Proceedings of the IV Brazilian Conference on Neural Networks - IV Congresso Brasileiro de Redes Neurais
pp. 390-395, July 20-22, 1999 - ITA, São José dos Campos - SP - Brazil

Implementação de Classificadores Neuronais na Tecnologia de Processadores Digitais de Sinais

J. B. O. Souza Filho, J. M. Seixas
COPPE/EE/UFRJ, CP 68504, Rio de Janeiro 21945 – 970 Brasil
E-mails : nash@lps.ufrj.br, seixas@lps.ufrj.br

Abstract

This article describes the implementation of an artificial neural network using high-speed digital signal processors (DSPs). The target processor was the ADSP-2106x, a 32 bit floating-point DSP, running Assembly language codes. As a case study, a particle discriminator for the field of high-energy physics is developed for online operation. Accuracy and speed tests confirm the good matching of this technology to the requirements of network implementation

1. Introdução

As redes neurais necessitam basicamente do cálculo de produtos internos na sua fase de produção de saídas. Como conseqüência, as redes neurais podem ser facilmente implementáveis em diferentes ambientes computacionais e são estruturas velozes o suficiente para realizar o reconhecimento de padrões em tempo real. [1]

Os processadores de sinais digitais (DSPs) apresentam uma arquitetura que otimiza as operações envolvidas no produto interno, ou seja, as multiplicações e acumulações. Esta otimização se faz através de estruturas de endereçamento flexível e unidades lógico-aritméticas de alto desempenho. Os DSPs vêm portanto, se constituindo numa opção economicamente viável para problemas que exijam resposta *online* ou em tempo real de alto desempenho.[2]

Como vantagens adicionais, poderíamos apontar a possibilidade do desenvolvimento da aplicação em linguagem C ou *Assembly*, a crescente disponibilidade de ferramentas eficientes, a existência de processadores de ponto-flutuante e a disponibilidade de processadores otimizados para aplicações específicas.

A flexibilidade quanto a linguagem de programação reflete as prioridades da implementação. Caso a portabilidade, a facilidade de leitura e adaptação do código sejam importantes, opta-se pela programação em C. Caso o compromisso seja a velocidade, opta-se pela linguagem *Assembly*. Para ambas linguagens estão disponíveis diversas ferramentas, entre elas os simuladores que permitem o acompanhamento passo-a-passo da execução do programa e a avaliação do conteúdo de posições de memória e dos registradores, o que favorece bastante o processo de depuração, teste, avaliação e otimização da aplicação. Quanto à portabilidade, os DSPs de ponto flutuante facilitam a transposição de uma aplicação executada em ambientes de alto-nível para esta tecnologia, facilitando o mapeamento de aplicações *offline* em aplicações *online*.

Este artigo trata da implementação em DSP de uma rede neuronal na sua fase de produção das saídas. Primeiramente, realizamos uma descrição da tecnologia DSP e do processador utilizado. Segue-se a descrição de sua aplicação em física de altas energias, as etapas de de-senvolvimento e os resultados obtidos. Por fim, algumas conclusões são apresentadas.

2. Processadores Digitais de Sinais (DSPs)

Os DSPs são microprocessadores que otimizam as operações de multiplicação e acumulação (produto interno), bastante freqüentes no processamento de sinais digitais. Normalmente, estas operações são realizadas em um único ciclo de execução, utilizando unidades aritméticas de alta performance e de alta precisão.

Com o objetivo de otimizar o produto interno, executando-o, de preferência, em um único ciclo, os DSPs apresentam internamente um barramento duplo: o barramento de dados e o barramento de programa e apresentam várias unidades internas que operam em paralelo.

Nos DSPs, o acesso a posições de memória consecutivas é otimizado. Geralmente, em uma mesma instrução, um valor da memória é coletado e o ponteiro para esta região é atualizado para um próximo acesso ou coleta. Grande é o número de registradores de endereçamento e diversas formas de endereçamento estão disponíveis.

Na maior parte das aplicações em DSP, o processador deve lidar com várias fontes de dados do mundo real, recebendo e transmitindo dados *online*, sem interromper suas operações matemáticas internas. Estas fontes de dados são diversas, entre elas: os sinais a processar, sinais de sistemas de controle e sinais de outros DSPs do mesmo tipo. Deste modo, os DSPs são normalmente integrados com processadores de entrada e saída, controladores de acesso direto a memória (DMA), portas seriais e de ligação (para multiprocessamento) assim como memória.

Os DSPs de melhor desempenho são normalmente construídos com uma razoável quantidade de memória

interna, de forma que toda a aplicação possa estar contida em seu interior. Normalmente, o barramento de dados e de programa são multiplexados, dando origem a um único barramento externo, o que não permite um acesso simultâneo a ambos e portanto não permite que o produto interno relativo a componentes armazenadas em memória externa seja realizado com a mesma performance que o de componentes residentes em memória interna.

O conjunto de instruções de um DSP é normalmente constituído por funções comuns aos processadores de uso geral, tais como chamadas de subrotinas e saltos (condicionais e incondicionais), manipulação de bits (teste, modificação e deslocamento), operações lógicas e aritméticas simples (soma, subtração e produto), e, em alguns casos, de instruções otimizadas para a implementação de filtros e de outras estruturas utilizadas no processamento digital de sinais. As funções mais complexas como as funções trigonométricas, exponenciais e logarítmicas não estão disponíveis e são normalmente constituídas por séries numéricas ou por tabelas (*Look-Up-Tables*).

2.1. ADSP-2106x

O ADSP-2106x é um DSP de 32 bits ponto-flutuante com freqüência máxima de clock de 40 Mhz. Sua arquitetura interna é conhecida como *Super Harvard Architecture Computer (SHARC)* e se caracteriza por duplo barramento (dados e programa) e memória de instruções (cache), de forma que todas instruções possam ser executadas em apenas um ciclo de máquina, inclusive a tomada de dois operandos da memória, a multiplicação e sua acumulação, que são realizadas em paralelo.

Esta arquitetura une ainda ao processador estruturas que favorecem o interfaceamento com o mundo real, entre elas: um processador de entrada e saída e um sistema de interfaceamento para multiprocessamento, conforme Figura 1.

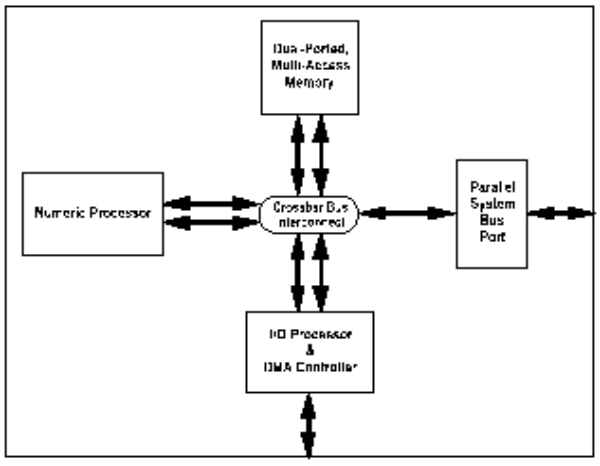

Figura 1: Arquitetura SHARC

No interior do processador numérico há um duplo gerador de endereços (DAG), 16 registradores de 40 bits (uso geral), um multiplicador, uma unidade lógica e aritmética (ALU) e um deslocador de *bits*, assim como um temporizador, uma memória tipo *cache* e um seqüenciador de programa (veja a Figura 2).

Cada DAG atua sobre um tipo de memória (dados ou programa), possuindo oito grupos de registradores. Cada grupo é constituído por outros quatro registradores, entre eles: base, indexador, modificador e comprimento. Através destes são possíveis diversas formas de endereçamento, entre elas o acesso pré-modificado e pós-modificado, assim como a constituição de *buffers* circulares em *hardware*.

Sua ALU é capaz de operar com números de 32 *bits* em ponto fixo e números de 32 ou 40 *bits* em ponto flutuante padrão IEEE, o que facilita a transposição de aplicações de alto nível. Para multiplicações em ponto-fixo existe um acumulador de 80 *bits*.

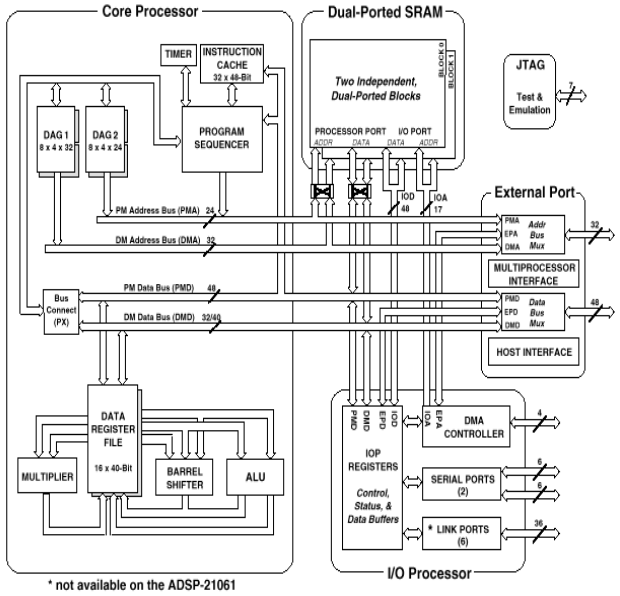

Figura 2: Estrutura do ADSP-2106x

O seqüenciador de programa permite que todas as instruções sejam condicionais, assim como todas as instruções no interior dos laços sejam executadas sem desperdício de ciclos de máquina. O processador ainda apresenta o recurso de salto atrasado para que não haja desperdício de ciclos de máquina em saltos (*jumps*) e em chamadas de subrotinas (*calls*) pela destruição do *pipeline*. O salto atrasado consiste em se colocar duas instruções após a instrução de salto ou chamada, instruções que são executadas antes do início da subrotina ou da porção de código para onde foi dirigido o salto.

Todas as instruções do ADSP-2106x são executadas em um único ciclo de máquina. Cada instrução é processada em três ciclos: um de busca da instrução, outro de decodificação e o último de execução. Através

da memória cache é possível que, enquanto a instrução atual é decodificada, por exemplo, uma nova instrução seja buscada na memória e a instrução anterior seja executada. Desta forma, temos um *pipeline* de instruções o que resulta na realização de uma instrução por ciclo, com latência de duas instruções.

Para o SHARC, o paralelismo da execução das instruções é definido a priori pelo programador. Cabe portanto ao mesmo definir quais as operações que podem ser realizadas em paralelo e onde pode ser utilizado o recurso de salto intervalado. Desta forma, cada linha do programa leva um ciclo de *clock* para ser executada e é possível avaliar, antes mesmo da transformação do programa fonte em executável, qual será o tempo demandado na aplicação em questão.

Quanto à programação, o ADSP-2106x possui um *Assembly* com sintaxe lógica, o que facilita bastante a elaboração e leitura de programas. Também é possível a programação em linguagem C. Através do simulador é possível monitorar, instrução por instrução, a execução do programa em ambas as linguagens, verificando o impacto de cada uma sobre as unidades internas do processador, o que permite uma debugagem eficiente da aplicação.

Para o desenvolvimento estão disponíveis placas que podem ser conectadas a *slots* do PC ou às portas seriais. Estas placas normalmente são constituídas por um ou mais processadores, conversores digital-analógicos (DACs) e analógico-digitais (ADC) e são acompanhadas de softwares para a execução de programas, coleta e inserção de dados no DSP, assim como rotinas em linguagem C para o desenvolvimento de aplicações próprias de interfaceamento entre o PC e a placa de desenvolvimento. Através destas placas é possível a execução, teste e depuração da aplicação *online*, assim como a própria prototipagem da mesma.

4. A Aplicação

A Física de Partículas busca, através do estudo do choque de partículas consideravelmente aceleradas e, portanto, com altíssima energia, desvendar os mistérios da constituição da matéria. Neste estudo duas estruturas são indispensáveis: os aceleradores de partículas e os detetores.

Os aceleradores de partículas são responsáveis pelo aumento da energia das partículas e se utilizam de campos elétricos e magnéticos. Os campos elétricos são utilizados para aceleração das partículas e os magnéticos para manutenção da rota das mesmas (direcionamento).

Como a energia para a produção de fenômenos interessantes é bastante elevada, normalmente os aceleradores possuem forma circular de maneira que os feixes de partículas possam ser continuamente acelerados.

Associados aos aceleradores, existem os detetores de partículas, os quais são bastante diversificados, de acordo com a informação que se deseja extrair num experimento com colisão de partículas. Normalmente, os detetores possuem estruturas cilíndricas ou esféricas visando cobrir todas as direções em torno de um ponto de colisão.

O CERN (Laboratório Europeu para Física de Partículas) que é sediado na Suíça, vem desenvolvendo, em âmbito internacional, o projeto LHC (*Large Hadron Collider*). Este experimento, que entrará em funcionamento em 2005, será o maior acelerador de partículas do mundo, colidindo prótons com energia de 14 Tev. O LHC deverá mergulhar mais fundo do que qualquer experimento precedente nos estudos da constituição da matéria, reconstituindo as condições que existiram um microsegundo após a grande explosão que teria dado origem ao Universo.

Paralelamente à construção do LHC, existe o desenvolvimento de diversos detetores e sistemas de processamento de dados e sinais, entre eles o ATLAS que possui como um de seus subdetetores o Calorímetro de Telhas Cintilantes.

4.1. O Calorímetro de Telhas Cintilantes

Os calorímetros vêm possuindo um papel de crescente importância nos experimentos com colisionadores. Estes detetores medem a energia das partículas incidentes por meio de sua absorção total. O perfil de deposição da energia no detetor depende do tipo de particula que com ele interage. Com isto, os calorímetros são usados em sistemas *online* de validação de eventos, aproveitando da sua alta eficiência de discriminação e de sua velocidade de processamento.

O Calorímetro de Telhas é constituído de ferro (para absorção de energia) e de telhas de material cintilante (para amostragem da energia) [3]. Quando a partícula perde sua energia na interação com o detetor, ela excita o material cintilante das telhas de forma que luz seja produzida. Para conduzir a informação luminosa produzida nas telhas, existem fibras ópticas colocadas em cada lado livre das telhas, que conduzem estes sinais de luz a células foto-multiplicadoras, responsáveis pela con-versão destes sinais em sinais elétricos (veja Figura 3).

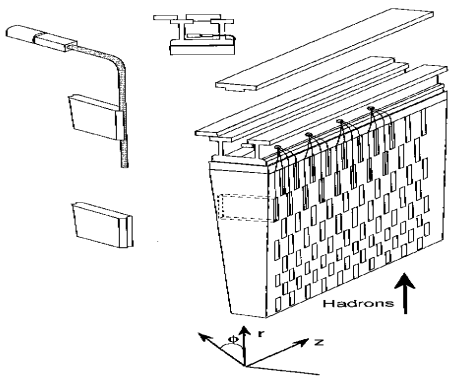

Figura 3: Visão de um módulo do Calorímetro de Telhas

A Figura 4 mostra o protótipo a ser utilizado neste trabalho. Ele é constituído por um módulo central envolvido por outros cinco módulos menores, totalizando 246 células. Estas 246 células (canais de leitura) estão divididas em 46 células no módulo central, conhecido como módulo zero, e 200 células provenientes dos demais módulos.[4]

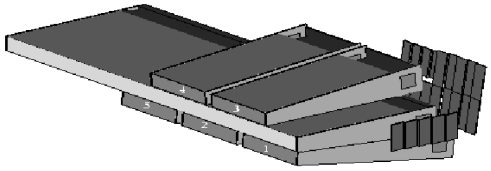

Figura 4: Os 5 módulos do protótipo do Calorímetro de Telhas

4.2. Desenvolvimento

A aplicação em foco concerne um discriminador de partículas (elétrons, píons e múons) de altas energias (cobrindo a faixa de 20 a 400 Gev) com base no calorímetro *Tilecal*. Este discriminador visa purificar o feixe de partículas para testes com protótipos desta técnica de calorimetria, pois havendo sido selecionado um determinado feixe de interesse, o processo físico que produz o feixe introduz partículas de contaminação que se deseja eliminar *online*.

No desenvolvimento da aplicação foram presentes duas etapas: uma análise *off-line* e o desenvolvimento da aplicação *online*. Para a primeira, foram utilizadas *workstations Risc System 6000* que executaram aplicações de redes neurais desenvolvidas em Fortran, utilizando o pacote de simulação Jetnet [5]. Realizou-se então,o dimensionamento da rede e do sistema de pré-processamento, assim como seu treinamento e o teste da eficiência de discriminação. Para o desenvolvimento da aplicação, utilizamos as ferramentas de programação *Assembly* da Analog Devices (*Assembly, Linker e Simulator*), a placa EZ-LAB da *BitWare Research* e o conjunto de rotinas em linguagem C para troca de dados e manipulação da aplicação na placa EZ-LAB. Vale lembrar que a placa EZ-LAB possui um ADSP-21062 e um AD-1877 que reúne as funções de conversor A/D e conversor D/A para audio.

Na etapa de pré-processamento dos dados experimentais, adotamos uma normalização realizada em duas fases: uma primeira envolvendo os neurônios associados ao Módulo 0 e uma segunda associada aos neurônios dos demais módulos. Esta normalização consistiu em dividir a entrada de cada neurônio pela energia total absorvida pelo calorímetro que foi obtida pela soma de todas as células de informação para cada evento e em seguida, realizar a intercalibração dos módulos através da multiplicação do valor de entrada por uma constante escolhida.

A topologia da rede neuronal apresenta 246 nós de entrada e 24 neurônios na camada escondida. A camada de saída apresenta 3 neurônios. A função de ativação utilizada é a tangente hiperbólica. O treinamento da rede foi realizado através do método de *backpropagation* e contava com um conjunto total de 7313 eventos de trei-no e 7313 eventos de teste, fornecendo os pesos e *tresholds* que foram utilizados na aplicação *online*. Seus três neurônios de saída visam a discriminação de elétrons, píons e múons, respectivamente, com cada um dos neurônios sendo associado a uma dada classe. Para fins de classificação, o critério de máxima probabilidade foi utilizado [7].

O desenvolvimento da aplicação *online* teve como metas a velocidade, a generalidade e a reconfigurabilidade. Assim, o programa foi desenvolvido em linguagem *Assembly* e foi estruturado em blocos ou procedimentos facilmente reconfiguráveis, o que permite que redes de diferentes topologias possam ser implementadas, assim como novos blocos de pré-processamento ou mesmo interfaceamento possam ser adicionados.

Na aplicação foram definidos 5 blocos de memória, 3 deles utilizados nas rotinas de inicialização, normalização e propagação de entradas e 2 deles utilizados no armazenamento dos pesos, *tresholds* e parâmetros de configuração da rede. É possível ainda uma completa reconfiguração dos parâmetros da rede, realizada através da modificação de uma área específica da memória, sem que seja necessária a modificação do programa fonte e uma nova compilação do mesmo.

A rotina de inicialização define registradores de uso geral que contêm constantes, e define os ponteiros da DAG1 e DAG2, ambos utilizados nas rotinas de normalização e propagação de entradas. Como estas rotinas não modificam o conteúdo destes registradores e como todo endereçamento é realizado através de *buffers* circulares, o procedimento de inicialização basta ser executado uma única vez, mesmo que sejam propagados vários eventos, diminuindo assim o tempo de processamento.

A rotina de normalização realiza um somatório de todas as entradas de um dado evento, inverte este somatório e multiplica o resultado pela constante de calibração. Cria-se assim uma nova constante que multiplicará todas as entradas relativas a etapa de normalização. Como o ADSP-2106x não possui uma instrução para cálculo do inverso ou da divisão em precisão de 32 *bits*, foi implementado uma algoritmo que toma uma aproximação inicial de 8 *bits*, fornecida por uma instrução específica, e por aproximações sucessivas calcula o inverso do número em alta precisão (32 bits). Este algoritmo é constituído por 8 instruções, sendo chamado apenas duas vezes em todo procedimento de normalização.

A rotina de propagação das entradas se utilizou de uma instrução que realiza paralelamente uma multiplicação, uma soma e dois acessos a memória. Isto permitiu que o cálculo de cada componente do produto in-

terno entre os vetores de entrada e o os vetores de peso seja realizado em um único ciclo de relógio (25 ns). Novamente, como processador não possui uma instrução para o cálculo da tangente hiperbólica, optamos pela sua implementação através de uma tabela (*Look-Up Table - LUT*), cuja utilização é indicada pela rapidez e pelo pequeno algoritmo envolvido em sua consulta. Funções periódicas como seno e cosseno, assim como funções que convergem para um dado valor como a tangente hiperbólica são freqüentemente implementadas através de LUTs. Devemos ressaltar porém que a substituição do cálculo da função por uma tabela demanda o estudo de seu impacto na aplicação em questão.

Na construção de uma tabela de tangente hiperbólica é necessário definir dois aspectos: o extremo e a resolução. O extremo está relacionado à saturação da função. Como esta função Y=tgh(x) é assintótica a +1 e –1, podemos adimitir que para x maior que um dado valor, todos y correspondentes serão iguais ao valor da função no extremo definido. Por outro lado, a resolução corresponde ao intervalo entre duas posições de x consecutivas .

O extremo define o valor máximo da saída dos neurônios e afeta mais diretamente os neurônios da última camada. Valores muito baixos para o extremo podem limitar artificialmente a capacidade de discriminação da rede. A resolução afeta principalmente os neurônios que apresentam saídas baixas já que praticamente todo o erro cometido em x se reflete em y.

Como a função tangente hiperbólica é impar, ou seja, tgh(-x)=-tgh(x), a tabela poderia ter sido construída apenas para valores positivos de x, cabendo ao software inverter o sinal da função caso x fosse negativo. Esta implementação seria mais econômica do ponto de vista de memória porém, demandaria um maior número de ciclos. Em virtude do compromisso com a velocidade, optamos pela tabela completa.

Visando verificar o impacto do uso da LUT na rede e realizar o ajuste dos extremos e da resolução para a construção de uma tabela otimizada, com um mínimo prejuízo para a capacidade de discriminação e a acuidade das saídas, realizamos uma simulação *offline* em *workstations*, onde foi construída uma rede que possui como função de ativação uma LUT de valores de extremos e resolução configuráveis. Foi possível assim experimentarmos várias configurações distintas e verificar seus impactos quanto a acuidade e discriminação.

Uma vez definida e gerada a LUT que seria utilizada, iniciamos a fase de teste da aplicação *online*. Para o teste foi desenvolvido um programa em linguagem C que lê um arquivo de eventos, carrega a aplicação e um dado evento na memória do DSP e comanda sua execução. Quando a aplicação está terminada, ela coleta os resultados da memória do DSP, salvando-os em disco. São carregados um evento por vez na memória do DSP.

A análise comparativa dos resultados assim como o teste da eficiência de discriminação foi realizado através de programas em linguagem Fortran.

4.5. Resultados Obtidos:

Na etapa *offline*, as eficiências de discriminação obtidas pela rede neuronal estão mostradas na Tabela 1 [6].

Para o ajuste da LUT, foram experimentados 17 conjuntos em que se variavam os valores extremos e a resolução e concluímos que uma tabela de 1025 posições (512 posições positivas; 512 negativas e 1 para o zero) e de extremo 4 mantinha a mesma capacidade de discriminação da rede que utilizava como função de ativação a tangente hiperbólica (rede original). A fim de prever a acuidade da futura implementação, realizamos também as distribuições do módulo do erro obtido, ou seja, do módulo da diferença entre a saída da simulação e a saída da rede original, obtendo o máximo erro cometido (Máx), o valor RMS (RMS) e o valor médio (Médio) da distribuição para cada neurônio da saída. A Tabela 2 indica os resultados obtidos numa *workstation*, para o segundo neurônio, responsável pela discriminação dos píons, o qual apresentou os maiores erros em todas as análises devido a LUT.

Tabela 1: Eficiências de discriminação para o conjunto de teste

Partícula	Total de Eventos	Eventos Identificados Corretamente	Acerto Percentual
Elétron	2166	2166	100,00 %
Píon	2623	2591	98,78 %
Múon	2524	2517	99,72 %

Repetimos os mesmos procedimentos com os arquivos gerados pela aplicação executada no DSP e pela rede original executada na *workstation*. A tabela 3 indica os resultados referentes ao segundo neurônio, o qual apresentou os maiores erros, de forma coerente com a simulação anterior.

Tabela 2: Erros devido a LUT (Simulação)

Erro=módulo (rede original – simulação)	
Eventos de Treino (TR)	Eventos de Teste (TST)
Médio: 0.3885 E-2	Médio: 0.4150 E-2
RMS : 0.4507 E-2	RMS: 0.4799 E-2
Máx. : 0.3011 E-1	Máx. : 0.3119 E-1

Observamos que os erros na aplicação DSP foram menores que os previstos na simulação em Fortran apesar de um maior espalhamento na distribuição dos erros, refletindo num maior desvio padrão. Isto está relacionado a diferença de acuidade apesar dos ambientes.

A aplicação desenvolvida na linguagem *Assembly* opera com palavras de 32 *bits*, enquanto que o Fortran opera com 64 *bits*.

Devemos lembrar que os erros numéricos devido à diferença de precisão dos ambientes atuam sobre as entradas, os pesos e os *thresholds* da aplicação, refletindo-se de forma não linear na saída.

Tabela 3: Erros devido a LUT (DSP)

Erro= módulo (rede original – rede em DSP)	
Eventos Treino (TR)	Eventos Teste (TST)
Médio: 0.3196 E-2	Médio: 0.3474 E-2
RMS : 0.5485 E-2	RMS : 0.5883 E-2
Máx. : 0.3076 E-1	Máx. : 0.3356 E-1

Quanto ao tempo de execução da aplicação, consideramos a aplicação em *Assembly* (veja Tabela 4). Como nossa aplicação se utilizou de dados previamente disponíveis na memória, na aplicação final somar-se-ão os tempos destinados ao *download* da aplicação de uma memória externa para a memória interna do SHARC e os tempos demandados pela rotina de coleta de dados e de interfaceamento.

Tabela 4: Tempos de Execução das Rotinas em DSP

Etapa	Ciclos	Tempo
Inicialização	55	1.375 μS
Normalização	771	19.275 μS
Propagação	6229	155.725 μS
Total	**7055**	**176.375 μS**
Médio	**7000**	**175 μS**

Para fins de comparação, realizamos a medida dos tempos de processamento das rotinas de normalização e propagação de entradas na rede implementanda em Fortran que foi executada em uma *workstation* equipada com um processador *Pentium II* de 300 Mhz, obtendo os tempos médios de processamento exibidos por evento na Tabela 5:

Tabela 5: Tempos de Execução das Rotinas em Fortran

Etapa	Tempo
Normalização	106.43 μS
Propagação	2007.89 μS
Total	**2144.32 μS**

Podemos perceber que a normalização em DSP é 5.5 vezes mais rápida que a realizada em *workstation*. Por outro lado, a propagação de entradas é quase 13 vezes mais rápida. Considerando ambas etapas, a aplicação no DSP é cerca de 12 vezes mais veloz.

5. Conclusões

Os processadores de sinais digitais (DSPs) são processadores que favorecem a implementação de redes neurais, dada sua arquitetura otimizada para realizar produtos internos. DSPs de ponto-flutuante são os mais indicados devido a facilidade de previsão dos resultados através de análise *off-line* em ambientes computacionais completamente distintos.

Neste trabalho apresentamos uma implementação de uma rede neuronal classificadora para o ambiente de física experimental de altas energias na teconolgia dos DSPs visando uma operação *online*. Estudos comparativos com as implementações *offline* correspondentes foram apresentados em termos do tempo de processamento e da sua acuidade.

Os resultados obtidos mostram que o processador de ponto-flutuante de 32 bits escolhido (ADSP-21062) é capaz de reproduzir a performance da realização *offline* e realizar a aplicação num tempo médio por evento significativamente inferior a plataforma de uso geral de alto desempenho.

Com base nos resultados desta aplicação e considerando o relativo baixo custo e o alto desempenho dos DSPs, pode-se considerar esta tecnologia bastante atraente para a implementação de processamento neuronal em diferentes aplicações que requeiram alta velocidade.

Referências

[1] B.Denby and D. Perret-Gallix. New Computing Techniques in Phisics Research. World Scientific,1995.

[2] C. Marven, G. Ewers. A Simple Approach to Digital Signal Processing. John Willey & Sons,1996

[3] F. Ariztizabal et al. Construction and Performance of an Iron-Scintilator Hadron Calorimeter with Longitudinal Tile Configuration. Nuclear Instruments and Methods A349:384-397, 1994.

[4] The Tilecal Collaboration. ATLAS - Tile Calorimeter Technical Desing Report. 332 páginas, CERN/LHC/96-42,1996.

[5] L. Lonnblad et al, Comp. Phys. Commun. 70 (1992) 167.

[6] J.M. Seixas, D.O. Damazio. A Neural Discriminator capable to identify impurities in the Data Sample. IEEE International Conference on Eletronics, Circuits and Systems, pp 261-264, Lisboa , Portugal, 1998.

[7] S. Haykin. Neural Networks: A Comprehensive Foundation. Second Edition. Pretince-Hall, 1999.

[8] Texas Instruments. Table Look-up and Interpolation on the TMS320C2xx. Applicacation Report, 1996.