Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação.
Departamento de Engenharia de Computação
e Automação Industrial

Hardware Evolutivo Aplicado à Geração Automática de Controladores para Servo-Mecanismos

Tatiane Jesus de Campos

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Doutor em Engenharia Elétrica.

Banca Examinadora:
Prof. Dr. José Raimundo de Oliveira (Orientador) - DCA/FEEC/UNICAMP
Prof. Dr. Nobuo Oki - UNESP Ilha Solteira
Prof. Dr. Carlos Augusto Paiva da Silva Martins - PUC-MINAS
Prof. Dr. Fernando Gomide - DCA/FEEC/UNICAMP
Prof. Dr. Rafael Santos Mendes - DCA/FEEC/UNICAMP
Prof. Dr. Marconi Kolm Madrid - DSCE/FEEC/UNICAMP

Campinas, 13 de Julho de 2007

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Campos, Tatiane Jesus de
C157h Hardware evolutivo aplicado à geração automática de controladores para servo-mecanismos / Tatiane Jesus de Campos. --Campinas, SP: [s.n.], 2007.

Orientador: José Raimundo de Oliveira.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Hardware computacional. 2. Algoritmos evolutivos. 3. Sistema de controle digital. I. Oliveira, José Raimundo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: Evolvable hardware applied to automatic design of servomecanisms.
Palavras-chave em Inglês: Computational hardware, Evolutionary algorithms,Digital control systems
Área de concentração: Engenharia de Computação
Titulação: Doutor em Engenharia Elétrica
Banca examinadora: Nobuo Oki, Carlos Augusto Paiva da Silva Martins,Fernando Gomide, Rafael Santos Mendes e Marconi Kolm Madrid
Data da defesa: 11/05/2007
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato(a): Tatiane Jesus de Campos

Data da Defesa: 11 de maio de 2007
Título da Tese: "Hardware Evolutivo Aplicado à Geração Automática de Controladores para Servo-Mecanismos"

Prof. Dr. José Raimundo de Oliveira (Presidente): _____

Prof. Dr. Nobuo Oki: _____

Prof. Dr. Carlos Augusto Paiva da Silva Martins: _____

Prof. Dr. Fernando Antônio Campos Gomide: _____

Prof. Dr. Rafael Santos Mendes: _____

Prof. Dr. Marconi Kolm Madrid: _____

Das Utopias

Mario Quintana

Se as coisas são inatingíveis
Ora... Não é motivo para não querê-las...
Que tristes os caminhos,
Se não fora a presença distante das estrelas!

Agradecimentos

Inicialmente agradeço ao meu orientador Prof. Dr. José Raimundo de Oliveira, por ter acreditado tanto neste trabalho, pela paciência, dedicação e competência demonstradas.

Ao Mario por ser minha fonte de inspiração diária e pelas valiosas discussões, sugestões e força durante toda a etapa de desenvolvimento deste trabalho.

A meus pais, Nilton e Maria Eliza, anjos que me guiaram intensivamente no início da minha caminhada e ainda hoje são meu porto seguro, meus eternos agradecimentos.

A minha irmã Viviane por todo o seu amor e por ter compreendido a minha ausência em tantas ocasiões em que precisei me dedicar ao trabalho, saiba que sempre estarei ao seu lado.

A minha avó Clory (*in memorian*), que será sempre fonte de orgulho, admiração e um exemplo de vida a ser seguido.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Capes- que me concedeu uma bolsa durante a realização deste doutorado, fato este que muito contribuiu para viabilização desta tese.

Sumário

Lista de Figuras

Resumo

Na última década os algoritmos evolutivos vem sendo aplicados na síntese e projeto de circuitos eletrônicos criando uma nova área de pesquisa denominada Hardware Evolutivo. Esta tese propõe o uso de Hardware Evolutivo como uma ferramenta para geração automática de circuitos aplicados ao controle de um pêndulo amortecido não linear. Inicialmente um amplo estudo sobre a utilização de computação evolutiva aplicada à síntese de circuitos eletrônicos foi realizado, de modo a identificar os principais benefícios, motivações, aplicações e desafios da área de Hardware Evolutivo. A seguir foi realizado um estudo de caso com o objetivo de realizar uma comparação experimental dos principais pontos que afetam o desempenho de um sistema de Hardware Evolutivo na evolução de circuitos digitais básicos. Após a realização destas etapas foi desenvolvido um Hardware Evolutivo para controle de um pêndulo não linear. O objetivo desta implementação foi apresentar comparações de desempenho entre diferentes abordagens para projetos de controladores. O uso do Hardware Evolutivo para obtenção do controlador tem como objetivo modelar o comportamento não linear do sistema e sintetizá-lo em um circuito digital combinacional criando assim uma alternativa de projeto automático para sistemas de controle. A análise e simulação do pêndulo não linear demonstra que a aplicação desta nova técnica de projeto de *hardware* apresenta resultados promissores.

Abstract

In the last decade evolutionary algorithms application in electronic circuits synthesis have been intensively investigated, starting a new research area called Evolvable Hardware. This thesis considers the use of Evolvable Hardware as a tool for automatic design of circuits applied to the control of a nonlinear damped pendulum. Initially a study on the use of applied evolutionary algorithms to the synthesis of electronic circuits was carried out, in order to identify the main benefits, motivations, applications and challenges of the field of Evolvable Hardware. A case study was carried out with the objective to provide an experimental comparison of the main points that affect the performance of a system in the evolution of basic digital circuits. Finally a Evolvable Hardware controller unit for control a nonlinear damped pendulum was evolved. The objective of this implementation was to present performance comparisons between two different controllers designs. The analysis and simulation of nonlinear pendulum demonstrate that the application of this new technique of design provides excellent results.

Capítulo 1

Introdução

A computação evolutiva procura resolver problemas usando algoritmos de busca inspirados na evolução biológica. Estes algoritmos são conhecidos como algoritmos evolutivos e modelam os princípios da seleção, variação e herança, que são a base da teoria de Darwin. A teoria evolutiva de Darwin enfatiza a sobrevivência do mais apto em um ambiente dinâmico. As possibilidades de utilizar algoritmos evolutivos para automatizar projeto e síntese de hardware vem sendo explorada tanto no domínio digital quanto no domínio analógico Gordon e Bentley (2005).

A automação vem sendo usada na síntese de circuitos por muitos anos. Um projeto simples e tradicional de circuitos digitais envolve o desenvolvimento de um circuito aplicado a uma tecnologia específica com a utilização de técnicas de minimização, além de algumas regras de posicionamento e de roteamento (placement and routing rules). Atualmente projetos de hardware precisam sintetizar circuitos cada vez mais complexos, o que gera a necessidade de técnicas mais apuradas, capazes de resolver problemas combinatórios com alto grau de complexidade. Técnicas "inteligentes", tais como o recozimento simulado Sechen (1988) e as RNAs Yih e Mazumder (1990), são muito usadas para explorar estes espaços na procura de soluções e, em alguns casos, obter soluções otimizadas. O Hardware Evolutivo permite que a automação na produção de circuitos torne-se um passo adicional do projeto, possibilitando a geração de um circuito a partir de uma descrição comportamental, assim como a automação do processo da síntese do circuito Stoica *et al.* (2004).

Este capítulo apresenta os objetivos da tese, a contribuição da pesquisa realizada, a descrição e a organização do trabalho.

1.1 Objetivos do Trabalho

Diversos trabalhos sobre a utilização de computação evolutiva em otimização de circuitos eletrônicos têm sido divulgados desde a década de 80. No entanto, a idéia da realização da síntese de circuitos eletrônicos por algoritmos evolutivos é mais recente Zebulum (1999). Esta tese avalia a aplicabilidade e o desempenho da computação evolutiva no desenvolvimento de circuitos para problemas de sistemas de controle, além de realizar um estudo sobre os parâmetros de evolução para projetos digitais simples.

Um circuito detector de números primos foi evoluído com intuito de investigar o desempenho da evolução em relação ao tipo de genoma, taxa de crossover, a taxa de mutação, tamanho da população e número de gerações. A plataforma para realização dos experimentos consistiu de simuladores de circuitos e também de circuitos integrados reconfiguráveis.

O principal objetivo desta tese é a obtenção de novas ferramentas de projeto de circuitos para a área de controle baseadas unicamente em hardware evolutivo. Ultimamente técnicas de inteligência computacional, como computação evolutiva, vem sendo usadas na área de controle com diferentes intuitos. Este trabalho propõe o uso de hardware evolutivo como uma alternativa aos projetos usuais de controle tendo em vista que o controlador obtido não utiliza as técnicas tradicionais. O controlador obtido por hardware evolutivo neste trabalho é o projeto digital encontrado pelo processo evolutivo, sem levar em conta as teorias de projeto de controladore. Controladores obtidos por hardware evolutivo, onde o circuito controlador não foi projetado por métodos tradicionais de controle mas sim por um sistema de hardware evolutivo, e controladores Proporcionais e Proporcionais-Derivativos com ganhos sintonizados por meio de Algoritmos Genéticos (AGs), foram avaliados e comparados quando aplicados ao controle de posição de um pêndulo não linear.

Busca-se avaliar a capacidade do sistema evolutivo de se adaptar a problemas de controle. O objetivo final é a obtenção de um hardware evolutivo aplicado ao controle de um pêndulo amortecido não linear que apresente respostas competitivas frente aos controladores tradicionais. Com isso, deseja-se obter novos métodos de síntese de circuitos, voltados a área de controle, que apresentem menos dependência do conhecimento prévio de seres humanos do que as técnicas usuais de projeto com o intuito de minimizar o tempo de projeto e auxiliar na obtenção de uma solução inicial rápida e com um comportamento aceitável, o que torna-se viável através do uso de computação evolutiva.

1.2 Contribuições da Tese

Esta tese contribui para o desenvolvimento científico das áreas de engenharia eletrônica e ciência da computação. No caso de engenharia eletrônica, apresentam-se circuitos para projetos de controle sintetizados por hardware evolutivo que apresentam um bom desempenho quando comparado com as técnicas tradicionais de projeto, como os controladores PID (Proporcional Integral Derivativo). Além disso, uma contribuição futura para o avanço da pesquisa na área de controle está no fato de que muitos circuitos produzidos por hardware evolutivo são bastante diferentes dos convencionais, assim, a investigação de novas metodologias de projeto utilizadas por sistemas evolutivos pode gerar novas metodologias para síntese de sistemas de controle.

No caso de ciência de computação, particularmente no ramo de computação evolutiva, foram concebidas comparações entre diversos parâmetros de evolução como diferentes codificações, valores de crossover e mutação, verificando a influência de cada um no processo evolutivo.

1.3 Organização da Dissertação

Esta tese possui mais quatro capítulos, cujos conteúdos são descritos a seguir. O capítulo 2 apresenta os principais conceitos abordados pela área. Inicialmente são apresentadas as principais características e metodologias de projeto da área de síntese de hardware. A seguir é apresentado um resumo da área de computação evolutiva, abrangendo algoritmos genéticos Holland (1975), programação genética Koza (1992), programação evolutiva Fogel *et al.* (1966) e estratégias evolutivas Rechenberg (1973). Descreve-se em detalhes os principais componentes de algoritmos evolutivos como a codificação, avaliação e operadores de seleção e reprodução, com ênfase em algoritmos genéticos. Por fim é apresentada a área de hardware evolutivo que tem como base os conceitos anteriormente abordados. Nesta seção são apresentados os principais conceitos, benefícios e motivações da área assim como as principais aplicações, suas tendências e desafios.

O capítulo 3 introduz as considerações práticas e apresenta de forma detalhada os conceitos e fundamentos necessários ao desenvolvimento de sistemas de hardware evolutivo, tais como a descrição de genomas, funções de avaliação e ope-

radores genéticos. Um estudo de caso é apresentado para avaliar como as decisões de projeto influenciam o desempenho de um sistema EHW (Evolvable Hardware) e a evolução de circuitos digitais básicos.

O capítulo 4 apresenta a aplicação desenvolvida e as comparações de desempenho entre diferentes abordagens para projetos de controladores. Inicialmente apresenta-se a aplicação descrevendo seu modelo, a seguir é apresentado o controlador tradicional e o método de sintonização dos parâmetros utilizado. Na seção seguinte será apresentado o hardware evolutivo que realizará o controle evolutivo em substituição ao controle tradicional. Por fim é apresentado o sistema de controle usado, detalhes de implementação e evolução e as comparações realizadas.

O capítulo 5 conclui a tese e apresenta uma discussão sobre a continuidade do trabalho. Pode-se observar ao final do trabalho que o controlador obtido com o hardware evolutivo apresentou resultados promissores quando comparado ao método clássico.

Capítulo 2

Conceitos Fundamentais

2.1 Introdução

Nos últimos 40 anos a indústria de semicondutores obteve altas taxas de crescimento tanto no volume de produção quanto na complexidade do sistema de *hardware*. Em conseqüência deste crescimento algumas metodologias de projeto de sistemas tornaram-se obsoletas enquanto outras, ainda atuais, não conseguem manter uma taxa de crescimento equivalente. Esta defasagem acarreta dois problemas freqüentes na área de projeto de hardware: complexidade e escalabilidade das técnicas de projeto.

O primeiro está relacionado com a dificuldade encontrada em interconectar milhões de componentes de forma a atender a um conjunto de especificações e/ou restrições. O segundo está relacionado aos avanços das tecnologias de fabricação que, conforme atesta a chamada lei de Moore, dobra a densidade média dos Circuitos Integrados (CIs) a cada 18 meses, Moore (1956), considerada ainda hoje uma taxa estável, exigindo que as técnicas voltadas ao projeto de hardware manipulem instâncias cada vez maiores do mesmo problema. Além disso, o tempo durante o qual um produto eletrônico gera lucros significativos foi reduzido de 3 a 5 anos, no final dos anos 80, para 1 ano ou menos na atualidade. Portanto o tempo que um produto necessita para chegar ao mercado (*time to market*) passa a ser tão importante, se não mais, do que outros parâmetros tradicionalmente considerados, como desempenho, ou mesmo os custos final e de produção, Schaller (1997).

Esta alta taxa de crescimento, somada a redução da vida útil do produto, aos avanços na tecnologia FPGA (*Field Programmable Gate Array*), a disponibilidade de CIs com velocidade, densidade e facilidade de reconfiguração e as novas exigências do mercado, tal como a necessidade de tolerância a falhas, exigem um maior conhecimento das tecnologias e métodos de projetos existentes, além de, em alguns casos, exigir o desenvolvimento de novas metodologias de projeto automático de hardware.

A necessidade de desenvolvimento de novas metodologias de projeto levou a aplicação de técnicas advindas da natureza ao projeto de hardware, um exemplo é a técnica de Recozimento Simulado (Simulated Annealing), Tessier (1999), usada na engenharia eletrônica em muitos algoritmos de particionamento de circuitos. A idéia de criar algoritmos baseados em processos observados na natureza tornou-se comum nos últimos anos, principalmente no desenvolvimento de softwares. O interesse em usar sistemas bio-inspirados levou à aplicação de técnicas evolutivas ao projeto de circuitos eletrônicos permitindo realizá-lo automaticamente ou, no mínimo, auxiliando em sua execução. Esta técnica é conhecida como *Hardware Evolutivo*Higuchi *et al.* (1995).

Muitas ferramentas modernas de Projeto Automático de Circuitos Eletrônicos (*Electronic Design Automation* - EDA) usam técnicas bio-inspiradas em seus algoritmos, e a pesquisa no uso da evolução para esta finalidade é substancial, Cohoon *et al.* (2003), Rothlauf *et al.* (2006). Embora o projeto evolutivo de circuitos eletrônicos possua forte relação com a tecnologia, a maior parte das decisões referentes ao projeto de circuito e sua otimização durante o processo de síntese, ainda estão sob o domínio do projetista humano. Recentemente o desenvolvimento de técnicas evolutivas para projetos com tecnologia VLSI vem despertando interesse significativo, buscando implementar circuitos equivalentes ou até mesmo superiores aos desenvolvidos pelo ser humano, Gordon e Bentley (2005).

2.2 Síntese de Hardware

Projetos de sistemas digitais podem ser definidos como a transformação de uma descrição inicial do sistema, também chamada especificação, em uma descrição final, através de uma sucessão de etapas que envolvem diferentes níveis de abstração onde a principal diferença está no fato desta última conter todas as informações necessárias a sua fabricação, Calazans (1998). O projeto de hardware pode ser

dividido em etapas as quais podem ser classificadas como síntese, extração, validação e otimização, Suzim (1988).

A operação de síntese é a tradução de uma descrição em um dado nível para uma descrição de nível inferior. Isto é realizado através do acréscimo de informação que permite a criação de uma descrição menos abstrata. A operação de extração, ao contrário da síntese, gera uma descrição mais abstrata, com menos detalhes e características de implementação. As operações de validação e otimização ocorrem sobre descrições de um mesmo nível, Figura 2.1. A validação consiste em verificar se o circuito sintetizado realiza o comportamento especificado, e a otimização busca transformar a descrição original em outra equivalente que melhor satisfaça algum critério pré estabelecido.

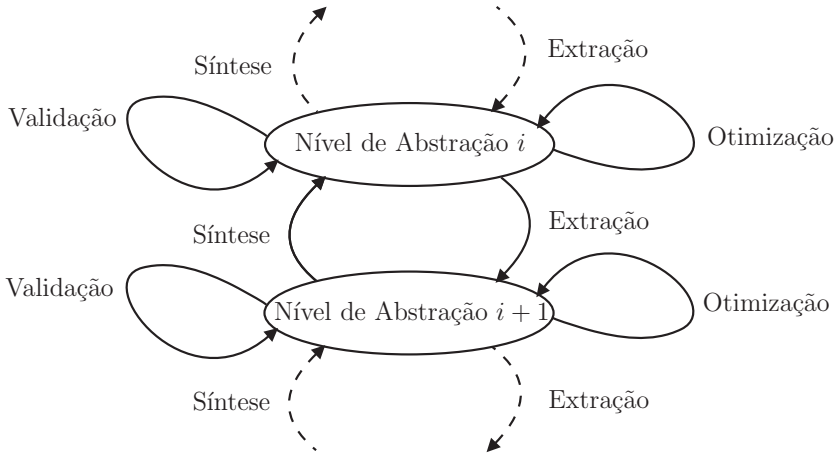


Figura 2.1: Modelo Proposto Por Suzim Para Representar o Processo de Projeto de Sistemas Digitais

Existem diversas metodologias para executar um projeto de hardware, sendo a mais comum o Diagrama Y. Este diagrama, proposto originalmente por Gajski e Kuhn (1983), tem como finalidade estratificar o processo de projeto de sistemas digitais em diferentes níveis de abstração e domínios de descrição. Nesta representação, os círculos correspondem aos níveis de abstração, enquanto que os segmentos de reta correspondem aos domínios de descrição, Figura 2.2.

Os sistemas digitais em geral podem ser classificados em quatro níveis de abstração diferentes. Descrições que se encontram em um mesmo nível contém a mesma quantidade de informação. Os Níveis de abstração são:

- Nível elétrico: como exemplo pode-se citar os transistores e capacitores;

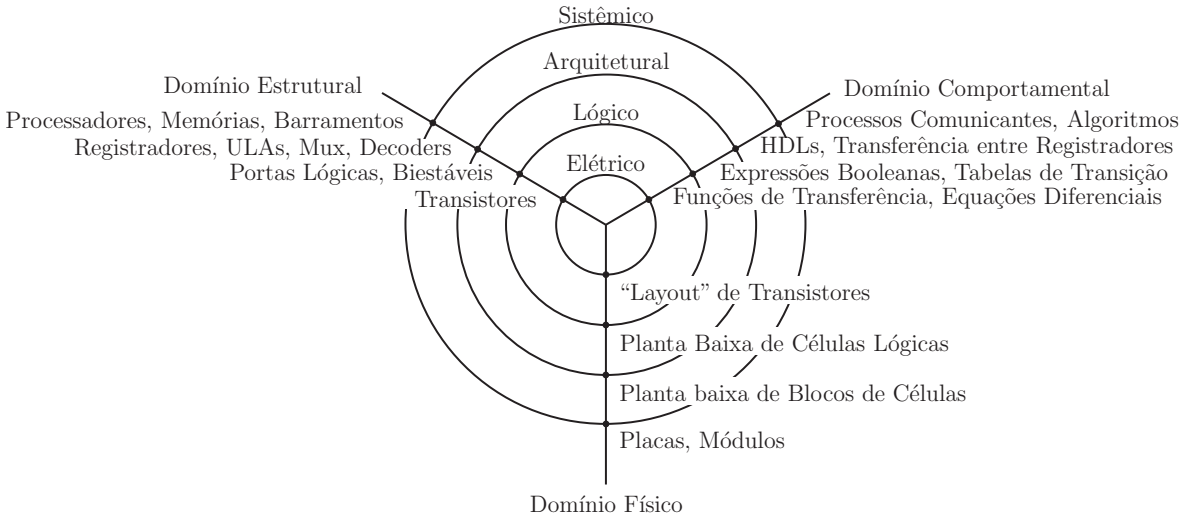


Figura 2.2: Diagrama Y, Gajski and Kuhn(1983)

- nível lógico: onde aparecem as portas lógicas, os flip-flops e equações booleanas;
- o nível arquitetural: como exemplo deste nível pode-se citar as ULAs e os multiplexadores;
- nível sistêmico : onde aparecem os processadores e memórias.

Além dos níveis de abstração é possível classificar as descrições de acordo com o tipo de informação que elas carregam. Neste contexto as descrições podem ser classificadas em três níveis diferentes. Estes níveis são do tipo físico, que contém informações sobre os componentes/módulos; nível comportamental, que contém informação sobre o comportamento do sistema, e por último o nível estrutural, que contém informação sobre como interconectar os vários blocos.

O centro do diagrama corresponde à descrição que contém toda a informação necessária à fabricação do sistema, também chamada descrição final. Cada intersecção de um círculo com um segmento representa um tipo de descrição diferente.

A automação das etapas de descrições acima desempenha um papel primordial na redução do tempo de projeto. Entretanto, à medida que a escala de integração aumenta cresce a dificuldade de se obter projetos sem erros, Calazans (1998). Assim, ferramentas para automatizar o processo de projeto devem estar em constante evolução.

2.3 Computação Evolutiva

A Computação Evolutiva (CE) é uma área de pesquisa relativamente nova. O termo Computação Evolutiva foi proposto recentemente, 1991 segundo Bäck *et al.* (2000*a*), e se refere ao estudo de heurísticas baseadas nos princípios da evolução natural. As principais heurísticas que compõe a CE são os Algoritmos Genéticos (AG), as Estratégias Evolutivas (EE), a Programação Evolutiva (PE), a Programação Genética (PG) e os Sistemas Classificadores (SC). Todas estas heurísticas possuem algum princípio baseado na evolução natural, como reprodução, variação aleatória, seleção, indivíduos e populações, Bäck *et al.* (2000*a*), Bäck *et al.* (2000*b*) e Eiben e Smith (2003).

A Computação Evolutiva consiste, basicamente, em analisar o comportamento da evolução natural e implementar suas principais características computacionalmente. Todas as heurísticas citadas anteriormente, embora tenham origens distintas, possuem em comum a idéia de que, dada uma população de indivíduos, a pressão exercida pelo meio ambiente sobre os mesmos (pressão evolutiva) leva à seleção natural (sobrevivência do mais adaptado), que por sua vez, produz um aumento da aptidão da população como um todo. O processo de seleção possui uma função com capacidade para medir a aptidão de cada indivíduo. Os indivíduos mais adaptados possuem maior probabilidade de serem selecionados para participar do processo de reprodução, e deste modo perpetuar suas características através de seus descendentes, aumentando a qualidade da população a cada geração. A reprodução ocorre através de dois operadores genéticos, crossover e mutação. A estrutura geral de um algoritmo evolutivo pode ser visualizada, de forma simplificada, na Figura 2.3.

Neste processo podem ser identificadas duas operações que formam o conceito principal dos sistemas evolutivos:

- Operadores genéticos, como crossover e mutação, introduzem diversidade genética à população, permitindo o surgimento de novos e diferentes indivíduos.
- A seleção age de modo a aumentar a qualidade da população, como um resposta a pressão evolutiva.

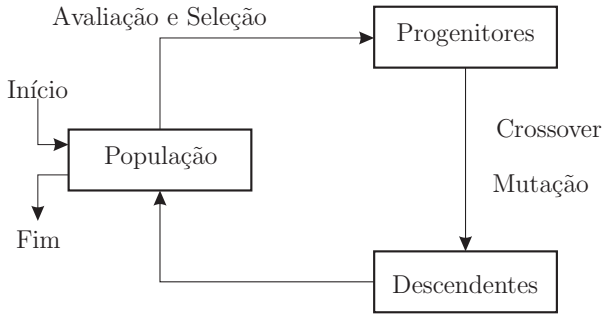


Figura 2.3: Estrutura Geral de um Algoritmo Evolutivo, Michalewicz(1996)

2.3.1 Algoritmos Genéticos

Os Algoritmos Genéticos (AGs) são considerados a classe mais usual dos Algoritmos Evolutivos e foram propostos inicialmente por Holland (1975). São métodos de busca que trabalham com uma população de indivíduos, formada inicialmente de modo aleatório, que representam as possíveis soluções para um determinado problema (soluções candidatas), e usam operadores de crossover (recombinação), mutação e seleção. Cada indivíduo é testado e avaliado para receber uma nota que reflete sua aptidão a determinado ambiente o que permite ao AG criar sucessivas gerações de indivíduos, cada vez mais adaptadas a este ambiente.

O problema de otimização de uma função de n variáveis, $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, consiste em encontrar um elemento $\mathbf{x} \in \mathbb{R}^n$ tal que a função $f(\mathbf{x})$ seja maximizada ou minimizada. A função que determina a aptidão de um determinado individuo é fortemente dependente da função objetivo $f(\mathbf{x})$ e é chamada de função de *fitness*. A função de *fitness* avalia cada indivíduo, medindo quantitativamente o nível de adaptação do mesmo. Quanto maior o valor do *fitness* de um cromossomo melhor é a solução codificada por este cromossomo. A busca é guiada apenas pelo valor de aptidão associado a cada indivíduo na população.

O algoritmo genético clássico usa a codificação binária para representar um ponto no espaço de busca. Esta codificação binária consiste de um vetor de elementos binários que representa um ponto (possível solução), conforme pode ser visto na Figura 2.4. Cada vetor da codificação é chamado de cromossomo e o espaço dos cromossomos é chamado de espaço do genótipo (*genotype space*).

A codificação binária, embora muito útil, não é a única representação possível para codificar uma determinada solução. Em determinados problemas com alta

$$\mathbf{x} = (x_1, x_2, \cdots, x_n) = \underbrace{010\cdots00}_{x_1}\underbrace{101\cdots01}_{x_2}\cdots\underbrace{111\cdots10}_{x_n}$$

Figura 2.4: Codificação Binária das Variáveis de Busca.

precisão numérica e grande número de variáveis o uso desta codificação pode ser proibitivo, Michalewicz (1996). Nestes casos a representação inteira ou em ponto flutuante pode ser usada, modificando consideravelmente o espaço de genótipo. Na codificação em ponto flutuante, por exemplo, cada cromossomo é codificado como um vetor de números reais, com mesma dimensão do vetor solução.

$$\mathbf{x} = (x_1, x_2, \cdots, x_n) \quad x_i \in \mathbb{R}, \quad i = 1, 2, \cdots, n$$

Figura 2.5: Codificação em Ponto Flutuante das Variáveis de Busca.

Os cromossomos no espaço do genótipo são objetos matemáticos abstratos nos quais os operadores genéticos agem. Uma função de decodificação transforma o cromossomo no espaço de genótipo em um ponto no espaço de fenótipo, que é o conjunto formado pelo domínio $\mathbb{R}^n$ da função $f(\mathbf{x})$, também denominado espaço de busca.

O algoritmo genético é um processo iterativo onde, a cada iteração, as informações geradas pela população atual são usadas para direcionar a busca para regiões mais promissoras do espaço de busca. Desse modo gerar uma nova população, baseada nestas informações, é um ponto chave dos algoritmos genéticos. Na sua forma mais simples, o algoritmo genético utiliza três operadores genéticos para criar esta nova população, sendo eles: crossover, mutação e seleção.

A função do operador de seleção é de selecionar cromossomos (cromossomos pais) para o processo de reprodução, a probabilidade de seleção de um determinado crossover é proporcional a seu *fitness*. O crossover é uma operação genética que permite criar novos descendentes (cromossomos filhos), a partir de cromossomos existentes (cromossomos pais), pela troca de informações contidas nos vetores binários, criando uma recombinação das características dos pais de modo que os filhos herdem estas características.

O processo de crossover é gerado pela escolha aleatória dos indivíduos e do ponto de corte seguido pela troca de material genético. O número de cortes geralmente varia entre 1 (crossover de um ponto) e 2 (crossover de dois pontos).

O operador de mutação introduz diversidade genética à população, alte-

Pais	Ruptura	Troca	Filhos
01001011	010010-11	010010-10	01001010
11110010	111100-10	111100-11	11110011

(a)

Pais	Ruptura	Troca	Filhos
01001011	010-010-11	010-100-11	01010011
11110010	111-100-10	111-010-10	11101010

(b)

Figura 2.6: Exemplo de Operação: (a) Crossover de um Ponto. (b) Crossover de dois Pontos.

rando arbitrariamente um ou mais gens do cromossomo, permitindo a introdução de novos elementos na população. Desta forma, a mutação ajuda a solucionar o problema do confinamento a mínimos locais na otimização, pois promove alterações que direcionam a pesquisa para outros locais da superfície de resposta assegurando que a probabilidade de se chegar a qualquer ponto do espaço de busca nunca será nula.

2.3.2 Programação Genética

Programação Genética (PG) é considerada uma extensão dos algoritmos genéticos, desenvolvida por Koza (1992). Nesta técnica a estrutura de dados onde ocorre a adaptação é representada por programas computacionais. Devido a sua representação particular os operadores de reprodução e a avaliação são diferenciados.

Em programação genética um indivíduo é representado por uma expressão simbólica, definida pelo projetista. A resolução de um problema por PG consiste em explorar o espaço de soluções através de possíveis combinações entre diferentes expressões simbólicas. As expressões são codificadas como estruturas de dados do tipo árvore, também chamada de programa computacional. A população é então formada por um conjunto de árvores, apresentando comprimento variável e subdividida em nós, cada uma representando um programa (indivíduo).

Estas árvores possuem funções e terminais, que determinam suas características e definem seu comportamento no ambiente. Cada função é um nó da árvore, e cada terminal é uma folha. As funções podem ser, por exemplo, operações lógicas

ou matemáticas, e os terminais podem ser variáveis, constantes, ou funções que não recebem argumentos. Um exemplo de um indivíduo representado por uma estrutura do tipo árvore pode ser visto na Figura 2.7.

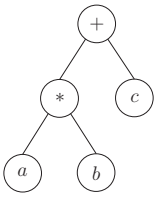
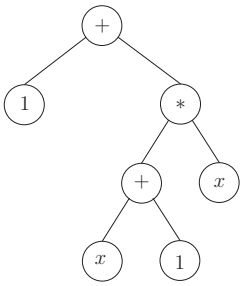


Figura 2.7: Exemplo de Indivíduo

Como cada cromossomo, em PG, é a representação de uma expressão simbólica codificada em uma estrutura do tipo árvore, esta pode ser percorrida de forma pré-fixada, central ou pós-fixada. A Figura 2.8 mostra as representações possíveis.



Pré-fixado: $(+1(*(+x1)x))$
Central: $(1 + ((x + 1) * x))$
Pós-fixado: $(1((x1+)x*)+)$

Figura 2.8: Exemplo de Indivíduo

A primeira etapa no desenvolvimento da programação genética consiste na definição de um conjunto de funções e terminais adequados ao problema e na sua codificação, definindo o espaço do genótipo.

A partir da codificação gera-se a população inicial que será avaliada através da função de *fitness*. Baseado nos valores obtidos realiza-se a seleção e inicia-se o processo de reprodução. A função de avaliação será definida observando-se o problema em questão, que neste trabalho é a codificação de circuitos digitais. Como método de reprodução tem-se o crossover e a mutação. No crossover são escolhidos dois indivíduos (pais) bem adaptados, dos quais são criados dois descendentes que vão estar na próxima geração.

Existem tipos diferentes de crossover para os problemas de programação genética pois a troca de material genético entre os pais ocasiona a troca de nós ou de ramos das árvores, gerando filhos de tipos e tamanhos diferentes. Nestes

casos, dependendo da codificação utilizada, é necessário verificar a consistências das respostas geradas. A figura 2.9 mostra um exemplo do método descrito.

A mutação não é um operador comumente utilizado em PG. A mutação insere a variabilidade genética na população e é uma alteração aleatória em um determinado indivíduo. Pode-se ter diferentes tipos de mutação pois pode-se aplicar a mutação alterando um nó ou uma folha da estrutura.

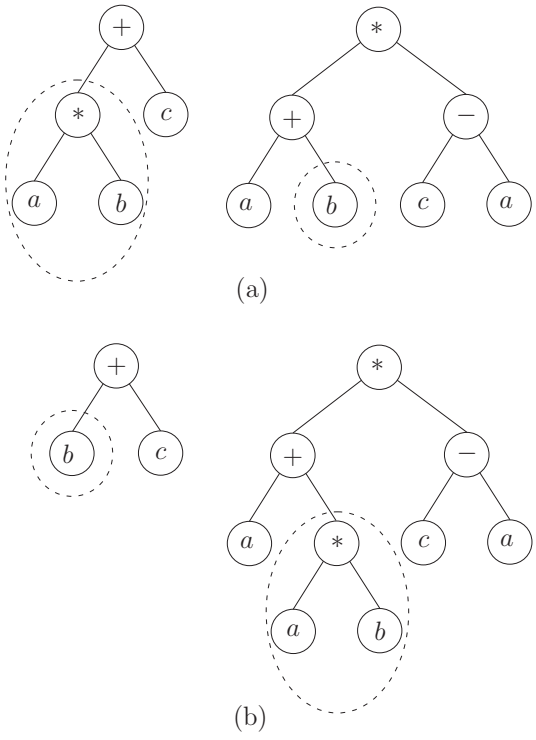


Figura 2.9: Exemplo de Crossover

A programação genética permite a utilização de genomas de comprimentos variáveis, que podem ser expressos por estruturas de dados hierárquicas, esta abordagem tem sido utilizada em diversas áreas do conhecimento, tais como: desenvolvimento de filtros, equações não-lineares de sistemas caóticos, indução de árvore de decisão, detecção de características, otimização e evolução de redes neurais artificiais. Devido as suas características, apresenta-se como um método eficiente de evolução de sistemas de hardware, Koza *et al.* (2000).

2.3.3 Estratégia Evolutiva

As estratégias evolutivas são técnicas de computação evolutiva que tem por objetivo solucionar problemas de otimização de parâmetros que não podem ser resolvidos por métodos convencionais por serem muito complexos ou de alto custo computacional. Esta técnica foi desenvolvida no início dos anos 60 por Rechenberg e Schwefel, Eiben e Smith (2003), na universidade de Berlim, Alemanha.

Inicialmente esta técnica era um processo evolutivo baseado em uma população de somente um indivíduo e um único operador genético: mutação. Assim, um único filho era gerado a partir de um único pai, através da mutação deste, e o melhor indivíduo entre eles permanecia na população enquanto o outro era eliminado do processo.

Este processo é denominado $(1 + 1) - EE$, onde a principal idéia é a representação de um indivíduo como um par de vetores reais na forma $\mathbf{v} = (\mathbf{x}, \sigma)$, onde $\mathbf{x} \in \mathbb{R}^n$, representa o ponto de busca no espaço, e $\sigma \in \mathbb{R}^n$, o vetor de desvio padrão associado, Michalewicz (1996). O processo de geração de um novo indivíduo (solução) é realizado pela mutação de x, seguindo a regra:

$$x^{t+1} = x^t + N(0, \sigma)$$

Onde $N(0, \sigma)$ é uma distribuição gaussiana com média zero e desvio padrão σ. A $(1+1)-EE$ possui uma convergência lenta, além do problema de ser suscetível a estagnar em ótimos locais devido a busca ponto a ponto.

Outras versões desta técnica foram desenvolvidas com o objetivo de solucionar os problemas apresentados acima. Estas estratégias, denominadas EEs com multimembros, são assim chamadas pois o número de ancestrais (pais), μ, e o número de descendentes (filhos), λ, possuem tamanho maior que 1.

A segunda EE, proposta por Rechenberg, foi a estratégia $(\mu + 1) - EE$. Esta técnica apresenta μ pais e somente um filho. Neste processo são selecionados aleatoriamente dois pais pertencentes a população para a geração de um filho. Esta geração ocorre através do operador de mutação, descrito anteriormente, e um operando de recombinação, similar ao crossover uniforme presente nos algoritmos genéticos, Michalewicz (1996). Os três indivíduos competem e o pior entre eles é eliminado.

As EEs foram aperfeiçoadas tendo-se atualmente dois tipos principais: $EE-(\mu+\lambda)$ e $EE-(\mu,\lambda)$, Michalewicz (1996), Bäck *et al.* (2000*a*) e Eiben e Smith (2003). No método $EE-(\mu+\lambda)$, a cada geração, são gerados λ filhos através dos operadores de mutação e dos operadores de recombinação similares aos aplicados em representação real nos algoritmos genéticos, estes novos filhos competem com os pais e somente os melhores indivíduos sobrevivem, ou seja, a população $\mu+\lambda$ é posteriormente reduzida para μ indivíduos. Isto ocorre para manter o número de indivíduos na população constante.

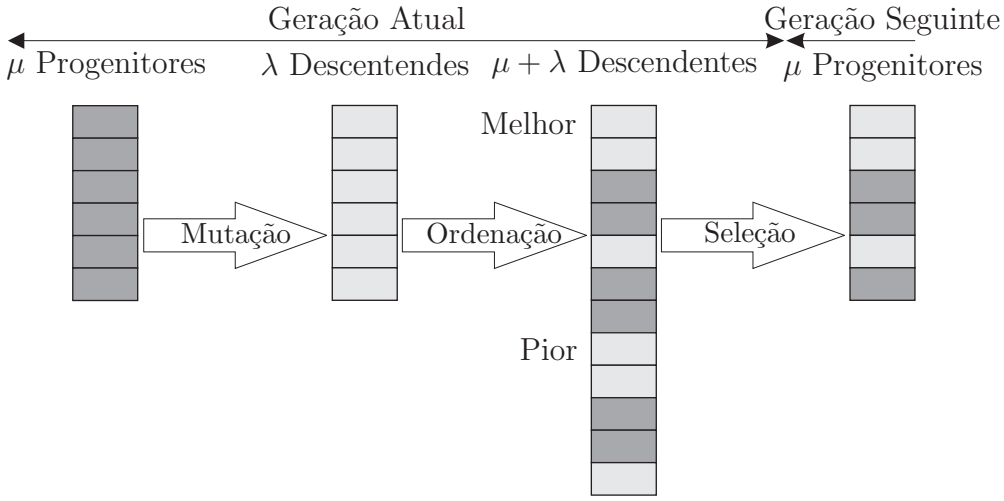


Figura 2.10: Aspecto Geral da Estratégia Evolutiva $(\mu+\lambda)$

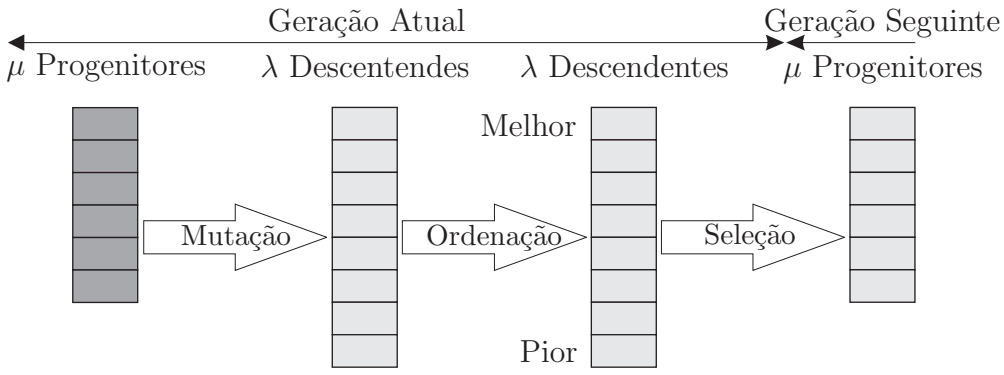


Figura 2.11: Aspecto Geral da Estratégia Evolutiva (μ,λ)

O outro método desenvolvido por Schwefel é o $EE-(\mu,\lambda)$. Neste método a seleção ocorre somente sobre os descendentes, ou seja, o período de vida de cada indivíduo está restrito a uma geração. É importante salientar que nas versões atuais a descendência é obtida submetendo-se os indivíduos da geração a dois operadores: crossover e mutação, conforme descrito anteriormente. Porém outros tipos de

operadores ou variações destes podem ser aplicados no processo evolutivo.

Uma idéia importante introduzida nos algoritmos mais recentes é a adaptação *online*, também chamada auto-adaptação, dos parâmetros da estratégia durante o processo evolutivo, através da introdução dos mesmos na representação genética dos indivíduos. A expressão *parâmetros da estratégia* refere-se aos parâmetros que controlam o processo evolutivo de busca, como taxas de mutação, desvios padrões das mutações, probabilidades de recombinação, entre outros. A idéia consiste na evolução dos parâmetros da estratégia, em adição à evolução dos atributos da estrutura de dados. A *auto-adaptação* de *parâmetros da estratégia* é um dos pontos chaves do sucesso das estratégias evolutivas e da programação evolutiva, pois ambas utilizam processos evolutivos para otimizar o *espaço de atributos* e o *espaço de parâmetros*, Castro e von Zuben (2004).

2.3.4 Programação Evolutiva

A programação evolutiva foi proposta por Fogel, Fogel *et al.* (1966), em meados da década de 60. Esta técnica foi originalmente desenvolvida para simular a evolução como um processo de aprendizado buscando uma forma alternativa de inteligência artificial. Inteligência, nesta época, era vista como a capacidade de descobrir um determinado ambiente e responder apropriadamente a ele, Michalewicz (1996). A proposta original trata da predição de comportamento de máquina de estado finito, Eiben e Smith (2003), mas o enfoque da PE se adapta a uma variedade de problemas.

Nesta técnica uma população inicial de soluções está exposta ao ambiente, a seleção é determinística, ou seja, cada indivíduo gera um único descendente através da mutação e não existe operador de recombinação. Assim como ocorre na EE, a programação evolutiva cria os descendentes e após este processo é que ocorre a seleção dos indivíduos que participarão da próxima geração. A quantidade de mutações pode ser escolhida de acordo com uma distribuição de probabilidade ou pode ser fixa.

Em 1992 Fogel, Bäck *et al.* (2000*a*), apresentou o conceito de programação metaevolucionária, ou Meta-PE, para operar com vetores reais de atributos, sujeitos a mutações com distribuição normal, ou seja, um vetor de variâncias substitui o valor padrão da taxa de mutação, similarmente ao que ocorre na auto adaptação das estratégias evolutivas. Sendo que a variância fará parte da estrutura do indivíduo

e, portanto, poderá também sofrer mutação. Esta técnica permite a evolução dos operadores genéticos, parâmetros dos algoritmos, entre outros, juntamente com a estrutura de dado.

Capítulo 3

Hardware Evolutivo

3.1 Introdução

O projeto de circuitos lógicos é uma área de pesquisa que desperta grande interesse devido a crescente complexidade dos circuitos e especialmente pela grande dificuldade de otimização. Inicialmente a álgebra booleana foi utilizada como ferramenta mais comum no projeto de sistemas digitais, posteriormente as técnicas dos Mapas de Karnaugh e Quine-McCluskey, Karnaugh (1953) e McCluskey (1956), passaram a ser usadas no desenvolvimento e simplificação de projetos Bonatti e Madureira (1995). Atualmente várias heurísticas foram incorporadas como ferramenta de projeto e simplificação de circuitos, entre elas a computação evolutiva, permitindo o surgimento de uma nova área de pesquisa denominada hardware evolutivo (EHW) de Garis (1993).

No Hardware Evolutivo aplica-se um algoritmo evolutivo de busca, usualmente o algoritmo genético, que opera sobre uma população de indivíduos. Geralmente estes indivíduos são *strings* binárias de tamanho fixo e pertencem a uma população que mantém seu tamanho constante. Cada cromossomo codifica ou representa um circuito eletrônico. O conjunto de todas as combinações possíveis de valores de parâmetros define o espaço de busca do algoritmo. Cada conjunto de parâmetros no espaço de busca codifica uma descrição de circuito no espaço de solução. Os cromossomos que representam os circuitos gerados são o genótipo enquanto o circuito é chamado de fenótipo.

Na fase inicial do processo os bits de cada cromossomo são inicializados com valores aleatórios, os cromossomos são avaliados através da simulação do modelo do circuito (evolução extrínseca) ou pela implementação em hardware reconfigurável (evolução intrínseca). A aptidão do circuito em executar uma determinada tarefa é medida aplicando-se um conjunto de valores de teste e avaliando-se o comportamento dos sinais de saída do circuito. O operador de seleção, presente no processo evolutivo, seleciona,de forma probabilística, os cromossomos que serão a base para a próxima geração. Os cromossomos com maior aptidão terão maior probabilidade de seleção. Aplica-se então os operadores de reprodução, crossover e mutação, sobre todos os indivíduos selecionados, gerando-se uma nova população. Este processo se repete de modo cíclico até que um circuito responda adequadamente ou até que um número pré-determinado de gerações seja alcançado, Zebulum *et al.* (1996).

Esta seção apresenta as principais abordagens encontradas durante a revisão bibliográfica e que serviram de inspiração para o desenvolvimento deste trabalho. São apresentados os benefícios, as tendências e aplicações do hardware evolutivo, assim como os principais conceitos, motivações e desafios encontrados na área.

3.2 Benefícios, Tendências e Aplicações

3.2.1 Benefícios

A automação vem sendo usada na síntese de circuitos por muitos anos. Um projeto simples e tradicional de circuitos digitais envolve o desenvolvimento de um circuito aplicado a uma tecnologia específica com a utilização de técnicas de minimização, além de algumas regras de posicionamento e de roteamento. Atualmente projetos de hardware exigem a capacidade de sintetizar circuitos cada vez mais complexos, gerando a necessidade de técnicas mais apuradas, capazes de resolver problemas combinatórios com alto grau de complexidade.

O hardware evolutivo busca automatizar o processo de projeto e síntese de circuitos possibilitando a geração de um circuito a partir de uma decrição comportamental. A especificação comportamental apresentada ao sistema evolutivo pode ser simples como um conjunto de sinais de entrada e saída, ou mais complexa, como condições do ambiente ou dados obtidos por simulações. Os métodos de especificação do circuito afetam sua funcionalidade e são, atualmente, fontes de pesquisa.

Usar a evolução para projetar circuitos traz benefícios importantes ao projeto permitindo sua automação e a inovação de projetos para uma escala crescente de aplicações. Nas aplicações onde uma especificação comportamental completa está disponível, o hardware evolutivo pode eliminar a necessidade de um projetista, ou reduzir o tempo de projeto, reduzindo custos. Isto é particularmente útil quando os custos do projeto são uma proporção significativa do custo total, por exemplo no desenvolvimento de hardware produzido em baixos volumes. Hardware evolutivo permite também o desenvolvimento de projetos individuais. Muitas aplicações médicas não tem adequadas soluções em hardware devido ao custo de personalização do projeto. O EHW fornece soluções rápidas e baratas a tais aplicações. Como exemplo pode-se citar um sistema desenvolvido para controlar uma mão protética reconhecendo padrões de sinais no braço de um determinado usuário, Kajitani *et al.* (1999) e Torresen (2001). As soluções são inteiramente baseadas em hardware com lógica reconfigurável.

A evolução pode também ser usada para reduzir custos de fabricação em escalas maiores, otimizando os circuitos que possuem especificações alteradas devido às variações durante o processo de fabricação. Por exemplo, em Murakawa *et al.* (1998) as variações na freqüência de corte de determinados filtros foram corrigidas usando a evolução para controlar a impedância de saída de amplificadores. O hardware evolutivo também se tornou uma ferramenta útil em projeto de circuitos analógicos, onde os parâmetros dos componentes podem variar muito, além de ser útil para os projetos onde a potência e o tamanho dos componentes é importante, minimizando consumo, volume e custos.

Além da utilização do hardware evolutivo para o desenvolvimento de projetos automáticos de baixo custo é possível aplicá-lo como solução em projetos que possuem especificação insuficiente. Para alguns problemas é mais simples especificar uma descrição comportamental do que obter, de modo prático, sua funcionalidade. Em desenvolvimento de software, problemas com estas características estão sendo tratados, há alguns anos, com o uso de evolução ou com o uso de redes neurais artificiais. Situações que apresentam ruído, como reconhecimento de padrão em ambiente ruidoso, podem ser citados como um exemplos deste caso, Rumelhart *et al.* (1994). Estas duas técnicas possuem similaridades e foram comparadas em Yao e Higuchi (1999).

Quando o problema necessita de processamento rápido sem comprometer sua tratabilidade, o hardware evolutivo é uma solução apropriada pois tem a vantagem da rapidez do hardware e muitas vezes apresenta uma solução mais fácil de ser

compreendida do que a obtida por Redes Neurais Artificiais, RNAs.

Problemas de reconhecimento de padrões são aplicações clássicas de redes neurais artificiais, mas sistemas de hardware evolutivo podem tratar estes problemas gerando soluções semelhantes as obtidas por RNAs. Sekanina e Ruzicka (2003) evoluíram com sucesso filtros, para aplicação em imagens, que demonstram qualidades semelhantes aos circuitos projetados de modo convencional. Higuchi *et al.* (1995) e Iwata *et al.* (1996) concluem que uma das vantagens dos sistemas evolutivos é a facilidade com que as caracteríticas de aprendizagem podem ser incorporadas, demonstrando este conceito em seus trabalhos onde evoluíram classificadores robustos de alta velocidade, sendo as boas características de generalização incorporadas nas soluções pela especificação de uma técnica de aprendizado baseada na teoria da aprendizagem de máquina.

Além dos benefícios citados acima pode-se apresentar ainda o desenvolvimento de técnicas de projetos em novas tecnologias. Os avanços na engenharia eletrônica estão gerando novas tecnologias de circuitos, sistemas onde o domínio das técnicas de projeto ainda é incompleto. Nestes casos a evolução pode ser uma técnica útil para encontrar novas soluções, por ser guiada basicamente pelo comportamento do circuito. Um exemplo é a tecnologia nano-eletrônica, onde Thompson e Wasshuber (2000) evoluíram com sucesso portas NOR.

Técnicas tradicionais de projeto tendem a trabalhar usando a metodologia *top down*, decompondo um problema em sub-problemas até que este se torne um conjunto bem especificado de pequenos problemas. Cada decomposição direciona o problema de modo a permitir o uso de regras formais de projeto em sua solução. A evolução, por sua vez, trabalha usando a metodologia *bottom up*, agrupando componentes para gerar soluções parciais ao problema, as quais por sua vez são combinadas e ajustadas até que a solução contemple todos os critérios do projeto. Os casos mais adequados para a aplicação desta característica são projetos para tecnologias em que o conhecimento sobre a interação entre seus componentes é muito limitado, com regras pouco desenvolvidas ou sem regras claras de projeto. Atualmente existem algumas tecnologias onde as técnicas convencionais para síntese lógica não são aplicáveis, porém são tecnologias importantes e muito úteis para projetistas de circuitos lógicos, como as tecnologias de lógica programável. Várias tecnologias de lógica programável fornecem portas XOR (Exclusive OR) e multiplexadores como elementos de projeto, porém técnicas convencionais de projeto estão adequadas para gerar soluções na forma de soma-de-produtos, que não mapeiam estes elementos. Nestes casos uma abordagem evolutiva pode trabalhar com um nível

de abstração adequado à tecnologia, explorar áreas não convencionais no espaço de soluções e encontrar soluções mais adequadas, como demonstrado por Miller *et al.* (2000).

Existem ainda tecnologias onde as regras de projeto são tão complexas que não foi possível desenvolver métodos formais para dividir ou decompor o espaço de projeto. Por exemplo, quando comparado ao espaço de projeto de circuitos lógicos digitais, o projeto de circuitos analógicos requer conhecimento mais especializado. Os algoritmos evolutivos mostram-se uma boa alternativa para gerar soluções aos projetos de circuitos analógicos Koza *et al.* (2000) e Aggarwal (2003). Uma aplicação bem sucedida da evolução em sistemas complexos é o projeto automático de antenas. Os projetos convencionais de antenas são baseados em um conjunto regular e muito bem conhecido de topologias, além das quais a complexidade dos elementos torna-se muito alta. Linden e Altshuler (1999) demonstraram que a evolução é capaz de descobrir topologias com disposição altamente não convencional, e que as antenas projetadas pela evolução podem operar de maneira eficaz em transmissão de dados reais, operando inclusive onde o trajeto do sinal é obstruído Linden (2002), Lohn *et al.* (2003*b*) e Lohn *et al.* (2005), isto demonstra que a evolução também explora regiões no espaço de busca não conhecidas.

A evolução explora uma região diferente do espaço de busca em relação aos projetistas convencionais, o que torna possível à evolução descobrir soluções inovadoras mesmo para espaços de busca bem conhecidos, gerando alguns circuitos que se encontram além das regiões do espaço de solução normalmente explorado. Isto exige que a evolução trabalhe sem as restrições usuais em projeto de circuitos, Thompson (1995).

Atualmente as técnicas evolutivas geram bons resultados para problemas pequenos, mas como os espaços da busca podem se tornar muito grandes para circuitos mais complexos, muita pesquisa está direcionada à escalabilidade. Uma abordagem é empregar a evolução para geração de circuitos menores, para serem usados como blocos construtivos na elaboração de circuitos maiores, tais blocos foram evoluídos para projetos analógicos e digitais por Miller *et al.* (1999) e Aggarwal (2003). Este enfoque também foi aplicado em níveis de abstração mais elevados em Sekanina (2003a), onde se sugeriu que núcleos *IP (Intellectual Property Cores)* evoluídos poderiam ser fornecidos para uso comercial em dispositivos de lógica programável. Zebulum *et al.* (2003) propôs que blocos previamente evoluídos auxiliem na evolução de circuitos maiores.

A evolução mostra-se, ainda, muito bem sucedida na geração dos circuitos que incorporam diversas funções dentro de um conjunto de componentes, uma operação onde há pouco conhecimento formalizado como regras de projeto. Esta é a função da eletrônica polimórfica, apresentada em Stoica *et al.* (2002), onde um circuito é evoluído para executar funções múltiplas usando um conjunto de componentes, em que cada função é ativada sob circunstâncias ambientais diferentes. Por exemplo, um circuito é capaz de executar função de porta E (AND) em uma temperatura e função de porta OU (OR) em outra, tais circuitos mostram grande utilidade para aplicações militares.

Esta nova área de pesquisa traz ainda outros benefícios que também podem ser apontados como motivação e desafios aos pesquisadores que trabalham nesta área. Primeiramente pode-se citar o fato desta técnica ter a capacidade de produzir circuitos funcionais cuja estrutura espacial e dinâmica está fora do escopo convencional de projeto. Com isso, os circuitos obtidos podem revelar propriedades eletrônicas ou configurações de componentes ainda não exploradas no projeto convencional. Além disso, quando implementada intrinsicamente, a evolução é capaz de explorar propriedades físicas que não podem ser modeladas na abordagem tradicional. Outro benefício observado durante a evolução é a capacidade de produzir a funcionalidade desejada mesmo com a ausência de componentes que seriam necessários em projetos convencionais.

3.2.2 Tendências e Aplicações

Sistemas adaptativos são sistemas que podem mudar sua configuração ou seu comportamento durante a fase operacional de acordo com mudanças que possam ocorrer no ambiente ao qual está inserido, ou por motivos de falhas que possam ocorrer no próprio sistemaStoica *et al.* (1998).

Com o poder da automação atual (a síntese em tempo real fornecida por PLDs), o Hadware Evolutivo tem potencial de adaptar-se de modo autônomo às mudanças em seu ambiente. Isto pode ser muito útil em situações onde ações de controle “manual” sobre o sistema não são possíveis, como em missões espaciais, onde a adaptação do sistema de controle poderia ser particularmente útil quando circunstâncias inesperadas são encontradas. Stoica *et al.* (1998), no entanto, relata que a inexistência de métodos de teste e validação (verificação) que possam ser aplicados aos sistemas evolutivos em tempo real não permite que sistemas de controle críticos sejam colocados sob o controle de sistemas evolutivos.

Greenwood e Song (2002) propuseram o uso de técnicas evolutivas em conjunto com o uso de técnicas formais de verificação para prevenir este problema, entretanto somente sistemas não críticos tais como sistemas de processamento de sinais de sensores e sistemas de compressão de dados adaptativos foram explorados, Fukunaga e Stechert (1998). Outros sistemas podem se beneficiar da habilidade de evoluir de modo autônomo, tais como: sistemas de gerência de potência e controladores para antenas, Plante *et al.* (2003), sistemas adaptativos de compressão, Sakanashi *et al.* (2001) e filtros de imagens em ambientes em constante alteração, Sekanina (2002).

Muitos filtros adaptativos foram evoluídos, incluindo filtros digitais de resposta finita ao impulso (FIR), usados geralmente em aplicações de áudio tais como o cancelamento de ruído e eco, Tufte e Haddow (2000), e seus equivalentes mais complexos filtros digitais da resposta infinita ao impulso (IIR), Sundaralingam e Sharman (1998). Filtros analógicos adaptativos também foram evoluídos. Zebulum *et al.* (2003) apresentou filtros capazes de amplificar o componente principal do sinal de entrada atenuando outros, melhorando a relação sinal/ruído do filtro. A evolução permite que estes circuitos sejam adaptados a novos perfis de sinal de entrada. Hardware para o seqüenciamento em tempo real também estão sendo desenvolvidos, principalmente sistemas de seqüenciamento em redes ATM - *Asynchronous Transfer Mode*, que respondem pela mudança no fluxo de dados Liu *et al.* (1996) e Li e Lim (2003).

Outra aplicação muito promissora na área de hardware evolutivo é a dos sistemas tolerantes a falhas. Os avanços mais recentes na área de miniaturização de componentes não foram complementados por melhorias na confiabilidade do processo de fabricação. Isto significa que projetos modernos de circuitos em VLSI requerem tolerância a falhas no processo de fabricação. Este aspecto será uma das características mais importantes no desenvolvimento de futuras tecnologias de circuitos. Miniaturização expõe os componentes a um risco maior de falhas operacionais, como por exemplo variações na potência (fontes de alimentação) ou devido aos efeitos de ionização por radiação. A confiabilidade é de grande importância para muitos sistemas, tais como equipamentos médicos e sistemas de controle de transporte. Sistemas militares e aeroespaciais são particularmente suscetíveis aos problemas de confiabilidade porque estão regularmente sujeitos à condições severas.

A abordagem mais comum para produzir sistemas tolerantes a falhas é a redundância de componentes, que incorpora partes adicionais ao sistema, que serão usados quando uma falha for detectada, além de testes completos no processo de

manufatura. Isto aumenta consideravelmente a complexidade e o custo do projeto. No desenvolvimento de alguns sistemas existem restrições de tamanho e custo o que pode tornar impraticável a redundância. Nestes casos procura-se alguma forma de suavizar o problema causado pela falha, caso esta ocorra.

A técnica de hardware evolutivo fornece mecanismos para introduzir a tolerância de falha no projeto de circuitos. Uma classe de sistemas adaptativos compreende os circuitos que podem se adaptar às falhas em seu próprio hardware, fornecendo assim um mecanismo de recuperação de falha. Uma demonstração desta habilidade encontra-se em Higuchi *et al.* (1995), onde um sistema adaptativo em hardware aprende o comportamento de um controlador robótico usando algoritmos genéticos. Mais recentemente Vigander (2001) demonstrou que um sistema evolutivo simples poderia restaurar a maioria, mas não toda a funcionalidade, de um multiplicador 4bit $\times$ 4bit sujeito à falhas aleatórias. A funcionalidade completa poderia ser restaurada usando um sistema para selecionar um entre diversos circuitos alternativos que haviam sido reparados pela evolução. Sinohara *et al.* (2001) usou um algoritmo evolutivo multi-objetivo que permite que a funcionalidade essencial seja restaurada à custa do comportamento secundário, que normalmente não é visto como importante pelo projetista, como, por exemplo, a dissipação de potência. Isto foi demonstrado no reparo de portas NOR e inversores. Hounsell e Arslan (2001) exploraram o reparo de um filtro FIR após a injeção de múltiplas falhas, dois métodos diferentes de recuperação foram examinados, o primeiro recorda a população final do funcionamento evolutivo que criou o projeto original do filtro, e o segundo cria uma população aleatória nova com uma cópia do projeto original. Ambos os mecanismos recuperaram a funcionalidade mais rapidamente do que uma nova evolução com uma população completamente aleatória.

Zebulum *et al.* (1998) demonstrou a recuperação evolutiva em conversores digital/analógicos de 4bits, evoluídos inicialmente usando amplificadores operacionais tradicionais e DACs menores, evoluídos previamente, usados como blocos funcionais. As falhas foram introduzidas em um dos amplificadores operacionais e o circuito recuperado demonstrou eficiência superior ao circuito evoluído inicialmente. Gwaltney e Ferguson (2003) investigaram a recuperação de falha em um controlador analógico de um motor, projetado por evolução, re-evoluindo a população e gerando um controlador não defeituoso superior ao original. Estes experimentos mostram que a evolução pode recuperar falhas em alguns componentes melhor do que outros, embora ao menos alguma funcionalidade fosse restaurada em todos os casos. Louis (2003) combinou um enfoque evolutivo com um "sistema especialista" (*cased based memory*), onde soluções parciais a problemas similares, previamente resolvidos, fo-

ram introduzidos na população evolutiva. Observou-se que soluções de qualidade superior, para problemas de paridade, poderiam ser evoluídas em menos tempo do que ao usar unicamente a evolução, sugerindo que este método pode ser útil para a recuperação de falhas.

A maioria dos sistemas evolutivos de recuperação de falhas que foram estudados até o momento exploraram somente a recuperação de erros introduzidos na lógica do circuito. Lohn *et al.* (2003*a*) mostraram um sistema evolutivo de recuperação de falhas capaz de reparar o roteamento, além da lógica, sendo muito útil para aplicações em dispositivos programáveis modernos. Um outro tipo de falha é a falha de um componente em temperaturas extremas. Stoica *et al.* (2002) observaram que multiplicadores, geradores de curva gaussiana e as portas lógicas evoluídos sob circunstâncias padrão degradam-se, ou falham, em temperaturas extremas. Entretanto quando re-evoluídos nestas temperaturas, os circuitos mantém a funcionalidade em todos os casos. A detecção de falha tradicionalmente incorpora algum hardware adicional ao projeto para executar testes internos (Built-In Self Test - BIST). Garvie e Thompson (2003) demonstraram que a evolução pode adicionar ao projeto pequenos somadores e multiplicadores que incorporam o BIST e compartilham componentes com a função do circuito, resultando em Hardware/BIST de baixo custo. Vários outros sistemas de hardware bio-inspirados com tolerância e recuperação à falhas estão sendo desenvolvidos e estudados, como em Bradley e Tyrrell (2001), Macias e Durbeck (2002) e Tyrrell *et al.* (2003). A tolerância à falha faz referência aos sistemas que são intrinsecamente tolerantes às falhas, e não aos sistemas que podem detectar e/ou recuperar as falhas. A evolução apresenta-se como candidata ideal para a exploração de sistemas tolerantes à falhas.

3.3 Conceitos, Motivações e Desafios

Apresentados os benefícios do Hardware Evolutivo, e algumas das tendências e aplicações que estes benefícios permitem, esta seção revê os principais conceitos, motivações e desafios da pesquisa neste campo, que é decomposto em três áreas - inovação, generalização e evolutibilidade.

3.3.1 Conceitos

Hardware evolutivo consiste em aplicar algoritmos evolutivos no projeto e síntese de circuitos lógicos. Os algoritmos evolutivos são algoritmos de busca e sua tarefa é procurar um conjunto de parâmetros em um determinado domínio que corresponda a uma solução para o problema em questão. Para o projeto de circuitos eletrônicos, o domínio do problema é a derivação ou a construção de um circuito, e as soluções são os circuitos obtidos, a partir deste domínio, que se comportam de acordo com algumas especificações dadas. Os parâmetros definem as quantidades e os tipos de componentes disponíveis e como estes estão interconectados. Cada conjunto de valores e parâmetros distintos correspondem a um circuito único, e são chamados de indivíduos no processo evolutivo.

Inicialmente deve-se definir um método de representação dos indivíduos da população e uma função para calcular a aptidão destes indivíduos. No início do processo de evolução uma população inicial é gerada, geralmente de forma aleatória, e todos os indivíduos desta população (circuitos) são avaliados através da construção deste circuito em hardware reconfigurável ou através de simulação. Os circuitos são então avaliados e recebem uma nota, de acordo com a função de aptidão ou fitness, que reflete o seu comportamento com relação ao comportamento desejado. Uma vez que toda a população foi avaliada ocorre o processo de seleção e reprodução para a construção da próxima geração. Existem diferentes métodos que podem ser aplicados ao processo evolutivo nas etapas de seleção e reprodução, conforme pode ser observado ao longo deste trabalho. O processo termina quando uma solução que satisfaça as restrições seja encontrada ou até que um determinado número de gerações foi processado. Algumas propriedades podem ser consideradas para a implementação de trabalhos nesta área e serão abordadas com maiores detalhes no capítulo 3. Pode-se evoluir projetos para otimização e/ou síntese de circuitos analógicos ou digitais, em diferentes plataformas. Pode-se também escolher o algoritmo evolutivo e formas diferentes de representação que serão usados durante o processo, Zebulum (1999).

3.3.2 Motivações

A evolução opera com a técnica *bottom up*, conforme mencionado anteriormente, tentando encontrar correlações entre o conjunto de componentes que melhorem consideravelmente o comportamento de um circuito com respeito ao problema

em questão. Ao contrário dos métodos convencionais de projeto, o sentido da busca é independente das complexidades das interações dentro do circuito, e as soluções são descobertas usando somente o comportamento externo como guia. Devido ao tipo de exploração que acontece durante o processo de busca o hardware evolutivo apresenta uma das suas principais motivações: a inovação. Quatro pontos principais são explorados com ênfase em buscar projetos inovadores: Projeto com restrições amenas para tecnologias conhecidas, novas abstrações na lógica programável, projeto em tecnologias complexas e projeto em novas tecnologias; estes ítens são discutidos a seguir.

Um dos principais trabalhos sobre relaxamento de restrições foi realizado por Harvey e Thompson (1996) e mostrou, primeiramente, que a evolução poderia manipular com sucesso a dinâmica e a estrutura dos circuitos quando as restrições das quais os projetistas tradicionais dependem, fossem relaxadas. O trabalho consistiu na evolução de um circuito realimentado, com portas lógicas de alta velocidade e codificado como um *netlist*, para comportar-se como um oscilador de baixa freqüência. A aptidão foi medida pelo erro médio baseado na soma das diferenças entre períodos de transição desejados e medidos em simulação digital assíncrona. O espaço de busca conteve somente os circuitos que possuíam comportamento modelado pelo simulador, com o espaço dividido estritamente em unidades de portas lógicas. Entretanto o simulador permitiu que as portas interagissem assincronamente, desse modo o operador da seleção poderia explorar a dinâmica assíncrona do modelo, com liberdade para explorar ou ignorar um determinado comportamento.

O circuito evoluído mostrou que a evolução é capaz de gerar soluções sem as restrições (neste caso restrições de sincronismo), geralmente impostos à projetistas tradicionais. O circuito solução não contém nenhum módulo estrutural significativo, como seria visto em uma abordagem *top-down* convencional, além de possuir um comportamento que não se enquadra nas regras de projeto usadas por projetistas convencionais. O trabalho demonstra que a evolução não só foi capaz de encontrar uma solução explorando um espaço de busca além do espaço convencional, como encontrou a solução fora deste espaço. Harvey e Thompson (1996) mostraram que a evolução com poucas restrições na dinâmica do circuito era possível no *hardware* físico, gerando resultados superiores a simulação. O trabalho mostra a evolução de uma máquina de estado finito aplicada ao controle de um robô, o controlador evoluído usou uma mistura do comportamento síncrono e assíncrono, interagindo com o ambiente de maneira dinâmica muito complexa, produzindo comportamento que não seria possível usando regras tradicionais para projeto de máquinas de estados finitos, mostrando que a habilidade do controlador em interagir com o ambiente de

uma maneira tão complexa não era atribuída à arquitetura da máquina de estados e sim a habilidade da evolução de explorar este tipo de soluções.

Thompson (1996) realizou a primeira evolução intrínseca de um circuito em um FPGA. Neste trabalho o *bitstream* de uma área 10×10 de um Xilinx XC6126 foi evoluído. Os *bits* deste *bitstream* foram evoluídos diretamente como os *bits* de um cromossomo de um algoritmo genético. Desse modo Thompson evoluiu um circuito no nível de abstração mais baixo possível com o dispositivo em questão, ou seja, o comportamento físico da tecnologia. A tarefa era evoluir um circuito com habilidade para discriminar dois sinais, um de 1kHz e outro de 10kHz. A aptidão de cada circuito foi calculada aplicando cinco períodos de 500ms de cada sinal em uma ordem aleatória, e concedendo aptidão elevada aos circuitos com uma diferença grande entre a tensão média, medidas com um integrador analógico, da saída durante estes períodos. A única entrada ao circuito era o sinal de 1kHz/10kHz - nenhum pulso de clock foi fornecido. A tarefa consistia em encontrar um circuito capaz de discriminar entre sinais com tempos muito maiores do que o tempo de atraso de cada componente individualmente. O circuito resultante usou uma fração dos recursos que um projetista convencional necessitaria para resolver a mesma tarefa. Thompson e Layzell (2000) descreveram a funcionalidade do circuito como "rara", referindo-se ao mecanismo de funcionamento pouco compreendido do circuito, embora postulassem que o circuito empregou o comportamento físico do substrato em uma maneira que os projetistas convencionais consideram complexa demais para ser levada em consideração.

A maioria das metodologias de projeto de circuitos digitais produzem circuitos usando a descrição da lógica na forma de soma-de-produtos. Entretanto, muitos dispositivos de lógica programável possuem componentes adicionais que não são representados facilmente por tal descrição, como portas XOR, multiplexadores e tabelas (*lookup tables - LUTs*). Miller *et al.* (1999) pesquisam novos princípios de projeto, para criar níveis de abstração que poderiam ser aplicados em dispositivos de lógica programável, buscando mostrar que a abordagem evolutiva pode explorar não apenas a classe de expressões algébricas, mas um espaço maior de representações lógicas que vai além das álgebras geralmente usadas. Com objetivo de demonstrar isto evoluíram com sucesso somadores de um e dois *bits*, baseados no princípio do somador de *ripple* usando uma representação em *netlist* de portas E, OU, NÃO, e XOR e de MUX. Esta representação encontra-se além dos espaços geralmente usados na álgebra booleana, sendo de grande interesse porque representa elementos disponíveis como unidades básicas em muitas tecnologias. Este argumento é similar ao de Thompson, onde a descoberta de circuitos novos pode ser facilitada com a

modificação das restrições do projeto.

Muitos dos circuitos apresentados em Miller *et al.* (1998) e Miller *et al.* (2000) são considerados incomuns, porém interessantes devido a eficiência em termos de otimização de literais. Estes circuitos empregam codificações com multiplexadores e portas de XOR, fora do espaço tradicional da lógica booleana. Estes trabalhos mostraram circuitos que dificilmente seriam encontrados por métodos algébricos tradicionais, e que o sistema evolutivo de "montar-e-testar" é uma metodologia útil para explorar tal espaço. O trabalho continuou com a evolução de multiplicadores de dois e três bits. Todos os trabalhos foram realizados usando simulação e codificação no nível de portas. Trabalho similar foi realizado com álgebras de múltiplos valores Kalganova *et al.* (1998). Outro aspecto do trabalho deste grupo é o argumento que os princípios de projeto úteis para os projetistas convencionais podem ser encontrados, ou descobertos, pela busca de padrões em circuitos evoluídos. Kalganova e Miller (1999) e Miller *et al.* (1998) evoluíram somadores binários de um e dois bits e evoluiram o princípio de contador de *ripple*. Embora o conhecimento sobre este princípio já existisse, os autores argumentaram que a evolução o descobriu e o utilizou, sem conhecimento prévio ou direcionamento para tal. Por meio deste experimento concluíram que a evolução pode ser usada para descobrir novas metodologias de projeto.

Um dos trabalhos mais marcantes nesta área Miller *et al.* (2000), concentra-se em desenvolver métodos automáticos de detecção de técnicas. Vassilev *et al.* (2000) evoluíram com sucesso multiplicadores de dois e três bits muito mais compactos que os tradicionais, um sistema de *data mining* foi usado para encontrar princípios ou regras de projetos. O sistema de *data mining* usou um algoritmo de aprendizado conhecido como Sistema Especialista - *Case Based Reasoning*, Mitchell (1997). Deste modo podemos esperar que a evolução desenvolva modelos biológicos e automaticamente empregue estes princípios para o projeto de circuitos, sem a necessidade de incluir outros algoritmos de aprendizagem no sistema evolutivo.

Sob outro ponto de vista é observado que algumas tecnologias tem espaços de projeto tão complexos que não é possível desenvolver métodos formais de particionamento para decompor os problemas, e neste caso os algoritmos evolutivos aparecem como alternativa ao projetista humano. Um exemplo desta classe de problemas é o projeto de circuito analógicos.

Uma técnica tradicional para simplificar um espaço de projeto analógico é escolher a topologia de um circuito a ser projetado, com características bem conhe-

cidas, e modificar somente parâmetros que se relacionam com os componentes do circuito. Muitos trabalhos usam este enfoque no projeto de circuitos analógicos, o que pode ser considerado como otimização evolutiva, Arslan e Horrocks (1995) e Murakawa *et al.* (1998). No entanto a evolução também vem sendo usada para desenvolver novas topologias de hardware analógico. Grimbleby (2000) desenvolveu um algoritmo híbrido, que usa algoritmos genéticos para busca de topologias e métodos de busca numérica para otimização dos valores dos componentes destes da topologia evoluídas. Lohn e Colombano (1998), Koza *et al.* (1999) desenvolveram métodos para evolução de circuitos que evoluem tanto a topologia quanto os valores dos parâmetros dos componentes contidos na topologia.

Estes trabalhos levaram Gallagher (2003) a defender o retorno dos computadores analógicos. Gallagher divide os computadores analógicos em duas classes, a primeira são os computadores analógicos diretos, que são projetados para reproduzir diretamente o comportamento de um sistema físico, um exemplo disso é um circuito RLC, que pode reproduzir facilmente o comportamento de um sistema massa-mola-amortecedor. Os computadores analógicos indiretos possuem somente funções matemáticas simples como somadores e integradores. No entanto sabe-se que a modelagem de sistemas por meio de componentes analógicos sofre com a falta de precisão de seus componentes, para evitar isso a evolução seria usada para reproduzir um comportamento, e não uma função matemática, podendo ainda ser uma boa alternativa onde potência e tamanho são restrições de projeto, além de servir como alternativa aos modelos digitais de sistemas analógicos.

Outro ponto fundamental é que evolução pode ser uma ferramenta útil para projetos de circuitos em novas tecnologias, onde pouco ou nenhum conhecimento sobre a tecnologia está disponível. Thompson (2002) afirma que enquanto uma tecnologia não possui uma modelagem completamente especificada somente técnicas de busca cega são capazes de projetar circuitos para tal tecnologia. Neste contexto pode-se aplicar uma técnica de busca heurística, CE, que explore o espaço de solução em menor tempo e com eficiência. Buscando apresentar um exemplo disto Thompson evoluiu uma porta NOR em nano-tecnologia de circuitos. Sabendo que os efeitos de um quantum não podem ser ignorados em seu modelo macroscópico, um sistema onde um conjunto de quantuns (*quantum dots*) pelos quais um elétron pode passar apenas por um túnel mecânico quântico (*quantum mechanical tunnelling*) foi criado, e a evolução usada para modificar o tamanho a forma e a posição dos pontos. Os circuitos evoluídos alteram os efeitos dos túneis para executar a tarefa lógica (somente simulação) e usaram uma propriedade chamada de ressonância estocástica, onde a energia térmica dos elétrons permite a transmissão estocástica de um sinal. Esta

propriedade nunca havia sido usada antes em um projeto de circuitos, a evolução descobriu esta propriedade e demonstrou sua habilidade em projetar circuitos na ausência de qualquer regra de projeto, Gordon e Bentley (2005).

Existem outras pesquisas que buscam explorar as características do quantum aplicados à computação quântica. Computadores quânticos possuem poder de processamento que cresce exponencialmente em relação aos computadores tradicionais, a teoria apresenta apenas uma pequena quantidade de circuitos rudimentares para desenvolver circuitos quânticos, enquanto a prática mostra que o número de conexão entre estes circuitos é uma forte restrição para projeto de circuitos mais complexos. Estes problemas têm levado um número crescente de pesquisadores a pesquisar e propor novos circuitos quânticos baseados na evolução, Lukac *et al.* (2003) e Surkan e Khuskivadze (2002).

Muitos pesquisadores defendem a idéia que novas tecnologias devem ser desenvolvidas levando em consideração a aplicação da evolução no projeto de circuitos e não o contrário. Miller e Downing (2002) lembram que devido a capacidade de explorar as características físicas de uma tecnologia, esta deve ser projetada para ser reconfigurável, permitir um grande número de conexões, baixas tensões de funcionamento, tudo para facilitar a aplicação de técnicas evolutivas. Sugere-se ainda que novas tecnologias como as baseadas em cristais líquidos, polímeros eletroativos e colóides controlados por tensão são possíveis candidatas. A computação amorfa tem sido indicada como um substrato sensível a evolução. Computadores amorfos são um grande conjunto de unidades "*wireless*" simples que executam cálculos. De um modo geral estas unidades não são confiáveis, tem geometria variável e só são capazes de se comunicar localmente, no entanto são fáceis de sintetizar mesmo em grandes arranjos, se comparados com outras novas tecnologias. Haddow e Tufte (2001) sugerem a combinação de princípios de projetos biológicos e hardware evolutivo como uma possível ferramenta para projetar computadores amorfos.

Os pontos citados acima são vistos como motivações para pesquisadores desta área. Existem ainda outros fatores que podem ser citados como motivação. Um destes é o desenvolvimento de sistemas construídos via especificação comportamental, que possa ser estabelecida com quase nenhum conhecimento de eletrônica, possibilitando, desse modo, que projetos de hardware possam ser concebidos por pessoas não especializadas.

Desafios

Na seção anterior foram apresentadas algumas motivações para os pesquisadores da área de hardware evolutivo. Apesar das promessas e benefícios citados, existem uma série de dificuldades e incertezas cuja resolução será um fator importante no futuro da área como uma ferramenta de engenharia. Nesta seção serão abordados alguns dos maiores obstáculos para viabilizar o desenvolvimento de hardware evolutivo para aplicações do mundo real, como generalização, escalabilidade, performance e evolutibilidade.

1. Generalização

Algoritmos de aprendizado indutivo tais como os algoritmos evolutivos criam hipóteses observando e tomando como exemplo os circuitos treinados. Em hardware evolutivo os circuitos são testados expondo-os à circunstâncias diferentes, geralmente um conjunto de sinais de entrada, e observando as saídas do circuito a fim avaliar sua aptidão. Sendo infactível para todos os exemplos possíveis do treinamento, pois o algoritmo generaliza além dos casos que observou. Circuitos modernos podem processar centenas de sinais de entrada, o que pode facilmente levar a milhões de casos de treinamento, isto pode resultar em um tempo de execução muito grande ou até mesmo impossível computacionalmente. Para circuitos seqüenciais o número de casos de treinamento pode tornar-se infinito. Assim espera-se que as entradas de sinal não apresentadas, ou exemplos de operações não apresentadas, sejam generalizadas. Com isso, fica evidente que a capacidade de generalização é uma característica vital para o Hardware Evolutivo.

Dois enfoques são apresentados na literatura com relação a generalização:

(a) Conhecimento de domínio estrutural dos circuitos que possuam as características de generalização, provavelmente na forma de heurísticas.

(b) Conhecimento de domínio comportamental dos circuitos que possuam as características de generalização, deixando para a evolução o aprendizado sobre a estrutura dos circuitos que exibem o comportamento requerido.

1.1 Generalização em Relação aos Sinais de Entrada

Diversos pesquisadores exploram a generalização em relação aos sinais de entrada usando como enfoque a teoria de reconhecimento de padrões, que é um

problema que possui estreita relação com a generalização e é considerada bem estabelecida neste contexto. Muitos sistemas desenvolvidos demonstram que o hardware evolutivo pode generalizar casos de testes não apresentados como na classificação de sinais de imagem Higuchi *et al.* (1995) Murakawa *et al.* (1999), e na filtragem de ruído em sinais Sekanina (2003a), Vinger e Torresen (2003). Yao e Higuchi (1999) acreditam que o sucesso do Hardware Evolutivo aplicado a esta classe de problemas consiste no uso de elementos de processamento não lineares dispostos em forma de rede, similares as RNAs. Em seu trabalho Iwata *et al.* (1996) melhoraram a capacidade de generalização deste tipo de sistema introduzindo conhecimento adicional ao sistema através de heurísticas (*Minimum Description Length - MDL*), mais detalhes podem ser encontrados em Mitchell (1997).

Miller e Thomson (1998*a*), Miller e Thomson (1998*b*) e Miller e Thomson (1998*c*) pesquisaram a capacidade de generalização de um sistema através da evolução de multiplicadores, de dois e três bits, em relação ao tamanho do conjunto de dados de treinamento, ou seja, o circuito foi evoluído a partir de um subconjunto da tabela verdade. Concluíram que se a evolução fosse realizada com um subconjunto de casos de treinamento o algoritmo evolutivo não produzia soluções gerais. Isto indica que a configuração deste problema e do algoritmo não continham nenhum direcionamento para a generalização. Concluíram também que, mesmo quando o conjunto de treinamento era extraído aleatoriamente da tabela a cada geração, a evolução não fornecia soluções gerais, sugerindo que a evolução teve pouca memória no contexto deste problema. Neste mesmo trabalho foram evoluídos circuitos para executar a função raiz quadrada. Neste caso concluíram que a evolução gerou soluções aceitáveis com um conjunto incompleto de dados de treinamento. Imamura *et al.* (2000) considerando o mesmo experimento de Miller e Thomson (1998*b*) concluíram que evoluir multiplicadores completamente corretos é uma tarefa extremamente difícil sem um conjunto completo de dados. Eles referem-se especialmente a conjuntos que possuem a mesma quantidade de informação para gerar a solução final. No entanto demonstraram que para um problema que possui uma grande quantidade de estados não especificados (*don't care values*) o hardware evolutivo é capaz de evoluir com sucesso circuitos com um subconjunto de dados de treinamento. Concluíram ainda que problemas de classificação, ou reconhecimento de padrão, possuem uma grande quantidade de informação redundante, diferente de um problema de multiplicação. No entanto aceitam como hipótese razoável que a maior parte dos problemas reais possuem uma certa quantidade de informação redundante, mas apontam como um problema em aberto a metodologia para selecionar o sub-conjunto de dados corretos para o treinamento.

1.2 Generalização através do Ambiente de Evolução com Relação a Representação

Assim como em muitos problemas é impraticável usar o conjunto completo de dados de entradas para evoluir o circuito, em muitos casos é impossível evoluir o circuito em todos os ambientes aos quais o circuito pode ser submetido. Os ambientes podem incluir desde o conjunto de tecnologias ou plataformas onde o circuito projetado deve operar, até a gama de condições que o circuito pode ser submetido.

Os projetistas tradicionais controlam tal generalização impondo restrições severas à natureza do circuito. Estas restrições são regras que codificam um padrão para produzir um determinado comportamento e refletem-se diretamente no hardware projetado. Um circuito que se comporte corretamente em todas as circunstâncias ou tecnologias ainda não tem possibilidade de ser projetado. Por exemplo, um projeto em nível de portas requer que as portas evoluídas comportem-se como operadores lógicos perfeitos para todas as tecnologias. Na maioria das tecnologias as portas são criadas por meio de transistores de alto ganho e as restrições de ambiente/tecnologia são resolvidas por meio de circuitos de sincronismo. O projeto de circuitos evolutivos geralmente apresenta um enfoque similar ao processo tradicional de projeto, aplicando as mesmas restrições de projeto usados por projetistas convencionais. Muitos circuitos foram evoluídos em níveis de abstração que poderiam limitar a busca de circuitos com boas características de generalização. Entretanto no caso em que a representação dos níveis de abstração foram impostos especificamente para assegurar a generalização foi apresentado em Stoica *et al.* (2003), onde foi requerido um nível muito elevado de generalização. A experiência envolveu a evolução de circuitos com representação no nível dos transistores, e uma determinada representação cujo objetivo era melhorar determinadas características dos circuitos evoluídos.

1.3 Generalização através do Ambiente de Evolução com Relação a Informação de Exemplos

Nos casos onde nenhum conhecimento sobre a estrutura das soluções a serem generalizadas está disponível, a solução é inferir esta informação dos exemplos de treinamento. O trabalho que aborda a evolução intrínseca de circuitos feito por Thompson (1996) focalizou a inovação do projeto com a relaxação das restrições. Neste trabalho ele evoluiu com sucesso um circuito para distinguir entre duas freqüências, usando uma FPGA Xilinx XC6200. Porém notou que condições ambientais tais como a temperatura, os periféricos eletrônicos, e a fonte de alimentação

influenciaram na performance do circuito. Notou também que o projeto não era portável, não somente quando movido para um FPGA diferente, mas também quando movido para uma área diferente do mesmo FPGA. Resultados similares foram encontrados por Masner *et al.* (1999). Thompson passou a estudar como soluções, com boa capacidade de generalização, poderiam ser evoluídas em relação a um conjunto de ambientes de operação, Thompson (1998). Partindo de um circuito previamente evoluído, para um determinado FPGA, com capacidade para discriminar entre dois tons, especificou um número de parâmetros para o ambiente de operação que, quando variados, afetassem o desempenho deste circuito como temperatura, fonte de alimentação, variações de fabricação, encapsulamento, periféricos, carga de saída e posição no FPGA. A população final desta experiência continuou a evoluir, desta vez em cinco FPGAs diferentes, operando nos limites das circunstâncias especificadas pelos parâmetros de operação. Embora não houvesse nenhuma garantia que o circuito alcançaria a generalização para se comportar de maneira robusta a todas as condições propostas, Thompson encontrou um nível de robustez para quatro dos cinco casos.

De um modo similar o trabalho de Stoica *et al.* (2001) explorou a operação dos circuitos em temperaturas extremas. A experiência usou circuitos projetados tradicionalmente e circuitos evoluídos (como multiplicadores e portas lógicas), ambos projetados para atuar sob circunstância padrão e para temperaturas extremas, e concluiu que os circuitos degradam ou falham a estas temperaturas. Este experimento, inicialmente proposto para estudar tolerância a falhas, demonstrou que todos os circuitos poderiam manter a funcionalidade quando evoluídos sob circunstâncias extremas. Porém é interessante notar que uma população de 50 circuitos re-evoluída por mais 200 gerações geralmente exibiam desempenho degenerado sob circunstâncias padrão, onde antes funcionavam perfeitamente. Isto sugere que a qualidade da generalização pode ser facilmente perdida se uma polarização (direcionamento) consistente não for adicionada durante a evolução. Um problema relacionado com o estudo de Thompson é a portabilidade de circuitos analógicos evoluídos extrínsecamente. Os simuladores de circuitos analógicos tendem a simular muito bem o comportamento dos circuitos, e assim espera-se que circuitos evoluídos extrinsecamente generalizem o comportamento de circuitos reais. Entretanto, isto não acontece na prática. Uma causa disso é que o modelo físico programado no simulador para um determinado elemento pode ser impraticável em determinada tecnologia. Um exemplo comum é que os simuladores não impedem a simulação de correntes extremamente elevadas, e assim a evolução está livre para extrair vantagem dessas características em seu projeto. Koza *et al.* (1999) evoluíram muitos circuitos extrinsicamente usando o simulador SPICE, mas os circuitos que encontraram são praticamente infactíveis devido às correntes extremamente elevadas. Adicionalmente os simuladores

analógicos usam circunstâncias de operação muito precisas, os circuitos de Koza *et al.* (1999) são simulados para operar em 27^0C, e não há nenhum direcionamento explícito para a generalização em relação a uma escala das temperaturas.

Ao evoluir redes de transistores intrinsecamente Stoica *et al.* (1999) encontraram o problema reverso: os circuitos evoluídos operavam satisfatoriamente no circuito físico (intrínseco), mas não produziam resultados adequados em simulação (extrínseco). Propuseram que o problema devia avaliar os circuitos nos dois modos de execução, e denominaram esta estratégia de mixtrínseca, Stoica *et al.* (2000). Apontaram também a possibilidade de recompensar as soluções que operam diferentemente na simulação e no circuito físico como incentivo ao surgimento de soluções inovadoras, com características físicas não capturados pela simulação. Em Guo *et al.* (2003) um método para incluir diferentes técnicas de modelagem, e diferentes métodos numéricos para solução dos modelos foi proposta com o intuito de amenizar este problema.

As discussões sobre portabilidade apresentadas anteriormente trataram somente da portabilidade entre a simulação e os PLDs. Um ponto de extrema importância é se os circuitos evolutivos intrínsecos ou extrínsecos são portáveis o suficiente para serem aplicados em ASICs (*Application Specific Integrated Circuits*), característica que não pode ser testada pela abordagem mixtrínseca. Esta pergunta permanecia sem resposta até Stoica *et al.* (2003) evoluírem portas ao nível de transistor usando uma detalhada função de avaliação, que incluíam análise transitória a uma grande variedade de cargas. As evoluções mixtrínsecas incluíam simulação de diversos modelos, em diferentes condições de temperatura e escala de tensão. Todos os testes foram aplicados a todos os indivíduos da evolução, e as soluções finais foram validadas com sucesso em silício. Isto mostra que por meio de critérios de avaliação cuidadosamente selecionados, a portabilidade para AISCs pode ser possível.

2. Performance, Escalabilidade e Evolutibilidade

Muita pesquisa na área de hardware evolutivo é dedicada à melhora na qualidade das soluções, ao aumento da escalabilidade e a melhora na velocidade com que a evolução encontra soluções aceitáveis. Estas idéias estão altamente relacionadas e aspiram melhorar o desempenho da busca evolutiva.

Um ponto crítico em hardware evolutivo é a representação das soluções. Selecionar uma boa representação é crucial ao desempenho de um algoritmo evolutivo. A representação de um algoritmo evolutivo define como o espaço da solução (fenó-

tipo) é mapeado no espaço da busca (genótipo). A representação afeta diretamente o desempenho do algoritmo, limitando e fixando a densidade de soluções factíveis no espaço de busca. Muitos pesquisadores acreditam que, com uma representação eficiente, o desempenho da busca possa ser melhorado reduzindo o tamanho do espaço da busca e aumentando a densidade das soluções encontradas nele. Dawkins (1987), Altenberg (1994) e Thompson (1996) apontam que a representação é um aspecto fundamental por definir a natureza do espaço de busca e concluem que mais importante do que ter um espaço de busca pequeno é ter um espaço de busca que permita que a evolução descubra boas soluções. Miller e Thomson (1998*a*) exploraram como a mudança na geometria dos circuitos afeta a evolução de um multiplicador de dois bits, e como a relação funcionalidade/localização (*functionality-to-routing*) afeta sua evolução, mostrando que a evolução é sensível a ambos os fatores.

Existem basicamente dois níveis de representação para aplicação de hardware evolutivo no nível digital, a representação no nível de porta, onde a unidade básica manipulada durante o processo evolutivo é uma porta lógica (AND, OR, NOR, NAND ou XOR), é considerada o tipo de representação tradicional Thompson (1996) Miller *et al.* (1998), e a representação no nível funcional Liu *et al.* (1996) onde a unidade básica é uma função digital mais complexa como um somador, um multiplicador ou um produto de variáveis lógicas. Outras variações podem ocorrer no processo de representação, como por exemplo, a definição da estrutura de dados sobre a qual o cromossomo será manipulado. Deve-se citar também que a natureza do processo (analógico ou digital) deve ser levada em consideração no momento da representação possibilitando a geração de diferentes tipos de representação além das formas usuais. Maiores detalhes relacionados a representação das soluções serão abordadas no capítulo três deste trabalho.

O uso de evolução no nível de função para melhorar a evolutibilidade, proposta por Murakawa *et al.* (1996) e estudada por muitos outros Torresen (2000), Sekanina (2002) e Thomson e Arslan (2003), indicam que o tamanho do espaço de busca para um algoritmo genético binário aumenta a uma taxa de 2^n para cada adição de n genes, e apontam que quando a evolução enfrenta problemas maiores a explosão no tamanho do espaço de busca impede que soluções eficazes sejam facilmente encontradas. Para evitar este problema a representação no nível de função pode englobar algumas unidades funcionais mais complexas, como somadores, multiplicadores e geradores de funções codificados diretamente no cromossomo, reduzindo significativamente seu tamanho. Embora esta abordagem tenha sido testada com sucesso em alguns problemas, alguns aspectos precisam ser discutidos para que seja possível aceitar esta solução como definitiva. Em primeiro lugar é necessário um

bom conhecimento no domínio do problema, para que as funções corretas sejam selecionadas, o que é extremamente difícil quando um problema possui um domínio pouco conhecido. Em segundo lugar a abordagem não apresenta escalabilidade para tratar problemas maiores e mais complexos sem que mais conhecimento de domínio sejam incluídos nas funções de seleção. Em terceiro lugar, uma vez que um nível de abstração é definido por meio da seleção das unidades funcionais, o espaço de busca fica limitado a este nível de abstração impedindo a busca por soluções inovadoras. Finalmente, e talvez o aspecto mais importante, é que as funções selecionadas são projetadas por metodologias tradicionais, levando em consideração a abordagem "*top-down*", o que adiciona pouca ajuda ao algoritmo evolutivo. Thompson (1996) argumenta que as representações no nível de função reduzem a evolutibilidade e defende fortemente que a evolução tradicional tem maior poder de busca além de um espaço de busca maior que a evolução proposta por Murakawa *et al.* (1996).

Baseado na representação ao nível de função, Torresen (2000) sugeriu que a evolução poderia, em níveis mais elevados, desenvolver módulos mais complexos baseados em uma série de módulos anteriormente desenvolvidos (funções automaticamente definidas) e, pela repetição do processo, melhorar a escalabilidade em função do processo incremental "bottom-up", estratégia denominada de aprendizado incremental. Embora esta abordagem possua resultados expressivos, necessita de um mecanismo para dividir uma tarefa complexa em sub-tarefas de complexidade menor, para iniciar o processo de evolução. Torresen (2000) demonstrou esta idéia aplicando-a em um problema de reconhecimento de caracteres em uma imagem. Primeiramente circuitos capazes de reconhecer um único caractere foram evoluídos, em blocos funcionais, e finalmente um circuito completo usando esses blocos foi evoluído, demonstrando um grande incremento no desempenho evolutivo. No entanto o experimento demonstrou que haverá uma grande dependência na decomposição do problema, pela aplicação da abordagem "top-down" além de demonstrar poucos benefícios à escalabilidade. A oportunidade para encontrar soluções inovadoras, nesta técnica, fica impossibilitada além de excluir problemas onde a decomposição é impossível. Hounsell e Arslan (2000) usaram esta abordagem para gerar um multiplicador de três bits, o problema foi decomposto em função dos bits de saída do circuito, e sub-circuitos foram gerados usando lógica tradicional. O circuito completo foi então evoluído. Kazadi *et al.* (2001) repetiram o experimento, porém usaram evolução também na geração dos sub-circuitos.

Lohn *et al.* (1999) compararam vários tipos de sistemas incrementais em relação à função de fitness. Três tipos de função de fitness dinâmicas foram comparadas com uma função de fitness estática. Na primeira função de fitness dinâmica a

complexidade do problema foi incrementada baseando-se no conhecimento do problema, na segunda o incremento na complexidade foi feito de modo proporcional ao fitness do melhor indivíduo da população, e no terceiro o controle do nível de complexidade foi submetido ao algoritmo, co-evoluindo o problema e a solução. Neste trabalho os autores mostraram que o fitness estático apresentou-se com a melhor eficiência de evolução, e concluíram que a característica de descontinuidade das funções de fitness reduz o desempenho evolutivo dos sistemas.

Outras abordagens buscando um melhor desempenho do processo evolutivo são apresentadas. Kajitani *et al.* (1996) propuseram o uso de cromossomos de tamanho variável com o objetivo de reduzir o espaço de busca necessário para encontrar uma solução. Esta abordagem mostrou desempenho superior ao cromossomo de tamanho fixo quando aplicado a um problema de reconhecimento de padrões. Zebulum *et al.* (1997) usou um enfoque similar evoluindo funções booleanas. Baseado na observação de que organismos complexos podem evoluir de organismos simples, a população foi gerada com cromossomos pequenos assumindo que existe uma correlação entre comportamento complexo e estrutura complexa. Harvey e Thompson (1996) demonstraram que esta premissa não é necessariamente verdadeira, argumentando que o comportamento complexo pode surgir da relação entre um sistema simples e um sistema complexo. Em ambos os casos a representação mapeou diretamente uma função booleana e a busca foi feita por evolução no primeiro caso e por uma heurística no segundo, nos dois experimentos apenas o espaço de busca era modificado, mantendo a capacidade de evolução inalterada.

Usar a evolução para explorar a representação, como uma meta-busca, em adição a exploração do espaço de busca é uma idéia atrativa, no entanto deve ser implementada visando permitir que a evolução explore espaços de busca com diferentes capacidades de evolução. A natureza, mais precisamente a biológica, demonstra que esta estratégia tem alto poder evolutivo. Dawkins (1987) aponta que a evolução biológica, com o passar do tempo, desenvolveu mecanismos para aumentar a capacidade de evolução, e usa estes mecanismos para gerar organismos cada vez mais complexos, demonstrando que existe uma evolução da capacidade de evolução, além de indicar que existem diferenças entre os mecanismos empregados para a evolução de sistemas simples e sistemas complexos.

Este capítulo buscou apresentar as principais motivações e desafios envolvidos na área de hardware evolutivo e contextualizar este trabalho. Os conceitos e decisões de projetos tomadas durante o desenvolvimento deste trabalho serão abordados com detalhes no próximo capítulo.

Capítulo 4

Considerações Práticas

4.1 Introdução

Na última década técnicas de computação evolutiva foram aplicadas ao projeto de circuitos eletrônicos criando uma nova área de pesquisa denominada *Hardware* Evolutivo (EHW) ou Eletrônica Evolutiva. Esta nova área busca o desenvolvimento de projetos automáticos de sistemas eletrônicos usando algoritmos de busca, baseados nos princípios da evolução natural e na tecnologia de dispositivos de *hardware* reconfigurável.

A possibilidade de reconfigurar dinamicamente o *hardware* de modo autônomo cria diversas expectativas de uso e superação das arquiteturas de computação tradicionais. Com isso, muitos pesquisadores desta área são motivados pelo desenvolvimento de *hardware* auto-reconfigurável e auto-adaptável que, em contraste com o *hardware* convencional, é projetado para se adaptar às mudanças de objetivos ou às mudanças no ambiente, através da sua habilidade de se reconfigurar dinamicamente.

Este capítulo apresenta de forma detalhada os conceitos e fundamentos necessários ao desenvolvimento de sistemas de *hardware* evolutivo, tais como a descrição de genomas, funções de avaliação e operadores genéticos. Um estudo de caso é apresentado para avaliar como as decisões de projeto influenciam no desempenho de um sistema EHW e na evolução de circuitos digitais básicos.

4.2 Descrição da Técnica

A primeira etapa em um processo evolutivo é a definição da representação das soluções. As decisões tomadas nesta etapa definem o mapeamento das possíveis soluções presentes no espaço de busca em uma estrutura de dados que possa ser manipulada computacionalmente. A codificação das soluções, ou especificação dos cromossomos, está intimamente relacionada ao problema a ser resolvido. Após a definição da representação inicia-se o processo evolutivo propriamente dito que opera sobre uma população de candidatos em paralelo, ou seja, realiza a busca em diferentes áreas do espaço de solução.

A implementação computacional do processo evolutivo inicia-se pela geração aleatória de uma população de genótipos, também denominados cromossomos, que é formada por um número pré-determinado de indivíduos. As soluções geradas são avaliadas através da função de aptidão e inicia-se o processo de seleção. O operador de seleção baseia-se no princípio de sobrevivência dos mais aptos, ou seleção natural, observado na evolução biológica. O critério de aptidão de um determinado indivíduo é dado pela função de avaliação que deve considerar as características do processo que se deseja otimizar. Esta função determinará a probabilidade de um indivíduo ser selecionado e contribuir para a criação de indivíduos em uma próxima geração.

Na etapa seguinte tem-se o processo de reprodução que ocorre através dos operadores de crossover e mutação. O operador de crossover cria descendentes, a partir de cromossomos pais, pela troca de informações contidas nos cromossomos, criando uma recombinação das características dos pais de modo que os filhos herdem estas características. O operador de mutação introduz diversidade genética à população, alterando arbitrariamente um ou mais gens do cromossomo, permitindo a introdução de novos elementos na população. Estes operadores são aplicados a taxas pré-estabelecidas.

A área de hardware evolutivo pode ser classificada de acordo com as seguintes propriedades, Zebulum *et al.* (1996): Quanto a abordagem evolutiva, relacionando a técnica de computação evolutiva que será aplicada ao processo de evolução. Quanto à natureza do projeto, as aplicações podem ser classificadas em analógica e digital. E finalmente, quanto ao processo de evolução, uma aplicação pode ser Intrínseca ou Extrínseca. As aplicações intrínsecas são aquelas em que a evolução ocorre no dispositivo reconfigurável e por este fato leva em consideração características físicas da implementação. Nas aplicações extrínsecas a avaliação é feita por simuladores de circuitos, como o SPICE, e somente os melhores indivíduos são car-

regados no dispositivo reconfigurável.

As duas classes têm vantagens e desvantagens. A evolução extrínseca fornece portabilidade com relação a plataforma de evolução, já a evolução intrínseca é extremamente específica. Por outro lado, algumas características importantes no processo de busca podem ser melhor avaliadas no processamento intrínseco. Em Stoica *et al.* (2004) foram evoluídos alguns indivíduos de forma intrínseca e outros de forma extrínseca no mesmo processo evolutivo, os autores chamaram esta evolução de evolução mixtrínseca.

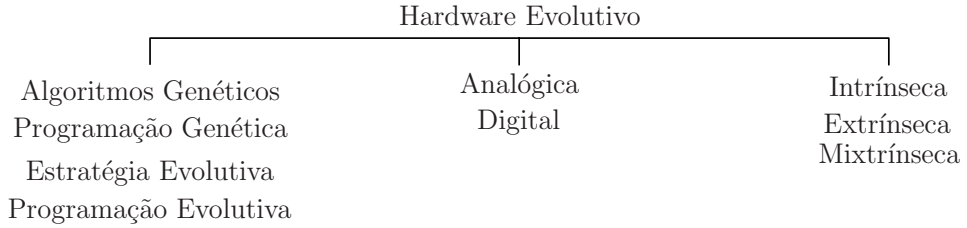


Figura 4.1: Classificação dos Sistemas Evolutivos (Zebulum, 1999)

4.3 Representação e Codificação de Genomas

A literatura apresenta duas representações principais para hardware evolutivo: a representação ao nível de portas, onde a unidade básica de circuito manipulada pelo sistema evolutivo é uma porta lógica, e a representação ao nível de funções, onde a unidade básica manipulada pelo algoritmo evolutivo é uma função digital mais complexa, como uma unidade lógica aritmética ou um produto de variáveis lógicas. Uma representação ao nível de funções corresponde a um mapeamento de alto nível entre o cromossomo e o circuito.

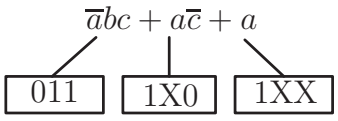
Serão apresentados cinco tipos diferentes de codificação de circuitos em hardware evolutivo, e estes quando apresentados são classificados como codificação ao nível de porta ou de função.

Genoma 1

A codificação apresentada por Zebulum (1999), é classificada como representação ao nível de funções. Nesta codificação toda função combinacional pode

ser expressa na forma de soma de produtos de variáveis lógicas, assim, os genótipos codificam todos os possíveis produtos das variáveis, completos ou incompletos.

No caso de um problema com n entradas cada gene terá n posições, os quais podem assumir três possíveis valores: 0, se a variável correspondente aparecer complementada; 1, se a variável correspondente não aparecer complementada; e 2, se a variável correspondente estiver ausente. Um exemplo desta codificação pode ser observado na Figura 4.2.



Cromossomo: 0111X01XX

Figura 4.2: Codificação proposta por Zebulum.

Esta codificação apresenta algumas desvantagens. A primeira desvantagem que pode-se observar é a obtenção de circuitos com, no máximo, dois níveis. Além disso, quanto maior o número de entradas na tabela mais lenta é a busca no espaço de soluções (cromossomo muito grande). Por fim pode-se observar também que algumas funções combinacionais são mais eficientemente representadas através de uma soma por operador ou-exclusivo (XOR) Miller e Thomson (1995), ao invés da soma de produtos e este tipo de solução não pode ser obtida através desta representação. Como vantagem desta representação pode-se citar sua simplicidade e facilidade de implementação.

Genoma 2

Na codificação apresentada por Louis e Rawlins (1991) e Coello *et al.* (1996) o genótipo codifica diretamente o arranjo de portas digitais que compõe o circuito e é classificado como representação ao nível de portas. Nesta representação cada circuito é codificado como uma matriz de duas dimensões na qual as colunas indicam o nível do elemento no circuito e as linhas indicam as diferentes portas dentro de um mesmo nível. Nesta codificação os genes constituintes dos cromossomos codificam as características de cada porta lógica: [Entrada 1, Entrada 2, Natureza da Porta]. São pré definidos tipos diferentes de portas, todas de duas entradas. As entradas das portas podem ser oriundas da entrada do circuito ou das saídas das portas existentes em qualquer um dos níveis anteriores à porta em questão, conforme pode

ser observado na Figura 4.3.

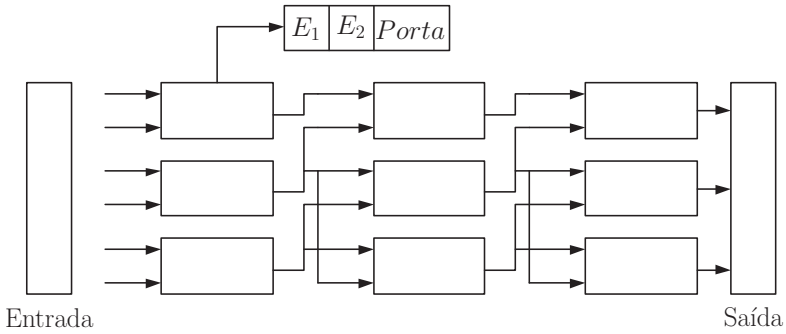


Figura 4.3: Codificação por Arranjo de Portas

O tamanho do cromossomo é proporcional ao número de células presentes na matriz usada para representar o circuito. Inicia-se a representação com matrizes de 5 linhas e 5 colunas. Se nenhuma solução factível para o problema for encontrada aumenta-se o número de colunas e o número de linhas respectivamente. Esta codificação, embora melhor que a anterior, também apresenta algumas desvantagens como a quantidade de níveis pré-determinados e portas somente com duas entradas.

Genoma 3

A terceira codificação abordada neste trabalho também é uma codificação ao nível de portas lógicas, mas os cromossomos são formados por expressões simbólicas LISP, Koza (1992), e a evolução neste caso ocorre através de programação genética. As expressões LISP adotadas são melhor mostradas graficamente como uma árvore. A expressão simbólica $F(A, B, C, D) = (+(*ABC)(*DE))$ tem a árvore mostrada na Figura 4.4. Esta codificação também apresenta algumas desvantagens como a necessidade do tratamento de soluções inconsistentes e operadores genéticos especiais. Além de não permitir, de modo direto, a criação de circuitos com mais de uma saída.

Genoma 4

A codificação proposta por Ferreira (2001) também utiliza programação genética para realizar o processo evolutivo e é uma alternativa à representação anterior. A árvore é percorrida por níveis e os gens são compostos por cabeça e cauda $(h+t)$.

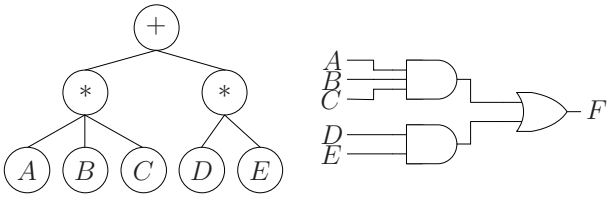


Figura 4.4: Codificação LISP

Para cada problema o tamanho da cabeça (h) é definido pelo projetista (de acordo com o número de níveis do circuito especifica-se quantos nós existem na árvore completa). O tamanho da cauda (t) é calculado $t = h(n-1)+1$, n é o número máximo de filhos permitidos por nó. Na cauda só é permitido a inclusão de folhas. A principal vantagem desta codificação é a possibilidade de se aplicar qualquer operador genético, sem restrição, não há geração de inconsistência. A desvantagem é a geração de circuitos com somente uma saída. A Figura 4.5 exemplifica esta codificação.

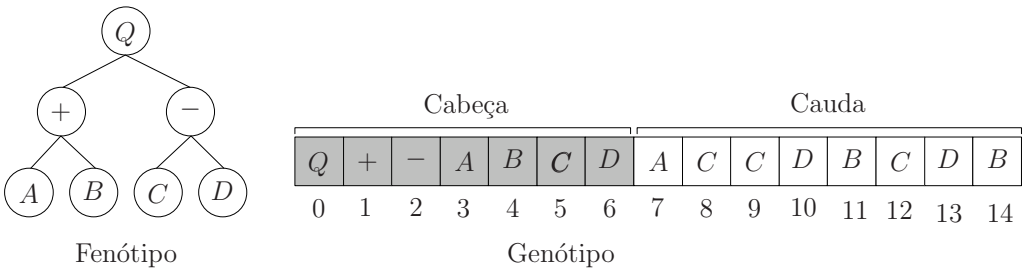


Figura 4.5: Codificação em Árvore Funcional

Genoma 5

A última codificação descrita também é muito utilizada por pesquisadores da área de EHW, Hartmann *et al.* (2005). É uma codificação ao nível de porta na qual os cromossomos são descritos como *netlist*, lista de portas lógicas e seus pontos da conexão em um projeto de circuito. Inicialmente uma biblioteca de portas é pré definida, com 2 ou mais entradas, e os cromossomos são gerados como na Figura Figura 4.6.

Nesta codificação os cromossomos podem ser manipulados por operadores genéticos simples, sem a geração de inconsistências. Permite múltiplas saídas e circuitos de tamanho variável, embora cada cromossomo possua tamanho fixo.

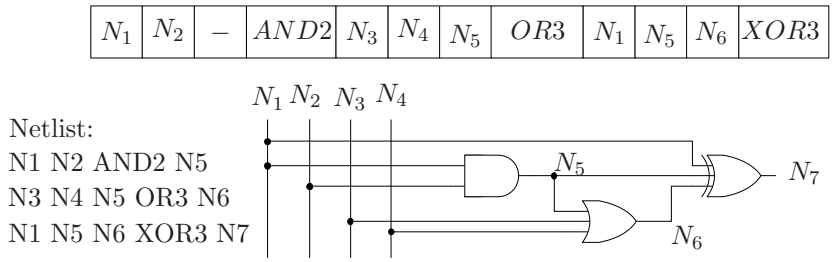


Figura 4.6: Representação por NETLIST

4.4 Operadores Genéticos

Os operadores genéticos, como crossover e mutação, introduzem diversidade genética à população, permitindo o surgimento de novos e diferentes indivíduos.

4.4.1 Crossover

Os genomas apresentados na seção 4.3 são codificados por cadeia binária, o que permite a aplicação direta dos métodos de crossover descritos na seção 2.3.1, Figura 2.6, com exceção do "genoma 3", que deve ser manipulado pelo operador de crossover descrito na seção 2.3.2, Figura 2.9.

4.4.2 Mutação

A mutação é um operador genético muito útil na manutenção da diversidade populacional e na variabilidade dos cromossomos em evolução. Nos genomas apresentados a alteração de um bit pode acarretar alterações distintas no fenótipo, podendo alterar uma função (porta lógica), ou a ligação entre os elementos do circuito. A Figura 4.7 ilustra este processo.

4.5 Função de Fitness

A função de *fitness* é uma característica comum a todas as técnicas de computação evolutiva, sendo geralmente formada por uma função matemática explícita

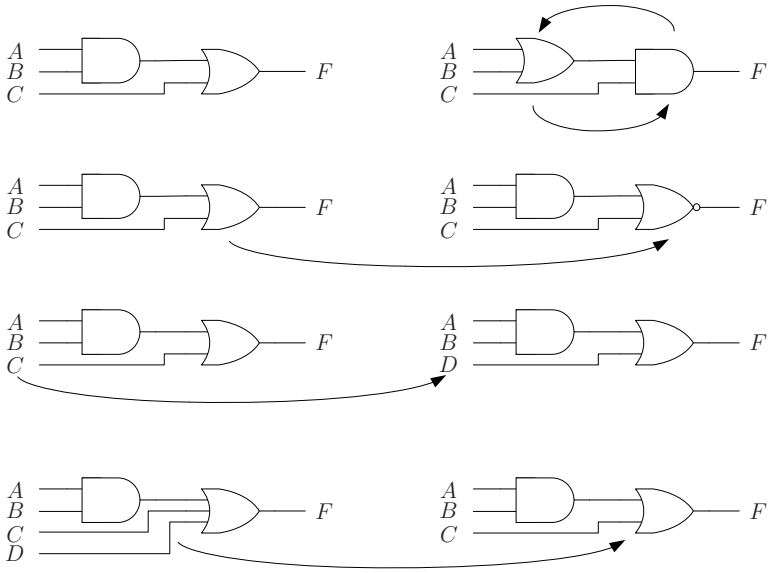


Figura 4.7: Tipos de Mutação

fortemente associada ao problema a ser solucionado.

Para problemas relacionados ao projeto de circuitos combinacionais sabe-se que um circuito lógico é a realização (implementação) de uma função booleana em hardware, e a complexidade de um circuito é uma função do número de portas do circuito, sendo que a complexidade de uma porta geralmente é uma função do número de entradas da porta.

Desse modo a função de fitness deve estar relacionada com a factibilidade do circuito codificado pelo cromossomo, e com a quantidade de literais contidos neste circuito, pois pela redução do número de literais de um circuito se reduz o número de entradas em cada porta e o número de portas no circuito, reduzindo sua complexidade Coello *et al.* (2001).

4.6 Comparação Experimental

Esta seção apresenta implementações de hardware evolutivo extrínseco com o objetivo de avaliar o desempenho de um sistema de hardware evolutivo na evolução de circuitos digitais básicos. Existem alguns métodos tradicionais para projetar circuitos lógicos, dentre eles destacam-se os Mapas de Karnaugh, Karnaugh (1953),

o método Quine-McCluskey, McCluskey (1956), e os processos de simplificação algébrica. Este experimento busca projetar circuitos digitais combinacionais usando hardware evolutivo evoluído através da utilização do genoma 5, apresentado na seção 4.3, sem o uso dos métodos tradicionais. Para o desenvolvimento da aplicação foi utilizado o Matlab/Simulink e o simulador PSPICE. Todo processo evolutivo ocorre em Matlab e, quando a função de fitness é calculada chama-se o PSPICE através do Simulink, usando a ferramenta XXX.

Um circuito detector de números primos foi evoluído com o intuito de investigar o desempenho da evolução em relação ao tipo de genoma, taxa de crossover, a taxa de mutação, tamanho da população e número de gerações. Para estudar o comportamento da evolução em relação a complexidade do problema o circuito foi evoluído em duas versões: a primeira versão consiste em um detector de números primos de quatro bits, Eq. 4.1, e a segunda de um detector de números primos de seis bits, Eq. 4.2. Cada solução encontrada foi avaliada no simulador de circuitos digitais $PSPICE$ (EHW extrínseco) e a melhor solução de cada genoma foi comparada com a solução gerada pelo método tradicional Quine-McCluskey, que será considerada, para efeitos de comparação, como a solução ótima em relação ao número de literais.

$$F_1 = \Sigma_{A,B,C,D}(2,3,5,7,11,13) \tag{4.1}$$

$$F_2 = \Sigma_{A,B,C,D,E,F}(2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61) \tag{4.2}$$

A codificação do genoma 5 para o circuito detector de números primos de 4 bits consiste de um cromossomo de 384 bits, e o circuito detector de números primos de 6 bits consiste de um cromossomo de 960. Ambos sistemas foram evoluídos com um Algoritmo Genético para populações com tamanhos de 500, 1000, 1500, 2000 e 2500 indivíduos, com taxas de crossover de 1,0%, 2,5%, 5,0%, 7,5%, 10,0%, 12,5%, 15,0%, 17,5%, 20%, e 22,5%. As taxas de crossover usadas variaram entre 70%, 75%, 80%, 85%, 90% e 95%. Cada combinação de parâmetros foi evoluída 100 vezes, e os resultados destas evoluções estão sintetizados nas Figuras 4.8, 4.9, 4.10 e 4.11 para o detector de 4 bits, e nas Figuras 4.12, 4.13 e 4.14 para o detector de 6 bits.

Os dados apresentados nestas figuras mostram que a evolução é altamente dependente dos parâmetros evolutivos, sendo que para o detector de 4 bits melhores resultados foram encontrados para taxa de mutação de 5%, taxa de crossover entre 80% e 95% e populações acima de 1000 indivíduos. Para o detector de 6 bits os melhores resultados foram encontrados para taxa de mutação de $7,5\%$, taxa de crossover entre 80% e 90% e populações acima de 1500 indivíduos.

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	45%	48%	49%	54%	59%	58%
2,50%	89%	88%	85%	88%	90%	90%
5,00%	100%	100%	100%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	100%	100%	100%	100%	100%	100%
12,50%	100%	100%	100%	100%	100%	100%
15,00%	90%	94%	95%	94%	94%	96%
17,50%	80%	79%	68%	84%	70%	86%
20,00%	76%	66%	75%	66%	64%	69%
22,50%	49%	70%	59%	59%	50%	59%

(a)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	66%	68%	64%	68%	69%	69%
2,50%	99%	100%	99%	100%	99%	100%
5,00%	100%	100%	100%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	100%	100%	100%	100%	100%	100%
12,50%	100%	100%	99%	99%	100%	100%
15,00%	94%	94%	94%	94%	94%	93%
17,50%	100%	93%	99%	90%	99%	98%
20,00%	96%	88%	94%	90%	90%	89%
22,50%	86%	89%	78%	84%	81%	81%

(b)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	78%	76%	80%	83%	83%	88%
2,50%	100%	100%	100%	100%	100%	100%
5,00%	100%	100%	100%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	100%	100%	100%	100%	100%	100%
12,50%	100%	100%	100%	100%	100%	100%
15,00%	94%	93%	99%	99%	94%	94%
17,50%	98%	99%	100%	98%	99%	96%
20,00%	98%	99%	96%	98%	95%	98%
22,50%	83%	92%	94%	94%	93%	59%

(c)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	86%	88%	86%	82%	93%	90%
2,50%	100%	98%	100%	100%	100%	100%
5,00%	100%	100%	100%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	100%	100%	100%	100%	100%	100%
12,50%	100%	100%	100%	100%	100%	100%
15,00%	90%	90%	99%	96%	93%	90%
17,50%	100%	100%	99%	100%	100%	100%
20,00%	94%	100%	100%	98%	100%	98%
22,50%	100%	94%	96%	98%	96%	94%

(d)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	90%	90%	92%	96%	98%	98%
2,50%	100%	100%	100%	100%	100%	100%
5,00%	100%	100%	100%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	100%	100%	98%	100%	100%	100%
12,50%	98%	100%	100%	100%	100%	100%
15,00%	90%	90%	96%	97%	90%	90%
17,50%	100%	100%	100%	100%	100%	100%
20,00%	100%	100%	100%	100%	98%	98%
22,50%	98%	98%	100%	97%	100%	100%

(e)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	72,9%	74,1%	74,2%	76,5%	80,3%	80,5%
2,50%	97,6%	97,2%	96,8%	97,6%	97,8%	98,0%
5,00%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
7,50%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%
10,00%	100,0%	100,0%	99,6%	100,0%	100,0%	100,0%
12,50%	99,6%	100,0%	99,8%	99,8%	100,0%	100,0%
15,00%	91,5%	92,3%	96,5%	96,0%	92,9%	92,6%
17,50%	95,5%	94,1%	93,2%	94,3%	93,5%	96,1%
20,00%	92,7%	90,5%	93,1%	90,3%	89,4%	90,1%
22,50%	83,1%	88,6%	85,4%	86,4%	83,9%	78,6%

(f)

Figura 4.8: Capacidade para Encontrar Solução Factível - Detector de 4 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	673	693	513	349	347	373
	2,50%	183	179	161	153	138	144
	5,00%	311	269	278	246	261	247
	7,50%	383	404	307	346	272	277
	10,00%	801	630	769	618	569	549
	12,50%	1001	862	928	792	837	947
	15,00%	1676	1414	1503	1450	1281	1695
	17,50%	1727	1549	1676	1634	1629	1760
	20,00%	1526	1703	1969	1916	1736	1716
	22,50%	2003	1555	2269	2439	2615	2492

(a)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	139	116	126	92	115	63
	2,50%	129	111	117	105	117	96
	5,00%	213	172	157	148	173	148
	7,50%	283	198	210	205	197	192
	10,00%	414	375	365	349	328	330
	12,50%	585	429	520	456	422	516
	15,00%	822	870	852	878	886	980
	17,50%	1940	1485	1170	1274	1180	1282
	20,00%	1530	1471	1961	1478	1448	1702
	22,50%	2759	2128	1982	2244	2144	2152

(b)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	76	72	70	60	62	58
	2,50%	86	85	95	91	83	80
	5,00%	128	141	132	123	129	125
	7,50%	162	146	169	169	175	167
	10,00%	251	245	284	277	282	306
	12,50%	378	361	405	393	355	362
	15,00%	577	659	681	629	799	732
	17,50%	958	1155	939	1104	1272	922
	20,00%	1609	1369	1132	1344	1260	1370
	22,50%	1670	1837	1906	1929	1926	2143

(c)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	66	62	73	69	61	61
	2,50%	79	76	79	72	78	78
	5,00%	123	127	115	112	122	117
	7,50%	150	143	144	149	154	167
	10,00%	232	242	236	264	243	262
	12,50%	327	287	318	311	337	342
	15,00%	306	552	573	591	653	616
	17,50%	872	748	813	943	899	936
	20,00%	1191	982	1154	1095	1200	1190
	22,50%	1738	1642	1654	1779	1829	1883

(d)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	57	62	56	61	57	57
	2,50%	71	70	82	72	71	73
	5,00%	111	110	112	117	117	120
	7,50%	142	138	142	147	147	150
	10,00%	249	235	275	240	248	258
	12,50%	315	288	303	317	275	361
	15,00%	534	511	519	513	560	629
	17,50%	796	757	760	996	806	971
	20,00%	902	1071	1013	1165	1235	1038
	22,50%	1347	1593	1702	1753	1487	1712

(e)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	202	201	168	126	128	122
	2,50%	110	104	107	99	97	94
	5,00%	177	164	159	149	160	151
	7,50%	224	206	194	203	189	191
	10,00%	389	345	386	350	334	341
	12,50%	521	445	495	454	445	506
	15,00%	783	801	826	812	836	930
	17,50%	1259	1139	1072	1190	1157	1174
	20,00%	1352	1319	1446	1400	1376	1403
	22,50%	1903	1751	1903	2029	2000	2076

(f)

Figura 4.9: Geração Média para Encontrar Solução Factível - Detector de 4 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	12%	15%	9%	15%	15%	10%
	2,50%	23%	20%	20%	21%	13%	21%
	5,00%	85%	88%	85%	77%	80%	86%
	7,50%	53%	65%	50%	66%	68%	82%
	10,00%	32%	0%	0%	2%	1%	2%
	12,50%	0%	0%	0%	0%	0%	0%
	15,00%	0%	0%	0%	0%	0%	0%
	17,50%	0%	0%	0%	0%	0%	0%
	20,00%	0%	0%	0%	0%	0%	0%
	22,50%	0%	0%	0%	0%	0%	0%

(a)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	3%	3%	4%	4%	8%	4%
	2,50%	24%	23%	35%	25%	39%	49%
	5,00%	98%	98%	100%	100%	100%	100%
	7,50%	93%	98%	95%	99%	99%	100%
	10,00%	0%	0%	0%	0%	0%	0%
	12,50%	1%	0%	0%	0%	0%	0%
	15,00%	0%	0%	0%	0%	0%	0%
	17,50%	0%	0%	0%	0%	0%	0%
	20,00%	0%	0%	0%	0%	0%	0%
	22,50%	0%	0%	0%	0%	0%	0%

(b)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	3%	1%	9%	9%	16%	26%
	2,50%	36%	31%	55%	69%	76%	85%
	5,00%	100%	100%	100%	100%	100%	100%
	7,50%	99%	100%	100%	100%	100%	100%
	10,00%	0%	5%	0%	1%	0%	0%
	12,50%	0%	0%	100%	1%	0%	0%
	15,00%	0%	0%	0%	0%	0%	0%
	17,50%	0%	0%	0%	0%	0%	0%
	20,00%	0%	0%	0%	0%	0%	0%
	22,50%	0%	0%	0%	0%	0%	0%

(c)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	24%	10%	19%	18%	33%	58%
	2,50%	67%	49%	83%	82%	91%	94%
	5,00%	100%	100%	100%	98%	100%	100%
	7,50%	100%	100%	99%	100%	100%	100%
	10,00%	0%	0%	0%	0%	0%	0%
	12,50%	0%	0%	0%	0%	0%	0%
	15,00%	0%	0%	0%	0%	0%	0%
	17,50%	0%	0%	0%	0%	0%	0%
	20,00%	0%	0%	0%	0%	0%	0%
	22,50%	0%	0%	0%	0%	0%	0%

(d)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	14%	20%	40%	56%	60%	72%
	2,50%	80%	84%	92%	97%	98%	100%
	5,00%	100%	100%	100%	100%	100%	100%
	7,50%	100%	100%	98%	99%	100%	100%
	10,00%	0%	0%	0%	0%	0%	0%
	12,50%	0%	0%	0%	0%	0%	0%
	15,00%	0%	0%	0%	0%	0%	0%
	17,50%	0%	0%	0%	0%	0%	0%
	20,00%	0%	0%	0%	0%	0%	0%
	22,50%	20%	0%	0%	0%	0%	0%

(e)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	11,2%	9,8%	16,2%	20,4%	26,4%	34,0%
	2,50%	46,0%	41,4%	57,0%	58,8%	63,4%	69,8%
	5,00%	96,6%	97,2%	97,0%	95,0%	96,0%	97,2%
	7,50%	89,0%	92,6%	88,4%	92,8%	93,4%	96,4%
	10,00%	6,4%	1,0%	0,0%	0,6%	0,2%	0,4%
	12,50%	0,2%	0,0%	20,0%	0,2%	0,0%	0,0%
	15,00%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
	17,50%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
	20,00%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
	22,50%	4,0%	0,0%	0,0%	0,0%	0,0%	0,0%

(f)

Figura 4.10: Capacidade para Encontrar Solução Ótima - Detector de 4 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	1344	1266	1498	1350	832	1274
	2,50%	429	498	661	423	479	587
	5,00%	2615	2719	2518	2493	2341	2406
	7,50%	3221	3271	3447	3110	3424	3398
	10,00%	4720	X	X	4073	4780	4608
	12,50%	X	X	X	X	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(a)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	639	840	761	382	508	76
	2,50%	946	775	1017	1098	932	1322
	5,00%	2250	1861	2057	1748	1618	1583
	7,50%	3498	3507	3304	2940	2892	2809
	10,00%	X	X	X	X	X	X
	12,50%	4829	X	X	X	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(b)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	140	479	1218	1828	640	939
	2,50%	2065	1967	1224	1346	931	907
	5,00%	1660	1784	1577	1476	1318	1205
	7,50%	2980	3174	2765	2607	2539	2347
	10,00%	X	X	X	3796	X	X
	12,50%	X	X	X	3869	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(c)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	773	623	1350	1488	1113	290
	2,50%	1227	1531	951	591	670	665
	5,00%	1371	1490	1342	1151	1197	1162
	7,50%	2682	2761	2569	2207	2286	2173
	10,00%	X	X	X	X	X	X
	12,50%	X	X	X	X	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(d)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	913	552	1195	463	350	233
	2,50%	1051	1031	759	664	513	309
	5,00%	1318	1251	1264	1188	1140	1065
	7,50%	2502	2442	2407	2317	2227	2259
	10,00%	X	X	X	X	X	X
	12,50%	X	X	X	X	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(e)

		Crossover					
		70%	75%	80%	85%	90%	95%
Mutação	1,00%	762	752	1204	1102	689	562
	2,50%	1144	1160	922	824	705	758
	5,00%	1843	1821	1752	1611	1523	1484
	7,50%	2977	3031	2898	2636	2674	2597
	10,00%	X	X	X	X	X	X
	12,50%	X	X	X	X	X	X
	15,00%	X	X	X	X	X	X
	17,50%	X	X	X	X	X	X
	20,00%	X	X	X	X	X	X
	22,50%	X	X	X	X	X	X

(f)

Figura 4.11: Geração Média para Encontrar Solução Ótima - Detector de 4 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	43%	45%	46%	49%	59%	55%
2,50%	84%	81%	79%	84%	90%	83%
5,00%	100%	94%	86%	92%	92%	95%
7,50%	98%	96%	100%	100%	100%	90%
10,00%	100%	96%	97%	96%	100%	98%
12,50%	97%	98%	93%	91%	91%	98%
15,00%	84%	89%	88%	89%	92%	94%
17,50%	75%	69%	66%	77%	61%	81%
20,00%	70%	58%	68%	59%	54%	69%
22,50%	43%	68%	49%	59%	50%	50%

(a)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	60%	59%	60%	65%	68%	62%
2,50%	93%	93%	92%	98%	93%	99%
5,00%	98%	100%	97%	98%	99%	99%
7,50%	98%	99%	100%	100%	100%	97%
10,00%	98%	99%	98%	100%	98%	97%
12,50%	98%	92%	94%	94%	100%	94%
15,00%	91%	90%	90%	92%	90%	92%
17,50%	92%	86%	96%	87%	95%	92%
20,00%	95%	85%	89%	81%	86%	81%
22,50%	85%	89%	73%	78%	72%	73%

(b)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	74%	67%	73%	78%	73%	79%
2,50%	99%	100%	97%	100%	97%	98%
5,00%	100%	98%	100%	100%	98%	99%
7,50%	100%	100%	100%	100%	100%	99%
10,00%	99%	100%	100%	100%	99%	98%
12,50%	98%	99%	97%	99%	99%	98%
15,00%	87%	92%	90%	99%	87%	94%
17,50%	93%	91%	94%	95%	89%	96%
20,00%	93%	97%	90%	89%	87%	93%
22,50%	80%	85%	84%	87%	89%	56%

(c)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	79%	80%	77%	76%	88%	87%
2,50%	97%	97%	99%	97%	97%	98%
5,00%	100%	99%	98%	98%	100%	99%
7,50%	99%	100%	100%	100%	100%	99%
10,00%	100%	100%	98%	99%	99%	98%
12,50%	99%	100%	98%	97%	97%	100%
15,00%	86%	87%	98%	86%	87%	90%
17,50%	99%	99%	89%	99%	97%	100%
20,00%	93%	93%	96%	95%	97%	97%
22,50%	97%	85%	87%	91%	89%	92%

(d)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	86%	90%	89%	93%	97%	97%
2,50%	97%	97%	99%	99%	98%	97%
5,00%	100%	98%	98%	100%	100%	100%
7,50%	100%	100%	100%	100%	100%	100%
10,00%	99%	98%	97%	100%	98%	98%
12,50%	97%	97%	98%	100%	100%	98%
15,00%	83%	86%	94%	88%	81%	89%
17,50%	97%	99%	96%	96%	96%	97%
20,00%	98%	96%	100%	99%	91%	95%
22,50%	90%	89%	98%	88%	96%	97%

(e)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	68,4%	68,2%	69,0%	72,2%	77,0%	76,0%
2,50%	94,0%	93,6%	93,2%	95,6%	95,0%	95,0%
5,00%	99,6%	97,8%	95,8%	97,6%	97,8%	98,4%
7,50%	99,0%	99,0%	100,0%	100,0%	100,0%	97,0%
10,00%	99,2%	98,6%	98,0%	99,0%	98,8%	97,8%
12,50%	97,8%	97,2%	96,0%	96,2%	97,4%	97,6%
15,00%	86,2%	88,8%	92,0%	90,8%	87,4%	91,8%
17,50%	91,2%	88,8%	88,2%	90,8%	87,6%	93,2%
20,00%	89,8%	85,8%	88,6%	84,6%	83,0%	87,0%
22,50%	79,0%	83,2%	78,2%	80,6%	79,2%	73,6%

(f)

Figura 4.12: Capacidade para Encontrar Solução Factível - Detector de 6 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	2154	2366	1673	1151	1151	1238
2,50%	567	577	531	518	426	447
5,00%	1043	926	847	834	857	856
7,50%	1313	1381	985	1163	854	841
10,00%	2638	2008	2528	2048	1986	1894
12,50%	3428	2763	2888	2634	2845	3241
15,00%	5606	4438	5151	4662	3990	5123
17,50%	5990	5282	5520	5677	5444	5578
20,00%	4886	5483	6496	6435	5286	5575
22,50%	6326	5321	7717	7681	8369	8548

(a)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	454	376	436	285	355	189
2,50%	410	354	404	337	400	291
5,00%	645	571	532	499	591	450
7,50%	876	631	658	624	688	581
10,00%	1341	1135	1105	1216	997	1081
12,50%	1921	1321	1682	1410	1436	1645
15,00%	2800	3038	2784	2934	2722	2989
17,50%	6685	4909	4042	4209	3799	4115
20,00%	4630	4897	5931	4834	4776	5895
22,50%	8321	7109	6645	7580	7092	6636

(b)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	237	229	231	187	190	177
2,50%	262	271	312	281	273	244
5,00%	390	450	425	383	417	414
7,50%	495	507	510	589	548	538
10,00%	874	807	948	964	951	940
12,50%	1232	1096	1346	1368	1141	1158
15,00%	2008	2264	2128	2129	2476	2328
17,50%	3107	3938	3107	3718	3894	3054
20,00%	5298	4742	3488	4379	3849	4780
22,50%	5416	5845	6102	6619	6034	6547

(c)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	227	194	236	236	206	212
2,50%	244	240	260	235	244	244
5,00%	386	425	392	363	397	399
7,50%	521	449	453	520	535	516
10,00%	719	835	744	857	794	843
12,50%	1110	868	1057	1081	1156	1047
15,00%	1034	1930	1772	1782	2058	1850
17,50%	2660	2505	2715	2887	2779	3122
20,00%	3929	3102	3876	3365	3803	3864
22,50%	5989	5026	5532	6013	6333	5830

(d)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	192	192	182	190	172	194
2,50%	239	218	265	220	235	250
5,00%	356	349	366	386	397	383
7,50%	487	415	496	493	451	517
10,00%	766	755	952	799	847	796
12,50%	1050	942	1039	1090	898	1243
15,00%	1613	1755	1616	1733	1875	2111
17,50%	2715	2516	2565	3198	2453	3374
20,00%	2860	3387	3528	3722	4042	3184
22,50%	4519	5056	5258	5790	4752	5358

(e)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	653	671	552	410	415	402
2,50%	344	332	354	318	316	295
5,00%	564	544	512	493	532	500
7,50%	738	677	620	678	615	599
10,00%	1268	1108	1255	1177	1115	1111
12,50%	1748	1398	1602	1516	1495	1667
15,00%	2612	2685	2690	2648	2624	2880
17,50%	4231	3830	3590	3938	3674	3849
20,00%	4321	4322	4664	4547	4351	4660
22,50%	6114	5671	6251	6737	6516	6584

(f)

Figura 4.13: Geração Média para Encontrar Solução Factível - Detector de 6 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	32%	34%	36%	37%	44%	41%
2,50%	65%	63%	59%	64%	70%	64%
5,00%	75%	73%	67%	71%	79%	72%
7,50%	77%	75%	75%	82%	85%	85%
10,00%	0%	0%	0%	0%	0%	0%
12,50%	0%	0%	0%	0%	0%	0%
15,00%	0%	0%	0%	0%	0%	0%
17,50%	0%	0%	0%	0%	0%	0%
20,00%	0%	0%	0%	0%	0%	0%
22,50%	0%	0%	0%	0%	0%	0%

(a)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	45%	45%	47%	49%	52%	46%
2,50%	73%	70%	71%	75%	70%	78%
5,00%	76%	79%	83%	84%	85%	74%
7,50%	73%	78%	78%	79%	89%	96%
10,00%	0%	0%	0%	0%	0%	0%
12,50%	0%	0%	0%	0%	0%	0%
15,00%	0%	0%	0%	0%	0%	0%
17,50%	0%	0%	0%	0%	0%	0%
20,00%	0%	0%	0%	0%	0%	0%
22,50%	0%	0%	0%	0%	0%	0%

(b)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	56%	53%	56%	59%	57%	62%
2,50%	74%	75%	75%	75%	72%	76%
5,00%	77%	76%	84%	100%	83%	75%
7,50%	100%	76%	100%	100%	100%	75%
10,00%	0%	0%	0%	0%	0%	0%
12,50%	0%	0%	0%	0%	0%	0%
15,00%	0%	0%	0%	0%	0%	0%
17,50%	0%	0%	0%	0%	0%	0%
20,00%	0%	0%	0%	0%	0%	0%
22,50%	0%	0%	0%	0%	0%	0%

(c)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	62%	62%	60%	57%	65%	66%
2,50%	72%	72%	74%	73%	76%	74%
5,00%	77%	74%	76%	86%	100%	85%
7,50%	76%	75%	100%	100%	100%	85%
10,00%	0%	0%	0%	0%	0%	0%
12,50%	0%	0%	0%	0%	0%	0%
15,00%	0%	0%	0%	0%	0%	0%
17,50%	0%	0%	0%	0%	0%	0%
20,00%	0%	0%	0%	0%	0%	0%
22,50%	0%	0%	0%	0%	0%	0%

(d)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	64%	69%	66%	70%	76%	73%
2,50%	72%	74%	73%	75%	73%	77%
5,00%	79%	76%	73%	75%	75%	87%
7,50%	77%	75%	100%	100%	100%	100%
10,00%	0%	0%	0%	0%	0%	0%
12,50%	0%	0%	0%	0%	0%	0%
15,00%	0%	0%	0%	0%	0%	0%
17,50%	0%	0%	0%	0%	0%	0%
20,00%	0%	0%	0%	0%	0%	0%
22,50%	0%	0%	0%	0%	0%	0%

(e)

Mutação	Crossover 70%	75%	80%	85%	90%	95%
1,00%	51,8%	52,6%	52,8%	54,4%	58,8%	57,6%
2,50%	71,3%	70,9%	70,7%	72,2%	72,2%	73,7%
5,00%	76,7%	75,4%	76,8%	83,3%	84,3%	78,6%
7,50%	80,6%	75,7%	90,6%	92,2%	94,8%	88,3%
10,00%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
12,50%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
15,00%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
17,50%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
20,00%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
22,50%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%

(f)

Figura 4.14: Capacidade para Encontrar Solução Ótima - Detector de 6 Bits. População: (a) 500 indivíduos, (b) 1000 indivíduos, (c) 1500 indivíduos, (d) 2000 indivíduos, (e) 2500 indivíduos e (f) Média.

Após a análise dos resultados encontrados e a obtenção da experiência necessária para a implementação de variações da técnica, buscou-se uma aplicação prática inovadora para que a análise obtida viesse a contribuir com a nova área de aplicação para EHW.

A seção seguinte apresenta a técnica de hardware evolutivo como uma nova abordagem para projeto automático de sistemas de controle, visando a obtenção rápida e eficiente de uma solução inicial para problemas complexos, como controle não linear. O principal objetivo é demonstrar a eficiência desta nova área de estudo quando aplicado a sistemas não lineares.

Capítulo 5

Aplicação

5.1 Introdução

Esta seção apresenta comparações de desempenho entre duas abordagens diferentes para projetos de controladores. Os controladores apresentados aqui são estudos de casos com o intuito de verificar a eficiência da técnica proposta. Controladores obtidos por hardware evolutivo extrínseco e controladores com arquitetura linear, P e PD, com ganhos sintonizados por meio de Algoritmos Genéticos (AGs), são avaliados quando aplicados ao controle de posição de um pêndulo não linear.

A resposta dinâmica de um sistema não linear é também dependente da intensidade dos sinais de entrada, portanto, dadas as especificações de projeto, é necessário encontrar um controlador capaz de garantir que determinadas especificações sejam alcançadas toda vez que a entrada está dentro de uma faixa de operação. Algumas aproximações empregam compensadores lineares em série com ganhos, que são funções das amplitudes de entrada, visando compensar um sistema não linear. No entanto a síntese dessas funções não é simples nem fácil, Marquez (2003). O uso do hardware evolutivo para obtenção do controlador é uma abordagem alternativa de projeto e tem como objetivo modelar o comportamento não linear do sistema e sintetizá-lo em um circuito digital combinacional.

5.2 Pêndulo Amortecido

Por ser um problema clássico em controle o sistema dinâmico utilizado para a obtenção dos resultados apresentados nesta seção é o pêndulo amortecido. Este é um sistema mecânico relativamente simples, conforme pode ser observado na Figura 5.1. A equação dinâmica deste sistema pode ser apresentada de duas formas:

- Sistema Linear, com linearização em torno da origem:

$$ml^2\ddot{\theta} + kl^2\dot{\theta} + mgl\theta = \tau \tag{5.1}$$

- Sistema Não Linear:

$$ml^2\ddot{\theta} + kl^2\dot{\theta} + mgl\sin(\theta) = \tau \tag{5.2}$$

Considerou-se um pêndulo com massa $m = 0.210Kg$, concentrada em CM $g = 9.82m/s^2$ a aceleração da gravidade, $k = 0.3Nms/rad$ a viscosidade e $l = 0.285m$ o comprimento da haste do pêndulo. O programa Matlab/Simulink foi usado para a implementação do modelo, simulação e avaliação da performance do sistema de controle, Figura 5.2.

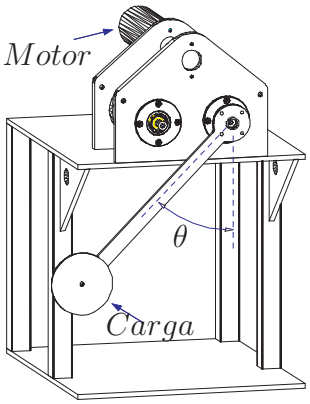


Figura 5.1: Pêndulo Não Linear

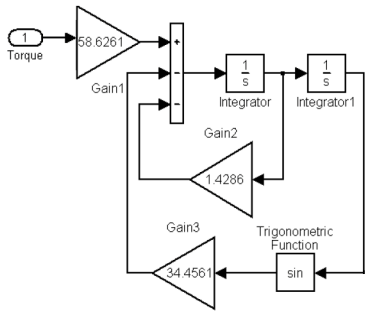


Figura 5.2: Modelo Matlab/Simulink

5.3 Controladores PID

Os controladores PID (Proporcional-Integral-Derivativo), e suas variações, são métodos convencionais de controle usados em muitos processos automatizados. Entre os principais motivos de sua extensa utilização pode-se mencionar características como: estrutura simples, robustez, reduzido número de parâmetros a serem configurados, conhecimento intuitivo sobre o desempenho dos parâmetros e fácil implementação em sistemas computacionais discretos. Entretanto, a teoria de controle clássica, usada para a resolução de problemas quando o processo é definido adequadamente, falha no tratamento de alguns processos complexos, devido principalmente a presença de não-linearidades e comportamentos variantes no tempo.

O PID discreto pode ser implementado de diferentes formas. Diferentes estruturas correspondem a diferentes controladores PIDs, Aström e Hägglund (1988). As estruturas $PIDs$ discretas mais comuns são descritas por:

- Controlador Proporcional

$$u(n) = K_P e(n) \tag{5.3}$$

- Controlador Proporcional-Derivativo

$$u(n) = \left[K_P + \frac{K_D}{T_S}\right] e(n) - \frac{K_D}{T_S} e(n-1) \tag{5.4}$$

- Controlador Proporcional-Integral

$$u(n) = u(n-1) + \left[K_P + \frac{K_I T_S}{2}\right] e(n) + \left[-K_P + \frac{K_I T_S}{2}\right] e(n-1) \tag{5.5}$$

- Controlador Proporcional-Integral-Derivativo

$$\begin{aligned} u(n) &= u(n-1) + \left[K_P + \frac{K_D}{T_S} + \frac{K_I T_S}{2}\right] e(n) + \left[\frac{K_I T_S}{2} - K_P - 2\frac{K_D}{T_S}\right] e(n-1) + \\ &\quad + \frac{K_D}{T_S} e(n-2) \end{aligned} \tag{5.6}$$

sendo que $e(t)$ descreve o sinal de erro, dado por:

$$e(n) = r(n) - y(n) \tag{5.7}$$

$r(t)$ é o sinal referência e $y(t)$ é a variável do sistema. K_P, K_D e K_I são os ganhos proporcional, diferencial e integral, respectivamente. T_S é o tempo amostragem.

A sintonia dos parâmetros de controle de um controlador PID, Eq. (5.6), é extremamente importante, pois o desempenho do controle depende fortemente destes parâmetros. Embora possa ser uma tarefa simples em alguns processos, em algumas plantas a sintonia pode consumir um tempo considerável, ou até mesmo tornar-se extremamente complicada devido a presença de efeitos não lineares. Nos últimos anos uma série de técnicas de sintonia para controladores clássicos, derivadas do trabalho de Ziegler e Nichols, têm sido apresentadas na literatura, Goodwin *et al.* (2001).

Geralmente estas técnicas consistem em selecionar adequadamente um ou mais parâmetros para melhorar o desempenho do sistema. Se uma medida ou um índice de desempenho puder ser expressa matematicamente, o problema pode ser resolvido pela seleção adequada dos parâmetros. O sistema resultante é denominado ótimo com respeito ao critério selecionado.

Os índices de desempenho baseados em integrais são clássicos na teoria de controle, o valor do índice de erro ISE (*Integral of the Square Error*), Eq. 5.8, foi usado como critério de minimização dos controladores desenvolvidos durante este trabalho. Este índice foi escolhido por penalizar fortemente valores altos na variável de erro, e por possuir um bom tratamento matemático.

$$J_{ISE} = \int_0^T e(t)^2 dt \tag{5.8}$$

5.3.1 Função de *Fitness*

Em um algoritmo evolutivo, durante o processo de seleção, todas as soluções geradas são avaliadas pela função de *fitness*. A função de *fitness* avalia a solução evoluída baseado-se em seu comportamento. Neste experimento a função de *fitness* é baseada no valor do índice de erro J_{ISE}, e o objetivo do processo evolutivo é a minimização deste valor.

$$Fitness = f(J_{ISE}) = \frac{1}{1 + J_{ISE}} \tag{5.9}$$

5.3.2 Codificação

Um dos elementos mais importantes para o processo evolutivo é a definição do cromossomo. Para a sintonia do controlador PID a codificação binária tradicional foi usada, pois os ganhos associados ao controlador são facilmente convertidos em informação genética, Jungbeck (2001) , Figura 5.3.

$$(K_P, K_D, K_I) = \underbrace{010\cdots00}_{K_P}\underbrace{101\cdots01}_{K_D}\underbrace{111\cdots10}_{K_I}$$

Figura 5.3: Codificação dos Ganhos do Controlador

5.4 Controlador Hardware Evolutivo

No desenvolvimento do Controlador por Hardware Evolutivo apenas o critério funcional foi usado como elemento de projeto, ou seja, apenas as entradas e as saídas produzidas pelo sistema foram consideradas (caixa preta). A estrutura interna do circuito digital do controlador não foi considerada, desse modo apenas ações que produzem uma saída visível ao sistema são cobertas pela função de *fitness*.

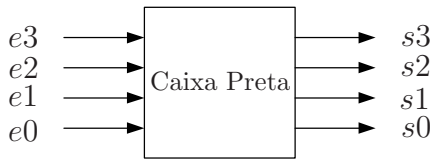


Figura 5.4: Sistema tipo Caixa Preta

Dentre os algoritmos evolutivos o método mais usual para o desenvolvimento de hardware evolutivo é o algoritmo genético. O algoritmo genético clássico opera sobre uma população de tamanho constante formada por strings de tamanho fixo denominadas cromossomos, Thompson (1996). Cada cromossomo codifica um conjunto de parâmetros que representa uma coleção de componentes eletrônicos e suas interconexões, assim, cada conjunto de valores que formam um cromossomo representa um circuito eletrônico. O conjunto de todas as combinações possíveis dos parâmetros define o espaço de busca do problema, Gordon e Bentley (2005). Cada conjunto de parâmetros no espaço de busca codifica um único circuito o qual é associado a um código genético ou cromossomo. O cromossomo é chamado de genótipo e o circuito é o fenótipo.

5.4.1 Codificação

O genótipo determina como uma função booleana é representada, utilizando as portas lógicas da Figura 5.5, e como estas portas estão interconectadas. O cromossomo é formado por um conjunto de n segmentos, cada segmento representa uma porta lógica e é um nó do circuito digital formado pela função booleana, além de conter as informações sobre os nós que devem ser conectados às entradas da porta lógica por ele representada. Cada segmento é formado por uma palavra de 15 bits, conforme descrito pela Tabela 5.1.

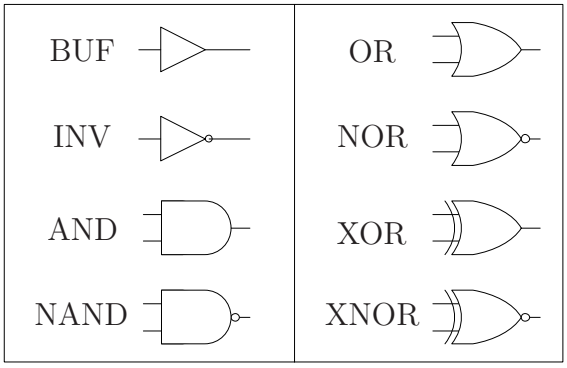


Figura 5.5: Funções dos Nós

Tabela 5.1: Segmento do cromossomo

Bits	Significado
0-2	Função do Nó
3-8	Conexão da Primeira Entrada
9-14	Conexão da Segunda Entrada

Desse modo uma função boolena pode ser representada em um cromossomo pela codificação binária e este pode ser facilmente traduzido em um netlist, sendo que cada segmento do cromossomo representa uma linha do netlist. Cada linha representa uma porta e fornece informações como: o tipo de porta usada e quais suas fontes de entrada. As conexões de uma porta podem ser realizadas somente com portas geradas anteriormente. Esta codificação utiliza três bits para representar os diferentes tipos de portas. Uma população inicial é gerada aleatoriamente de acordo com as regras de codificação.

A topologia é especificada por uma lista de tipos de componentes junto com seus nós terminais (PSPICE netlist). Os componentes não conectados são desconsiderados, o que possibilita ter um número de componentes variáveis em um circuito, embora o tamanho do cromossomo seja fixo. A Figura 5.6 apresenta uma ilustração simplificada do método de codificação para um circuito com 4 entradas. Como pode ser observado no cromossomo representado na figura abaixo os três primeiros valores de cada segmento representam o tipo de porta, os próximos seis elementos a primeira entrada desta porta, e os últimos seis a segunda entrada. É importante salientar que o buffer e o inversor desconsideram a segunda entrada, ou seja, os últimos seis valores de um segmento.

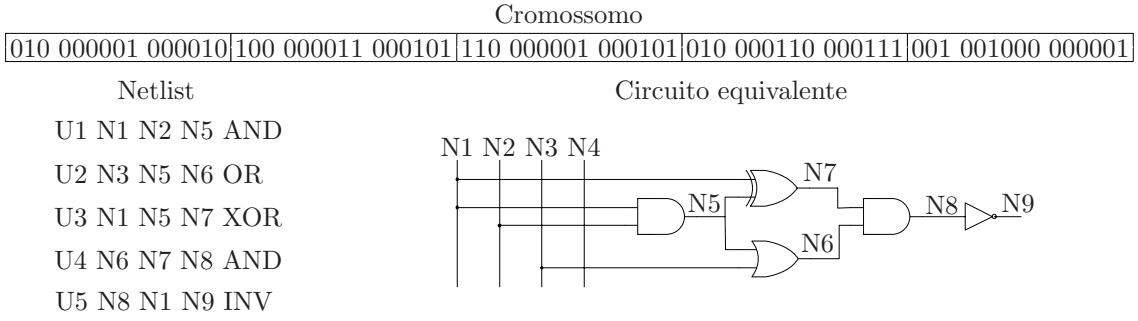


Figura 5.6: Codificação - Genótipo e Fenótipo

5.5 Processo Evolutivo

O primeiro passo na síntese evolutiva é a geração de uma população inicial aleatória de cromossomos. Na evolução extrínseca os cromossomos são convertidos em um modelo simulado (SPICE) e são avaliados de acordo com as respostas geradas, por meio da função de *fitness* apresentada na Eq. 5.9. A preparação de uma nova iteração envolve a geração de uma nova população de indivíduos a partir de indivíduos selecionados da população anterior. Realiza-se o processo de seleção onde a técnica usada é a roleta com estratégia elitista. A seguir realiza-se o processo de reprodução usando os operadores de crossover e mutação. Neste trabalho foi utilizada uma taxa de 90% de crossover e 5% de mutação. Estes parâmetros buscam diversidade nas soluções obtidas através do processo evolutivo. O crossover de um ponto recombina dois indivíduos a partir de uma posição escolhida aleatoriamente (chamado ponto de corte). A operação realiza a troca dos bits das strings a partir deste ponto. Durante a operação de mutação os pontos são escolhidos, de acordo com a probabilidade de mutação, e os valores destes bits são invertidos. A partir

deste ponto uma nova população é gerada e uma nova iteração pode ser iniciada. Este processo se repete até que se obtenha um circuito que responda adequadamente a uma determinada função ou até que um número pré-determinado de gerações seja realizado.

5.6 Sistema de Controle

Neste trabalho foi utilizado um controlador que opera em malha fechada, como mostrado na Figura 5.7. Esta figura mostra o processo contínuo obtido pelo modelo do pêndulo, o segurador de ordem zero (Z.O.H) e o bloco de saturação.

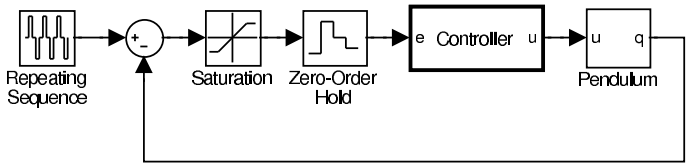


Figura 5.7: Diagrama de blocos do sistema em malha fechada

Experimentos com duas plantas diferentes foram realizados para avaliar o desempenho do circuito controlador obtido via Hardware Evolutivo. Os resultados são comparados com um controlador Proporcional e um controlador Proporcional - Derivativo, sendo os ganhos destes controladores obtidos através de Algoritmos Genéticos.

O tempo de amostragem foi estabelecido em $T_S = 1ms$ e os sinais foram discretizados por um conversor analógico-digital (ADC), codificado em $MATLAB$, para converter o sinal de erro de entrada, contido em uma faixa de -1 a 1, em uma representação digital discreta de 4 bits, considerando números com sinal em complemento de dois. A operação reversa é realizada por um conversor digital-analógico (DAC).

Os resultados apresentados foram obtidos pela co-simulação do controlador implementado, usando $Matlab/Simulink$ para o controlador e $PSPICE$ para o circuito digital.

5.7 Controladores de 4 Bits

O controlador digital de 4-bits considera a representação digital do sinal de erro, apresentada pela Figura 5.8, e o sinal de referência, $r(t)$, definido pela Eq. 5.10, onde $\mu(t)$ é a função degrau unitário.

$$r(t) = \frac{\pi}{2}\left[\mu(t) - \mu(t-8) - \mu(t-16) + \mu(t-24)\right]. \tag{5.10}$$

$$\begin{array}{rcrcl}
0.8125 & \leq e < & \infty & \rightarrow & e_D = 0111 \\
0.6875 & \leq e < & 0.8125 & \rightarrow & e_D = 0110 \\
0.5625 & \leq e < & 0.6875 & \rightarrow & e_D = 0101 \\
0.4375 & \leq e < & 0.5625 & \rightarrow & e_D = 0100 \\
0.3125 & \leq e < & 0.4375 & \rightarrow & e_D = 0011 \\
0.1875 & \leq e < & 0.3125 & \rightarrow & e_D = 0010 \\
0.0625 & \leq e < & 0.1875 & \rightarrow & e_D = 0001 \\
-0.0625 & \leq e < & 0.0625 & \rightarrow & e_D = 0000 \\
-0.1875 & \leq e < & -0.0625 & \rightarrow & e_D = 1111 \\
-0.3125 & \leq e < & -0.1875 & \rightarrow & e_D = 1110 \\
-0.4375 & \leq e < & -0.3125 & \rightarrow & e_D = 1101 \\
-0.5682 & \leq e < & -0.4375 & \rightarrow & e_D = 1100 \\
-0.6875 & \leq e < & -0.5625 & \rightarrow & e_D = 1011 \\
-0.8125 & \leq e < & -0.6875 & \rightarrow & e_D = 1010 \\
-1.0625 & \leq e < & -0.8125 & \rightarrow & e_D = 1001 \\
-\infty & < e < & -1.0625 & \rightarrow & e_D = 1000
\end{array}$$

Figura 5.8: Conversor Analógico-Digital de 4-bits

5.7.1 Controlador Proporcional e Controlador Evolutivo

O controle proporcional realizado pelo MATLAB é descrito na Eq. 5.11, e o controlador obtido via hardware evolutivo é um circuito combinacional com quatro bits de entrada e quatro bits de saída conforme Figura 5.9.

$$u_D(n) = K_P e_D(n); \tag{5.11}$$

Os controladores foram evoluídos tanto para o sistema linear quanto para o

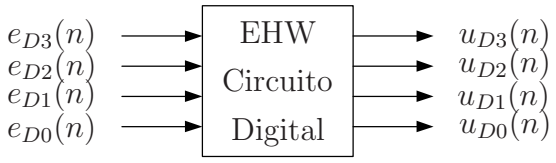


Figura 5.9: Controlador EHW de 4-bits

sistema não linear apresentados nas Equações 5.1 e 5.2 e os resultados são apresentados a seguir.

Sistema Linear

Para o sistema linear foram executados cinco experimentos, com o objetivo de criar uma base comparativa em relação ao comportamento gerado pelo circuito evoluído. No primeiro caso o comportamento do índice de erro foi estudado em relação a um sistema linear puro, sem presença de elementos não lineares na malha de realimentação, conforme mostra a Figura 5.10. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,1897$ para um ganho $Kp = 7,085$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.11 e 5.12.

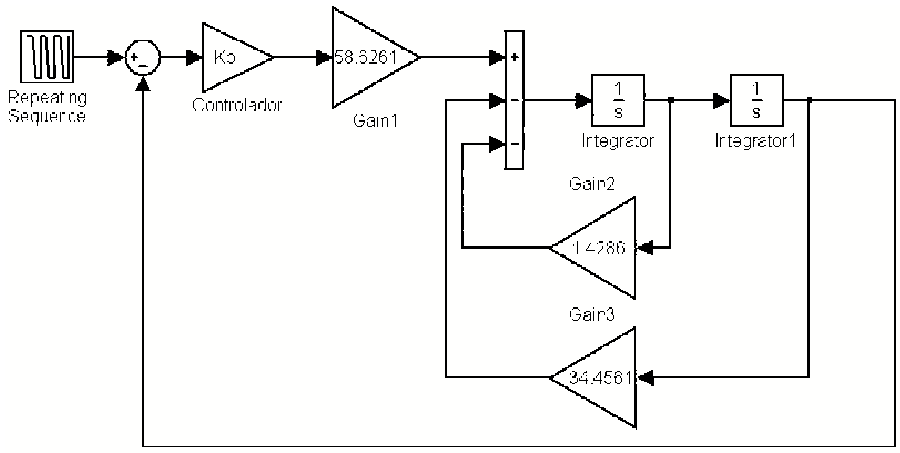


Figura 5.10: Controlador Proporcional Linear

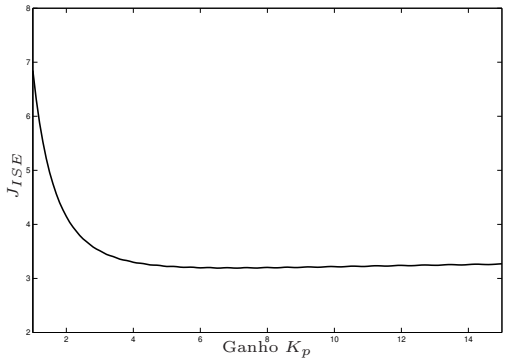


Figura 5.11: Índice de desempenho J_{ISE} em função do Ganho K_p

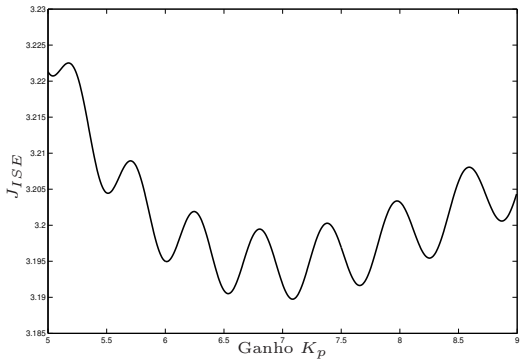


Figura 5.12: Detalhe do mínimo do Índice de desempenho J_{ISE}

Neste segundo caso o comportamento do índice de erro foi estudado em relação a um sistema linear com a presença de um elemento de saturação na malha de realimentação, conforme mostra a Figura 5.13. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,0022$ para um ganho $Kp = 7,715$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.14 e 5.15.

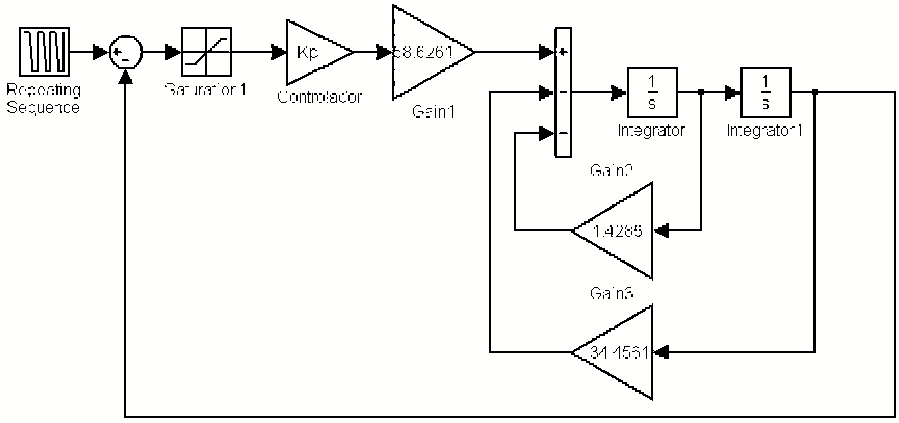


Figura 5.13: Controlador Proporcional Linear Saturado

No terceiro caso o comportamento do índice de erro foi estudado em relação a um sistema linear com a presença de um elemento de saturação e um segurador de ordem zero na malha de realimentação, para simular uma taxa de aquisição de dados de $1ms$, conforme mostra a Figura 5.16. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,3344$ para um ganho $Kp = 5,02$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.17 e 5.18.

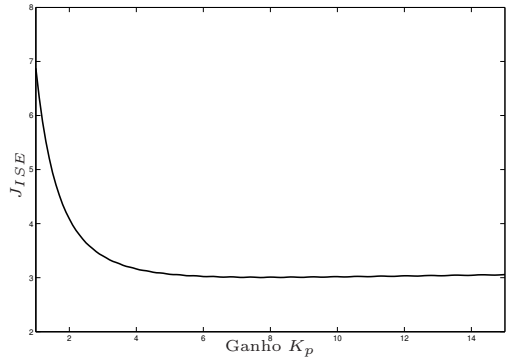


Figura 5.14: Índice de desempenho J_{ISE} em função do Ganho K_p

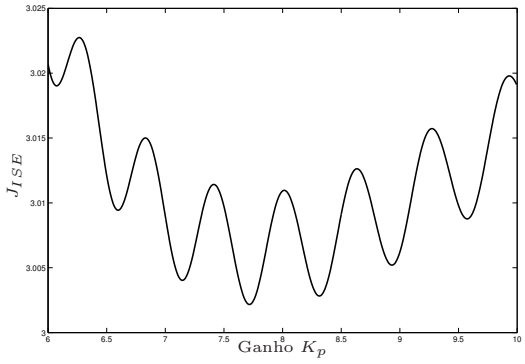


Figura 5.15: Detalhe do mínimo do Índice de desempenho J_{ISE}

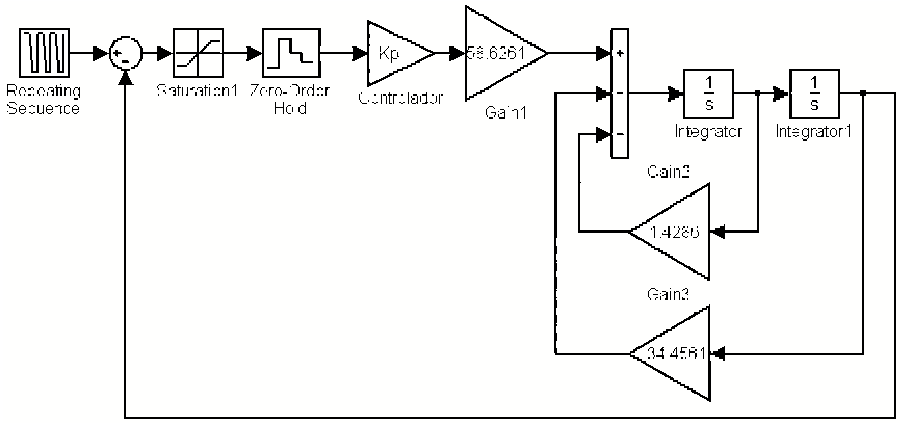


Figura 5.16: Controlador Proporcional Linear ZOH

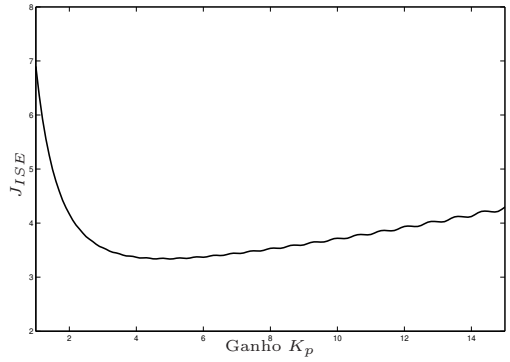


Figura 5.17: Índice de desempenho J_{ISE} em função do Ganho K_p

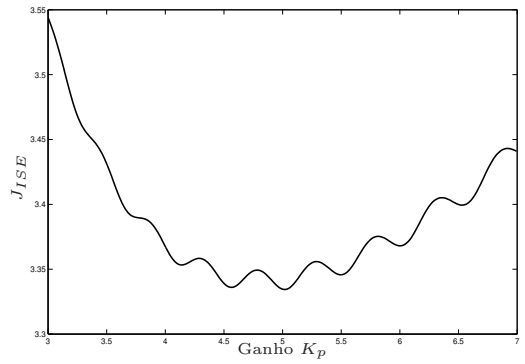


Figura 5.18: Detalhe do mínimo do Índice de desempenho J_{ISE}

O quarto caso apresenta um sistema linear com a presença de um elemento de saturação, um segurador de ordem zero e um quantizador na malha de realimentação, para simular a presença de um conversor analógico-digital, conforme mostra a Figura 5.19. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,1914$ para um ganho $Kp = 6,01$, como pode ser observado nas Figuras 5.20 e 5.21.

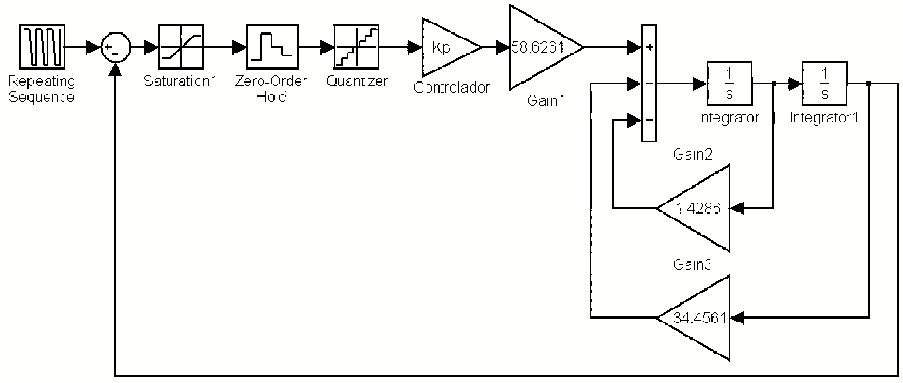


Figura 5.19: Controlador Proporcional Linear Quantizado

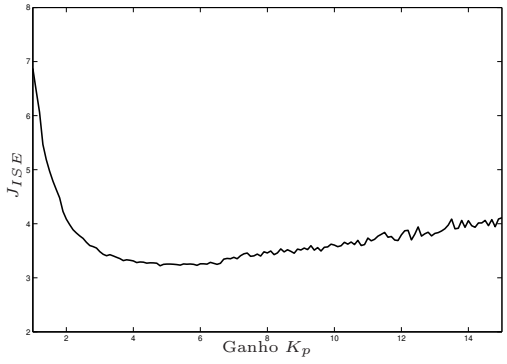


Figura 5.20: Índice de desempenho J_{ISE} em função do Ganho K_p

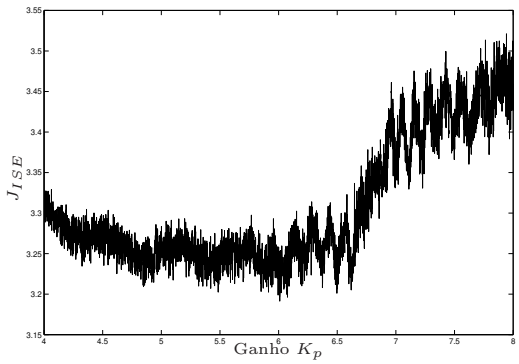


Figura 5.21: Detalhe do mínimo do Índice de desempenho J_{ISE}

O quinto sistema apresenta um sistema linear com a presença de um elemento de saturação, um segurador de ordem zero e um conjunto DAC/DAC na malha de realimentação, para simular a presença de um conversor digital-analógico de quatro bits (inteiros no intervalo $[-8, 7]$), conforme mostra a Figura 5.22. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,2289$ para um ganho $Kp = 4$, como pode ser observado na Figura 5.23. Este sistema foi usado como elemento de comparação para o sistema otimizado por Algoritmo Genético e por *Hardware* Evolutivo.

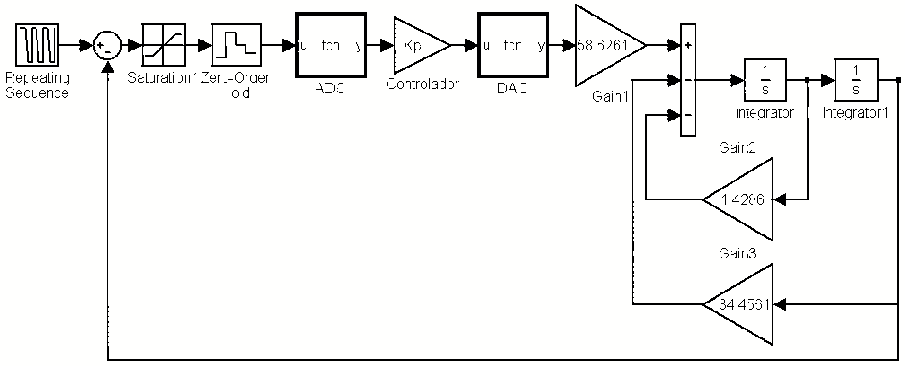


Figura 5.22: Controlador Proporcional Linear Quantizado Inteiro $4 - bits$

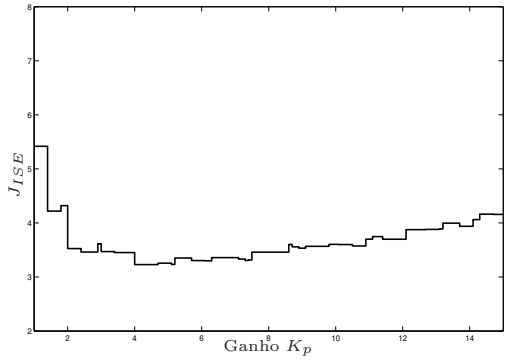


Figura 5.23: Índice de desempenho J_{ISE} em função do Ganho K_p

O controlador proporcional evoluído por GA levou a um ganho proporcional $Kp = 4.31$, com $J_{ISE} = 3,2289$, ou seja, o ponto mínimo global. O comportamento evolutivo do sistema, para 500 gerações, plotado na figura em intervalos de 10 gerações, taxa de crossover de um ponto de 90%, taxa de mutação de $7,5\%$ e população de 30 indivíduos, é apresentado na Figura 5.25. Já o controlador obtido por Hardware Evolutivo produz o circuito apresentado no *Netlist* na Figura 5.24 com um

$J_{ISE} = 2,3847$ sendo o comportamento evolutivo, para os mesmos parâmetros anteriores é apresentado na Figura 5.26 e também plotado em intervalos de 10 gerações. A resposta temporal de ambos controladores é apresentada nas Figuras 5.27 e 5.28. É importante salientar a dificuldade para otimizar este problema considerando a saturação e a quantização.

```
U1  N1  N3  N5  XOR              U19 N1  N19 N23 AND
U2  N2  N5  N6  AND              U20 N14 N20 N24 AND
U3  N1  --  N7  INV              U21 N17 N23 N25 OR
U4  N3  N7  N8  OR               U22 N13 N22 N26 AND
U5  N2  --  N9  INV              U23 N3  N21 N27 AND
U6  N8  N9  N10 AND              U24 N23 N24 N28 AND
U7  N6  N10 N11 OR               U25 N26 N27 N29 OR
U8  N4  N11 N12 AND              U26 N18 N25 N30 OR
U9  N3  --  N13 INV              U27 N1  N3  N31 AND
U10 N2  N13 N14 OR               U28 N2  N7  N32 AND
U11 N4  --  N15 INV              U29 N9  N29 N33 AND
U12 N1  N15 N16 AND              U30 N9  N32 U34 AND
U13 N7  N11 N17 AND              U31 N28 N33 N35 OR
U14 N14 N15 N18 AND              U32 N14 N31 N36 OR
U15 N2  N4  N19 AND              U33 N9  N36 N37 AND
U16 N1  N4  N20 XNOR             U34 N17 N37 N38 OR
U17 N1  N15 N21 OR               U35 N34 N38 N40 OR
U18 N4  N7  N22 AND

N4  = E3        N3  = E2        N2  = E1        N1  = E0
N12 = S3        N30 = S2        N35 = S1        N40 = S0
```

Figura 5.24: *Netlist* do Circuito Evoluído Para Pêndulo Linear

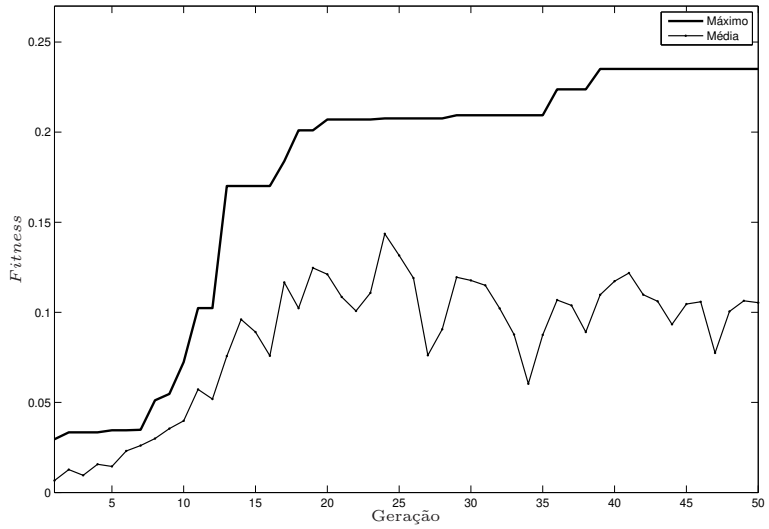


Figura 5.25: Comportamento Evolutivo do Sistema para Controlador Proporcional

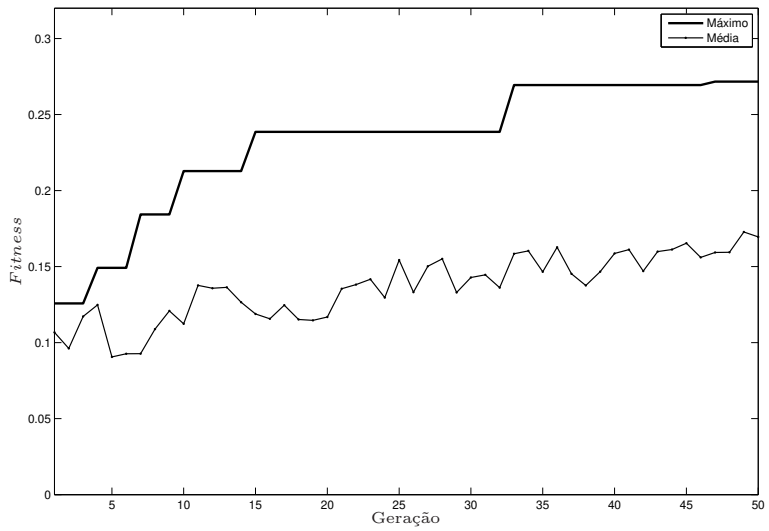


Figura 5.26: Comportamento Evolutivo do Sistema para Controlador EHW

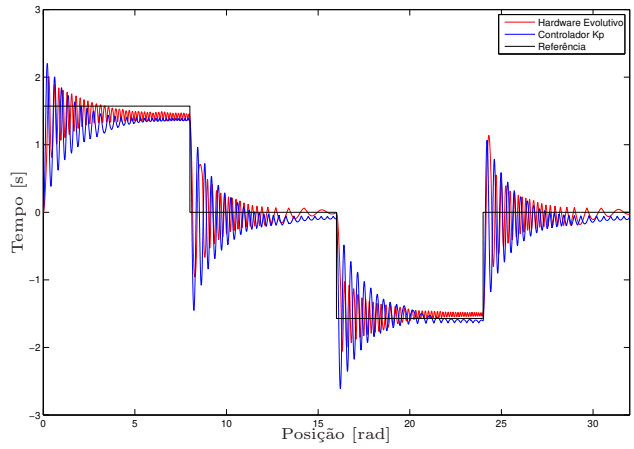


Figura 5.27: Resposta temporal do sinal de posição, $[\theta]$

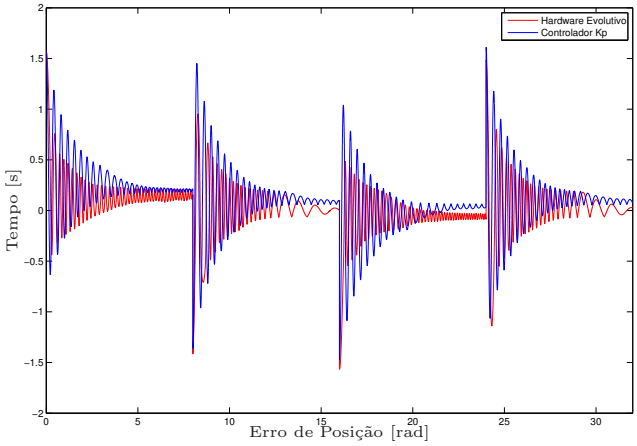


Figura 5.28: Resposta temporal do sinal de erro, $[e]$

Para facilitar a comparação entre os dois controladores e, principalmente ilustrar o motivo pelo qual o índice de desempenho do controlador EHW tem um valor menor ao mínimo global do controlador proporcional e a diferença qualitativa da resposta temporal dos sistemas, está apresentada a Tabela 5.2 e a Figura 5.29, onde pode-se obervar a característica não linear do controlador EHW. O controlador obtido via EHW consegue modelar o comportamento não linear do restante da malha.

Tabela 5.2: Tabela verdade

Entada	Saída	Saída
e_D	K_p	EHW
0000	0000	0000
0001	0001	0100
0010	0001	0011
0011	0010	0101
0100	0011	0010
0101	0011	0011
0110	0100	0001
0111	0100	0100
1000	1100	1111
1001	1100	0000
1010	1100	0000
1011	1101	1010
1100	1101	1101
1101	1110	1011
1110	1111	1110
1111	1111	0000

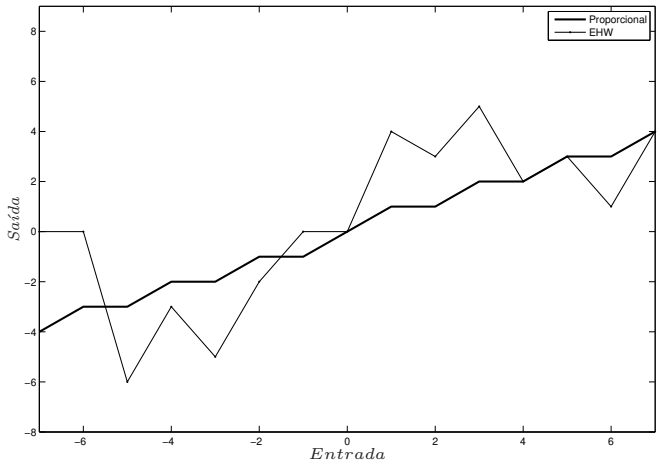


Figura 5.29: Comparação Entre as Leis de Controle Proporcional e EHW

Sistema Não Linear

Para o sistema não linear foram executados os mesmos cinco experimentos, com o objetivo de criar uma base comparativa em relação ao comportamento gerado pelo circuito evoluído. No primeiro caso o comportamento do índice de erro foi estudado em relação a um sistema não linear puro, sem presença de elementos não lineares adicionais na malha de realimentação, conforme mostram as Figuras 5.30, 5.31 e 5.32. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,2486$ para um ganho $Kp = 5,06$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.31 e 5.32.

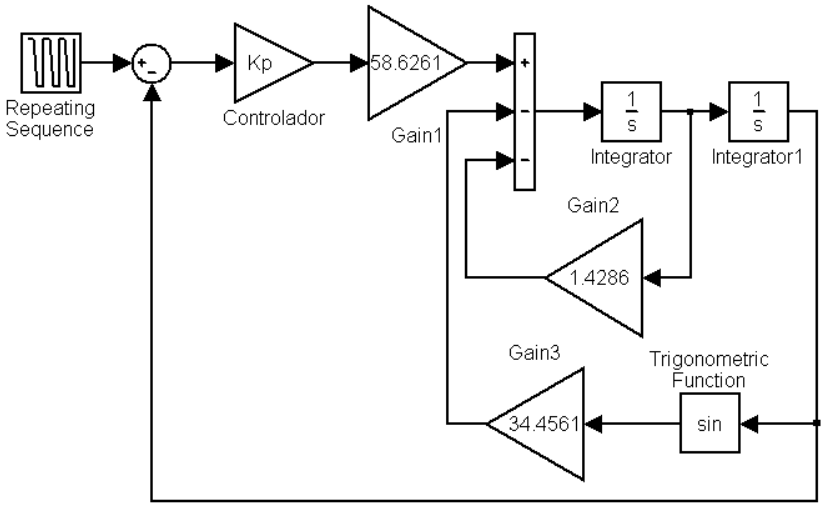


Figura 5.30: Controlador Proporcional Não Linear

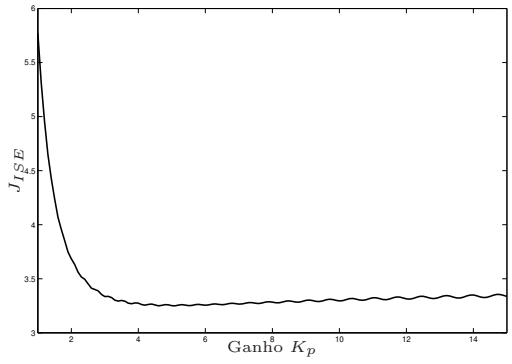


Figura 5.31: Índice de desempenho J_{ISE} em função do Ganho K_p

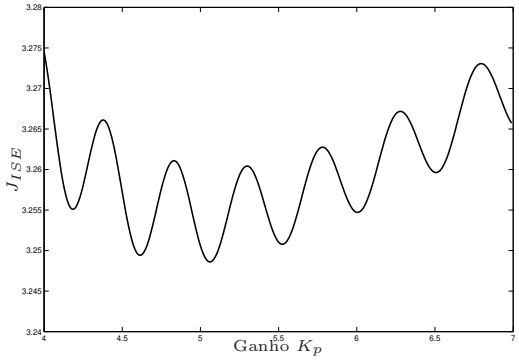


Figura 5.32: Detalhe do mínimo do Índice de desempenho J_{ISE}

Neste segundo caso o comportamento do índice de erro foi estudado em relação a um sistema não linear com a presença de um elemento de saturação na malha de realimentação, conforme mostra a Figura 5.33. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,0607$ para um ganho $Kp = 5,57$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.34 e 5.35.

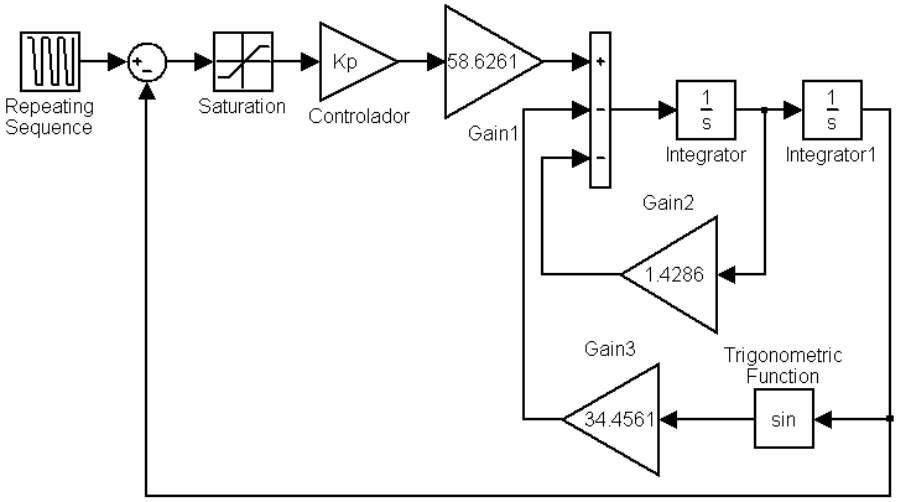


Figura 5.33: Controlador Proporcional Não Linear Saturado

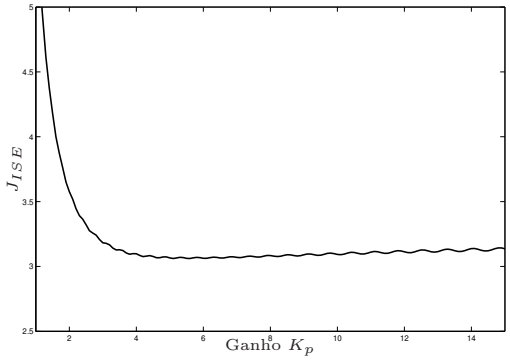


Figura 5.34: Índice de desempenho J_{ISE} em função do Ganho K_p

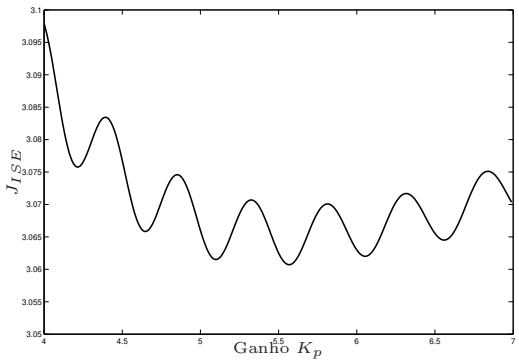


Figura 5.35: Detalhe do mínimo do Índice de desempenho J_{ISE}

No terceiro caso o comportamento do índice de erro foi estudado em relação ao sistema não linear com a presença de um elemento de saturação e um segurador de ordem zero na malha de realimentação, para simular uma taxa de aquisição de dados de $1ms$, conforme mostra a Figura 5.36. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,3007$ para um ganho $Kp = 3,77$, sendo este o ponto mínimo global, como pode ser observado nas Figuras 5.37 e 5.38.

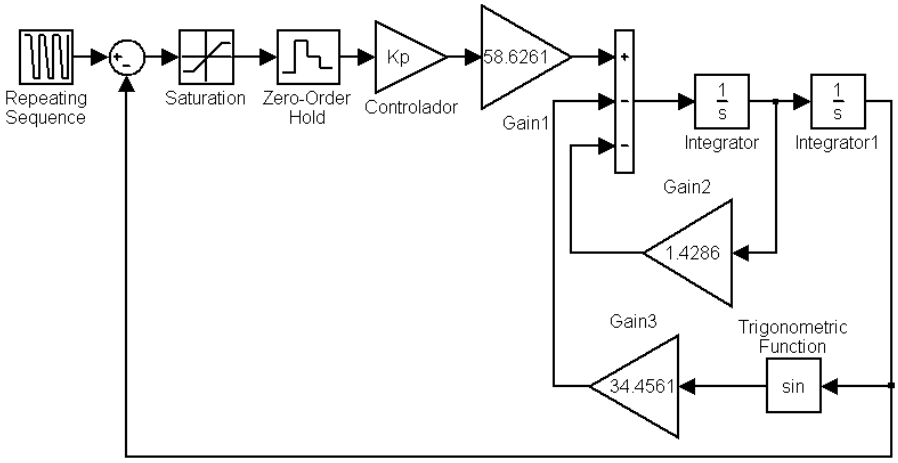


Figura 5.36: Controlador Proporcional Não Linear ZOH

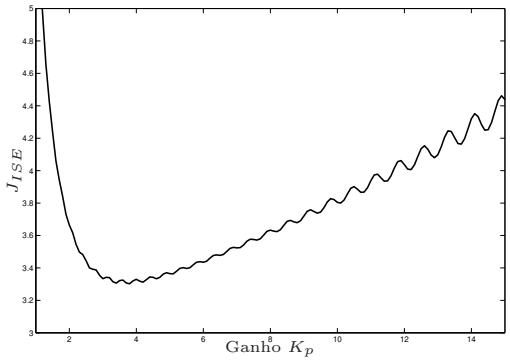


Figura 5.37: Índice de desempenho J_{ISE} em função do Ganho K_p

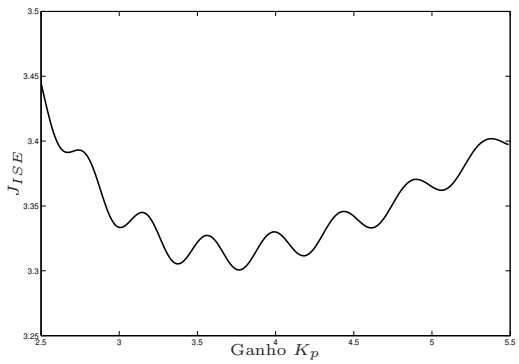


Figura 5.38: Detalhe do mínimo do Índice de desempenho J_{ISE}

O quarto caso apresenta um sistema não linear com a presença de um elemento de saturação, um segurador de ordem zero e um quantizador na malha de realimentação, para simular a presença de um conversor analógico-digital, conforme mostra a Figura 5.39. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,1748$ para um ganho $Kp = 4,48$, como pode ser observado nas Figuras 5.40 e 5.41.

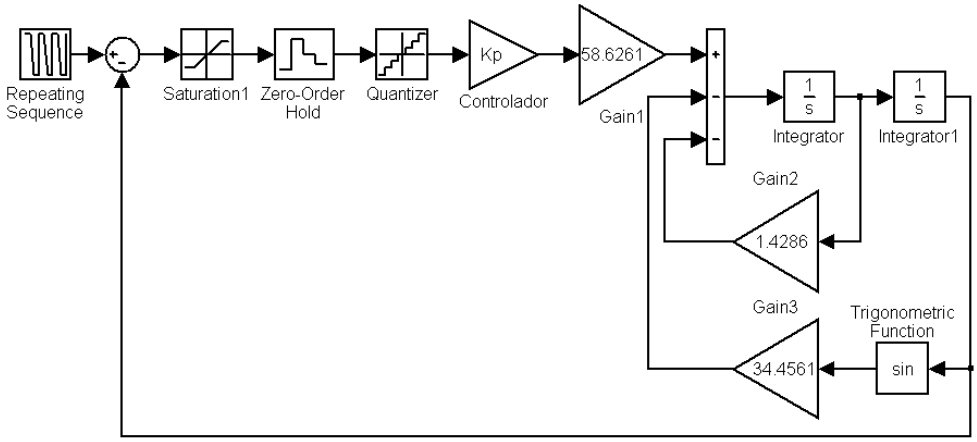


Figura 5.39: Controlador Proporcional Não Linear Quantizado

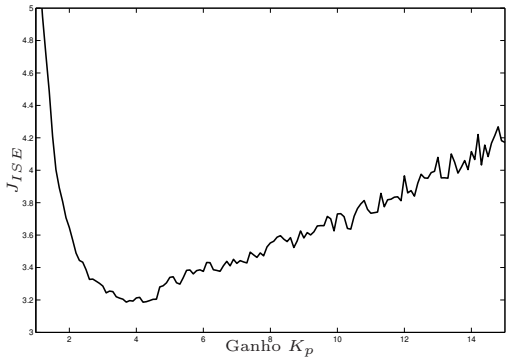


Figura 5.40: Índice de desempenho J_{ISE} em função do Ganho K_p

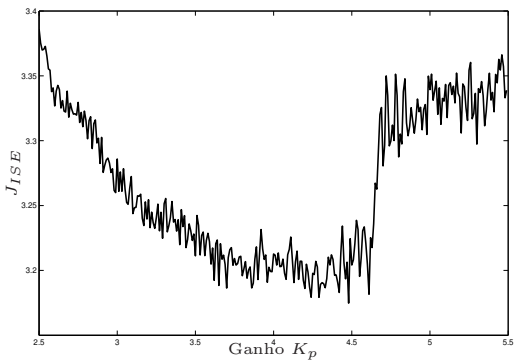


Figura 5.41: Detalhe do mínimo do Índice de desempenho J_{ISE}

O quinto sistema apresenta um sistema não linear com a presença de um elemento de saturação, um segurador de ordem zero e um conjunto DAC/DAC na malha de realimentação, para simular a presença de um conversor digital-analógico de quatro bits (inteiros no intervalo $[-8, 7]$), conforme mostra a Figura 5.42. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 3,1860$ para um ganho $Kp = 5,1$, como pode ser observado na Figura 5.43. Este sistema foi usado como elemento de comparação para os sistema otimizado por Algoritmo Genético e por *Hardware* Evolutivo.

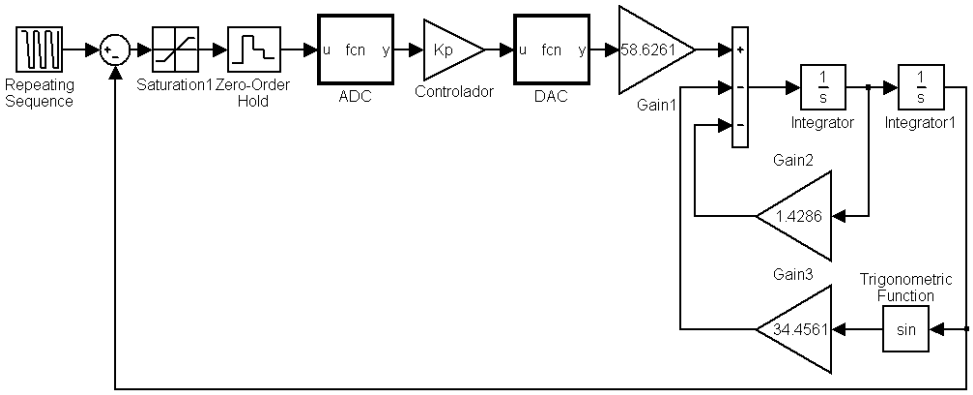


Figura 5.42: Controlador Proporcional Linear Quantizado Inteiro $4 - bits$

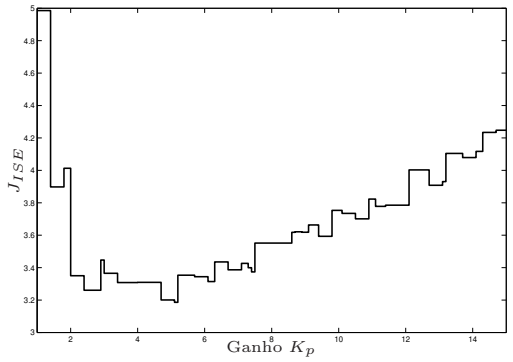


Figura 5.43: Índice de desempenho J_{ISE} em função do Ganho K_p

O controlador proporcional evoluído por algoritmos genéticos levou a um ganho proporcional $Kp = 5.35$, que corresponde a um $J_{ISE} = 3,1860$, ou seja, o ponto mínimo global. O AG foi evoluído com 500 gerações, taxa de crossover de um ponto de 90%, taxa de mutação de $7,5\%$ e população de 30 indivíduos. Já o controlador obtido por Hardware Evolutivo produz o circuito apresentado na Figura 5.44 com um $J_{ISE} = 2,3895$ para os mesmos parâmetros anteriores. A resposta temporal de ambos controladores é apresentada nas Figuras 5.57 e 5.58.

```
U1   N2  N4  N5 XOR      U26  N6 N14  N30 AND
U2   N2  --  N6 INV      U27  N7 N28  N31 OR
U3   N4  N6  N7 AND      U28 N12 N29  N32 OR
U4   N3  N4  N8 OR       U29  N4 N10  N33 AND
U5   N1  N2  N9 AND      U30  N9 N17  N34 AND
U6   N1  N2  N10 XOR     U31 N11 N31  N35 AND
U7   N1  --  N11 INV     U32 N24 N35  N36 OR
U8   N4  N9  N12 AND     U33 N30 N33  N37 OR
U9   N3  N7  N13 AND     U34 N17 N32  N38 AND
U10  N4  --  N14 INV     U35  N3 N37  N39 AND
U11  N7 N11  N15 AND     U36  N9 N34  N40 AND
U12  N3 N10  N16 AND     U37 N38 N39  N41 OR
U13  N3 --   N17 INV     U38 N16 N40  N42 OR
U14  N2 N14  N18 AND     U39  N4 N42  N43 AND
U15  N8  --  N19 INV     U40 N15 N43  N44 OR
U16  N5 N11  N20 AND
U17 N18 N19  N21 OR
U18  N2  N3  N22 AND
U19  N1 N13  N23 AND
U20 N20 N23  N24 OR
U21  N1 N21  N25 AND
U22 N17 N18  N26 AND
U23 N24 N26  N27 OR
U24  N4 N22  N28 AND
U25  N5 N11  N29 AND

N4  = E3        N3  = E2        N2  = E1        N1  = E0
N44 = S3        N36 = S2        N41 = S1        N27 = S0
```

Figura 5.44: *Netlist* do Circuito Evoluído Para Pêndulo Não Linear

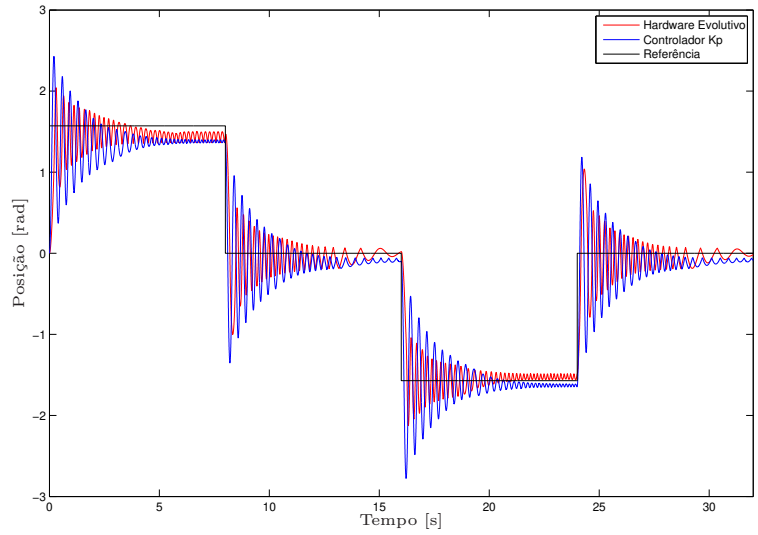


Figura 5.45: Resposta temporal do sinal de posição, $[\theta]$

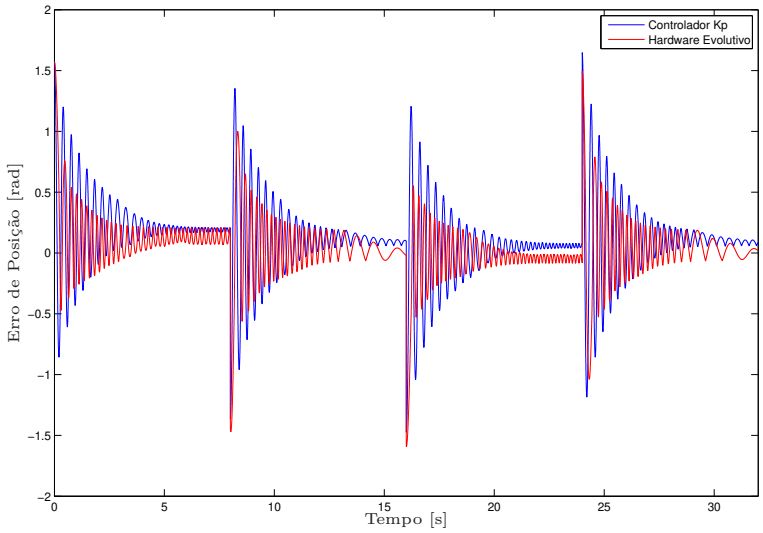


Figura 5.46: Resposta temporal do sinal de erro, $[e]$

Novamente, para facilitar a comparação entre os dois controladores e ilustrar o motivo pelo qual o índice de desempenho do controlador EHW tem um valor menor ao mínimo global do controlador proporcional e a diferença qualitativa da resposta temporal dos sistemas, apresenta-se a Tabela 5.3 e a Figura 5.47, onde fica evidente a característica não linear do controlador EHW.

Tabela 5.3: Tabela verdade

Entada	Saída	Saída
e_D	K_p	EHW
0000	0000	0000
0001	0001	0100
0010	0001	0011
0011	0010	0101
0100	0011	0010
0101	0011	0010
0110	0100	0001
0111	0100	0100
1000	1100	1111
1001	1100	0000
1010	1100	0000
1011	1101	1010
1100	1101	1101
1101	1110	1011
1110	1111	1110
1111	1111	0000

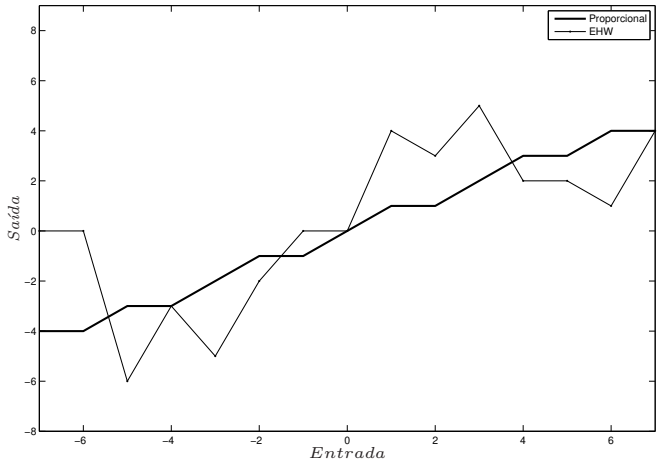


Figura 5.47: Comparação Entre as Leis de Controle Proporcional e EHW

5.7.2 Controlador Proporcional-Derivativo e Controlador Evolutivo

O controle proporcional-derivativo realizado pelo MATLAB é descrito na Eq. 5.4. O controlador obtido via hardware evolutivo é um circuito combinacional com oito bits de entrada e quatro bits de saída conforme Figura 5.48.

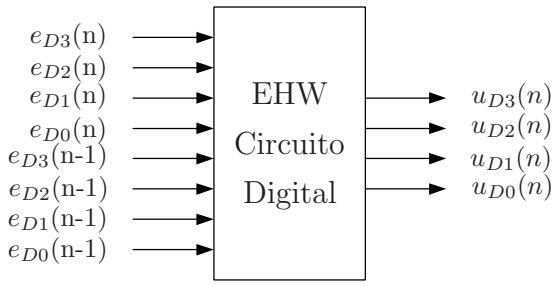


Figura 5.48: Controlador EHW 2×4-bits

Os controladores PD e Evolutivo foram evoluídos tanto para o sistema linear quanto para o sistema não linear apresentados nas Equações 5.1 e 5.2 e os resultados são apresentados a seguir.

Sistema Linear

O sistema linear foi simulado com a presença de um elemento de saturação, um segurador de ordem zero, um conjunto DAC/DAC na malha de realimentação, para simular a presença de um conversor digital-analógico de quatro bits (inteiros no intervalo $[-8, 7]$) e um elemento de memória com atraso unitário, conforme mostra a Figura 5.49. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 1,3834$ para um controlador PD com ganhos $Kp = 17,35$ e $Kd = 0.195$. A Figura 5.51 apresenta o comportamento do índice J_{ISE} em função dos ganhos do controlador. As Figuras 5.52, 5.53,5.54 apresentam diferentes vistas do índice J_{ISE} em função dos ganhos do controlador, para facilitar a visualização de um dos pontos de mínimo do sistema. A Figura 5.55 e a Tabela 5.4 apresentam a superfície de controle do controlador. É possível observar que a figura é a representação gráfica da tabela e modela o comportamento linear do sistema. Este sistema foi usado como elemento de comparação para os sistema otimizado por Algoritmo Genético e por *Hardware* Evolutivo. A evolução dos ganhos PD com um AG de 500 gerações, taxa de crossover de um ponto de 90%, taxa de mutação de $7,5\%$ e população de 30 indivíduos,

apresentou ganhos $Kp = 26$ e $Kd = 0.2$, para um $J_{ISE} = 1,3834$. Nota-se, no entanto, que devido a característica inteira de 4 bits do controlador estes valores correspondem a mesma superfície de controle apresentada na Figura 5.55.

Já o controlador obtido por Hardware Evolutivo, apresentado na Figura 5.50, produz um circuito com $J_{ISE} = 1,2498$ para os mesmos parâmetros anteriores. A resposta temporal de ambos controladores é apresentada nas Figuras 5.57 e 5.58. A Figura 5.56 e a Tabela 5.5 apresentam a superfície de controle do controlador gerado pelo hardware evolutivo onde pode-se observar o alto grau de não linearidade do controlador.

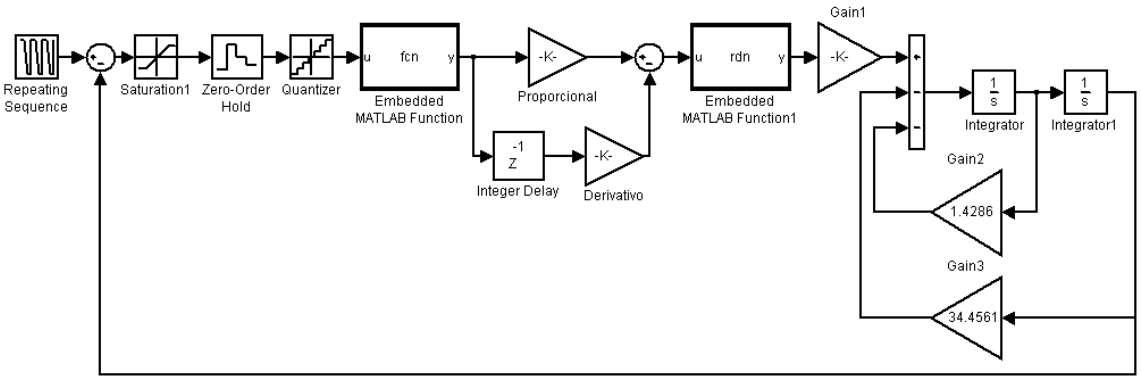


Figura 5.49: Controlador Proporcional-Derivativo Inteiro de $4 - bits$

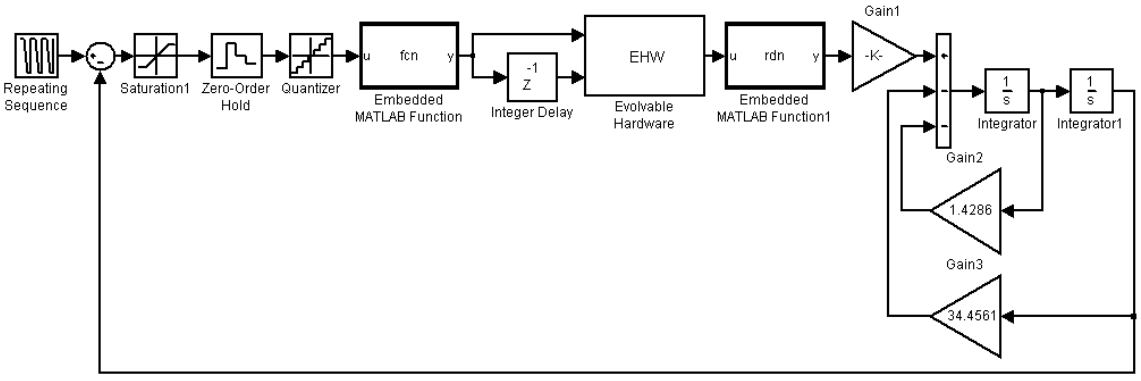


Figura 5.50: Controlador EHW $2 \times 4 - bits$

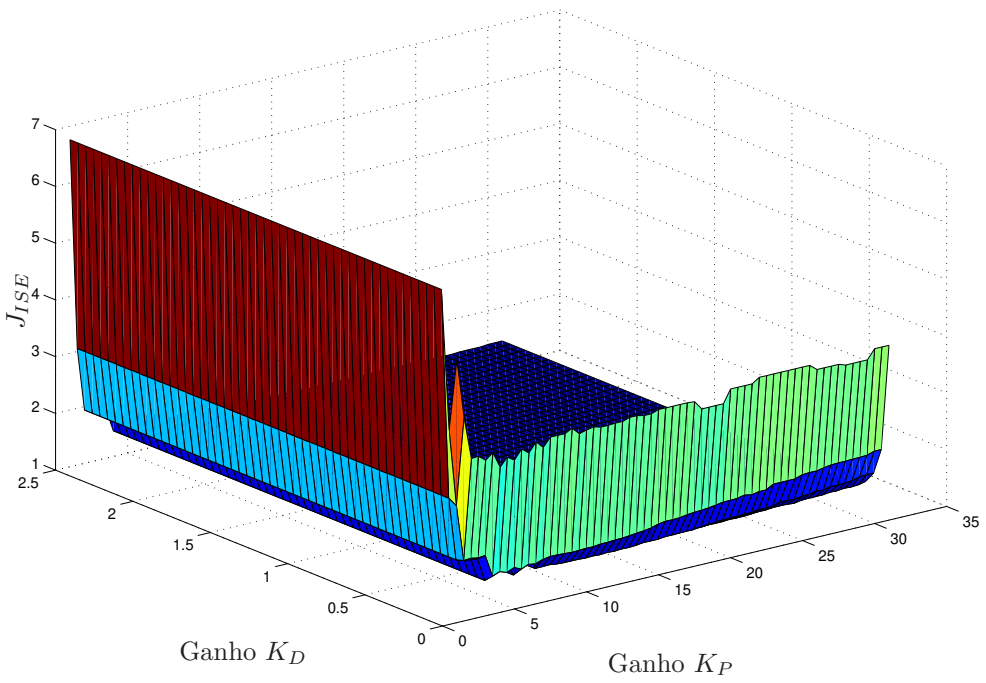


Figura 5.51: Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D

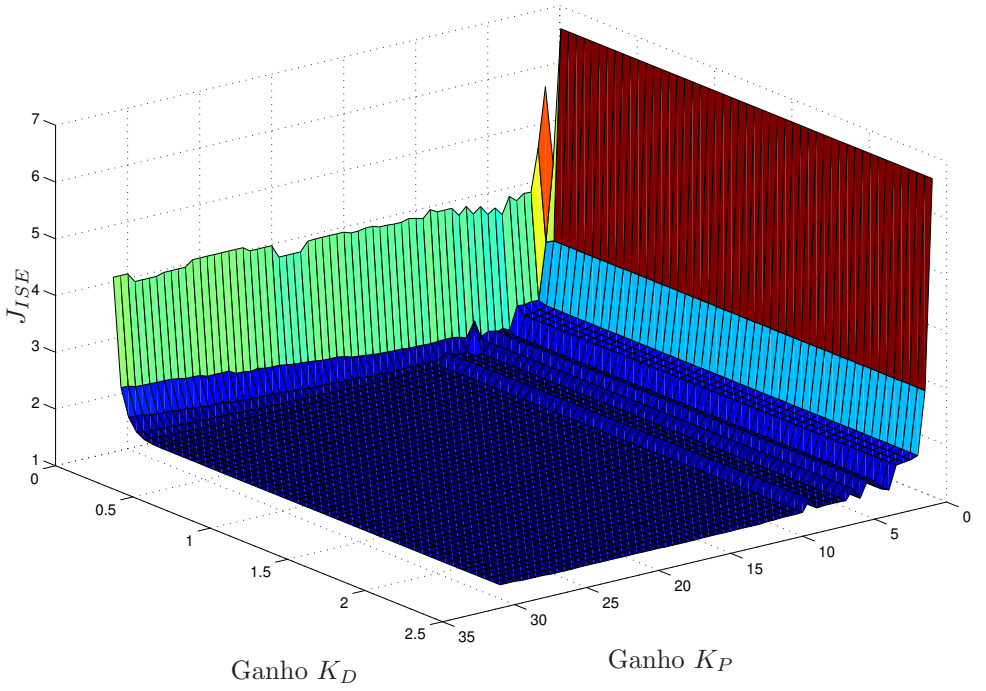


Figura 5.52: Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D - Rotação de Imagem

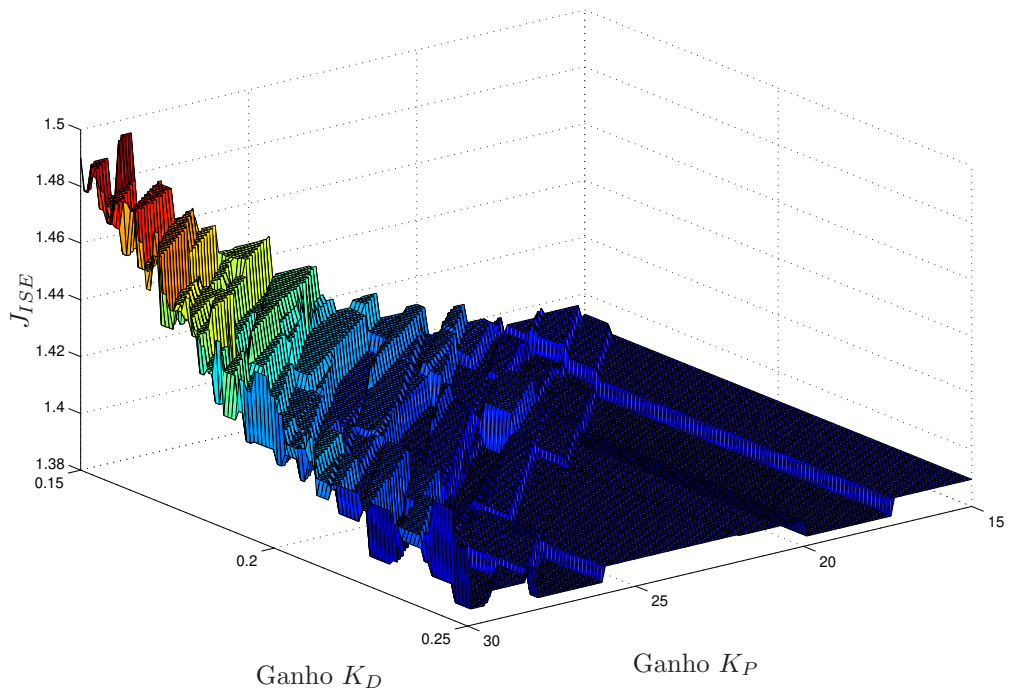


Figura 5.53: Detalhe do Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D

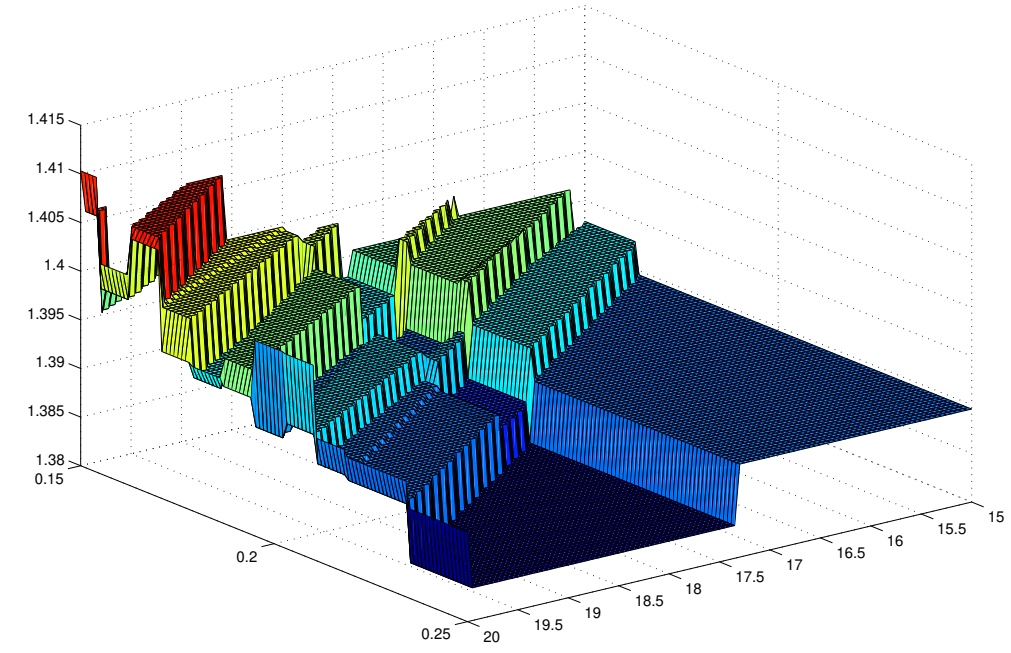


Figura 5.54: Detalhe do Ponto Mínimo do Índice de desempenho J_{ISE}

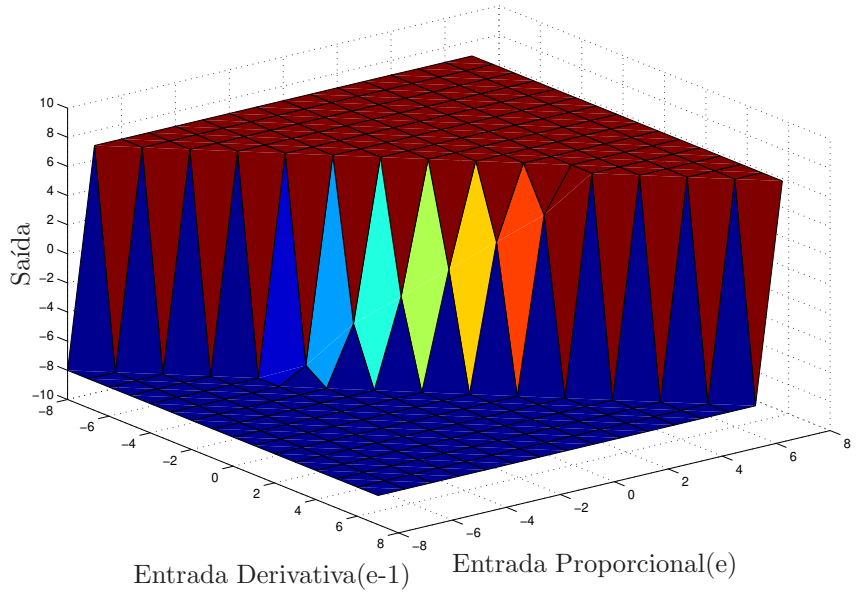


Figura 5.55: Superfície de Controle do Controlador Proporcional Derivativo

A superfície de controle do controlador PD está descrita em forma de tabela verdade na Tabela 5.4, onde as entradas estão codificadas de acordo com o sistema apresentado na Figura 5.8, e a saída é o sinal de controle na forma de um inteiro de 4 bits.

Tabela 5.4: Tabela Verdade Controlador Proporcional Derivativo

$e \backslash e-1$	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
6	-8	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7	7	7
4	-8	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7	7
3	-8	-8	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7
2	-8	-8	-8	-8	-8	4	7	7	7	7	7	7	7	7	7	7
1	-8	-8	-8	-8	-8	-8	2	7	7	7	7	7	7	7	7	7
0	-8	-8	-8	-8	-8	-8	-8	0	7	7	7	7	7	7	7	7
-1	-8	-8	-8	-8	-8	-8	-8	-8	-2	7	7	7	7	7	7	7
-2	-8	-8	-8	-8	-8	-8	-8	-8	-8	-4	7	7	7	7	7	7
-3	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-7	7	7	7	7	7
-4	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7	7	7
-5	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7	7
-6	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7
-7	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7
-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8

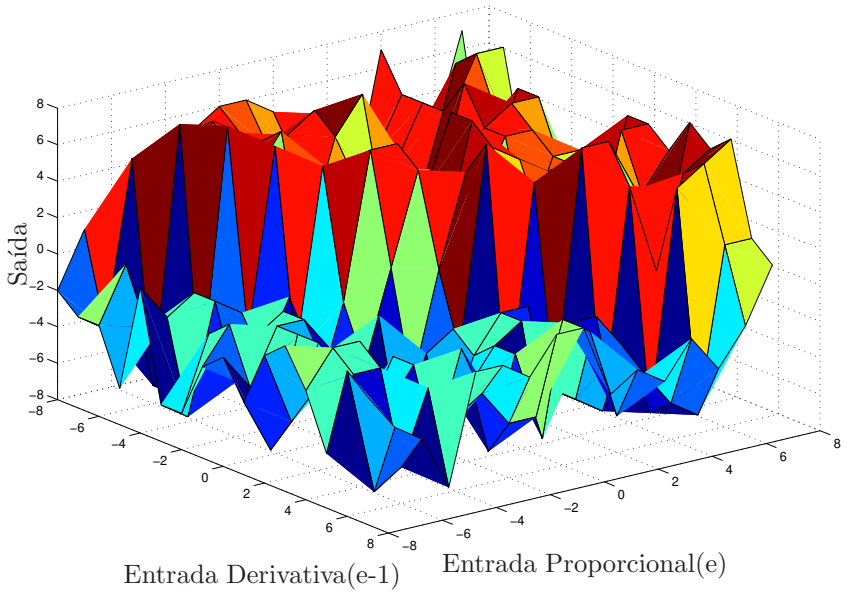


Figura 5.56: Superfície de Controle do Controlador EHW

Tabela 5.5: Tabela Verdade Controlador EHW

$e \backslash e-1$	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
7	1	2	7	6	7	5	6	6	1	0	0	2	1	7	0	7
6	-2	2	6	5	0	3	6	1	4	3	6	6	4	7	6	2
5	-4	-7	5	0	4	1	0	4	4	5	6	2	3	4	1	1
4	-2	-6	-7	5	7	0	4	4	0	5	5	7	5	5	5	7
3	-4	-4	-3	-7	7	5	2	5	0	2	3	1	0	2	2	0
2	-3	-6	-1	-6	-6	5	4	6	7	7	3	3	3	5	4	1
1	-4	-1	-1	-5	-1	-7	7	5	1	3	0	1	7	3	5	2
0	0	0	-7	-1	-1	-7	-4	0	5	6	0	3	5	6	3	5
-1	0	-5	-2	-2	-3	-2	-5	-6	0	6	5	4	0	5	4	6
-2	-2	-5	-7	-4	-7	-3	-6	-7	-5	-2	5	0	6	5	5	6
-3	-1	-3	-1	-3	-4	-7	-1	-4	-7	-7	-5	6	4	4	6	5
-4	-1	-4	-6	-2	-2	-6	-6	-3	-6	-1	-5	-4	7	3	3	3
-5	-7	-2	-4	-4	-5	-6	-3	-1	-1	-4	-2	-3	-7	7	2	4
-6	-4	-6	-1	0	-2	-1	-1	-1	-2	-5	-1	-1	-7	-7	5	0
-7	-3	-7	-1	0	-3	-5	-7	-4	-1	-2	-7	-2	-3	0	-4	1
-8	-1	0	-5	-2	-0	-1	-4	-2	-3	-6	-6	0	-6	-3	-4	-2

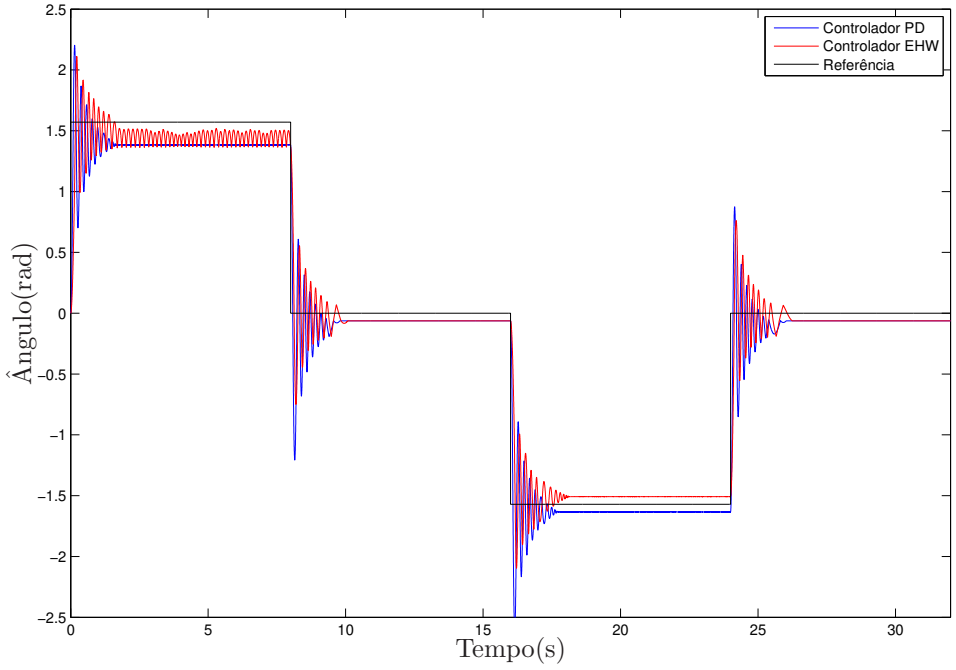


Figura 5.57: Resposta temporal do sinal de posição, [θ]

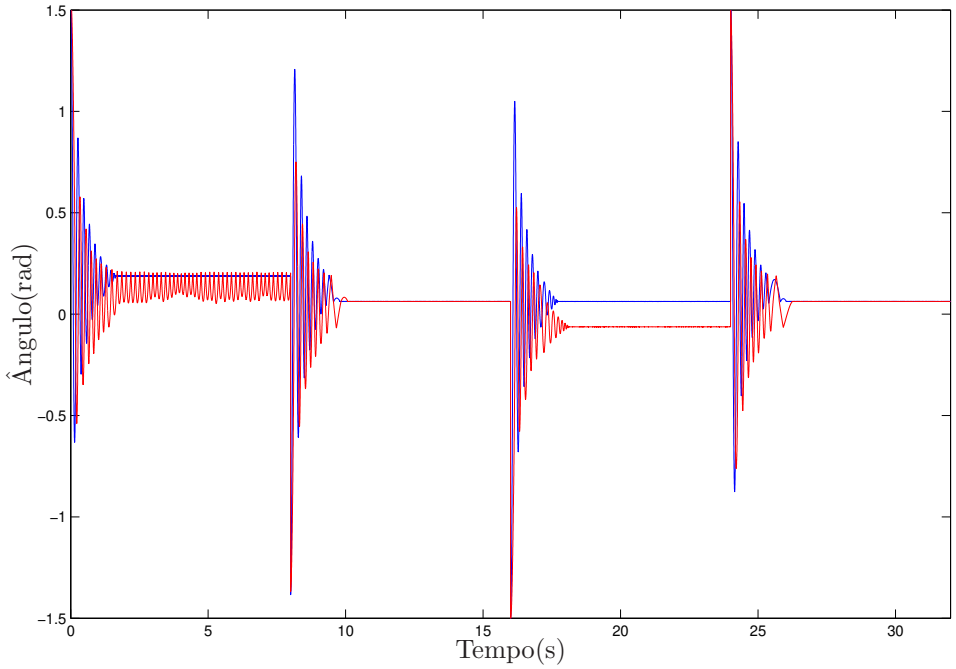


Figura 5.58: Resposta temporal do sinal de erro, [e]

Sistema Não Linear

O sistema não linear foi simulado com a presença de um elemento de saturação, um segurador de ordem zero, um conjunto DAC/DAC na malha de realimentação, para simular a presença de um conversor digital-analógico de quatro bits (inteiros no intervalo $[-8, 7]$) e um elemento de memória com atraso unitário, conforme mostra a Figura 5.59. Este sistema apresenta índice de desempenho mínimo de $J_{ISE} = 1,4186$ para um controlador PD com ganhos $Kp = 26$ e $Kd = 0.21$. A Figura 5.61 apresenta o comportamento do índice J_{ISE} em função do ganhos do controlador. As Figuras 5.62, 5.63 e 5.64 apresentam diferentes vistas do índice J_{ISE} em função do ganhos do controlador, para facilitar a visualização de um dos pontos de mínimo do sistema. A Figura 5.65 e a Tabela 5.6 apresentam a superfície de controle do controlador.

Este sistema foi usado como elemento de comparação para os sistema otimizado por Algoritmo Genético e por *Hardware* Evolutivo. A evolução dos ganhos PD com um AG de 500 gerações, taxa de crossover de um ponto de 90%, taxa de mutação de $7,5\%$ e população de 30 indivíduos, apresentou ganhos $Kp = 27,25$ e $Kd = 0.22$, para um $J_{ISE} = 1,4186$. Nota-se, novamente, que devido a característica inteira de 4 bits do controlador estes valores correspondem a mesma superfície de controle apresentada na Figura 5.65.

Já o controlador obtido por Hardware Evolutivo, apresentado na Figura 5.60, produz um circuito com $J_{ISE} = 1,2735$ para os mesmos parâmetros anteriores. A resposta temporal de ambos controladores é apresentada nas Figuras 5.67 e 5.68. A Figura 5.66 e a Tabela 5.7 apresentam a superfície de controle do controlador gerado pelo hardware evolutivo onde pode-se observar o alto grau de não linearidade do controlador.

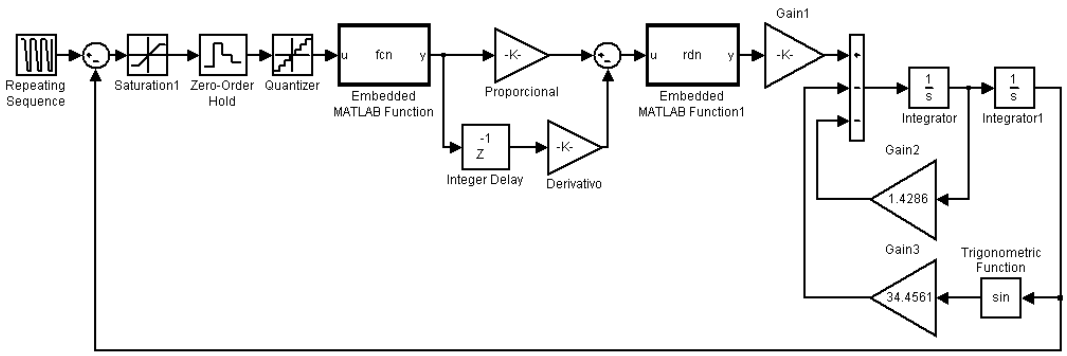


Figura 5.59: Controlador Proporcional-Derivativo Inteiro de 4-bits

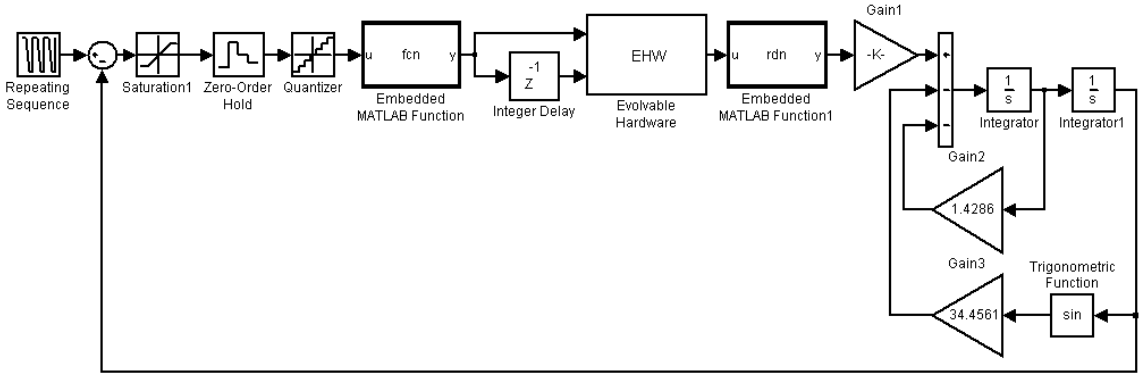


Figura 5.60: Controlador EHW 2×4-bits

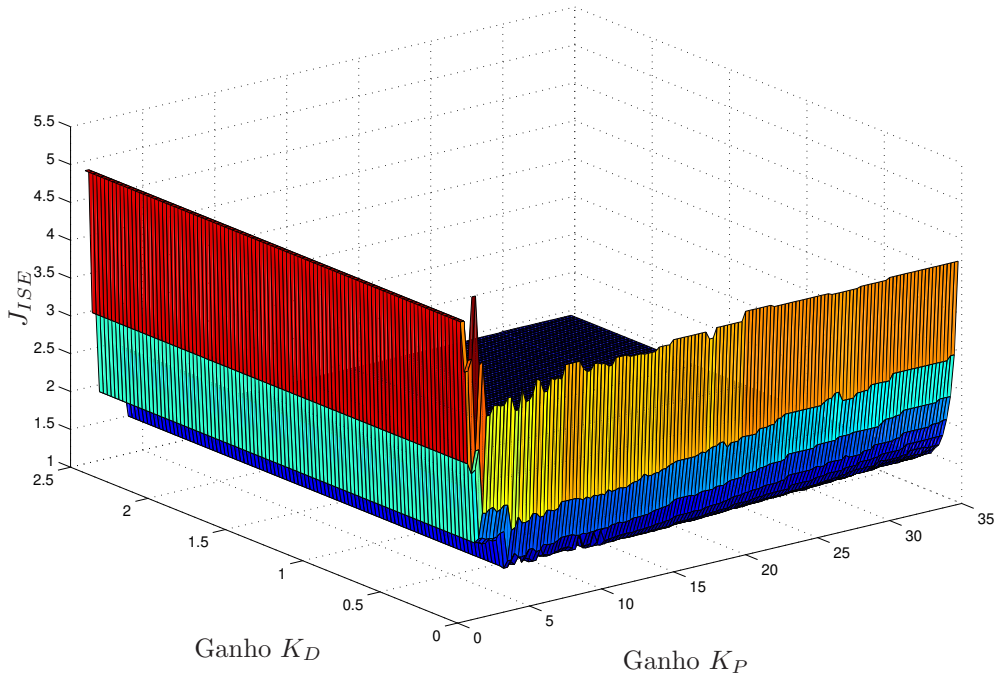


Figura 5.61: Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D

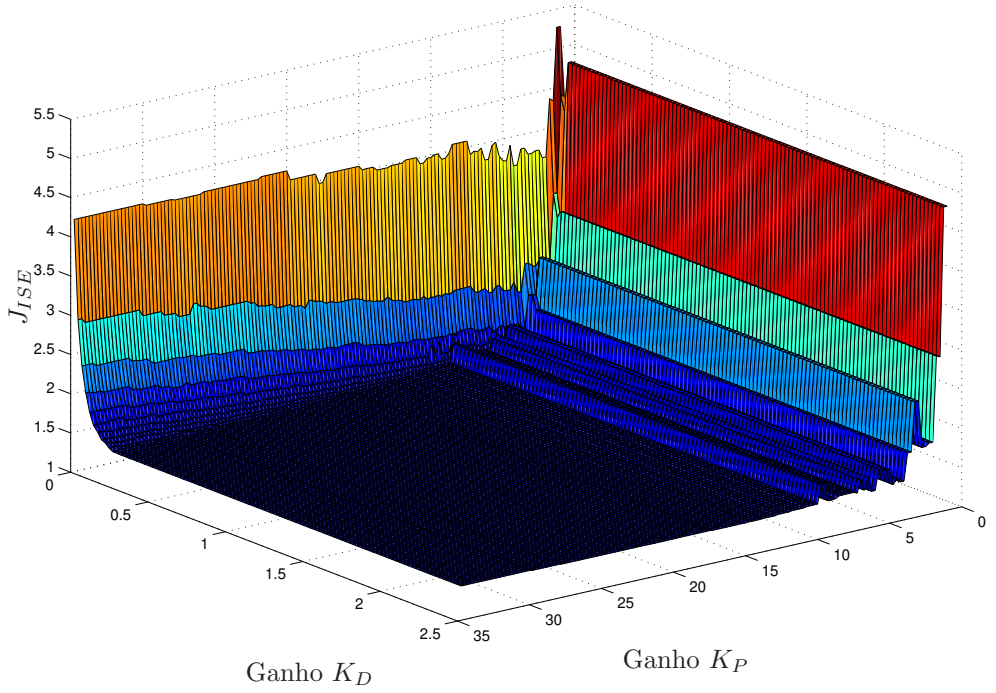


Figura 5.62: Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D - Rotação de Imagem

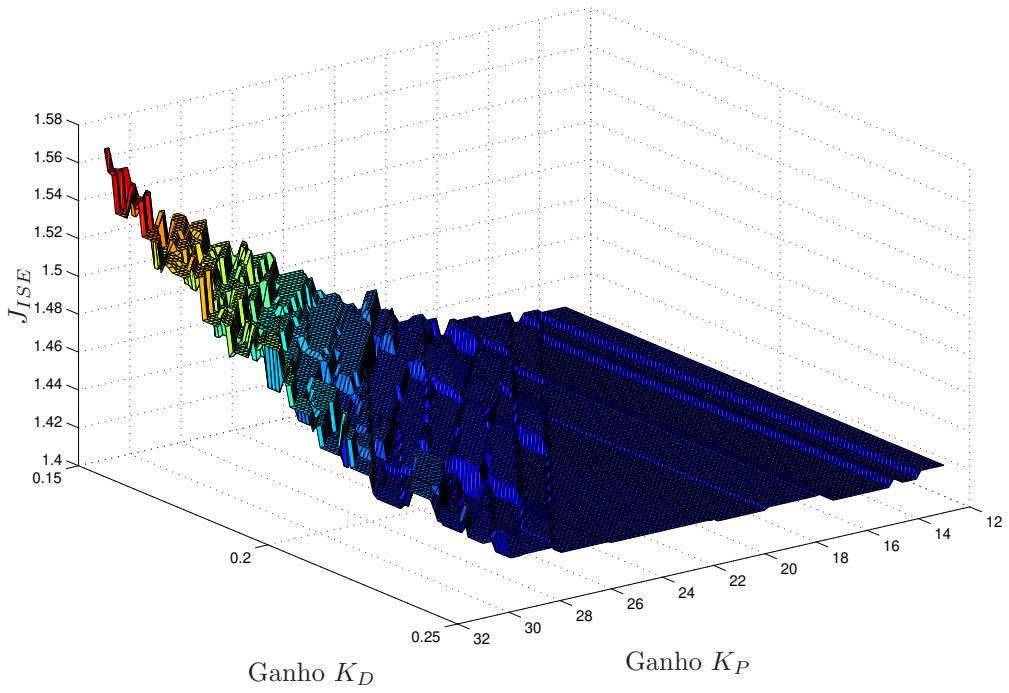


Figura 5.63: Detalhe do Índice de desempenho J_{ISE} em função do Ganhos K_P e K_D

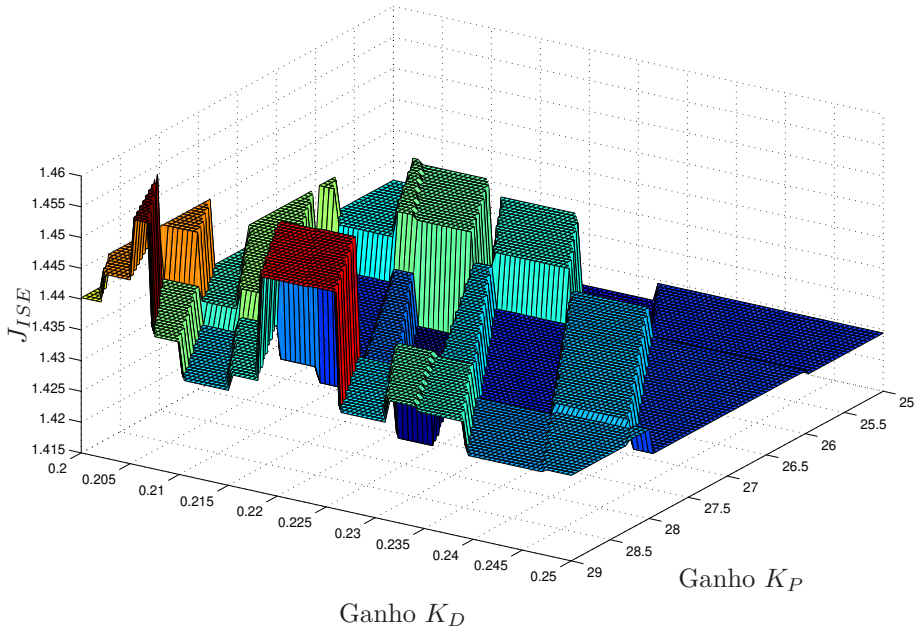


Figura 5.64: Detalhe do Ponto Mínimo do Índice de desempenho J_{ISE}

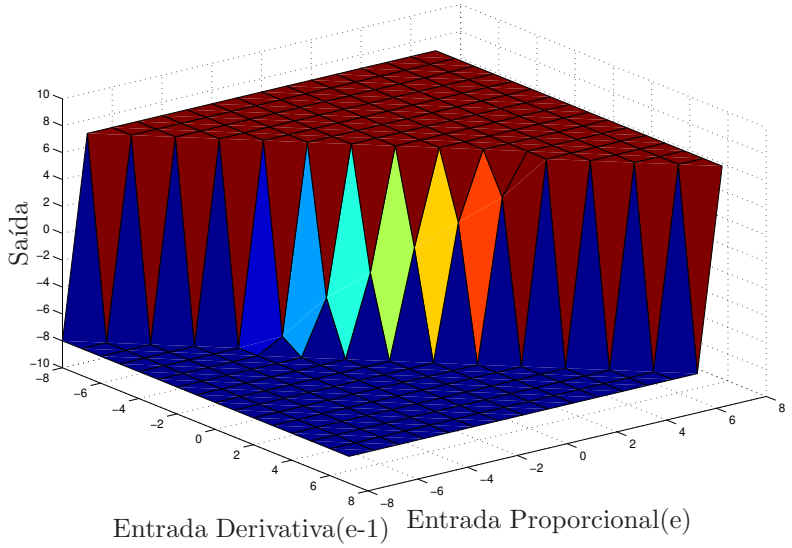


Figura 5.65: Superfície de Controle do Controlador Proporcional Derivativo

Tabela 5.6: Tabela Verdade Controlador Proporcional

$e \backslash e-1$	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
6	-8	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7	7	7
4	-8	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7	7
3	-8	-8	-8	-8	7	7	7	7	7	7	7	7	7	7	7	7
2	-8	-8	-8	-8	-8	4	7	7	7	7	7	7	7	7	7	7
1	-8	-8	-8	-8	-8	-8	2	7	7	7	7	7	7	7	7	7
0	-8	-8	-8	-8	-8	-8	-8	0	7	7	7	7	7	7	7	7
-1	-8	-8	-8	-8	-8	-8	-8	-8	-2	7	7	7	7	7	7	7
-2	-8	-8	-8	-8	-8	-8	-8	-8	-8	-4	7	7	7	7	7	7
-3	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-7	7	7	7	7	7
-4	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7	7	7
-5	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7	7
-6	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7	7
-7	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	7
-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8

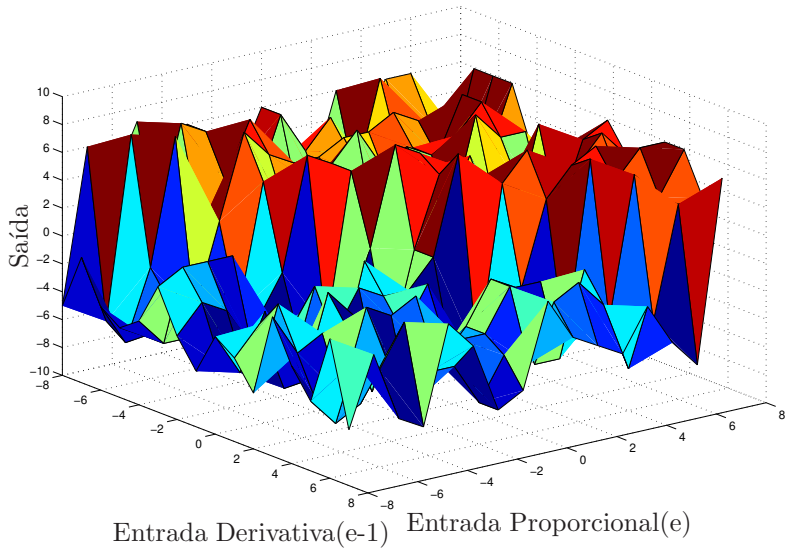


Figura 5.66: Superfície de Controle do Controlador EHW

Este capítulo apresentou os dados obtidos durante a simulação do controlador P e PD sintonizáveis por GA e os resultados obtidos pelo controlador obtido por Hardware Evolutivo. Pode-se observar que o hardware evolutivo tem a capacidade de explorar um espaço de solução mais amplo, abrangendo o espaço dos controladores não lineares, enquanto os controladores tradicionais estão restritos ao espaço dos controladores lineares.

Tabela 5.7: Tabela Verdade Controlador EHW

$e\backslash e-1$	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8
7	6	4	7	7	4	5	7	2	2	1	3	7	7	7	2	1
6	-7	4	4	7	4	7	6	5	0	5	4	2	1	0	3	6
5	-5	-4	6	5	7	5	1	7	2	5	7	7	0	4	2	6
4	-2	-7	-4	7	6	3	5	0	1	7	5	3	5	2	7	4
3	-5	-2	0	-6	4	7	0	2	5	0	4	4	6	1	0	6
2	-4	0	-6	-1	-2	5	5	5	1	4	5	2	5	3	5	0
1	-3	-1	-6	-1	-5	-7	7	5	6	0	4	3	5	0	3	2
0	0	-5	0	0	-6	-7	0	0	7	0	7	3	4	0	6	6
-1	-6	-4	-3	-3	-5	-1	-2	-7	0	5	5	3	4	0	1	2
-2	-7	-7	-3	-5	-6	-4	-2	0	-5	-6	6	1	5	7	3	2
-3	-2	-4	-4	0	-3	-7	-1	-7	-3	-7	-2	4	7	3	6	6
-4	0	-4	-7	-1	-7	-2	-4	-1	-4	-5	-5	-5	1	0	7	3
-5	-7	0	-3	0	-2	0	-6	-7	-7	-6	0	-7	-5	7	0	7
-6	-6	-3	-6	-6	-2	-6	-7	-2	-7	-3	0	-6	-4	-2	7	3
-7	-1	-2	-7	-6	-3	0	-2	-6	-6	-2	0	-6	-6	-7	-6	6
-8	-6	0	-5	-3	-0	-6	-4	-3	-6	-4	-1	-4	-5	-4	-1	-5

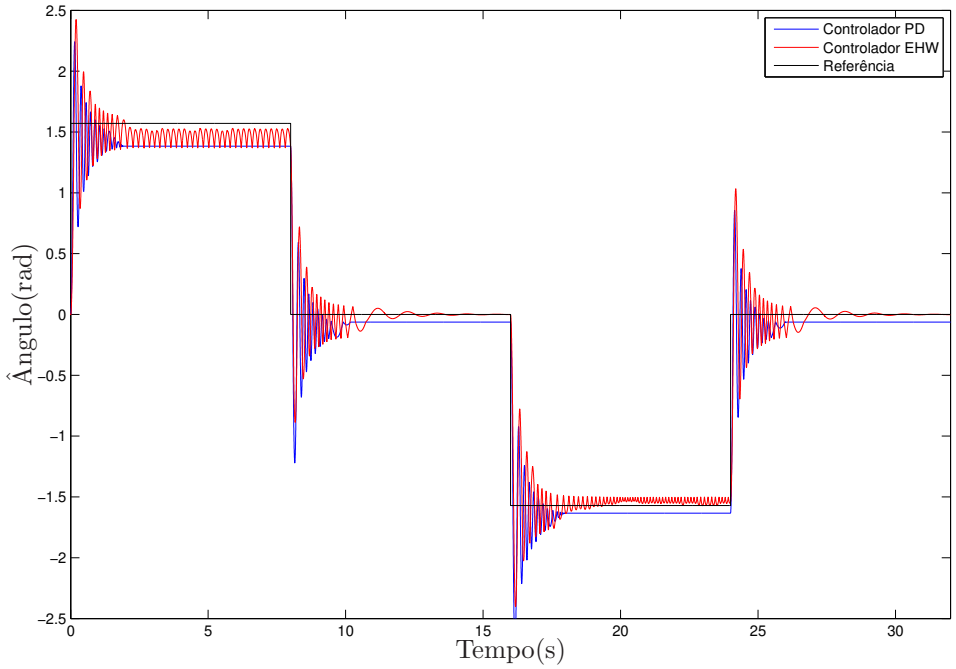


Figura 5.67: Resposta temporal do sinal de posição, $[\theta]$

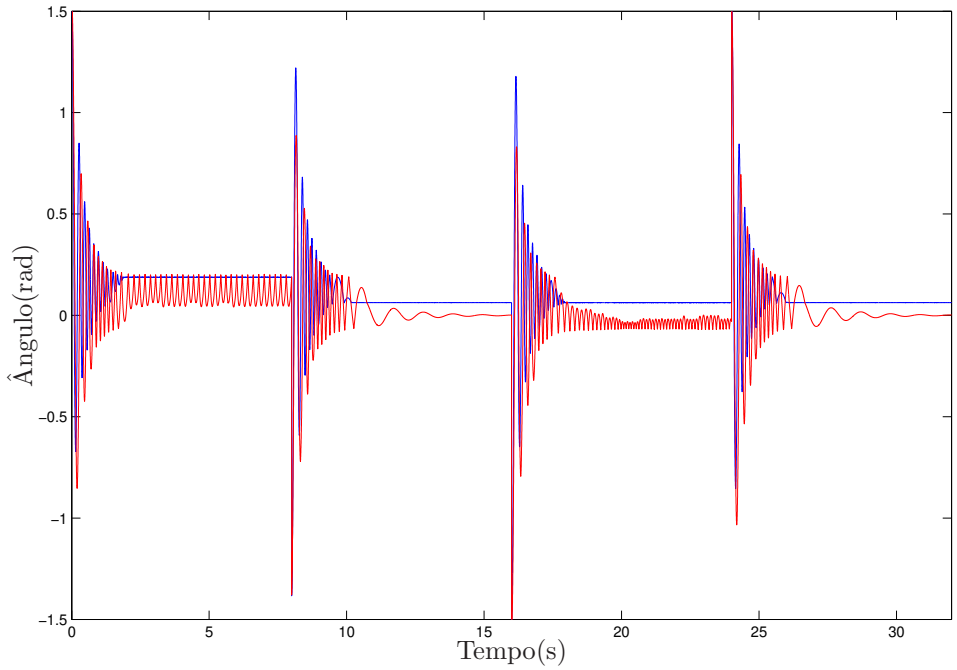


Figura 5.68: Resposta temporal do sinal de erro, $[e]$

Capítulo 6

Conclusão

6.1 Introdução

A necessidade de desenvolvimento de novas metodologias de projeto de circuitos eletrônicos proporcionou a aplicação de técnicas bio-inspiradas ao projeto de hardware, e o interesse em usar sistemas bio-inspirados levou à aplicação de técnicas evolutivas ao projeto de circuitos eletrônicos digitais, de modo que estes possam ser projetados automaticamente, criando uma técnica conhecida como *Hardware Evolutivo*.

Esta tese avalia a aplicabilidade e o desempenho da computação evolutiva no desenvolvimento de circuitos para problemas de sistemas de controle, além de realizar um estudo sobre os parâmetros de evolução para o projeto de um circuito combinacional simples, o detector de números primos de 4 e 6 bits.

6.2 Detector de Números Primos

Neste trabalho o circuito detector de números primos foi evoluído extrinsicamente com o intuito de investigar o desempenho da evolução em relação a taxa de crossover, a taxa de mutação, tamanho da população e número de gerações. Esta investigação permitiu que as particularidades do genoma frente as variáveis evolutivas fossem expostas, tornando-se uma ferramenta importante para auxiliar a escolha de

parâmetros adequados ao projeto evolutivo dos controladores do sistema de controle do pêndulo.

Para estudar o comportamento da evolução em relação a complexidade do problema o circuito foi evoluído em duas versões: a primeira versão consistiu em um detector de números primos de quatro bits, Eq. 4.1, e a segunda de um detector de números primos de seis bits, Eq. 4.2. Cada solução encontrada foi avaliada no simulador de circuitos digitais $PSPICE$ (EHW extrínseco) e a melhor solução foi comparada com a solução gerada pelo método tradicional Quine-McCluskeyCampos *et al.* (2004).

Os dados apresentados nestas evoluções mostram uma alta dependência da evolução aos parâmetros evolutivos, sendo que para o detector de 4 bits os melhores resultados foram encontrados para taxa de mutação de 5%, taxa de crossover entre 80% e 95% e populações acima de 1000 indivíduos. Para o detector de 6 bits os melhores resultados foram encontrados para taxa de mutação de $7,5\%$, taxa de crossover entre 80% e 90% e populações acima de 1500 indivíduos.

6.3 Controladores

Os controladores Proporcionais e Proporcionais-Derivativos sintonizados por meio de Algoritmos Genéticos e os controladores obtidos por EHW , foram evoluídos, comparados e avaliados quando aplicados ao controle de posição de um pêndulo não linearCampos *et al.* (2006*b*)Campos *et al.* (2006*a*).

Neste processo de comparação observamos que o Algoritmo Genético alcançou sintonia dos Ganhos K_P e K_D para o ponto ótimo do índice de desempenho, enquanto os controladores obtidos por Hardware Evolutivo alcançaram um valor de índice de desempenho inferior ao valor mínimo global do índice de desempenho apresentado para os Controladores Proporcionais e Proporcionais-DerivativosCampos *et al.* (2006*c*). Esta característica se explica pelo fato do Hardware Evolutivo ter a capacidade de explorar um espaço de soluções mais amplo, abrangendo o espaço dos controladores não lineares, enquanto os controladores tradicionais estão restritos ao espaço dos controladores lineares. As Tabelas a seguir ilustram o comportamento descrito acima:

Pêndulo Linear	K_P	J_{ISE}
Controlador P	4.00	$3,2289$
Controlador P-GA	4.31	$3,2289$
Controlador EHW	-	$2,3847$

Tabela 6.1: Pêndulo Linear, Controlador Proporcional e Controlador EHW

Pêndulo Não Linear	K_P	J_{ISE}
Controlador P	5.10	$3,1860$
Controlador P-GA	5.35	$3,1860$
Controlador EHW	-	$2,3895$

Tabela 6.2: Pêndulo Não Linear, Controlador Proporcional e Controlador EHW

Pêndulo Linear	K_P	K_D	J_{ISE}
Controlador PD	$17,35$	0.195	$1,3834$
Controlador PD-GA	$26,00$	0.20	$1,3834$
Controlador EHW	-	-	$1,2498$

Tabela 6.3: Pêndulo Linear, Controlador PD e Controlador EHW

Pêndulo Não Linear	K_P	K_D	J_{ISE}
Controlador PD	$26,00$	0.210	$1,3834$
Controlador PD-GA	$27,25$	0.22	$1,4186$
Controlador EHW	-	-	$1,2735$

Tabela 6.4: Pêndulo Não Linear, Controlador PD e Controlador EHW

Os Controladores Evoluídos apresentam ainda um excelente comportamento da resposta do sistema para sinais de entradas ausentes no processo evolutivo, como exemplo de entrada ausente tem-se a referência senoidal, conforme mostram as Figuras 6.1 e 6.2. Este comportamento demonstra que uma capacidade de generalização está embutida no processo evolutivo do controlador EHW.

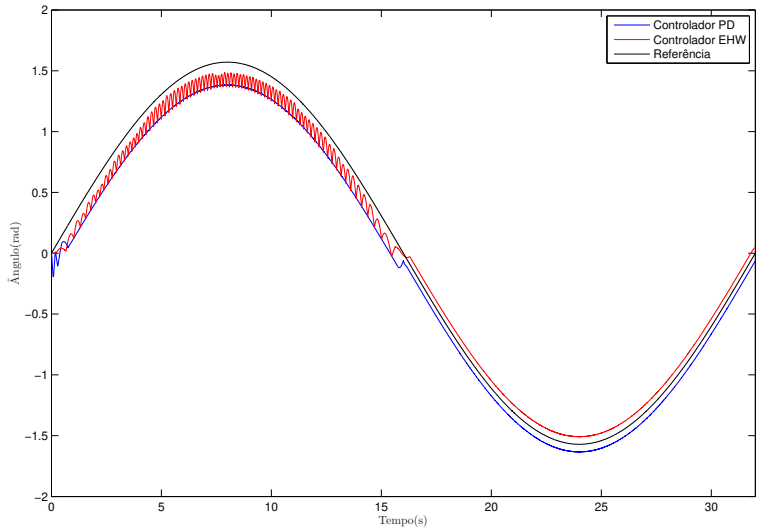


Figura 6.1: Resposta Temporal do Sistema Linear

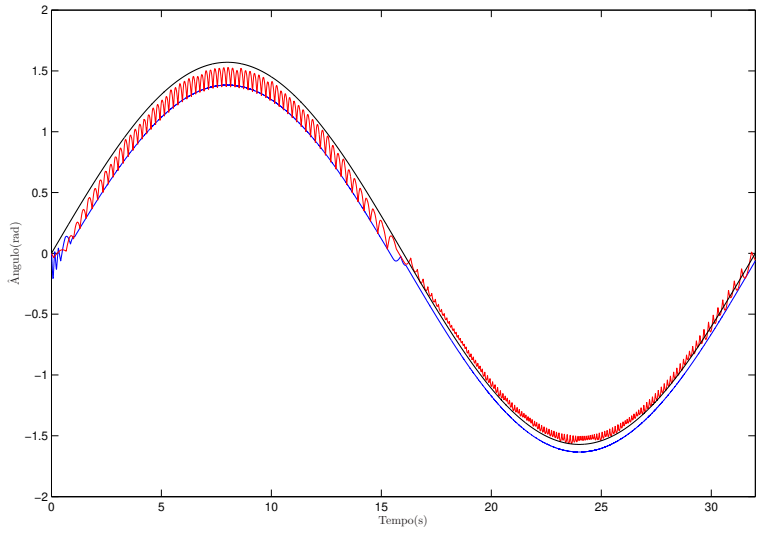


Figura 6.2: Resposta Temporal do Sistema Não Linear

6.4 Conclusão

Um campo de pesquisa de recente desenvolvimento foi estudado nesta tese. Hardware evolutivo é a área em que se estuda o projeto de circuitos eletrônicos por computação evolutiva Rothlauf *et al.* (2006). Inicialmente foram apresentados os principais conceitos e características da área, assim como as principais considerações práticas. Particularmente, este trabalho apresentou um estudo de caso, na área sistemas de controle. O objetivo deste trabalho foi projetar, implementar e analisar um sistema de controle através da técnica de hardware evolutivo.

Foram apresentados os resultados obtidos pelo controlador via hardware evolutivo para o problema de controle de posição do pêndulo, comparando-o com os resultados dos controladores P e PD sintonizados por GA. Todos os experimentos foram realizados utilizando uma formulação matemática da dinâmica de comportamento do pêndulo, conforme descrito anteriormente. No desenvolvimento do controlador por Hardware Evolutivo apenas o critério funcional foi usado como elemento de projeto, ou seja, apenas as entradas e as saídas produzidas pelo sistema foram consideradas (caixa preta). A estrutura interna do circuito digital do controlador não foi considerada, deste modo apenas ações que produzem uma saída visível ao sistema são cobertas pela função de *fitness*.

O desenvolvimento de controladores para sistemas dinâmicos não lineares é um desafio. Neste trabalho pode-se observar que a técnica de hardware evolutivo apresentou melhor resultado que os controladores tradicionais, isto se deve a abordagem *bottom up* do EHW que encontra e modela o comportamento não linear do problema, com a mesma quantidade de informação dos modelos tradicionais. Este fato indica que a técnica de hardware evolutivo é uma abordagem promissora para projeto de sistemas de controle, uma vez que os circuitos apresentados podem ser facilmente implementados nas estruturas $FPGAs$ mais simples.

6.5 Trabalhos Futuros

Embora a eficiência da abordagem proposta tenha sido comprovada através de muitos testes e simulações, algumas etapas ainda requerem experimentos e estudos mais avançados e deverão ser tratadas em trabalhos posteriores.

Este foi um trabalho preliminar desenvolvido como suporte a aplicação de um novo método para implementar sistemas de controle. Partindo dos resultados encontrados nesta implementação indica-se como trabalho futuro a comparação do controlador obtido via EHW com controladores PID assim como o desenvolvimento de hardware evolutivo mixtrínseco ou intrínseco.

Pode-se observar que estes próximos passos serão cruciais na validação da técnica como uma nova abordagem aceitável pois para o controlador PID-GA serão necessárias 16 bits de entradas, valores que encontram-se próximos a escalabilidade atual da técnica de EHW Campos *et al.* (2006*a*).

Além destes experimentos, um aumento na quantidade de bits do sistema, buscando a aproximação dos sistemas reais (maior precisão) assim como a inclusão de restrições de estabilidade e robustez no desenvolvimento e na avaliação do sistema, são abordagens que devem ser estudadas.

Por fim sugere-se que Hardware Evolutivo tenha seu comportamento comparado com outras técnicas bio-inspiradas, como redes neurais artificiais, sistemas nebulosos entre outros, de modo que seu comportamento de controle e sua implementação em Hardware possam ser comparados e avaliados, permitindo que esta nova proposta para projeto de controladores se desenvolva e alcance robustez adequado a projetos industriais.

Referências Bibliográficas

Aggarwal, Varun (2003). Evolving sinusoidal oscillators using genetic algorithms. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 67–76.

Altenberg, Lee (1994). The evolution of evolvability in genetic programming. Working Papers 94-02-007. Santa Fe Institute.

Arslan, T. e D.H. Horrocks (1995). The design of analogue and digital filters using genetic algorithms. In: *Colloquium on Digital and Analogue Filters and Filtering Systems, IEE 15th SARAGA*. pp. 2/1 – 2/5.

Aström, K. J. e T. Hägglund (1988). *Automatic Tuning of PID Controllers*. Instrument Society of America.

Bäck, T., Fogel, D.B. e Michalewicz, Z., Eds.) (2000*a*). *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publisher.

Bäck, T., Fogel, D.B. e Michalewicz, Z., Eds.) (2000*b*). *Evolutionary Computation 2: Advanced Algorithms and Operators*. Institute of Physics Publisher.

Bonatti, I. e M. Madureira (1995). *Introdução à Análise e Síntese de Circuitos Lógicos*. Editora da Unicamp.

Bradley, D. W. e Andrew M. Tyrrell (2001). The architecture for a hardware immune system. In: *Proceedings of 3th NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 193–200.

Calazans, N.L.V. (1998). *Projeto Lógico Automatizado de Sistemas Digitais Seqüenciais*. DCC/IM, COPPE Sistemas , NCE, 11^a Escola de Computação, Universidade Federal do Rio de Janeiro.

Campos, T. J., J. R. Oliveira e M. Jungbeck (2004). Programação genética aplicada ao desenvolvimento de hardware evolutivo. In: *Anais do XV Congresso Brasileiro de Automática - CBA2004*. Gramado, RS, Brasil.

Campos, T. J., J. R. Oliveira e M. Jungbeck (2006*a*). Evolutionary hardware for control design systems. In: *WSEAS Transactions on Computers*. Vol. 5. pp. 1670–1675.

Campos, T. J., J. R. Oliveira e M. Jungbeck (2006*b*). Evolvable hardware a new approach for control design. In: *7th WSEAS Int.Conf. on EVOLUTIONARY COMPUTING (EC'06*. Cavtat, Croácia.

Campos, T. J., J. R. Oliveira e M. Jungbeck (2006*c*). Hardware evolutivo aplicado ao problema de controle de servo-mecanismos. In: *Anais do XVI Congresso Brasileiro de Automática - CBA2006*. Salvador, Bahia, Brasil.

Castro, L. e F. von Zuben (2004). *Técnicas de Solução de Problemas Inspiradas na Natureza*. DCA, FEEC, Universidade Estadual de Campinas - Unicamp.

Coello, C. A., A. Christiansen e A. Aguirre (2001). Towards automated evolutionary design of combination circuits.. In: *Computer and Electrical Engineering*. pp. pp. 1 – 28.

Coello, Carlos A., Alan D. Christiansen e Hernández Aguirre (1996). Using genetic algorithms to design combinational logic circuits. *ANNIE'96. Intelligent Engineering through Artificial Neural Networks* **6**, pp. 391–396.

Cohoon, James, John Karro e Jens Lienig (2003). Evolutionary algorithms for the physical design of VLSI circuits. In: *Advances in evolutionary computing: theory and applications*. Springer-Verlag New York, Inc. New York, NY, USA. pp. 683–711.

Dawkins, Richard (1987). The evolution of evolvability. In: *Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE'87)* (Chris Langton, Ed.). Vol. 6 of *Santa Fe Institute Studies in the Sciences of Complexity*. Addison-Wesley. Los Alamos, NM, USA. pp. 201–220.

de Garis, H. (1993). Evolvable hardware : The genetic programming of darwin machines. *International Conference on Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria*.

Eiben, A.E. e J.E. Smith (2003). *Introduction to Evolutiorary Computing*. Natural Computing Series. Springer.

Ferreira, Candida (2001). Gene expression programming: a new adaptive algorithm for solving problems. *COMPLEX SYSTEMS* **13**, 87.

Fogel, L.J., A.J. Owens e M.J. Walsh (1966). *Artificial Inteligence Through Simulated Evolution*. John Wiley.

Fukunaga, Alex e Andre Stechert (1998). Evolving nonlinear predictive models for lossless image compression with genetic programming. In: *Genetic Programming 1998: Proceedings of the Third Annual Conference* (John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba e Rick Riolo, Eds.). Morgan Kaufmann. University of Wisconsin, Madison, Wisconsin, USA. pp. 95–102.

Gajski, D.D. e R.H. Kuhn (1983). New vlsi tools. *IEEE Computer* **16**(12), 11–14.

Gallagher, John C. (2003). The once and future analog alternative: Evolvable hardware and analog computation.. In: *Proceedings of the 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 43–57.

Garvie, Miguel e Adrian Thompson (2003). Evolution of self-diagnosing hardware. In: *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2003* (Andrew M. Tyrrell, Pauline C. Haddow e Jim Torresen, Eds.). Vol. 2606 of *Lecture Notes in Computer Science*. Springer. pp. 238–248.

Goodwin, G. C., S. F. Graebe e M. E. Salgado (2001). *Control System Design*. Prentice Hall.

Gordon, Timothy G. W. e Peter J. Bentley (2005). Evolving hardware. In: *Handbook of Innovative Computational Paradigms: Biological And Adaptive Computing* (Albert Zomaya, Ed.). Springer-Verlag. pp. 387–432.

Greenwood, Garrison W. e X. Song (2002). How to evolve safe control strategies. In: *Proceedings of 4th NASA/DoD Workshop on Evolvable Hardware - EH 2002*. IEEE Computer Society. Alexandria, VA, USA. pp. 129–132.

Grimbleby, James (2000). Automatic analogue circuit synthesis using genetic algorithms. In: *Proceedings of the IEE: Circuits, Devices and Systems*. Vol. 147. pp. 319–323.

Guo, Xin, Adrian Stoica, Ricardo Salem Zebulum e Didier Keymeulen (2003). Development of consistent equivalent models by mixed-model search. In: *Proceedings*

of the IASTED International Conference on Modelling and Simulation (MS 2003) (M. H. Hamza, Ed.). IASTED/ACTA Press. Palm Springs, California, USA. pp. 626–631.

Gwaltney, David A. e Michael I. Ferguson (2003). Intrinsic hardware evolution for the design and reconfiguration of analog speed controllers for a dc motor. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 81–90.

Haddow, Pauline C. e Gunnar Tufte (2001). Bridging the genotype-phenotype mapping for digital fpgas. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 109–115.

Hartmann, Morten, Per Kristian Lehre e Pauline C. Haddow (2005). Evolved digital circuits and genome complexity.. In: *Proceedings of the NASA/DoD Workshop on Evolvable Hardware - EH 2005*. IEEE Computer Society. Washington DC, USA.

Harvey, Inman e Adrian Thompson (1996). Through the labyrinth evolution finds a way: A silicon ridge. In: *Proceedings of the 1th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1996* (Tetsuya Higuchi, Masaya Iwata e Weixin Liu, Eds.). Vol. 1259 of *Lecture Notes in Computer Science*. Springer. pp. 406–422.

Higuchi, Tetsuya, Masaya Iwata, Isamu Kajitani, Hitoshi Iba, Yuji Hirao, Tatsumi Furuya e Bernard Manderick (1995). Evolvable hardware and its applications to pattern recognition and fault-tolerant systems. In: *Towards Evolvable Hardware, The Evolutionary Engineering Approach, Papers from an international workshop*. Vol. 1062 of *Lecture Notes in Computer Science*. Springer. Lausanne, Switzerland. pp. 118–135.

Holland, John H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press (2nd ed.: MIT Press, 1992). Ann Arbor, MI.

Hounsell, Ben I. e Tughrul Arslan (2000). A novel evolvable hardware framework for the evolution of high performance digital circuits.. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)* (L. Darrell Whitley, David E. Goldberg, Erick Cantú-Paz, Lee Spector, Ian C. Parmee e Hans-Georg Beyer, Eds.). Morgan Kaufmann. Las Vegas, Nevada, USA. pp. 525–.

Hounsell, Ben I. e Tughrul Arslan (2001). Evolutionary design and adaptation of digital filters within an embedded fault tolerant hardware platform. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 127–135.

Imamura, Kosuke, James A. Foster e Axel W. Krings (2000). The test vector problem and limitations to evolving digital circuits. In: *Proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware - EH 2000*. IEEE Computer Society. Palo Alto, CA, USA. pp. 75–80.

Iwata, Masaya, Isamu Kajitani, Hitoshi Yamada, Hitoshi Iba e Tetsuya Higuchi (1996). A pattern recognition system using evolvable hardware. In: *Proceedings of The 4th International Conference on Parallel Problem Solving from Nature - PPSN IV*. Vol. 1141 of *Lecture Notes in Computer Science*. Springer. Berlin, Germany. pp. 761–770.

Jungbeck, Mario (2001). Implementação de controladores neurais de kim-lewis-dawson com parâmetros otimizados por algoritmos genéticos. Dissertação de Mestrado. Universidade Estadual de Campinas - UNICAMP.

Kajitani, I., T. Hoshino, N. Kajihara, M. Iwata e T. Higuchi (1999). An evolvable hardware chip and its application as a multi-function prosthetic hand controller. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*. AAAI Press / The MIT Press. Florida, USA. pp. 182–187.

Kajitani, Isamu, Tsutomu Hoshino, Masaya Iwata e Tetsuya Higuchi (1996). Variable length chromosome ga for evolvable hardware. In: *Proceedings of the IEEE International Conference on Evolutionary Computation*. pp. 443–447.

Kalganova, Tatiana e Julian F. Miller (1999). Evolving more efficient digital circuits by allowing circuit layout evolution and multi-objective fitness. In: *Proceedings of the 1th NASA/DoD Workshop on Evolvable Hardware - EH 1999*. IEEE Computer Society. Pasadena, CA, USA. pp. 54–.

Kalganova, Tatiana, Julian F. Miller e Terence C. Fogarty (1998). Some aspects of an evolvable hardware approach for multiple-valued combinational circuit design. In: *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1998* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Vol. 1478 of *Lecture Notes in Computer Science*. Springer. pp. 78–89.

Karnaugh, M (1953). A map method for synthesis of combinational logic circuits.. In: *Transactions of the AIEE, Comunications and Electronics*. Vol. 72(I). pp. pp. 593 – 599.

Kazadi, S., Y. Qi, I. Park, N. Huang, P. Hwu, B. Kwan, W. Lue e H. Li (2001). Insufficiency of piecewise evolution.. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 223–231.

Koza, John R., Forrest H. Bennett III, David Andre e Martin A. Keane (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufman. San Francisco, CA.

Koza, John R., Martin A. Keane, Jessen Yu, Forrest H Bennett III e William Mydlowec (2000). Automatic creation of human-competitive programs and controllers by means of genetic programming. In: *Genetic Programming and Evolvable Machines*. Vol. 1. Springer Netherlands. pp. 121–164.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Li, J.H. e M.H. Lim (2003). Evolvable fuzzy system for atm cell scheduling. In: *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2003* (Andrew M. Tyrrell, Pauline C. Haddow e Jim Torresen, Eds.). Vol. 2606 of *Lecture Notes in Computer Science*. Springer. Trondheim, Norway. pp. 208–217.

Linden, Derek S. (2002). Optimizing signal strength in-situ using an evolvable antenna system. In: *Proceedings of 4th NASA/DoD Workshop on Evolvable Hardware - EH 2002*. IEEE Computer Society. Alexandria, VA, USA. pp. 147–151.

Linden, Derek S. e Edward E. Altshuler (1999). Evolving wire antennas using genetic algorithms: A review. In: *Proceedings of 1st NASA/DoD Workshop on Evolvable Hardware - EH 1999*. IEEE Computer Society. Pasadena, CA, USA. pp. 225–232.

Liu, Weixin, Masahiro Murakawa e Tetsuya Higuchi (1996). Atm cell scheduling by function level evolvable hardware. In: *Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware - ICES 96* (Tetsuya Higuchi, Masaya Iwata e Weixin Liu, Eds.). Vol. 1259 of *Lecture Notes in Computer Science*. Springer. Tsukuba, Japan. pp. 180–192.

Lohn, Jason D., Gary L. Haith, Silvano Colombano e Dimitris Stassinopoulos (1999). A comparison of dynamic fitness schedules for evolutionary design of amplifiers. In: *Proceedings of 1st NASA/DoD Workshop on Evolvable Hardware - EH 1999*. IEEE Computer Society. Pasadena, CA, USA. pp. 87–92.

Lohn, Jason D., Gregory Hornby e Derek S. Linden (2005). Evolution, re-evolution, and prototype of an x-band antenna for nasa's space technology 5 mission. In: *Proceedings of 6th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2005* (Juan Manuel Moreno, Jordi Madrenas e Jordi Cosp, Eds.). Lecture Notes in Computer Science. Springer. Sitges, Spain. pp. 205–214.

Lohn, Jason D., Gregory V. Larchev e Ronald F. DeMara (2003*a*). A genetic representation for evolutionary fault recovery in virtex fpgas. In: *Proceedings of 5th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2003* (Andrew M. Tyrrell, Pauline C. Haddow e Jim Torresen, Eds.). Vol. 2606 of *Lecture Notes in Computer Science*. Springer. pp. 47–56.

Lohn, Jason D. e Silvano Colombano (1998). Automated analog circuit sythesis using a linear representation. In: *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1998* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Vol. 1478 of *Lecture Notes in Computer Science*. Springer. pp. 125–133.

Lohn, J.D., D.S. Linden, G.S. Hornby, W.F. Kraus, A. Rodriguez-Arroyo e S.E. Seufert (2003*b*). Evolutionary design of an x-band antenna for nasa's space technology 5 mission. In: *Proceedings of the 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 155– 163.

Louis, Sushil (2003). Learning for evolutionary design. In: *Proceedings of 5nd NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA.

Louis, Sushil J. e Gregory J. Rawlins (1991). Designer genetic algorithms: Genetic algorithms in structure design. *Proceedings of the Fourth International Conference on Genetic Algorithms* pp. pp. 53–60.

Lukac, Martin, Marek Perkowski, Hilton Goi, Mikhail Pivtoraiko, Chung Hyo Yu, Kyusik Chung, Hyunkoo Jeech, Byung-Guk Kim e Yong-Duk Kim (2003). Evolutionary approach to quantum andreversible circuits synthesis. *Artificial Intelligence Review* **20**(3-4), 361–417.

Macias, Nicholas J. e Lisa J. K. Durbeck (2002). Self-assembling circuits with autonomous fault handling. In: *Proceedings of 4th NASA/DoD Workshop on Evolvable Hardware - EH 2002*. IEEE Computer Society. Alexandria, VA, USA. pp. 46–55.

Marquez, H. J. (2003). *Nonlinear Control Systems - Analysis and Design*.

Masner, Jason, John Cavalieri, James F. Frenzel e James A. Foster (1999). Representation and robustness for evolved sorting networks. In: *Proceedings of 1st NASA/DoD Workshop on Evolvable Hardware - EH 1999*. IEEE Computer Society. Pasadena, CA, USA. pp. 255–261.

McCluskey, E. J. (1956). Minimization of boolean functions.. *Bell Systems Technical Journal* **35(5)**, pp. 1417 – 1444.

Michalewicz, Z. (1996). *Genetic Algorithms+Data Structures=Evolution Programs*. Artificial Intelligence. 3 ed.. Springer-Verlag.

Miller, J. F. e P. Thomson (1995). Combinational and sequential logic optimization using genetic algorithms. pp. pp.34–38.

Miller, J. F., P. Thomson e Terence Fogarty (1998). Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. In: *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science* (D. Quagliarella, J. Périaux, C. Poloni e G. Winter, Eds.). pp. 105–131. John Wiley and Sons. Chichester.

Miller, J., T. Kalganova, N. Lipnitskaya e D. Job (1999). The genetic algorithm as a discovery engine: Strange circuits and new principles. In: *Proceedings of AISB Symposium on Creative Evolutionary Systems - CES'99*. Edinburgh, UK.

Miller, Julian F., Dominic Job e Vesselin K. Vassilev (2000). Principles in the evolutionary design of digital circuits - part ii. In: *Genetic Programming and Evolvable Machines*. Vol. 1. Springer Netherlands. pp. 259–288.

Miller, Julian F. e Keith L. Downing (2002). Evolution in materio: Looking beyond the silicon box. In: *Proceedings of 4th NASA/DoD Workshop on Evolvable Hardware - EH 2002*. IEEE Computer Society. Alexandria, VA, USA. pp. 167–176.

Miller, Julian F. e Peter Thomson (1998*a*). Aspects of digital evolution: Evolvability and architecture. *Lecture Notes in Computer Science* **1498**, 927.

Miller, Julian F. e Peter Thomson (1998*b*). Aspects of digital evolution: Geometry and learning. *Lecture Notes in Computer Science* **1478**, 25.

Miller, Julian F. e Peter Thomson (1998*c*). Evolving digital electronic circuits for real-valued function generation using a genetic algorithm. In: *Genetic Programming 1998: Proceedings of the Third Annual Conference* (John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba e Rick Riolo, Eds.). Morgan Kaufmann. University of Wisconsin, Madison, Wisconsin, USA. pp. 863–868.

Mitchell, Tom M. (1997). Machine learning meets natural language. In: *Proceedings of the 8th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence, EPIA '97*. Vol. 1323 of *Lecture Notes in Computer Science*. Springer. Coimbra, Portugal. p. 391.

Moore, E. F. (1956). Gedanken experiments on sequential machines.. *Automata Studies* pp. pp. 129 – 153.

Murakawa, Masahiro, Shuji Yoshizawa e Tetsuya Higuchi (1996). Adaptive equalization of digital communication channels using evolvable hardware. In: *Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware - ICES96* (Tetsuya Higuchi, Masaya Iwata e Weixin Liu, Eds.). Vol. 1259 of *Lecture Notes in Computer Science*. Springer. Tsukuba, Japan. pp. 379–389.

Murakawa, Masahiro, Shuji Yoshizawa, Isamu Kajitani, Xin Yao, Nobuki Kajihara, Masaya Iwata e Tetsuya Higuchi (1999). The grd chip: Genetic reconfiguration of dsps for neural network processing. *IEEE Transactions Computers* **48**(6), 628–639.

Murakawa, Masahiro, Shuji Yoshizawa, Toshio Adachi, Shiro Suzuki, Kaoru Takasuka, Masaya Iwata e Tetsuya Higuchi (1998). Analogue ehw chip for intermediate frequency filters. In: *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1998* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Vol. 1478 of *Lecture Notes in Computer Science*. Springer. pp. 134–143.

Plante, Jeannette, Harry Shaw, Lisa P. Mickens e Charles T. Johnson-Bey (2003). Overview of field programmable analog arrays as enabling technology for evolvable hardware for high reliability systems. In: *Proceedings of the 5th NASA/DoD*

Workshop on Evolvable Hardware - EH 2003. IEEE Computer Society. Chicago, IL, USA. pp. 77–80.

Rechenberg, I. (1973). *Evolutionsstratgie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*. Frommann-Holzboog Verlag. Stuttgart.

Rothlauf, Franz, Branke, Jürgen, Cagnoni, Stefano, Costa, Ernesto, Cotta, Carlos, Drechsler, Rolf, Lutton, Evelyne, Machado, Penousal, Moore, Jason H., Romero, Juan, Smith, George D., Squillero, Giovanni e Takagi, Hideyuki, Eds.) (2006). *Proceedings of EvoWorkshops: Applications of Evolutionary Computing, EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, and EvoSTOC*. Vol. 3907 of *Lecture Notes in Computer Science*. Springer. Budapest, Hungary.

Rumelhart, David E., Bernard Widrow e Michael A. Lehr (1994). The basic ideas in neural networks. *Commun. ACM* **37**(3), 87–92.

Sakanashi, H., M. Iwata e T. Higuchi (2001). A lossless compression method for halftone images using evolvable hardware. In: *Proceedings of 4th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2001* (Y. Liu, K. Tanaka, M. Iwata, T. Higuchi e M. Yasunaga, Eds.). Vol. 2210 of *Lecture Notes in Computer Science*. Springer Verlag. Tokyo, Japan. pp. 314–326.

Schaller, R. R. (1997). Moore's law: past, present, and future.. *IEEE Spectrum* pp. pp. 52 – 59.

Sechen, Carl (1988). Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing.. In: *DAC*. pp. 73–80.

Sekanina, Lukás (2003a). Towards evolvable ip cores for fpgas. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 145–154.

Sekanina, Lukás (2002). Evolution of digital circuits operating as image filters in dynamically changing environment. In: *Mendel 2002 - 8th International Conference on Soft Computing*. Brno University of Technology. pp. 33–38.

Sekanina, Lukás e Richard Ruzicka (2003). Easily testable image operators: The class of circuits where evolution beats engineers. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 135–144.

Sinohara, Helio Takahiro, Marco Aurélio Cavalcanti Pacheco e Marley B. R. Vellasco (2001). Repair of analog circuits: Extrinsic and intrinsic evolutionary techniques. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 44–47.

Stoica, Adrian, Alex S. Fukunaga, Ken Hayworth e Carlos Salazar-Lazaro (1998). Evolvable hardware for space applications. In: *Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware - ICES98* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Vol. 1478 of *Lecture Notes in Computer Science*. Springer. Lausanne, Switzerland. pp. 166–173.

Stoica, Adrian, Didier Keymeulen e Ricardo Salem Zebulum (2001). Evolvable hardware solutions for extreme temperature electronics. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 93–97.

Stoica, Adrian, Didier Keymeulen, Raoul Tawel, Carlos Salazar-Lazaro e Wei te Li (1999). Evolutionary experiments with a fine-grained reconfigurable architecture for analog and digital cmos circuits. In: *Proceedings of 1st NASA/DoD Workshop on Evolvable Hardware - EH 1999*. IEEE Computer Society. Pasadena, CA, USA. pp. 76–84.

Stoica, Adrian, Didier Keymeulen, Ricardo Salem Zebulum, Michael I. Ferguson, T. Daud e T. Arslan (2004). Evolvable hardware for autonomous systems. In: *Proceedings of 6th NASA/DoD Workshop on Evolvable Hardware - EH 2004*. IEEE Computer Society. Seattle, Washington, USA.

Stoica, Adrian, Ricardo Salem Zebulum, Didier Keymeulen e Jason Lohn (2002). On polymorphic circuits and their design using evolutionary algorithms. In: *Proceedings of IASTED International Conference on Applied Informatics - AI2002*. Innsbruck, Austria.

Stoica, Adrian, Ricardo Salem Zebulum e Didier Keymeulen (2000). Mixtrinsic evolution. In: *Proceedings of the 3th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2000* (Julian F. Miller, Adrian Thompson, Peter Thomson e Terence C. Fogarty, Eds.). Vol. 1801 of *Lecture Notes in Computer Science*. Springer. pp. 208–217.

Stoica, Adrian, Ricardo Salem Zebulum, Xin Guo, Didier Keymeulen, Michael I. Ferguson e Vu Duong (2003). Silicon validation of evolution-designed circuits. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 21–25.

Sundaralingam, Seelan e Ken Sharman (1998). Evolving iir filters in multipath environments. In: *Proceedings of the 7th International Conference on Evolutionary Programming - EP98* (V. William Porto, N. Saravanan, Donald E. Waagen e A. E. Eiben, Eds.). Vol. 1447 of *Lecture Notes in Computer Science*. Springer. San Diego, CA, USA. pp. 397–406.

Surkan, Alvin J. e Amiran Khuskivadze (2002). Evolution of quantum algorithms for computer of reversible operators. In: *Proceedings of 4th NASA/DoD Workshop on Evolvable Hardware - EH 2002*. IEEE Computer Society. Alexandria, VA, USA. pp. 186–190.

Suzim, A.A. (1988). A cad frame for vlsi design. *Simpósio Brasileiro de Concepção de Circuitos Integrados*.

Tessier, Russell G. (1999). Fast Place and Route Approaches for FPGAs. Tese de Doutourado. Massachusetts Institute of Technology - MIT.

Thompson, A. (2002). Notes on design through artificial evolution: Opportunities and algorithms. In: *Adaptive computing in design and manufacture V* (I. C. Parmee, Ed.). Springer-Verlag. pp. 17–26.

Thompson, Adrian (1995). Evolving electronic robot controller that exploit hardware resources. In: *Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life - ECAL* (Federico Morán, Alvaro Moreno, Juan J. Merelo Guervós e Pablo Chacón, Eds.). Vol. 929 of *Lecture Notes in Computer Science*. Springer. Granada, Spain. pp. 640–656.

Thompson, Adrian (1996). An evolved circuit, intrinsic in silicon, entwined with physics. In: *Proceedings of the 1th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1996* (Tetsuya Higuchi, Masaya Iwata e Weixin Liu, Eds.). Vol. 1259 of *Lecture Notes in Computer Science*. Springer. pp. 390–405.

Thompson, Adrian (1998). On the automatic design of robust electronics through artificial evolution. In: *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1998* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Vol. 1478 of *Lecture Notes in Computer Science*. Springer. pp. 13–24.

Thompson, Adrian e Christoph Wasshuber (2000). Evolutionary design of single electron systems. In: *Proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware - EH 2000*. IEEE Computer Society. Palo Alto, CA, USA. pp. 109–116.

Thompson, Adrian e Paul J. Layzell (2000). Evolution of robustness in an electronics design. In: *Proceedings of the 3th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2000* (Julian F. Miller, Adrian Thompson, Peter Thomson e Terence C. Fogarty, Eds.). Vol. 1801 of *Lecture Notes in Computer Science*. Springer. pp. 218–228.

Thomson, Robert e Tughrul Arslan (2003). The evolutionary design and synthesis of non-linear digital vlsi systems. In: *Proceedings of the 3rd NASA/DoD Workshop on Evolvable Hardware - EH 2001*. IEEE Computer Society. Long Beach, CA, USA. pp. 125–134.

Torresen, Jim (2000). Possibilities and limitations of applying evolvable hardware to real-world applications.. In: *Proceedings of The 10th International Workshop on Field-Programmable Logic and Applications: Roadmap to Reconfigurable Computing - FPL* (Reiner W. Hartenstein e Herbert Grünbacher, Eds.). Vol. 1896 of *Lecture Notes in Computer Science*. Springer. Villach, Austria. pp. 230–239.

Torresen, Jim (2001). Two-step incremental evolution of a prosthetic hand controller based on digital logic gates. In: *Proceedings of 4th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2001* (Y. Liu, K. Tanaka, M. Iwata, T. Higuchi e M. Yasunaga, Eds.). Vol. 2210 of *Lecture Notes in Computer Science*.

Tufte, Gunnar e Pauline C. Haddow (2000). Evolving an adaptive digital filter. In: *Proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware - EH 2000*. IEEE Computer Society. Palo Alto, CA, USA. pp. 143–150.

Tyrrell, Andrew M., Eduardo Sanchez, Dario Floreano, Gianluca Tempesti, Daniel Mange, Juan Manuel Moreno, Jay Rosenberg e Alessandro E. P. Villa (2003). Poetic tissue: An integrated architecture for bio-inspired hardware. In: *Proceedings of the 5th International Conference on Evolvable Systems: From Biology to Hardware - ICES 2003* (Andrew M. Tyrrell, Pauline C. Haddow e Jim Torresen, Eds.). Vol. 2606 of *Lecture Notes in Computer Science*. Springer. pp. 129–140.

Vassilev, Vesselin K., Dominic Job e Julian F. Miller (2000). Towards the automatic design of more efficient digital circuits. In: *Proceedings of 2nd NASA/DoD Workshop on Evolvable Hardware - EH 2000*. IEEE Computer Society. Palo Alto, CA, USA. pp. 151–160.

Vigander, Sverre (2001). Evolutionary fault repair of electronics in space applications. Tese de Doutourado. University of Sussex. Galmer, Brighton, UK.

Vinger, Knut Arne e Jim Torresen (2003). Implementing evolution of fir-filters efficiently in an fpga. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 26–32.

Yao, Xin e Tetsuya Higuchi (1999). Promises and challenges of evolvable hardware. *IEEE Transactions on Systems, Man, and Cybernetics, Part C* **29**(1), 87–97.

Yih, J.-S. e P. Mazumder (1990). A neural network design for circuit partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **9**(12), 1265–1271.

Zebulum, Ricardo Salem (1999). Síntese de Circuitos Eletrônicos por Computação Evolutiva. Doctor. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, Brasil.

Zebulum, Ricardo Salem, Didier Keymeulen, Vu Duong, Xin Guo, Michael I. Ferguson e Adrian Stoica (2003). Experimental results in evolutionary fault-recovery for field programmable. In: *Proceedings of 5th NASA/DoD Workshop on Evolvable Hardware - EH 2003*. IEEE Computer Society. Chicago, IL, USA. pp. 192–198.

Zebulum, Ricardo Salem, Marco Aurélio Cavalcanti Pacheco e Marley B. R. Vellasco (1996). Evolvable systems in hardware design: Taxonomy, survey and applications. In: *Proceedings of The First International Conference on Evolvable Systems: From Biology to Hardware - ICES96* (Tetsuya Higuchi, Masaya Iwata e Weixin Liu, Eds.). Vol. 1259 of *Lecture Notes in Computer Science*. Springer. Tsukuba, Japan. pp. 344–358.

Zebulum, Ricardo Salem, Marco Aurélio Cavalcanti Pacheco e Marley B. R. Vellasco (1998). Analog circuits evolution in extrinsic and intrinsic modes. In: *Proceedings of the 2th International Conference on Evolvable Systems: From Biology to Hardware - ICES 1998* (Moshe Sipper, Daniel Mange e Andrés Pérez-Uribe, Eds.). Lecture Notes in Computer Science. Springer. pp. 154–165.

Zebulum, Ricardo Salem, Marco Aurelio Pacheco e Marley Vellasco (1997). Increasing length genotypes in evolutionary electronics. In: *Workshop on Variable Length Genotypes, Int. Conf. on Genetic Algorithms (ICGA97)*. East Lansing, MI, USA.