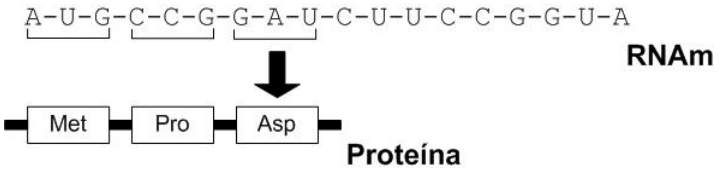


Figura 5: Processo de tradução

A cada códon lido do RNAm, a tradução é realizada conectando o aminoácido correspondente na cadeia que está se formando. A relação entre códons e aminoácidos envolvidos na tradução pode ser vista na Tabela 1. A tradução encerra quando o ribossomo encontra um dos possíveis códons de terminação, que não codificam nenhum aminoácido, e são eles: UGA, UAG e UAA.

Após a tradução as proteínas ainda têm que passar por algumas modificações para que possam exercer adequadamente as suas funções.

Tabela 1: Aminoácidos, abreviaturas e códons

Aminoácido	**Abreviatura**	**Códons**
Alanina	Ala	GCU, GCC, GCA, GCG
Cisteína	Cys	UGU, UGC
Ácido aspártico	Asp	GAU, GAC
Ácido glutâmico	Glu	GAA, GAG
Fenilalanina	Phe	UUU, UUC
Glicina	Gly	GGU, GGC, GGA, GGG
Histidina	His	CAU, CAC
Isoleucina	Ile	AUU, AUC, AUA
Lisina	Lys	AAA, AAG
Leucina	Leu	UUA, UUG, CUU, CUC, CUA, CUG
Metionina	Met	AUG
Asparagina	Asn	AAU, AAC
Prolina	Pro	CCU, CCC, CCA, CCG
Glutamina	Gln	CAA, CAG
Arginina	Arg	CGU, CGC, CGA, CGG, AGA, AGG
Serina	Ser	UCU, UCC, UCA, UCG, AGU, AGC
Treonina	Thr	ACU, ACC, ACA, ACG
Valina	Val	GUU, GUC, GUA, GUG
Tiptofano	Trp	UGG
Tirosina	Tyr	UAU, UAC
STOP		UAG, UGA, UAA

2.1.4 Genes

Genes são fragmentos de DNA cromossômico capazes de determinar a síntese de uma cadeia polipeptídica, que normalmente trata-se de uma proteína. As proteínas controlam não só a estrutura e as funções metabólicas das células, mas também todo o organismo.

A estrutura de um gene é complexa. Inclui sequências que participam de diversos processos como início e fim da transcrição e tradução e outros. Para fins didáticos é possível simplificar sua estrutura e considerar apenas as seguintes regiões características:

1. Região promotora (região de reconhecimento)
2. Região não-traduzida 5', que regula a transcrição gênica
3. Início de tradução, ou códon de início (*start codon*)
4. Região codificadora de proteína. Nos genes de organismos procariotos existe uma única região. Entretanto, nos eucariotos existem dois tipos de região chamados éxons e íntrons. Os éxons são usados na codificação de proteínas, enquanto os íntrons, que não participam da codificação, são removidos.
5. Parada de tradução, ou códon de parada (*stop codon*).
6. Região não-traduzida.
7. Região de poliadenilação rica em Adenina (polyA). Presente apenas nos genes de organismos eucariotos.

A figura 6 mostra as principais regiões características de um gene eucarioto.

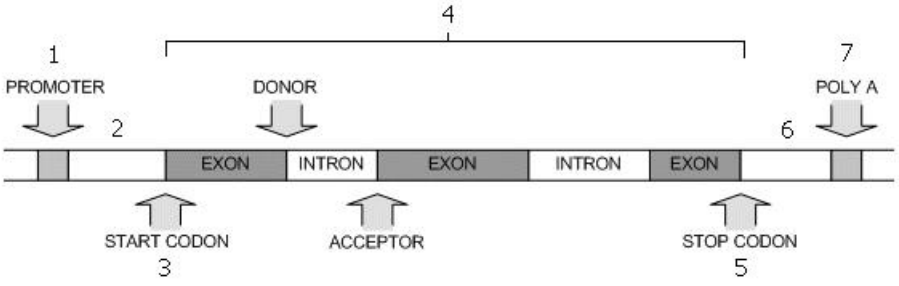


Figura 6: Regiões características de gene eucarioto

A principal diferença entre a expressão dos genes eucariotos e procariotos é o processo de *splicing*. Isto se deve ao fato de que os genes eucariotos são compostos pelos dois tipos de segmentos: éxons e íntrons. Durante o *splicing* os íntrons são removidos. Em geral, a função dos íntrons ainda não está totalmente esclarecida. O processo de *splicing* está representado na Figura 7.

Splice junctions são os pontos de fronteira onde o processo de *splicing* ocorre. A transição de um éxon para um íntron é comumente chamado junção éxon/íntron (EI). Da mesma forma, a transição de um íntron para um éxon é chamada de junção íntron/éxon (IE). O fato que faz com

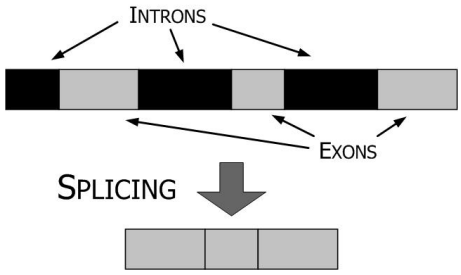


Figura 7: Representação do processo de *splicing*

que o processo de *splicing* ocorra é a presença de dois sinais na cadeia: um sinal chamado de *donor* para a junção éxon/íntron e *acceptor* para a junção íntron/éxon.

2.2 ESTRATÉGIAS PARA DETECÇÃO DE GENES

Depois de mapear e sequenciar uma parte do DNA, se faz necessário compreender a sua função. A busca por genes e outras estruturas biologicamente importantes são algumas das tarefas mais importantes da análise de sequências de DNA.

A localização de genes e a detecção de padrões são difíceis porque existe um número muito grande de interações entre o DNA e a proteína que ainda não foram comprovadas experimentalmente, e ainda diferem de um organismo para outro.

No nível de DNA, existem basicamente três estratégias para a identificação de genes (BAXEVANIS; OUELLETTE, 1998): os métodos baseados na busca por similaridades, os métodos baseados em conteúdo e os métodos baseados na busca por sinal.

2.2.1 Busca por similaridades

É a abordagem mais antiga utilizada na identificação de genes. Seu princípio é baseado na tendência que algumas regiões codificadoras têm de se conservarem ao longo da evolução. Neste método, é feita a busca por regiões similares entre as sequências em estudo e as sequências de um banco de dados conhecido (GISH; STATES, 1993; GELFAND; MIRONOV; PEVZNER, 1996). Essa busca pode ser feita no nível de nucleotídeos ou aminoácidos.

A vantagem deste método é que ao encontrar uma sequência com similaridades com a sequência de teste é possível também estimar a função do novo gene descoberto.

A maior desvantagem é que, neste método, a comparação é feita com genes já conhecidos e, portanto, a qualidade dos resultados é dependente da qualidade do banco de dados utilizado na busca.

O recente crescimento dos bancos de dados biológicos no mundo inteiro tem contribuído para tornar o uso desta abordagem promissora.

2.2.2 Busca por conteúdo

Neste tipo de método procura-se por segmentos que tenham as mesmas propriedades estatísticas das regiões do DNA que codificam proteínas (STADEN; MCLACHLAN, 1982; TIWARI et al., 1997; YAN; LIN; ZHANG, 1998). Algumas medidas que normalmente são utilizadas neste tipo de método são a frequência dos códons utilizados, ocorrência de determinados hexâmeros (6 nucleotídeos), frequência dos aminoácidos e diaminoácidos utilizados.

Para discriminar regiões codificadoras das não-codificadoras são utilizados apenas modelos estatísticos. A busca não depende de bancos de dados para comparação, como nos métodos baseados na busca por similaridade, e essa é a principal vantagem dos métodos baseados em conteúdo.

A principal desvantagem dos métodos estatísticos reside no fato de que a exatidão de seus resultados diminui à medida que o tamanho da janela de busca fica menor.

2.2.3 Busca por sinal

O foco desta estratégia está na verificação da presença ou ausência de sequências específicas envolvidas na expressão gênica, que são chamadas de sinais. A busca por promotores, *donors*, *acceptors*, *start* e *stop codons*, regiões de poliadenilação, entre outros, são utilizados nestes métodos.

Na busca por sinais, o problema da predição de genes pode ser encarado como um conjunto de sub-problemas de predição de diversos sinais biológicos que podem, inclusive, ser tratados isoladamente. Para esta abordagem é comum utilizar o termo "sensor" para designar o subsistema utilizado para detecção de um determinado sinal. Técnicas como a busca por sequência consenso, matrizes de pesos (BURGE, 1997; GERSHENZON; STORMO; IOSHIKHES, 2005), redes neurais (TOWELL; SHAVLIK, 1991; UBERBACHER; MURAL, 1991) e outras são usadas para implementar modelos para reconhecimento destes sinais, os chamados de sensores de sinais.

O objetivo deste tipo de método não é apenas identificar uma sequência como provável região codificadora, mas também fornecer um resultado mais elaborado como a posição exata do início e fim do gene, os éxons que o compõe, etc.

A grande dificuldade para estes métodos é que os sinais nem sempre estão presentes nas sequências de DNA, e quando estão, nem sempre são facilmente reconhecidos.

2.3 PRINCIPAIS MÉTODOS DE DETECÇÃO DE GENES

Nesta seção são apresentados alguns métodos de detecção de genes: busca por ORFs, matrizes de pesos, redes neurais artificiais, modelo oculto de Markov, árvores de decisão e técnicas de processamento digital de sinais.

2.3.1 Busca por ORFs

O método mais simples para encontrar sequências de DNA que codificam proteínas é a busca por ORFs, ou *Open Reading Frames* (MOUNT, 2001). Um ORF é um trecho de sequência de DNA compreendido entre um códon de início de tradução e um códon de parada de tradução e que pode, eventualmente, corresponder a um gene. Para essa busca, existem seis possíveis quadros de leitura em toda sequência, três começando nas posições 1, 2 e 3 e indo da direção 5' para 3' de uma dada sequência, e outras três começando nas posições 1, 2 e 3 e indo da direção 5' para 3' da sequência complementar.

Em genomas de organismos procariotos, as sequências de DNA que codificam proteínas são transcritas no mRNA, e este é traduzido diretamente em proteínas sem modificações significativas. Os ORFs mais longos que vão do primeiro códon de iniciação até o próximo códon de parada no mesmo quadro de leitura geralmente proporcionam um bom método de detecção de região codificante. O incoveniente desta técnica é o elevado número de falsos positivos gerados, isto é, a quantidade de sequências encontradas que não correspondem a genes reais é alta.

Em organismos eucariotos, a predição de genes que codificam proteínas é uma tarefa mais difícil do que em procariotos devido à presença dos íntrons. Como consequência deste fato, um ORF que corresponda a um gene terá a sua sequência interrompida pela presença dos íntrons, que normalmente geram códons de parada.

Três possíveis testes podem ser feitos para verificar se o ORF detectado realmente corresponde a um gene que codifica proteínas. O primeiro teste é baseado em um tipo incomum de variação da sequência que é encontrada em ORFs. A cada 3 bases na sequência, a terceira tende a ser muito mais frequente que as demais (FICKETT, 1982). O programa TESTCODE (Genetics Computer Group) fornece um gráfico com a informação da não-aleatoriedade da terceira base na sequência. A Figura 8 mostra o resultado da análise feita com o programa TESTCODE em uma sequência de 200 pares de bases. No eixo horizontal está representada a extensão da

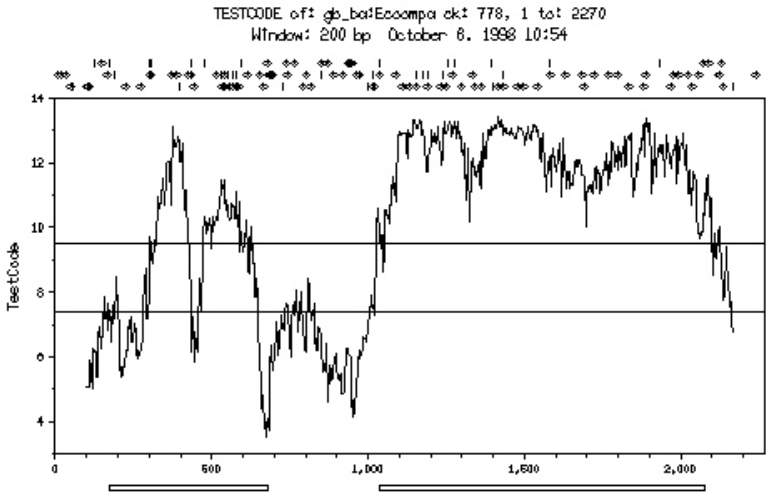


Figura 8: Análise de sequência realizada pelo TESTCODE

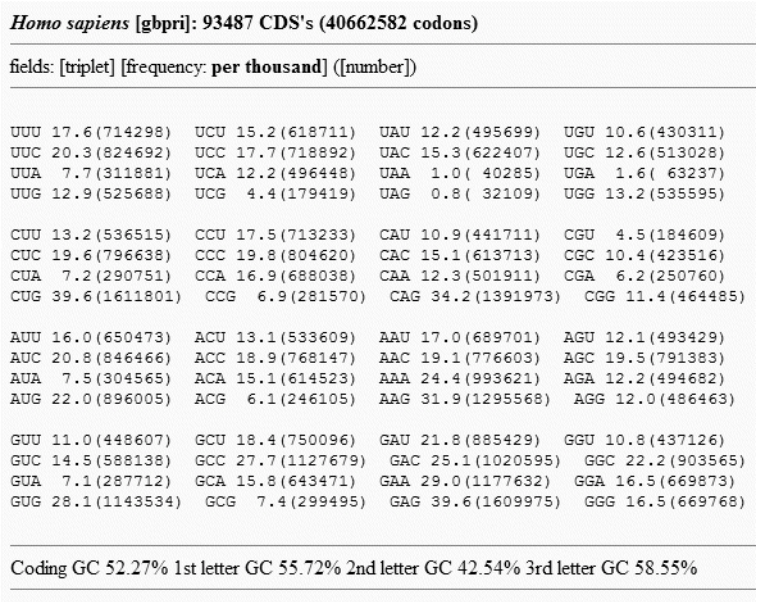
sequência de teste e no eixo vertical o grau de não-aleatoriedade da terceira base para cada posição da sequência de teste. As regiões onde este grau é alto possuem maior probabilidade de serem regiões codificantes.

Na maioria das espécies há uma certa preferência por determinados grupos de códons em relação aos demais, nas regiões que codificam proteínas. Esta característica é a base para o segundo teste. Trata-se de uma análise para determinar se os códons do ORF correspondem àqueles contidos em outros genes do mesmo organismo (STADEN; MCLACHLAN, 1982). Para este teste, a informação sobre a frequência com que os códons aparecem nos genes de um determinado organismo é necessária. A Figura 9 é um exemplo utilizado neste tipo de teste. Nesta tabela são mostrados os códons existentes nas regiões codificantes do Homo Sapiens e a frequência com que cada um dos códons aparece nestas regiões.

No terceiro teste possível, o ORF pode ser traduzido em uma sequência de aminoácidos e a sequência resultante comparada a um banco de dados de proteínas conhecidas. Se uma ou mais similaridades significativas forem encontradas nas sequências comparadas, há então uma boa chance do ORF detectado corresponder a uma região codificante (GISH; STATES, 1993).

2.3.2 Matrizes de Pesos

Um tipo de análise um pouco mais complexa, e utilizada normalmente para detecção de pequenas sequências é o Método da Matriz de Pesos (STADEN, 1984). Este método consiste em construir uma matriz a partir de um alinhamento de um conjunto de sequências previamente identificadas. Cada linha da matriz corresponde a um dos quatro nucleotídeos e cada coluna



Homo sapiens **[gbpri]: 93487 CDS's (40662582 codons)**

fields: [triplet] [frequency: **per thousand**] ([number])

```
UUU 17.6(714298)  UCU 15.2(618711)  UAU 12.2(495699)  UGU 10.6(430311)
UUC 20.3(824692)  UCC 17.7(718892)  UAC 15.3(622407)  UGC 12.6(513028)
UUA  7.7(311881)  UCA 12.2(496448)  UAA  1.0( 40285)  UGA  1.6( 63237)
UUG 12.9(525688)  UCG  4.4(179419)  UAG  0.8( 32109)  UGG 13.2(535595)

CUU 13.2(536515)  CCU 17.5(713233)  CAU 10.9(441711)  CGU  4.5(184609)
CUC 19.6(796638)  CCC 19.8(804620)  CAC 15.1(613713)  CGC 10.4(423516)
CUA  7.2(290751)  CCA 16.9(688038)  CAA 12.3(501911)  CGA  6.2(250760)
CUG 39.6(1611801)  CCG  6.9(281570)  CAG 34.2(1391973)  CGG 11.4(464485)

AUU 16.0(650473)  ACU 13.1(533609)  AAU 17.0(689701)  AGU 12.1(493429)
AUC 20.8(846466)  ACC 18.9(768147)  AAC 19.1(776603)  AGC 19.5(791383)
AUA  7.5(304565)  ACA 15.1(614523)  AAA 24.4(993621)  AGA 12.2(494682)
AUG 22.0(896005)  ACG  6.1(246105)  AAG 31.9(1295568)  AGG 12.0(486463)

GUU 11.0(448607)  GCU 18.4(750096)  GAU 21.8(885429)  GGU 10.8(437126)
GUC 14.5(588138)  GCC 27.7(1127679)  GAC 25.1(1020595)  GGC 22.2(903565)
GUA  7.1(287712)  GCA 15.8(643471)  GAA 29.0(1177632)  GGA 16.5(669873)
GUG 28.1(1143534)  GCG  7.4(299495)  GAG 39.6(1609975)  GGG 16.5(669768)
```

Coding GC 52.27% 1st letter GC 55.72% 2nd letter GC 42.54% 3rd letter GC 58.55%

Figura 9: Ocorrências dos códons em genes do Homo Sapiens

Tabela 2: Tabela de ocorrências do sinal CAP (BUCHER, 1990)

	-2	**1**	**0**	**1**	**2**	**3**	**4**	**5**
#A	49	0	288	26	77	67	45	50
#C	48	303	0	81	95	118	85	96
#G	69	0	0	116	0	46	73	56
#T	137	0	15	80	131	72	100	101

a uma posição do alinhamento. Cada elemento da matriz corresponde ao peso dado para a ocorrência de determinado nucleotídeo na posição do alinhamento em uma sequência teste, de tal forma a fornecer uma medida que quantifica o quanto esta sequência teste se adere ao modelo.

Existem várias abordagens para a construção das Matrizes de Pesos, no entanto, o método mais amplamente utilizado foi proposto por Staden (STADEN, 1984). Utilizando um conjunto de sequências alinhadas de algum sinal (por exemplo, o sinal CAP que está presente em promotores eucariotos), uma tabela de ocorrências dos nucleotídeos é construída através da contagem do número de vezes que cada base ocorre em cada posição. Esta tabela possui quatro linhas (uma linha para cada nucleotídeo: 'A', 'C', 'G' e 'T') e o número de colunas é igual ao comprimento do sinal analisado. Um exemplo de uma tabela de ocorrência para o sinal CAP está ilustrada na Tabela 2 que foi construída por Bucher (BUCHER, 1990), utilizando 303 sequências do sinal CAP.

O número de ocorrências na tabela pode ser convertida em frequências. Por exemplo, se 137 das 303 sequências têm um 'T' na primeira coluna, o frequência de 'T' nesta coluna é 0,45.

Do mesmo modo, para a sinal CAP, um ‘C’ ocorre na segunda coluna com frequência 1,00 e assim por diante.

A matriz de frequências indica a probabilidade de que uma determinada base apareça em cada posição do sinal. Para medir a similaridade de uma nova sequência com o sinal analisado é então necessário multiplicar as probabilidades de cada nucleotídeo em cada posição na matriz conforme a Equação 1. Quanto maior for a similaridade de X com o sinal analisado, maior será o score $P_{WMM}(X)$.

$$P_{WMM}(X) = \prod_{i=1}^{L} p_i(x_i) \tag{1}$$

A matriz de frequências de nucleotídeos também pode ser convertida em um outro formato baseado no logaritmo das frequências. Esta operação resulta no propriamente dito Modelo da Matriz de Pesos (do inglês *Weight Matrix Model*, ou simplesmente WMM). Um exemplo de um WMM, ilustrado na Tabela 3, corresponde ao mesmo sinal ilustrado na Tabela 2 (BUCHER, 1990).

Tabela 3: Matriz de Pesos do sinal CAP (BUCHER, 1990)

	-2	**1**	**0**	**1**	**2**	**3**	**4**	**5**
W(A)	-1.14	-5.26	0.00	-1.51	-0.65	-0.55	-0.91	-0.82
W(C)	-1.16	0.00	-5.21	-0.41	-0.45	0.00	-0.29	-0.18
W(G)	-0.75	-5.26	-5.21	0.00	-4.56	-0.86	-0.38	-0.65
W(T)	0.00	-5.26	-2.74	-0.29	0.00	-0.36	0.00	0.00

A medida de similaridade de uma sequência de teste com o sinal procurado usando o Modelo da Matriz de Pesos se resume a soma algébrica do peso de cada nucleotídeo em cada posição da matriz, conforme a Equação 2.

$$Score(X) = \sum_{i=1}^{L} s_i(x_i) \tag{2}$$

O WMM original, entretanto, não considera a dependência entre as posições (nucleotídeos ou aminoácidos), em uma cadeia de DNA ou proteína. Uma conhecida derivação do WMM, chamada de *Weight Array Model* (WAM) (ZHANG; MARR, 1993) considera as dependências entre posições adjacentes. No WAM, a pontuação final depende de cada posição na sequência para cada palavra com comprimento k (quando k = 1, os dois métodos são os mesmos).

Uma variante da WAM, chamada de *Windowed Second-Order WAM* (WWAM) (BURGE; KARLIN, 1997), tem algumas modificações no processo de treinamento do modelo para reduzir a incidência de erros de amostragem.

Outro modelo muito popular derivado do WMM é o *Maximum Dependence Decomposition* (MDD) proposto por Burge (BURGE; KARLIN, 1997). O MDD é composto por dois componentes: uma árvore de decisão, que primeiramente considera os nucleotídeos em posições importantes com maior influência sobre os demais, e um WMM em cada folha da árvore. A Figura 10 mostra uma representação do MDD. Nesta figura a árvore de decisão está representada na sequência de estruturas condicionais que verificam os nucleotídeos contidos em posições específicas da sequência de entrada ($S[7]$, $S[2]$, $S[1]$ e $S[8]$). Conforme a ocorrência dos nucleotídeos nestas posições é decidido qual WMM deve ser aplicado à sequência de entrada.

Estes modelos citados permitem varrer sequências de DNA na busca por sinais semelhantes ao conjunto inicial de sinais conhecidos, normalmente utilizando um valor de corte ou limiar de pontuação. Entretanto, este limiar de pontuação não é facilmente definido e é fortemente dependente dos dados utilizados.

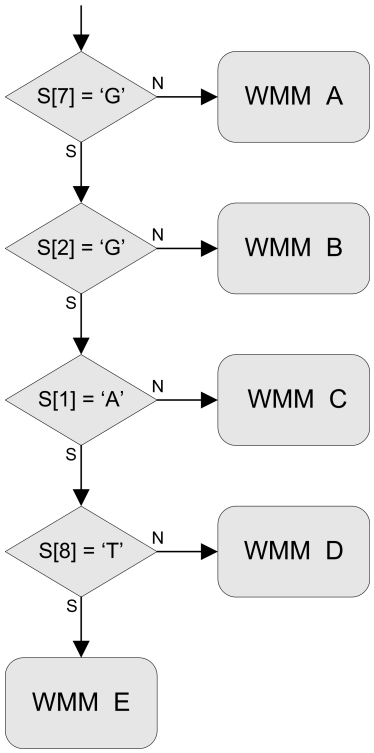


Figura 10: Modelo MDD para detecção de *donors*

Outro fato que sugere a qualidade dos WMMs é a sua ampla utilização em diversas ferramentas de busca por sinais em bioinformática. *Softwares* como Genscan (BURGE; KARLIN, 1997), Twinscan (KORF et al., 2001), N-Scan (GROSS; BRENT, 2005), TigrScan (MAJOROS; PERTEA; SALZBERG, 2004), Glimmer (ARTHUR et al., 1998; DELCHER et al., 1999), e bases de dados, tais como TRANSFAC (WINGENDER et al., 2000), usam WMM e derivações para representar um grande número de sinais.

2.3.3 Redes Neurais Artificiais

Uma rede neural artificial (ou simplesmente RNA) é um modelo computacional inspirado na estrutura paralela e densamente conectada do cérebro humano. São sistemas constituídos de unidades de processamento simples, chamados neurônios, que estão distribuídos e conectados em paralelo e executam funções matemáticas, normalmente não-lineares. Essas unidades são organizadas em camadas e interligadas por um grande número de conexões. Estas conexões estão associadas a pesos e estes armazenam o conhecimento distribuído no modelo.

A Figura 11 mostra um exemplo de uma rede neural utilizada no reconhecimento de promotores de Escherichia Coli (TOWELL; SHAVLIK; NOORDEWIER, 1990).

O aprendizado nas redes neurais artificiais ocorre por meio da apresentação de um conjunto de padrões à rede. Estes padrões são utilizados por um algoritmo de treinamento para, iterativamente, ajustar os pesos das conexões. O objetivo do processo de treinamento é extrair o conhecimento necessário para a resolução do problema em questão. O conhecimento armazenado nos pesos é usado, posteriormente, para gerar a resposta da rede para novos padrões.

As redes neurais apresentam uma série de vantagens sobre outros métodos, como tolerância a dados ruidosos, capacidade de representar funções lineares e não-lineares, e habilidade em lidar com atributos contínuos ou discretos. Dentre as principais desvantagens estão a dificuldade em definir seus parâmetros e a dificuldade de compreensão dos conceitos aprendidos pela rede.

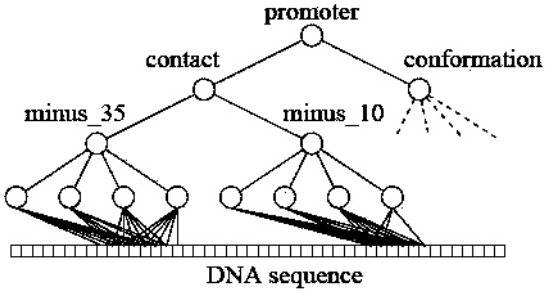


Figura 11: Exemplo de rede neural para detecção de promotores (TOWELL; SHAVLIK; NOORDEWIER, 1990)

2.3.4 Modelo Oculto de Markov

Sequências biológicas como, por exemplo, regiões promotoras, regiões de *splicing* e outras, podem ser encaradas resumidamente como sequências de nucleotídeos que apresentam bases com diferentes graus de conservação em cada posição. Para analisar as similaridades entre as duas ou mais sequências de forma global ou local, normalmente são utilizados métodos de alinhamento de sequências, ou seja, métodos que realizam a comparação do tipo de nucleotídeo que aparece em cada posição em um conjunto de sequências.

O modelo de Markov também executa essa tarefa com a vantagem de considerar a possibilidade de deleção de uma base da sequência e não apenas a hipótese de uma base estar ou não pareada com as demais que ocupam uma mesma posição (MOUNT, 2001).

O método do Modelo Oculto de Markov (HMM, do inglês *Hidden Markov Model*) é um processo estocástico definido por:

- Um conjunto de S estados.
- Um alfabeto A de m símbolos.
- Uma matriz de probabilidade de transição $T = (t_{ij})$.
- Uma matriz de probabilidade de emissão $E = (e_i(x))$.

O sistema evolui de estado a estado aleatoriamente, enquanto emite símbolos do alfabeto. Quando o sistema está em um estado i, ele tem a probabilidade t_{ij} de mudar do estado i para o estado j, e a probabilidade $e_i(x)$ de emitir o símbolo X.

Um modelo de Markov para sequências biológicas pode ser construído com três tipos de estados: principal, em que ocorre pareamento de bases; inserção, no caso de não pareamento; e de deleção. Além disso, há a possibilidade de mais dois estados silenciosos, ou seja, que não emitem símbolos ao serem atingidos, para representar o início e fim da sequência. Na Figura 12, está representada uma arquitetura genérica de HMM usado em Bioinformática. Os estados P, I e D, se referem a Pareamento, Inserção e Deleção, respectivamente.

A partir de algumas sequências de treinamento (DNA ou proteínas) é possível, através do HMM, construir um perfil que possa representar uma determinada classe, como por exemplo, promotor, não-promotor, fronteira intron-exon, etc. Esse perfil é construído baseado na composição e nos padrões identificados nas sequências de treinamento. Desta forma, um perfil HMM pode ser utilizado como um classificador atuando na identificação de padrões em sequências.

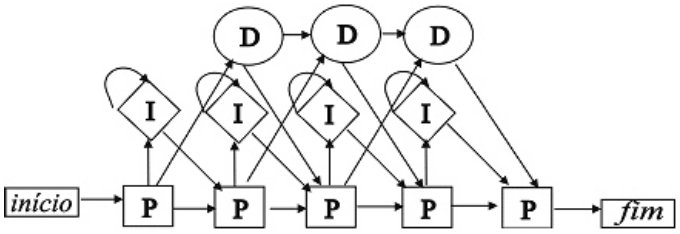


Figura 12: Exemplo de HMM para sequências biológicas

2.3.5 Árvores de Decisão

Uma árvore de decisão é basicamente uma lista de perguntas cujas respostas são do tipo SIM e NÃO, organizadas hierarquicamente, e que levam a uma decisão.

Na tarefa de reconhecimento de genes, as árvores de decisão são normalmente construídas para atuar como classificador, que identifica se determinada sequência corresponde ou não a um determinado sinal como promotor, sítio de *splicing*, etc.

Existem inúmeros algoritmos destinados a construção das árvores de decisão. Estes algoritmos, conhecidos como algoritmos de indução, são aplicados a um conjunto de dados, chamado de conjunto de treinamento, com a finalidade de encontrar regras que recursivamente bifurcam o conjunto de dados. A Figura 13 mostra um exemplo de uma árvore de decisão construída para reconhecer junções éxon-íntron (*donors*) (CRAVEN; SHAVLIK, 1994).

Uma vantagem das árvores de decisão sobre os outros métodos é que estas produzem modelos que podem ser facilmente interpretados por especialistas. Esta característica é importante porque permite que especialistas possam avaliar o conjunto de regras aprendidas por uma árvore de decisão e determinar se o modelo aprendido é plausível, isto é, se possui coerência do ponto de vista biológico.

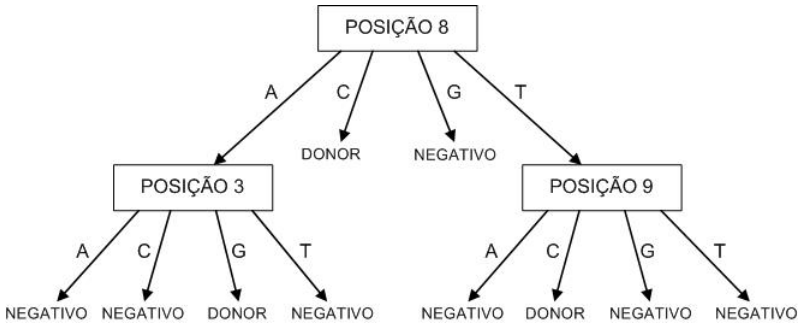


Figura 13: Exemplo de árvore de decisão para detectar donors (CRAVEN; SHAVLIK, 1994)

2.3.6 Técnicas de Processamento Digital de Sinais

Estas técnicas estão entre os métodos baseados na busca por conteúdo, pois se limitam em verificar propriedades estatísticas, ao contrário da busca por sinais específicos.

As regiões do DNA que codificam proteínas apresentam tipicamente uma organização periódica de três bases que não é encontrada em outras regiões, como as regiões intergênicas e íntrons. Essa característica, conhecida na literatura como periodicidade de três bases (TBP, do inglês *Three-Base Periodicity*), tem sido amplamente discutida nos últimos anos e baseado nela foram propostos vários modelos para reconhecer genes (TIWARI et al., 1997; YAN; LIN; ZHANG, 1998).

Neste tipo de análise, a sequência de teste original, na forma de nucleotídeos, é mapeada em um formato numérico para permitir a aplicação das técnicas de processamento digital de sinais. Este mapeamento pode ser do tipo fixo ou baseado em algum critério de otimização.

Análises baseadas em transformadas de Fourier, *wavelets* e derivações são as mais comuns (ANASTASSIOU, 2000; KAUER; BLOCKER, 2003). A Figura 14 mostra o espectro em frequência de uma sequência codificante, apresentado no trabalho de Tiwari et al. (TIWARI et al., 1997). Neste gráfico é possível verificar uma predominância na componente de frequência igual a um terço da frequência de amostragem, o que indica a periodicidade de três bases, característica das regiões codificantes.

A principal desvantagem destas técnicas, e que é comum a todos os métodos baseados na busca por conteúdo, é que a qualidade de seus resultados diminui à medida que o tamanho da janela de busca diminui.

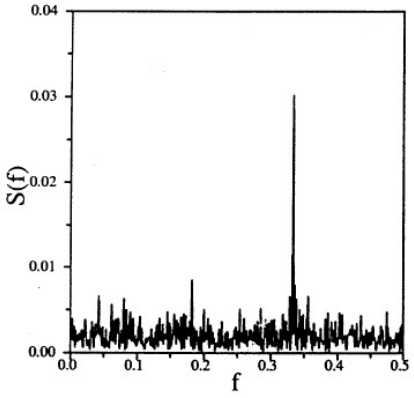


Figura 14: Espectro em frequência de sequência codificante (TIWARI et al., 1997)

2.4 FERRAMENTAS PARA DETECÇÃO DE GENES

Esta seção descreve as principais ferramentas para predição de genes usadas atualmente, além das técnicas utilizadas.

2.4.1 FGENES

O FGENES (SOLOVYEV; SALAMOV; LAWRENCE, 1994) é um *software* para predição de exons internos através da observação de características estruturais, como *splice sites* (*acceptors* e *donors*), promotores, etc.

Este programa utiliza uma técnica matemática, conhecida como análise linear discriminante, que permite obter uma combinação linear de duas ou mais variáveis independentes que melhor discrimina grupos definidos a priori.

Outra técnica utilizada pelo FGENES é a programação dinâmica. Esta é utilizada para encontrar a melhor combinação de exons, promotores, e regiões de poliadenilação detectados por algoritmos de reconhecimento de padrões.

O seu modelo é bem flexível e permite que sejam encontrados genes simples, múltiplos genes ou partes destes em uma sequência de teste.

O FGENES pode ser encontrado no endereço http://genomic.sanger.ac.uk/.

2.4.2 GeneID

O GeneID (GUIGO et al., 1992) utiliza um sistema fundamentado em regras para examinar supostos éxons e agrupá-los em uma estrutura que considera o gene mais provável para a sequência de teste.

Este programa utiliza matrizes de pesos para localizar sequências que possam representar os sinais de *splice sites* (*acceptors* e *donors*), *start codons* e *stop codons*. Uma vez feita esta busca, são construídos modelos de supostos éxons. Com base neste conjunto de supostos éxons e utilizando as suas regras, o GeneID faz um refinamento dos resultados produzindo a estrutura genética mais provável da sequência de entrada.

O conjunto de regras do GeneID foi obtido utilizando um conjunto de 169 genes de vertebrados e os resultados apresentados pelos autores mostraram que esta técnica é eficaz para a descoberta de genes.

O GeneID pode ser acessado em http://www1.imim.es/geneid.html.

2.4.3 GeneMark.hmm

O GeneMark.hmm (LUKASHIN; BORODOVSKY, 1998) é um *software* para predição de genes tanto em procariotos (versão GeneMark.hmm-P) como eucariotos (versão GeneMark.hmm-E).

Este programa utiliza um Modelo Oculto de Markov de duração explícita, com semelhanças com os modelos do Genie e do GenScan.

Além do HMM, outras duas técnicas são utilizadas para fornecer o resultado final: a programação dinâmica e um algoritmo de reconhecimento de sítios de ligação ribossômica. A sequência ótima é encontrada pelo algoritmo de Viterbi e um processamento posterior busca minimizar a sobreposição entre os genes preditos através da busca pelos sítios de ligações ribossômicas.

O GeneMark.hmm pode ser acessado em http://exon.biology.gatech.edu/hmmchoice.html.

2.4.4 Genie

O Genie (KULP et al., 1996) utiliza um método conhecido como Modelo Oculto de Markov Generalizado (do inglês *Generalized Hidden Markov Model*) para predição da estrutura genética. Este sistema é modular, isto é, cada estado do modelo é treinado isoladamente e permite que novos estados possam ser adicionados com facilidade.

Para detecção de *splice sites* (*acceptors* e *donors*), o Genie utiliza duas redes neurais *feed-foward* com treinamento *backpropagation* e uma camada escondida. Cada nucleotídeo foi codificado utilizando 4 entradas. Para *donors* foi utilizada uma janela de 15 nucleotídeos e para *acceptors* uma janela de 41 nucleotídeos.

Para modelar íntrons e éxons, o Genie utiliza a estratégia da busca por conteúdo. Uma janela de 300 nucleotídeos é usada para calcular a frequência dos nucleotídeos e compor a probabilidade de um trecho fazer parte de um íntron. Para exons, são usadas duas estatísticas. A primeira é composta da medida GC e de uma medida similar a dos introns (janela de 300 nucleotídeos). A segunda é gerada por uma cadeia de Markov de primeira ordem com a distribuição dos 61 possíveis códons. Essas duas medidas são combinadas em uma rede neural com duas camadas escondidas e 17 neurônios em cada uma. A saída desta rede neural corresponde à probabilidade de um determinado trecho pertencer a um exon.

O Genie pode ser obtido em http://cbl.cs.umass.edu/Research/GeneFinding.

2.4.5 GENSCAN

O GENSCAN é uma ferramenta para identificação de genes em cadeias de DNA baseada num modelo probabilístico para a estrutura do gene descrito por Chris Burge e Samuel Karlin (BURGE; KARLIN, 1997, 1998). O GENSCAN é capaz de detectar estruturas de genes completas identificando separadamente estruturas menores como íntrons, exons, etc.

O método adotado pelos autores do GENSCAN é o Modelo Oculto de Markov Generalizado. O mesmo utilizado no programa Genie (KULP et al., 1996).

Dentre as principais diferenças do GENSCAN em relação aos demais programas de detecção de genes está no fato de que este faz a busca nas duas fitas do DNA. Outra diferença está na característica da maioria dos programas de assumir que a sequência de entrada contém exatamente um gene, ou um sinal como, por exemplo, um promotor, etc. O GENSCAN é capaz de detectar na sequência de entrada genes parciais, genes completos e até mesmo vários genes separados por regiões intergênicas. Essa capacidade torna o GENSCAN um *software* extremamente útil para identificação genética.

O GENSCAN está disponível em http://genes.mit.edu/GENSCAN.html e a submissão de buscas é feita através do próprio navegador de internet.

2.4.6 HMMGene

O HMMgene (KROGH, 1997) utiliza uma variação do Modelo Oculto de Markov chamada de CHMM (do inglês *Class Hidden Markov Model*) para predizer a estrutura completa de um gene.

Diferente dos demais, o HMMgene não apenas fornece a melhor estrutura genética predita, mas também fornece predições alternativas a uma dada sequência de entrada, segundo a probabilidade fornecida pelo HMM. A vantagem desta abordagem é que retornando múltiplas predições para uma mesma região, o usuário pode aproveitar este resultado em outras análises.

O modelo do HMMgene possui três estados de quarta ordem para modelar regiões codificadoras, que são essencialmente uma cadeia de Markov não homogênea. Os outros estados do modelo são de primeira ordem, exceto para os estados que modelam íntrons internos e as regiões intergênicas, que são de terceira ordem. Para agregar as regras de *splicing*, são necessárias três instâncias do modelo de íntrons, sendo os estados destes três modelos ligados.

Para aumentar a velocidade do treinamento, é utilizada inicialmente a técnica da máxima verossimilhança seguida pela aplicação da técnica de máxima verossimilhança condicional. O

número máximo de iterações utilizado no treinamento foi 20. Este programa foi desenvolvido utilizando 353 genes humanos, todos contendo, no mínimo, um íntron.

O HMMgene pode ser encontrado em http://www.cbs.dtu.dk/services/HMMgene.

2.4.7 MORGAN

O MORGAN (SALZBERG et al., 1998) (*Multi-frame Optimal Rule-based Gene Analyzer*) é um sistema para predição de genes em sequências de DNA de vertebrados. Este programa combina as técnicas de árvore de decisão, algoritmos de detecção de sinais e programação dinâmica para identificar *start codons*, *acceptors*, *donors* e *stop codons*.

Para detectar os sinais, isto é, reconhecer *start* e *stop codons*, *acceptors* e *donors* individualmente, o MORGAN utiliza cadeias de Markov de ordem fixa. Para *start* e *stop codons* são utilizadas cadeias de Markov de primeira ordem, e para *acceptors* e *donors*, cadeias de segunda ordem.

Uma árvore de decisão é usada para estimar a probabilidade de uma determinada sequência ser um éxon inicial, um éxon interno, um éxon final ou um íntron.

Para encontrar o segmento ótimo que contenha a estrutura completa do gene predito numa sequência de DNA é utilizada programação dinâmica.

O MORGAN pode ser obtido em http://www.cbcb.umd.edu/ salzberg/morgan.html.

2.4.8 MZEF

O MZEF (*Michael Zhangs Exon Finder*) (ZHANG, 1997) é um programa para predição de exons internos que utiliza uma técnica matemática conhecida como análise discriminante quadrática.

Este programa executa a busca somente por éxons internos sem nenhuma outra informação sobre a estrutura genética. No desenvolvimento do MZEF foram utilizados, como conjunto de treinamento, 1879 exons verdadeiros e 184217 exons falsos, retirados do GenBank.

A técnica da análise discriminante quadrática é aplicada na separação dos dois grupos: exons verdadeiros e exons falsos. Na sua análise, o MZEF utiliza nove medidas:

- Comprimento do exon;
- Transições exon-intron;

- Pontuação do sítio de ramificação (*branch*);
- Pontuação do sítio de *splice* 3';
- Pontuação do exon;
- Pontuação do filamento;
- Pontuação da estrutura;
- Pontuação do sítio de *splice* 5';
- Transições intron-exon.

O MZEF pode ser encontrado no endereço http://rulai.cshl.org/tools/genefinder/.

2.4.9 PROCRUSTES

O PROCRUSTES (GELFAND; MIRONOV; PEVZNER, 1996) é um *software* para identificação de genes que utiliza a abordagem da busca por similaridades. Este *software* recebe uma sequência de teste e busca a se ajustar aos padrões definidos por uma proteína alvo. Ao contrário dos outros métodos que não utilizam a busca por similaridades (só exigem como entrada a sequência de teste), o PROCRUSTES exige que o usuário identifique supostos produtos do gene, isto é, proteínas candidatas, antes de fazer a predição.

Este método usa um algoritmo de alinhamento para explorar sequencialmente todas as possibilidades de blocos de éxons, buscando o melhor ajuste para a estrutura genética. Se uma proteína candidata for identificada na sequência de teste, o PROCRUSTES sinaliza a estrutura genética predita.

Através deste método é possível reconhecer com sucesso genes com exons curtos bem como genes complexos com mais de 20 exons.

O PROCRUSTES pode ser encontrado em http://www-hto.usc.edu/software/procrustes/.

2.5 MÉTRICAS DE ACURÁCIA

Algumas métricas são normalmente utilizadas para avaliar a qualidade de um sistema classificador, e portanto, são importantes na avaliação do desempenho das ferramentas de busca de genes.

2.5.1 Sensibilidade e especificidade

Sensibilidade e especificidade são duas medidas de acurácia preditiva que têm sido usadas com frequência em classificação, especialmente em bioinformática (BURSET; GUIGO, 1996).

Basicamente, a sensibilidade mede a proporção de casos positivos que são corretamente identificados como tal e a especificidade mede a proporção de casos negativos que também são corretamente identificados.

A Equação 3 mostra como a sensibilidade (Sn) e a especificidade (Sp) são calculadas, baseadas nos quatro possíveis resultados de um classificador:

- VP: verdadeiro positivo - número de casos positivos que foram corretamente classificados como positivos;
- FN: falso negativo - número de casos positivos que foram erroneamente classificados como negativos;
- FP: falso positivo - número de ocorrências negativas que foram erroneamente classificados como positivas;
- VN: verdadeiro negativo - número de ocorrências negativas que foram corretamente classificados como negativas.

$$Sn = \frac{VP}{(VP+FN)} \qquad Sp = \frac{VN}{(VN+FP)} \tag{3}$$

2.5.2 Coeficiente de Correlação de Matthews

Ambas, sensibilidade e especificidade, são definidas no intervalo $[0..1]$, com predição perfeita ocorrendo se, e somente se, ambas (Sn e Sp) forem iguais a 1. No entanto, é possível ter um classificador com alta sensibilidade e baixa especificidade, ou o oposto.

Isoladas, tanto Sn como Sp, não formam uma boa métrica de exatidão da predição. Portanto, é necessário conceber um único valor de exatidão global para resumir as duas medidas.

Uma possível métrica que atende a esse requisito é Coeficiente de Correlação de Matthews (MCC) (MATTHEWS, 1975) (Equação 5), considerado como uma medida balanceada e fre-

quentemente utilizada em bioinformática (BALDI et al., 2000; TAVARES; LOPES; LIMA, 2007, 2008).

$$MCC = \frac{(VP \cdot VN) - (FN \cdot FP)}{\sqrt{(VP+FN) \cdot (VN+FP) \cdot (VP+FP) \cdot (VN+FN)}} \quad (5)$$

2.5.3 Gráfico ROC

Outra maneira de avaliar um sistema classificador é através do gráfico ROC (FAWCETT, 2006). Um gráfico ROC (do inglês, *Receiver Operating Characteristics*) é uma técnica muito útil para comparação de classificadores e observação visual do desempenho.

Este tipo de gráfico é comumente usado não apenas na tomada de decisão, mas também em aprendizagem de máquina, mineração de dados e bioinformática (SING et al., 2005). Em um gráfico ROC, os eixos x e y são definidos, respectivamente, como $(1 - Especificidade)$ e $Sensibilidade$. Estes eixos podem ser interpretados como a relação custo-benefício de um classificador.

Na comparação de classificadores utilizando o gráfico ROC, o melhor método será aquele que mais se aproximar do canto superior esquerdo (coordenadas $(0,1)$), que representa 100% de sensibilidade e 100% de especificidade.

Um classificador completamente aleatório seria representado por um ponto ao longo de uma linha diagonal de coordenadas $(0,0)$ a $(1,1)$, do canto inferior esquerdo para o canto superior direito. A Figura 15 ilustra o posicionamento de um classificador perfeito e um classificador aleatório em um Gráfico ROC.

2.6 COMPUTAÇÃO RECONFIGURÁVEL

Esta seção aborda o conceito da computação reconfigurável, o componente FPGA, as características das linguagens de descrição de hardware citando as principais e cita alguns trabalhos recentes que envolvem a bioinformática e a computação reconfigurável.

2.6.1 Conceito

Segundo Compton e Hauck (COMPTON; HAUCK, 2002), existem, na computação convencional, basicamente dois métodos para execução de algoritmos.

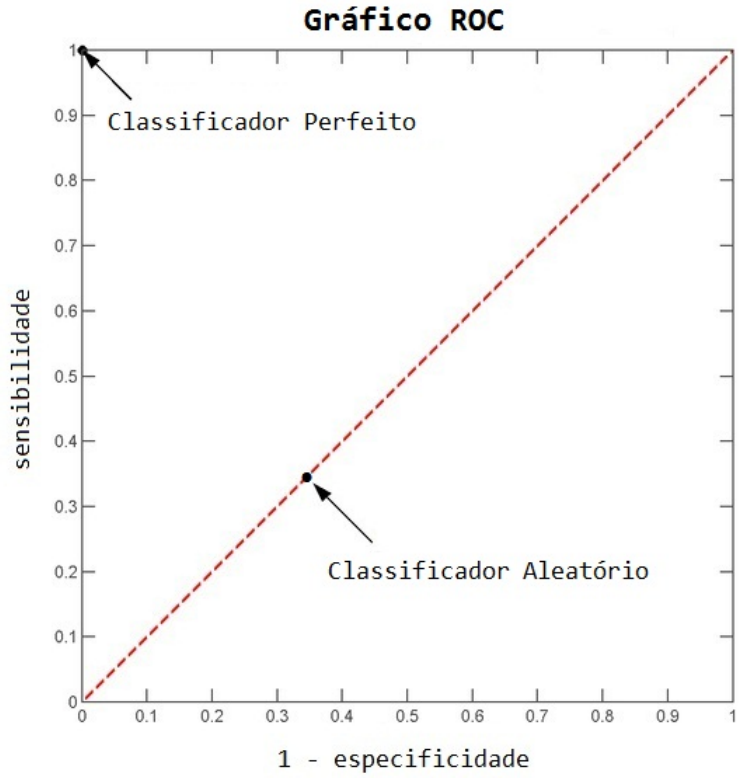


Figura 15: Exemplo de um Gráfico ROC

O primeiro é baseado em *hardware* projetado especificamente para executar determinada tarefa, que são chamados de ASICs (do inglês, *Application-Specific Integrated Circuit*). Estes possuem alto desempenho, entretanto, não são flexíveis. A alteração de um circuito integrado ASIC depois de fabricado não é possível.

O segundo método é baseado em *software* executado por um processador. Esta solução é extremamente flexível, pois a alteração das instruções do *software* são suficientes para modificar radicalmente sua funcionalidade sem alterar o *hardware* original. Apesar da flexibilidade, o método baseado em *software* apresenta uma grande desvantagem quando é analisada a questão do desempenho temporal. A cada instrução, é necessário que o processador faça um acesso à memória de programa para ler a instrução, decodificar e depois executá-la.

O objetivo da computação reconfigurável é preencher uma lacuna que há entre estas duas abordagens tradicionais. Segundo Martins et al (MARTINS et al., 2003), a computação reconfigurável pode ser considerada um tipo de implementação (modelo ou paradigma) de solução intermediária, entre as tradicionais soluções em *hardware* e em *software*.

A Figura 16 mostra o posicionamento da computação reconfigurável frente às duas abordagens tradicionais.

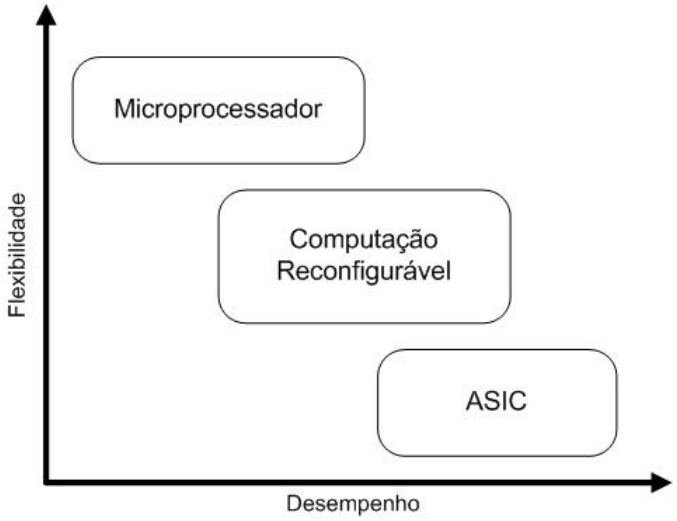


Figura 16: Posicionamento da Computação Reconfigurável

2.6.2 FPGA

Os FPGAs (do inglês *Field-Programmable Gate Arrays*) foram introduzidos no mercado em 1985 pela Xilinx. Estes dispositivos permitem a integração de lógica e memória em um único circuito, além de serem programáveis. Fato este que permite a reconfiguração do *hardware* e a rápida prototipação de complexos sistemas digitais.

A alta capacidade de armazenamento e reconfiguração dos FPGAs fizeram destes os principais responsáveis pela ampla expansão dos sistemas reconfiguráveis.

A estrutura básica de um FPGA pode variar de fabricante para fabricante, entretanto todos são compostos basicamente dos seguintes recursos:

- Funções lógicas programáveis (blocos lógicos);
- Rede de conexão para interligação dos blocos lógicos;
- *Flip-flops* ou registradores para o armazenamento de informações;
- Amplificadores de corrente de entrada e saída;
- Blocos de memória RAM interna (nos dispositivos mais modernos).

2.6.3 Linguagens de Descrição de *Hardware*

Atualmente as linguagens de descrição de *hardware* (do inglês, HDL - *Hardware Description Languages*) fazem parte dos métodos mais comuns para projetos que utilizam componentes de lógica programável.

Linguagens de programação de alto nível, como por exemplo a linguagem C, são inerentemente sequenciais. Estas definem o comportamento da aplicação seguindo uma série de instruções que são executadas sequencialmente. Nas linguagens de descrição de *hardware*, a sintaxe e a semântica descrevem simultaneidade e concorrência, características inerentes ao *hardware*.

Algumas vantagens da utilização de uma linguagem de descrição de *hardware* são:

- permitem maior poder de abstração ao projetista, facilitando o desenvolvimento de projetos complexos;
- a descrição é bem definida pois, como em toda linguagem de programação, as HDLs possuem regras sintáticas que não permitem dupla interpretação;
- a descrição do circuito, na maioria das vezes, serve como documentação e sugere os objetivos do projeto;
- o uso da síntese aumenta a produtividade;
- a padronização das HDLs resulta em portabilidade, tornando o código reutilizável em diferentes ambientes de desenvolvimento.

A utilização de uma linguagem de descrição de *hardware* também possui alguns problemas. Dentre estes pode-se citar:

- investimento inicial em treinamento dos projetistas;
- não é adequada para todo tipo de projeto de *hardware*;
- nem todas as construções possíveis da linguagem podem ser efetivamente sintetizadas;
- não atende a algumas restrições como, por exemplo, circuitos voltados ao baixo consumo;
- até o momento, não existem HDLs para a descrição de circuitos analógicos e mistos.

A primeira linguagem de descrição de *hardware* foi o ISP (do inglês, *Instruction Set Processor*). Esta linguagem foi desenvolvida em 1977 na Universidade Carnegie Mellon.

Em 1983, surgiu outra linguagem, o ABEL (do inglês, *Advanced Boolean Equation Language*). Sua principal utilização se deu na descrição de máquinas de estados.

O Verilog HDL, ou simplesmente Verilog, foi a primeira linguagem de descrição de *hardware* moderna. Esta linguagem foi desenvolvida em 1985 na Gateway Design Automation, a antiga Automated Integrated Design Systems.

Em 1987, surgiu o VHDL (do inglês, *Very High Speed Integrated Circuit Hardware Description Language*). O projeto desta linguagem teve, no seu início, incentivo do Departamento de Defesa Norte-Americano e, mais tarde, se tornou uma das linguagens mais utilizadas para descrição de *hardware* no mundo. Neste ano de 1987, após revisões propostas por acadêmicos e representantes da indústria e do governo dos Estados Unidos, surgiu o padrão IEEE 1076-1987. Em 1993, uma nova versão da linguagem, denominada IEEE 1076-1993, foi aprovada pelo IEEE (*Institute of Electrical and Electronic Engineer*).

No início da década de 90, a VHDL era usada basicamente para projetos em ASIC. Na segunda metade desta década, o uso da VHDL moveu-se para a área dos dispositivos lógicos programáveis (CPLDs e FPGAs).

A VHDL é uma linguagem concorrente. Os comandos envolvidos na ocorrência de um evento são executados simultaneamente, da mesma forma que os elementos de um sistema digital executam suas tarefas de forma concorrente.

Além disso, a VHDL suporta projetos com múltiplos níveis de hierarquia, onde a descrição pode ir da simples interligação de outras descrições menores, a um código que representa o comportamento esperado do circuito inteiro. Essas abordagens de descrição podem ser de três tipos:

- Comportamental: o circuito é definido na forma de um algoritmo, utilizando construções similares às das linguagens de programação;
- Fluxo de dados: tem-se a visão dos dados como um fluxo através do circuito, da entrada até a saída;
- Estrutural: a visão mais próxima do *hardware*. Um modelo onde os componentes do circuito são instanciados e as interligações entre eles descritas.

O estilo de descrição comportamental, embora mais abstrato, deve levar em conta os aspectos característicos de um projeto de *hardware* para que o circuito possa ser sintetizado corretamente.

A estrutura de um programa escrito em VHDL baseia-se em níveis hierárquicos que podem ser resumidos em:

- Pacotes: permitem agregar componentes e entidades previamente definidos em um determinado projeto. Pode ser encarado como uma biblioteca de componentes dentro de um projeto;

- Entidades: descrevem os comportamentos dos componentes como caixas-pretas, isto é, apenas suas portas de entrada e saída são conhecidas;

- Arquitetura: é um conjunto de primitivas que efetivamente farão a descrição do *hardware*. É na arquitetura onde são definidas as abordagens comportamental, de fluxo de dados ou estrutural;

- Processos: um processo permite definir uma área contendo comandos sequenciais. Trata-se de uma abstração de *hardware* que está sempre atuando. Um processo possui uma lista de sinais dos quais depende (chamada de lista de sensitividade). Estes podem ser síncronos ou assíncronos e diversos deles podem ser definidos dentro de uma arquitetura. Os processos descritos em uma arquitetura são sempre concorrentes, mas o fluxo dentro de um processo é sequencial.

Outra característica importante da VHDL é que ela é uma linguagem fortemente tipada, isto quer dizer que as conversões de tipo de dados devem ser realizadas expressamente.

2.6.4 Computação Reconfigurável e Bioinformática

Algumas propriedades dos sistemas de computação reconfigurável têm sido exploradas nos últimos anos na implementação de soluções para diversos problemas de bioinformática. Dentre todos os problemas, o alinhamento de sequências é, provavelmente, o que mais foi abordado com recursos da computação reconfigurável. Como exemplo, pode-se citar alguns trabalhos:

(OLIVER et al., 2005) apresentam uma abordagem para o alinhamento múltiplo de sequências usando um acelerador constituído de uma matriz sistólica linear unidimensional em FPGA. O sistema desenvolvido funciona como um acelerador para o conhecido *software* de alinhamento de sequências chamado ClustalW. O sistema foi escrito em Verilog e implementado em uma placa da Virtex-II com interface PCI. O ClustalW executado com este acelerador é até 13 vezes mais rápido que o ClustalW sem acelerador.

(YAMAGUCHI; MARUYAMA; KONAGAYA, 2002) propõem uma versão do clássico algoritmo de alinhamento local Smith-Waterman, em *hardware*, aplicado à análise de sequências de DNA. O sistema foi implementado em uma placa com FPGA Virtex-II com interface PCI. O *speed up* alcançado comparado a um Pentium III 1GHz é de 327 vezes para uma sequência de

teste de 2048 elementos. Uma versão com um FPGA com menos recursos também foi implementado. Este alcança *speed up* de 30 vezes para uma sequência de teste de 1024 elementos.

(LIMA et al., 2007; MORITZ; LOPES; LIMA, 2006) apresentam um sistema com vários níveis de pipeline, chamado de *Hardalign*, para alinhamento pareado, cujo objetivo é acelerar uma das tarefas realizadas pelo *software* Clustal, em *hardware*. Experimentos foram realizados com proteínas de até 2000 aminoácidos para medir o desempenho do sistema. Este apresenta um *speed up* de 5 vezes em relação a um algoritmo similar implementado em *software* e executado em PC.

(MARONGIU; PALAZZARI; ROSATO, 2003a, 2003b) apresentam um sistema chamado PROSIDIS, destinado a buscar ocorrência de sequências similares na análise de proteínas. Este sistema foi implementado na forma de um coprocessador de propósito específico com interface PCI. Testes demonstraram *speed ups* que variam de 5 a 50 vezes em relação aos processadores de propósito geral.

(ARMSTRONG JUNIOR; LOPES; LIMA, 2006, 2007) apresentam os resultados de um sistema, baseado em lógica reconfigurável, aplicado a um modelo de dobramento conhecido como 2D-HP. Neste trabalho foi proposta uma abordagem para alcançar uma redução do espaço de busca de dobramentos possíveis. Várias simulações foram feitas para avaliar o desempenho do sistema, bem como a demanda por recursos do FPGA. Em alguns casos, a redução do espaço de busca alcançado foi de 0,001% do espaço de busca original.

Para detecção de genes, e seus múltiplos subproblemas, os métodos em *hardware* mais citados são derivados das técnicas de alinhamento de sequências. Algumas poucas abordagens diferentes destas primeiras também foram propostas:

(LOPES; LIMA; MURATA, 2007) apresentam a implementação de uma árvore de decisão, na forma de um circuito combinacional em *hardware*, aplicada ao problema da detecção de *splice junctions*. Neste trabalho as árvores de decisão foram construídas através do algoritmo de indução C4.5 aplicado a um conjunto de dados conhecido. Essas árvores foram convertidas em equações lógicas e submetidas à minimização booleana. O circuito resultante foi implementado em FPGA e avaliado utilizando o processo de validação cruzada de 5 partes. A precisão média obtida foi de 90,41% e o tempo de processamento de uma janela de 60 nucleotídeos foi 18 ns.

(CHRYSOS et al., 2009) utiliza uma técnica chamada de *Interpolated Markov Model* (IMM) em um coprocessador, baseado em FPGA, destinado a detecção de genes. Este modelo (IMM) está presente no *software* Glimmer, que é um sistema de detecção de genes em DNA microbiano, como genomas de bactérias, arqueobactérias e vírus. Neste trabalho também é

apresentado um método para construção de grandes *look up tables*.

(STEPANOVA; LIN; LIN, 2007) apresenta um classificador, baseado em uma rede neural do tipo Hopfield, aplicado ao reconhecimento de um importante sinal chamado de *Hormone Response Element* (HRE). A rede neural, constituída de 60 neurônios, foi treinada com base em um conjunto de 600 instâncias do sinal HRE. O circuito foi descrito em Verilog e testado em uma placa da Virtex-4. Uma comparação com a mesma rede neural implementada em *software* foi feita. A versão em *hardware* chega a ser mais de 20 vezes mais rápida em alguns casos e o desempenho preditivo muito próximo do *software*.

Em (STEPANOVA; LIN; LIN, 2008), os autores apresentam uma abordagem sistemática para o reconhecimento de HREs nos promotores de genes vertebrados. O modelo utilizado é conhecido como Modelo de Markov em Cascata (em inglês *Cascade Markov Model*). A implementação do modelo em FPGA foi descrita em Verilog e testada em uma placa da Virtex-4. Devido à complexidade do modelo, esta implementação não é adequada para analisar grandes sequências genômicas. Estas podem elevar o tempo de processamento a níveis proibitivos.

Para o problema da reconstrução filogenética, (BAKOS, 2007) utiliza idéias do GRAPPA (*Genome Rearrangement Analysis through Parsimony and other Phylogenetic Algorithms*), que é um importante projeto que se utiliza de sistemas paralelos de alto desempenho para a análise filogenética.

(DANDASS et al., 2008) apresenta um modelo baseado em máquinas de estado finitas paralelas baseadas no algoritmo Aho-Corasick, para detecção de sequências. Resultados experimentais demonstram a eficiência de armazenamento de mais de 80% para vários conjuntos de dados. Além disso, o sistema em FPGA executado a 100 MHz é quase 20 vezes mais rápido que uma implementação tradicional do algoritmo Aho-Corasick executado em uma estação a 2,67 GHz.

Para o problema de classificação de famílias de proteínas, (BECKSTETTE et al., 2009) mostram a utilização de perfis HMMs em *hardware*. Os experimentos demonstraram que a versão em *hardware* possui um desempenho de classificação um pouco menor que o *hmmsearch*, que é um conhecido programa utilizado para classificação de proteínas. Com relação ao desempenho temporal, a versão em *hardware* alcança *speed ups* maiores que 64 vezes em relação ao *hmmsearch*.

3 METODOLOGIA

O objetivo deste capítulo é apresentar a metodologia utilizada no desenvolvimento do presente trabalho.

Este capítulo descreve a escolha da abordagem, citando os experimentos preliminares com modelos conhecidos, baseados em *software*, e o sistema desenvolvido, constituído de uma arquitetura híbrida, baseada em *software* embarcado e *hardware* reconfigurável.

3.1 ESCOLHA DA ABORDAGEM

Segundo (BAXEVANIS; OUELLETTE, 1998), existem basicamente 3 abordagens para a detecção computacional de genes: a busca por similaridades, a busca por conteúdo e a busca por sinais. Sabe-se que o método mais promissor é aquele que combina duas ou mais estratégias. Entretanto, dentre as três, a abordagem da busca por sinais é a mais eficiente e que leva muitas vezes a resultados mais precisos (CHRYSOS et al., 2009). Outro fato importante é que esta abordagem possui um paralelismo potencial que é desperdiçado nas atuais ferramentas baseadas em *software* sequencial. Como a busca por sinais pode ser encarada como um conjunto de subproblemas de classificação que podem ser tratados isoladamente, é possível executar todas estas tarefas ao mesmo tempo, desde que a arquitetura adotada permita este paralelismo. Devido a estes fatos, a busca por sinais foi a abordagem escolhida para o desenvolvimento do presente trabalho.

Foram conduzidos experimentos de forma a determinar os modelos a serem adotados na implementação dos sensores deste sistema. Duas características essenciais nortearam o processo de escolha do método:

1. Bom desempenho preditivo;
2. Adequação do modelo à implementação em *hardware*.

O objetivo principal deste trabalho é buscar uma alternativa para reduzir o tempo de análise das sequências de DNA. Entretanto, esta redução do tempo não pode comprometer a qualidade preditiva do sistema total. Por este motivo, a qualidade preditiva foi considerada característica essencial para o modelo a ser adotado.

Outra característica essencial é a adequação do modelo à implementação em *hardware*. Existem modelos que possuem boa qualidade preditiva, entretanto, não trazem benefícios se implementados em *hardware*, ou até mesmo, apresentam grande complexidade quando descritos em alguma linguagem de descrição de *hardware*. Por isso, é essencial que o modelo a ser adotado seja adequado à implementação em *hardware*.

3.1.1 Matrizes de Pesos

Nos últimos anos, vários métodos têm sido aplicados na construção de sensores para reconhecimento de sinais em DNA, tais como redes neurais, árvores de decisão, HMMs e muitas outras abordagens. Uma técnica amplamente utilizada, chamada de Modelo da Matriz de Pesos, ou simplesmente WMM (do inglês, *Weight Matrix Model*), foi utilizada neste trabalho.

Este modelo, membro da família dos métodos probabilísticos, fornece resultados muito bons e possui um paralelismo potencial que a torna extremamente adequada para ser implementada em *hardware* reconfigurável.

Muitos trabalhos demonstram a aplicação das matrizes de pesos em diversos problemas na bioinformática. Por exemplo, (STADEN, 1984) utiliza matrizes de pesos para representar diversos sinais de sequência de nucleotídeos, (SENAPATHY; SHAPIRO; HARRIS., 1990) utilizam na predição de *splice sites*, (BUCHER, 1990; FICKETT, 1996; WINGENDER et al., 2000) utilizam na representação de elementos de promotores, (GERSHENZON; STORMO; IOSHIKHES, 2005) empregam na detecção de *binding sites*.

Nesta etapa foi conduzido um experimento destinado a avaliar os WMMs na construção de sensores de sinais. Desta vez, no entanto, os modelos não foram treinados com base em algum conjunto de dados. Neste, as matrizes foram obtidas prontas do N-Scan (que é um *software* de predição de genes bem conhecido) e testadas em duas bases de dados distintas, um conjunto de dados do UCI (BLAKE; MERZ, 1998) e HS3D (POLLASTRO; RAMPONE, 2002).

O sub-problema escolhido para este experimento foi a detecção de *splice junctions*, sinais presentes apenas em genes eucariotos. O reconhecimento de *splice junctions* é uma parte importante do processo de predição de genes eucariotos e, trata-se basicamente de um problema da detecção de *donors* e *acceptors*. Nos últimos anos, vários modelos computacionais têm sido

propostos para detecção destes sinais. Por exemplo, (TOWELL; SHAVLIK; NOORDEWIER, 1990) e (RAMPONE, 1998) utilizam redes neurais artificiais, (GELFAND; ROYTBERG, 1993) utiliza uma abordagem baseada em programação dinâmica, (CAI et al., 2000) cita redes Bayesianas, e (LOPES; LIMA; MURATA, 2007) árvores de decisão.

Os resultados sugerem a aplicabilidade do modelo, e atestam o fato deste modelo ser amplamente utilizado atualmente nas ferramentas de predição de genes.

Adequação da matriz de pesos à descrição em *hardware*

Outro fato que motivou a adoção da matriz de pesos neste trabalho é o potencial paralelismo que existe no seu algoritmo original. Esta característica a torna extremamente apropriada à implementação baseada em *hardware* reconfigurável.

A medida de similaridade de uma sequência de teste com o sinal procurado usando WMM se resume a soma algébrica do peso de cada nucleotídeo em cada posição da matriz, conforme a Equação 6.

$$Score(X) = \sum_{i=1}^{L} s_i(x_i) \tag{6}$$

Um exemplo de código em linguagem C, que implementa o cálculo da pontuação dada por um WMM, pode ser visualizado na Figura 17. Nesta figura, a matriz de pesos propriamente dita está implementada na forma de uma matriz de inteiros de 4 linhas e 12 colunas denominada "start". Esta matriz foi obtida do *software* NScan (GROSS; BRENT, 2005) e é utilizada na detecção de *start codons*. É possível notar, na função "ScoreStart", que não há dependência entre as instruções de leitura dos pesos da matriz, permitindo que essas possam ser realizadas em paralelo.

Outro fato a ser observado é que a acumulação da pontuação (somatório citado na Equação 6) pode ser realizada em cascata, desde que se utilizem vários somadores como mostra a Figura 18. Esta estratégia pode ser adotada em arquiteturas massivamente paralelas, como FPGAs e GPUs e contribui para a redução do tempo total de processamento da matriz de pesos.

3.2 SISTEMA PROPOSTO

Um dos passos mais importantes no processo de busca por sinais é a varredura da sequência de teste pelos sensores de sinais. Esta varredura gera uma lista de possíveis ocorrências que

```
const int start[4][12] = {
{    -5,   -6,   -2,    7,   -2,   -8,   18, -100, -100,   -3,   -1,   -9 },
{     3,    7,   10,   -9,   13,    9, -100, -100, -100,   -4,   10,    5 },
{     7,    4,    0,    5,   -4,    2, -100, -100,   23,    9,   -5,    6 },
{    -7,   -7,  -12,  -22,  -10,  -14, -100,   18, -100,  -11,   -8,   -5 } };

int ScoreStart (char * seq)
{
    int score = 0;

    for(int i=0;i<12;i++)
    {
        switch(seq[i])
        {
            case 'A': score += start[0][i]; break;
            case 'C': score += start[1][i]; break;
            case 'G': score += start[2][i]; break;
            case 'T': score += start[3][i]; break;
        }
    }

    return score;
}
```

Figura 17: Cálculo da pontuação de WMM em linguagem C

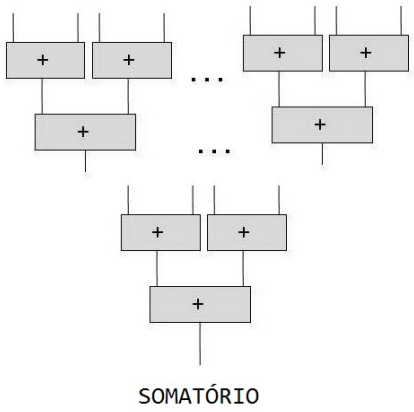


Figura 18: Exemplo de somadores em cascata

são posteriormente analisados por outros métodos para, em seguida, estimar a estrutura mais provável do gene.

O objetivo deste trabalho é atuar nesta primeira etapa: a varredura da sequência de teste. Este processo é realizado em toda a sua extensão por diversos sensores.

Em uma ferramenta tradicional baseada em *software* sequencial, a varredura só pode ser feita sequencialmente, isto é, um sensor e um trecho (janela) da sequência do DNA por vez. Em uma arquitetura paralela, esta limitação não existe. Vários sensores podem ser submetidos a uma única janela da sequência de teste ao mesmo tempo.

Além disso, a varredura da fita do DNA deve ser realizada em ambos os sentidos (direto e reverso). Em uma arquitetura paralela, a simples adição de sensores em posição inversa pode eliminar a busca em um dos sentidos, reduzindo o tempo de varredura pela metade.

3.2.1 Arquitetura do sistema

A Figura 19 mostra um diagrama em blocos simplificado da arquitetura utilizada.

Dentre as principais partes componentes, internas à FPGA, pode-se citar:

- Processador embarcado Altera Nios II (ALTERA, 2009b);
- Memórias ROM (4kbytes) e RAM (4kbytes) On-chip;
- UART (*Universal Asynchronous Receiver Transmitter*);
- Bloco de Varredura;
- Barramento Avalon.

A função do processador é executar as funções de alto-nível do sistema. Trocar dados (sequências de teste, limiares e resultados) com o PC, armazenar as sequências de teste na SRAM e comandar o bloco de varredura são as principais. No processador é executado um programa escrito em linguagem C, e compilado no ambiente Nios II IDE, versão 6.1.

O código binário do *software* embarcado é armazenado e executado a partir da ROM interna. Para a execução do *software* embarcado, também se faz necessária a presença da RAM interna. Esta é utilizada para armazenar variáveis do programa, regiões de memória alocadas dinamicamente, etc.

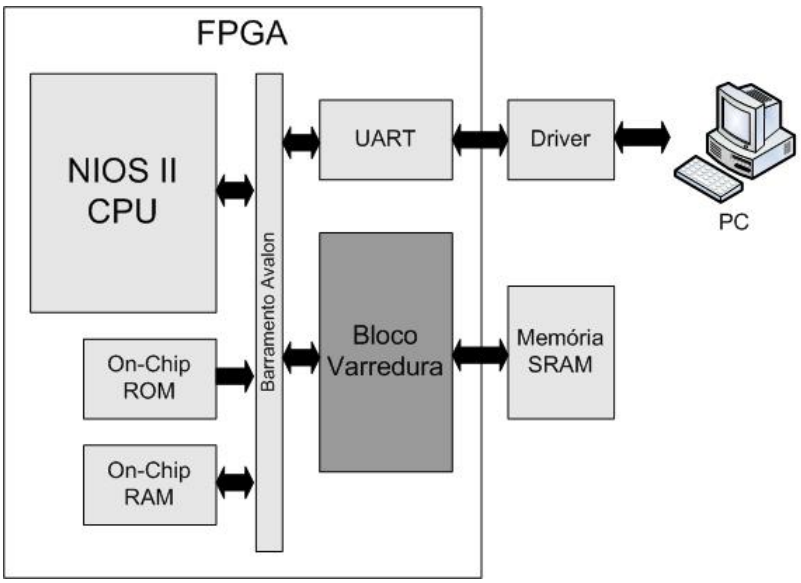


Figura 19: Diagrama em blocos da arquitetura utilizada

A UART se faz necessária para realizar a troca de dados com o PC de forma segura e facilitada. Da mesma forma que a CPU e as memórias internas, a UART também é um componente pronto disponibilizado pelo próprio SOPC Builder.

O Bloco de Varredura, mostrado na Figura 19, é um bloco de processamento dedicado, que funciona como um periférico do processador. A função deste bloco é realizar a varredura das sequências de teste de forma paralelizada.

Todos os periféricos internos à FPGA estão conectados ao processador através do barramento Avalon. Este barramento dispõe de um padrão de acesso (escrita e leitura), desenvolvido pela Altera, que permite que novos periférios sejam implementados e facilmente adicionados ao sistema.

A Figura 20 mostra a organização do processador e periféricos na compilação realizada através da ferramenta SOPC Builder.

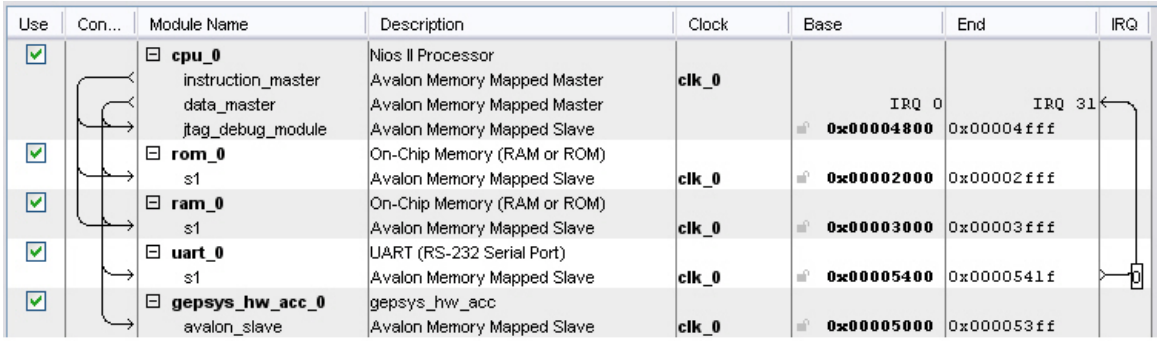


Figura 20: Organização do processador e periféricos

As principais partes, externas à FPGA, são:

- PC;
- Driver de comunicação serial;
- Memória SRAM (512kbyte).

O PC executa o programa de interface com o usuário. Neste programa é possível enviar as sequências de teste, enviar e receber os limiares dos sensores e obter os resultados da pesquisa.

O driver de comunicação serial é usado para adaptar os níveis de tensão utilizados no PC e na FPGA.

Na memória SRAM, externa à FPGA, é onde são armazenadas as sequências de teste. Ela está conectada diretamente ao bloco de varredura, e por meio deste, é que o processador envia as sequências para armazenamento.

3.2.2 Processador Nios II

O Nios II (ALTERA, 2009b) é um processador RISC, 32 bits, *softcore*, configurável e de propósito geral. O termo configurável significa que o processador possui características que podem ser adicionadas ou removidas, se adequando aos requisitos de desempenho e custo. O termo *softcore* significa que o núcleo da CPU é codificado em uma linguagem de descrição de *hardware*, como por exemplo, VHDL ou Verilog, e pode ser implementado em qualquer FPGA da Altera.

O Nios II possui, resumidamente:

- 94 instruções de propósito geral;
- 32 registradores de propósito geral;
- 32 fontes de interrupção externa;
- barramento de instruções separado do de dados, sendo uma arquitetura do tipo Harvard;
- instruções simples de multiplicação e divisão em 32 bits;
- instruções dedicadas para calcular multiplicações em 64 bits e 128 bits;
- acesso a uma variedade de periféricos *on-chip*, e interfaces para memória e periféricos *off-chip*;

- ambiente de desenvolvimento de *software* baseado na ferramenta GNU C/C++ e Eclipse IDE (*Integrated Development Environment*);
- desempenho de até 250 DMIPS (*Dhrystone Millions of Instruction per Second*).

Com o objetivo de atender as mais diversas demandas por processamento e disponibilidade de elementos lógicos, o Nios II possui três configurações básicas: Nios II/e (economy), Nios II/s (standard) e Nios II/f (fast).

Nios II/e (*economy*)

É chamada de versão econômica porque utiliza a menor quantidade de elementos lógicos dentre as três configurações. Suas características são:

- Módulo de depuração JTAG;
- Sistema completo em menos de 700 elementos lógicos;
- Até 256 instruções customizáveis.

Nios II/s (*standard*)

A versão padrão é a que possui o melhor equilíbrio entre desempenho e custo. Possui:

- Módulo de depuração JTAG;
- *Cache* de instruções;
- *Pipeline* de 6 níveis;
- *Static branch prediction*;
- Instruções de multiplicação, divisão e *shift* por *hardware*;
- Até 256 instruções customizáveis.

Nios II/f (*fast*)

Esta é a versão que possui o maior desempenho dentre as três configurações. Possui:

- Módulo de depuração JTAG;
- *Cache* de instruções e dados separados;
- MMU ou MPU opcionais;
- *Pipeline* de 6 níveis;
- *Dynamic branch prediction*;
- Instruções de multiplicação por *hardware* executadas em um ciclo de *clock*;
- Até 256 instruções customizáveis e aceleradores por *hardware* ilimitados.

3.2.3 Bloco de Varredura

A Figura 21 mostra um diagrama simplificado do bloco de varredura. Alguns sinais foram omitidos para facilitar a compreensão do diagrama. Este bloco possui uma interface baseada no padrão Avalon, da Altera, de forma a permitir a comunicação com o processador Nios II (ALTERA, 2009b).

O decodificador de comandos, cuja interface está representada na Figura 22, serve para controlar as principais funções do bloco, como escrever e ler os limiares, comandar a varredura, redefinir os contadores, resultados da leitura, etc.

Todos os sinais de entrada deste bloco (clk, cs_n, read_n, reset_n, write_n e address[7..0]) são provenientes diretamente do barramento do processador, e portanto, atendem rigorosamente à especificação da interface Avalon (ALTERA, 2009a). A Tabela 4 apresenta uma descrição sucinta da função de cada um dos sinais de entrada.

São utilizados, neste bloco, 8 bits do barramento de endereços para selecionar o comando adequado. O modelo adotado na construção do bloco foi o *Memory-Mapped Interface* com transferência de dados com número de *wait-states* fixo. Isto significa que o acesso a este bloco é feito com instruções semelhantes ao acesso à memórias com barramento paralelo.

A Figura 23 ilustra o comportamento deste sinais durante um acesso de leitura seguido por uma escrita. Nesta figura o comando de leitura foi implementado com 1 *wait-state* e a escrita com 2 *wait-states*. Os sinais readdata e writedata, que fazem parte do barramento Avalon, não

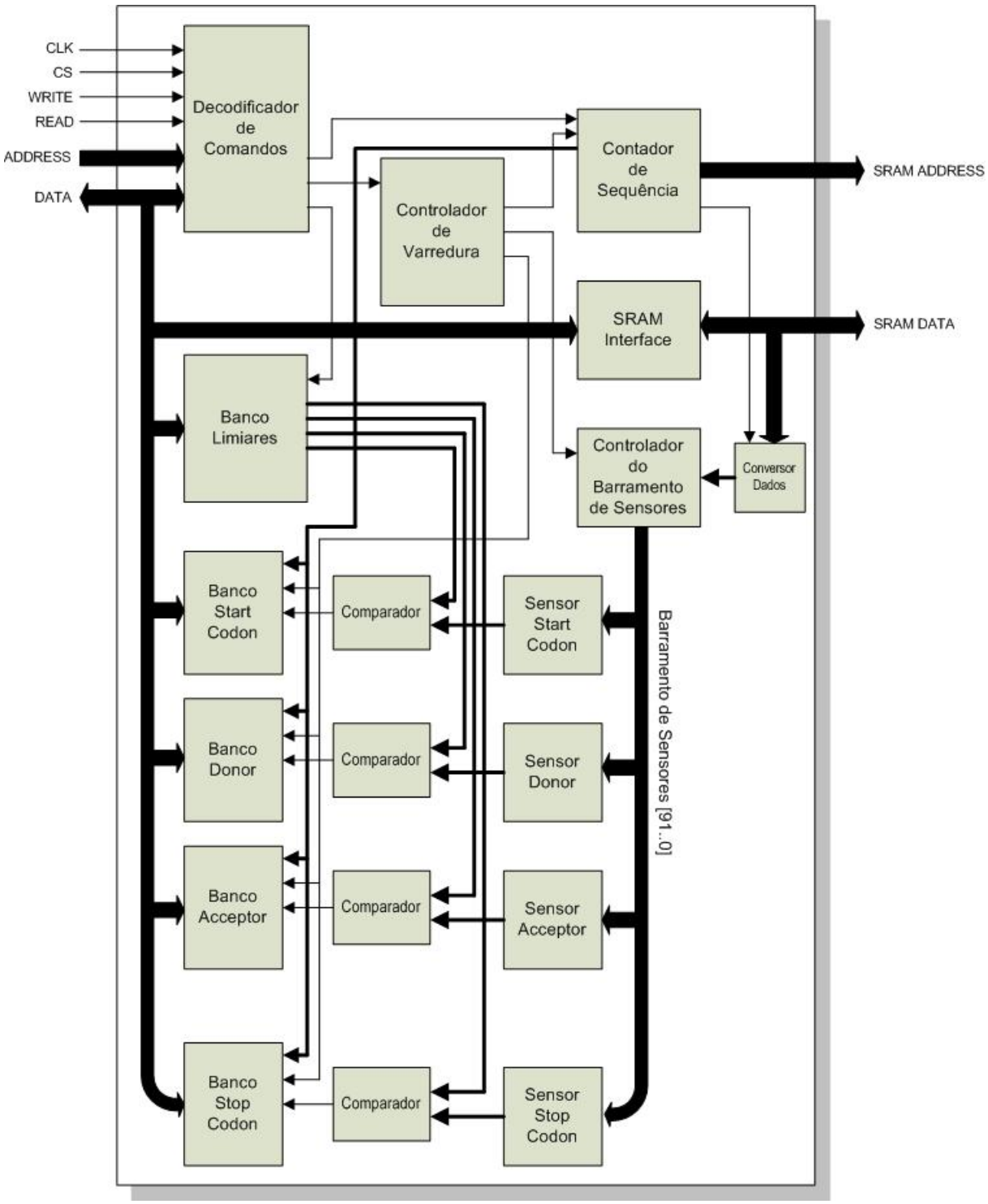


Figura 21: Diagrama do Bloco de Varredura

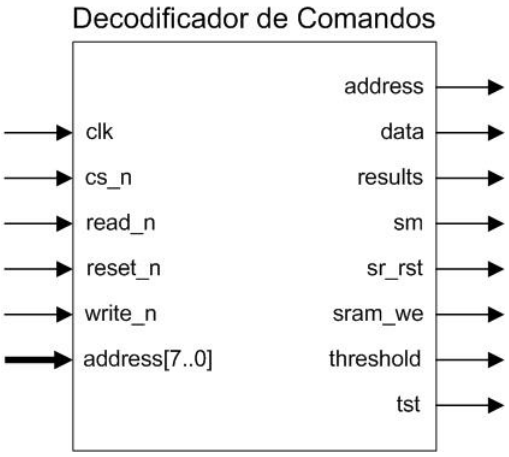


Figura 22: Diagrama do Decodificador de Comandos

Tabela 4: Sinais de entrada do decodificador de endereços

Sinal	Função
clk	Sinal de *clock*. Fornece sincronização para o bloco de varredura.
cs_n	Sinal de *chip select*. Sinaliza que o processador está acessando o bloco.
read_n	Ativo em nível baixo. Indica uma transferência de dados do tipo leitura.
reset_n	Entrada de *reset*. Ativo em nível baixo e sincronizado com o sinal de *clock*.
write_n	Ativo em nível baixo. Indica uma transferência de dados do tipo escrita.
address[7..0]	Especifica o endereço onde se deseja efetuar a leitura ou escrita.

foram utilizados na implementação do decodificador de endereços, visto que, a decodificação é feita exclusivamente pelo endereço e não pelo dado transferido.

A Tabela 5 mostra os endereços de acesso e as funções correspondentes implementados no decodificador de comandos.

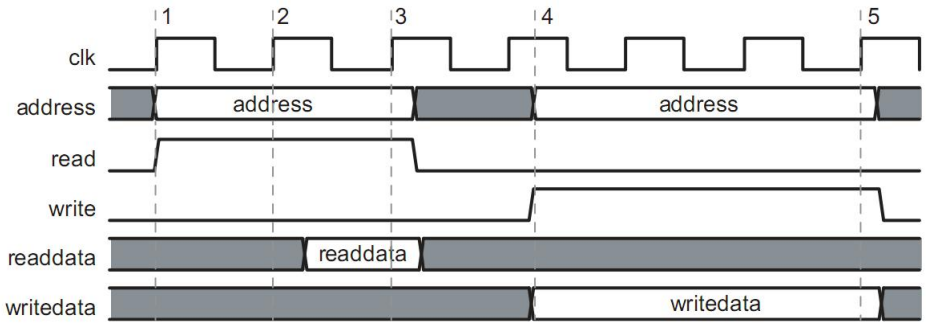


Figura 23: Ciclo de leitura e escrita no barramento Avalon

Tabela 5: Endereços e funções correspondentes

Comando	Endereço	Operação	Função
1	0x5004	Escrita	Armazena valor na memória SRAM externa
2	0x5020	Leitura	Carrega valor do banco de registradores do sinal Promotor
3	0x5024	Leitura	Carrega valor do banco de registradores do sinal *Start Codon*
4	0x5028	Leitura	Carrega valor do banco de registradores do sinal *Donor*
5	0x502C	Leitura	Carrega valor do banco de registradores do sinal *Acceptor*
6	0x5030	Leitura	Carrega valor do banco de registradores do sinal *Stop Codon*
7	0x5034	Leitura	Carrega valor do banco de registradores do sinal *PolyA*
8	0x5080	Escrita	Incrementa o contador de endereços
9	0x5100	Escrita/Leitura	Armazena/carrega o limiar para sinal Promotor
10	0x5104	Escrita/Leitura	Armazena/carrega o limiar para sinal *Start Codon*
11	0x5108	Escrita/Leitura	Armazena/carrega o limiar para sinal *Donor*
12	0x510C	Escrita/Leitura	Armazena/carrega o limiar para sinal *Acceptor*
13	0x5110	Escrita/Leitura	Armazena/carrega o limiar para sinal *Stop Codon*
14	0x5114	Escrita/Leitura	Armazena/carrega o limiar para sinal *PolyA*
15	0x5208	Escrita	Inicia o processo de varredura
16	0x5280	Escrita	Reinicia o contador de endereços
17	0x5300	Escrita	Incrementa o contador de endereços dos bancos de registradores
18	0x5380	Escrita	Limpa o banco de registradores dos resultados

O alfabeto de símbolos da sequência de DNA possui apenas 4 elementos (‘A’, ‘C’, ‘G’ e ‘T’), portanto são necessários apenas 2 bits para codificar cada um. A Tabela 6 mostra como é feita esta codificação.

Tabela 6: Codificação dos nucleotídeos

Nucleotídeo	Codificação
A	00
C	01
G	10
T	11

Como a SRAM utilizada possui barramento de dados de 16 bits, faz-se necessária a adaptação deste barramento para conectá-lo ao Controlador do Barramento de Sensores. Isto foi feito através do bloco Conversor de Dados.

Este bloco funciona como um multiplexador de 8 nucleotídeos para 1 nucleotídeo com seleção feita por 3 bits, como ilustra a Figura 24. Estes 3 bits são provenientes dos 3 bits menos significativos do Contador de Sequência. Para se obter este comportamento foram implementados dois multiplexadores de 8x1 bits em paralelo, conforme ilustra a Figura 25.

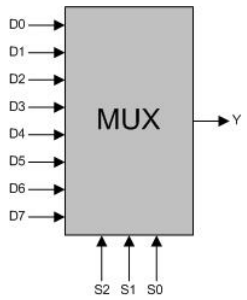


Figura 24: Diagrama do Conversor de Dados

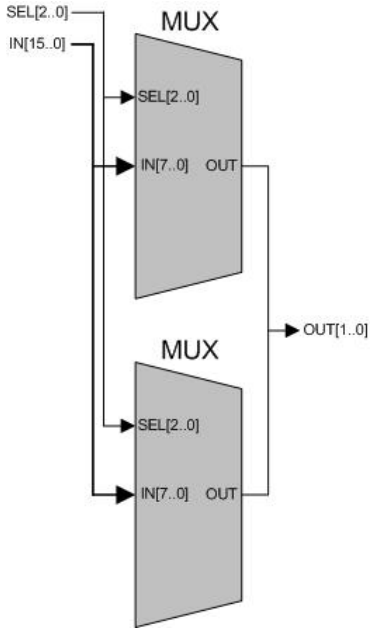


Figura 25: Implementação do Conversor de Dados com Multiplexadores

O Controlador do Barramento de Sensores (Figura 27) tem a função de fornecer uma janela da sequência de dados a todos os sensores conectados em paralelo. Este bloco possui na entrada um sinal de sincronização e um sinal de 2 bits que representa um nucleotídeo. Na saída deste bloco está o barramento de sensores propriamente dito, que é um sinal de 92 bits de largura. A cada pulso na entrada do controlador, um novo nucleotídeo é adicionado à janela deslocando os demais em uma posição (2 bits).

O Controlador do Barramento de Sensores foi implementado na forma de um *shift register*. O código VHDL que implementa este bloco está descrito na Figura 26. Nesta figura é possível notar a variável "reg". Esta variável é responsável por manter o dado no barramento de sensores propriamente dito. Também é possível observar que uma transição para nível lógico zero no sinal de sincronização "clk" provoca o deslocamento do dado da variável "reg" em duas posições e a inserção de 2 bits provenientes do sinal de entrada "data_in" nas posições 0 e 1 da

```
process(clk)
variable reg : std_logic_vector(91 downto 0);
begin
    if (clk'event and clk='0') then
        for i in 91 downto 2 loop
            reg(i) := reg(i-2);
        end loop;
        reg(1) := data_in(1);
        reg(0) := data_in(0);
    end if;

    sensor_bus <= reg;

end process;
```

Figura 26: Trecho do código em VHDL do Controlador do Barramento de Sensores

variável “reg”.

A Figura 27 mostra o controlador do barramento, o próprio barramento de sensores e alguns sensores interconectados. A largura do barramento de sensores é de 92 bits, valor necessário para conectar o sensor de maior janela deste trabalho; o sensor WWAM para *Acceptors*. Os demais sensores, que possuem entrada menores que 92 bits, conectam-se ao barramento de sensores a partir do bit menos significativo.

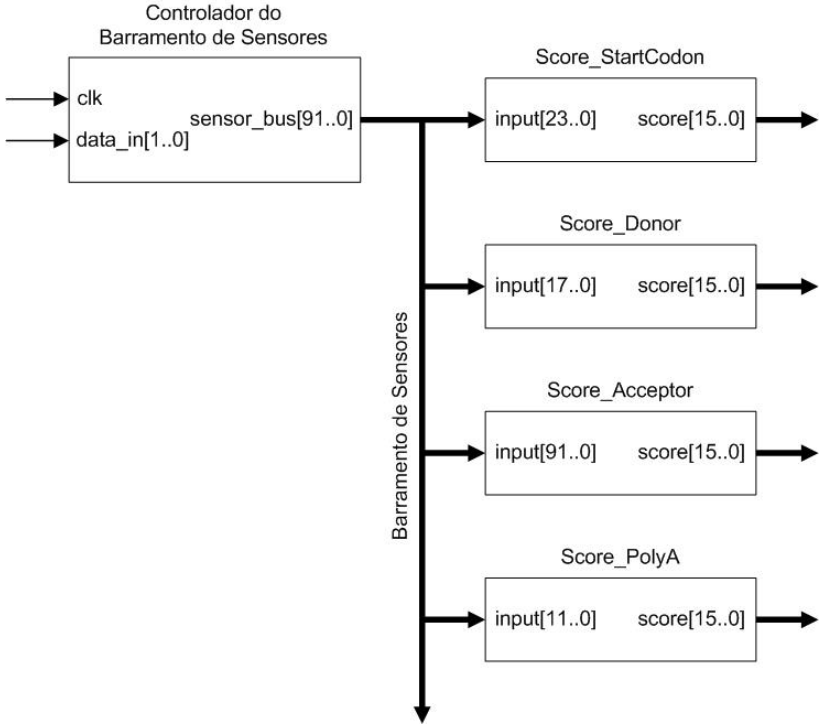


Figura 27: Diagrama do Barramento de Sensores

A interface com a memória SRAM externa é feita por dois blocos principais: o Contador de Sequência, conectado ao barramento de endereços da SRAM, e o SRAM Interface, conectado ao barramento de dados da SRAM.

Cada Sensor fornece um número inteiro (16 bits) como saída que representa a pontuação da sequência em sua janela. Se esse valor atinge o respectivo limiar, o Comparador sinaliza o evento em sua saída, e esta posição é então armazenada no banco de registradores do respectivo sensor. Estes blocos estão identificados na Figura 21 por Banco *Start Codon*, Banco *Donor*, Banco *Acceptor*, etc. Este esquema de sensor e comparador está representado simplificadamente na Figura 28. Nesta, o sensor está referenciado por Score_Donor.

O limiar, identificado na Figura 28 pelo sinal *threshold*, é fornecido pelo bloco Banco de Limiares. Este bloco está conectado ao barramento de dados e tem a função de armazenar os limiares e fornecê-los aos comparadores conectados à saída dos sensores.

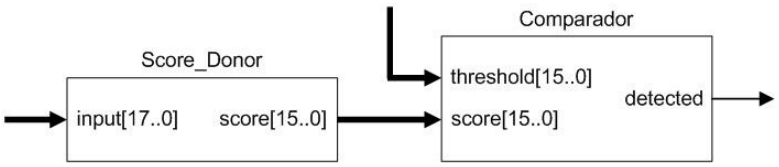


Figura 28: Diagrama do esquema sensor/comparador

Uma grande vantagem desta arquitetura é a possibilidade de fácil expansão para mais sensores, sem aumentar o tempo total de varredura.

O Controlador de Varredura, cuja interface está representada na Figura 29, é o principal bloco da arquitetura. É responsável por comandar os demais blocos de forma a executar o mecanismo de varredura em ciclos de 3 etapas que são:

1. Inserir um novo nucleotídeo no barramento de sensores.
2. Verificar se a pontuação de cada sensor atinge o limiar respectivo.
3. Incrementar o contador de sequência.

Verificou-se experimentalmente que a propagação do sinal no cálculo da pontuação leva aproximadamente 30ns. Portanto, são necessários 2 ciclos de *clock* (a 50MHz) para executar esta tarefa. A primeira tarefa (inserção de um nucleotídeo no barramento) e a terceira (incremento no contador de sequência) podem ser executadas em um ciclo de *clock*, mas foi decidido

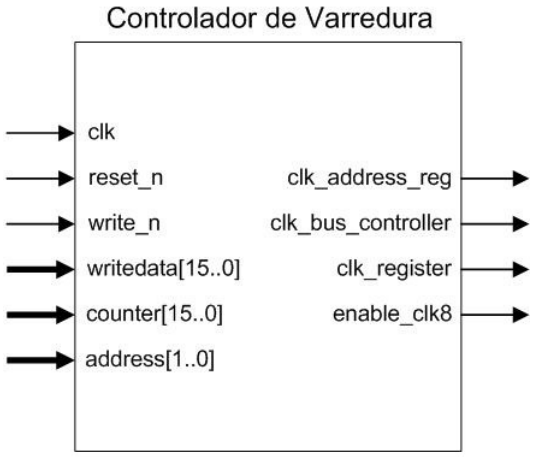


Figura 29: Diagrama do Controlador de Varredura

implementar um *pipeline* de 3 níveis, para reduzir o tempo total. Desta forma, a avaliação de uma janela da sequência de teste leva 40ns (2 ciclos de *clock* a 50 MHz), e não depende do número de sensores utilizados. A Figura 30 mostra as instruções em *pipeline* executadas pelo controlador de varredura.

A Figura 31 mostra como foi implementado o mecanismo de varredura na forma de uma máquina de estados implementada no bloco Controlador de Varredura. Dentro de cada estado estão representados os principais sinais de saída deste bloco.

O sinal "bus_ctrl" controla a inserção de um novo nucleotídeo no barramento de sensores. Este sinal é sensível à borda de descida.

A transição para nível zero do sinal "reg" faz com que o valor do Contador de Sequência seja armazenado nos Bancos de Registradores que estiverem com o sinal proveniente dos Comparadores em nível lógico alto.

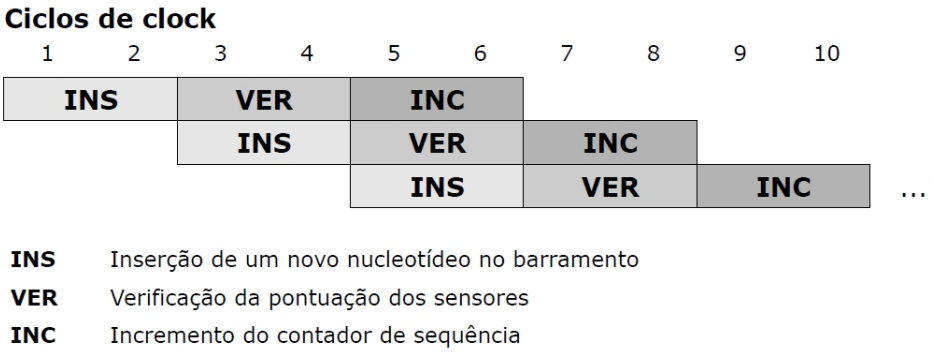


Figura 30: Instruções em pipeline executadas pelo sistema

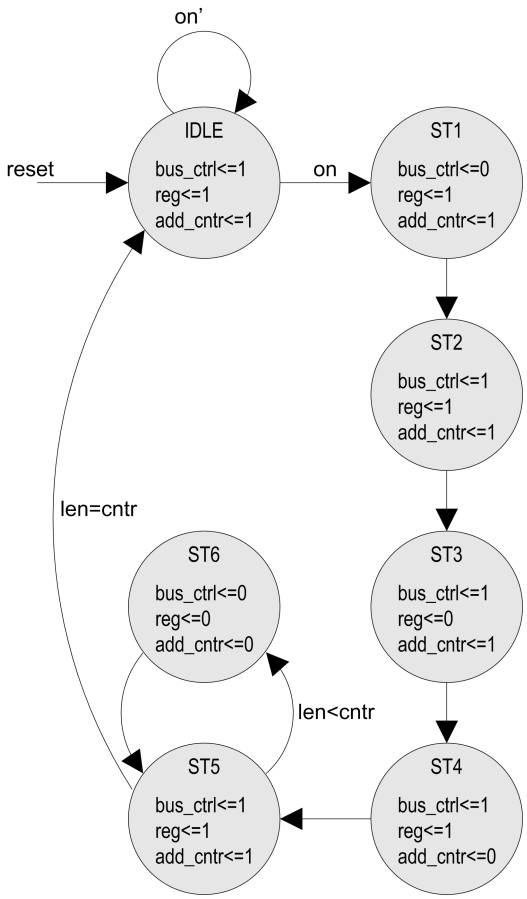


Figura 31: Máquina de estados do Controlador de Varredura

O sinal "add_cntr" é utilizado para incrementar o Contador de Sequência, que indica a posição da janela corrente na sequência de teste completa armazenada na SRAM.

A transição dos estados é sincronizada com o sinal de *clk*. Quando a máquina chega aos estados ST5 e ST6, esta permanece transitando entre estes dois estados até que toda a sequência de teste seja percorrida. Nestes dois estados o mecanismo de varredura executa as três etapas citadas em *pipeline*.

3.2.4 Sensores

Os sensores utilizados neste trabalho foram obtidos do *software* N-Scan (GROSS; BRENT, 2005), que é um *software* de predição de genes bem conhecido, desenvolvido na Universidade de Washington. O N-Scan é uma derivação do *software* Genscan (BURGE, 1997) com algumas modificações no modelo de gerenciamento para aumentar o desempenho preditivo.

Em (BURGE, 1997) o autor apresenta a fundamentação para a escolha dos sensores mais adequados para cada tipo de sinal. Estes sensores, na forma de matrizes de pesos, são os mesmos adotados pelo N-Scan, e que no presente trabalho foram descritos em *hardware* usando VHDL.

Tabela 7: Modelos dos sensores por sinal

Modelo	Sinal
WMM	Promotores (Tata-box)
WMM	*Start Codons*
MDD	*Donors*
WWAM	*Acceptors*
WMM	*Stop Codons* do tipo TAA
WMM	*Stop Codons* do tipo TAG
WMM	*Stop Codons* do tipo TGA
WMM	*Polyadenylation sites* (PolyA)

A Tabela 7 mostra os modelos dos sensores para cada um dos sinais biológicos utilizados neste trabalho.

Para verificar o funcionamento do *hardware* proposto foi realizado um experimento onde uma base de dados conhecida foi submetida ao N-Scan e ao *hardware* proposto. Algumas instâncias desta base de dados foram submetidas a três diferentes sensores, um WMM, o MDD e o WWAM. Neste experimento foi verificado que os sensores em *software* e *hardware* geram os mesmos *scores* para uma mesma sequência de entrada.

Para comparar o desempenho temporal, foi implementado um programa reduzido com as rotinas das matrizes de pesos originais do N-Scan sem o mecanismo de gerenciamento dos sensores. O programa verifica uma sequência de entrada com um modelo escolhido pelo usuário e mostra a pontuação da sequência, quando esta atinge um determinado valor limiar. Para fins de comparação, as rotinas das matrizes de peso também foram compiladas em outras duas plataformas: um processador ARM7, e no processador Nios II. Nestes processadores, a troca de dados com o usuário é feita pela interface serial, auxiliado por outro aplicativo executado no PC.

Nenhuma alteração foi feita nas rotinas originais retiradas do N-Scan, a fim de reduzir o tempo de execução ou otimizar o código. Nesta etapa, também não houve preocupação com a precisão de classificação dos modelos escolhidos, uma vez que o principal objetivo aqui foi propor uma metodologia para acelerar a análise dos dados. Os resultados deste experimento estão descritos em detalhes no Capítulo 4.

Como mencionado anteriormente, cada nucleotídeo foi representado por uma palavra de 2 bits sendo: ‘A’ (00), ‘C’ (01), ‘G’ (10) e ‘T’ (11).

Um diagrama em blocos simplificado de um sensor WMM é mostrado na Figura 32. Cada coluna do WMM foi implementada como um simples bloco de *hardware* com uma palavra de 2

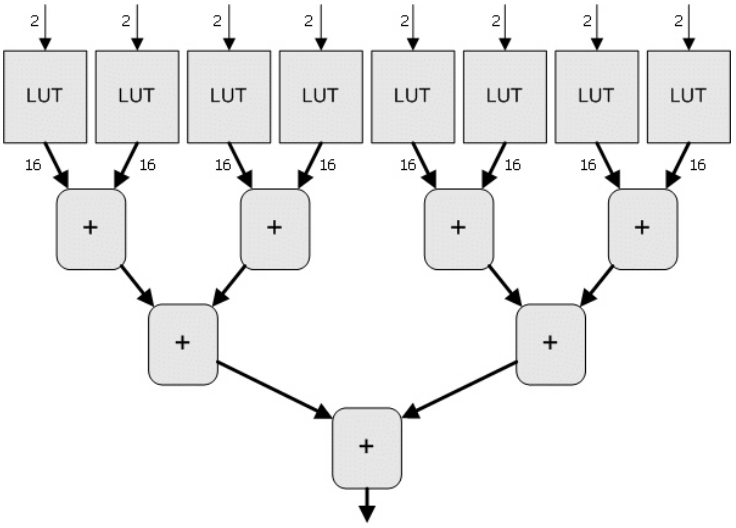


Figura 32: Diagrama de sensor WMM descrito em *hardware*

bits como sinal de entrada e uma palavra de 16 bits como sinal de saída, na forma de uma *Look up Table* (ou simplesmte, LUT).

A Tabela 8 mostra a matriz de pesos utilizada para o sinal *Start Codon*, e na Figura 33 mostra um trecho do código em VHDL que apresenta a forma como foram implementadas as LUTs. Nesta figura são mostradas apenas as 3 primeiras LUTs referentes às 3 primeiras colunas da matriz de pesos.

Tabela 8: WMM para o sinal *Start Codon*

A	-5	-6	-2	7	-2	-8	18	-100	-100	-3	-1	-9
C	3	7	10	-9	13	9	-100	-100	-100	-4	10	5
G	7	4	0	5	-4	2	-100	-100	23	9	-5	6
T	-7	-7	-12	-22	-10	-14	-100	18	-100	-11	-8	-5

O tempo de processamento desses blocos básicos não depende do comprimento da matriz. Esse período é o mesmo para todas as matrizes, porque todos os blocos são processados em paralelo. O tempo de processamento final será aumentado apenas pela inclusão dos somadores. Por consequência, esta abordagem pode atingir um desempenho maior que as tradicionais arquiteturas que executam *software* sequencial. A pontuação final é calculada pela soma dos resultados de cada coluna do bloco.

A Figura 34 mostra a mesma estratégia adotada para o modelo WWAM. Neste modelo existem dependências entre os nucleotídeos próximos, como observado na entrada de cada LUT,

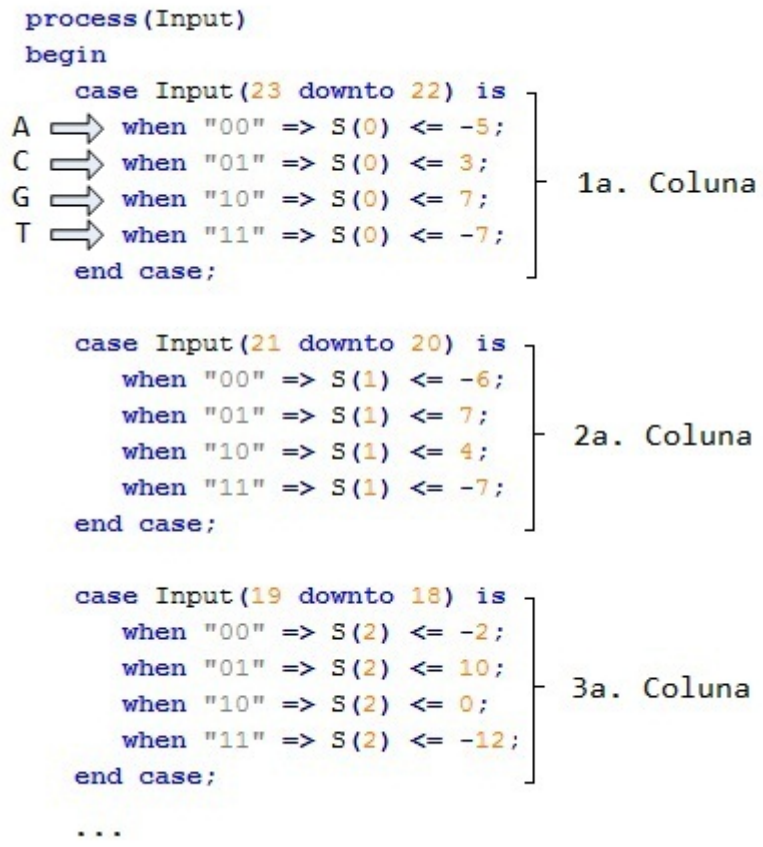


Figura 33: Trecho do código em VHDL para o WMM do sinal *Start Codon*

que é de 8 bits. Este modelo utiliza um número maior de elementos lógicos. Entretanto, sua versão em *hardware* tende a alcançar maior *speed up* em relação à versão em *software*, que o equivalente WMM. Isto se deve ao fato de que este modelo em *hardware* possui uma quantidade maior de instruções executadas em paralelo.

Como as LUTs do modelo WWAM possuem 8 bits de entrada, a sua implementação em VHDL necessita de 256 *whens* para cada *case*. A Figura 35 mostra um trecho do código em VHDL que implementa uma LUT do modelo WWAM.

O modelo MDD é composto de duas partes: uma árvore de decisão, que foi implementada com instruções do tipo *if-then-else* do VHDL, e WMMs em cada extremidade da árvore (Figura 10). A implementação em *hardware* do WMM já foi mostrada.

A Tabela 9 mostra a largura da janela e o número total de elementos lógicos em cada sensor. O percentual mostrado na última coluna refere-se ao percentual de elementos lógicos do dispositivo utilizado neste trabalho, o FPGA Cyclone II EP2C20F484C.

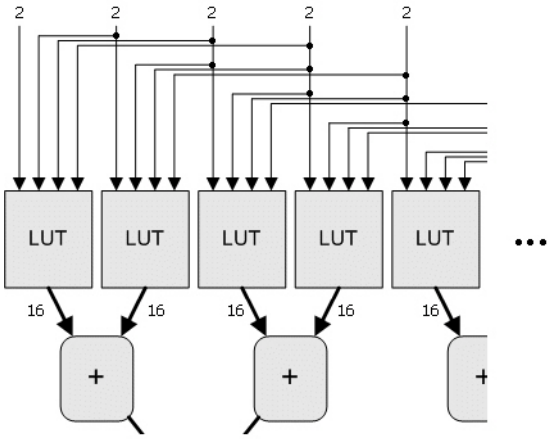


Figura 34: Diagrama de sensor WWAM descrito em *hardware*

```
    process(Input)
    begin
       case Input(91 downto 84) is
AAAA ⇨ when "00000000" => S(0) <= 0;
AAAC ⇨ when "00000001" => S(0) <= -1;
AAAG ⇨ when "00000010" => S(0) <= -3;
AAAT ⇨ when "00000011" => S(0) <= 4;
AACA ⇨ when "00000100" => S(0) <= 1;
AACC ⇨ when "00000101" => S(0) <= 0;
AACG ⇨ when "00000110" => S(0) <= -6;
AACT ⇨ when "00000111" => S(0) <= 1;
 ...   when "00001000" => S(0) <= 0;
       when "00001001" => S(0) <= -1;
       when "00001010" => S(0) <= -1;
       when "00001011" => S(0) <= 3;
       when "00001100" => S(0) <= 1;
       when "00001101" => S(0) <= -4;
       when "00001110" => S(0) <= 0;
       when "00001111" => S(0) <= 1;
       when "00010000" => S(0) <= 0;
       when "00010001" => S(0) <= 0;

       ...
```

Figura 35: Trecho do código em VHDL para o WWAM do sinal *Acceptor*

Tabela 9: Taxa de utilização dos sensores

Modelo	Sensor	Largura da janela	No. elementos lógicos
WMM	Promotor	30 bits	157 (<1%)
WMM	*Start Codon*	24 bits	108 (<1%)
MDD	*Donor*	18 bits	146 (<1%)
WWAM	*Acceptor*	92 bits	1479 (8%)
WMM	*Stop Codon*	12 bits	39 (<1%)
WMM	*PolyA*	12 bits	47 (<1%)

3.2.5 Aplicativo de Interface com o Usuário

Para interagir com o sistema em *hardware* reconfigurável foi necessária a implementação de um aplicativo de interface com o usuário. Através deste, é possível enviar as sequências de teste, enviar e receber os limiares de corte e receber os resultados da varredura.

O aplicativo, cuja tela principal está mostrada na Figura 36, foi desenvolvido utilizando a linguagem C# e compilado no Microsoft Visual C# 2008 Express Edition.

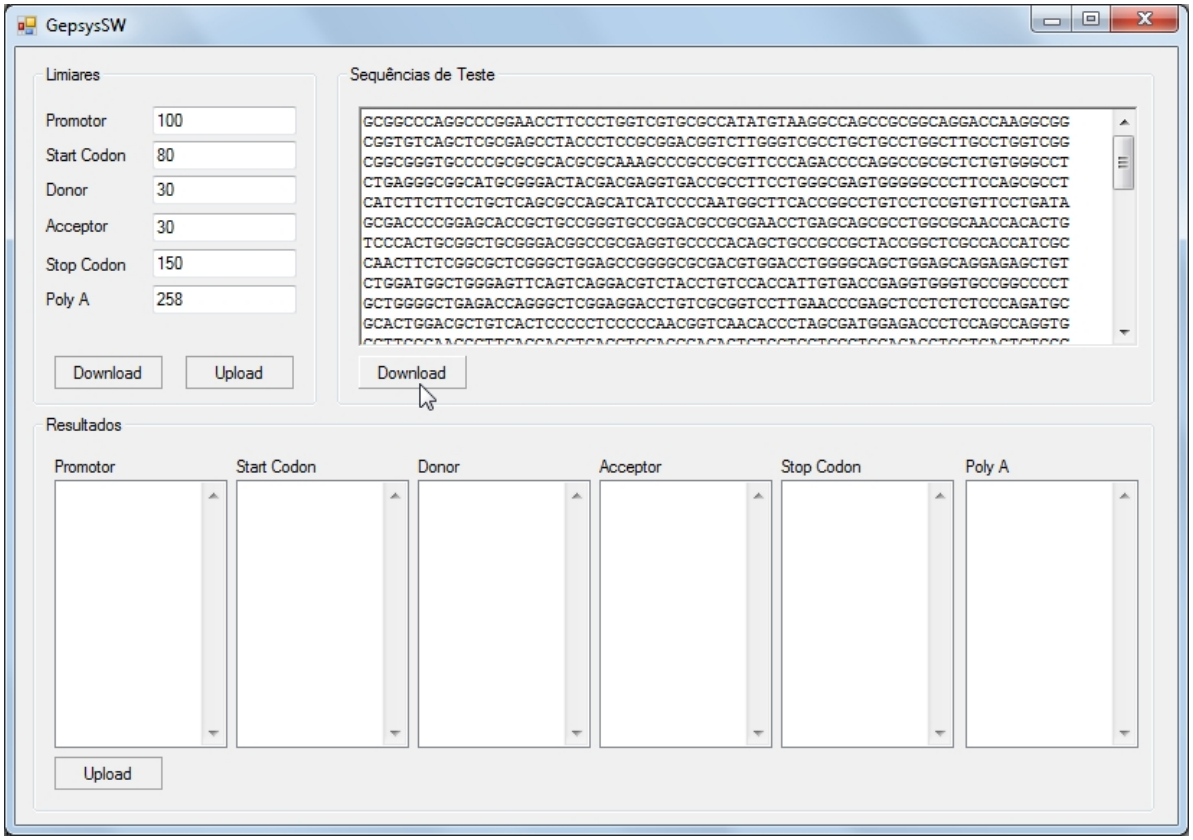


Figura 36: Aplicativo de Interface com o Usuário

3.2.6 *Software* Embarcado

O *software* embarcado foi desenvolvido para realizar as tarefas de mais alto nível, como por exemplo a comunicação com o aplicativo do usuário, o controle do bloco de varredura e etc. Este programa foi escrito em linguagem C e compilado no Nios II IDE, versão 6.1.

A comunicação com o aplicativo do usuário é feita através da interface serial disponível no kit utilizado. Foram implementados 5 comandos destinados ao comando do sistema embarcado que estão descritos na Tabela 10. A descrição dos comandos se refere ao aplicativo do usuário.

Tabela 10: Comandos do *Software* Embarcado

Comando	Descrição
0x00	Envia limiares dos sensores
0x01	Recebe limiares dos sensores
0x02	Envia sequência de teste
0x03	Inicia varredura
0x04	Recebe resultado da varredura

A Figura 37 mostra uma representação simplificada do comportamento do *software* embarcado em resposta aos comandos recebidos pelo aplicativo do usuário. Quando o sistema é resetado, o *software* realiza algumas inicializações e em seguida, vai para o estado *default* de "Aguardando_Comando" onde permanece até que um comando (Tabela 10) seja recebido do aplicativo do usuário. Na Figura 37 estão indicados os comandos (por CMD *N*) que provocam uma transição temporária. Ao final da execução de cada comando o sotware volta ao estado *default* de "Aguardando_Comando".

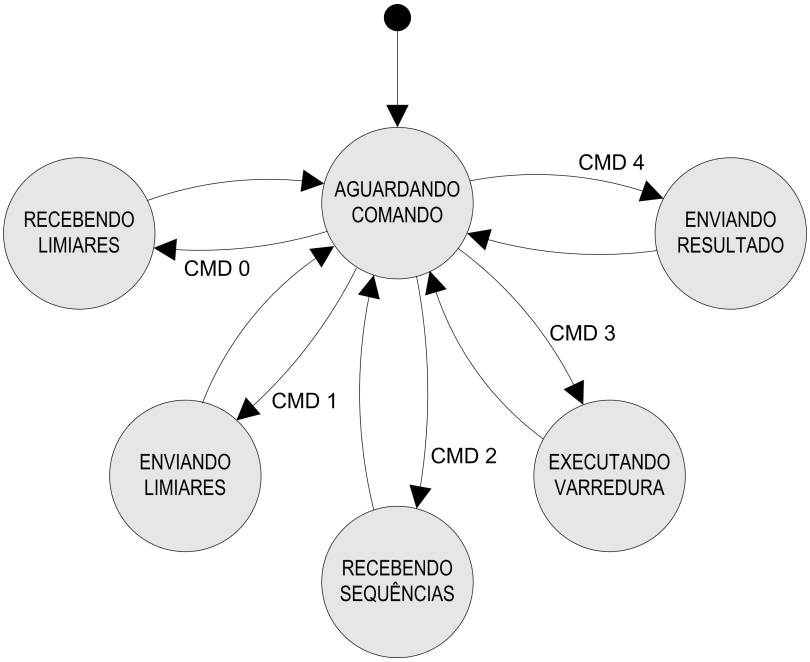


Figura 37: Representação simplificada do comportamento do *software* embarcado

4 EXPERIMENTOS E RESULTADOS

Este capítulo descreve os principais experimentos realizados e os resultados obtidos.

Inicialmente foi conduzido um experimento com o objetivo de avaliar a técnica conhecida como Modelo da Matriz de Pesos em software. Esta foi observada por possuir uma característica interessante, que é o paralelismo inerente ao modelo. Duas variações deste modelo (WWAM e MDD) foram avaliadas. Os demais experimentos foram realizados para avaliar a arquitetura proposta.

Primeiramente foi realizado um experimento para avaliar o funcionamento dos sensores implementados em *hardware*. Neste experimento, os resultados obtidos pelo *hardware* foram comparados aos obtidos com o *software* NScan.

Posteriormente foi realizado um estudo comparativo do desempenho temporal dos sensores individualmente. Neste, várias arquiteturas foram avaliadas e comparadas com a arquitetura proposta.

O último experimento se destinou a avaliar o mecanismo de varredura das sequências de testes em relação às abordagens tradicionais, baseadas em *software*.

4.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento deste sistema foi utilizado um kit da Terasic denominado Cyclone II FPGA Starter Development Board.

Dentre os principais componentes, este kit possui:

- FPGA Cyclone II EP2C20F484C7N;
- SDRAM de 8 MBytes;
- SRAM de 512 kbytes;
- Memória Flash de 4 MBytes;

- Interface para SD Card;
- Interface Serial RS-232;
- *Push-buttons*, *Switches*, Leds, Displays de 7 segmentos, etc.

A FPGA disponível neste kit possui as seguintes características (Tabela 11):

Tabela 11: Características da Cyclone II EP2C20F484C7N

Característica	Quantidade
Elementos lógicos	18752
Blocos de M4K RAM	52
Total de RAM bits	239616
Multiplicadores	26
PLLs	4
Pinos de I/O disponíveis	315

Para o desenvolvimento do aplicativo de interface com o usuário foi utilizado o Microsoft Visual C# 2008 Express Edition. Este faz parte do pacote de programas da Microsoft que se utilizam do Framework .NET (em inglês *dot net*), e é atualmente distribuído livremente.

O Visual C# foi escolhido por ser uma linguagem simples, moderna, orientada a objetos, apresentar muita semelhança com as linguagens C e C++ e possuir um ambiente de desenvolvimento extremamente eficiente e prático.

4.2 MATRIZES DE PESOS

Neste experimento foram avaliados dois modelos derivados da Matriz de Pesos aplicados ao problema da detecção de regiões de *splicing* de eucariotos (outro subproblema da detecção de genes).

Os modelos escolhidos, propostos Burge (BURGE; KARLIN, 1997), são conhecidos como: *Maximal Dependence Decomposition* (MDD), utilizado na detecção de *donors*; e o *Windowed Weight Array Model* (WWAM), utilizado na detecção de *acceptors*. Ambos são largamente utilizados em vários programas de predição de genes atualmente em uso (BURGE; KARLIN, 1997; KORF et al., 2001; GROSS; BRENT, 2005; MAJOROS; PERTEA; SALZBERG, 2004; ARTHUR et al., 1998; DELCHER et al., 1999).

4.2.1 Bancos de Dados

Dois bancos de dados de *splice junctions* foram utilizados neste estudo. O primeiro foi obtido do conhecido repositório de dados *UCI Machine Learning Repository* (BLAKE; MERZ, 1998) e trata-se de um banco de dados de DNA de primatas.

Embora esse banco de dados possua 3.190 instâncias no total, algumas pequenas mudanças foram feitas neste trabalho. Em primeiro lugar, todas as seqüências contendo valores diferentes dos símbolos de nucleotídeos padrão 'A', 'C', 'G' e 'T' foram removidas. Em seguida, algumas sequências foram escolhidas aleatoriamente para serem removidas de forma a obter um banco de dados com o número de instâncias por classe em uma proporção específica: 25% para os *donors*, 25% para *acceptors* e 50% para outras sequências, de modo a manter o conjunto de dados aproximadamente igual ao original. Tal equilíbrio entre as classes também evita distorções na avaliação dos classificadores.

A segunda base de dados utilizada foi retirada do *Homo Sapiens Splice Sites Data Set* (HS3D) (POLLASTRO; RAMPONE, 2002). Este banco de dados contém originalmente 5.676 para sequências de DNA humano com os *splice sites* conhecidos (*donors* e *acceptors*) e 604.233 sequências com falsos *splice sites*. Para este conjunto de dados, 11.184 sequências foram escolhidas aleatoriamente, distribuídas na mesma proporção do conjunto de dados anterior: 25% para *donors*, 25% para *acceptors* e 50% para *splice sites* falsos. Instâncias com valores diferentes de 'A', 'C', 'G' e 'T' foram descartadas.

A Tabela 12 mostra os conjuntos de dados utilizados neste experimento. Contém o número total de casos (No. de Instâncias) e distribuídos por classe (Instâncias por Classe). EI, IE e N significam junção *exon-intron* (*donor*), junção *intron-exon* (*acceptor*) e falso *splice site*, respectivamente.

Tabela 12: Resumo dos bancos de dados de *splice junctions*

Banco de dados	**No. de Instâncias**	**Instâncias por Classe (%)**		
		EI (25%)	**IE** (25%)	**N** (50%)
UCI	3048	762	762	1524
HS3D	11184	2796	2796	5592

4.2.2 Ferramentas computacionais

As matrizes e o código fonte das rotinas utilizadas neste trabalho foram obtidos do N-Scan (GROSS; BRENT, 2005).

Este programa foi escolhido não apenas por possuir código fonte aberto, mas principalmente por possuir um dos maiores desempenhos preditivos entre os *softwares* de predição de genes atualmente em uso. O N-Scan está disponível para download em http://mblab.wustl.edu.

4.2.3 Experimento e resultados

Ambos os modelos (MDD e WWAM) foram executados em um PC com os dois conjuntos de dados (UCI e HS3D) variando o limiar de corte de -80 a 100, em passos unitários.

Para cada valor de limiar, os valores correspondentes de *VP*, *FP*, *FN* e *VN* obtidos da classificação foram registrados. Em seguida, *Sp*, *Sn* e *MCC* foram calculados.

A Figura 38 mostra os gráficos resultantes da avaliação do modelo MDD aplicado aos dados do UCI. Em (a) é possível observar o comportamento de *Sp*, *Sn* e *MCC* para diferentes limiares. Neste, o eixo vertical representa o valor absoluto de *Sp*, *Sn* e *MCC*, e o eixo horizontal representa os valores dos limiares de corte utilizados em cada avaliação. Em (b) é possível observar o gráfico ROC para o mesmo modelo e conjunto de dados. Como mencionado anteriormente, o gráfico ROC representa nos eixos horizontal e vertical, $1 - Sp$ e *Sn*, respectivamente.

A Figura 39 mostra os gráficos correspondentes à avaliação do MDD aplicado ao banco de dados HS3D. É possível observar que o comportamento de *Sp*, *Sn* e *MCC* é semelhante ao primeiro caso (MDD aplicado banco de dados do UCI).

Observando a curva de *MCC*, é possível notar que o valor máximo de *Sp*, *Sn* e *MCC* é aproximadamente igual para os dois casos. No entanto, os gráficos ROC mostram que este modelo é mais sensível ao limiar quando aplicados ao banco de dados HS3D. Isto é devido ao fato de que este conjunto de dados apresenta um desafio maior para os métodos de classificação.

A Tabela 13 mostra os valores com o melhor custo-benefício para este modelo. O melhor limiar para classificação foram similares para ambos os bancos de dados (41 e 44).

A mesma metodologia foi empregada para o modelo WWAM. A Figura 40 mostra os resultados da avaliação do modelo WWAM aplicado ao banco de dados UCI. Em (a) é possível observar o comportamento da *Sp*, *Sn* e *MCC*, e em (b) o gráfico ROC para o mesmo modelo e conjunto de dados. De maneira similar, a Figura 41 mostra os gráficos correspondentes à avaliação do modelo WWAM aplicado ao banco de dados HS3D.

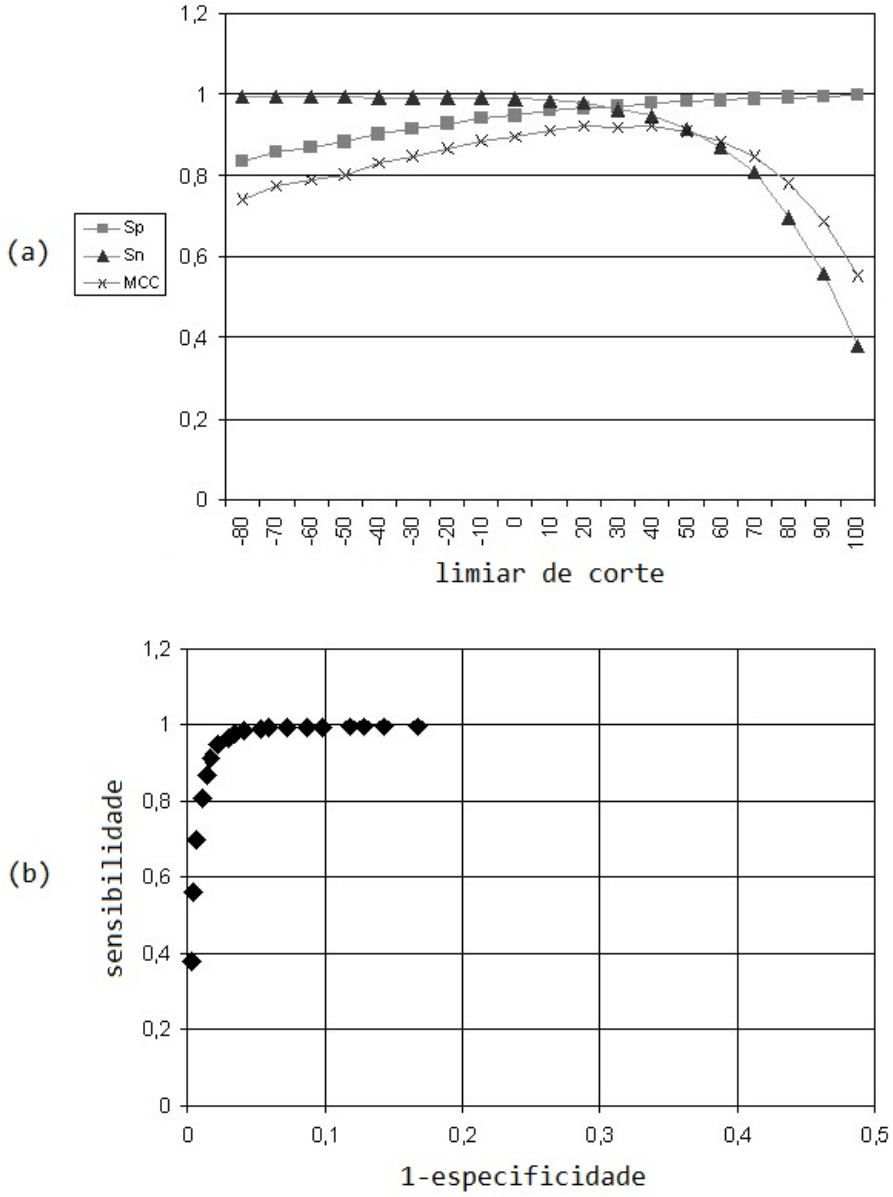


Figura 38: MDD aplicado ao banco de dados UCI

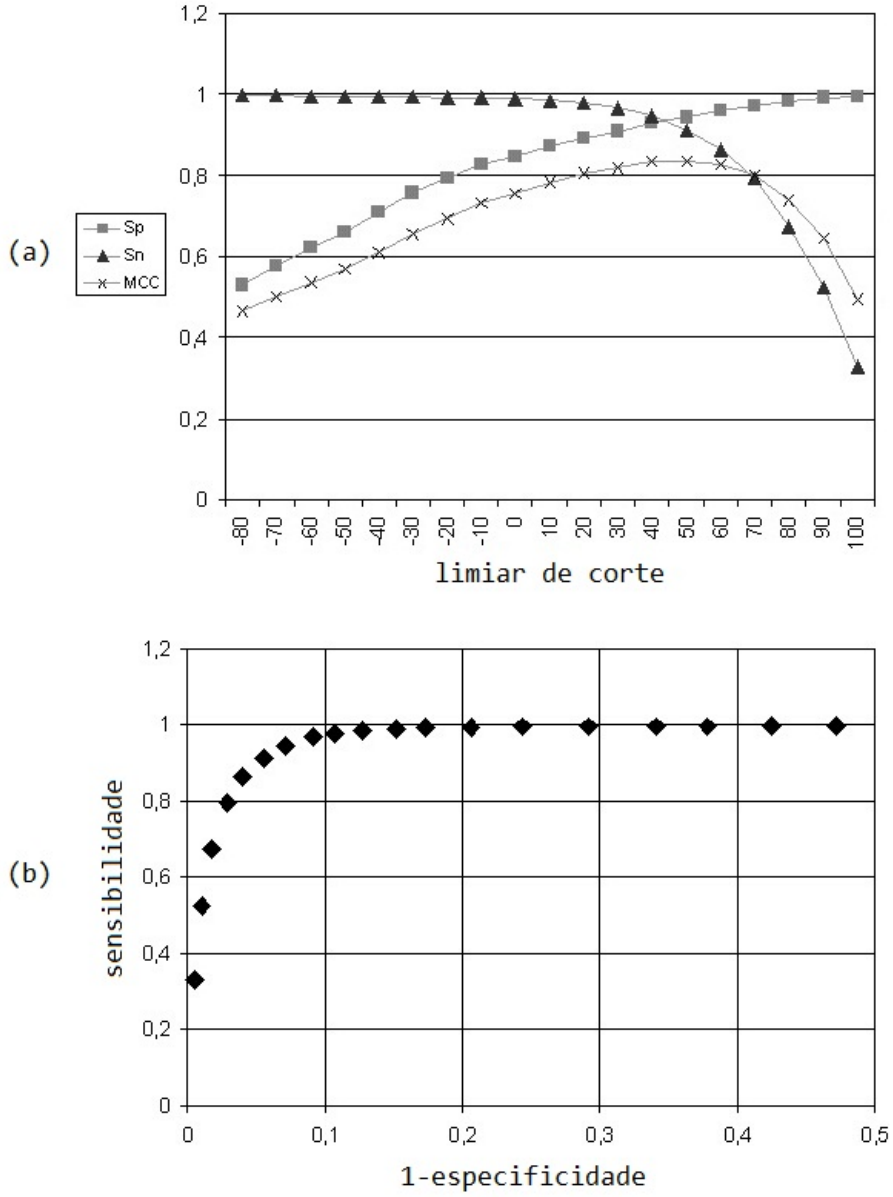


Figura 39: MDD aplicado ao banco de dados HS3D

Tabela 13: Limiar ótimo para o MDD

Banco de dados	Limiar Ótimo	Sp	Sn	MCC
UCI	41	0,979	0,948	0,92330
HS3D	44	0,936	0,936	0,83948

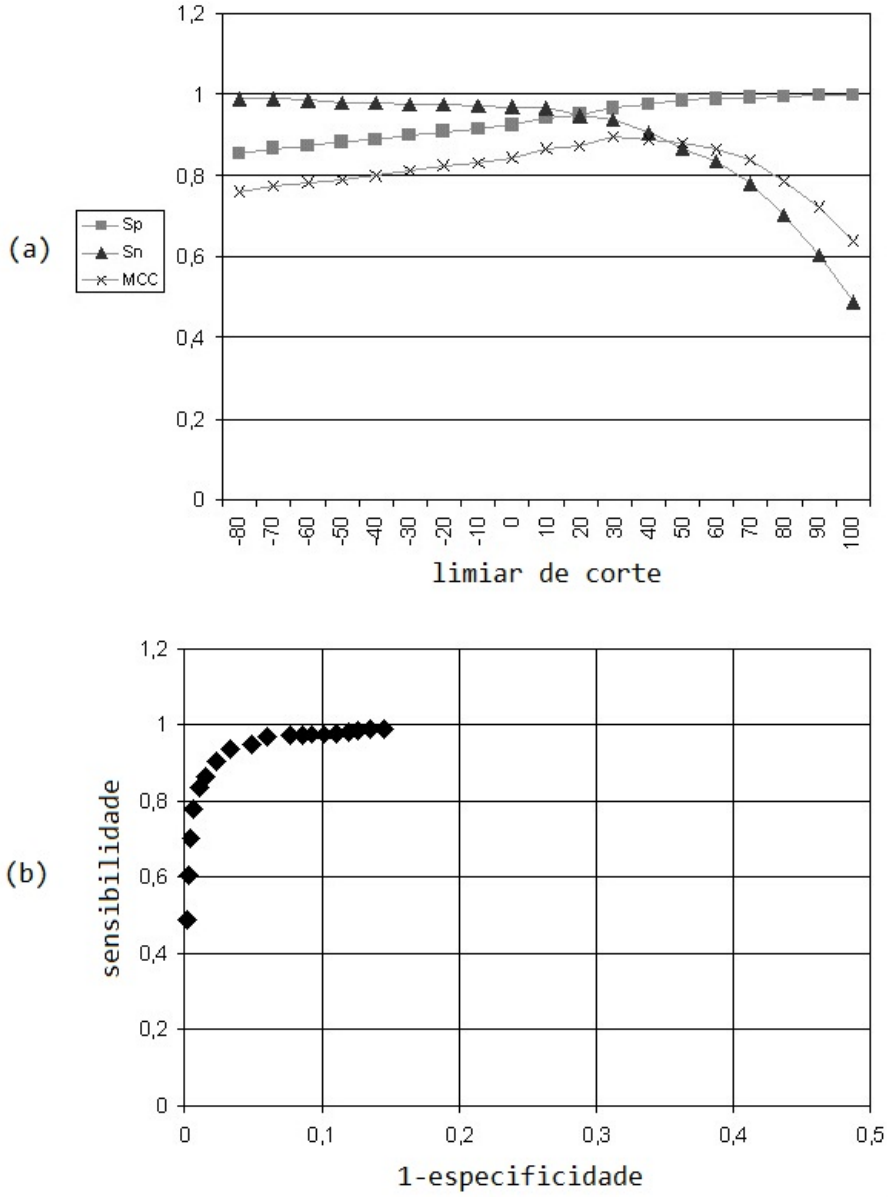


Figura 40: WWAM aplicado ao banco de dados UCI

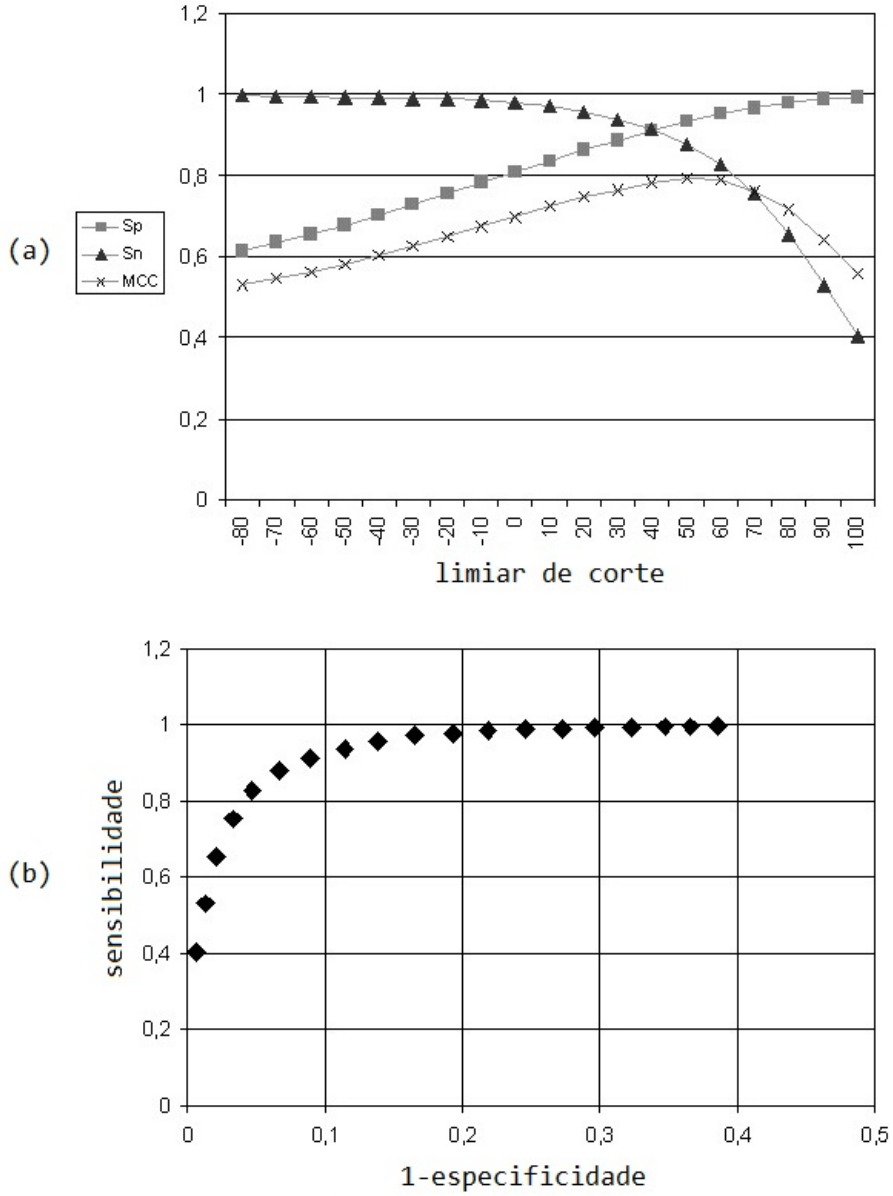


Figura 41: WWAM aplicado ao banco de dados HS3D

Os resultados para o segundo modelo (o WWAM) foram ligeiramente semelhantes aos do MDD, exceto para o melhor limiar. Neste caso, os melhores limiares não eram tão próximos entre si do que na avaliação do MDD. A Tabela 14 mostra que a melhor situação ocorre para um limiar de cerca de 31 para o conjunto de dados do UCI, e cerca de 50 para a base de dados HS3D.

Tabela 14: Limiar ótimo para o WWAM

Banco de dados	Limiar ótimo	Sp	Sn	MCC
UCI	31	0,969	0,936	0,89618
HS3D	50	0,933	0,877	0,79133

A comparação dos gráficos ROC confirma o fato de que o banco de dados HS3D apresenta maior dificuldade de classificação que o banco de dados UCI, para ambos os sinais, *donors* e *acceptors*.

Os resultados deste experimento foram publicados em (TAVARES; LOPES; LIMA, 2009).

4.3 FUNCIONAMENTO DOS SENSORES EM *HARDWARE*

Este experimento se destinou a verificar a exatidão dos sensores implementados em *hardware* comparando-os com os originais contidos no *software* N-Scan.

4.3.1 Banco de Dados

O banco de dados utilizado neste experimento é um conjunto sequências de DNA que contém genes de mamíferos. Este conjunto foi compilado por Rogic et al. (ROGIC; MACKWORTH; OUELLETTE, 2001) com a finalidade de realizar uma comparação entre diversos *softwares* de predição de genes.

A base, chamada de HMR195, possui 195 instâncias e possui as seguintes características:

- Todas as sequências foram extraídas do Genbank;
- As sequências correspondem a genes de Homo sapiens (103), Mus musculus (82) e Rattus norvegicus (10);
- Cada sequência possui um único gene;
- O tamanho médio das sequências é de 7096 pares de bases;
- O número médio de éxons por gene é 4,86.

A base HMR195 está disponível em http://blogs.ubc.ca/sanja/datasets/hmr195-dataset/.

4.3.2 Ferramentas Computacionais

Foram utilizados neste experimento o sistema proposto baseado em *hardware* auxiliado pelo aplicativo do usuário e o próprio *software* N-Scan.

4.3.3 Experimento e resultado

Foram avaliados 3 sensores de diferentes modelos de matrizes de pesos:

- WMM para o sinal *Start Codon*;
- MDD para o sinal *Donor*;
- WWAM para o sinal *Acceptor*;

Foram escolhidas aleatoriamente 3 instâncias do conjunto HMR195 para esta avaliação.

Para cada sequência, o *software* N-Scan foi executado 3 vezes. Na primeira vez foi solicitado que este listasse todas as ocorrências de *Start Codons* na fita direta e na fita reversa, isto é, considerando a sequência de entrada do início para o fim e do fim para o início. Os valores da posição de cada ocorrência e os respectivos *scores* obtidos foram anotados. O mesmo processo foi executado outras 2 vezes solicitando as ocorrências de *Donors* e *Acceptors*.

As mesmas 3 sequências escolhidas também foram submetidas ao sistema proposto baseado em *hardware*. Com este sistema, que varre a sequência de teste com todos os sensores em paralelo, foi obtida a posição de ocorrência de cada um dos sinais para cada sequência. Obviamente que nem todas as ocorrências obtidas pelo processo de varredura correspondem a verdadeiros positivos. Entretanto o objetivo deste experimento é verificar se a implementação em *hardware* tem o mesmo comportamento do N-Scan, e não avaliar a qualidade preditiva das matrizes de pesos.

Em todos os casos, os resultados obtidos com o N-Scan foram iguais ao sistema proposto em *hardware*, mostrando que a implementação dos sensores em *hardware* é adequada. A Tabela 15 mostra o número de ocorrências de cada sinal para cada sequência utilizada. A primeira coluna mostra o nome da instância. A segunda coluna indica o tamanho da sequência em pares de bases. Nas demais colunas são apresentados os números de ocorrência do respectivo sinal na fita direta e na fita reversa.

Tabela 15: Resultado da varredura de sequências

Sequência	**Tamanho (bp)**	**No. *Start Codons***		**No. *Donors***		**No. *Acceptors***	
		Direta	Reversa	Direta	Reversa	Direta	Reversa
AB021866	3611	32	31	194	165	274	336
AF019563	3877	48	44	188	203	301	315
U25134	3416	54	52	180	206	255	264

4.4 TEMPO DE PROCESSAMENTO DOS SENSORES

Este experimento se destinou a medir o tempo de execução dos sensores individualmente, em cada abordagem. Como mencionado anteriormente, o código fonte, escrito em linguagem C, foi compilado e executado em três diferentes arquiteturas. Primeiro, o programa foi executado em PC, com processador AMD Sempron 2800+ de 1,59 GHz com 1 GByte de memória RAM, em Windows XP. As rotinas que implementam os modelos das matrizes de pesos, foram executadas inúmeras vezes, e o tempo médio de processamento foi calculado, utilizando o número de *tick counts* do processador.

O segundo sistema utilizado para medir o tempo de processamento foi um kit baseado no processador Atmel ARM7 AT91M55800, com *clock* de 32 MHz. Este tipo de processador foi escolhido para dar uma idéia do tempo de processamento dos modelos propostos utilizando um processador embarcado. Um analisador lógico foi utilizado para medir o tempo de processamento. O mesmo procedimento foi executado no processador Nios II, da Altera. Este processador foi escolhido porque também é utilizado na versão em *hardware*. A parte mais importante deste experimento foi a obtenção do tempo de processamento da versão dos modelos de matriz de pesos em *hardware*.

A síntese do *hardware* foi feita em um dispositivo FPGA Cyclone II EP2C20F484C7, com *clock* de 50MHz. Para a compilação, simulação e realização da síntese do circuito descrito, utilizou-se o Altera Quartus II versão 8.1 Web Edition e o Altera SOPC Builder versão 8.1. O firmware a ser executado no processador Nios II foi escrito em linguagem C e compilado no ambiente Nios II IDE, versão 6.1.

Nesta versão foram utilizados inteiros de 16 bits para representar as pontuações e 2 bits para representar os nucleotídeos. O sistema foi executado diversas vezes e o desempenho foi medido com auxílio de um analisador lógico. Alguns sinais internos foram ligados a pinos externos da FPGA para permitir a amostragem e verificação de desempenho.

A Tabela 16 mostra o tempo de execução, em nanosegundos, para as quatro abordagens diferentes. Em cada arquitetura, o tempo de processamento multiplicado pelo respectivo *clock* fornece uma aproximação do número de ciclos de *clock* envolvidos no processo e, assim, pode-se avaliar a eficiência da arquitetura. A Tabela 17 mostra o número aproximado de ciclos de *clock* nas diferentes arquiteturas.

A Tabela 18 mostra a quantidade de elementos lógicos do FPGA utilizados pelos principais blocos da plataforma utilizada. Nesta tabela também é possível verificar o percentual de elementos lógicos utilizados em relação ao dispositivo utilizado, o FPGA Cyclone II EP2C20F484C7.

Tabela 16: Tempo de processamento (em nanosegundos)

Model	Sensor	PC	ARM7	Nios II	HW
WMM	Promoter	300	50800	65500	24
WMM	Start Codon	250	42800	55300	24
MDD	Donor	220	37400	44700	28
WWAM	Acceptor	5160	556000	1156000	28
WMM	Stop Codon	150	26400	34100	24
WMM	PolyA	150	26400	34100	24

Tabela 17: Número aproximado de ciclos de clock

Model	Sensor	PC	ARM7	Nios II	HW
WMM	Promoter	477	1626	3275	2
WMM	Start Codon	398	1370	2765	2
MDD	Donor	350	1197	2235	2
WWAM	Acceptor	8205	17792	57800	2
WMM	Stop Codon	239	845	1705	2
WMM	PolyA	239	845	1705	2

4.5 TEMPO DE PROCESSAMENTO DO MECANISMO DE VARREDURA

Este experimento foi destinado a medir o tempo de execução do mecanismo de varredura da arquitetura proposta. Neste experimento não foram considerados os tempos de envio e armazenamento das sequências na memória.

Tabela 18: Número de elementos lógicos utilizados

Bloco	**No. elementos lógicos**	**Percentual**
Total	10690	57,01%
CPU (Nios II)	2297	12,25%
Bloco de Varredura	7802	41,61%
Sensor Promotor	157	0,84%
Sensor *Start Codon*	108	0,58%
Sensor *Donor*	146	0,78%
Sensor *Acceptor*	1479	7,89%
Sensor *Stop Codon*	39	0,21%
Sensor *PolyA*	47	0,25%

Usando um analisador lógico, foi possível medir o tempo de processamento de uma única janela pelo mecanismo de varredura. Esse tempo é 40 ns, exatamente 2 ciclos de *clock* a 50 MHz. O correto funcionamento do mecanismo de *pipeline* também pôde ser observado com auxílio do analisador lógico.

Para processar 1 MByte de sequências de DNA, a solução em *hardware* gasta aproximadamente 42 ms. Entretanto, a execução de todos os sensores de forma sequencial, em um PC (com *clock* de 1.59GHz) exigiu 12 s. Estes valores demonstram um *speed up* de aproximadamente 280 vezes.

Sabe-se contudo que o tempo de envio das sequências é muito alto, levando-se em consideração que foi utilizada uma interface de comunicação lenta. Outro fato é que o próprio N-Scan utiliza outros recursos para evitar que a sequência de teste inteira seja varrida pelos sensores, principalmente pelo mais lento, o WWAM.

5 DISCUSSÃO E CONCLUSÕES

5.1 DISCUSSÃO DOS RESULTADOS

Nesta seção são apresentadas as análises dos resultados obtidos, seguindo a ordem em que estes foram apresentados no capítulo anterior.

5.1.1 Matrizes de Pesos

Este experimento teve por objetivo a investigação de dois modelos derivados do Modelo da Matriz de Pesos: o *Maximal Dependence Decomposition* (MDD) e o *Windowed Weight Array Model* (WWAM), ambos aplicados ao problema da detecção de *splice junctions*. O MDD se destina ao reconhecimento de *donors*, e o WWAM ao reconhecimento de *acceptors*. A questão de estabelecer um limiar de classificação foi dirigido para ambos os modelos, avaliados com dois conjuntos de dados diferentes.

As medidas de desempenho preditivo usando *Sn*, *Sp* e *MCC*, bem como o gráfico ROC permitiram uma boa avaliação dos resultados. Em particular, o *MCC* oferece uma boa forma de avaliação, sugerindo a sua adequação para outros problemas de classificação semelhantes em bioinformática.

Os experimentos realizados mostraram uma pequena diferença no desempenho dos dois métodos, MDD e WWAM, em favor do primeiro. Entretanto, ambos os métodos apresentaram desempenho satisfatório.

A comparação direta dos resultados aqui obtidos com outros trabalhos, pode levar a erros de interpretação se não forem observados os procedimentos de validação cruzada que possam ter sido adotados. No entanto, como mostrado em (TAVARES; LOPES; LIMA, 2008), os métodos probabilísticos (a exemplo do WWAM e do MDD) e as redes neurais, em geral, tendem a obter um melhor desempenho preditivo do que outros métodos (como árvores de decisão e modelos baseados em regras), quando submetidos a sequências biológicas.

Apesar de ambas as bases de dados utilizadas estarem relacionadas ao mesmo problema

e serem relativas ao mesmo subconjunto de organismos (primatas/humanos), a metodologia original para construir cada uma foi, possivelmente, diferente. Este fato reflete-se claramente na diferença do desempenho dos classificadores para as duas bases de dados. O banco de dados do UCI acabou por ser mais fácil de ser classificado do que o HS3D para ambos os sinais, *donors* e *acceptors*.

Foi avaliada a utilidade dos dois métodos para um importante sub-problema da predição de genes e, com base neste experimento, foram obtidos limiares ótimos de classificação. Estes limiares são importantes porque podem ser adotados em um sistema automático de detecção de genes.

Adicionalmente pode-se citar que os modelos baseados em matrizes de pesos possuem um paralelismo intrínseco, que os torna extremamente interessantes se implementados em *hardware* reconfigurável, que é o objetivo maior do trabalho.

5.1.2 Sistema Proposto

Como resultado da avaliação das matrizes de pesos para construção de sensores de sinais, foi proposta uma arquitetura em *hardware* reconfigurável, explorando o paralelismo em múltiplos níveis.

O resultado mais importante deste trabalho está relacionado ao tempo de processamento dos sensores, especialmente para o modelo WWAM. Usando a versão em *hardware*, a janela de 43 nucleotídeos do sensor de *acceptor* é processada em aproximadamente 28 ns. Usando exatamente o mesmo modelo, mas na implementação baseada em *software*, a média de tempo de processamento em um PC foi de cerca de 5160 ns, cerca de 180 vezes mais lenta que a abordagem em *hardware*.

Para o sensor de *start codon*, o *speed-up* foi cerca de 10 vezes, e para o sensor de *donors*, cerca de 8 vezes. Esta comparação, no entanto, deve ser interpretada com cautela e não pode ser generalizada porque tratam-se de arquiteturas radicalmente diferentes.

A avaliação do mecanismo de varredura, com todos os sensores em paralelo, apresentou *speed-up* em torno de 280 vezes, para 1 MBytes de dados.

Uma desvantagem da atual implementação é a velocidade de comunicação limitada. Para a análise de grandes quantidades de dados, como o genoma completo de um determinado organismo, esta limitação de velocidade poderia representar um gargalo importante. Esta questão pode ser resolvida adotando-se um kit de FPGA com interface de comunicação mais rápida, como por exemplo, PCI Express.

A alternativa proposta teve por objetivo reduzir o custo computacional dos tradicionais sensores de sinal baseados em *software*, com uma melhora dramática na velocidade de processamento. Os resultados obtidos até agora sugerem a aplicabilidade de um sistema de detecção de genes inteiramente implementado em *hardware* reconfigurável.

5.2 CONCLUSÕES

A proposta de uma arquitetura para análise de sequências biológicas, que possui múltiplos níveis de paralelismo e de fácil expansão, é a maior contribuição do presente trabalho.

A descrição em *hardware* dos sensores baseados em matrizes de pesos é uma contribuição relevante pois propõe um modelo simples e, ao mesmo tempo, eficiente sem, no entanto, perder a sua qualidade preditiva.

A arquitetura do mecanismo de varredura, por sua vez, explora a capacidade de *pipeline* do sistema permitindo que um elevado *speed-up* seja alcançado, mesmo em relação aos atuais PCs, que possuem elevados *clocks*.

Também é importante ressaltar que este trabalho não tem a pretensão de esgotar o assunto da predição de genes ou se apresentar como uma ferramenta completa e concluída para a análise de DNA. Este trabalho propõe uma alternativa ao processo de varredura de forma paralelizada, entretanto, pode ser adaptado a uma ferramenta tradicional, baseada em *software*, de forma que atenda às expectativas dos biológos e demais pesquisadores da área.

5.3 TRABALHOS FUTUROS

O presente trabalho apresentou uma estratégia bastante promissora para a redução do tempo de análise de sequências biológicas. Além disso, abriu inúmeras possibilidades para a sua expansão em trabalhos futuros, dentre as quais pode-se citar:

- Utilização de um kit com tecnologia de FPGA mais moderna. O objetivo deste seria executar o mesmo sistema a *clocks* mais elevados que o atual 50MHz.
- Utilização de um kit de FPGA com interface de comunicação mais rápida (como, por exemplo, PCI Express). A finalidade deste é reduzir o tempo de transferência dos dados, aumentando o desempenho total do sistema.
- Integração com alguma ferramenta de predição de genes de código aberto, como o próprio N-Scan ou outro, de forma que este sistema se apresente como um acelerador para o

processo de varredura. Isso só terá validade se o item anterior for atendido.

- Implementação de um barramento de sensores com uma janela maior e com posicionamento múltiplo de sensores do mesmo tipo deslocados na janela. Nesta abordagem, o controlador poderia inserir mais de um nucleotídeo na janela por ciclo de clock, permitindo alcançar *speed-ups* ainda mais altos. A única limitação para esta abordagem é o número de células lógicas do dispositivo FPGA.

REFERÊNCIAS

ALTERA. **Avalon Interface Specifications**. San Jose, USA: Altera Corporation, 2009.

ALTERA. **NIOS II Processor Reference Handbook**. San Jose, USA: Altera Corporation, 2009.

ANASTASSIOU, D. Frequency-domain analysis of biomolecular sequences. **Bioinformatics**, v. 16, p. 1073–1081, 2000.

ARMSTRONG JUNIOR, N.; LOPES, H.; LIMA, C. Preliminary steps towards protein folding prediction using reconfigurable computing. In: **IEEE International Conference on Reconfigurable Computing and FPGA's (ReConFig 2006)**. San Luis Potosi, Mexico: IEEE Computer Society, 2006. p. 92–98.

ARMSTRONG JUNIOR, N.; LOPES, H.; LIMA, C. Reconfigurable computing for accelerating protein folding simulations. **Lecture Notes in Computer Science**, v. 4419, p. 314–325, 2007.

ARTHUR, S. S. et al. Microbial gene identification using interpolated markov models. **Nucleic Acids Research**, v. 26, p. 544–548, 1998.

BAKOS, J. D. FPGA acceleration of gene rearrangement analysis. In: **International Symposium on Field-Programmable Custom Computing Machines**. Napa, USA: IEEE Computer Society, 2007. p. 85–94.

BALDI, P. et al. Assessing the accuracy of prediction algorithms for classication: an overview. **Bioinformatics**, v. 16, p. 412–424, 2000.

BAXEVANIS, A. D.; OUELLETTE, B. F. **Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins**. New York, USA: Wiley-Interscience, 1998.

BECKSTETTE, M. et al. Significant speedup of database searches with HMMs by search space reduction with PSSM family models. **Bioinformatics**, v. 25, n. 24, p. 3251–3258, 2009.

BENSON, D. A. et al. GenBank. **Nucleic Acids Research**, v. 36, p. 25–30, 2008.

BLAKE, C. L.; MERZ, C. J. **UCI Repository of Machine Learning Databases**. University 1998. Disponível em: <http://www.ics.uci.edu/ mlearn/MLRepository.html>.

BUCHER, P. Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. **Journal of Molecular Biology**, v. 212, p. 563–578, 1990.

BURGE, C. **Identification of genes in human genomic DNA**. Tese (Doutorado) — Stanford University, 1997.

BURGE, C.; KARLIN, S. Prediction of complete gene structures in human genomic DNA. **Journal of Molecular Biology**, v. 268, n. 1, p. 78–94, 1997.

BURGE, C.; KARLIN, S. Finding the genes in genomic DNA. **Current Opinion in Structural Biology**, v. 8, n. 3, p. 346–354, 1998.

BURSET, M.; GUIGO, R. Evaluation of gene structure prediction programs. **Genomics**, v. 34, n. 3, p. 353–367, 1996.

CAI, D. et al. Modeling splice sites with Bayes networks. **Bioinformatics**, v. 16, p. 152–158, 2000.

CHRYSOS, G. et al. A FPGA based coprocessor for gene finding using Interpolated Markov Model (IMM). In: **International Conference on Field Programmable Logic and Applications (FPL 2009)**. Prague, Czech Republic: IEEE Computer Society, 2009. p. 683–686.

COMPTON, K.; HAUCK, S. Reconfigurable computing: a survey of systems and software. **ACM Computing Surveys**, v. 34, p. 171–210, 2002.

CRAVEN, M.; SHAVLIK, J. Machine learning approaches to gene recognition. **IEEE Expert**, v. 9, n. 2, p. 2–10, 1994.

DANDASS, Y. et al. Accelerating string set matching in FPGA hardware for bioinformatics research. **BMC Bioinformatics**, v. 9, p. 197, 2008.

DELCHER, A. L. et al. Improved microbial gene identification with GLIMMER. **Nucleic Acids Research**, v. 27, n. 23, p. 4636–4641, 1999.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, v. 27, n. 8, p. 861–874, 2006.

FICKETT, J. Recognition of protein coding regions in DNA sequences. **Nucleic Acids Research**, v. 10, p. 5303–5318, 1982.

FICKETT, J. The gene identification problem: an overview for developers. **Computers in Chemistry**, v. 20, p. 103–118, 1996.

FICKETT, J.; TUNG, C. Assessment of protein coding measures. **Nucleic Acids Research**, v. 20, p. 6441–6450, 1992.

GELFAND, M.; ROYTBERG, M. Prediction of the intron-exon structure by a dynamic programming approach. **BioSystems**, v. 30, p. 173–182, 1993.

GELFAND, S.; MIRONOV, A.; PEVZNER, P. Gene recognition via spliced sequence alignment. **Proceedings of the National Academy of Sciences of USA**, v. 93, p. 9061–9066, 1996.

GERSHENZON, N.; STORMO, G.; IOSHIKHES, I. Computational technique for improvement of the position-weight matrices for the DNA/protein binding sites. **Nucleic Acids Research**, v. 33, n. 7, p. 2290–2301, 2005.

GISH, W.; STATES, D. J. Identification of protein coding regions by database similarity search. **Nature Genetics**, v. 3, p. 266–272, 1993.

GROSS, S. S.; BRENT, M. R. Using multiple alignments to improve gene prediction. **Journal of Computational Biology**, Springer, p. 379–393, 2005.

GUIGO, R. et al. Prediction of gene structure. **Journal of Molecular Biology**, v. 226, p. 141–157, 1992.

KANEHISA, M. **Post-Genome Informatics**. Oxford, UK: Oxford University Press, 2000.

KAUER, G.; BLOCKER, H. Applying signal theory to the analysis of biomolecules. **Bioinformatics**, v. 19, p. 2016–2021, 2003.

KORF, I. et al. Integrating genomic homology into gene structure prediction. **Bioinformatics**, v. 1, p. S1–S9, 2001.

KROGH, A. Two methods for improving performance of an HMM and their application for gene-finding. In: **Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology**. Halkidiki, Greece: AAAI Press, 1997. p. 179–186.

KULP, D. et al. A generalized Hidden Markov Model for the recognition of human genes in DNA. In: **Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology**. St. Louis, USA: AAAI Press, 1996. p. 134–142.

LIMA, C. et al. Multiple sequence alignment using reconfigurable computing. **Lecture Notes in Computer Science**, v. 4419, p. 379–384, 2007.

LOPES, H.; LIMA, C.; MURATA, N. A configware approach for high-speed parallel analysis of genomic data. **Journal of Circuits, Systems, and Computers**, v. 16, p. 527–540, 2007.

LUKASHIN, A.; BORODOVSKY, M. Genemark.hmm: the new solutions for gene-finding. **Nucleic Acids Research**, v. 26, p. 1107–1115, 1998.

MAJOROS, W. H.; PERTEA, M.; SALZBERG, S. L. TigrScan and GlimmerHMM: two open source ab initio eukaryotic gene-finders. **Bioinformatics**, Oxford University Press, Oxford, UK, v. 20, n. 16, p. 2878–2879, 2004.

MARONGIU, A.; PALAZZARI, P.; ROSATO, V. Designing hardware for protein sequence analysis. **Bioinformatics**, v. 19, n. 14, p. 1739–1740, 2003.

MARONGIU, A.; PALAZZARI, P.; ROSATO, V. PROSIDIS: A special purpose processor for protein similarity discovery. In: **International of Parallel and Distributed Processing Symposium**. Los Alamitos, USA: IEEE Computer Society, 2003. v. 0, p. 155b.

MARTINS, C. et al. Computação reconfigurável: conceitos, tendências e aplicações. In: **Anais do XXII Jornadas de Atualização em Informática**. [S.l.: s.n.], 2003. p. 339–388.

MATTHEWS, B. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. **Biochimica et Biophysica Acta (BBA) - Protein Structure**, v. 405, p. 442–451, 1975.

MORITZ, G.; LOPES, H.; LIMA, C. Hardalign: a parallel pairwise alignment hardware application. In: **Proceedings of IEEE International Conference on Field Programmable Technology**. Bangkok, Thailand: IEEE Computer Society, 2006. p. 369–372.

MOUNT, D. W. **Bioinformatics: Sequence and Genome Analysis**. New York, USA: Cold Spring Harbor Laboratory Press, 2001.

OLIVER, T. et al. Using reconfigurable hardware to accelerate multiple sequence alignment with ClustalW. **Bioinformatics**, v. 21, p. 3431–3432, 2005.

POLLASTRO, P.; RAMPONE, S. HS3D, a dataset of Homo Sapiens splice regions, and its extraction procedure from a major public database. **International Journal of Modern Physics**, v. 13, n. 8, p. 1105–1117, 2002.

RAMPONE, S. Splice-junction recognition on DNA sequences by BRAIN learning algorithm. **Bioinformatics**, v. 14, p. 676–684, 1998.

ROGIC, S.; MACKWORTH, A. K.; OUELLETTE, F. B. Evaluation of gene-finding programs on mammalian sequences. **Genome Research**, v. 11, n. 5, p. 817–832, 2001.

SALZBERG, S. et al. A Decision Tree system for finding genes in DNA. **Journal of Computational Biology**, v. 5, n. 4, p. 667–680, 1998.

SENAPATHY, P.; SHAPIRO, M.; HARRIS., N. Splice junctions, branch point sites, and exons: sequence statistics identification, and applications to genome project. **Methods in Enzymology**, v. 183, p. 252–278, 1990.

SING, T. et al. ROCR: visualizing classifier performance in R. **Bioinformatics**, v. 21, p. 3940–3941, 2005.

SOLOVYEV, V.; SALAMOV, A.; LAWRENCE, C. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. **Nucleic Acids Research**, v. 22, p. 5156–5163, 1994.

STADEN, R. Computer methods to locate signals in nucleic acid sequences. **Nucleic Acids Research**, v. 12, p. 505–519, 1984.

STADEN, R.; MCLACHLAN, A. Codon preference and its use in identifying protein coding regions in long DNA sequences. **Nucleic Acids Research**, v. 10, p. 141–156, 1982.

STEPANOVA, M.; LIN, F.; LIN, V. A hopfield neural classifier and its fpga implementation for identification of symmetrically structured dna motifs. **Journal of VLSI Signal Processing**, v. 48, p. 239–254, 2007.

STEPANOVA, M.; LIN, F.; LIN, V. Design and development of a FPGA-based cascade Markov model for recognition of steroid hormone response elements. In: **IEEE International Parallel and Distributed Processing Symposium (IPDPS 2008)**. Miami, USA: IEEE Computer Society, 2008.

TAVARES, L. G.; LOPES, H. S.; LIMA, C. R. E. Estudo comparativo de métodos de aprendizado de máquina na detecção de regiões promotoras de genes de Escherichia Coli. In: **Anais do I Simpósio de Brasileiro de Inteligência Computacional**. Florianópolis, SC: ., 2007.

TAVARES, L. G.; LOPES, H. S.; LIMA, C. R. E. A comparative study of machine learning methods for detecting promoters in bacterial DNA sequences. In: **ICIC '08: Proceedings of the 4th International Conference on Intelligent Computing**. Shanghai, China: Lecture Notes in Artifical Inteligence, 2008. p. 959–966.

TAVARES, L. G.; LOPES, H. S.; LIMA, C. R. E. Evaluation of weight matrix models in the splice junction recognition problem. In: **Proceedings of the IEEE International Conference on Bioinformatics & Biomedicine (BIBM)**. Washington, USA: IEEE Computer Society, 2009. v. 1, p. 14–19.

TIWARI, S. et al. Prediction of probable genes by Fourier analysis of genomic sequences. **Computer Applications in Biosciences**, v. 13, n. 3, p. 263–270, 1997.

TOWELL, G. G.; SHAVLIK, J. W. Refining symbolic knowledge using neural networks. In: **Proceedings of the International Workshop on Multistrategy Learning**. Harpers Ferry, USA: Morgan Kaufmann, 1991. p. 405–429.

TOWELL, G. G.; SHAVLIK, J. W.; NOORDEWIER, M. O. Refinement of approximate domain theories by knowledge-based neural networks. In: **Proceedings of the Eighth National Conference on Artificial Intelligence**. Boston, USA: AAAI Press, 1990. p. 861–866.

UBERBACHER, E. C.; MURAL, R. J. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. **Proceedings of the National Academy of Sciences of USA**, v. 88, p. 11261–11265, 1991.

WINGENDER, E. et al. Transfac: an integrated system for gene expression regulation. **Nucleic Acids Research**, v. 28, p. 3431–3432, 2000.

YAMAGUCHI, Y.; MARUYAMA, T.; KONAGAYA, A. High speed homology search with FPGAs. In: **Proceedings of the 7th Pacific Symposium on Biocomputing (PSB 2002)**. Lihue, Hawaii, USA: ,, 2002. p. 271–282.

YAN, M.; LIN, Z.-S.; ZHANG, C.-T. A new Fourier transform approach for protein coding measure based on the format of the Z curve. **Bioinformatics**, v. 14, p. 685–690, 1998.

ZHANG, M.; MARR, T. A weight array method for splicing signal analysis. **Computer Applications in Biosciences**, v. 9, n. 5, p. 499–509, 1993.

ZHANG, M. Q. Identification of protein coding regions in the human genome by quadratic discriminant analysis. **Proceedings of the National Academy of Sciences of U.S.A**, v. 94, p. 565–568, 1997.

ANEXO A – MATRIZES DE PESOS UTILIZADAS

WMM - SENSOR *START CODON*

A	-5	-6	-2	7	-2	-8	18	-100	-100	-3	-1	-9
C	3	7	10	-9	13	9	-100	-100	-100	-4	10	5
G	7	4	0	5	-4	2	-100	-100	23	9	-5	6
T	-7	-7	-12	-22	-10	-14	-100	18	-100	-11	-8	-5

WMM - SENSOR PROMOTOR *TATA-BOX*

A	-6	-26	18	-49	19	14	19	12	7	-8	-2	-2	-2	-5	-3
C	6	-10	-300	-32	-300	-300	-50	-56	-11	5	6	4	3	1	1
G	6	-24	-56	-55	-43	-100	-22	-11	7	6	4	4	4	5	5
T	-16	16	-15	19	-17	3	-40	3	-16	-10	-16	-9	-7	-4	-5

WMM - SENSOR *POLYA*

A	18	17	-20	17	18	18
C	-24	-31	-27	-47	-37	-31
G	-37	-37	-37	-47	-37	-46
T	-38	-11	17	-11	-38	-37

WMM - SENSOR *STOP CODON TAA*

A	-100	18	18	0	-1	0
C	-100	-100	-100	3	3	2
G	-100	-100	-100	1	2	2
T	-1	-100	-100	-4	-3	-3

WMM - SENSOR *STOP CODON TGA*

A	-100	-100	18	-1	-3	0
C	-100	-100	-100	2	4	3
G	-100	23	-100	1	4	2
T	7	-100	-100	-4	-6	-5

WMM - SENSOR *STOP CODON TAG*

A	-100	18	-100	0	-1	-4
C	-100	-100	-100	4	4	6
G	-100	-100	23	1	1	3
T	-3	-100	-100	-6	-4	-5

MDD - SENSOR *DONOR*

A) NNNGTNNAN, NNNGTNNCN e NNNGTNNTN

A	-15	18	-41	-100	-100	22	14	4	0
C	-13	-29	-49	-100	-100	-46	-3	-4	-5
G	-30	-30	19	23	-100	-15	-15	-100	2
T	-46	-20	-52	-100	18	-42	-10	-1	1

B) NNAGTNNGN, NNCGTNNGN e NNTGTNNGN

A	-18	11	5	-100	-100	22	18	-100	-17
C	-19	1	0	-100	-100	-60	-19	-100	-16
G	-29	-11	-100	23	-100	-2	-30	23	-25
T	-33	-8	-5	-100	18	-58	-29	-100	15

C) NCGGTNNGN, NGGGTNNGN e NTGGTNNGN

A	-15	-100	-100	-100	-100	22	19	-100	-5
C	-16	8	-100	-100	-100	-37	-18	-100	-4
G	-32	-8	23	23	-100	0	-21	23	-13
T	-21	5	-100	-100	18	-38	-19	-100	15

D) NAGGTNNGA, NAGGTNNGC e NAGGTNNGG

A	-14	18	-100	-100	-100	21	12	-100	1
C	-15	-100	-100	-100	-100	-28	-7	-100	5
G	-31	-100	23	23	-100	-1	-11	23	-4
T	-37	-100	-100	-100	18	-28	-12	-100	-100

E) NAGGTNNGT

A	-23	18	-100	-100	-100	14	13	-100	-100
C	-26	-100	-100	-100	-100	-14	-6	-100	-100
G	-34	-100	23	23	-100	0	-6	23	-100
T	-42	-100	-100	-100	18	-12	-10	-100	18

WWAM - SENSOR *ACCEPTOR*

0	-1	-3	4	0	0	-3	3	0	-1	-3	3	0	0	-4	3	0	1	-4	3	0	0	-4	3	-1	1	-5	5	-1	1	-5	5
1	0	-6	1	0	0	-7	1	0	0	-7	2	1	-1	-7	2	0	1	-7	2	-1	0	-9	3	0	-2	-7	3	0	0	-8	2
0	-1	-1	3	-1	2	-2	3	-1	2	-3	3	-1	0	-2	3	0	2	-3	3	-2	3	-3	3	-1	2	-4	4	-2	3	-5	5
1	-4	0	1	1	-3	-1	2	1	-3	-1	1	1	-3	-1	1	2	-1	-1	0	1	-2	-2	2	1	-3	-2	2	0	-2	-1	2
0	0	-2	4	-2	0	-1	4	0	0	-1	3	0	0	-2	4	-1	1	-2	3	0	0	-3	4	0	0	-3	4	0	0	-3	4
1	0	-9	2	1	0	-8	2	0	1	-10	3	0	1	-8	2	0	2	-11	1	0	1	-10	2	0	1	-10	3	0	1	-8	1
0	-2	-3	6	-3	-1	-1	6	-1	-1	-2	5	-2	-1	0	3	-2	-1	-2	6	-2	-1	-3	7	0	0	-3	5	-1	-1	-2	6
0	0	0	1	0	-1	0	0	0	-1	0	1	0	-1	0	1	1	0	-1	1	2	-1	0	1	1	1	-2	1	0	1	-1	0
1	1	-3	3	2	2	-5	4	2	2	-4	2	2	3	-5	3	0	1	-4	4	1	2	-4	4	2	4	-6	4	0	4	-5	5
-1	2	-14	3	0	2	-12	2	1	2	-12	2	0	2	-13	4	0	3	-12	2	0	2	-11	1	0	1	-13	3	0	2	-13	3
-2	2	-1	2	-1	2	-1	1	-2	1	0	2	-1	2	-2	2	0	2	-2	1	-3	3	-2	3	-1	4	-3	1	-2	3	-2	2
1	0	-2	2	1	0	-2	2	1	1	-2	2	1	0	-2	2	2	1	-3	1	1	1	-2	1	1	0	-2	2	0	1	-3	2
1	-2	-4	3	1	-2	-5	3	1	-1	-5	3	2	-2	-6	3	2	0	-6	2	3	-1	-8	3	3	0	-9	3	3	-1	-8	3
1	-2	-10	3	1	-2	-11	3	1	-1	-9	2	1	-2	-12	3	2	0	-10	1	1	-1	-12	2	0	-1	-13	3	-1	-1	-12	3
1	0	-3	3	-1	0	-2	3	-1	1	-2	2	1	-1	-3	3	1	0	-3	2	1	2	-4	2	1	1	-4	2	2	-1	-3	1
2	-1	-1	0	1	-1	-2	1	1	0	-1	0	1	0	-2	1	2	0	-3	0	1	1	-3	1	2	-1	-3	1	2	0	-4	1
0	-1	-3	4	0	0	-3	3	0	0	-3	3	0	0	-4	3	0	1	-4	3	0	0	-4	3	-1	1	-5	5	-1	1	-5	5
1	0	-6	1	0	0	-7	1	0	0	-7	2	1	-1	-7	2	0	1	-7	2	-1	0	-9	3	0	-2	-7	3	0	0	-8	2
0	-1	-1	3	-1	2	-2	3	-1	2	-3	3	-1	2	-3	4	0	2	-3	3	-2	3	-3	3	-1	2	-4	4	-2	2	-4	5
1	-4	0	1	1	-3	-1	2	1	-3	-1	1	2	-4	0	1	2	-1	-1	0	1	-2	-2	2	1	-3	-2	2	1	-1	-2	2
0	0	-2	4	-2	0	-1	4	0	0	-1	3	0	0	-3	3	-1	1	-2	3	0	0	-3	4	0	0	-3	4	-2	1	-3	5
1	0	-9	2	1	0	-8	2	0	1	-10	3	1	1	-8	1	0	2	-11	1	0	1	-10	2	0	1	-10	3	-1	1	-11	3
0	-2	-3	6	-3	-1	-1	6	-1	-1	-2	5	1	-1	-1	2	-2	-1	-2	6	-2	-1	-3	7	0	0	-3	5	-2	0	-3	6
0	0	0	1	0	-1	0	0	0	-1	0	1	0	0	0	0	1	0	-1	1	2	-1	0	1	1	1	-2	1	1	1	-2	1
1	1	-3	3	2	2	-5	4	2	2	-4	2	1	2	-4	3	0	1	-4	4	1	2	-4	4	2	4	-6	4	0	5	-7	4
-1	2	-14	3	0	2	-12	2	1	2	-12	2	1	0	-12	2	0	3	-12	2	0	2	-11	1	0	1	-13	3	0	1	-13	3
-2	2	-1	2	-1	2	-1	1	-2	1	0	2	0	1	-2	3	0	2	-2	1	-3	3	-2	3	-1	4	-3	1	-2	2	-2	3
1	0	-2	2	1	0	-2	2	1	1	-2	2	2	-1	-2	1	2	1	-3	1	1	1	-2	1	1	0	-2	2	2	0	-2	1
1	-2	-4	3	1	-2	-5	3	1	-1	-5	3	3	0	-6	2	2	0	-6	2	3	-1	-8	3	3	0	-9	3	3	-1	-10	3
1	-2	-10	3	1	-2	-11	3	1	-1	-9	2	1	-1	-9	2	2	0	-10	1	1	-1	-12	2	0	-1	-13	3	1	-2	-12	2
1	0	-3	3	-1	0	-2	3	-1	1	-2	2	2	0	-3	2	1	0	-3	2	1	2	-4	2	1	1	-4	2	4	0	-7	3
2	-1	-1	0	1	-1	-2	1	1	0	-1	0	2	-1	-1	0	2	0	-3	0	1	1	-3	1	2	-1	-3	1	1	0	-3	1
0	-1	-3	4	0	0	-3	3	0	0	-4	3	0	0	-4	3	0	1	-4	3	0	0	-4	3	-1	1	-5	5	-1	1	-5	5
1	0	-6	1	0	0	-7	1	1	-1	-7	2	0	-1	-7	2	0	1	-7	2	-1	0	-9	3	0	-2	-7	3	0	0	-8	2
0	-1	-1	3	-1	2	-2	3	-1	0	-2	3	-1	2	-3	4	0	2	-3	3	-2	3	-3	3	-2	3	-5	5	-2	2	-4	5
1	-4	0	1	1	-3	-1	2	1	-3	-1	1	2	-4	0	1	2	-1	-1	0	1	-2	-2	2	0	-2	-1	2	1	-1	-2	2
0	0	-2	4	-2	0	-1	4	0	0	-1	3	0	0	-3	3	-1	1	-2	3	0	0	-3	4	0	0	-3	4	-2	1	-3	5
1	0	-9	2	1	0	-8	2	0	1	-8	2	1	1	-8	1	0	2	-11	1	0	1	-10	3	0	1	-8	1	-1	1	-11	3
0	-2	-3	6	-3	-1	-1	6	-2	-1	0	3	1	-1	-1	2	-2	-1	-2	6	-2	-1	-3	7	-1	-1	-2	6	-2	0	-3	6
0	0	0	1	0	-1	0	0	0	-1	0	1	0	0	0	0	1	0	-1	1	2	-1	0	1	0	1	-1	0	1	1	-2	1
1	1	-3	3	2	2	-5	4	2	3	-5	3	1	2	-4	3	0	1	-4	4	1	2	-4	4	0	4	-5	5	0	5	-7	4
-1	2	-14	3	0	2	-12	2	0	2	-13	4	1	0	-12	2	0	3	-12	2	0	2	-11	1	0	2	-13	3	0	1	-13	3
-2	2	-1	2	-1	2	-1	1	-1	2	-2	2	0	1	-2	3	0	2	-2	1	-3	3	-2	3	-2	3	-2	2	-2	2	-2	3
1	0	-2	2	1	0	-2	2	1	0	-2	2	2	-1	-2	1	2	1	-3	1	1	1	-2	1	0	1	-3	2	2	0	-2	1
1	-2	-4	3	1	-2	-5	3	1	-1	-5	3	3	0	-6	2	2	0	-6	2	3	-1	-8	3	3	0	-9	3	3	-1	-10	3
1	-2	-10	3	1	-2	-11	3	1	-2	-12	3	1	-1	-9	2	2	0	-10	1	1	-1	-12	2	-1	-1	-12	3	1	-2	-12	2
1	0	-3	3	-1	0	-2	3	1	-1	-3	3	2	0	-3	2	1	0	-3	2	1	2	-4	2	2	-1	-4	3	4	0	-7	3
2	-1	-1	0	1	-1	-2	1	1	0	-2	1	2	-1	-1	0	2	0	-3	0	1	1	-3	1	2	0	-4	1	1	0	-3	1
0	-1	-3	4	0	-1	-3	3	0	0	-4	3	0	0	-4	3	0	1	-4	3	-1	1	-5	5	-1	1	-5	5	-1	1	-5	5
1	0	-6	1	0	0	-7	2	1	-1	-7	2	0	-1	-7	2	0	1	-7	2	0	-2	-7	3	0	0	-8	2	0	-1	-10	3
0	-1	-1	3	-1	2	-3	3	-1	0	-2	3	-1	2	-3	4	0	2	-3	3	-1	2	-4	4	-2	3	-5	5	-2	2	-4	5
1	-4	0	1	1	-3	-1	1	1	-3	-1	1	2	-4	0	1	2	-1	-1	0	1	-3	-2	2	0	-2	-1	2	1	-1	-2	2
0	0	-2	4	0	0	-1	3	0	0	-2	4	0	0	-3	3	-1	1	-2	3	0	0	-3	4	0	0	-3	4	-2	1	-3	5
1	0	-9	2	0	1	-10	3	0	1	-8	2	1	1	-8	1	0	2	-11	1	0	1	-10	3	0	1	-8	1	-1	1	-11	3
0	-2	-3	6	-1	-1	-2	5	-2	-1	0	3	1	-1	-1	2	-2	-1	-2	6	0	0	-3	5	-1	-1	-2	6	-2	0	-3	6
0	0	0	1	0	-1	0	1	0	-1	0	1	0	0	0	0	1	0	-1	1	1	1	-2	1	0	1	-1	0	1	1	-2	1
1	1	-3	3	2	2	-4	2	2	3	-5	3	1	2	-4	3	0	1	-4	4	2	4	-6	4	0	4	-5	5	0	5	-7	4
-1	2	-14	3	1	2	-12	2	0	2	-13	4	1	0	-12	2	0	3	-12	2	0	1	-13	3	0	2	-13	3	0	1	-13	3
-2	2	-1	2	-2	1	0	2	-1	2	-2	2	0	1	-2	3	0	2	-2	1	-1	4	-3	1	-2	3	-2	2	-2	2	-2	3
1	0	-2	2	1	1	-2	2	1	0	-2	2	2	-1	-2	1	2	1	-3	1	1	0	-2	2	0	1	-3	2	2	0	-2	1
1	-2	-4	3	1	-1	-5	3	2	-2	-6	3	3	0	-6	2	2	0	-6	2	3	0	-9	3	3	-1	-8	3	3	-1	-10	3
1	-2	-10	3	1	-1	-9	2	1	-2	-12	3	1	-1	-9	2	2	0	-10	1	0	-1	-13	3	-1	-1	-12	3	1	-2	-12	2
1	0	-3	3	-1	1	-2	2	1	-1	-3	3	2	0	-3	2	1	0	-3	2	1	1	-4	2	2	-1	-4	3	4	0	-7	3
2	-1	-1	0	1	0	-1	0	1	0	-2	1	2	-1	-1	0	2	0	-3	0	2	-1	-3	1	2	0	-4	1	1	0	-3	1
0	0	-3	3	0	-1	-3	3	0	0	-4	3	0	0	-4	3	0	0	-4	3	-1	1	-5	5	-1	1	-5	5	-1	1	-5	5
0	0	-7	1	0	0	-7	2	1	-1	-7	2	0	-1	-7	2	-1	0	-9	3	0	-2	-7	3	0	0	-8	2	0	-1	-10	3
-1	2	-2	3	-1	2	-3	3	-1	0	-2	3	-1	2	-3	4	-2	3	-3	3	-1	2	-4	4	-2	3	-5	5	-2	2	-4	5
1	-3	-1	2	1	-3	-1	1	1	-3	-1	1	2	-4	0	1	1	-2	-2	2	1	-3	-2	2	0	-2	-1	2	1	-1	-2	2
-2	0	-1	4	0	0	-1	3	0	0	-2	4	0	0	-3	3	0	0	-3	4	0	0	-3	4	0	0	-3	4	-2	1	-3	5
1	0	-8	2	0	1	-10	3	0	1	-8	2	1	1	-8	1	0	1	-10	2	0	1	-10	3	0	1	-8	1	-1	1	-11	3
-3	-1	-1	6	-1	-1	-2	5	-2	-1	0	3	1	-1	-1	2	-2	-1	-3	7	0	0	-3	5	-1	-1	-2	6	-2	0	-3	6
0	-1	0	0	0	-1	0	1	0	-1	0	1	0	0	0	0	2	-1	0	1	1	1	-2	1	0	1	-1	0	1	1	-2	1
2	2	-5	4	2	2	-4	2	2	3	-5	3	1	2	-4	3	1	2	-4	4	2	4	-6	4	0	4	-5	5	0	5	-7	4
0	2	-12	2	1	2	-12	2	0	2	-13	4	1	0	-12	2	0	2	-11	1	0	1	-13	3	0	2	-13	3	0	1	-13	3
-1	2	-1	1	-2	1	0	2	-1	2	-2	2	0	1	-2	3	-3	3	-2	3	-1	4	-3	1	-2	3	-2	2	-2	2	-2	3
1	0	-2	2	1	1	-2	2	1	0	-2	2	2	-1	-2	1	1	1	-2	1	1	0	-2	2	0	1	-3	2	2	0	-2	1
1	-2	-5	3	1	-1	-5	3	2	-2	-6	3	3	0	-6	2	3	-1	-8	3	3	0	-9	3	3	-1	-8	3	3	-1	-10	3
1	-2	-11	3	1	-1	-9	2	1	-2	-12	3	1	-1	-9	2	1	-1	-12	2	0	-1	-13	3	-1	-1	-12	3	1	-2	-12	2
-1	0	-2	3	-1	1	-2	2	1	-1	-3	3	2	0	-3	2	1	2	-4	2	1	1	-4	2	2	-1	-4	3	4	0	-7	3
1	-1	-2	1	1	0	-1	0	1	0	-2	1	2	-1	-1	0	1	1	-3	1	2	-1	-3	1	2	0	-4	1	1	0	-3	1

WWAM - SENSOR *ACCEPTOR* (cont.)

0	2	-6	4	-1	2	-7	5	-1	3	-8	5	-1	5	-9	5	-2	4	-10	7	-3	5	-10	6	-3	7	-9	4	-2	6	-11	6
-1	1	-7	2	-1	-1	-9	4	-1	0	-7	3	-3	1	-9	4	-3	1	-11	5	-2	1	-8	2	-3	2	-6	2	-4	4	-10	3
-2	3	-5	4	-1	4	-6	3	-2	5	-6	5	-3	4	-7	6	-4	4	-6	6	-4	4	-7	7	-4	6	-8	6	-4	6	-9	7
1	-3	-1	2	2	-5	-1	2	1	-1	-3	2	0	-3	-2	3	0	-1	-2	2	0	-2	-3	3	0	-3	-2	3	1	-4	-1	3
0	1	-3	4	0	2	-6	6	-2	3	-5	6	0	2	-6	5	1	4	-9	6	-3	4	-7	6	-1	4	-9	7	-4	6	-10	7
-1	2	-11	2	-1	1	-10	2	0	2	-10	1	-2	1	-11	4	-2	1	-12	4	-3	1	-13	5	-4	2	-13	4	-2	2	-12	2
-1	-2	-2	7	0	0	-5	7	-3	2	-6	7	-3	1	-5	7	0	1	-7	7	-3	0	-6	8	-2	3	-10	9	0	2	-8	8
1	1	-2	1	2	0	-1	0	2	0	-1	1	3	0	-2	1	0	2	-2	0	0	2	-3	1	-1	2	-3	1	-3	1	-1	3
1	5	-7	4	-1	6	-7	5	-1	6	-7	4	-1	7	-9	6	-1	9	-11	7	-4	10	-10	5	-2	9	-10	3	-5	10	-11	7
-1	2	-13	3	0	1	-14	4	-2	1	-14	4	-3	2	-15	4	-4	1	-17	6	-5	3	-17	5	-4	0	-13	5	-5	2	-16	5
-1	2	-3	4	-3	3	-4	5	-2	3	-5	6	-4	4	-5	6	-4	3	-6	7	-4	4	-6	7	-4	4	-7	7	-5	5	-6	7
1	0	-3	2	1	-1	-3	3	0	1	-3	3	1	0	-3	2	-1	1	-3	4	1	-1	-3	4	0	1	-3	3	-2	0	-2	3
5	-1	-11	2	5	-1	-12	2	5	-1	-13	2	6	-1	-15	2	6	-2	-15	2	5	-1	-16	3	7	-3	-17	2	6	-1	-18	3
0	-2	-14	3	0	-2	-13	3	2	-3	-14	3	1	-3	-15	3	2	-4	-18	3	2	-3	-18	3	2	-2	-16	2	1	-3	-18	3
5	-1	-6	2	5	0	-8	3	4	0	-8	3	3	-1	-8	4	6	-1	-11	3	6	-2	-11	4	6	0	-13	5	6	-1	-13	5
3	-1	-3	1	2	0	-4	1	2	0	-5	1	3	-1	-5	2	2	0	-6	2	1	-1	-5	3	0	0	-5	2	-1	0	-5	3
0	2	-6	4	-1	2	-7	5	-1	3	-8	5	-2	4	-9	5	-2	4	-10	7	-3	5	-10	6	-3	7	-9	4	-2	3	-10	7
-1	1	-7	2	-1	-1	-9	4	-1	0	-7	3	-3	2	-9	3	-3	1	-11	5	-2	1	-8	2	-3	2	-6	2	-4	2	-6	3
-2	3	-5	4	-1	4	-6	3	-2	5	-6	5	-3	5	-8	7	-4	4	-6	6	-4	4	-7	7	-4	6	-8	6	-4	5	-8	6
1	-3	-1	2	2	-5	-1	2	1	-1	-3	2	1	-2	-2	2	0	-1	-2	2	0	-2	-3	3	0	-3	-2	3	-2	-1	-2	3
0	1	-3	4	0	2	-6	6	-2	3	-5	6	-1	3	-7	6	1	4	-9	6	-3	4	-7	6	-1	4	-9	7	-2	5	-11	8
-1	2	-11	2	-1	1	-10	2	0	2	-10	1	-3	1	-12	5	-2	1	-12	4	-3	1	-13	5	-4	2	-13	4	-4	1	-14	5
-1	-2	-2	7	0	0	-5	7	-3	2	-6	7	-1	1	-6	8	0	1	-7	7	-3	0	-6	8	-2	3	-10	9	-2	3	-10	10
1	1	-2	1	2	0	-1	0	2	0	-1	1	1	2	-3	1	0	2	-2	0	0	2	-3	1	-1	2	-3	1	-3	2	-4	4
1	5	-7	4	-1	6	-7	5	-1	6	-7	4	-2	9	-9	5	-1	9	-11	7	-4	10	-10	5	-2	9	-10	3	-4	10	-11	6
-1	2	-13	3	0	1	-14	4	-2	1	-14	4	-5	3	-16	5	-4	1	-17	6	-5	3	-17	5	-4	0	-13	5	-7	3	-15	6
-1	2	-3	4	-3	3	-4	5	-2	3	-5	6	-5	5	-6	7	-4	3	-6	7	-4	4	-6	7	-4	4	-7	7	-5	5	-6	7
1	0	-3	2	1	-1	-3	3	0	1	-3	3	2	-1	-4	4	-1	1	-3	4	1	-1	-3	4	0	1	-3	3	-2	2	-4	5
5	-1	-11	2	5	-1	-12	2	5	-1	-13	2	6	-1	-15	1	6	-2	-15	2	5	-1	-16	3	7	-3	-17	2	4	-2	-18	5
0	-2	-14	3	0	-2	-13	3	2	-3	-14	3	0	-1	-19	4	2	-4	-18	3	2	-3	-18	3	2	-2	-16	2	0	-4	-17	6
5	-1	-6	2	5	0	-8	3	4	0	-8	3	6	-2	-8	2	6	-1	-11	3	6	-2	-11	4	6	0	-13	5	3	0	-12	6
3	-1	-3	1	2	0	-4	1	2	0	-5	1	2	0	-6	2	2	0	-6	2	1	-1	-5	3	0	0	-5	2	-2	-2	-6	4
0	2	-6	4	-1	2	-7	5	-1	5	-9	5	-2	4	-9	5	-2	4	-10	7	-3	5	-10	6	-2	6	-11	6	-2	3	-10	7
-1	1	-7	2	-1	-1	-9	4	-3	1	-9	4	-3	2	-9	3	-3	1	-11	5	-2	1	-8	2	-4	4	-10	3	-4	2	-6	3
-2	3	-5	4	-1	4	-6	3	-3	4	-7	6	-3	5	-8	7	-4	4	-6	6	-4	4	-7	7	-4	6	-9	7	-4	5	-8	6
1	-3	-1	2	2	-5	-1	2	0	-3	-2	3	1	-2	-2	2	0	-1	-2	2	0	-2	-3	3	1	-4	-1	3	-2	-1	-2	3
0	1	-3	4	0	2	-6	6	0	2	-6	5	-1	3	-7	6	1	4	-9	6	-3	4	-7	6	-4	6	-10	7	-2	5	-11	8
-1	2	-11	2	-1	1	-10	2	-2	1	-11	4	-3	1	-12	5	-2	1	-12	4	-3	1	-13	5	-2	2	-12	2	-4	1	-14	5
-1	-2	-2	7	0	0	-5	7	-3	1	-5	7	-1	1	-6	8	0	1	-7	7	-3	0	-6	8	0	2	-8	8	-2	3	-10	10
1	1	-2	1	2	0	-1	0	3	0	-2	1	1	2	-3	1	0	2	-2	0	0	2	-3	1	-3	1	-1	3	-3	2	-4	4
1	5	-7	4	-1	6	-7	5	-1	7	-9	6	-2	9	-9	5	-1	9	-11	7	-4	10	-10	5	-5	10	-11	7	-4	10	-11	6
-1	2	-13	3	0	1	-14	4	-3	2	-15	4	-5	3	-16	5	-4	1	-17	6	-5	3	-17	5	-5	2	-16	5	-7	3	-15	6
-1	2	-3	4	-3	3	-4	5	-4	4	-5	6	-5	5	-6	7	-4	3	-6	7	-4	4	-6	7	-5	5	-6	7	-5	5	-6	7
1	0	-3	2	1	-1	-3	3	1	0	-3	2	2	-1	-4	4	-1	1	-3	4	1	-1	-3	4	-2	0	-2	3	-2	2	-4	5
5	-1	-11	2	5	-1	-12	2	6	-1	-15	2	6	-1	-15	1	6	-2	-15	2	5	-1	-16	3	6	-1	-18	3	4	-2	-18	5
0	-2	-14	3	0	-2	-13	3	1	-3	-15	3	0	-1	-19	4	2	-4	-18	3	2	-3	-18	3	1	-3	-18	3	0	-4	-17	6
5	-1	-6	2	5	0	-8	3	3	-1	-8	4	6	-2	-8	2	6	-1	-11	3	6	-2	-11	4	6	-1	-13	5	3	0	-12	6
3	-1	-3	1	2	0	-4	1	3	-1	-5	2	2	0	-6	2	2	0	-6	2	1	-1	-5	3	-1	0	-5	3	-2	-2	-6	4
0	2	-6	4	-1	3	-8	5	-1	5	-9	5	-2	4	-9	5	-2	4	-10	7	-3	7	-9	4	-2	6	-11	6	-2	3	-10	7
-1	1	-7	2	-1	0	-7	3	-3	1	-9	4	-3	2	-9	3	-3	1	-11	5	-3	2	-6	2	-4	4	-10	3	-4	2	-6	3
-2	3	-5	4	-2	5	-6	5	-3	4	-7	6	-3	5	-8	7	-4	4	-6	6	-4	6	-8	6	-4	6	-9	7	-4	5	-8	6
1	-3	-1	2	1	-1	-3	2	0	-3	-2	3	1	-2	-2	2	0	-1	-2	2	0	-3	-2	3	1	-4	-1	3	-2	-1	-2	3
0	1	-3	4	-2	3	-5	6	0	2	-6	5	-1	3	-7	6	1	4	-9	6	-1	4	-9	7	-4	6	-10	7	-2	5	-11	8
-1	2	-11	2	0	2	-10	1	-2	1	-11	4	-3	1	-12	5	-2	1	-12	4	-4	2	-13	4	-2	2	-12	2	-4	1	-14	5
-1	-2	-2	7	-3	2	-6	7	-3	1	-5	7	-1	1	-6	8	0	1	-7	7	-2	3	-10	9	0	2	-8	8	-2	3	-10	10
1	1	-2	1	2	0	-1	1	3	0	-2	1	1	2	-3	1	0	2	-2	0	-1	2	-3	1	-3	1	-1	3	-3	2	-4	4
1	5	-7	4	-1	6	-7	4	-1	7	-9	6	-2	9	-9	5	-1	9	-11	7	-2	9	-10	3	-5	10	-11	7	-4	10	-11	6
-1	2	-13	3	-2	1	-14	4	-3	2	-15	4	-5	3	-16	5	-4	1	-17	6	-4	0	-13	5	-5	2	-16	5	-7	3	-15	6
-1	2	-3	4	-2	3	-5	6	-4	4	-5	6	-5	5	-6	7	-4	3	-6	7	-4	4	-7	7	-5	5	-6	7	-5	5	-6	7
1	0	-3	2	0	1	-3	3	1	0	-3	2	2	-1	-4	4	-1	1	-3	4	0	1	-3	3	-2	0	-2	3	-2	2	-4	5
5	-1	-11	2	5	-1	-13	2	6	-1	-15	2	6	-1	-15	1	6	-2	-15	2	7	-3	-17	2	6	-1	-18	3	4	-2	-18	5
0	-2	-14	3	2	-3	-14	3	1	-3	-15	3	0	-1	-19	4	2	-4	-18	3	2	-2	-16	2	1	-3	-18	3	0	-4	-17	6
5	-1	-6	2	4	0	-8	3	3	-1	-8	4	6	-2	-8	2	6	-1	-11	3	6	0	-13	5	6	-1	-13	5	3	0	-12	6
3	-1	-3	1	2	0	-5	1	3	-1	-5	2	2	0	-6	2	2	0	-6	2	0	0	-5	2	-1	0	-5	3	-2	-2	-6	4
-1	2	-7	5	-1	3	-8	5	-1	5	-9	5	-2	4	-9	5	-3	5	-10	6	-3	7	-9	4	-2	6	-11	6	-2	3	-10	7
-1	-1	-9	4	-1	0	-7	3	-3	1	-9	4	-3	2	-9	3	-2	1	-8	2	-3	2	-6	2	-4	4	-10	3	-4	2	-6	3
-1	4	-6	3	-2	5	-6	5	-3	4	-7	6	-3	5	-8	7	-4	4	-7	7	-4	6	-8	6	-4	6	-9	7	-4	5	-8	6
2	-5	-1	2	1	-1	-3	2	0	-3	-2	3	1	-2	-2	2	0	-2	-3	3	0	-3	-2	3	1	-4	-1	3	-2	-1	-2	3
0	2	-6	6	-2	3	-5	6	0	2	-6	5	-1	3	-7	6	-3	4	-7	6	-1	4	-9	7	-4	6	-10	7	-2	5	-11	8
-1	1	-10	2	0	2	-10	1	-2	1	-11	4	-3	1	-12	5	-3	1	-13	5	-4	2	-13	4	-2	2	-12	2	-4	1	-14	5
0	0	-5	7	-3	2	-6	7	-3	1	-5	7	-1	1	-6	8	-3	0	-6	8	-2	3	-10	9	0	2	-8	8	-2	3	-10	10
2	0	-1	0	2	0	-1	1	3	0	-2	1	1	2	-3	1	0	2	-3	1	-1	2	-3	1	-3	1	-1	3	-3	2	-4	4
-1	6	-7	5	-1	6	-7	4	-1	7	-9	6	-2	9	-9	5	-4	10	-10	5	-2	9	-10	3	-5	10	-11	7	-4	10	-11	6
0	1	-14	4	-2	1	-14	4	-3	2	-15	4	-5	3	-16	5	-5	3	-17	5	-4	0	-13	5	-5	2	-16	5	-7	3	-15	6
-3	3	-4	5	-2	3	-5	6	-4	4	-5	6	-5	5	-6	7	-4	4	-6	7	-4	4	-7	7	-5	5	-6	7	-5	5	-6	7
1	-1	-3	3	0	1	-3	3	1	0	-3	2	2	-1	-4	4	1	-1	-3	4	0	1	-3	3	-2	0	-2	3	-2	2	-4	5
5	-1	-12	2	5	-1	-13	2	6	-1	-15	2	6	-1	-15	1	5	-1	-16	3	7	-3	-17	2	6	-1	-18	3	4	-2	-18	5
0	-2	-13	3	2	-3	-14	3	1	-3	-15	3	0	-1	-19	4	2	-3	-18	3	2	-2	-16	2	1	-3	-18	3	0	-4	-17	6
5	0	-8	3	4	0	-8	3	3	-1	-8	4	6	-2	-8	2	6	-2	-11	4	6	0	-13	5	6	-1	-13	5	3	0	-12	6
2	0	-4	1	2	0	-5	1	3	-1	-5	2	2	0	-6	2	1	-1	-5	3	0	0	-5	2	-1	0	-5	3	-2	-2	-6	4

WWAM - SENSOR *ACCEPTOR* (cont.)

-3	7	-11	6	-3	6	-12	7	-3	6	-13	8	-2	6	-14	6	-3	6	-23	10	-2	3	-27	10	-3	5	-36	11	-3	4	-41	11
-3	1	-5	2	-4	3	-5	1	-5	4	-5	2	-6	4	-6	2	-9	3	-9	5	-9	4	-10	5	-10	2	-9	5	-12	2	-12	7
-5	8	-10	7	-6	7	-9	7	-5	6	-9	7	-4	4	-9	7	-8	8	-8	6	-8	7	-10	8	-11	7	-10	10	-17	6	-10	11
-1	-2	-2	3	-3	1	-2	3	-3	0	-2	3	-5	0	-2	4	-7	1	-4	6	-8	0	-6	6	-9	-1	-6	7	-11	1	-10	8
-2	3	-10	8	-4	6	-11	8	-4	6	-13	9	-5	6	-15	9	-4	4	-20	12	-5	5	-25	12	-6	6	-33	14	-6	7	-42	14
-4	0	-12	5	-5	1	-11	5	-6	2	-14	5	-7	3	-13	4	-9	2	-12	5	-11	2	-11	6	-11	4	-11	5	-12	3	-14	6
-3	1	-9	10	-5	3	-9	9	-6	4	-8	8	-8	4	-9	10	-11	3	-8	10	-9	4	-10	10	-11	2	-9	11	-13	4	-12	11
-7	2	-3	3	-9	1	-3	5	-8	1	-3	4	-11	3	-5	5	-13	3	-5	6	-13	4	-6	5	-14	3	-6	5	-15	2	-7	7
-4	11	-13	8	-3	10	-13	7	-6	11	-14	8	-4	10	-13	8	-5	9	-18	12	-4	9	-21	11	-5	8	-27	13	-4	8	-32	14
-6	3	-16	5	-6	3	-14	4	-6	5	-16	3	-8	2	-14	6	-10	4	-18	6	-11	3	-15	6	-12	3	-16	7	-15	2	-19	8
-5	5	-7	8	-6	4	-6	8	-8	5	-6	8	-8	4	-6	8	-11	6	-7	10	-14	7	-7	10	-13	5	-7	12	-15	4	-8	12
-2	1	-4	5	-7	0	-2	6	-5	2	-4	5	-8	1	-3	5	-10	3	-4	6	-13	3	-4	6	-12	3	-6	8	-14	2	-7	9
3	0	-18	5	0	0	-20	8	-2	1	-21	10	-3	1	-23	9	-5	2	-31	11	-4	0	-42	11	-5	2	-49	13	-5	2	-52	11
-3	-3	-15	6	-5	-1	-17	5	-7	-1	-17	7	-9	-1	-18	6	-12	-1	-20	8	-16	1	-19	8	-16	0	-20	8	-17	0	-22	9
4	0	-12	5	1	0	-12	7	-2	0	-11	8	-5	1	-12	10	-10	2	-11	10	-11	3	-13	10	-11	2	-13	11	-15	2	-14	11
-4	0	-5	4	-5	0	-6	4	-7	1	-6	4	-6	2	-7	4	-8	1	-8	5	-10	2	-7	4	-11	3	-10	5	-12	1	-8	5
-3	7	-11	6	-3	6	-12	7	-3	6	-13	8	-3	5	-17	9	-3	6	-23	10	-2	3	-27	10	-3	5	-36	11	-5	4	-46	13
-3	1	-5	2	-4	3	-5	1	-5	4	-5	2	-8	3	-7	4	-9	3	-9	5	-9	4	-10	5	-10	2	-9	5	-15	2	-17	8
-5	8	-10	7	-6	7	-9	7	-5	6	-9	7	-8	8	-9	8	-8	8	-8	6	-8	7	-10	8	-11	7	-10	10	-21	5	-10	12
-1	-2	-2	3	-3	1	-2	3	-3	0	-2	3	-5	-3	-3	6	-7	1	-4	6	-8	0	-6	6	-9	-1	-6	7	-15	1	-11	9
-2	3	-10	8	-4	6	-11	8	-4	6	-13	9	-4	5	-17	11	-4	4	-20	12	-5	5	-25	12	-6	6	-33	14	-7	4	-50	15
-4	0	-12	5	-5	1	-11	5	-6	2	-14	5	-9	4	-12	4	-9	2	-12	5	-11	2	-11	6	-11	4	-11	5	-14	3	-16	7
-3	1	-9	10	-5	3	-9	9	-6	4	-8	8	-10	3	-8	10	-11	3	-8	10	-9	4	-10	10	-11	2	-9	11	-17	3	-15	14
-7	2	-3	3	-9	1	-3	5	-8	1	-3	4	-10	3	-6	6	-13	3	-5	6	-13	4	-6	5	-14	3	-6	5	-17	3	-10	9
-4	11	-13	8	-3	10	-13	7	-6	11	-14	8	-5	9	-15	10	-5	9	-18	12	-4	9	-21	11	-5	8	-27	13	-7	7	-41	17
-6	3	-16	5	-6	3	-14	4	-6	5	-16	3	-10	4	-17	7	-10	4	-18	6	-11	3	-15	6	-12	3	-16	7	-16	1	-21	8
-5	5	-7	8	-6	4	-6	8	-8	5	-6	8	-8	5	-6	9	-11	6	-7	10	-14	7	-7	10	-13	5	-7	12	-14	2	-9	14
-2	1	-4	5	-7	0	-2	6	-5	2	-4	5	-8	2	-5	6	-10	3	-4	6	-13	3	-4	6	-12	3	-6	8	-17	2	-8	10
3	0	-18	5	0	0	-20	8	-2	1	-21	10	-4	2	-29	10	-5	2	-31	11	-4	0	-42	11	-5	2	-49	13	-7	0	-69	13
-3	-3	-15	6	-5	-1	-17	5	-7	-1	-17	7	-13	1	-15	7	-12	-1	-20	8	-16	1	-19	8	-16	0	-20	8	-19	-1	-20	9
4	0	-12	5	1	0	-12	7	-2	0	-11	8	-9	3	-11	10	-10	2	-11	10	-11	3	-13	10	-11	2	-13	11	-14	2	-15	13
-4	0	-5	4	-5	0	-6	4	-7	1	-6	4	-8	2	-6	4	-8	1	-8	5	-10	2	-7	4	-11	3	-10	5	-14	2	-10	6
-3	7	-11	6	-3	6	-12	7	-2	6	-14	6	-3	5	-17	9	-3	6	-23	10	-2	3	-27	10	-3	4	-41	11	-5	4	-46	13
-3	1	-5	2	-4	3	-5	1	-6	4	-6	2	-8	3	-7	4	-9	3	-9	5	-9	4	-10	5	-12	2	-14	6	-15	2	-17	8
-5	8	-10	7	-6	7	-9	7	-4	4	-9	7	-8	8	-9	8	-8	8	-8	6	-8	7	-10	8	-17	6	-10	11	-21	5	-10	12
-1	-2	-2	3	-3	1	-2	3	-5	0	-2	4	-5	-3	-3	6	-7	1	-4	6	-8	0	-6	6	-11	1	-10	8	-15	1	-11	9
-2	3	-10	8	-4	6	-11	8	-5	6	-15	9	-4	5	-17	11	-4	4	-20	12	-5	5	-25	12	-6	7	-42	14	-7	4	-50	15
-4	0	-12	5	-5	1	-11	5	-7	3	-13	4	-9	4	-12	4	-9	2	-12	5	-11	2	-11	6	-12	3	-14	6	-14	3	-16	7
-3	1	-9	10	-5	3	-9	9	-6	4	-8	8	-10	3	-8	10	-11	3	-8	10	-9	4	-10	10	-11	2	-9	11	-17	3	-15	14
-7	2	-3	3	-9	1	-3	5	-11	3	-5	5	-10	3	-6	6	-13	3	-5	6	-13	4	-6	5	-15	2	-7	7	-17	3	-10	9
-4	11	-13	8	-3	10	-13	7	-4	10	-13	8	-5	9	-15	10	-5	9	-18	12	-4	9	-21	11	-4	8	-32	14	-7	7	-41	17
-6	3	-16	5	-6	3	-14	4	-8	2	-14	6	-10	4	-17	7	-10	4	-18	6	-11	3	-15	6	-15	2	-19	8	-16	1	-21	8
-5	5	-7	8	-6	4	-6	8	-8	4	-6	8	-8	5	-6	9	-11	6	-7	10	-14	7	-7	10	-15	4	-8	12	-14	2	-9	14
-2	1	-4	5	-7	0	-2	6	-8	1	-3	5	-8	2	-5	6	-10	3	-4	6	-13	3	-4	6	-14	2	-7	9	-17	2	-8	10
3	0	-18	5	0	0	-20	8	-3	1	-23	9	-4	2	-29	10	-5	2	-31	11	-4	0	-42	11	-5	2	-52	11	-7	0	-69	13
-3	-3	-15	6	-5	-1	-17	5	-9	-1	-18	6	-13	1	-15	7	-12	-1	-20	8	-16	1	-19	8	-17	0	-22	9	-19	-1	-20	9
4	0	-12	5	1	0	-12	7	-5	1	-12	10	-9	3	-11	10	-10	2	-11	10	-11	3	-13	10	-15	2	-14	11	-14	2	-15	13
-4	0	-5	4	-5	0	-6	4	-6	2	-7	4	-8	2	-6	4	-8	1	-8	5	-10	2	-7	4	-12	1	-8	5	-14	2	-10	6
-3	7	-11	6	-3	6	-13	8	-2	6	-14	6	-3	5	-17	9	-3	6	-23	10	-3	5	-36	11	-3	4	-41	11	-5	4	-46	13
-3	1	-5	2	-5	4	-5	2	-6	4	-6	2	-8	3	-7	4	-9	3	-9	5	-10	2	-9	5	-12	2	-12	7	-15	2	-17	8
-5	8	-10	7	-5	6	-9	7	-4	4	-9	7	-8	8	-9	8	-8	8	-8	6	-11	7	-10	10	-17	6	-10	11	-21	5	-10	12
-1	-2	-2	3	-3	0	-2	3	-5	0	-2	4	-5	-3	-3	6	-7	1	-4	6	-9	-1	-6	7	-11	1	-10	8	-15	1	-11	9
-2	3	-10	8	-4	6	-13	9	-5	6	-15	9	-4	5	-17	11	-4	4	-20	12	-6	6	-33	14	-6	7	-42	14	-7	4	-50	15
-4	0	-12	5	-6	2	-14	5	-7	3	-13	4	-9	4	-12	4	-9	2	-12	5	-11	4	-11	5	-12	3	-14	6	-14	3	-16	7
-3	1	-9	10	-6	4	-8	8	-8	4	-9	10	-10	3	-8	10	-11	3	-8	10	-11	2	-9	11	-13	4	-12	11	-17	3	-15	14
-7	2	-3	3	-8	1	-3	4	-11	3	-5	5	-10	3	-6	6	-13	3	-5	6	-14	3	-6	5	-15	2	-7	7	-17	3	-10	9
-4	11	-13	8	-6	11	-14	8	-4	10	-13	8	-5	9	-15	10	-5	9	-18	12	-5	8	-27	13	-4	8	-32	14	-7	7	-41	17
-6	3	-16	5	-6	5	-16	3	-8	2	-14	6	-10	4	-17	7	-10	4	-18	6	-12	3	-16	7	-15	2	-19	8	-16	1	-21	8
-5	5	-7	8	-8	5	-6	8	-8	4	-6	8	-8	5	-6	9	-11	6	-7	10	-13	5	-7	12	-15	4	-8	12	-14	2	-9	14
-2	1	-4	5	-5	2	-4	5	-8	1	-3	5	-8	2	-5	6	-10	3	-4	6	-12	3	-6	8	-14	2	-7	9	-17	2	-8	10
3	0	-18	5	-2	1	-21	10	-3	1	-23	9	-4	2	-29	10	-5	2	-31	11	-5	2	-49	13	-5	2	-52	11	-7	0	-69	13
-3	-3	-15	6	-7	-1	-17	7	-9	-1	-18	6	-13	1	-15	7	-12	-1	-20	8	-16	0	-20	8	-17	0	-22	9	-19	-1	-20	9
4	0	-12	5	-2	0	-11	8	-5	1	-12	10	-9	3	-11	10	-10	2	-11	10	-11	2	-13	11	-15	2	-14	11	-14	2	-15	13
-4	0	-5	4	-7	1	-6	4	-6	2	-7	4	-8	2	-6	4	-8	1	-8	5	-11	3	-10	5	-12	1	-8	5	-14	2	-10	6
-3	6	-12	7	-3	6	-13	8	-2	6	-14	6	-3	5	-17	9	-2	3	-27	10	-3	5	-36	11	-3	4	-41	11	-5	4	-46	13
-4	3	-5	1	-5	4	-5	2	-6	4	-6	2	-8	3	-7	4	-9	4	-10	5	-10	2	-9	5	-12	2	-12	7	-15	2	-17	8
-6	7	-9	7	-5	6	-9	7	-4	4	-9	7	-8	8	-9	8	-8	7	-10	8	-11	7	-10	10	-17	6	-10	11	-21	5	-10	12
-3	1	-2	3	-3	0	-2	3	-5	0	-2	4	-5	-3	-3	6	-8	0	-6	6	-9	-1	-6	7	-11	1	-10	8	-15	1	-11	9
-4	6	-11	8	-4	6	-13	9	-5	6	-15	9	-4	5	-17	11	-5	5	-25	12	-6	6	-33	14	-6	7	-42	14	-7	4	-50	15
-5	1	-11	5	-6	2	-14	5	-7	3	-13	4	-9	4	-12	4	-11	2	-11	6	-11	4	-11	5	-12	3	-14	6	-14	3	-16	7
-5	3	-9	9	-6	4	-8	8	-8	4	-9	10	-10	3	-8	10	-9	4	-10	10	-11	2	-9	11	-13	4	-12	11	-17	3	-15	14
-9	1	-3	5	-8	1	-3	4	-11	3	-5	5	-10	3	-6	6	-13	4	-6	5	-14	3	-6	5	-15	2	-7	7	-17	3	-10	9
-3	10	-13	7	-6	11	-14	8	-4	10	-13	8	-5	9	-15	10	-4	9	-21	11	-5	8	-27	13	-4	8	-32	14	-7	7	-41	17
-6	3	-14	4	-6	5	-16	3	-8	2	-14	6	-10	4	-17	7	-11	3	-15	6	-12	3	-16	7	-15	2	-19	8	-16	1	-21	8
-6	4	-6	8	-8	5	-6	8	-8	4	-6	8	-8	5	-6	9	-14	7	-7	10	-13	5	-7	12	-15	4	-8	12	-14	2	-9	14
-7	0	-2	6	-5	2	-4	5	-8	1	-3	5	-8	2	-5	6	-13	3	-4	6	-12	3	-6	8	-14	2	-7	9	-17	2	-8	10
0	0	-20	8	-2	1	-21	10	-3	1	-23	9	-4	2	-29	10	-4	0	-42	11	-5	2	-49	13	-5	2	-52	11	-7	0	-69	13
-5	-1	-17	5	-7	-1	-17	7	-9	-1	-18	6	-13	1	-15	7	-16	1	-19	8	-16	0	-20	8	-17	0	-22	9	-19	-1	-20	9
1	0	-12	7	-2	0	-11	8	-5	1	-12	10	-9	3	-11	10	-11	3	-13	10	-11	2	-13	11	-15	2	-14	11	-14	2	-15	13
-5	0	-6	4	-7	1	-6	4	-6	2	-7	4	-8	2	-6	4	-10	2	-7	4	-11	3	-10	5	-12	1	-8	5	-14	2	-10	6

WWAM - SENSOR *ACCEPTOR* (cont.)

-4	3	-61	13	-3	5	-58	12	-3	5	-62	12	-2	6	-60	11	-1	2	-66	12	7	3	-22	-2	-20	16	-100	8	18	-100	-100	-100
-13	5	-15	5	-15	6	-12	6	-12	8	-12	3	-10	6	-17	3	-14	4	-14	4	1	3	-4	-3	-24	7	-90	4	18	-100	-100	-100
-16	4	-13	15	-11	1	-7	12	-12	10	-9	8	-12	8	-9	11	-8	5	-12	10	5	3	-2	-16	-21	18	-51	-3	18	-100	-100	-100
-12	2	-11	8	-10	4	-8	6	-9	4	-10	7	-10	3	-11	7	-19	-10	-20	12	-1	1	-2	2	-19	9	-64	10	18	-100	-100	-100
-7	8	-54	14	-5	7	-66	13	-6	10	-69	13	-5	9	-66	12	-2	7	-62	13	9	10	-26	1	-21	16	-96	5	18	-100	-100	-100
-12	3	-13	5	-14	5	-16	6	-9	5	-13	3	-9	4	-18	4	-13	4	-21	4	3	3	-1	-6	-24	5	-66	8	18	-100	-100	-100
-16	6	-14	13	-15	5	-13	12	-12	6	-9	10	-15	7	-11	9	-7	4	-10	10	2	4	-1	-7	-31	14	-72	6	18	-100	-100	-100
-16	5	-11	9	-16	7	-9	7	-13	7	-8	4	-12	6	-11	6	-17	-2	-16	12	-2	2	1	-2	-18	8	-75	11	18	-100	-100	-100
-6	6	-45	16	-5	9	-46	14	-4	9	-44	15	-3	10	-43	15	-1	4	-22	14	5	2	-7	1	-19	16	-85	8	18	-100	-100	-100
-16	4	-22	7	-17	5	-19	6	-11	6	-21	4	-14	7	-25	5	-17	5	-29	6	1	3	-7	-3	-27	7	-73	4	18	-100	-100	-100
-12	2	-8	13	-13	5	-9	13	-15	8	-7	12	-12	8	-9	10	-7	3	-8	10	0	4	-2	-2	-25	16	-83	5	18	-100	-100	-100
-15	5	-8	7	-11	6	-7	5	-16	10	-7	6	-11	5	-8	5	-21	-5	-21	13	-6	3	-2	4	-14	8	-51	13	18	-100	-100	-100
-5	-1	-77	12	-5	2	-82	12	-5	1	-105	12	-5	6	-86	11	-1	3	-69	10	8	5	-37	0	-14	11	-107	8	18	-100	-100	-100
-17	0	-20	8	-17	0	-20	8	-15	1	-20	7	-15	4	-22	6	-16	4	-25	5	1	2	-6	-2	-26	5	-65	8	18	-100	-100	-100
-16	4	-16	13	-14	4	-16	11	-14	4	-15	12	-14	6	-14	11	-6	2	-13	10	2	4	-1	-4	-23	14	-72	6	18	-100	-100	-100
-12	3	-9	5	-10	4	-7	4	-9	6	-9	3	-10	5	-9	3	-14	-1	-15	8	-4	1	-2	2	-17	9	-59	7	18	-100	-100	-100
-4	3	-61	13	-3	5	-58	12	-3	5	-62	12	-5	4	-59	12	-1	2	-66	12	7	3	-22	-2	-20	16	-100	8	-100	-100	23	-100
-13	5	-15	5	-15	6	-12	6	-12	8	-12	3	-16	8	-19	3	-14	4	-14	4	1	3	-4	-3	-24	7	-90	4	-100	-100	-100	-100
-16	4	-13	15	-11	1	-7	12	-12	10	-9	8	-10	5	-5	9	-8	5	-12	10	5	3	-2	-16	-21	18	-51	-3	-100	-100	-100	-100
-12	2	-11	8	-10	4	-8	6	-9	4	-10	7	-16	3	-16	9	-19	-10	-20	12	-1	1	-2	2	-19	9	-64	10	-100	-100	-100	-100
-7	8	-54	14	-5	7	-66	13	-6	10	-69	13	-7	5	-60	15	-2	7	-62	13	9	10	-26	1	-21	16	-96	5	-100	-100	23	-100
-12	3	-13	5	-14	5	-16	6	-9	5	-13	3	-14	7	-22	3	-13	4	-21	4	3	3	-1	-6	-24	5	-66	8	-100	-100	-100	-100
-16	6	-14	13	-15	5	-13	12	-12	6	-9	10	-14	3	-13	13	-7	4	-10	10	2	4	-1	-7	-31	14	-72	6	-100	-100	-100	-100
-16	5	-11	9	-16	7	-9	7	-13	7	-8	4	-17	6	-15	7	-17	-2	-16	12	-2	2	1	-2	-18	8	-75	11	-100	-100	-100	-100
-6	6	-45	16	-5	9	-46	14	-4	9	-44	15	-7	6	-27	15	-1	4	-22	14	5	2	-7	1	-19	16	-85	8	-100	-100	23	-100
-16	4	-22	7	-17	5	-19	6	-11	6	-21	4	-19	10	-30	2	-17	5	-29	6	1	3	-7	-3	-27	7	-73	4	-100	-100	-100	-100
-12	2	-8	13	-13	5	-9	13	-15	8	-7	12	-14	5	-10	11	-7	3	-8	10	0	4	-2	-2	-25	16	-83	5	-100	-100	-100	-100
-15	5	-8	7	-11	6	-7	5	-16	10	-7	6	-18	7	-15	9	-21	-5	-21	13	-6	3	-2	4	-14	8	-51	13	-100	-100	-100	-100
-5	-1	-77	12	-5	2	-82	12	-5	1	-105	12	-9	1	-72	15	-1	3	-69	10	8	5	-37	0	-14	11	-107	8	-100	-100	23	-100
-17	0	-20	8	-17	0	-20	8	-15	1	-20	7	-19	3	-30	7	-16	4	-25	5	1	2	-6	-2	-26	5	-65	8	-100	-100	-100	-100
-16	4	-16	13	-14	4	-16	11	-14	4	-15	12	-15	2	-17	14	-6	2	-13	10	2	4	-1	-4	-23	14	-72	6	-100	-100	-100	-100
-12	3	-9	5	-10	4	-7	4	-9	6	-9	3	-14	5	-16	6	-14	-1	-15	8	-4	1	-2	2	-17	9	-59	7	-100	-100	-100	-100
-4	3	-61	13	-3	5	-58	12	-2	6	-60	11	-5	4	-59	12	-1	2	-66	12	7	3	-22	-2	18	-100	-100	-100	-100	-100	23	-100
-13	5	-15	5	-15	6	-12	6	-10	6	-17	3	-16	8	-19	3	-14	4	-14	4	1	3	-4	-3	18	-100	-100	-100	-100	-100	-100	-100
-16	4	-13	15	-11	1	-7	12	-12	8	-9	11	-10	5	-5	9	-8	5	-12	10	5	3	-2	-16	18	-100	-100	-100	-100	-100	-100	-100
-12	2	-11	8	-10	4	-8	6	-10	3	-11	7	-16	3	-16	9	-19	-10	-20	12	-1	1	-2	2	18	-100	-100	-100	-100	-100	-100	-100
-7	8	-54	14	-5	7	-66	13	-5	9	-66	12	-7	5	-60	15	-2	7	-62	13	9	10	-26	1	18	-100	-100	-100	-100	-100	23	-100
-12	3	-13	5	-14	5	-16	6	-9	4	-18	4	-14	7	-22	3	-13	4	-21	4	3	3	-1	-6	18	-100	-100	-100	-100	-100	-100	-100
-16	6	-14	13	-15	5	-13	12	-15	7	-11	9	-14	3	-13	13	-7	4	-10	10	2	4	-1	-7	18	-100	-100	-100	-100	-100	-100	-100
-16	5	-11	9	-16	7	-9	7	-12	6	-11	6	-17	6	-15	7	-17	-2	-16	12	-2	2	1	-2	18	-100	-100	-100	-100	-100	-100	-100
-6	6	-45	16	-5	9	-46	14	-3	10	-43	15	-7	6	-27	15	-1	4	-22	14	5	2	-7	1	18	-100	-100	-100	-100	-100	23	-100
-16	4	-22	7	-17	5	-19	6	-14	7	-25	5	-19	10	-30	2	-17	5	-29	6	1	3	-7	-3	18	-100	-100	-100	-100	-100	-100	-100
-12	2	-8	13	-13	5	-9	13	-12	8	-9	10	-14	5	-10	11	-7	3	-8	10	0	4	-2	-2	18	-100	-100	-100	-100	-100	-100	-100
-15	5	-8	7	-11	6	-7	5	-11	5	-8	5	-18	7	-15	9	-21	-5	-21	13	-6	3	-2	4	18	-100	-100	-100	-100	-100	-100	-100
-5	-1	-77	12	-5	2	-82	12	-5	6	-86	11	-9	1	-72	15	-1	3	-69	10	8	5	-37	0	18	-100	-100	-100	-100	-100	23	-100
-17	0	-20	8	-17	0	-20	8	-15	4	-22	6	-19	3	-30	7	-16	4	-25	5	1	2	-6	-2	18	-100	-100	-100	-100	-100	-100	-100
-16	4	-16	13	-14	4	-16	11	-14	6	-14	11	-15	2	-17	14	-6	2	-13	10	2	4	-1	-4	18	-100	-100	-100	-100	-100	-100	-100
-12	3	-9	5	-10	4	-7	4	-10	5	-10	4	-14	5	-16	6	-14	-1	-15	8	-4	1	-2	2	18	-100	-100	-100	-100	-100	-100	-100
-4	3	-61	13	-3	5	-62	12	-2	6	-60	11	-5	4	-59	12	-1	2	-66	12	-20	16	-100	8	18	-100	-100	-100	-100	-100	23	-100
-13	5	-15	5	-12	8	-12	3	-10	6	-17	3	-16	8	-19	3	-14	4	-14	4	-24	7	-90	4	18	-100	-100	-100	-100	-100	-100	-100
-16	4	-13	15	-12	10	-9	8	-12	8	-9	11	-10	5	-5	9	-8	5	-12	10	-21	18	-51	-3	18	-100	-100	-100	-100	-100	-100	-100
-12	2	-11	8	-9	4	-10	7	-10	3	-11	7	-16	3	-16	9	-19	-10	-20	12	-19	9	-64	10	18	-100	-100	-100	-100	-100	-100	-100
-7	8	-54	14	-6	10	-69	13	-5	9	-66	12	-7	5	-60	15	-2	7	-62	13	-21	16	-96	5	18	-100	-100	-100	-100	-100	23	-100
-12	3	-13	5	-9	5	-13	3	-9	4	-18	4	-14	7	-22	3	-13	4	-21	4	-24	5	-66	8	18	-100	-100	-100	-100	-100	-100	-100
-16	6	-14	13	-12	6	-9	10	-15	7	-11	9	-14	3	-13	13	-7	4	-10	10	-31	14	-72	6	18	-100	-100	-100	-100	-100	-100	-100
-16	5	-11	9	-13	7	-8	4	-12	6	-11	6	-17	6	-15	7	-17	-2	-16	12	-18	8	-75	11	18	-100	-100	-100	-100	-100	-100	-100
-6	6	-45	16	-4	9	-44	15	-3	10	-43	15	-7	6	-27	15	-1	4	-22	14	-19	16	-85	8	18	-100	-100	-100	-100	-100	23	-100
-16	4	-22	7	-11	6	-21	4	-14	7	-25	5	-19	10	-30	2	-17	5	-29	6	-27	7	-73	4	18	-100	-100	-100	-100	-100	-100	-100
-12	2	-8	13	-15	8	-7	12	-12	8	-9	10	-14	5	-10	11	-7	3	-8	10	-25	16	-83	5	18	-100	-100	-100	-100	-100	-100	-100
-15	5	-8	7	-16	10	-7	6	-11	5	-8	5	-18	7	-15	9	-21	-5	-21	13	-14	8	-51	13	18	-100	-100	-100	-100	-100	-100	-100
-5	-1	-77	12	-5	1	-105	12	-5	6	-86	11	-9	1	-72	15	-1	3	-69	10	-14	11	-107	8	18	-100	-100	-100	-100	-100	23	-100
-17	0	-20	8	-15	1	-20	7	-15	4	-22	6	-19	3	-30	7	-16	4	-25	5	-26	5	-65	8	18	-100	-100	-100	-100	-100	-100	-100
-16	4	-16	13	-14	4	-15	12	-14	6	-14	11	-15	2	-17	14	-6	2	-13	10	-23	14	-72	6	18	-100	-100	-100	-100	-100	-100	-100
-12	3	-9	5	-9	6	-9	3	-10	5	-10	4	-14	5	-16	6	-14	-1	-15	8	-17	9	-59	7	18	-100	-100	-100	-100	-100	-100	-100
-3	5	-58	12	-3	5	-62	12	-2	6	-60	11	-5	4	-59	12	7	3	-22	-2	-20	16	-100	8	18	-100	-100	-100	-100	-100	23	-100
-15	6	-12	6	-12	8	-12	3	-10	6	-17	3	-16	8	-19	3	1	3	-4	-3	-24	7	-90	4	18	-100	-100	-100	-100	-100	-100	-100
-11	1	-7	12	-12	10	-9	8	-12	8	-9	11	-10	5	-5	9	5	3	-2	-16	-21	18	-51	-3	18	-100	-100	-100	-100	-100	-100	-100
-10	4	-8	6	-9	4	-10	7	-10	3	-11	7	-16	3	-16	9	-1	1	-2	2	-19	9	-64	10	18	-100	-100	-100	-100	-100	-100	-100
-5	7	-66	13	-6	10	-69	13	-5	9	-66	12	-7	5	-60	15	9	10	-26	1	-21	16	-96	5	18	-100	-100	-100	-100	-100	23	-100
-14	5	-16	6	-9	5	-13	3	-9	4	-18	4	-14	7	-22	3	3	3	-1	-6	-24	5	-66	8	18	-100	-100	-100	-100	-100	-100	-100
-15	5	-13	12	-12	6	-9	10	-15	7	-11	9	-14	3	-13	13	2	4	-1	-7	-31	14	-72	6	18	-100	-100	-100	-100	-100	-100	-100
-16	7	-9	7	-13	7	-8	4	-12	6	-11	6	-17	6	-15	7	-2	2	1	-2	-18	8	-75	11	18	-100	-100	-100	-100	-100	-100	-100
-5	9	-46	14	-4	9	-44	15	-3	10	-43	15	-7	6	-27	15	5	2	-7	1	-19	16	-85	8	18	-100	-100	-100	-100	-100	23	-100
-17	5	-19	6	-11	6	-21	4	-14	7	-25	5	-19	10	-30	2	1	3	-7	-3	-27	7	-73	4	18	-100	-100	-100	-100	-100	-100	-100
-13	5	-9	13	-15	8	-7	12	-12	8	-9	10	-14	5	-10	11	0	4	-2	-2	-25	16	-83	5	18	-100	-100	-100	-100	-100	-100	-100
-11	6	-7	5	-16	10	-7	6	-11	5	-8	5	-18	7	-15	9	-6	3	-2	4	-14	8	-51	13	18	-100	-100	-100	-100	-100	-100	-100
-5	2	-82	12	-5	1	-105	12	-5	6	-86	11	-9	1	-72	15	8	5	-37	0	-14	11	-107	8	18	-100	-100	-100	-100	-100	23	-100
-17	0	-20	8	-15	1	-20	7	-15	4	-22	6	-19	3	-30	7	1	2	-6	-2	-26	5	-65	8	18	-100	-100	-100	-100	-100	-100	-100
-14	4	-16	11	-14	4	-15	12	-14	6	-14	11	-15	2	-17	14	2	4	-1	-4	-23	14	-72	6	18	-100	-100	-100	-100	-100	-100	-100
-10	4	-7	4	-9	6	-9	3	-10	5	-10	4	-14	5	-16	6	-4	1	-2	2	-17	9	-59	7	18	-100	-100	-100	-100	-100	-100	-100

WWAM - SENSOR *ACCEPTOR* (cont.)

-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-2	-1	-7	9	-100	-100	-100	-100	-2	-1	-7	9	0	5	-7	3	1	2	-5	4
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-1	-3	-6	5	-100	-100	-100	-100	-1	-3	-6	5	1	0	-2	0	1	1	-8	1
-1	-6	7	-9	-100	-100	-100	-100	-1	-6	7	-9	0	-4	-7	9	-100	-100	-100	-100	0	-4	-7	9	-1	4	-5	3	1	2	-7	4
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-6	-3	2	3	-100	-100	-100	-100	-6	-3	2	3	-2	3	-1	0	2	-2	0	0
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	4	1	-10	4	-6	11	-23	8
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	2	0	-4	0	0	1	-1	-1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	1	-3	1	1	-1	1	-1	1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-3	2	0	0	-2	3	0	-1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-2	-1	-7	9	0	5	-7	3	1	2	-5	4	0	5	-7	3
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-1	-3	-6	5	1	0	-2	0	1	1	-8	1	1	0	-2	0
-100	-100	-100	-100	-1	-6	7	-9	-100	-100	-100	-100	-100	-100	-100	-100	0	-4	-7	9	-1	4	-5	3	1	2	-7	4	-1	4	-5	3
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-6	-3	2	3	-2	3	-1	0	2	-2	0	0	-2	3	-1	0
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	4	1	-10	4	-6	11	-23	8	4	1	-10	4
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	2	0	-4	0	0	1	-1	-1	2	0	-4	0
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	1	-3	1	1	-1	1	-1	1	1	-3	1	1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-3	2	0	0	-2	3	0	-1	-3	2	0	0
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-2	-1	-7	9	-100	-100	-100	-100	1	2	-5	4	0	5	-7	3	1	2	-5	4
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-1	-3	-6	5	-100	-100	-100	-100	1	1	-8	1	1	0	-2	0	1	1	-8	1
-1	-6	7	-9	-100	-100	-100	-100	-100	-100	-100	-100	0	-4	-7	9	-100	-100	-100	-100	1	2	-7	4	-1	4	-5	3	1	2	-7	4
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-6	-3	2	3	-100	-100	-100	-100	2	-2	0	0	-2	3	-1	0	2	-2	0	0
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-6	11	-23	8	4	1	-10	4	-6	11	-23	8
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	0	1	-1	-1	2	0	-4	0	0	1	-1	-1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-1	1	-1	1	1	-3	1	1	-1	1	-1	1
-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-100	-2	3	0	-1	-3	2	0	0	-2	3	0	-1