APRENDIZADO POR REFORÇO EM INTERFACES ANALÓGICO-DIGITAIS

ADÃO DE SOUZA JR.; LUIGI CARRO
Laboratório de Prototipação, Departamento de Engenharia Elétrica, Universidade Federal do RGS
Rua Oswaldo Aranha, 103, 90035-190, Porto Alegre, RS, Brasil.
E-mails: adaojr@iee.ufrgs.br, carro@iee.ufrgs.br

Resumo— No processo de interfaceamento entre sistemas digitais e o mundo físico, a modulação sigma-delta é crescentemente utilizada tanto por não exigir circuitos analógicos de grande complexidade, quanto por permitir um bom compromisso entre custo em área e desempenho. Colocando uma rede neural em substituição ao tradicional bloco de controle da cascata de integradores do modulador, pretende-se que esta venha a aprender, de forma adaptativa, a realizar modulação sigma delta, mantendo as características de estabilidade do sistema, compromissadas sempre que se usar um modulador de ordem maior que um. O objetivo do trabalho é projetar e treinar uma rede neural a fim de implementar um classificador para modulação sigma-delta. Esta implementação foi desenvolvida digitalmente sobre uma plataforma de lógica programável. Aplicações relacionadas a este trabalho vão de sensores integrados ao controle de robôs móveis**.**

Abstract— Delta-sigma modulling based AD converters are increasingly used to inteface digital systems to the physical world. Since they offer a good tradeoff between cost in area and dynamical performance and also does not require complex analog circuits By replacing a neural network to perform the task of the control block in the integration chain of the modulator, one seeks to make it learn adaptively how to behave in order to control modulation of higher orders. The neural network is implemented and tested using EPLD technology. Applications range from integrated sensors to mobile robot control.

Keywords: neural nwtworks, reinforcement learning, delta-sigma moduling.

1- Introdução

Com o aumento da disponibilidade de processamento digital, avançaram as possibilidades de utilização de métodos computacionalmente intensivos em dispositivos de baixo custo. Com esta capacidade computacional barata surge a possibilidade de transferência de algumas das tarefas que antes se realizavam exclusivamente no domínio analógico para o domínio digital. Ao mesmo tempo, a busca pela criação de sistemas integrados oferece os sistemas reconfiguráveis em hardware como novos candidatos a serem paradigma dominante para o projeto de sistemas embarcados.

O processamento analógico é, cada vez mais, confinado a condições muito especiais de operação (freqüências elevadas, consumo muito baixo) e a preencher o espaço que separa as aplicações digitais dos sensores que lêem diretamente as variáveis físicas. O processamento digital é realizado sobre representações simbólicas discretas de elementos ou variáveis físicas. O condicionamento destas variáveis físicas é o processo pelas quais as mesmas são individualizadas.

Em arranjos de dispositivos sensores que apresentem múltiplas saídas simples do tipo digital ou pulsada, a eliminação de uma etapa de processamento analógico é fato corriqueiro, sendo o pré-processamento e determinação das variáveis de interesse do sistema realizados totalmente no domínio digital. Ao contrário, quando o dispositivo sensor apresenta saídas na forma de uma escala analógica de corrente ou tensão, seja esta linear ou não, faz-se necessária uma etapa de conversão AD. O condicionamento dos valores lidos será feito apenas após a conversão dos dados para o domínio digital (Dempsey *et alii*, 1999; Medrano-Marquez e Martin-del-Brío, 2000).

Um processo de conversão AD que oferece grandes vantagens em se tratando do projeto de sistemas integrados é o uso de modulação sigma-delta (Candy e Themes, 1992). A parte analógica exigida por tais sistemas é usualmente simples, podendo ser realizada com componentes com alta tolerância, enquanto a maior parte do processo de conversão é realizada já no domínio digital. Sendo um processo baseado em sobre-amostragem, pode-se facilmente aumentar a resolução de um conversor sigma-delta sacrificando-se seu desempenho dinâmico.

Sistemas digitais podem hoje ser implementados com relativa facilidade. Ferramentas de CAD, prototipação e testes são amplamente disponíveis. A existência de dispositivos lógicos configuráveis (FPGAs e EPLDs) torna a prototipação de projetos digitais mais barata e mais rápida que a do projeto dos sistemas analógicos. Existe, devido a isso, grande interesse na possibilidade da substituição de circuitos analógicos presentes em sistemas integrados por equivalentes no domínio digital.

Uma abordagem de projeto que leva em conta tais fatores é a que busca desenvolver todos os circuitos analógicos de um dado sistema híbrido sobre substrato digital e com componentes não-lineares que ocupam menor área. A degradação do desempenho destes circuitos decorrente de tal escolha de projeto é então compensada numa etapa posterior já no domínio digital (Carro *et alii*, 2000). Pode-se, por exemplo, realizar esta compensação é através de processamento adaptativo dos sinais recebidos (Carro *et alii*, 2000).

Trabalhos que também lidam com a implementação de sistemas analógicos utilizando-se recursos digitais podem ser vistos nas áreas de hardware evolutivo e filtragem Booleana (Miller,

1999). Autores como Dias (1995) e Salapura (1994) apresentam formas alternativas de representação de dados que apresentam vantagens no processamento de dados usando dispositivos configuráveis.

Tendo em vista a clara necessidade de interfaceamento com o mundo analógico, e os diferentes métodos a disposição para realização da interface mais conveniente sob um conjunto de critérios (área, resolução, potência consumida, velocidade), neste trabalho foca-se no estudo de uma técnica que possa ser facilmente implementada no domínio digital, consumindo um mínimo de área, e permitindo um desempenho ajustavel ao custo desejado. Dentro destes critérios, a modulação Sigma-Delta para ser a mais adequada para se atingir os objetivos.

Toma-se como ponto de partida para a realização deste trabalho a rede neural analógica implementada por Cawenberghs (1999). Uma rede em regime de aprendizado por reforço é implementada utilizando-se uma arquitetura de dispositivos neuronais. Obedece-se a divisão entre elemento associativo de seleção e elemento associativo crítico (ASE/ACE) conforme proposta por Barto *et alii* (1983). São consideradas as alternativas de implementação de argumentos dígito-seriais, bem como alternativas de implementação de redes com codificação por pulsos (Johnson *et alii*, 1999). A rede final é programada em um FPGA e testada, e o desempenho de cada versão de hardware é analisado e comparado quanto a custos e resolução.

O artigo é organizado da seguinte forma: na seção dois é feita uma breve digressão a respeito de aprendizado por reforço e da versão do algoritmo adotada. A seção três revisa as formas de implementação de redes neurais em circuitos reconfiguráveis, e explica a estrutura geral do controlador implementado. Resultados de síntese e testes são apresentados na seção quatro, enquanto a conclusão indica os rumos futuros do trabalho.

2 - Aprendizado por reforço

No regime de aprendizado por reforço, a rede deve ser capaz de identificar um padrão e relacioná-lo a uma dada recompensa (Haykin, 1994). Durante o aprendizado por reforço, a saída, no caso o rótulo da categoria, não é especificamente indicado, devendo o sistema aprender por tentativa e erro, a relacionar a saída correta com um dado sinal de recompensa. Mais possivelmente, a recompensa pode estar em um futuro não imediato, requerendo uma seqüência acertada de eventos para ser alcançada.

Um problema tradicional do aprendizado por reforço é o controle do pêndulo invertido. Este problema vem tradicionalmente sendo utilizado para a avaliação de diferentes estratégias de controle adaptativo e aprendizado de máquina.

De uma forma sucinta o problema consiste-se em equilibrar um bastão preso sobre um carro e a ele articulado, mantendo-o na vertical, podendo-se,

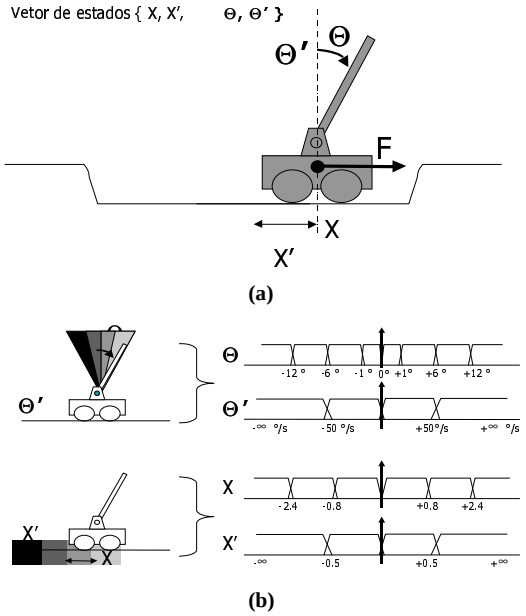

Figura 1. Variáveis de estado no *pole-balancing*.

para tanto, apenas imprimir movimentos laterais ao veículo. Como únicas entradas o sistema de controle recebe a posição do carro (X), sua velocidade (X´), a posição angular do bastão (Θ) e sua velocidade angular (Θ'), além de um sinal que informa em caso de falha (queda do bastão). Toda a movimentação, tanto do bastão quanto do carro, é confinada ao mesmo plano (fig.1a).

O funcionamento do método de aprendizado utilizado pode ser mais facilmente entendido através da analise do funcionamento do método das caixas (Michie e Chambers, 1968) que o precedeu. Neste método, imagina-se o espaço de estados dividido em um número finito de regiões disjuntas, cada uma destas, denominadas caixas, possui uma ação de saída associada. Deste modo, a medida que o sistema evolui dinamicamente no tempo, estas regiões são percorridas e suas regras de saída correspondentes são acionadas. Na figura 1b pode-se ver a uma discretização possível das variáveis de estado no caso do equilíbrio do bastão (Barto *et alii*, 1983). Em cada uma das caixas foi definido um conjunto de variáveis correspondentes ao número de atitudes mutuamente excludentes possíveis de serem tomadas. Essas variáveis são denominadas eligibilidades, e a saída efetiva será escolhida como sendo sempre aquela que tiver a maio eligibilidade. No exemplo estudado, cada caixa contém duas eligibilidades contendo cada uma o tempo decorrido desde a última vez em que ela foi escolhida até o momento que houve uma falha. Com o passar do tempo, emerge um padrão que indica a melhor estratégia global de controle para evitar que o bastão caia.

Uma implementação de aprendizado por reforço análoga ao método das caixas e que utiliza um elemento neural é o chamado Elemento de Busca Associativa (Associative Search Element - ASE) (Barto *et alii*, 1983). A saída deste elemento tem um potencial de ativação constantemente perturbado por ruído aleatório, de forma a ser exatamente zero

ASE:

$$P[y(t)] = \Sigma_{i=1:n} w_i(t)\, x_i(t) + noise$$
$$w_i(t+1) = w_i(t) + \alpha . r(t) . e_i(t)$$
$$e_i(t+1) = \delta\, e_i(t) + (1-\delta) y(t) x_i(t)$$

ACE:

$$P[rp(t)] = \Sigma_{i=1:n} v_i(t)\, x_i(t)$$
$$v_i(t+1) = v_i(t) + \beta[r(t) + \gamma P(t) - P(t-1)] x_i(t)$$
$$x_i(t+1) = \lambda x_i(t) + (1-\lambda) x_i(t)$$
$$rp(t) = r(t) + \gamma P(t) - P(t-1)$$

Figura 2. Aprendizado por reforço (associador/crítico).

quando todos os pesos são iguais. Deste modo, a probabilidade de aplicação inicialmente é igual tanto para a força vinda da esquerda quanto para a força vinda da direita. Um decodificador externo transforma cada um dos agrupamentos correspondentes às caixas do espaço de estados original em um vetor de excitação onde todos as entradas menos uma são zero. Um sinal de reforço negativo (-1) é fornecido sempre que houver uma falha. Modificando-se o valor dos componentes do vetor de pesos w associa-se estocasticamente a cada estado do sistema uma ação (fig.2).

Implementado apenas com um módulo de busca associativa, não há qualquer melhoria com relação ao método clássico das caixas. A medida que os reforços vão se tornando mais raros o aprendizado se torna mais lento e os incrementos no desempenho menores. Entretanto, o acréscimo de um segundo elemento, encarregado de fazer a previsão de reforços melhora grandemente o desempenho do sistema como um todo, tendo se tornado uma solução clássica de aprendizado por reforço. A figura 2 sintetiza o eqüacionamento completo do método ASE/ACE. Para uma extensiva discussão dos diversos métodos de aprendizado por reforço, o leitor deve se reportar a Sutton *et alii* (1998).

Na aplicação ao problema da modulação sigma-delta, para utilização dos conceitos de aprendizado por reforço toma-se a mesma abordagem apresentada por Cawemberghs (1999), formando-se o vetor de estados a partir da polaridade do sinal de entrada e da saída de cada etapa de integração. A saída modulada resultante pode ser computada diretamente da saída da rede.

3 - A rede descrita

A implementação de redes neurais em hardware é um assunto bastante amplo. As diversas soluções descritas na literatura variam vastamente com respeito à intercomunicação entre unidades processadoras, ao grau de paralelismo, à forma de representação numérica entre outras coisas (shoenauer *et alii*, 1998). Devido ao grande número de possibilidade de projeto, optou-se por partir da análise das características de processamento das equações de atualização. A partir das características desta célula foram feitos alguns experimentos a fim de definir a forma de representação dos dados, a

$$y_i(t+1) = histerese(y_i(t) - \alpha . sinal(y_i(t)) . UPD_i(t))$$
$$q_i(t+1) = q_i(t) - \beta\ UPD_i(t)$$
$$UPD_i(t) = \begin{cases} -rp(t), & se\ \boldsymbol{e}_i(t) > 0. \\ 0, & senão. \end{cases}$$
$$e_i(t+1) = \begin{cases} 1, & se\ i = \chi(t) \\ e_i(t) - \delta, & senão. \end{cases}$$

Figura 3. Aprendizado simplificado, a histerese é opcional.

resolução necessária e o paralelismo entre células. Os elementos de comunicação entre as unidades mínimas de processamento foram definidos a seguir delineando a estrutura geral do dispositivo.

3.1 - Unidade de processamento

O cálculo da eligibilidade e do traço é feito para cada uma das entradas dos neurônios. Já o potencial de ativação da saída e o reforço previsto são computados localmente, porém totalizados globalmente. Desta maneira deveria ser possível armazenar para cada uma das entradas quatro variáveis.

Segundo as equações definidas para a atualização dos pesos em cada elemento de busca associativa (ASE) e de cada elemento de crítica adaptativa (ACE), postulando-se que seja N o número de estados do sistema, haverá 6 x N + 1 multiplicadores. Analisando-se as operações realizadas pode-se ver que destes multiplicadores apenas 4 x N, multiplicam duas grandezas variáveis, sendo os demais operações simples escalonamentos por constantes que consomem bem menos área.

Utilizando-se a modificação sugerida por Cawemberghs (1999), descarta-se a etapa de totalização das saídas sendo esta substituída por uma seleção com base no estado do sistema do potencial de ativação a ser manifesto. Com isso implementa-se uma versão do algoritmo de saída não probabilística. O efeito desta modificação é uma perda de desempenho geral do sistema o qual passa a ter o chamado comportamento 'guloso' (greedy). A nova proposta para as equações de atualização dos parâmetros de cada célula pode ser vista na figura 3.

Percebe-se nessa configuração que a adição de um fator de atualização (UPDi) no lugar do reforço previsto e a utilização de um simples contador para o calculo do decaimento de eligibilidade reduz bastante a complexidade da célula de processamento. O número de multiplicações total reduz-se a 3 x N, das quais apenas N exigem um multiplicador completo. Cumpre notar que adotando estas mudanças está-se trocando desempenho no processo de aprendizado por uma redução na área do circuito resultante

3.2- Considerações sobre a forma de representação dos argumentos

O projeto visando implementação em dispositivos lógicos configuráveis tem algumas características peculiares que devem ser observadas quando da escolha da forma de representar os dados. Apesar de se utilizar representação binária posicional como forma mais comum de representação em

circuitos digitais, sabe-se que barramentos muito largos de dados tendem a ter uma implementação pouco eficiente em grande parte dos dispositivos programáveis. Esquemas alternativos de representação propostos incluem freqüência (Murray e Tarassenko, 1994; Hikawa, 1999), largura (Murray e Tarassenko, 1994) e densidade de pulso (Mead, 1989; Rossmann *et alii*, 1997). Pode-se ainda subdividir as formas de modulação de densidade em modulação delta (Salapura, 1994), e estocástica (Rossmann *et alii*, 1997; vanDaalen *et alii*, 1993ab; Bade e Hutchings, 1994). Bem mais recentemente, a representação através de intervalo entre espículas vem sendo investigada (Korkin *et alii*, 1998).

Todas estas técnicas utilizando fluxos de bits demandam um número de ciclos de relógio que cresce exponencialmente com a resolução em bits do sinal que se quer representado (um fluxo de N ciclos pode representar apenas N+1 números, contra 2^N no caso de representação posicional). Além disso, demandam uma grande quantidade de memória quando comparados às representações posicionais. Sua principal vantagem diz respeito ao custo da multiplicação, normalmente implementada com umas poucas portas lógicas, o que viabiliza a construção de redes com muito mais nós. Algumas destas implementações estão sujeitas a erros devido a considerações de correlação entre fases dos fluxos, e necessitam de hardware adicional para corrigi-los. Para maiores referências, Dias (1995) faz um estudo completo das características de processamento de sinais no domínio sigma-delta. Já no caso de aritmética de sinais estocásticos, vanDaalen (1993) oferece uma solução para o problema da correlação em aritmética estocástica, Rossmann (1997) apresenta uma versão otimizada para uso em FPGA.

Caso se fizesse necessária uma maior resolução dos operandos a implementação deveria ser feita utilizando representação posicional ou redundante (Perez-Uribe, 1999). Mesmo se trabalhando com representação binária posicional, é possível realizar de diferentes formas a implementação dos operadores de multiplicação e soma. Se o número de bits de resolução necessário for muito grande, poder-se-á optar por utilizar partes operativas digito-seriais (Chang *et alii*, 1998) ou usar um sistema híbrido de representação (Dick e Harris, 2000).

Para se poder optar por uma forma de representação dos operandos foi realizada a implementação de um bloco simples multiplica-soma para oito bits utilizando modulação de pulso , segundo (Hikawa, 1999), e comparado este resultado com uma implementação posicional. Observou-se um certo ganho de área (69LCs + 33% mem contra 80LCs), porém, considerando-se a alta utilização de memória do bloco resultante, decidiu-se por dar preferência a uma representação posicional.

3.3 - Resolução necessária

Uma vez que este algoritmo específico de aprendizagem por reforço não havia sido implementado previamente em sistema digital, a determinação da resolução mínima dos operandos teve de ser definida experimentalmente. Para tanto, foram realizadas diversas descrições da célula fundamental com número de bits variando entre oito e trinta e dois. As características destas implementações são descritas em pormenores na seção quatro.

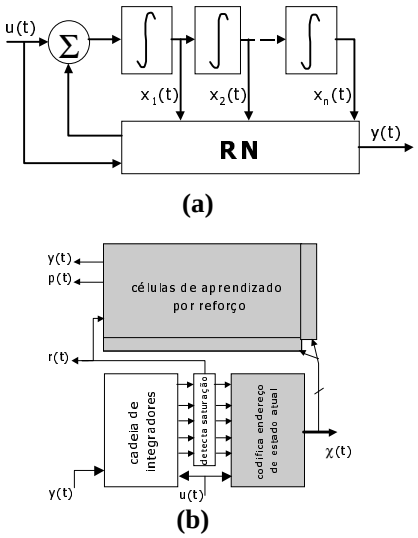

Figura 4. Conexões externas do controlador

3.4 - Estrutura do dispositivo

Os elementos que formam o sistema como um todo podem ser vistos na figura 4a. A partir de um conjunto de integradores, os sinais de saída de cada etapa passam por um circuito comparador, que detecta quando o sinal excedeu os limites de operação, e chegam diretamente ao bloco digital. Deste último bloco parte o sinal de comparação que é subtraído do sinal a ser modulado. O controlador digital projetado corresponde à parte em cinza da figura 4b. Vê-se junto a ele os sinais de U e X_i oriundos da cascata de integradores, após terem entrado em um banco de comparadores que determina se seus limites de operação foram excedidos, servirem para alimentar o codificador de estado atual da rede.

A partir do sinal de estado um bloco de codificação determina qual das células de processamento deve receber um valor não nulo em sua entrada **x**. Um segundo bloco se encarrega de receber os valores calculados e totalizá-los, determinando o valor da saída **y**. Essa totalização pode ser tanto o processo de acumulação de acordo com a rede descrita por Perez-Uribe (1999), quanto a seleção da saída da caixa vencedora (como no método das caixas).

No interior das células de processamento pode-se utilizar o número de ciclos que for necessário para o calculo das saídas. Um mecanismo de propagação de habilitação garante que apenas valores já calculados sejam passados adiante. O valor

Tabela 1. Resultados de síntese obtidos

Circuito	Algoritmo	Resolução (bits)	Ocupação
Células processadoras aritméticas por elemento de busca associativa			
case08	associador	8	43 LCs
case12	"	12	63 LCs
case16	"	16	83 LCs
case32	"	32	161 LCs
Células processadoras usando algoritmo reduzido ASE/ACE			
caseacepar08	Associador / crítico	8	110LCs
caseacemux08	"	8	93LCs
caseacepar12	"	12	142LCs
caseacemux12	"	12	128LCs
Redes completas de elementos, vide texto para explicação.			
rede4_casemux12	"	12	200LCs
rede8_casemux12	"	12	420 LCs
rede8_casepar08	"	8	285 LCs
rede8_caseace08*	"	8	1958 LCs
rede8_caseace08	"	8	883 LCs
Todos utilizam o FPGA da Altera EPF10k20RC84, exceto (*) EPF10k40RC208			

final de ativação para **c**omputado passa por uma função de limiar que determina a saída da rede.

4 - Resultados obtidos

Foram feitas células utilizando diferentes arquiteturas de acordo com números de bits de argumento. Para todas aquelas com menos de doze bits utilizaram-se somadores sem qualquer forma de otimização (*carry-select*, *carry-look-ahead*, etc.), uma vez que já foi determinado que em projetos com FPGA deste tamanho não existe qualquer ganho em desempenho (Xing e Yu, 1998). Para as demais, utilizou-se um esquema de *carry-select.* Realizou-se toda multiplicação com mais de doze bits apenas através de operações digito-seriais.

Quanto ao número de multiplicadores por célula, testou-se duas possibilidades: o uso de um único multiplicador compartilhado entre as três operações da célula e o uso de circuitos dedicados para as multiplicações das constantes em paralelo. Foram também projetadas versões com as equações originais para o elemento de busca associativa.

O projeto foi adaptado para lidar com cascatas de até dois integradores, ou seja, até oito possíveis estados. Devido ao tamanho de uma implementação completa da rede a síntese completa somente foi gerada para redes de até oito elementos com no máximo doze bits de resolução. A extensão para os demais circuitos é trivial, não tendo sido realizada apenas pelo tempo de síntese excessivo e a impossibilidade de montagem de um protótipo com os recursos disponíveis.

Nos resultados de compilação (tabela 1) pode-se observar a grande economia de área obtida pelo uso da forma simplificada de algoritmo de reforço. O dispositivo utilizado pela rede de oito elementos com critico e elemento de busca para o caso de oito bits tem praticamente o dobro do tamanho e somente pôde ser colocada em um elemento maior da mesma família de dispositivos.

5 - Conclusões e trabalho futuro

Este trabalho é um dos passos iniciais de um processo de estudo de propostas para a definição de um conjunto de arquiteturas e técnicas que se preste ao desenvolvimento e prototipação de sistemas híbridos sobre circuitos de tecnologia reconfigurável predominantemente digital.

Demonstrou-se a possibilidade da implementação de uma rede capaz de apreender e estabilizar o processo de modulação sigma-delta sobre um sistema digital relativamente pouco exigente. Pôde-se determinar que tal problema não requer para tanto uma alta resolução dos operandos. Os resultados encontrados, apesar de serem preliminares e necessitarem uma análise quantitativa mais sistemática, foram promissores, apontando possibilidades de desenvolvimento futuro.

Além destes desenvolvimentos, algumas arquiteturas de redes pulsadas, como a proposta por Salapura (1994) devem ainda ser testadas, sendo, uma comparação sistemática das diversas propostas para este forma de representação dos dados com vistas à otimização de representação através de algoritmos evolutivos, um tópico ainda em aberto.

Referências bibliográficas

Bade, S.L.; Hutchings, B.L. (1994). FPGA-based stochastic neural networks – Implementation. *IEEE Workshop on FPGAs for custom computing machines*, Napa Valley, EUA, pp.189-198.

Barto, A.G.; Sutton, R.S.; Anderson, C.W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man and Cybernetics (setembro/outubro), Vol.13, n.º 5, pp. 834-846.

Candy, J.C. and Temes, G.C. (1992). Oversampling delta-sigma data converters: Theory, design and simulation. IEEE Press, New York, EUA.

Carro, L.; Souza Jr., A.; Negreiros, M.; Jahn, G.P.; Franco, D. (2000). Non-linear components for

mixed circuits analog front-end. *Design Automation and Test in Europe (DATE' 2000)*, Paris, França.

Cawenberghs, G. (1999). A nonlinear noise-shaping delta-sigma modulator with on-chip reinforcement learning. Analog Integrated Circuits and Signal Processing (Fevereiro), Vol.2/3, n.º 18, pp.289-300.

Chang, Y-N; Satyanarayana, H; Janardhan. P.; Keshab K. (1998) Systematic design of high-speed and low-power digit-serial multipliers. IEEE Transactions on Circuits and Systems-II. Analog and Digital Signal Processing (Dezembro), Vol.45, n.º 12, pp. 1585-1596.

Dempsey, G.L.; Alig, J.S. & Redfield, D.E. (1996). Using analog neural networks for control sensor linearization. Department of Electrical Engineering and Computer Engineering, Bradley University. Peoria, EUA.

Dias, V.F. (1995). Signal processing in the sigma-delta domain. Microeletronics Journal, Vol.1, n.º 26, pp.543-562.

Dick, C.; Harris, F. (2000). FPGA signal processing using sigma-delta modulation. IEEE Signal Processing Magazine (Janeiro), Vol. 17, n.º 1, pp. 30-35.

Haykin, S. (1994). Neural networks: A comprehensive foundation. 2,nd Edition, MacMillan Publishing, New York, EUA.

Hikawa, H. (1999). Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics. IEEE Transactions on Neural Networks (Maio), Vol.10, n.º 3, 545-553.

Johnson, J.L.; Padgett, M.L. & Omidvar, O. (1999). Overview of Pulse Coupled Neural Networks (PCNN Special Issue). Transactions on Neural Networks (Maio), Vol.10, n.º 3, pp. 461-463.

Korkin, M.; Nawa, N.E.; deGaris, H. (1998). A 'spike interval information coding' representation for ATR's CAM-brain machine (CBM). *Second International Conference on Evolvable Systems: From Biology to Hardware (ICES'98).* Lausanne, Suiça.

Mead, C. (1989). Adaptive retina. C. Mead and M. Ismail, ed. Analog VLSI Implementation of Neural Systems. pp. 239-246, Kluwer Academic Publisher, Boston, EUA.

Medrano-Marquez, N.J.; Martín-del-Brío, B. (2000). A General Method for Sensor Linearization based on Neural Networks. *ISCAS 2000 - IEEE International Symposium on Circuits and Systems.* Genebra, Suiça.

Michie, D.; Chambers, R.A. (1968). BOXES: An Experiment in Adaptive Control. E. Dale and D. Michie, eds. Machine Intelligence, Vol. 2., pp.137-152, Oliver and Boyd.

Miller, J. (1999). Evolution of digital filters using a gate array model. *First EvolASP'99 Workshop on Image Analysis and Signal Processing*, pp. 17-30, Goteborg, Suiça.

Murray, A.; Tarassenko, L. (1994). Analogue Neural VLSI: A pulse stream approach, Chapman & Hall, London, UK.

Perez-Uríbe, A. (1999). Structure Adaptable Digital Neural Networks. Tese de doutorado, n° 2052, École Politechnique Fédérale de Lausanne, Lausanne, Suiça.

Rossman, M.; Bülmeier, A.; Manteuffel, G.; Goser, K. (1997). Short and Long-Term dynamics in a Stochastic Pulse Stream Neurom Implemented in FPGA. *Artificial Neural Networks: 7th International Conference (ICANN 97).* Vol. 1327 of Lecture Notes in Computer Science, pp. 1241-1246, Springer-Verlag, Berlin, Alemanha.

Salapura, V. (1994). Neural networks using bit stream arithmetic: A space efficient implementation. *IEEE International Symposium on Circuits and Systems*, London, UK.

Schoenauer, T.; Jahnke, A.; Roth, U. and Klar, H. (1998). Digital neurohardware: Principles and perspectives. *Neuronal Networks in Applications - NN'98*, Magdeburg, Alemanha.

Sutton, Richard S.; Barto, Andrew G. (1998). Reinforcement learning: An introduction. MIT Press, Londres, Inglaterra.

vanDaalen, M.; Jeavons, P.; Shawe-Taylor, J. (1993a). A Stochastic Neural Architecture That Exploits Dynamically Reconfigurable FPGAs. *IEEE Workshop on FPGAs for Custom Computing Machines.* Los Alamitos, EUA.

vanDaalen, M.; Jeavons, P.; Shawe-Taylor, J.; Cohen, D. (1993b). Device for Generating Binary Sequences for Stochastic Computing, Electronics Letters (Janeiro), Vol.1, n.º 29, pp.80-81.

Xing, S.; Yu, W.H. (1998). FPGA Adders: Performance Evaluations and Optimal Design. IEEE Design and Test of Computers (Janeiro/Março), Vol.15, n.º 1, pp.24-29.