Hardware Evolutivo em FPGA usando o Processador NIOS II

Bruno de Abreu Silva[1] e Wilian Soares Lacerda[2]

[1] Universidade de São Paulo, Instituto de Ciências Matemáticas e Computação, São Carlos - SP, Brasil
brunoas@icmc.usp.br

[2] Universidade Federal de Lavras, Departamento de Ciência da Computação, Lavras - MG, Brasil
lacerda@ufla.br

Resumo The complexity of embedded systems is increasing more and more in the last years. Therefore, there is a necessity to create new methodologies to design and implement these systems that allow a faster and easier development. On the other hand, fault-tolerant systems capable to operate in critical environments is a feature extremely desired in many applications like space exploration. The Evolutionary Hardware (EHW) is the approach that develops, at reconfigurable hardware, systems that the reconfiguration is under the control of a Evolutionary Algorithm. The EHW can change its own hardware structure when the environment or the task changes. So, the EHW can adapt itself to the environment and this characteristic allow the implementation of fault-tolerant systems. In addition, EHW can be used to look for ways to design and implementation a specific circuit. The main goal of this work is present and validate a methodology to create an Evolutionary Hardware platform at FPGA using the Nios II Processor. This system is capable to implement combinational circuits and adapt itself in accord to new tasks required (online adaptation). After that, some discussions, conclusions and further works are presented.

1 Introdução

O projeto de sistemas embarcados envolve a preocupação com várias questões. Por exemplo, para uma dada aplicação qual o poder de computação que o processador deve ter? Será necessário um sistema operacional? Se sim, quais funcionalidades esse sistema deve ter? Qual a capacidade de memória necessária? Há operações críticas que devem obedecer um determinado tempo de execução? Onde colocar aceleradores de hardware para aumentar o desempenho? Quais partes do sistema serão implementadas em hardware ou em software? Diante dessas e outras questões, um ambiente de desenvolvimento deve permitir uma fácil criação de protótipos com diferentes características para facilitar a análise de quais são as melhores respostas a tais questões.

A dificuldade de encontrar boas respostas a tais perguntas tem aumentado nos últimos anos, pois os sistemas embarcados tornaram-se mais complexos e

cada vez mais aumenta-se a necessidade de sistemas capazes de realizar tarefas críticas onde não existe um especialista ou mesmo um técnico. Consequentemente, alternativas que simplificam o projeto e que ao mesmo tempo aumentam a robustez de tais sistemas têm se tornado cada vez mais importantes.

Uma das tentativas de criação de sistemas tolerantes a falhas e mais robustos que tem sido pesquisada é o Hardware Evolutivo - EHW (do inglês *Evolvable Hardware*). EHW é um hardware reconfigurável cuja configuração está sob controle de um Algoritmo Evolutivo. Esse tipo de hardware viabiliza a construção de sistemas capazes de alterar sua própria estrutura de hardware em função de mudanças no ambiente ou na tarefa requerida. Além disso, o EHW muda o foco de projeto: não se projeta o hardware de fato, mas procura-se por uma boa representação para o circuito (o genótipo) e o projeto de uma especificação que servirá para avaliar as funcionalidades do circuito (o *fitness*).

Geralmente, o dispositivo usado para implementar o EHW é o FPGA (*Field Programmable Gate Array*). FPGAs são dispositivos reconfiguráveis que possuem grande flexibilidade para o desenvolvimento de protótipos e de sistemas digitais propriamente ditos. Atualmente, empresas que fabricam FPGAs, como a Altera Corporation, desenvolvem também ferramentas de software que auxiliam a criação rápida e a custo reduzido de sistemas embarcados completos.

Neste artigo, será apresentado um trabalho cujo objetivo é a validação de uma metodologia de criação de um sistema de EHW em FPGA utilizando o processador Nios II. A principal motivação para a utilização do processador Nios II para implementação do Hardware Evolutivo é que não foi encontrado nenhum trabalho na literatura que fizesse o uso do mesmo. Além disso, o projeto do sistema completo é feito via software. Essa característica permite que várias arquiteturas sejam testadas facilmente e rapidamente pelo projetista, facilitando a busca pelas respostas às questões levantadas no início desta seção.

2 Hardware Evolutivo

O hardware reconfigurável cuja configuração está sob controle de um Algoritmo Evolutivo é chamado de *Hardware Evolutivo* [1].

O conceito de EHW foi introduzido em 1991 por De Garis [2]. Atualmente, segundo [3], o Hardware Evolutivo utiliza como plataforma reconfigurável dispositivos lógico-programáveis como *Field Programmable Transistor Arrays* (FPTAs) e *Field Programmable Gate Arrays* (FPGAs). E como Algoritmo Evolutivo, de acordo com [1], pesquisadores preferem usar Algoritmos Genéticos (GAs, do inglês *Genetic Algorithms*) por duas razões. Primeiro, Algoritmos Genéticos podem usar bits de configuração de uma arquitetura base (*bitstream*) como cromossomo. Segundo, os GAs podem ser facilmente implementados em hardware, e no caso executam mais rapidamente do que a implementação em software.

Como pode ser encontrado em [4], o EHW pode ser classificado como *intrínseco*, *extrínseco* ou *evolução on-chip*. Na evolução extrínseca, a evolução é realizada em software, assim como a avaliação dos circuitos. Então, somente a melhor solução é implementada em hardware reconfigurável. Já na evolução

intrínseca, a avaliação ocorre diretamente em hardware e em tempo real. Finalmente, na evolução on-chip, o processo evolutivo está localizado no mesmo chip que o circuito evoluído ou, em outros casos, se tem um processador on-board executando o algoritmo evolutivo.

A evolução extrínseca pode ser vantajosa se a independência do dispositivo é um requisito ou se o dispositivo não está disponível para testes on-chip/on-board. Já nos casos onde evoluir em um dispositivo específico faz parte dos requisitos, em ambientes reais ou em situações imprevistas (como exploração espacial) é mais adequado utilizar evolução on-chip ou evolução intrínseca [5].

De acordo com o tipo de circuitos a ser evoluído, EHW pode ser classificado como *digital*, *analógico* ou *híbrido* (analógico e digital).

Uma outra classificação é em relação ao nível de abstração dos circuitos evoluídos. Se o Algoritmo Evolutivo é usado para selecionar componentes ou valores dos componentes em um circuito analógico ou criar circuitos digitais a partir de blocos de construção de baixo nível, o EHW pode ser chamado de *fine-grained*. Por outro lado, se envolve a seleção de topologias e montagem de circuitos a partir de blocos funcionais de alto nível, o EHW é classificado como *function-level*.

O uso de Algoritmos Evolutivos torna possível o desenvolvimento de circuitos eletrônicos melhores que os obtidos por métodos convencionais. Entretanto, o EHW enfrenta atualmente uma grande limitação: escalabilidade, ou seja, o tamanho dos circuitos evoluídos tem sido bem pequeno para o EHW substituir de fato as técnicas convencionais. Além disto, o tempo para o cálculo do *fitness* cresce com o aumento da combinação do número de entradas e saídas do problema. A grande maioria dos pesquisadores, como [3], [6], [4], [7] e [8], tem dedicado esforços para melhorar a escalabilidade dos EHWs.

3 O Processador Nios II para Sistemas Embarcados

O Nios II é um processador RISC de propósito geral que possui, como consta em [9], as seguintes características, entre outras: ambiente de desenvolvimento baseado no GNU C/C++ e Eclipse IDE; ambiente de desenvolvimento integrado que permite depuração do hardware.

Um sistema com o processador Nios II é equivalente a um microcontrolador ou a um "computador em um chip", pois inclui o processador Nios II propriamente dito, um conjunto de periféricos on-chip, memória (on-chip e interfaces para off-chip). Tudo isso é implementado em um único dispositivo da Altera, utilizando seus softwares de desenvolvimento (no caso deste trabalho: Quartus II, SoPC Builder e Nios II IDE).

Segundo [10], os passos para desenvolvimento de um sistema Nios II em plataformas da Altera Corporation são essencialmente semelhantes aos passos de desenvolvimento de um sistema embarcado típico, são eles: análise de requisitos do sistema; definição e geração do hardware do sistema Nios II através do SoPC Builder; integração do sistema do SoPC Builder em um projeto no Quartus II; compilação do projeto do Quartus II e verificação do atendimento às restrições

de tempo; criação de um novo projeto no Nios II IDE; compilação do projeto; executar o software no Instruction Set Simulator (ISS) e/ou na plataforma alvo.

É importante ressaltar que todas as etapas de desenvolvimento do software pode ser feita paralelamente ao desenvolvimento do hardware. Isso tem grande importância quando se tem equipes trabalhando em um projeto complexo.

4 A Implementação do Sistema

O kit de desenvolvimento Nios II Development Kit - Stratix II Edition da Altera foi utilizado para implementar o EHW. O kit e suas características podem ser consultados em [11].

Uma vez que o objetivo deste trabalho é apenas apresentar e validar a metodologia proposta, não há a preocupação a princípio com questões relacionadas ao consumo de energia, ou com a escalabilidade do EHW. O sistema de EHW projetado é um sistema que deve possuir um módulo para receber os requisitos que ele deve cumprir para realizar uma determinada tarefa, deve possuir um módulo funcional capaz de gerar um circuito adequado e otimizado de acordo com a necessidade identificada, deve possuir um dispositivo de hardware que possa ser reconfigurado com o circuito encontrado. Por fim, o sistema deve ser capaz também de se adaptar a eventuais mudanças nas necessidades e assim executar novamente a otimização do circuito e a reconfiguração do hardware. Outra característica desejável é que o hardware reconfigurado continue funcionando enquanto o processo de evolução de outros circuitos ocorre.

Para atender aos requisitos desejados, as seguintes decisões de projeto foram tomadas: a entrada do sistema é um arquivo com uma tabela-verdade que descreve as características de um circuito digital combinacional com n entradas e 1 saída que representa a tarefa atual solicitada ao sistema; ao receber a tabela-verdade, o módulo otimizador de circuitos procura o melhor circuito possível através de um Algoritmo Genético; após a evolução, o módulo envia somente o melhor circuito encontrado para ser configurado no dispositivo de hardware; uma PAL (*Programmable Array Logic*) é o dispositivo a ser reconfigurado. A PAL é um dispositivo capaz de armazenar qualquer expressão Booleana na forma de soma de produtos dentro dos seus limites de tamanho. Ela é poderosa, pois é capaz de realizar qualquer tarefa que possa ser modelada como um circuito digital combinacional, como por exemplo reconhecimento de padrões binários [12].

O ciclo do funcionamento do sistema pode ser visto no fluxograma da Figura 1.

4.1 O Hardware do Sistema

O hardware é composto por um processador Nios II, uma memória RAM, um display LCD para exibir informações e uma PAL para ser reconfigurada. O diagrama simplificado do hardware pode ser visto na Figura 2.

Através da interface JTAG, a tabela-verdade é lida pelo software que está executando na CPU NIOS II. A CPU por sua vez é responsável por se comunicar

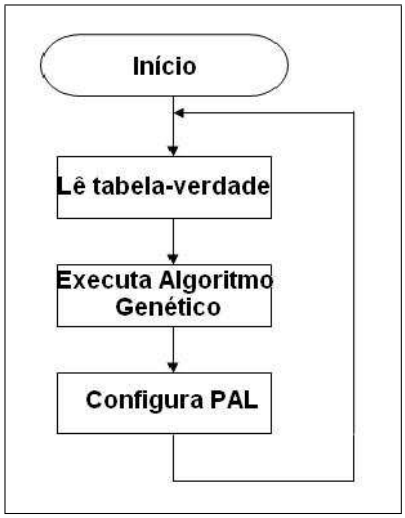

Figura 1. Fluxograma do Sistema

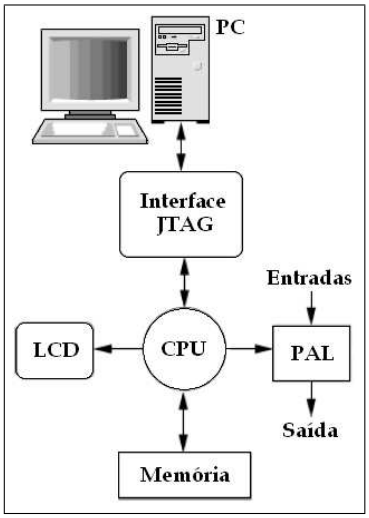

Figura 2. Hardware do Sistema

com o display LCD e também por configurar a PAL cada vez que um novo circuito for encontrado pelo Algoritmo Genético.

A PAL recebe entradas binárias e sua saída é determinada de acordo com o circuito que está configurado nela. A memória DDR SDRAM armazena o aplicativo em execução e seus dados. A PAL recebe os dados de entrada através dos 4 botões presentes no kit e a saída da PAL é indicada por um LED. A PAL foi implementada na linguagem de descrição de hardware VHDL e foi integrada ao sistema através do diagrama de bloco no programa Quartus II.

4.2 O Software do Sistema

O software executa três tarefas em loop infinito: realiza leitura da tabela-verdade através de arquivo; executa minimização do circuito através do Algoritmo Genético; grava o circuito na PAL através de comandos enviados pela interface paralela do processador NIOS II.

A seguir, será dado um detalhamento maior sobre o Algoritmo Genético implementado. Considere os seguintes indivíduos da Figura 3.

Cada indivíduo representa uma expressão Booleana na forma de soma de produtos. E cada cromossomo do indivíduo representa um termo de produto. O tamanho de cada cromossomo é determinado pelo número de entradas que a tabela-verdade possui. Os valores possíveis para cada gene são os seguintes: valor da entrada com operador NOT, representada pelo valor 0; valor da entrada sem o operador NOT, representada pelo valor 1; ausência da variável no termo de produto (*don't care*), representada pelo valor 2.

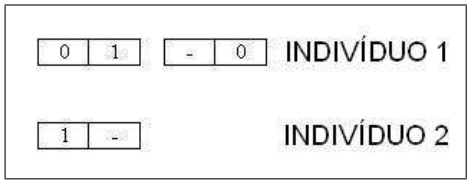

Figura 3. Indivíduos Exemplo

Os indivíduos dados como exemplo representam as seguintes expressões Booleanas respectivamente (considerando que a primeira entrada é A e a segunda é B):

- $(\neg A \wedge B) \vee \neg B$;
- A;

A população inicial é gerada de forma aleatória. Primeiramente, são sorteados quantos termos de produto cada indivíduo terá. Depois, são definidos os termos de produto para comporem o indivíduo. A estrutura da população utilizada e o método de seleção foram inspirados nos mesmos utilizados em [13].

O cruzamento utilizado foi o cruzamento uniforme e foram implementados três tipos de mutações diferentes: insere um novo termo de produto gerado aleatoriamente; remove um termo de produto escolhido aleatoriamente; troca o valor de uma posição.

A função *fitness* avalia o número de acertos e o tamanho do circuito. Para valorizar os indivíduos que acertam mais linhas da tabela-verdade, a função utilizada foi a mesma encontrada em [8], e é a seguinte:

$$f_1 = \frac{100 \times \sum_{j=0}^{2^i-1} 1 - \mid x_j - d_j \mid}{2^i}, \tag{1}$$

onde i é o número de entradas do sistema, cada j, variando de 0 a $2^i - 1$, representa uma combinação das entradas, x_j é a saída obtida pelo indivíduo para a combinação j das entradas e d_j é a saída desejada para a combinação j das entradas.

Para valorizar os indivíduos menores, utilizou-se a seguinte expressão:

$$f_2 = \frac{1}{100 + size}, \tag{2}$$

onde $size$ é o número de termos de produto do indivíduo. A constante somada no denominador foi obtida através de testes, foi a constante que proporcionou melhores resultados e pareceu ponderar adequadamente o valor relativo entre os objetivos nos casos testados. O *fitness* do indivíduo avaliado é dado por:

$$f = -(f_1 + f_2), \tag{3}$$

e o objetivo do Algoritmo Genético é encontrar o indivíduo com o menor valor de *fitness*.

Foram utilizados dois critérios de parada usados em conjunto para o GA: número máximo de gerações; e convergência do melhor indivíduo [14]. Este último foi realizado usando os valores do desempenho do melhor indivíduo da geração atual e da geração anterior conforme equação 4.

$$f_{currentMin} - f_{lastMin} = 0, \tag{4}$$

Se a situação descrita pela equação 4 ocorrer por n gerações consecutivas o Algoritmo Genético para sua execução. Onde $n = numero\ de\ linhas\ da\ tabela\text{-}verdade * numero\ de\ entradas\ da\ tabela\text{-}verdade$.

Após o Algoritmo Genético terminar sua execução, o indivíduo encontrado como solução é gravado na PAL. Depois do processo de gravação, a PAL está pronta para uso. O programa então reinicia sua execução lendo novamente os arquivos de entrada, que podem ter sido mudados simulando uma alteração do ambiente. E assim se define o ciclo de execução do software do sistema.

5 Resultados e Discussões

O hardware completo do sistema ocupou apenas 16% dos elementos lógicos da FPGA e 45% dos pinos de E/S. Somente a PAL ocupou 11% dos elementos lógicos.

Para a realização dos testes, os valores dos parâmetros do Algoritmo Genético eram os seguintes: Tamanho da população = 13 indivíduos; taxa de cruzamento = 100%; taxa de mutação = 20%.

Os resultados estão expressos na Tabela 1. Cada linha da tabela representa o cálculo do *fitness* para a solução dada pelo Algoritmo Genético e para a solução dada pelo ESPRESSO (minimizador de funções Booleanas [12]). O Algoritmo Genético implementado foi capaz de encontrar o circuito referente a porta XOR através de soma de produtos. Outros testes foram realizados como a porta OR, AND, NAND e NOR. Circuitos como do somador binário completo foram evoluídos também. Uma tabela arbitrária com quatro entradas foi testada e seu resultado foi comparado com o resultado do ESPRESSO. Os mesmos resultados foram obtidos tanto para o Algoritmo Genético quanto para o ESPRESSO. É importante destacar que os testes foram feitos enquanto o sistema estava funcionando, ou seja, o circuito implementado na PAL mudava assim que a tabela-verdade era modificada e continuava seu funcionamento já com outra função Booleana.

Os resultados apresentados foram suficientes para validar a metodologia proposta. Porém, uma vez que a metodologia está provada, algumas preocupações surgem com o objetivo de melhorar o sistema proposto. Como já foi dito, uma das maiores dificuldades atuais do EHW é a escalabilidade. Como consta em [3], a aplicação de técnicas evolutivas para desenvolver circuitos grandes ainda está longe da realidade, mas o processo evolutivo não está totalmente limitado a circuitos pequenos e é possível evoluir circuitos complexos e práticos. Segundo [6],

Tabela 1. Comparação entre Algoritmo Genético e ESPRESSO

	Fitness	
Circuito	**Algoritmo Genético**	**ESPRESSO**
XOR	-100.0098	-100.0098
OR	-100.0098	-100.0098
AND	-100.0098	-100.0098
NAND	-100.0098	-100.0098
NOR	-100.0098	-100.0098
Somador	-100.0096	-100.0096
Tabela-verdade Arbitrária	-100.0097	-100.0097

EHW, como demonstram resultados experimentais, também é capaz de competir com classificadores convencionais atuais.

Como pode ser visto em [4], outra dificuldade encontrada no EHW existe quando algum operador do Algoritmo Genético afeta a conexão com a saída do circuito representado por algum indivíduo. Nesse caso, não há maneira de se medir o quão bom é o resto desse circuito já que não há uma maneira de testar o circuito. Para ilustrar esse problema, imagine que determinado indivíduo é o circuito ótimo. Agora, suponha que o operador de mutação alterou algo no indivíduo que afetou a conexão com a saída do circuito. Dessa forma, o indivíduo resultante está muito próximo do ótimo (apenas um valor foi alterado na mutação), porém a saída do circuito foi drasticamente alterada. Conclusão: uma avaliação que considera apenas a saída do circuito é insuficiente para avaliar o circuito de uma forma justa. E a questão que fica é a seguinte: Como um circuito pode ser avaliado de uma forma justa que considera o grau de corretude das partes do circuito e não somente a saída do circuito? Um outro problema que ocorre (também ilustrado acima) é quando um gene tem seu valor modificado por algum operador do Algoritmo Genético e isso altera drasticamente o desempenho do circuito, muitas vezes para pior. É muito difícil encontrar uma representação onde a mutação ou cruzamento destrutivos não ocorrem. Além desses problemas, deve-se encontrar um mapeamento direto entre a representação do circuito e o circuito propriamente dito se desejamos evoluir circuitos grandes e complexos.

6 Conclusões e Trabalhos Futuros

A metodologia apresentada neste trabalho mostra que é possível criar um sistema de Hardware Evolutivo com as tecnologias atuais disponíveis de maneira rápida e fácil, se comparado com o desenvolvimento de uma plataforma desde a soldagem do hardware até a criação dos softwares do sistema.

Essa metodologia viabiliza o estudo de várias técnicas de projeto de sistemas embarcados e estudo de Algoritmos Genéticos com características diferentes e até outros métodos de aprendizado, uma vez que basta substituir a técnica do módulo que projeta o circuito do sistema pela técnica desejada. Outros experimentos

podem ser realizados com outros tipos de hardware reconfiguráveis diferentes da PAL.

O Nios II Development Kit é uma boa alternativa para implementação de EHW, pois possui um conjunto de softwares que facilitam a criação dos sistemas. Vários processadores ou outros dispositivos de hardware podem ser acrescentados ao projeto via software com as características desejadas pelo projetista. O programa SoPC Builder automaticamente integra os componentes no hardware do sistema. As ferramentas de depuração e simulação permitem que o sistema seja validado antes de sua implementação de fato em alguma aplicação. Além disso, devido a uma camada de abstração de hardware (HAL, do inglês *Hardware Abstraction Layer*) existente nos sistemas Nios II, pode-se acessar dispositivos através de funções da biblioteca padrão do ANSI C, como $printf()$, $fopen()$, $fwrite()$, etc. De acordo com [15], a HAL fornece uma consistente interface a vários dispositivos como memória flash, subsistemas de arquivo, timers, DMAs, Ethernet, controlador do display LCD 16207, UART code, JTAG UART core. E se o projetista/programador quiser, pode acrescentar por conta própria mais dispositivos na HAL.

FPGA é uma tecnologia extremamente interessante para o desenvolvimento de protótipos, pois facilmente se consegue realizar alterações e testes de hardware e além disso com custo bem reduzido.

Os trabalhos futuros se concentram em melhorar a escalabilidade do EHW e otimizar o sistema proposto. Um dos aspectos que limitam a escalabilidade é o aumento do tempo de execução para circuitos complexos. Para melhorar o tempo de execução pode-se implementar o Algoritmo Genético em hardware, ou acrescentar instruções específicas para realizar os operadores genéticos dentro da unidade aritmética lógica do NIOS II através do SoPC Builder. Somente a implementação em hardware do *fitness*, de acordo com [16], pode oferecer uma significativa vantagem no tempo gasto pela execução da avaliação.

Técnicas mais eficientes de avaliação que evitam percorrer novamente a tabela-verdade várias vezes para efetuar o mesmo cálculo, como por exemplo Programação Dinâmica, ou a técnica mostrada em [7], podem ser usadas para diminuir o tempo gasto na avaliação. Para melhorar a eficiência da avaliação pode ser implementado um Algoritmo Genético multi-objetivo para otimizar os dois objetivos conflitantes da avaliação.

Por fim, devido a facilidade de aumento do número de processadores em um sistema Nios II, pode ser implementado um Algoritmo Genético paralelo para ser executado em vários processadores ao mesmo tempo.

Agradecimentos

Os autores deste trabalho agradecem especialmente à FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais - Brasil), que foi financiadora do projeto de pesquisa relacionado a este trabalho.

Referências

1. D.-W. Lee, C.-B. Ban, K.-B. Sim, H.-S. Seok, K.-J. Lee, and B.-T. Zhang, "Behavior evolution of autonomous mobile robot using genetic programming based on evolvable hardware," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, Nashville, TN, USA, 8-11 Oct. 2000, pp. 3835–3840.
2. H. D. Garis, "Genetic programming: Artificial nervous systems artificial embryos and embryological electronics."
3. A. P. Shanthi and R. Parthasarathi, "Practical and scalable evolution of digital circuits," *Applied Soft Computing*, 2008, in Press, Corrected Proof. [Online]. Available: http://www.sciencedirect.com/science/article/B6W86-4T9CCPD-1/2/75fb89ae05df548f839cb79ad371e0df
4. T. G. Haddow PC and R. P. van, *Evolvable hardware: pumping life into dead silicon. In: Kumar S and Bentley PJ. (eds) On Growth, Form and Computers, pp. 404-422*, 1st ed. Oxford, UK: Elsevier Limited, 2003.
5. Y. Zhang, S. L. Smith, and A. M. Tyrrell, "Digital circuit design using intrinsic evolvable hardware," in *Evolvable Hardware*. IEEE Computer Society, 2004, pp. 55–62.
6. K. Glette, T. Gruber, P. Kaufmann, J. Torresen, B. Sick, and M. Platzner, "Comparing evolvable hardware to conventional classifiers for electromyographic prosthetic hand control," in *AHS '08: Proceedings of the 2008 NASA/ESA Conference on Adaptive Hardware and Systems*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 32–39.
7. M. Salami and T. Hendtlass, "The fast evaluation strategy for evolvable hardware," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 139–162, 2005.
8. E. Stomeo, "A novel genetic algorithm for evolvable hardware," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 134–141.
9. A. C. (2007b), *Nios II Processor Reference Handbook*, 1st ed., A. Corporation, Ed. San Jose: Altera Corporation, 2007.
10. A. C. (2007), *Nios II Hardware Development Tutorial*, 1st ed., A. Corporation, Ed. San Jose: Altera Corporation, 2007.
11. A. C. (2005), *Nios II Development Board, Stratix II Edition*, 1st ed., A. Corporation, Ed. San Jose: Altera Corporation, 2005.
12. W. S. Lacerda, "Projeto e implementação de circuitos classificadores digitais com controle da generalização baseado na regra do vizinho-mais-próximo modificada," Ph.D. dissertation, Universidade Federal de Minas Gerais, Belo Horizonte - MG, 2006.
13. C. F. M. Toledo, "Problema conjunto de dimensionamento de lotes e programação da produção," Ph.D. dissertation, UNICAMP, Campinas - SP, 2005.
14. G. L. Soares, "Algoritmos genéticos: Estudo, novas técnicas e aplicações," Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte - MG, 1997.
15. A. C. (2008), *Nios II Software Developer's Handbook*, 1st ed., A. Corporation, Ed. San Jose: Altera Corporation, 2008.
16. A. Stoica, "Evolvable hardware: From on-chip circuit synthesis to evolvable space systems," in *ISMVL '00: Proceedings of the 30th IEEE International Symposium on Multiple-Valued Logic*. Washington, DC, USA: IEEE Computer Society, 2000, p. 161.