Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciência

Faculdade de Engenharia

Fernando Schlemm Ribeiro

Plataforma de desenvolvimento de circuitos eletrônicos adaptativos

Rio de Janeiro

2012

Fernando Schlemm Ribeiro

Plataforma de desenvolvimento de circuitos eletrônicos adaptativos

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de Concentração: Sistemas Inteligentes e Automação.

Orientador: Prof. Dr. José Franco Machado do Amaral

Coorientador: Prof. Dr. Jorge Luís Machado do Amaral

Rio de Janeiro

2012

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRUS / BIBLIOTECA CTC/B

R484 Ribeiro, Fernando Schlemm.
Plataforma de desenvolvimento de circuitos eletrônicos adaptativos / Fernando Schlemm Ribeiro – 2012.
89f.

Orientador: José Franco Machado do Amaral.
Coorientador: Jorge Luís Machado do Amaral.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Elétrica. 2. Circuitos eletrônicos - Dissertação. 3. Algoritmos genéticos - Dissertação. I. Amaral, José Franco Machado do. II. Universidade do Estado do Rio de Janeiro – Faculdade de Engenharia. III. Título.

CDU 621.38.02

Assinatura

Data

Fernando Schlemm Ribeiro

Plataforma de Desenvolvimento de Circuitos Eletrônicos Adaptativos

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de Concentração: Sistemas Inteligentes e Automação.

Aprovado em: 29 de fevereiro de 2012.

Banca Examinadora:

Prof. Dr. José Franco Machado do Amaral (Orientador)
Faculdade de Engenharia - UERJ

Prof. Dr. Jorge Luís Machado do Amaral (Coorientador)
Faculdade de Engenharia - UERJ

Prof. Dr. Luiz Biondi Neto
Faculdade de Engenharia - UERJ

Prof. Dr. Paulo Sérgio Rodrigues Alonso
Faculdade de Engenharia - UERJ

Prof. Dr. Antonio Carneiro de Mesquita Filho
COPPE – Universidade Federal do Rio de Janeiro

Prof. Dr. Juan Guillermo Lazo Lazo
PUC-Rio

Rio de Janeiro

2012

DEDICATÓRIA

A minha mulher Nina e meu filho João Pedro.

AGRADECIMENTOS

Ao meu mestre e orientador José Franco Machado do Amaral, pela amizade, interesse constante, paciência, sabedoria e apoio irrestrito nos momentos mais difíceis do trabalho.

A Jorge Luís Machado do Amaral, mestre e amigo, pela ajuda e disponibilidade sem as quais eu não teria chegado a completar este trabalho.

Aos professores do Programa de Pós-Graduação em Engenharia Eletrônica da UERJ, por terem me recebido em minhas dúvidas e me fornecido apoio intelectual.

Ao meu amigo Américo de Castro, pela compreensão acima do normal nas minhas ausências e pelo que me foi dado de mais caro: o tempo necessário para trabalhar.

Mudam-se os tempos, mudam-se as vontades,
Muda-se o ser, muda-se a confiança;
Todo o mundo é composto de mudança,
Tomando sempre novas qualidades.

Continuamente vemos novidades,
Diferentes em tudo da esperança;
Do mal ficam as mágoas na lembrança,
E do bem, se algum houve, as saudades.

Luís Vaz de Camões

RESUMO

RIBEIRO, Fernando Schlemm. **Plataforma de desenvolvimento de circuitos eletrônicos adaptativos**. 2012. 89f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro. Rio de Janeiro, 2012.

Este trabalho apresenta uma arquitetura geral para evolução de circuitos eletrônicos analógicos baseada em algoritmos genéticos. A organização lógica privilegia a interoperabilidade de seus principais componentes, incluindo a possibilidade de substituição ou melhorias internas de suas funcionalidades. A plataforma implementada utiliza evolução extrínseca, isto é, baseada em simulação de circuitos, e visa facilidade e flexibilidade para experimentação. Ela viabiliza a interconexão de diversos componentes aos nós de um circuito eletrônico que será sintetizado ou adaptado. A técnica de Algoritmos Genéticos é usada para buscar a melhor forma de interconectar os componentes para implementar a função desejada. Esta versão da plataforma utiliza o ambiente MATLAB com um "*toolbox*" de Algoritmos Genéticos e o PSpice como simulador de circuitos. Os estudos de caso realizados apresentaram resultados que demonstram a potencialidade da plataforma no desenvolvimento de circuitos eletrônicos adaptativos.

Palavras-chave: Eletrônica evolucionária. Plataformas configuráveis. Hardware adaptativo. Sistemas adaptativos. Sistemas evolucionários. Algoritmos genéticos. Circuitos eletrônicos.

ABSTRACT

This work presents a general architecture of an evolutionary system for electronic analog circuits based on genetic algorithms. The platform design enables interoperability of its main components including module substitution or functionality improvement. In the current version it implements the extrinsic model, that means, circuit simulation aiming the flexibility and easy experimentation. It enables free interconnection on a number of nodes of a circuit to be synthesized or adapted. The evolutionary technique – Genetic Algorithms – is used to search for the best interconnection solution on the desired circuit or circuit function. In the current version it makes use of the MATLAB with a genetic algorithm toolbox and the PSpice to simulate circuits. The case studies presented demonstrate the potential of the platform to adapt electronic circuits.

Keywords: Evolutionary electronics. Reconfigurable platforms. Adaptive hardware. Adaptive systems. Evolutionary systems. Genetic algorithms. Electronic circuits.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

EHW	Evolvable Hardware
GA	Genetic Algorithms
EA	Evolvable Algorithms
AHS	Adaptive Hardware Systems
FPGA	Field Programmable Gate Array
FPAA	Field Programmable Analog Array
FPTA	Field Programmable Transistor Array
PROM	Programmable Read Only Memory
PAL	Programmable Array Logic
PLA	Programmable Logic Array
PLS	Programmable Logic Sequencer
GAL	Gate Array Logic
CPLD	Complex Programmable Logic Device
SRAA	Self-Reconfigurable Analog Array
SoC	System-on-Chip
GAOT	Genetic Algorithm for Optimization Toolbox
MATLAB	Matrix Laboratory
SPICE	Simulation Program with Integrated Circuit Emphasis
PAMA-NG	Programmable Analog Multiplexer Array – Next Generation

SUMÁRIO

INTRODUÇÃO

Motivação

Adaptabilidade pode ser definida como a capacidade de uma estrutura reagir a eventos internos ou externos de forma a manter sua estabilidade funcional (Amaral, 2003). Mesmo considerando que normalmente o foco principal é manter uma funcionalidade básica, este mesmo conceito pode ser estendido a adaptações mais extremas onde a estrutura poderia mapear novas funcionalidades e prover funções além das inicialmente oferecidas.

Neste sentido, o ser humano, ao acrescentar ferramentas ao seu cotidiano, mostra como é possível se adaptar. O uso de ferramentas permite novas interações que produzem modificações: no cérebro, no corpo e nas expectativas. Estas alterações produzem novas necessidades, e acabam por ampliar o espaço de atuação, além de ter o efeito de permitir maior especialização e refinamento funcional. Novos processos e ferramentas são gerados a partir da experiência e desta forma outro ciclo se inicia.

Assim, de forma semelhante, os sistemas eletrônicos podem ser concebidos e dotados de recursos que permitam adaptação a novas condições operacionais, por conta de falhas, variação de parâmetros internos ou condições externas (Lovay et al., 2010). Os sistemas adaptativos podem procurar em seus próprios recursos - hardware ou software – possíveis soluções para manter suas funções originais dentro de padrões operacionais mínimos.

Estes mesmos sistemas podem ser projetados com módulos redundantes ou com circuitos tolerantes a falha, de forma a maximizar as possibilidades de manutenção da especificação mesmo depois da ocorrência de falhas que poderiam ser classificadas como catastróficas ou inviabilizantes para os padrões funcionais esperados (Szász et al., 2010).

Adaptabilidade e sustentabilidade funcionais são dois conceitos que podem ser usados em conjunto na elaboração de sistemas que estejam submetidos a regimes de trabalho sem possibilidade de manutenção ou re-calibragem periódica, condições hostis de trabalho, submetidos a grandes variações de condições externas ou a grandes distâncias, orbitando, lançados no espaço em missões ou que atuam em grandes profundidades (Schultz et al., 1992).

Adaptação é um fundamento no campo de estudo da eletrônica evolucionária, no que diz respeito ao “hardware”, pode ser definida como “a capacidade de modificar a si mesmo para *manter* ou *melhorar* seu desempenho em função de objetivos internos e/ou em resposta a mudanças no ambiente (interno ou externo)” (Stoica et al., 2007).

O prolongamento do tempo de vida, a tolerância a falha e a possibilidade de atualização de produtos comerciais, de forma que estes não se percam por falhas simples ou possam ser reusados, são também de grande interesse.

Outro ponto de interesse é a crescente demanda pela síntese de circuitos analógicos mais complexos, para interagir com o meio ambiente e a necessidade de projetá-los com mais rapidez para o mercado (Amaral, et al., 2007; Sá, 2009; Otero et al., 2011). Isto impõe práticas de projeto mais dinâmicas e que possam gerar produtos em tempo cada vez menor (Lohn et al., 1999).

A Eletrônica Evolucionária viabiliza o desenvolvimento de práticas, métodos, algoritmos e estruturas de software e/ou hardware que nos permitam evoluir na concepção de circuitos mais robustos. Esta terminologia abrange um amplo campo de pesquisa de utilização de algoritmos evolutivos em otimização e síntese de circuitos eletrônicos.

Objetivos

O objetivo principal deste trabalho é implementar uma plataforma de evolução extrínseca (implementação em software) que forneça um ambiente adequado para síntese e pesquisa de circuitos eletrônicos utilizando algoritmos evolucionários. Nesta plataforma poderão ser estudados aspectos relativos à adaptabilidade em situação de falha, imunidade em relação a interferências internas e externas ou variação de parâmetros. Ela também deve permitir a síntese de circuitos a partir de especificações de entrada e saída definidas.

A arquitetura geral da plataforma concebida permite os seguintes tipos de evolução: extrínseca, intrínseca ou ambas (híbrida); entretanto focamos na implementação da abordagem extrínseca. Os circuitos devem ser principalmente baseados em componentes discretos de forma que seja possível projetar, simular e verificar a capacidade de adaptação dos mesmos. Blocos funcionais customizados que puderem ser descritos funcionalmente no PSpice também podem ser evoluídos nesta plataforma.

Objetivou-se também a concepção da plataforma de maneira que outros tipos de investigação fossem possíveis, ou que, pelo menos, não inviabilizasse o estudo de: Programação Genética, paralelismo na arquitetura do algoritmo com vistas à melhoria de performance e paralelismo na estrutura das populações ou sub-populações (Nettleton et al., 1995).

O algoritmo genético que serviu de base para este trabalho foi estudado de forma a garantir que pudessem ser feitas alterações em suas estruturas internas com o objetivo de

permitir principalmente a implementação de outras estratégias de: mutação, cruzamento, geração da população inicial e distribuição de demanda na avaliação de indivíduos.

Descrição do Trabalho

O trabalho começou com a pesquisa bibliográfica na área de eletrônica evolucionária e circuitos adaptativos. Trabalhos relacionados com tolerância a falha e circuitos auto-imunes foram estudados de forma a dar maior amplitude e profundidade ao conteúdo.

O foco do trabalho foi a implementação, em software (extrínseco), da PAMA-NG (*Programmable Analog Multiplexer Array – Next Generation*) (Amaral, 2003). A PAMA-NG é uma plataforma de síntese de circuitos analógicos diretamente em hardware (intrínseco).

Também foi dada atenção ao estudo de publicações sobre novas alternativas de plataformas de evolução de circuitos. Observou-se uma tendência a se utilizar circuitos pré-fabricados com blocos nativos de Algoritmos Genéticos (AG) configuráveis pelo usuário (Fernando et al., 2010) para serem incorporados a circuitos eletrônicos: em escala industrial para produtos comerciais, em aplicações críticas – militares, espaciais e sub-aquáticas, ou em aplicações específicas: equipamentos de teste, ajuste de circuitos em tempo de produção, outros (Higuchi et al., 1999).

A plataforma extrínseca implementada contempla o uso de três componentes principais: o software de simulação de circuitos, o MATLAB e um pacote de algoritmos genéticos que é executado de no ambiente do MATLAB.

<u>O Simulador de Circuitos Eletrônicos (PSpice)</u>

O simulador de circuitos PSpice (*Simulation Program with Integrated Circuit Emphasis*, o "P" inicial se refere a função "*probe*" que permite plotagem gráfica das simulações) é difundido mundialmente e suporta diversos desenvolvimentos científicos em suas diferentes versões, incluindo as gratuitas. Já se mostrou indubitavelmente robusto e confiável para os propósitos da implementação proposta neste trabalho.

Em suas versões mais antigas, e nem por isso menos valiosas para este trabalho, o PSpice permite chamada a suas funções usando a linha de comando e gera resultados em arquivos texto (*.OUT*) e arquivos de dados organizados em registros (*.DAT*).

O PSpice permite instanciação simultânea de sua interface (várias chamadas simultâneas), o que nos permite efetuar experiências com paralelismo (ainda que nesta situação específica possa haver forte concorrência em função de características de processamento, barramento e sistema operacional da máquina onde o sistema venha a ser executado).

Neste trabalho a versão do PSpice utilizada é: Evaluation Version (january 1994). Esta versão foi escolhida por estar disponível nos laboratórios de Graduação e Pós-Graduação da instituição.

O MATLAB

O MATLAB ("*Matrix Laboratory*") foi escolhido para integrar a plataforma pelos seguintes motivos:

- Está disponível nos laboratórios de graduação e pós-graduação da UERJ;
- Possui suporte para as demais áreas de Sistemas Inteligentes que são desenvolvidas na UERJ;
- É de conhecimento geral e já existem diversos trabalhos publicados, interna e externamente, que fazem uso do MATLAB como suporte;
- Permite reuso de código e alteração do comportamento de seus componentes para os propósitos específicos que nos interessam na plataforma: avaliar e pesquisar diferentes formas de síntese e evolução de circuitos objetivando principalmente: tolerância a falha, auto-reparo e adaptação.

Nesta versão da plataforma, um programa desenvolvido no ambiente do MATLAB atua como "maestro" do processo, pois o algoritmo genético e os processos de simulação de circuito e avaliação de aptidão ocorrem sob sua coordenação.

O Algoritmo Genético

O GAOT ("*Genetic Algorithms for Optimization Toolbox*") implementa o algoritmo genético utilizado no trabalho. Trata-se de um "*toolbox*" para o MATLAB, desenvolvido por Houck (Houck, C. R. et al., 1995) com módulos que são funções MATLAB distribuídas em

vários arquivos de programa tipo "*.m*", o que nos permite fazer chamadas específicas a cada função e também nos dá a flexibilidade de alterar o comportamento de alguma função de interesse.

Este pacote fornece todas as principais funções que devem ser implementadas em um algoritmo genético convencional, destacando-se:

- Gerador aleatório de População Inicial
- Escolha dos genes com representação binária ou número real
- Configuração dos parâmetros de Mutação e Cruzamento
- Escolha do método de Cruzamento e Mutação
- Escolha do processo de seleção de indivíduos
- Escolha das condições de término da evolução.

Organização do Trabalho

Esta dissertação está dividida em cinco capítulos conforme descrito a seguir.

O capítulo 2 apresenta conceitos básicos de Eletrônica Evolucionária e Hardware Adaptativo visando defini-los segundo o ponto de vista deste trabalho. Uma vasta gama de nomes e conceitos tem sido usada para representar elementos muito semelhantes, o que nos fez tomar o cuidado de referenciar os conceitos segundo nosso entendimento atual.

O capítulo 2 também discute alguns pontos importantes que nortearam a concepção da plataforma proposta e de suas potencialidades. Uma atualização sobre plataformas também é apresentada neste capítulo.

O capítulo 3 apresenta a arquitetura geral da plataforma extrínseca implementada e seus detalhes constitutivos.

O capítulo 4 aborda o estudo de casos e uma série de observações e resultados obtidos que demonstram a potencialidade da plataforma.

A conclusão deste trabalho é apresentada no capítulo 5 juntamente com outras linhas investigativas e com uma relação de possíveis trabalhos futuros.

1 ELETRÔNICA EVOLUTIVA E HARDWARE ADAPTATIVO

1.1 Introdução

O termo “Eletrônica Evolucionária” foi cunhado em 1997, a partir do desejo de se formalizar um nome para a então nova área de pesquisa que estava se desenhando em função dos resultados de pesquisadores do mundo todo, principalmente: Europa, Estados Unidos e Japão. A Eletrônica Evolucionária cobre os desenvolvimentos e assuntos relacionados a uso de Computação Evolucionária no projeto de circuitos eletrônicos (Salazar & Mesquita, 2000) (Zebulum et al., 2001).

O termo “Computação Evolucionária” se deve ao uso de “Algoritmos Evolucionários” na solução de problemas genéricos. Algoritmos Evolucionários, por sua vez, são processos inspirados em mecanismos naturais de evolução, como proposto por Charles Darwin e na genética. Os principais algoritmos evolucionários são:

- Algoritmos Genéticos
- Programação Genética

O pesquisador J. H. Holland e outros pesquisadores da Universidade de Michigan ainda nos anos 60 firmaram as bases dos “Algoritmos Genéticos”. Fogel e outros cientistas (Fogel L.J. et al., 1966), em 1966, apresentaram as bases da “Programação Genética” (Nettelton et al., 1995) que mais tarde foi desenvolvida principalmente por John Koza (Koza, J. et al., 1997).

A genética é *um meio* pelo qual um indivíduo evolui. A evolução também pode ocorrer por aprendizado ou por alteração estrutural ou funcional, o que leva ao conceito de “adaptação”, que é um meio pelo qual um sistema ou indivíduo tem alterada parte de suas características em função de necessidades internas ou externas.

Os Algoritmos Genéticos são algoritmos matemáticos inspirados nos mecanismos de evolução natural e recombinação genética. A técnica de algoritmos genéticos fornece um mecanismo de busca adaptativa que se baseia no princípio Darwiniano de reprodução, sobrevivência dos mais aptos e mutação genética.

Os Sistemas Adaptativos são aqueles que, por algum meio, podem se reconfigurar ou se desenvolver devido a condições internas ou externas. O “*hardware*” adaptativo é um caso particular em que o circuito pode, por seus próprios meios, recompor satisfatoriamente suas funcionalidades em função de condições internas de operação ou novas exigências operacionais causadas por alteração no ambiente em que o circuito opera.

1.1.1 Algoritmos Genéticos

Os Algoritmos Genéticos foram desenvolvidos por John Holland e outros pesquisadores na Universidade de Michigan ainda nos anos 60 (Holland, 1975). A principal vantagem no uso de algoritmos genéticos é seu desempenho em problemas com espaços de busca muito complexos ou grandes, o que inviabiliza o uso de técnicas convencionais. Os algoritmos genéticos, com uma população de soluções candidatas, usam regras probabilísticas para promover a evolução de outras gerações de soluções. Os mecanismos usados tentam imitar os processos de evolução biológica Darwiniana.

Estruturalmente os algoritmos genéticos são compostos por:

- Uma representação do modelo de solução através de um cromossoma
- Um processo de seleção
- O uso de operadores genéticos: cruzamento e mutação
- Uma função de avaliação da aptidão de cada indivíduo

A representação do problema corresponde ao mapeamento das possíveis soluções em uma estrutura de dados que possa ser manipulada computacionalmente. Os Algoritmos Genéticos geralmente codificam as possíveis soluções em palavras binárias denominadas cromossomas (Amaral, 2003).

Os cromossomas contem “genes” que representam a parte estrutural da solução que se busca. No caso deste trabalho, buscamos a relação de ligação dos terminais de componentes (pinos) com os nós do circuito candidato, portanto, como veremos mais adiante, cada gene representa um pino ou terminal de componente e o conteúdo do gene receberá o valor de um dos possíveis nós do circuito.

A Figura 1 mostra a representação de um cromossoma contendo três resistores e um transistor, cada terminal corresponde a um gene que poderá conter um valor de nó. A figura mostra a representação do cromossoma [5 4 4 3 1 2 1 3 0] supondo que +5Vdc, Vin, Vout e GND estão ligados, respectivamente, aos nós 2, 5, 1 e 0.

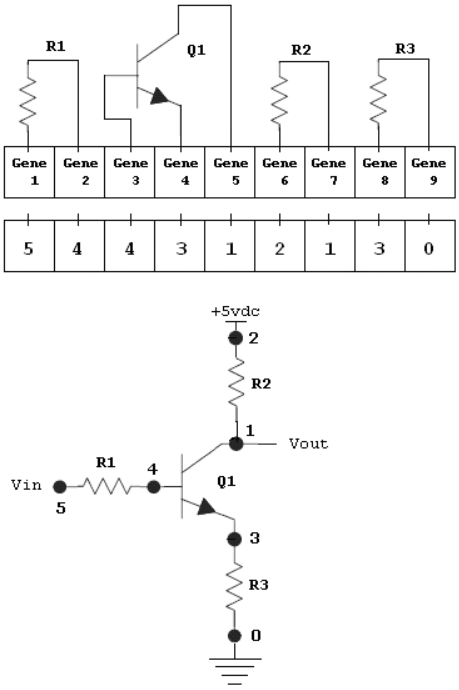


Figura 1 - Exemplo de representação cromossomial típica da técnica dos Algoritmos Genéticos

O operador de seleção é implementado de forma a privilegiar a sobrevivência dos mais aptos (melhor avaliados). A aptidão dos indivíduos é diretamente proporcional à probabilidade deles serem selecionados e contribuírem para a criação de indivíduos em uma próxima geração, em outras palavras, quanto melhor avaliado menor chance de ser descartado. A seleção é um processo probabilístico.

Após o processo de seleção, os indivíduos selecionados são escolhidos aleatoriamente aos pares para serem recombinados através do operador de cruzamento - "*crossover*". Este operador executa uma troca de informações entre dois indivíduos a partir do conteúdo de seus cromossomas, ou seja, faz um corte em algum lugar no meio da cadeia de genes e troca os conteúdos entre ambos. Desta recombinação resultam dois novos indivíduos descendentes dos primeiros, apresentando o material genético de seus progenitores. A aplicação ou não do cruzamento após a escolha de dois cromossomas é probabilística, isto é, dois novos indivíduos podem ser formados, substituindo os pais na próxima geração. A taxa de aplicação do operador de "*crossover*" é em geral alta (> 60%) (Goldberg, 1989 apud Amaral, 2003).

Após o cruzamento, será aplicado o operador de mutação. Este operador também é aplicado com uma determinada taxa de probabilidade, em geral bem mais baixa que a de cruzamento, desta forma alguns indivíduos podem ser alterados por este operador. O operador de mutação atua na população como um modificador que tem o poder de criar uma

variabilidade genética, potencializando a cobertura do espaço de busca e facilitando a convergência para uma solução viável.

Vários trabalhos publicados apresentam modificações nos operadores de mutação e cruzamento e adaptação das respectivas taxas, com o objetivo de favorecer o processo evolucionário sem comprometer a eficácia do algoritmo em buscar a melhor solução e evitar soluções particulares (mínimos ou máximos locais) (Xiangzhong et al., 2007; Ren Yu et al., 2010; Sinohara, 2001).

A função de avaliação de aptidão tem como finalidade atribuir um grau de aptidão – nota, ou avaliação - para o indivíduo. Esta mesma "nota" é a que será usada na fase de seleção de indivíduos que comporão a próxima geração, portanto, a avaliação é a função mais crítica do processo e também a mais dependente da especificação ou objetivo do problema.

No caso da plataforma apresentada neste trabalho a função de avaliação compara a curva de saída de um circuito simulado no PSpice com uma curva pré-estabelecida (padrão) e fornece uma nota conforme sua similaridade (minimização do erro médio), portanto, neste caso, tentamos minimizar o erro médio em relação à curva de saída desejada. Quando o erro médio está próximo de zero significa que a nota será alta e que temos um bom candidato à solução final.

A Figura 2 apresenta o diagrama de funcionamento de um algoritmo genético.

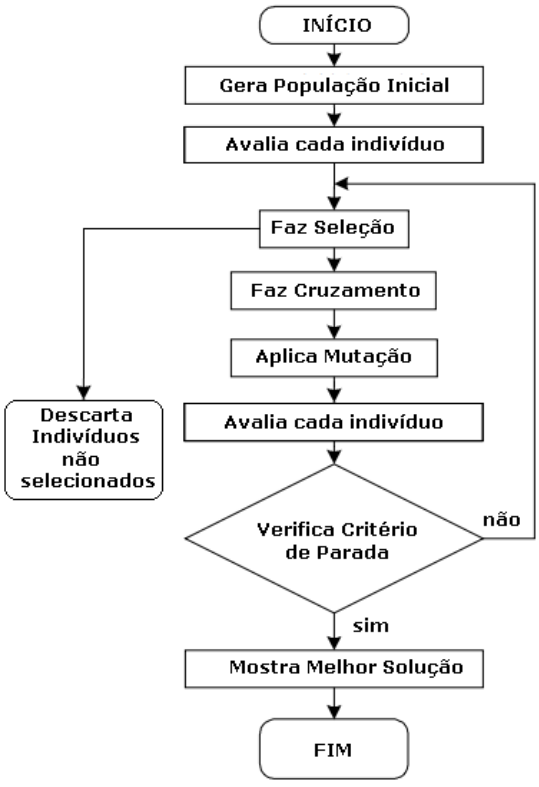


Figura 2 - Fluxograma Geral do Algoritmo Genético

A Figura 3 mostra o diagrama de funcionamento da plataforma proposta, onde o MATLAB e o PSpice se destacam. O programa desenvolvido no ambiente MATLAB, que coordena todo o processo, gera o arquivo *.CIR* que contém as ligações elétricas de cada componente do circuito que será simulado pelo Pspice. O arquivo *.CIR* é o formato de descrição de circuitos compatível com o PSpice. Após a simulação o PSpice gera um arquivo de saída com extensão *.DAT* que contém os dados referentes a tensões e correntes nos nós do circuito simulado.

A partir da leitura dos dados contidos no arquivo *.DAT* as curvas do circuito simulado e do circuito padrão são comparadas e então uma avaliação de aptidão é atribuída ao circuito simulado.

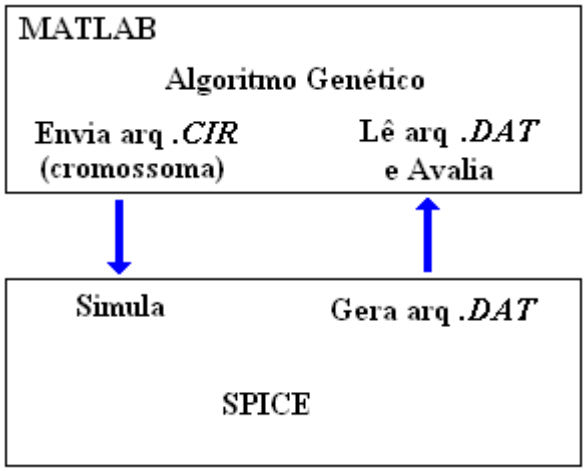


Figura 3 - Diagrama da plataforma proposta com Matlab e PSpice

Outro aspecto importante dos algoritmos genéticos é a substituição da população antiga pela nova geração e o descarte de indivíduos menos aptos. Há várias técnicas ou estratégias que podem ser empregadas. Em geral as técnicas são relacionadas a manutenção dos melhores e descarte dos piores em maior ou menor proporção (Pacheco, 1999).

O algoritmo genético termina em função de ter encontrado o objetivo (aptidão satisfatória de algum indivíduo) ou em função de ter chegado num número de gerações pré-determinado.

1.1.2 Programação Genética

A Programação Genética é uma técnica da área de computação evolucionária que resolve automaticamente problemas sem requerer conhecimento ou especificação estrutural da solução a ser encontrada (Poli et al., 2008). Esta técnica foi concebida por John Koza nos anos 90.

A Programação Genética faz uso de métodos semelhantes aos de evolução dos algoritmos genéticos, porém existe uma diferença fundamental entre ambos: Algoritmos Genéticos modelam palavras binárias (inteiro ou real) para representar as soluções, já a Programação Genética faz uso de árvores de tamanho arbitrário para representar os indivíduos. A abordagem da programação genética permite maior flexibilidade no processo de busca por uma solução, o controle das populações se torna mais crítico por conta do cuidado que se deve ter no tamanho da representação de um cromossoma. A Figura 4 mostra um exemplo do modelo de representação da Programação Genética. Dependendo do ponto de corte uma extensa árvore pode surgir da combinação (cruzamento) de dois cromossomas. Este problema é chamado de “Bloating” e funciona como se fosse uma explosão no tamanho do cromossoma.

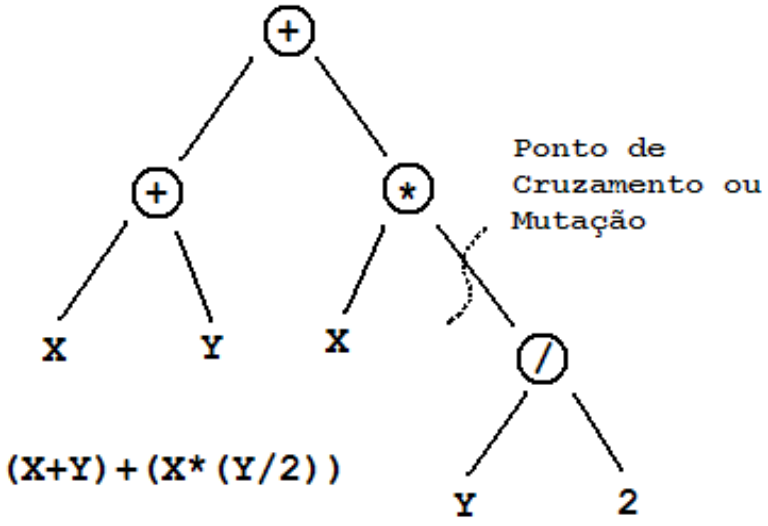


Figura 4 - Modelo de representação cromossomial típico da técnica de Programação Genética

Neste exemplo, o cromossoma representa diretamente a solução buscada, neste caso a equação (x+y)+(x*(y/2)). Por este motivo é freqüente encontrarmos textos que afirmam que a técnica programação genética representa o fenótipo (Poli et al., 2008), características

observáveis do organismo, neste sentido podemos perceber uma outra diferença fundamental entre as técnicas da Programação Genética e dos Algoritmos Genéticos, a primeira atua na busca de um fenótipo mais adequado, enquanto a segunda busca um genótipo que atenda aos requisitos desejados.

1.2 **Sistemas Evolutivos**

1.2.1 Síntese de Circuitos

A partir da possibilidade de síntese de circuitos por mecanismos evolutivos genéticos, podemos começar a pensar em outros temas como auto-reparo, capacidade adaptativa e tudo mais que estes representam: robustez, sobrevida e sustentabilidade (Gong et al., 2009)

Ainda que estes pudessem ser obtidos por outras técnicas, como unidades de substituição redundantes – como é o caso de fontes chaveadas de servidores e unidades de disco rígido chamadas "*hot-swaped*" ou "*hot-stand-by*", o desempenho de sistemas adaptativos é potencialmente mais promissor e mais barato.

Um menor número de componentes eletrônicos ou subsistemas poderiam complementar funcionalmente uma gama maior de circuitos eletrônicos defeituosos ou fora do ponto de operação, assim, por síntese e capacidade de interconexão, qualquer parte do circuito poderia ser consertada – por substituição ou por complementação.

Há inúmeras aplicações onde estas características são extremamente desejáveis, principalmente naquelas em que o acesso aos circuitos é difícil ou inviável, tais como: em equipamentos a grandes profundidades ou em missões espaciais de exploração.

A exploração de recursos naturais em outros planetas, corpos celestes ou grandes profundidades já aparece como uma opção a ser alcançada nas próximas décadas. Isto não parece viável sem que haja uma grande oferta de soluções com as características de robustez, sobrevida e adaptabilidade que podem ser obtidas pelo uso de circuitos evolucionários. Muito do que já desenvolvemos para medir, avaliar, explorar, minerar e garimpar terá que ser refeito com novas capacidades. A redundância de estruturas será apenas uma destas capacidades, e talvez não a melhor, a mais leve ou a de menor custo global (peso x espaço x robustez). Certamente esta terá que ser complementada com características fortemente adaptativas, dentro dos moldes dos sistemas de hardware evolutivo.

Keymeulen, 2010, observa que, tradicionalmente, o foco da NASA para missões espaciais, até então era proteger os circuitos eletrônicos contra ação de altas temperaturas através do uso de isolamento térmico, "*shielding*" contra radiação e redundância dos sistemas críticos, ao custo de mais energia e peso. Neste artigo ele aponta o uso de circuitos eletrônicos adaptativos implementados em FPGAs que podem conferir capacidade de recuperação autônoma da perda de funcionalidade de um circuito analógico reconfigurável (Zebulum et al., 2007) e também capacidade adaptativa a variações de temperatura.

A plataforma apresentada neste trabalho viabiliza a pesquisa sobre a síntese de circuitos eletrônicos por algoritmos genéticos.

1.2.2 Capacidade de adaptação

Dentro do contexto em que usamos os termos: genético e evolucionário, o termo adaptação está intimamente relacionado com evolução, a ponto de poderem ser usados quase com o mesmo sentido (a despeito do rigor que se impõe em seu uso neste trabalho).

Ao observarmos a natureza podemos facilmente relacionar evolução com adaptação e enunciar, sem receio de exagerar, que um não existe sem o outro.

Adaptação é um fundamento no campo de estudo da eletrônica evolucionária, no que diz respeito ao "*hardware*", pode ser definida como "a capacidade de modificar a si mesmo para *manter* ou *melhorar* seu desempenho em função de objetivos internos e/ou em resposta a mudanças no ambiente (interno ou externo)" (Stoica et al., 2007).

O "*hardware*" adaptativo abre importantes possibilidades tais como: auto-reparo e tolerância à falhas, imunidade (capacidade de detectar e consertar falhas).

Podemos enumerar diversas vantagens relacionadas com a capacidade adaptativa do "*hardware*":

- Reaproveitamento do circuito
- Aumento de confiabilidade no produto (robustez)
- Correção de ponto de operação por conta de alteração nas condições externas: temperatura, umidade, campos eletromagnéticos, pressão, radiações e outras.
- Correção de ponto de operação por conta de variações no comportamento de componentes internos (falha ou envelhecimento)

A plataforma proposta neste trabalho visa também fornecer uma ambiente favorável ao estudo de adaptabilidade de circuitos eletrônicos.

1.2.3 Auto-reparo, tolerância a falha e sistemas auto-imunes

Estas são características diretamente ligadas à capacidade adaptativa dos circuitos eletrônicos evolucionários. A síntese de circuitos em tempo real e a busca por uma solução que possa ser interconectada a um circuito defeituoso de forma que o mesmo volte a ter um desempenho aceitável são características extremamente desejáveis.

Vários trabalhos têm sido apresentados nesta área, com ênfase em sistemas digitais implementados em FPGA (Al-Naqi et al., 2011).

Outros trabalhos estão correlacionados com a manutenção do ponto de operação de sistemas que estão submetidos a condições operacionais adversas. Lovay (Lovay et al., 2010) apresenta um sistema que detecta falhas no ganho global de um circuito amplificador de três estágios e implementa correção “on-line’ usando técnicas de algoritmos genéticos.

Da mesma forma, sistemas analógicos críticos como: filtros, fontes chaveadas, moduladores e outros, podem ser monitorados e estarem ligados a uma série de componentes disponibilizados em paralelo (e interligados ao circuito por chaves analógicas) com os diversos circuitos de forma a corrigirem uma ou mais partes dos circuitos por eles supervisionados.

Amaral (Amaral et al., 2007) apresenta um sistema de detecção de falha em circuitos analógicos pela análise de resposta ao impulso. O sistema cria “detectores” de pontos de falha que ficam distribuídos pelo espaço de trabalho, no entorno das condições normais de operação, e indicam a falha assim que um deles é ativado. A Figura 5 mostra as áreas cobertas pelos detectores de falha (círculos com sinal “+” no centro) e as regiões de regime normal de trabalho (círculos na cor cinza). Os pontos marcados com “System OK” e “*Fault*” correspondem respectivamente a detecções de funcionamento normal e falha.

Fernando (Fernando et al., 2010) apresenta uma solução de implementação de algoritmos genéticos em um FPGA, este algoritmo permite a evolução de soluções genéricas pela customização de parâmetros internos e funções externas de avaliação. O sistema foi testado em aplicações de auto-reparo e se mostrou eficaz. Na seção 2.4 estes trabalhos serão mais bem explorados.

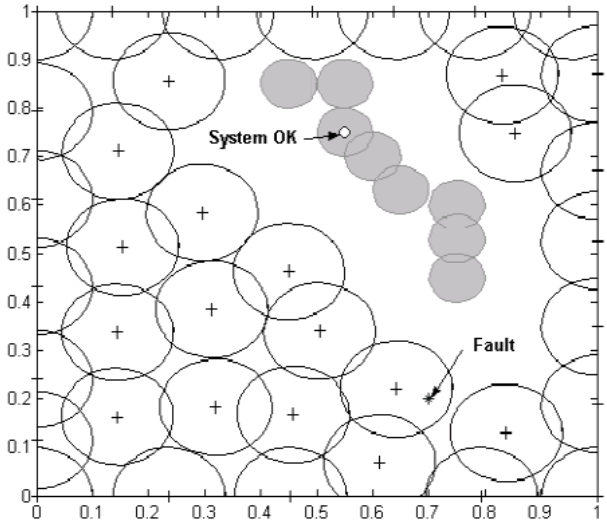


Figura 5 - Distribuição dos Detectores de Falha (Amaral et al., 2007)

1.3 **Evolução de Circuitos**

Para que possa haver evolução de circuitos é necessário que haja plataformas de simulação e avaliação.

A utilização de evolução extrínseca, na qual o circuito é implementado e simulado em "software", permite a exploração mais abrangente de soluções, sem restrição no uso de componentes, porém com tempo maior de simulação e sem implementação "real" do circuito.

Na evolução intrínseca, na qual o circuito é criado em software e testado em "hardware", as propriedades físicas dos componentes eletrônicos são efetivamente testadas e avaliadas com rapidez, porém, na prática, há limitações na quantidade de componentes que se pode usar e também na amplitude do espaço de busca por uma solução.

Adicionalmente, os dois tipos de evolução podem ser combinados de forma a se obter melhor resultado com o que se tem de melhor em cada uma delas. Adrian Stoica (Stoica et al., 2000) introduziu o termo "mixtrinsic" para se referir a um processo híbrido que procurava resolver problemas de validação do equivalente em hardware de circuitos simulados extrinsecamente e Tawdorss (Tawdross et al., 2007) também adaptou mecanismos híbridos para validar diferentes objetivos durante o processo evolucionário.

1.3.1 Evolução Extrínseca

Este tipo de evolução é feita com a utilização de simuladores de circuitos. Neste trabalho usamos o PSpice.

A plataforma extrínseca permite explorar com mais liberdade e amplitude o espaço de busca a fim de encontrar topologias de circuito que normalmente não seriam tentadas numa abordagem clássica. Em princípio não há limitação para os tipos e quantidades de componentes que se pode usar na população.

O tempo gasto neste tipo de evolução é tipicamente muito maior que seu equivalente intrínseco, e o desempenho depende fortemente da máquina e do sistema operacional no qual o algoritmo genético e o simulador estão sendo executados (ainda que se possa implementar vários tipos de paralelismo para ajudar no processo).

O diagrama da Figura 6 mostra o esquema geral de um sistema de evolução extrínseca com a presença de um software de algoritmo genético e um simulador de circuitos eletrônicos, Neste caso, tanto o AG como o simulador de circuitos são "*software*" e estão sendo executados em um computador ("PC").

A técnica dos Algoritmos Genéticos (AG) foi escolhida em função de ter sida a mesma técnica utilizada na plataforma de referência – PAMA-NG.

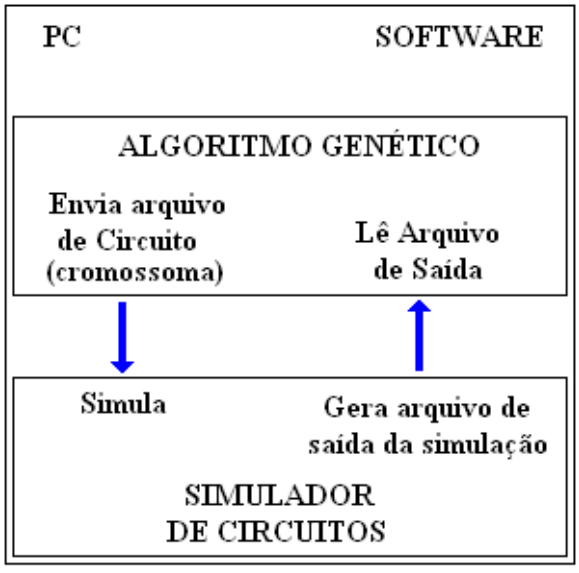


Figura 6 - Diagrama básico de sistema de evolução extrínseca

A simulação depende dos modelos de componentes, e respectivos parâmetros internos, bem como do "setup" do simulador, que pode ser ajustado para diferentes condições de operação, ou seja, mesmo que o processo evolutivo tenha encontrado uma boa solução, nada

garante que a implementação apresente os mesmos resultados. No entanto, o simulador pode ser configurado para fazer cálculos em regimes de trabalho diferenciados (temperatura, ruído e outros) e gerar candidatos (convencionais ou não-convencionais) em condições próximas a desejada, quando então outras técnicas poderiam ser aplicadas para definir o desenho da solução.

As principais vantagens da evolução extrínseca de circuitos estão relacionadas com a grande flexibilidade, a possibilidade de avaliar novas idéias de modo mais simples e a liberdade para experimentação de novas representações e avaliações (Amaral, 2003).

1.3.2 Evolução Intrínseca

Na evolução intrínseca o circuito que está sendo avaliado é o circuito real, ainda que não necessariamente nas condições de utilização (condições operacionais). Nos casos em que a plataforma evolucionária faz parte do próprio produto e este tenha características adaptativas, as condições de evolução serão as melhores e mais próximas do ideal, pois o circuito estará sendo adaptado nas condições vigentes de operação (temperatura, pressão, radiação e outras) – O item 1.4.1 mostra uma plataforma com estas características.

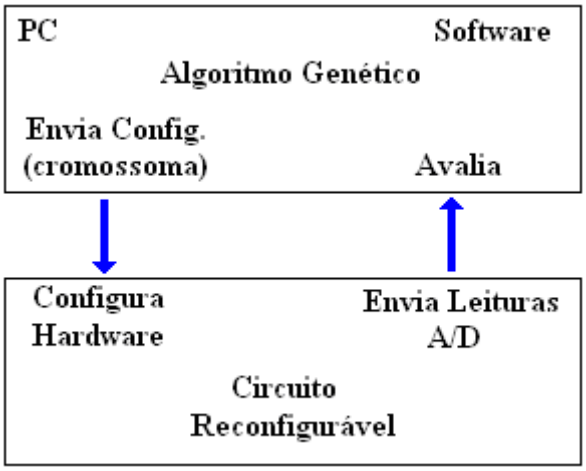


Figura 7 - Diagrama básico de evolução intrínseca

A avaliação do circuito é feita com base em sinais analógicos reais. O tempo de avaliação de um circuito é limitado somente pelo tempo necessário para fornecer ao circuito o conjunto de entradas segundo sua especificação (Santini, 2001 & Amaral, 2003).

Ambientes de Hardware Evolucionário para evolução intrínseca consistem de duas partes (Figura 7): um circuito reconfigurável (hardware programável), como uma FPGA

(“*Field Programmable Gate Array*”) ou FPAA (“*Field Programmable Analog Array*”), e alguma técnica de computação evolucionária, como os Algoritmos Genéticos. O hardware programável é reconfigurado pelos bits de configuração determinados pelo algoritmo evolucionário implementado em software.

Atualmente há Algoritmos Genéticos implementadas em FPGAs que podem ser integrados a um hardware que tem características adaptativas: digitais e/ou analógicas (Fernando et al., 2010). Ricardo Zebulum (Zebulum et al., 2007) apresentou uma nova classe de FPAAs, chamada SRAA (*Self-Reconfigurable Analog Array*), com muito mais recursos que antigas FPAAs, com possibilidade de se integrar a FPGA e capacidade de reconfiguração on-line sem perda da funcionalidade do sistema ao qual se integra.

Há ainda uma forte preocupação com desenvolvimento de sistemas adaptativos – e respectivos componentes eletrônicos – em relação a aspectos de temperatura e incidência de radiações, o que faz com que as plataformas FPGA e FPAA já estejam integradas em sistemas cada vez mais compactos e robustos, próprios para trabalharem em ambientes inóspitos (Keymeulen et al., 2007).

Em julho de 2010 a Xilinx (fabricante de componentes FPGA) incorporou em algumas linhas de FPGA a capacidade de reconfiguração parcial “*on-line*” de sua lógica interna – partes da programação interna do FPGA podem ser completamente atualizadas sem que aquela parte perca a funcionalidade. Espera-se que esta característica tenha forte influência nos desenvolvimentos na área de circuitos adaptativos e plataformas de evolução intrínseca (Xilinx, 2012).

1.3.3 Evolução Híbrida

A evolução híbrida é na realidade uma estratégia de uso da evolução intrínseca e extrínseca, buscando usar o melhor de cada um dos dois tipos de evolução.

Eventualmente pode ser necessário explorar com mais liberdade ou amplitude o espaço de busca a fim de encontrar topologias de circuito que servirão de ponto de partida para uma investigação mais detida de uma solução. Neste caso o espaço de busca poderia ser explorado usando-se a evolução extrínseca em uma ou mais rodadas, partindo de diferentes conjuntos de componentes de circuito. Desta forma uma gama de possíveis soluções poderia ser usada, por exemplo, como populações iniciais em uma plataforma de evolução intrínseca.

Outra possibilidade de uso de evolução híbrida é checar de tempos em tempos, durante o processo evolutivo, se o modelo físico (real) dos melhores candidatos de uma população está atendendo a determinadas características dentro da situação real de implementação. Assim o processo evolutivo passa a avaliar também as condições reais de operação do circuito. Esta estratégia permite potencializar o processo de avaliação e seleção.

Adrian Stoica introduziu o termo "*mixtrinsic*" para se referir a um processo híbrido que procurava resolver problemas de portabilidade – transladar o que foi evoluído em "*software*" para o correspondente "*hardware*" (Stoica et al., 2000).

Tawdross, se referindo ao trabalho de Stoica, fez uso dos dois tipos de evolução ao mesmo tempo: cada indivíduo é avaliado nas duas plataformas; em cada uma, diferentes objetivos são avaliados, o resultado final da avaliação é computado pelo somatório dos erros de cada medida e seu peso relativo, desta forma uma única "nota" é dada para o indivíduo (Tawdross et al., 2007).

A arquitetura da plataforma de desenvolvimento proposta neste trabalho foi concebida de forma a permitir evolução extrínseca, intrínseca ou a híbrida, sendo que somente a plataforma extrínseca foi implementada. Mais adiante, no capítulo 3, é descrita a plataforma de desenvolvimento que mostra como estas e outras funcionalidades podem ser acrescentadas.

1.4 **Plataformas Reconfiguráveis**

A utilização de plataformas reconfiguráveis abre inúmeras possibilidades de implementação de sistemas adaptativos e tem impacto no projeto de soluções especialmente interessantes em sistemas que precisam se manter funcionais em condições diversas e por longos tempos sem possibilidade de manutenção. Este é o caso de sistemas embarcados usados na indústria do petróleo em águas profundas ou em missões espaciais.

Os sistemas adaptativos podem ser concebidos como parte integrante de produtos comerciais, conferindo a estes produtos maior longevidade, usabilidade, facilidade de manutenção e atualização.

Analisando a literatura, pudemos identificar algumas plataformas e sistemas reconfiguráveis. Amaral e Santini (Amaral, 2003 & Santini 2001) descreveram diversas plataformas reconfiguráveis com possibilidades de evolução intrínseca de sistemas analógicos, dentre estas destacamos o FPTA (Stoica et al. 2000), a PAMA (Santini, 2001), e a PAMA-NG (Amaral, 2002).

O FPTA (“*Field Programmable Transistor Array*”), apresentado na Figura 8, é um componente que implementa uma matriz de 8 transistores tipo FET ligados a 24 chaves analógicas e permite a evolução de vários tipos de circuitos digitais e analógicos. Os estados das chaves podem ser selecionadas externamente (ligada / desligada) e os terminais dos transistores podem ser ligados a circuitos externos. Em uma última versão, o FPTA2 foi implementado com 64 células analógicas. Atualmente a série FPTA parece ter sido colocada de lado em função de necessidades mais objetivas e de foco maior em sistemas mais robustos desenvolvidos pelo mesmo grupo de pesquisadores. Ver trabalho de Adrian Stoica (Stoica et al., 2007).

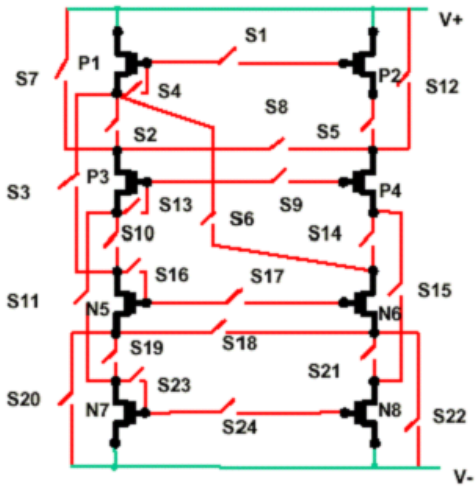


Figura 8 - Diagrama de uma célula da FPTA com 8 transistores e 24 chaves (Stoica et al., 2000)

A PAMA-NG é uma plataforma de evolução intrínseca constituída de um sistema multiplexador que suporta a interligação mútua de até 32 pinos de componentes com fontes externas de sinais e “*buffers*”. Uma placa de conversão A/D e D/A é acoplada ao sistema de forma que se possa gerar e ler múltiplos sinais.

Nesta plataforma, o algoritmo genético gera N cromossomas que são testados um a um na plataforma intrínseca (“*hardware*”). Injeta-se a curva de entrada e a curva de saída é lida e devolvida para o algoritmo genético que a compara com a curva padrão (esperada). Esta comparação gera um resultado que é chamado “avaliação de aptidão”, que indica em que grau o cromossoma testado representa a solução desejada. Este valor será utilizado pelo algoritmo genético no processo de seleção da próxima rodada, exceto no caso em que a solução tenha sido encontrada na rodada atual.

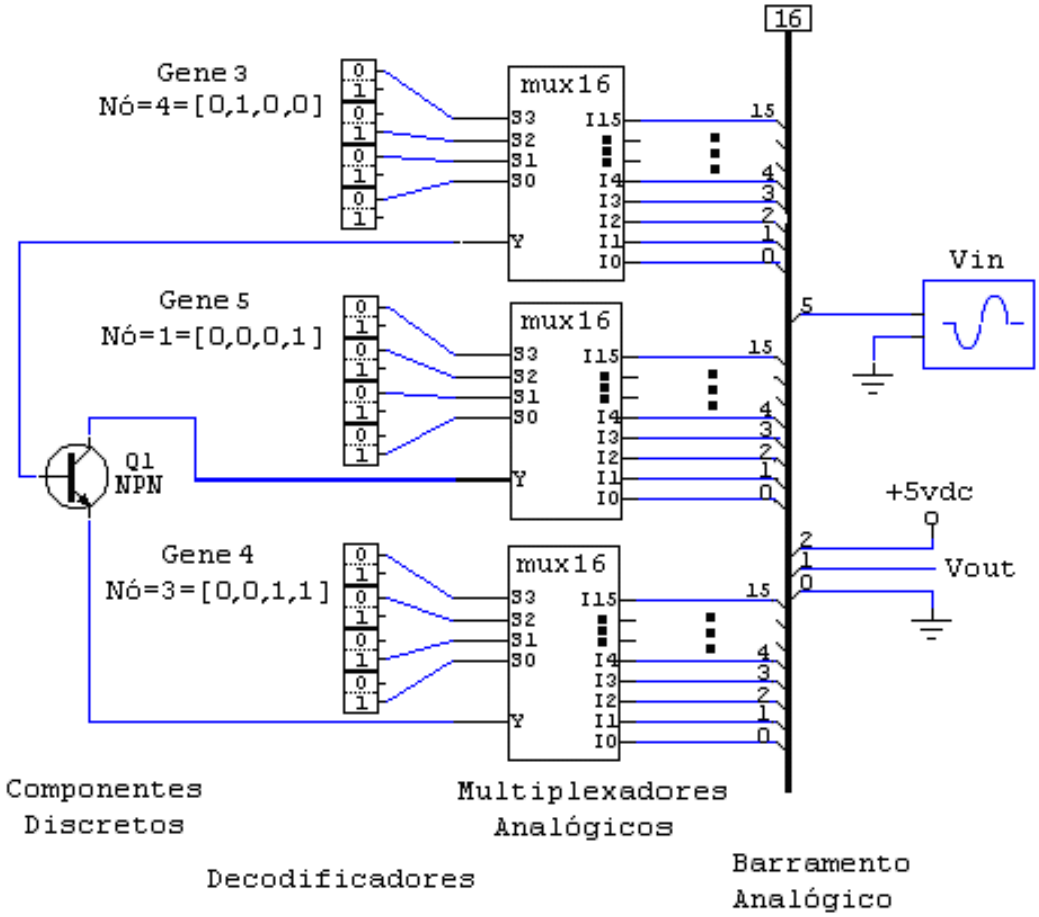


Figura 9 - Diagrama básico da PAMA-NG (Amaral, 2002)

A Figura 9 mostra a ligação de um transistor na plataforma. Cada terminal é um gene deste cromossoma. A cada gene é atribuído um número que representa o nó ao qual o terminal estará ligado no circuito a ser avaliado. Desta forma, o transistor Q1 da Figura 9 apresenta as ligações de nós conforme a representação do cromossoma A da Figura 10.

A Figura 10 mostra um cromossoma e duas possíveis configurações (cromossomas A e B). Os respectivos circuitos também são mostradas na mesma figura.

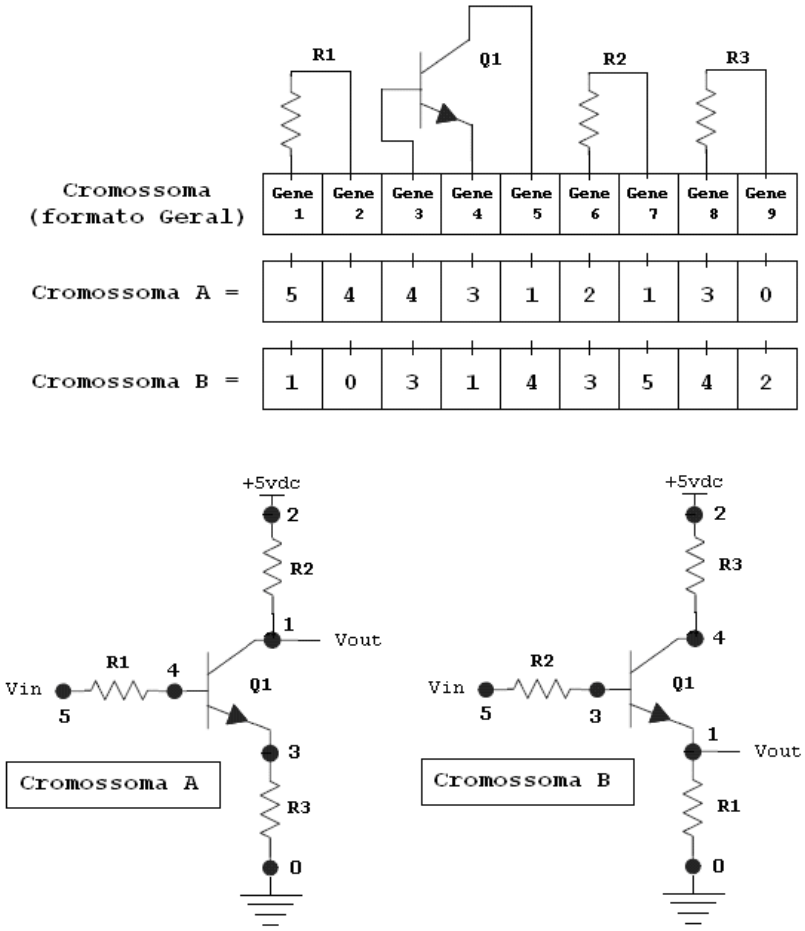


Figura 10 - Exemplo de representação genética de circuito eletrônico

Tendências Atuais

Os trabalhos mencionados acima (FPTA e PAMA-NG) representam um modelo de estudo e solução de problemas que impõe uma grande capacidade de chaveamento aliada a componentes que tendem a ser de uso genérico em diversas situações. Esta abordagem é realmente útil na pesquisa e investigação, e em fases mais preliminares no desenvolvimento de produtos, de conceitos e de "*know-how*", porém pouco aderentes a requisitos em aplicações comercias ou em larga escala.

Observamos também que algumas empresas que fabricavam FPAA ("*Field Programmable Analog Array*"), de uso geral, que pretendiam servir a uma ampla gama de possibilidades, já mostram seus produtos como obsoletos ou só os fabricam sob encomenda em quantidades acima de 10.000 peças (Como é o caso da Xilinx, da Lattice e outras).

O que vemos na literatura nos dias de hoje é uma tendência a se dividir o uso de hardware adaptativo entre: um FPGA com o algoritmo genético, um FPAA (ou equivalente)

com foco em aplicação específica (mas com capacidade aumentada de configuração) e uma terceira classe que seriam os sistemas auto-imunes implementados em FPGA, nos quais células funcionais podem assumir a funcionalidade de outras células, conferindo ao sistema uma capacidade extra de auto-reparo.

Vemos também a oferta de uma classe de FPGA com capacidade de reconfiguração sem perda funcional durante o processo de recarga, ou seja, todo o código de configuração pode ser trocado (em partes) sem que em nenhum momento seja necessário parar o processo funcional da parte em alteração (apenas um "*reset*" no final do processo) (Xilinx, 2012).

Estes novos componentes, aliados a uma abordagem mais comercial e integrada dos sistemas adaptativos baseados em algoritmos evolutivos embutidos confirmam a consolidação de um ramo que está em expansão – sistemas evolutivos/adaptativos comerciais.

A seguir apresentamos, resumidamente, alguns destes sistemas ou produtos que corroboram esta tendência.

1.4.1 Plataformas Reconfiguráveis Integradas

Apresentaremos a seguir, de forma sucinta, duas plataformas reconfiguráveis desenvolvidas recentemente. Ambas, dentre outras tantas não apresentadas neste trabalho, mostram uma clara tendência a buscar soluções em aplicações não-acadêmicas, mais voltadas à aplicação integrada em sistemas mais complexos – industriais, militares, aeronáuticos e espaciais.

Plataforma SoC (Otero et al. & Sekanina)

Esta plataforma oferece uma solução modular com vários FPGA em módulos. Um dos módulos contém um "*IP-Core*" de microprocessador que roda um algoritmo evolutivo e coordena as ações de outras partes do sistema; módulos adicionais de gerenciamento de memória, reconfiguração de hardware, em sua maioria implementados em FPGA, formam as demais partes da plataforma. Este sistema tem viabilidade de implementação em um único FPGA (Otero et al., 2011).

A Figura 11 mostra o diagrama da plataforma com seus principais componentes.

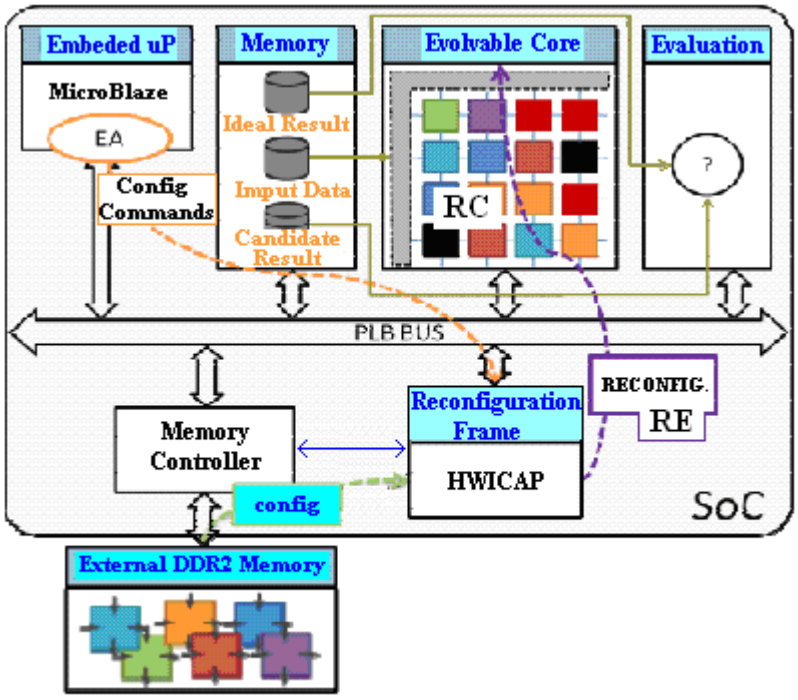


Figura 11 - Arquitetura do Sistema de Harware Evolutivo (Otero et al.,2011)

A arquitetura do sistema é proposta para ser construída como um "System-on-Chip' SoC. Os blocos funcionais que se destacam nesta arquitetura são:

- Microprocessador Embutido: MicroBlaze IP-Core
- Reconfiguration Engine (RE)
- Reconfiguration Core (RC)

O hardware é implementado no módulo RC que possui um "*array*" bi-dimensional de células ("cluster") que podem conter várias funções (Tabela 1). Cada "*cluster*" se comunica com a célula adjacente e com o barramento interno do módulo.

A configuração funcional do módulo RC é feita no módulo RE que recebe os comandos de configuração diretamente do processador. Esta arquitetura faz uso da capacidade de reconfiguração dinâmica do FPGA que permite a carga de uma nova configuração sem necessidade de parada do sistema.

O módulo RE faz uso de uma memória externa para executar suas funções de configuração do módulo RC.

Código	Função	Descrição
0	x + y	Adição
1	x << 1	Deslocamento a Esquerda
2	x $+_s$ y	Adição com Saturação
3	(x + y) >> 1	Media
4	255	Constante
5	x >> 1	Deslocamento a Direita
6	x	Identidade
7	max(x,y)	Máximo
8	min(x,y)	Mínimo
9	x $-_s$ y	Subtração com Saturação em 0

Tabela 1 - Tabela de Funções do módulo RC (Otero et al., 2011)

O módulo do microprocessador possui duas outras estruturas anexas: um módulo de memória e outra de avaliação. Estes módulos juntos são responsáveis pela execução do algoritmo evolutivo.

Vale observar os seguintes aspectos desta plataforma:

- Possui previsão para um módulo separado de avaliação de indivíduos para uso no algoritmo genético, esta função também está implementada na outra plataforma que apresentamos mais adiante, e sua arquitetura permite que a avaliação seja interna ou externa.
- As funções de cada cluster podem ser alteradas, mas tipicamente são funções com “statements” compatíveis com programação genética conforme abordamos no item 1.1.2.

Estas observações foram recorrentes na pesquisa bibliográfica e nos revelaram uma tendência convergente que pode ser evidenciada nas plataformas apresentadas neste capítulo.

Plataforma JPL/NASA (Stoica et al., 2007)

Esta plataforma vem sendo desenvolvida no Laboratório de Propulsão a Jato da NASA (JPL) pela equipe coordenada pelo pesquisador Adrian Stoica. Esta plataforma possui basicamente dois “*arrays*” programáveis de grande capacidade: um analógico (SRAA) e outro digital (FPGA). Juntos, formam uma plataforma completa e podem ser integrados em diversas aplicações como veremos mais adiante.

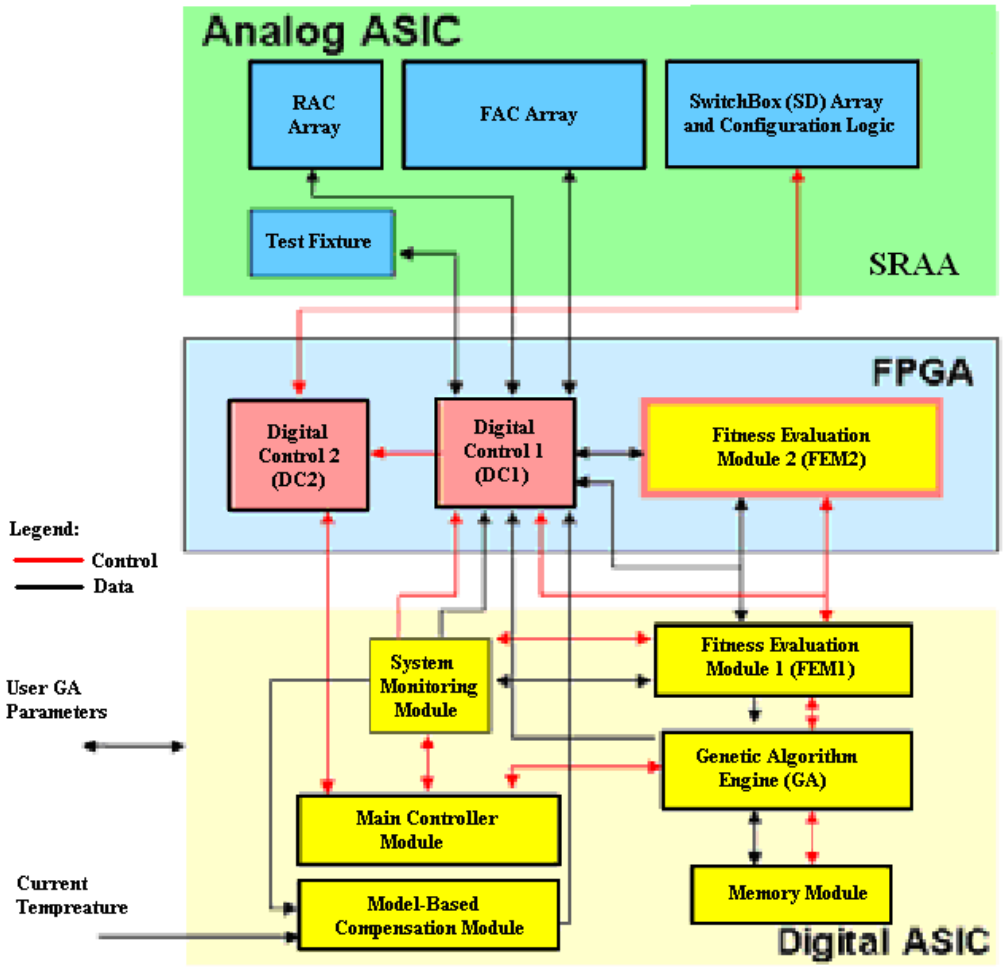


Figura 12 - Diagrama da plataforma JPL/NASA (Stoica et al., 2007)

O diagrama da plataforma é apresentado na Figura 12. Neste diagrama fica evidente o uso de dois componentes FPGA, um para o GA e outro para controle da SRAA.

O FPGA que faz o controle de configuração e carga da SRAA está previsto para ser integrado a mesma, por este motivo, vamos seguir analisando a plataforma como se fosse dividida em apenas duas partes principais.

Dois trabalhos separados integram as funcionalidades desta plataforma: o primeiro trabalho é relativo a SRAA ("*Self-Reconfigurable Analog Array*") (Zebulum et al., 2007), o segundo é relativo a implementação em FPGA de um algoritmo genético parametrizado pelo usuário.

A Figura 13 mostra a arquitetura da SRAA (SRAA). Vale observar que a SRAA aparece contendo dois sub-módulos (SRAA e FPGA), isto porque estes módulos serão integrados em versões posteriores. O módulo Digital ASIC se refere a um módulo extra que permite acoplamento do algoritmo evolucionário.

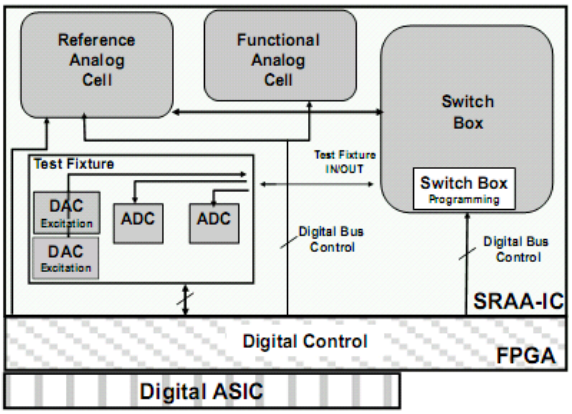
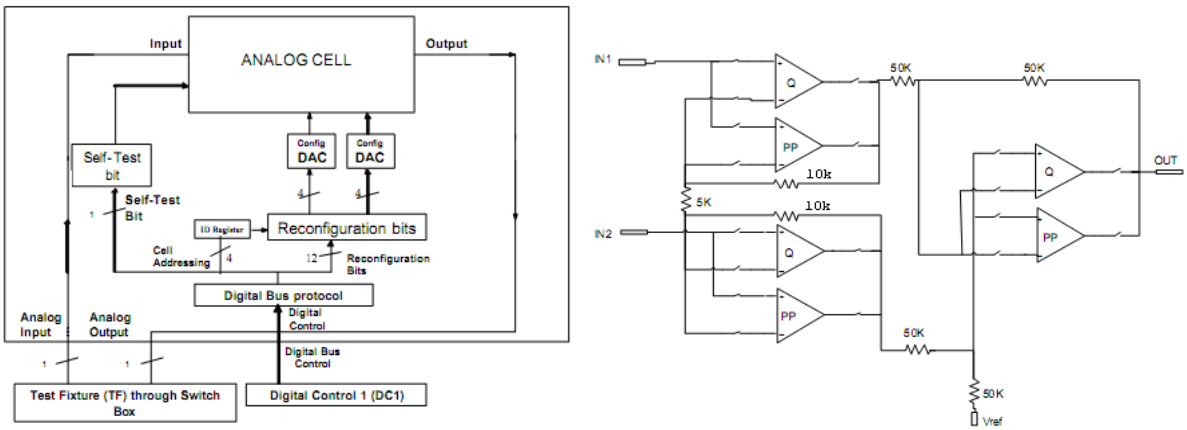


Figura 13 - Diagrama Geral da SRAA (Stoica et al., 2007)

A Figura 14 mostra a arquitetura de um dos módulos da SRAA. Estes módulos possuem conversores A/D e D/A e a matriz analógica configurável.



(a) **(b)**

Figura 14 - Esquema Geral da Célula Analógica de referência das SRAA (a) e um Circuito Analógico sintetizado (b) (Stoica et al., 2007)

O segundo componente da plataforma JPL/NASA é o Algoritmo Genético parametrizado pelo usuário e implementado em FPGA. Uma interessante característica deste componente é que ele permite a utilização de uma função de avaliação externa, como mostrado a Figura 15.

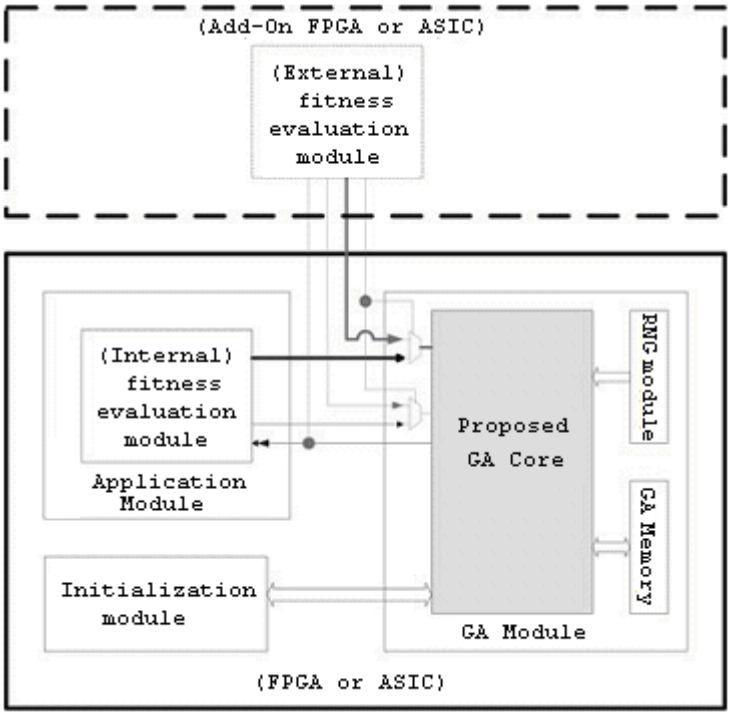


Figura 15 - Diagrama do FPGA com Algoritmo Genético (Stoica et al., 2007)

Este sistema permite também o uso de parâmetros configuráveis do algoritmo genético. A Tabela 2 mostra os parâmetros pré-setados e os ajustáveis que podem ser escolhidos pelo usuário para parametrizar o tamanho da população, número de genes, nível de corte das operações de cruzamento e mutação.

Mode		Pop. size	No. of Gens.	Thresholds	
				Xover	Mutn.
User	00	< 256	$< 2^{32}$	0–15	0–15
Preset	01	32	512	12	1
	10	64	1024	13	2
	11	128	4096	14	3

Tabela 2 - Configuração do Algoritmo Genético (JPL/NASA) (Stoica et al., 2007)

Outras importantes características deste módulo são:

- Foi modelado para ser “IP-Core” que pode ser usado em conjunto com outros sub-sistemas
- Aceita o uso de função de avaliação externa ou até oito funções de avaliação pré-configuradas internamente sem a necessidade de reprojetar o sistema.

- Permite fornecimento de uma semente (“*seed*”) para o gerador aleatório de números, de forma a conferir diferentes condições de execução para o mesmo conjunto de parâmetros.
- O Protocolo de comunicação para configuração e resultados é do tipo “*two-way*” e bem simples de ser implementado.

1.4.2 Sistemas auto-imunes baseados em FPGA

O sistema apresentado na Figura 16 mapeia cinco funções básicas (A,B,C,D,E) em cada célula funcional do FPGA. Um conjunto funcional corresponde a 9 células adjacentes sendo que apenas 5 estão ativas a cada instante (Figura 16a) – cada célula executa uma das cinco funções pré-definidas. Em caso de falha, uma das células do grupo assume a função da faltosa. Caso um grupo funcional inteiro perca a capacidade adaptativa, outro grupo funcional adjacente pode assumir integralmente sua funcionalidade (Figura 16b).

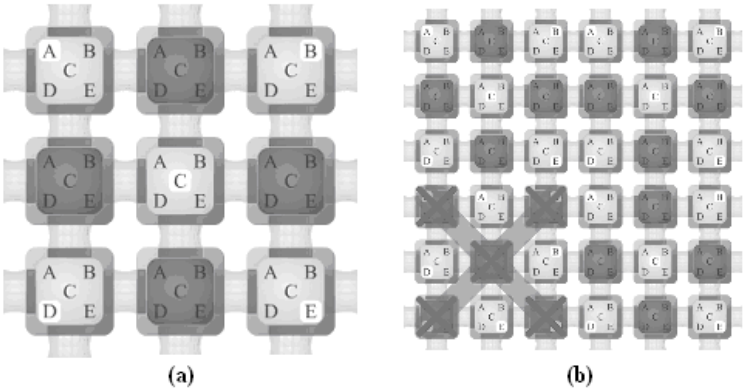
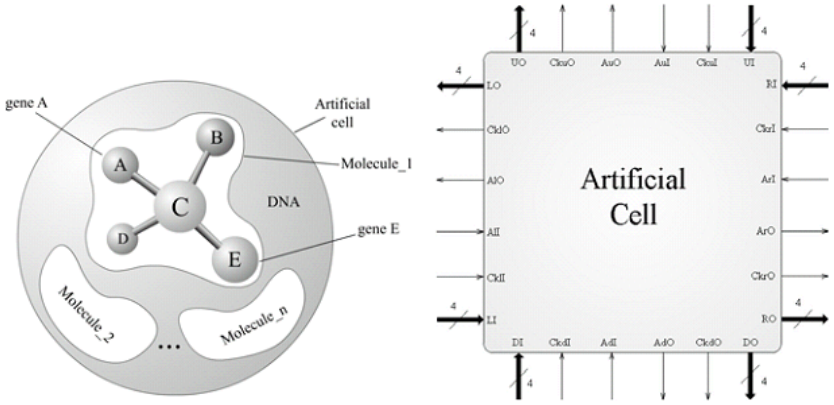


Figura 16 - Sistema Auto-Imune implementado em FPGA (Szász, 2010)

Neste trabalho o autor (Szász, 2010) promove a reprogramação das células por meio de um microprocessador externo cujas funções básicas são: carga de configuração, análise de falhas, ativação de funcionalidade em uma célula e ativação e desativação de grupos funcionais.

O trabalho foi desenvolvido com o uso do FPGA Xilinx XC3S100-E e sua estrutura funcional é baseada num modelo de sistema imunológico de organismos biológicos no qual cada célula contém toda a informação genética para produzir um novo indivíduo com a mesma funcionalidade.



(a) (b)

Figura 17 - Estrutura funcional genética (a) e sua respectiva implementação no FPGA (b) (Szász, 2010)

A Figura 17a apresenta o modelo funcional genético (equivalente genético) no qual a célula artificial proposta se baseia. A Figura 17b apresenta o modelo eletrônico da célula artificial com seus barramentos de conexão com células adjacentes e linhas de controle e sincronização.

A capacidade de substituição funcional e a organização celular-genética deste modelo, aliada à possibilidade de reprogramação on-line dos dispositivos FPGA, compõem uma infra-estrutura de plataforma reconfigurável para sistemas adaptativos onde cada célula pode ser individualmente codificada para ser uma cópia de outra célula (conferindo imunidade) ou uma célula com comportamento diferenciado das demais (conferindo adaptabilidade ao sistema).

Um outro modelo similar de estrutura auto-imune é representado pelo sistema chamado “Reconfiguration POEtic Tissue” (Tecido POEtic reconfigurável) financiado pela Comunidade Européia (Greenwood, W. et al., 2007). Este sistema se baseia numa arquitetura de três camadas sucessivas inspiradas na biologia:

- Filogenese (Philogenesis - P) diz respeito à carga genética em função da história de evolução da espécie – nesta camada do modelo está contido o genoma do tecido POEtic, arbitrariamente grande e contém a memória evolutiva do tecido.
- Ontogenese (Onthogenetic - O) diz respeito à formação do indivíduo, pela seleção das partes do genoma da camada filogênica que vão construir o indivíduo, determinado sua configuração e seu crescimento.
- Epigenese (epigenesis – E) diz respeito ao desenvolvimento do indivíduo através de processos de aprendizagem (sistema nervoso ou sistema imunológico) influenciado tanto pelo código genético quanto pelo ambiente.

Esta modelagem permite flexibilidade na escolha da camada que será alterada para que determinado problema seja resolvido. Na prática este modelo pode ser implementado em dispositivos similares aos FPGAs, com algumas alterações em sua estrutura de roteamento entre células (Barker, W. et al., 2007).

A Figura 18 mostra o diagrama conceitual do modelo POEtic para construção de sistemas dinamicamente configuráveis e auto-imunes.

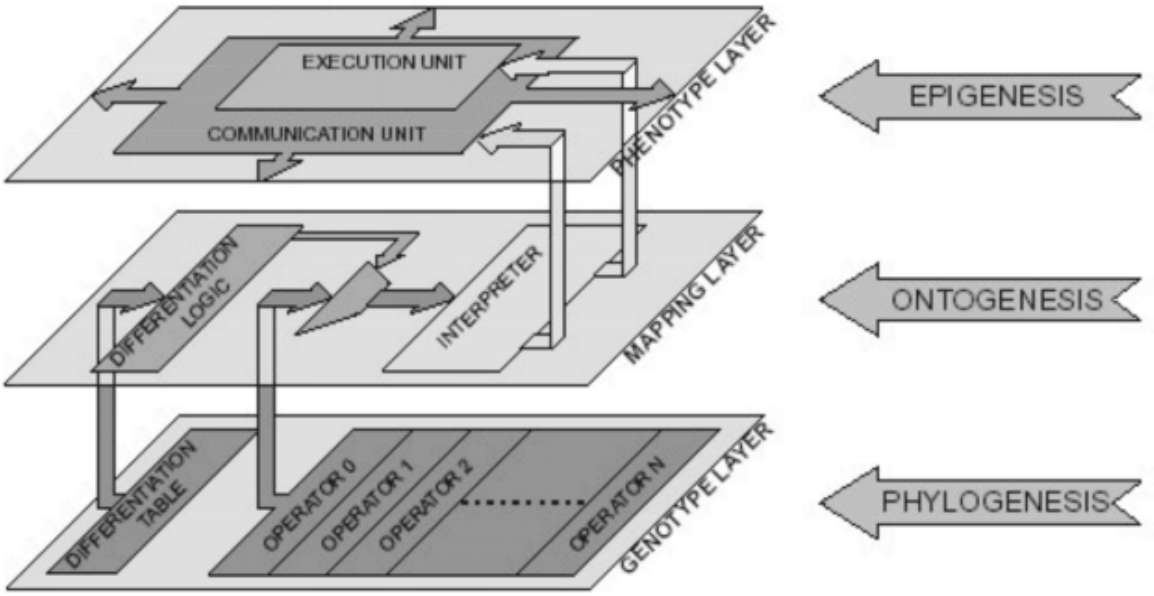


Figura 18 - Sistema de três camadas do modelo POEtic (Barker, W. et al., 2007).

Cada camada pode ser convenientemente mapeada para fornecer estruturas ou processos úteis na busca por um indivíduo apto para solução do problema no qual o sistema está envolvido.

2 IMPLEMENTAÇÃO DA PLATAFORMA

2.1 Arquitetura Geral da Plataforma de Desenvolvimento

A Figura 19 apresenta a arquitetura da plataforma de desenvolvimento com previsão para contemplar as técnicas de evolução extrínseca (com uso de simuladores como o PSpice) e intrínseca (com utilização de hardware reconfigurável) e híbrida. Este trabalho trata somente da implementação extrínseca.

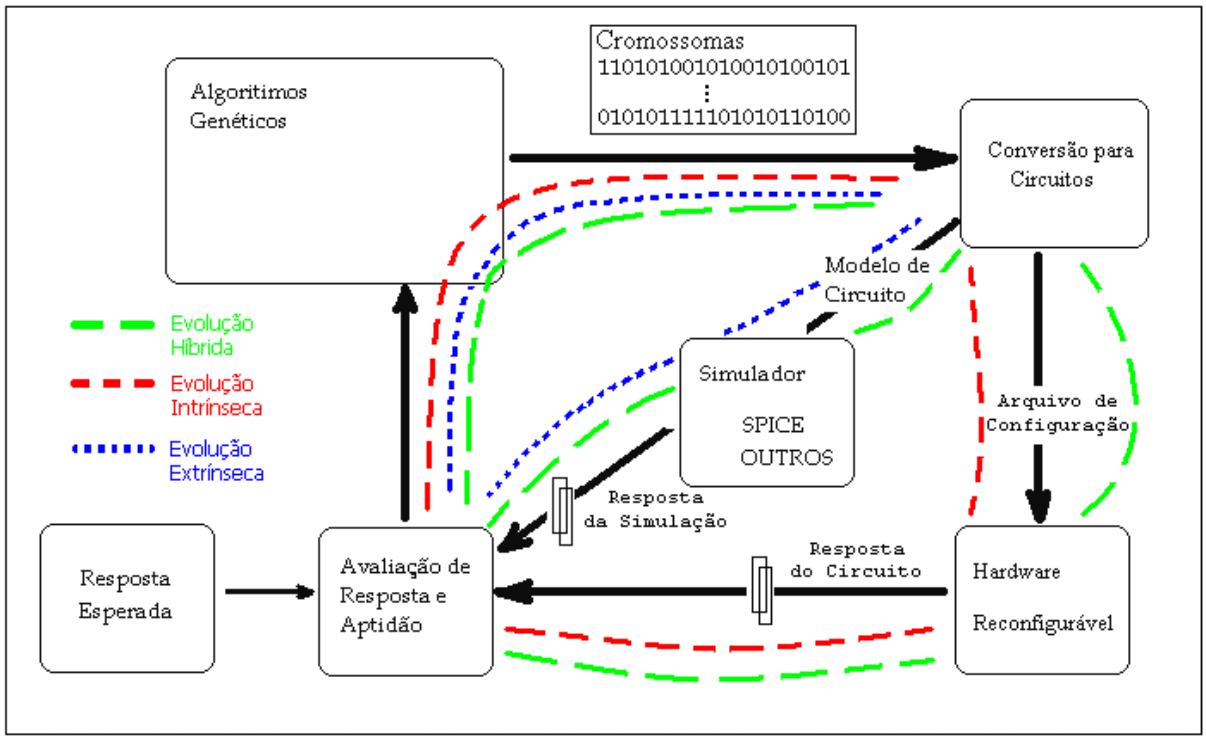


Figura 19 - Arquitetura da Plataforma de Desenvolvimento
Este trabalho apresenta a implementação da Evolução Extrínseca da arquitetura acima

A plataforma foi concebida dentro de um modelo de interoperabilidade e funcionalidades cuja estrutura permite sua evolução de acordo com novas necessidades ou novas tecnologias sem perder sua identidade com versões anteriores de seus módulos. Esta filosofia de concepção de sistemas tem a vantagem de permitir maior aproveitamento de módulos, curva de aprendizado mais rápida e consistente, maior possibilidade de difusão do conhecimento entre usuários, rastreabilidade em diferentes versões e arranjos específicos de componentes sistêmicos.

Na versão atual da plataforma extrínseca de desenvolvimento de circuitos adaptativos, um programa desenvolvido para o ambiente do MATLAB é o responsável pela coordenação de todas as atividades de simulação e avaliação: algoritmo genético, conversão do

cromossoma em arquivo "*.cir*" (PSpice) – incluindo a correção dos cromossomas para circuitos minimamente simuláveis e avaliação da aptidão.

As contribuições deste trabalho são evidenciadas na facilidade de uso da plataforma, na sua portabilidade para outros sistemas operacionais ou versões de seus componentes (principalmente MATLAB e PSpice) e na sua capacidade de trabalhar com os arquivos de dados do PSpice. A plataforma pode ser portada para computadores mais rápidos e também pode ser usada em processos paralelos de evolução, o que fará com que seu desempenho venha a ultrapassar o de plataformas intrínsecas, que são limitados pelos tempos de conversão A/D, D/A, bits de configuração e tempo de setup do hardware, aplicação e estabilização dos sinais de entrada.

A plataforma contempla o uso de três componentes principais: o software de simulação de circuitos PSpice, o MATLAB e o pacote de algoritmos genéticos (GAOT) – que é executado no MATLAB. O modelo de cromossoma utilizado é compatível com a plataforma intrínseca PAMA-NG apresentada por Amaral, 2002.

A Figura 20 apresenta o diagrama funcional, onde se vê mais detalhadamente as funções executadas por cada bloco.

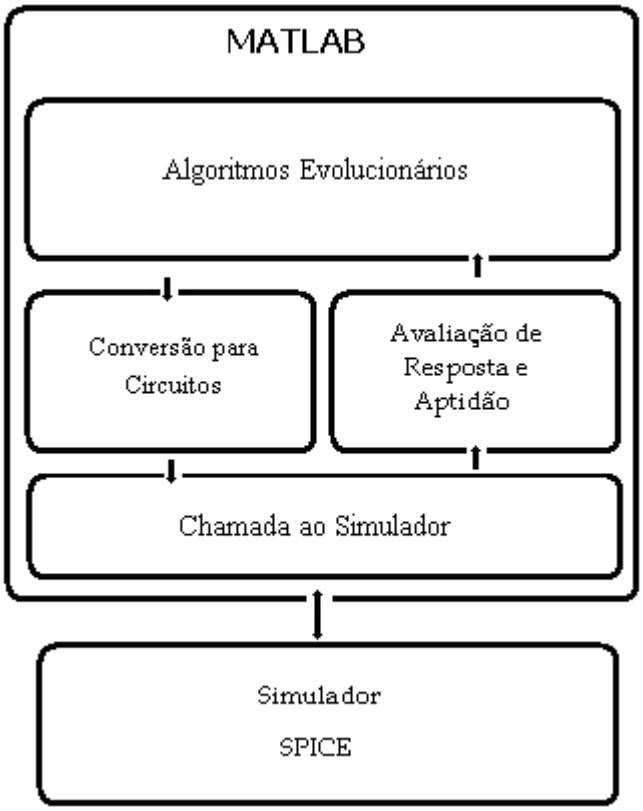


Figura 20 - Diagrama funcional da plataforma extrínseca

Os blocos funcionais apresentados reforçam a característica modular da arquitetura da plataforma uma vez que as funcionalidades de cada bloco podem ser implementadas com técnicas ou componentes diferentes.

2.1.1 Bloco Funcional do MATLAB

O programa desenvolvido para o ambiente do MATLAB pode conter e/ou gerenciar várias sub-funções da plataforma em suas diversas modalidades de modelo de evolução (extrínseco, intrínseco ou híbrido). Nesta versão implementamos apenas a evolução extrínseca (software), portanto apenas as funcionalidades pertinentes de cada bloco foi efetivamente codificada na plataforma.

O MATLAB versão 6.5 foi usado nesta implementação da plataforma. Esta versão foi a escolhida em função de estar disponível nos laboratório s de Graduação e Pós-Graduação da instituição.

O bloco de Algoritmos Evolucionários foi implementado com o "*toolbox*" de algoritmos genéticos (GAOT), porém outros tipos de algoritmos evolucionários podem ser implementados em substituição sem que seja necessário alterar profundamente outros blocos.

O bloco de Conversão para circuitos, nesta implementação que trata de evolução extrínseca, constrói um arquivo para PSpice, porém poderia fazer uso de uma chamada externa para montar um arquivo de configuração de um componente GAL ou PAL, ou outros tipos de arquivo de saída para um "*hardware*" configurável.

O bloco de Avaliação de Resposta ou Aptidão, nesta versão extrínseca, corresponde a uma função chamada de dentro do GAOT, com nome e funcionalidades configuráveis pelo usuário; o que permite fácil integração com outros tipos de algoritmos ou novas versões do algoritmo atual. Em outras versões este bloco poderia ser executado fora do MATLAB, e faria o envio e recebimento dos dados a serem avaliados para uso na geração seguinte ou verificação de critério de parada.

O bloco de chamada ao simulador e configuração de hardware externo, nesta versão extrínseca, faz a chamada ao simulador PSpice e devolve o resultado da simulação ao programa principal, porém o mesmo pode ser alterado para interfacear com outras estruturas de hardware ou software usando funções nativas do MATLAB ou chamadas a funções externas escritas em outras linguagens ou sistemas.

2.1.2 Bloco Funcional do Simulador (Extrínseco)

Este módulo corresponde às funções de simulação extrínseca. Sua implementação atual faz uso do PSpice, porém poderia ser utilizado outro simulador, ou funções ("*toolboxes*") do próprio MATLAB.

Os detalhes da utilização do Pspice na plataforma será melhor explicado no item 2.3.

2.1.3 Bloco Funcional do Configurador de Hardware Externo (Intrínseco)

Este módulo corresponde às funções de simulação intrínseca. Não foi feito desenvolvimento específico para este módulo neste trabalho, no entanto, o sistema atual já possui rotinas que fazem uso de interface serial para: envio de vetores de instrução, carga de aplicativos em posições de memória específicos e instruções de execução e parada de programas.

Estas rotinas foram implementadas apenas validar o uso da plataforma com dispositivos externos de simulação intrínseca.

2.2 **Plataforma de Evolução Extrínseca de Circuitos Eletrônicos Adaptativos**

A plataforma de desenvolvimento proposta neste trabalho implementa uma versão extrínseca ("*software*") da plataforma PAMA-NG (Amaral, 2002). A PAMA-NG é uma plataforma de evolução intrínseca - "*hardware*" - constituída de um sistema multiplexador que suporta a interligação mútua de até 32 pinos de componentes com fontes externas de sinais e "*buffers*" e uma placa de conversão A/D e D/A para gerar e ler múltiplos sinais.

Na plataforma proposta o número de pinos é indeterminado, limitando-se apenas pela capacidade de Processamento do PSpice (cujos limites não tivemos acesso) ou pela atribuição do usuário.

Os demais processos são equivalentes, uma vez que ambas geram representações de cromossomas e os enviam ao destino pertinente para que se possa obter uma resposta que será posteriormente comparada a um padrão e avaliada quanto à aptidão.

Ambas mapeiam os terminais de componentes em seus respectivos genes que formam o cromossoma representativo do circuito a ser sintetizado, adaptado ou corrigido, permitindo que milhares de topologias possam ser simuladas (plataforma de evolução extrínseca) ou testadas (PAMA-NG) em busca de uma solução viável para o problema.

A Figura 21 mostra a forma geral da representação de um cromossoma para síntese de um circuito com os seguintes componentes: três resistores (R1, R2 e R3) e um transistor NPN (Q1); também pode ser visto um possível arranjo de nós e finalmente o diagrama esquemático do circuito com os respectivos nós.

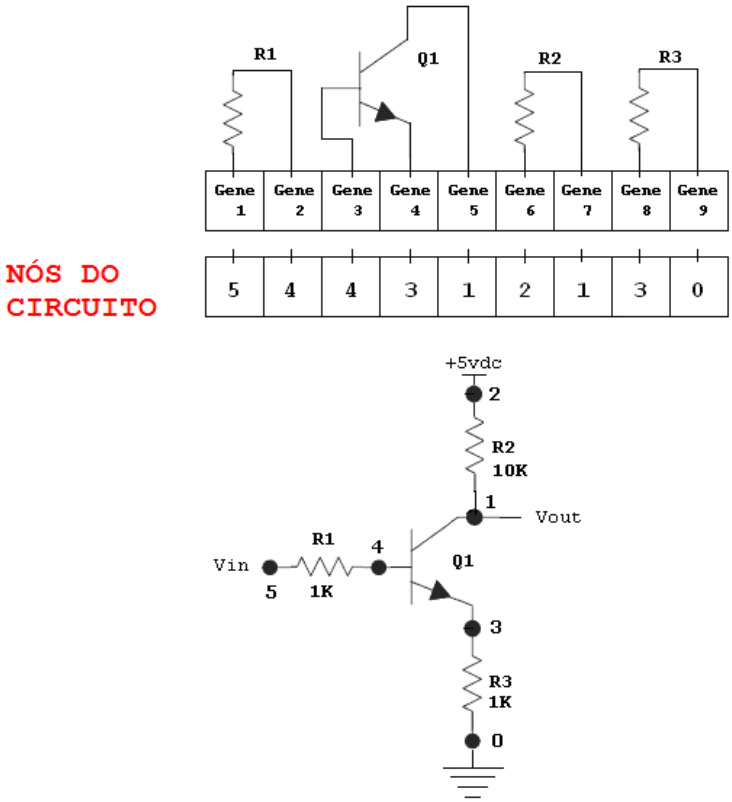


Figura 21 - Mapeamento do Cromossoma e sua implementação em circuito

Na plataforma PAMA-NG cada gene é enviado a um multiplexador que faz a ligação ao correspondente nó no barramento analógico. A Figura 22 mostra esta multiplexação.

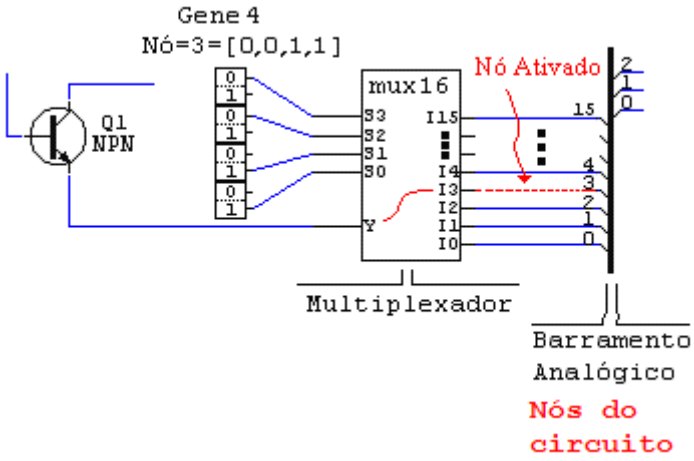


Figura 22 - Codificação do Gene "Emissor de Q1" na PAMA-NG (Amaral, 2002)

Na plataforma proposta cada gene já está com a informação que será inserida no corpo do arquivo *.CIR* e enviado ao PSpice para simulação. A Figura 23 apresenta o arquivo no formato PSpice (*.CIR*) com a representação do mesmo gene em destaque.

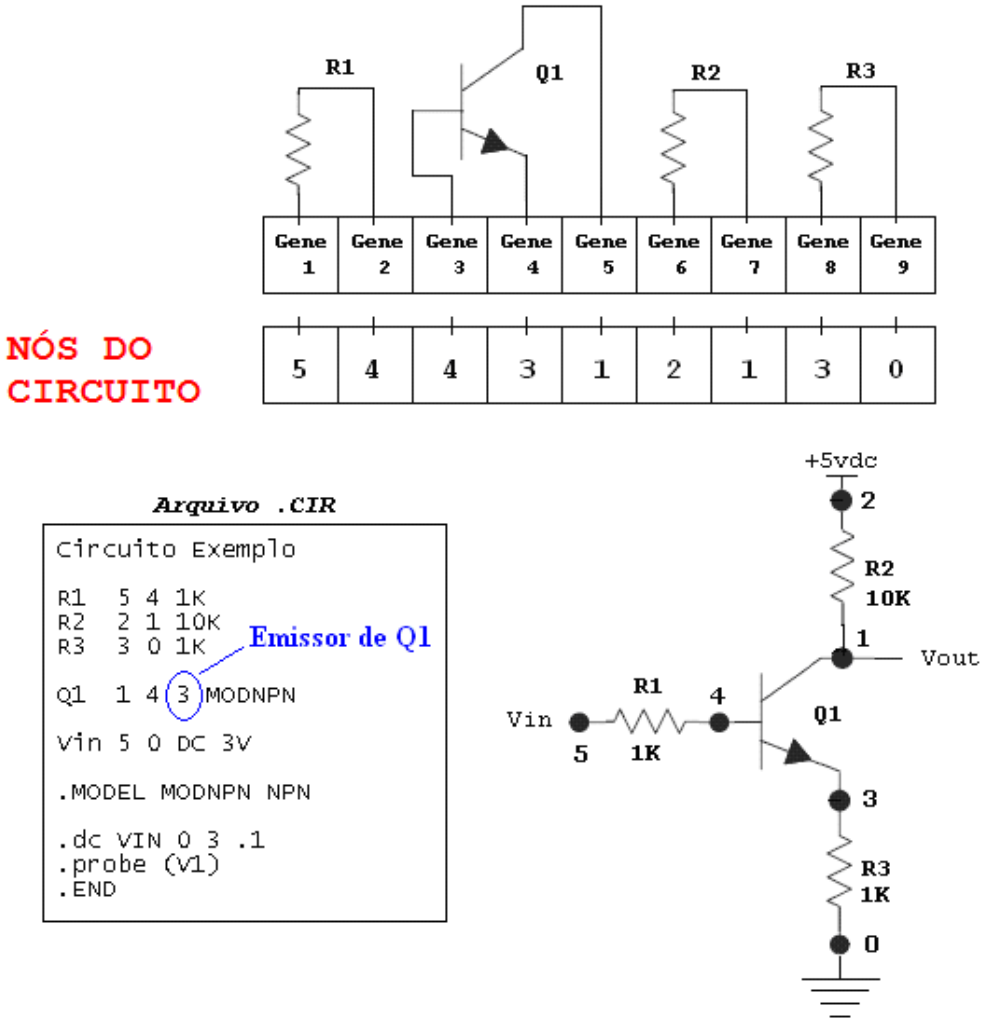


Figura 23 - Codificação do Gene "Emissor de Q1" na Plataforma Proposta (em destaque)

O arquivo *.CIR* mostrado na Figura 23 apresenta cada componente do circuito e suas ligações de nós. Os componentes polarizados ou com mais de dois terminais têm seus terminais determinados pelo modelo usado (a critério do usuário), no caso acima usamos o modelo do transistor nativo do PSpice (NPN) que é definido com a sequência: Coletor, Base e Emissor, respectivamente ligados aos nós 1, 4 e 3. Os terminais dos resistores são de livre conexão, pois não são polarizados. Será realizada a análise DC com Vin variando de zero a três volts com passos de .1v. O arquivo *.DAT* será gerado apenas com a tensão V1 em seu conteúdo. Caso a instrução *.probe* fosse deixada sem argumentos, todos os nós seriam inseridos no arquivo *.DAT* de saída.

2.2.1 O Algoritmo Genético

O algoritmo genético utilizado no trabalho é um pacote desenvolvido por Houck, C. R. et al., 1995, intitulado GAOT ("*Genetic Algorithm for Optimization Toolbox*"). Trata-se de uma ferramenta feita para ser incorporada como um "*toolbox*" do MATLAB, com módulos que são funções distribuídas em arquivos tipo "*.m*".

A principal função do pacote, mostrada na Figura 24, é a que faz chamada ao algoritmo genético propriamente dito, e possui um conjunto de parâmetros que define a totalidade das demais funções e características que serão automaticamente ativadas (chamadas) quando da execução do programa.

```
[x endPop bestPop trace] = GA (bounds, evalFn, evalOps, startPop, gaOpts,...
                termFn, termOps, selectFn, selectOps, xFns, xOpts, mFns, mOpts)
```

Figura 24 - Chamada da função GA
Seus vários parâmetros ilustram a modularidade e capacidade de controle do algoritmo pelo usuário

A função "GA" deve ser chamada de dentro de um programa MATLAB junto com um conjunto de parâmetros que definirão o comportamento e forma de trabalho do algoritmo genético. Abaixo apresentamos o detalhamento dos parâmetros mais importantes e que se relacionam diretamente com este trabalho apresentado e suas qualidades de modularidade.

Os parâmetros e tipos de cruzamento e mutação do GAOT foram escolhidos de forma consistente com a utilizada na plataforma de referência – PAMA-NG.

Estrutura do Cromossoma:

A variável **bounds** e sua respectiva estrutura contêm a quantidade de genes e, para cada gene, os limites de variação de seu conteúdo. Trata-se de uma matriz [m x 2] onde cada linha 'm' corresponde a um gene do cromossoma e cada coluna corresponde aos limites mínimo e máximo de variação do conteúdo de cada gene. No caso da plataforma descrita neste trabalho cada gene é um nó no circuito a ser simulado. A matriz "bounds" abaixo contém 4 linhas (uma por gene), onde cada uma das linhas contém os limites de variação do conteúdo de cada Gene (0 até 5):

```
Bounds = [ 0  5;
           0  5;
           0  5;
           0  5];
```

Função de Avaliação (evalFn):

É o nome da função (definida pelo usuário) que será chamada para avaliação de cada indivíduo da população. É nesta função que fazemos a chamada ao PSpice para simular o circuito e na qual também fazemos os cálculos para dar a "nota" final do indivíduo (avaliação de aptidão). A função de avaliação é implementada num arquivo tipo ".m" do MATLAB e pode conter chamadas externas para outros programas ou funções. Esta característica permite flexibilidade de opções de expansão de funcionalidades da plataforma.

População Inicial (startPop):

É a matriz que contém a população inicial (definida pelo usuário) usada na primeira rodada do algoritmo. É importante observar que o GAOT fornece uma função especial para gerar esta população automaticamente (*initializeGa*). Neste trabalho nós fizemos uso desta função, porém, o ideal é gerar uma população em que só estejam contidos circuitos simuláveis para a primeira rodada.

Função de Seleção (selectFns):

Permite ao usuário selecionar a função que será usada na escolha dos indivíduos que farão parte da próxima geração. A Tabela 3 mostra os tipos de função de seleção disponíveis.

Nesta versão da plataforma de desenvolvimento usamos apenas a Normalização Geométrica (*Normalized Geometric Select*)

Nome	Arquivo
Roulette Wheel	roulette.m
Normalized Geometric Select	normGeomSelect.m
Tournament	tourn.m

Tabela 3 - Funções de Seleção de Indivíduos

Função de Cruzamento (xFns):

Permite especificar a função de cruzamento "*crossover*" que será usada. Os parâmetros da função de cruzamento, incluindo a percentagem da população selecionada que será afetada, é informada no parâmetro xOpts. A Tabela 4 mostra os modelos de cruzamento disponíveis.

Nesta versão da plataforma de desenvolvimento usamos apenas o Cruzamento Simples (*Simple CrossOver*)

Nome	Arquivo
Arithmetic Crossover	arithXover.m
Heuristic Crossover	heuristicXover.m
Simple Crossover	simpleXover.m

Tabela 4 - Funções de Cruzamento

Função de Mutação (mFns):

Permite especificar a função de mutação que será usada. Os parâmetros da função de mutação, incluindo a percentagem da população selecionada que será afetada, é informada no parâmetro **mOpts**. A Tabela 5 mostra os tipos de função de mutação disponíveis.

Nesta versão da plataforma de desenvolvimento usamos apenas a Mutação Binária (*Binary Mutation*).

Nome	Arquivo
Binary Mutation	binary.m
Boundary Mutation	boundary.m
Multi-Non-Uniform Mutation	multiNonUnifMut.m
Non-Uniform Mutation	nonUnifMut.m
Uniform Mutation	unifMut.m

Tabela 5 - Funções de Mutação

Função de Terminação (termFns):

Permite ao usuário fornecer o critério de parada do algoritmo: número de gerações ou valor final da avaliação. A Tabela 6 mostra os tipos de função de seleção disponíveis.

Nome	Arquivo
Terminate at Specified Generation	maxGenTerm.m
Terminate at Optimal or max gen	maxGenOptTerm.m

Tabela 6 - Funções de Terminação de Evolução

As variáveis de saída são apresentadas abaixo:

- x: é o melhor cromossoma avaliado até a geração atual
- endPop: é a população final (não aparece até que a condição de terminação tenha sido atingida)
- bestPop: é a população com os melhores avaliados até a geração atual
- trace: é uma matriz que guarda, a cada geração, o melhor resultado e a média das avaliações.

O usuário pode escolher livremente os parâmetros e as funções do GAOT: seleção, terminação cruzamento e mutação.

2.2.2 A Estrutura do Cromossoma

Dentro da estrutura do algoritmo genético, a variável "*bounds*" define a estrutura do cromossoma e seus valores limites. Cada gene de "*bounds*" representa um terminal de um componente na plataforma e será preenchido com um número de nó no circuito a ser simulado.

Para um arranjo qualquer a representação seria feita da seguinte forma:

(1) Determinar a quantidade de pinos que será efetivamente manipulada pelo algoritmo genético: Quantidade de pinos (terminais de componentes). Não devem entrar nesta conta os pinos de ligação de alimentação de componentes como por exemplo, amplificadores operacionais, a menos que se deseje investigar situações fortemente não-convencionais, ou seja, por exemplo, aquelas em que os terminais de alimentação podem estar conectados como pinos funcionais do circuito, fazendo a função de um terminal de componente qualquer, podendo ser ligado livremente aos demais componentes e nós do circuito.

(2) Informar a quantidade de nós que se deseja – esta quantidade deve permitir ou contemplar pelo menos: o nó de entrada e o nó de saída. Adicionalmente outros nós "básicos" devem ser considerados: nó de referência das tensões (GND) e nós de alimentação, além de outros para permitir a conexão dos terminais de componentes. A quantidade efetiva (desejada) de nós poderá ser informada pelo projetista.

O cromossoma teria a forma geral de uma matriz [1 x T] onde T seria o número de pinos a serem conectados na plataforma. Assim, cada pino de um componente corresponde a um gene do cromossoma.

O conteúdo de cada gene será um número que corresponde a um dos possíveis valores de nós. A Figura 25 mostra a forma genérica do cromossoma da plataforma.

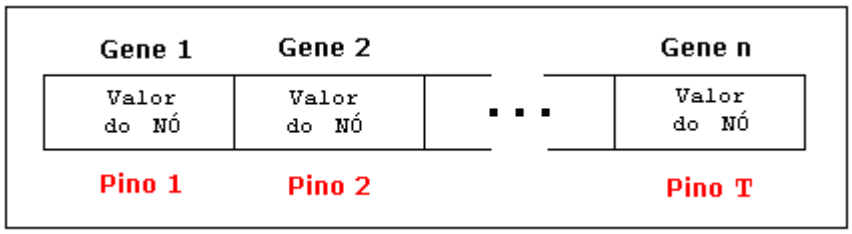


Figura 25 - Forma genérica do cromossoma da plataforma

Cada gene pode conter valores de nó variando de zero (“*ground*”) até um valor limite determinado pelo usuário.

A atribuição do número de possíveis nós tem conseqüências no desempenho da evolução do circuito uma vez que uma quantidade grande de nós pode levar a muitos circuitos não-simuláveis (por excesso de componentes sem nenhuma conexão); por outro lado, uma quantidade pequena de nós pode restringir demais as opções de ligação dos componentes, dificultando a busca por uma solução viável.

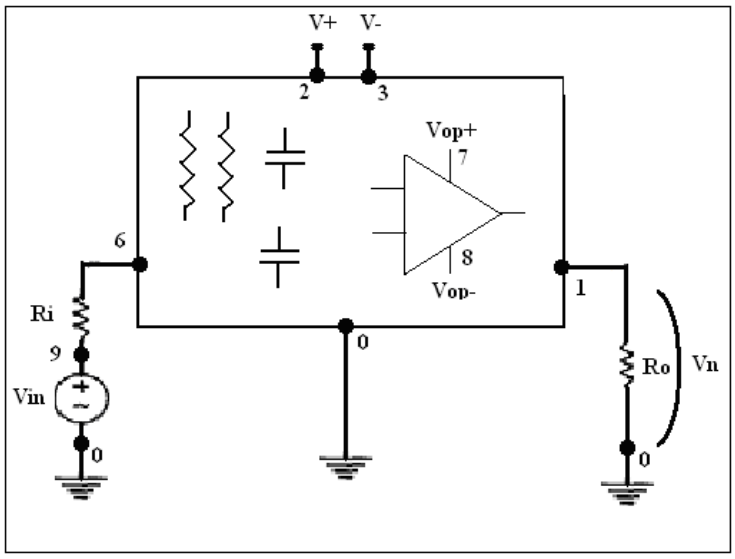


Figura 26 - Circuito Exemplo

Para o circuito da Figura 26, teríamos os seguintes parâmetros:

- Quantidade de Pinos = 11 (2 resistores, 2 capacitores, 1 amplificador operacional)
- Quantidade arbitrada de nós: 6.

O cromossoma teria a forma final apresentada na Figura 27:

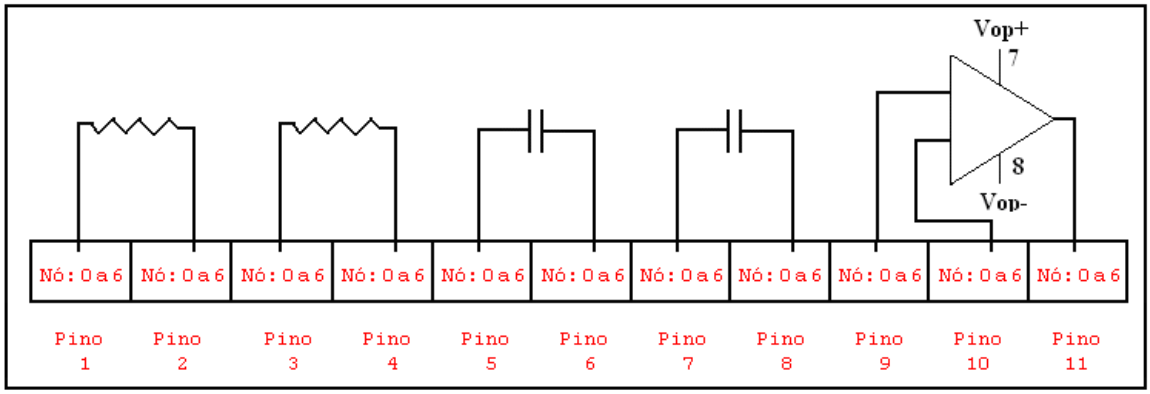


Figura 27 - Cromossoma do circuito exemplo com o posicionamento dos componentes nos genes.

2.2.3 A chamada ao PSpice

O PSpice usado na plataforma é uma versão para o sistema operacional DOS. Para que o PSpice possa ser chamado de dentro do MATLAB um programa especial de configuração de ambiente (dos4gw.exe) precisa estar presente no diretório do PSpice e tem que ser ativado na chamada. Outro Programa, o "psexec.exe", faz a chamada final ao PSpice e executa a simulação do circuito.

A chamada ao PSpice se faz com a execução da seguinte linha de comando (Figura 28):

```
dos(['c:\spice\dos4gw.exe c:\spice\psexec.exe c:\spice\circuito.cir','-echo']);
```

Figura 28 - Chamada ao PSpice de dentro do MATLAB

2.2.4 A função de avaliação

A função de avaliação é um programa do MATLAB cujo nome tem que ser fornecido na chamada do algoritmo.

No caso da plataforma apresentada, a função de avaliação compara a curva de saída de um circuito simulado em PSpice com uma curva pré-estabelecida (padrão) e fornece uma "nota" conforme sua similaridade. Tentamos minimizar o erro médio em relação à curva de saída desejada. Quando o erro médio está próximo de zero significa que a "nota" será alta e que temos um bom candidato a solução final.

A Curva-Padrão pode ser criada de duas maneiras:

- Usando o MATLAB para gerar uma tabela com os pontos desejados (quantidade de pontos e seus respectivos valores: ordenadas e abcissas). Esta tabela deverá ser convertida e salva como um arquivo tipo *.mat* para posterior uso pelo MATLAB na comparação com a saída do circuito simulado.
- Usando o próprio PSpice para gerar um arquivo *.DAT* a partir de um circuito conhecido. Este arquivo *.DAT* deverá ser convertido e salvo como um arquivo tipo *.mat* para posterior uso pelo MATLAB na comparação com a saída do circuito simulado (esta conversão pode ser feita com o uso da função "readdat.m" – ver item 2.3.2.

A avaliação da aptidão do indivíduo é calculada pela equação abaixo. Deste modo, a evolução é realizada com o objetivo de maximizar a aptidão. Os valores de aptidão, com o uso das equações da Figura 29, vão sempre estar entre 0 e 1, sendo 1 o mais apto dentro dos critérios de aptidão adotados no algoritmo de avaliação.

$$\text{Erro} = \frac{1}{N} \sum_{i=1}^{N} \left| V_{\text{esperado}}(i) - V_{\text{obtido}}(i) \right|$$

$$\underset{(\text{Val})}{\text{Aptidão}} = \frac{1}{1 + \text{Erro}}$$

N é o número de pontos medidos

Figura 29 - Cálculo da aptidão de um indivíduo

A curva padrão que serve como referência (Valor Esperado) pode ser gerada pelo PSpice (arquivo *.DAT*) após a simulação do circuito de referência, ou por outro meio qualquer que permita geração de um arquivo "*.mat*" do MATLAB com os dados da curva-padrão. O

arquivo “*.mat*” é um tipo de arquivo característico do MATLAB cuja finalidade é armazenar valores e estruturas de dados.

A função de avaliação pede dois parâmetros de entrada:

- “sol” - individuo a ser avaliado - cromossoma;
- “options” - contém parâmetros de execução do GA.

As varáveis de saída são:

- “sol” - individuo que foi efetivamente avaliado – este valor retorna para a população atual.
- “val” – valor da avaliação do indivíduo

O fato do indivíduo “sol” ser retornado para a população, após a execução da função de avaliação, é uma característica muito importante deste algoritmo, pois qualquer alteração produzida no circuito para torná-lo simulável retorna automaticamente à população atual.

A Figura 30 mostra a estrutura da chamada da função de avaliação com os respectivos parâmetros de entrada (“sol” e “options”) e de saída (“sol” e “val”).

```
[sol,val] = CromoFitness(sol,options)
```

Figura 30 - Parâmetros da função de Avaliação

2.3 **O Simulador de Circuito - PSpice**

Dois pontos podem ser destacados em relação ao trabalho com este componente da plataforma:

- A necessidade de se evitar circuitos que não possam ser simulados.
- A forma de acesso aos dados da simulação.

Ambos os pontos têm relação direta com o desempenho da plataforma na avaliação da aptidão dos indivíduos.

Evitando Circuitos Não-Simuláveis

No trabalho com o PSpice deve-se tomar certos cuidados para evitar circuitos que levem o PSpice a situações indesejáveis:

- Não-simulação do circuito;
- Permanecer executando em um laço contínuo sem conseguir convergir para uma resposta de simulação;

Os tipos mais comuns de circuitos que o PSpice rejeita são os que contém:

- componentes sem nenhuma ligação;
- componente somente ligado a ele mesmo;
- componentes ligados entre si mas sem nenhuma conexão com partes ativas do circuito eletrônico;

Os tipos mais comuns de circuitos que levam o PSpice a ficar simulando sem conseguir convergir são tipicamente aqueles em que pode haver realimentação positiva:

- circuitos com amplificadores operacionais com ligação direta entre terminais de saída e não inversor;
- circuitos com vários transistores.

Uma forma de evitar estes casos é procurar retirar do circuito ou adaptar as ligações dos componentes:

- ligando todos os pinos num mesmo ponto ativo do circuito (entrada ou saída);
- ligando a saída do componente num resistor de valor conveniente ligado a GND (sem efeito no circuito).

Em qualquer dos casos, o cromossoma modificado tem que ser mandado de volta a população atual através do uso da variável "*sol*", conforme abordado anteriormente.

Caso o Spice fique executando um circuito e laço por muito tempo, a melhor forma de resolver é parar a execução do PSpice ("matando" o processo com uso de comandos do sistema operacional) e atribuir àquele indivíduo uma avaliação baixa.

2.3.1 Mapeamento do Circuito no Cromossoma e geração do arquivo de simulação

No curso do desenvolvimento foi verificada a incidência de 70% de circuitos não simuláveis. Outros pesquisadores também relatam incidências de circuitos não-simuláveis na faixa entre 50% e 80% na população inicial (Salazar & Mesquita, 2000), levando-os a encontrar diferentes formas de minimizar ou evitar este problema, tais como: Matriz de adjacência (Mesquita et al., 2002) e Construtores de Circuitos (cc-bot) (Sá, L. B., 2009).

Para tentar minimizar estes problemas fizemos uma alteração no desenho básico do circuito a ser representado (Lohn et al., 2000), fixando entrada e saída em nós pré-determinados que podem ser endereçados como nós válidos dentro dos genes do cromossoma.

Os terminais "*start node*" e "*end node*" não fazem parte do cromossoma, mas o valor de seus nós pode ser atribuído a qualquer um dos genes do cromossoma.

A Figura 31 mostra a colocação dos nós de forma que possam ser manipulados posteriormente pelo algoritmo genético, facilitando o processo de simulação na plataforma proposta.

Os nós de alimentação (tipicamente: V+ e V-) também têm seus valores fixados de maneira que caiam dentro do limite de possibilidades de conexão do cromossoma, neste caso os nós 2 e 3, desta forma qualquer componente do cromossoma pode ser ligado diretamente a fontes de referência interna de tensão.

Os nós de alimentação de componentes complexos como amplificadores operacionais devem ficar fora dos limites de nós do cromossoma, neste caso nós 7 e 8 ("n+1" e "n+2", considerando que neste exemplo "n" é igual a 6). A princípio eles poderiam estar ligados a "V+" e "V-", porém, para o caso geral, é mais indicado que estejam ligados de forma independente.

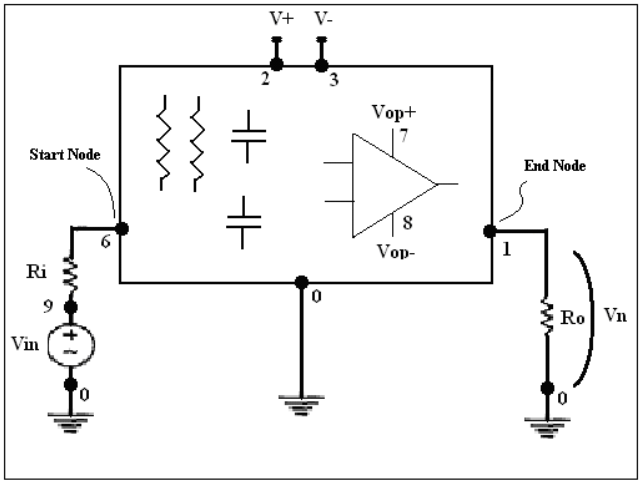


Figura 31 - Esquema geral do circuito a ser evoluído.

O pino referência das tensões (“ground”) fica sempre no nó 0 (Zero), o PSpice trabalha com este nó fixo.

Seguindo estas especificações, os genes do cromossoma podem assumir qualquer valor entre os nós de entrada e os nós de saída, incluindo o nó de referência (nó 0 ou ground).

Os geradores de sinais ficam com os nós mais externos, neste caso nó 9.

Resistores de Carga e de resistência interna de fontes devem ser inseridos com valores convenientes, sendo o resistor de carga sempre entre os nós de saída e ground, neste caso os nós 1 e 0; e a resistência interna de geradores de sinais sempre entre o nós das entradas e os respectivos nós de geradores, neste caso entre nós 6 e 9.

Veja que neste exemplo o número de nós estipulado “n” é 6, correspondendo ao pino do resistor Ri (resistência interna da fonte Vin) a partir do qual o circuito a ser evoluído será ligado.

A Figura 32 apresenta uma possível representação cromossomial e sua respectiva implementação em circuito.

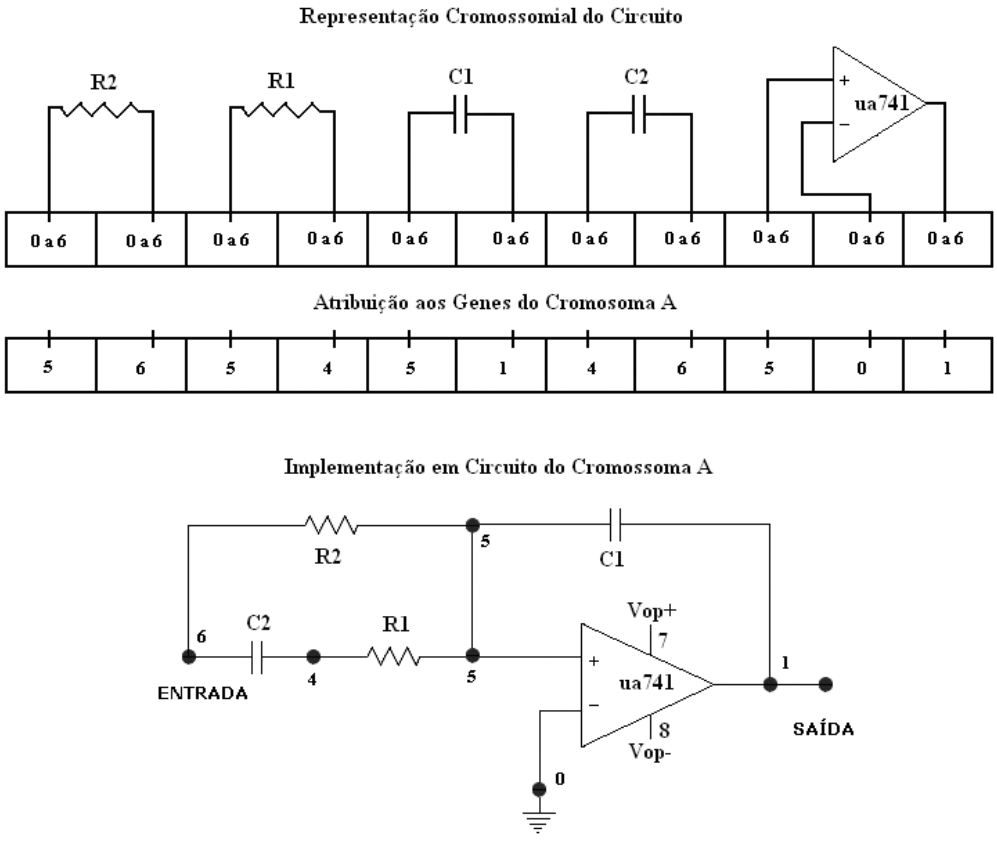


Figura 32 - Representação do Circuito e a implementação de um cromossoma

Os possíveis nós de ligação do cromossoma estão localizados ente os nós 0 e 6. A utilização desta abordagem na distribuição dos nós internos e externos ao cromossoma leva o

arranjo a uma situação com menor probabilidade de não-simulação, porque há menos nós, porém ainda não há garantia de que o circuito seja simulável.

A Figura 33 mostra o conteúdo do arquivo de descrição do circuito para simulação no formato do PSpice (*.CIR*).

```
Circuito_Exemplo2

Ri    9 6 600
Ro    0 1 15.00E+03
R2    5 6 100k
R1    4 5 15k
C1    1 5 318p
C2    6 4 1590n

X1    5 0 7 8 1 uA741

V+    2 0 DC 5
V-    0 3 DC 5
Vop+ 7 0 DC 5
Vop- 0 8 DC 5
Vin   9 0 AC 1 sin 0 0.1 100 0 0 0

.ac DEC 101 3 20.00K
.probe V(1)
.END
```

Figura 33 - Arquivo de simulação no Formato PSpice (*.CIR*)

Além dos cuidados e definições de topologia de circuitos mostrados acima, também foram tomadas as medidas já descritas para evitar circuito não-simuláveis.

2.3.2 Acesso aos dados da simulação

O PSpice fornece basicamente dois tipos de saída dos dados da simulação: o arquivo *.OUT* que é um arquivo Texto e um arquivo de dados *.DAT*.

O arquivo de saída precisa ser lido e manipulado pelo MATLAB para que o cálculo da função de avaliação possa ser completado.

O arquivo *.OUT* contém, tipicamente, todos os dados de entrada e os de saída solicitados pelo usuário e eventualmente, através do comando *.plot*, também apresenta um gráfico com valores solicitados pelo usuário. Os dados do arquivo *.OUT* são de fácil visualização, porém de difícil manipulação para os propósitos deste trabalho uma vez que os resultados tem que ser buscados por comparação de cadeias de caracteres (*strings*), incluindo as situações de erro.

O acesso a dados em arquivo texto mostrou-se muito demorado nas primeiras versões da plataforma, principalmente pela perda de tempo nos ciclos de busca e comparação de *strings* para localizar início e fim dos trechos referentes às variáveis de interesse; mesmo um

circuito que não teve sua simulação efetivamente realizada gera um arquivo *.OUT* que tem que ser aberto e lido para verificar se houve simulação ou não.

O arquivo *.DAT* contém apenas dados das simulações em um formato de difícil interpretação, que possui diferentes versões e é variável conforme o tipo de simulação.

Este tipo de arquivo só é gerado quando há uma simulação válida, com isso, não há perda de tempo na busca pela informação em arquivo texto, basta uma simples verificação da presença do arquivo, nativa do sistema operacional, que em geral consome o mínimo de recursos e tempo.

Uma vez que o arquivo existe no diretório, ele apresenta os dados organizados em registros e podem ser lidos com velocidade muito maior.

O arquivo tipo *.DAT* foi o escolhido e utilizamos um componente do MATLAB criado com base no trabalho de Frank Sommerhage (Sommerhage, 2011), para o qual contribuímos com acertos no algoritmo de tratamento do arquivo e revisão do formato de saída do arquivo.

O algoritmo de leitura, interpretação e formatação de arquivos *.DAT* chama-se "***readdat.m***". A versão anterior deste programa possuía erros na interpretação dos dados e também não fazia a leitura de todos os dados disponíveis nos registros internos do arquivo *.DAT*.

A versão corrigida permite um arquivo de saída com todos os nós internos do circuito. Foi feita uma comparação de leitura do arquivo quando carregado pelo MATLAB e quando carregado por outros "software" de simulação (versões do PSpice) e verificamos que as leituras foram idênticas.

Antes das correções os vetores de saída continham dados de mais de um nó e o usuário tinha que interpretar o vetor e extrair aquilo que desejava. Além disso, antes de nossa correção, o arquivo *.DAT* só apresentava metade dos dados disponíveis para cada nó.

Uma versão atualizada deste componente pode ser encontrada no site oficial da MathWorks (MATLAB Central – File Exchange) com os devidos créditos para a UERJ e para o autor.

3 ESTUDO DE CASOS

Os casos apresentados a seguir foram desenvolvidos com a finalidade de verificar o desempenho e a flexibilidade da plataforma.

Os objetivos foram determinar a funcionalidade da etapa de concepção da plataforma, verificar a técnica de implementação de circuitos eletrônicos através de algoritmos genéticos e principalmente verificar adaptabilidade de circuitos usando a plataforma.

Os estudos de casos estão divididos em duas partes. Na primeira, foi verificada a estrutura da plataforma e validada as formas de uso e chamada de seus respectivos componentes. Nesta fase foi desenvolvido o modelo de circuito básico, a estrutura dos cromossomas.

Na segunda parte outros circuitos foram evoluídos e comparados com trabalhos anteriores em outras plataformas (Santini, 2001, Amaral, 2003 & Sinohara, 2001). Apresentam-se circuitos desenvolvidos com sucesso tendo por base os diferentes componentes utilizados, desde componentes discretos, como resistores e transistores, até blocos mais complexos, como, por exemplo, amplificadores operacionais em sistema de controle PID.

O computador usado nos experimentos é um notebook DELL Vostro 1000 com 2.5GB de RAM Total – Processador AMD SEMPRON 1.99GHz – Sistema Operacional Windows XP Versão 5.1.26000 Home Edition SP3).

3.1 Circuito de Validação da Plataforma

Neste primeiro experimento usamos um filtro passa-faixa para o qual procuramos uma topologia específica que fornecesse a resposta de tensão de saída na faixa entre 3KHz e 20 KHz apresentada no Gráfico 1.

Nesta modelagem disponibilizamos vários componentes incluindo aqueles que poderiam apresentar a solução ideal. Nas inúmeras rodadas de evolução encontramos muitos circuitos que apresentavam curvas muito semelhantes à desejada.

Durante os testes fizemos diversas tentativas de alteração de quantidades de nós e variações nos parâmetros da simulação (mutação e cruzamento) com o objetivo de verificar o comportamento dos diversos componentes da plataforma em situações diversas. Não foi evidenciado nenhum cenário que inviabilizasse a utilização da plataforma em sua versão extrínseca. Contudo, observamos que quanto maior a quantidade de nós disponíveis na

simulação maior a quantidade de circuitos não-simuláveis. Estes circuitos não-simuláveis elevam muito o tempo de resposta do PSpice na análise. Tipicamente, o tempo de resposta a um circuito não simulável é duas vezes maior que a simulação de um circuito simulável com os mesmos componentes.

A Figura 34 mostra o circuito usado como padrão em todos os testes iniciais e também na fase de acerto e "*debug*" do algoritmo de leitura do arquivo *.DAT* do PSpice.

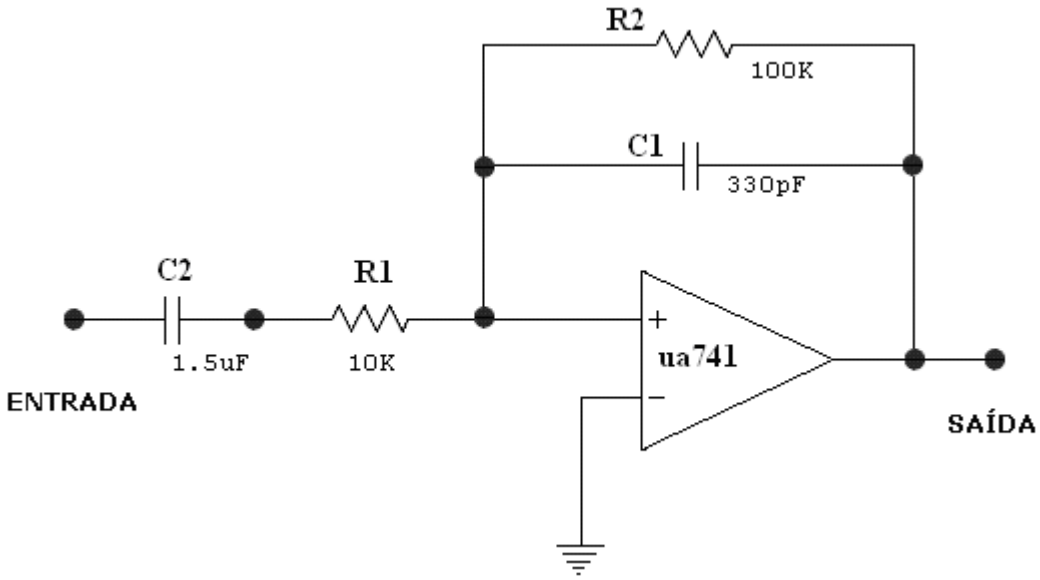


Figura 34 - Circuito Ideal do filtro com a saída desejada

O Gráfico 1 mostra a curva de resposta padrão do circuito da Figura 34.

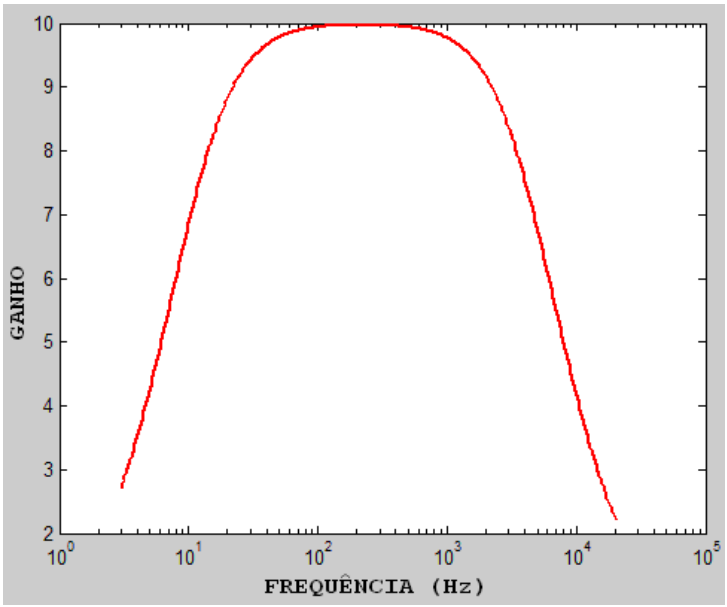


Gráfico 1 - Curva de resposta do filtro passa-faixa

Para esta simulação disponibilizamos para o algoritmo genético os seguintes componentes: um resistor de 100K, um resistor de 10K, um capacitor de 330pF, um capacitor de 1.5uF, um amplificador operacional ua741, um resistor de 82K, um resistor de 15K, um capacitor de 270nF, um capacitor de 2,7uF, um resistor de 47K e um resistor de 22K, respectivamente na mesma ordem em que aparecem na Figura 35, que mostra a representação do cromossoma usado para evolução do circuito do filtro.

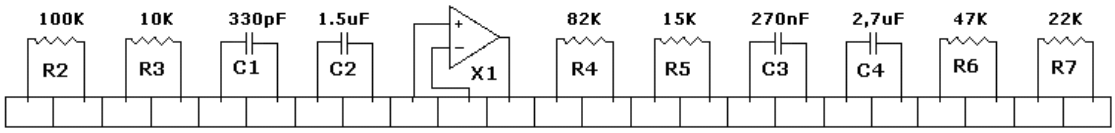


Figura 35 - Representação do cromossoma na evolução do filtro

Com esta quantidade de componentes adicionais foi observado um tempo grande para convergência do circuito, tendo encontrado a curva original depois de 50 gerações de 100 indivíduos. Cabe ressaltar que durante o processo evolucionário, vários circuitos com topologias as mais diversas, atendiam parcialmente às especificações da curva padrão.

A Figura 36 mostra um destes circuitos "evoluídos" que apresentou resposta similar (ainda que distante do padrão), com uma topologia muito diferente da esperada (ver Figura 34), mas que, em determinadas situações não críticas, poderia servir como circuito alternativo.

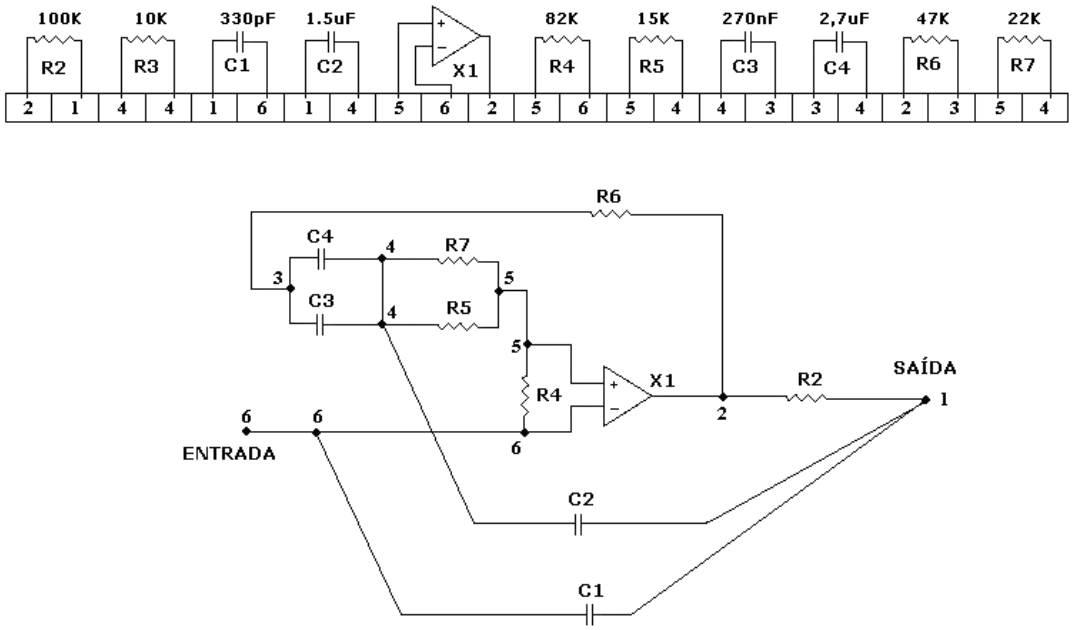


Figura 36 - Circuito evoluído com respectivo cromossoma

O Gráfico 2 apresenta as curvas de saída esperada (padrão) e saída conseguida com o circuito evoluído, de topologia não-convencional, apresentado na Figura 36.

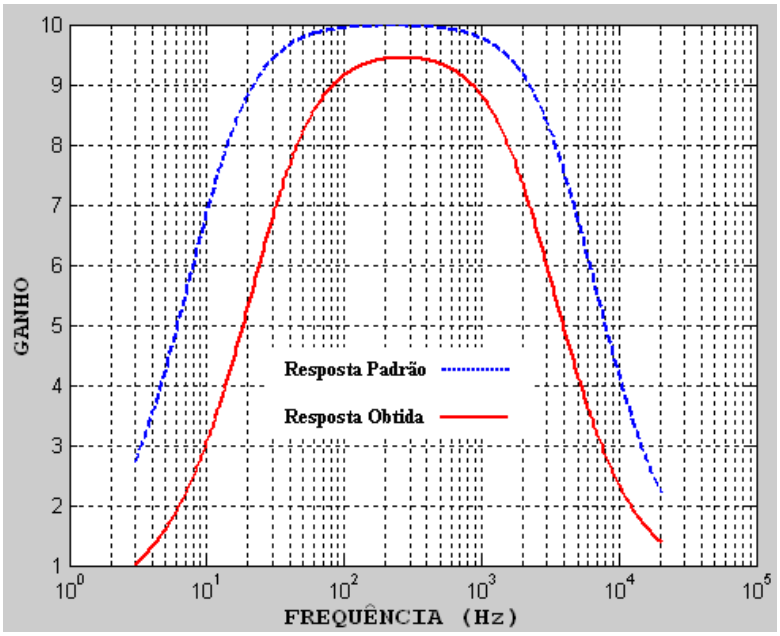


Gráfico 2 - Saída padrão (tracejado) e Saída obtida do circuito evoluído (contínuo)

3.2 Implementação de função de pertinência de sistema Fuzzy

Esta função foi sintetizada no trabalho de Amaral (Amaral, 2003) e nossa proposta para este experimento foi verificar o uso da plataforma de desenvolvimento na síntese de circuitos capazes de implementar funções, em particular as funções de pertinência.

O Gráfico 3 mostra a curva da função “Z” de referência usada neste experimento

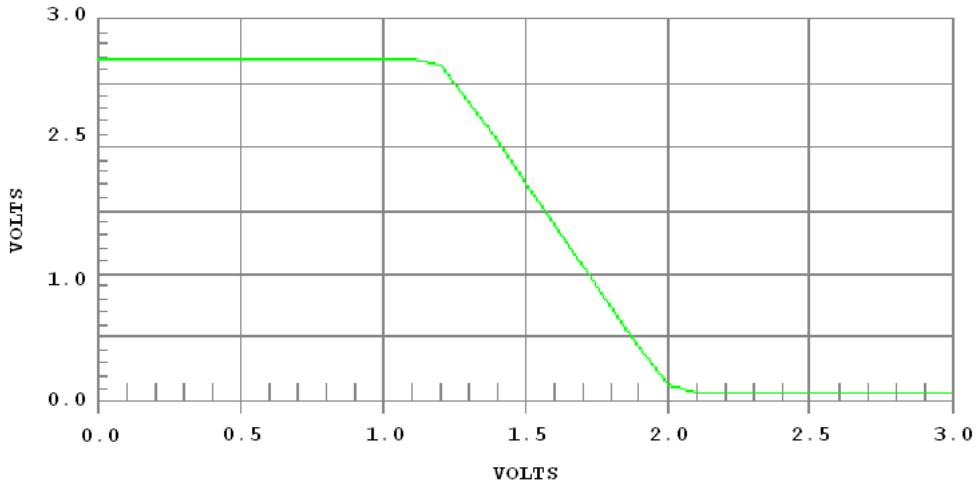


Gráfico 3 - Função de Pertinência do circuito Fuzzy (Amaral, 2003)

O diagrama da Figura 37 mostra o circuito sintetizado por Amaral usando uma plataforma intrínseca. Os resistores de 100 ohms correspondem à resistência interna dos multiplexadores que interligam os componentes na plataforma. Os valores destes resistores foram incorporados ao valor do resistor na plataforma extrínseca.

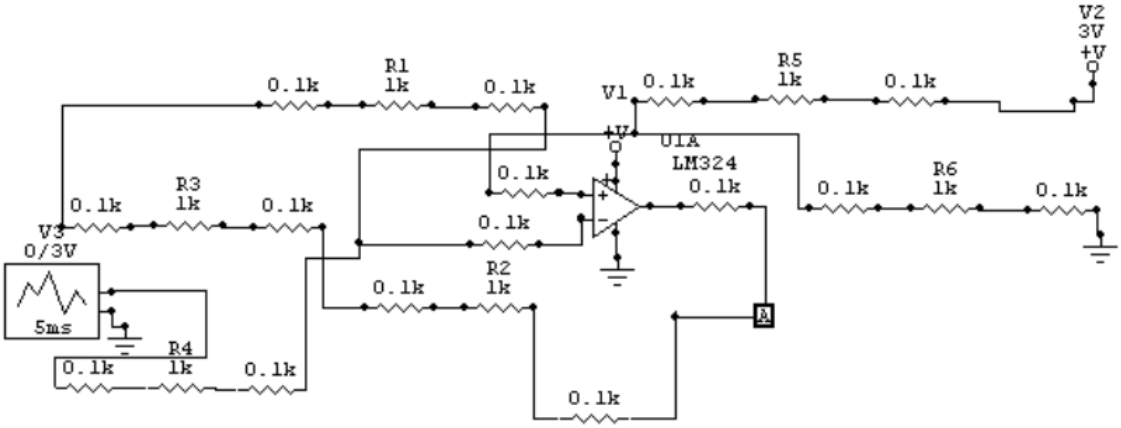


Figura 37 - Diagrama do circuito evoluído para a função Z (Amaral, 2003)

A Tabela 7 mostra a relação de parâmetros usados no desenvolvimento da função Z

Número de Gerações	50
Número de Indivíduos	200
Taxa de Crossover	0,7
Taxa de Mutação	0,1

Tabela 7 - Parâmetros do Algoritmo Genético da Função Z

Para o desenvolvimento da solução na plataforma proposta consideramos os resistores com 1.200 ohms, que equivale no circuito de referência ao resistor de 1K em série com as duas resistências de cada multiplexador.

Os resultados obtidos foram compatíveis com o evidenciado no trabalho de referência e o circuito evoluído foi rigorosamente o mesmo.

Ficou clara a diferença de tempo de execução do algoritmo até que o melhor indivíduo fosse encontrado. Na plataforma de referência (intrínseca) é possível simular um circuito a cada 320ms, enquanto na plataforma extrínseca simulamos um circuito a cada 3 segundos.

Cabe ressaltar que os circuitos não-simuláveis aumentam o tempo de simulação em até 3 vezes.

A plataforma proposta tem boas chances de ter desempenho compatível com o simulador intrínseco de referência com a utilização de computador com maior capacidade de processamento e com aplicação de outras práticas mais apuradas de construção e correção de circuitos.

3.3 Adaptação a Falha em circuito Pré-Amplificador

Este experimento teve como objetivo avaliar o comportamento da plataforma de desenvolvimento num processo de adaptação evolutiva para correção de falhas. Este mesmo circuito foi apresentado por Sinohara, 2001.

A Figura 38 mostra o diagrama do circuito proposto.

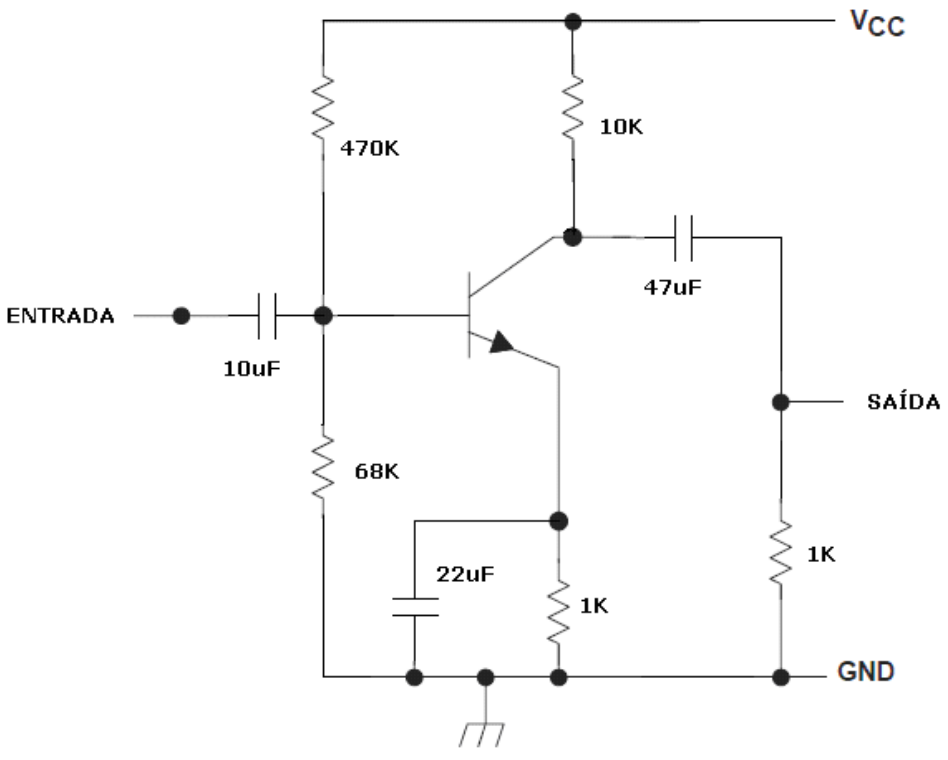


Figura 38 - Circuito pré-amplificador (Sinohara, 2001)

O Gráfico 4 mostra as curvas de Entrada (linha mais clara marcada com círculos) e Saída. A entrada tem valor de pico 0,01V e a saída tem valor de pico de 0,22V. A pequena quantidade de pontos no gráfico se deve a saída gerada pelo PSpice na simulação que forneceu os dados de referência (padrão) com o circuito original apresentado na Figura 38.

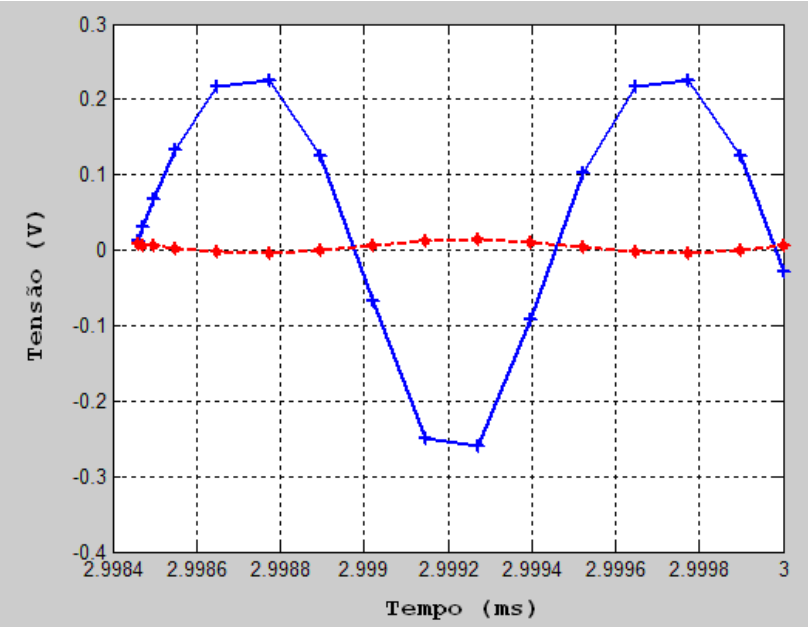


Gráfico 4 - Entrada e Saída Padrão do circuito pré-amplificador
A linha mais escura representa a saída

A Figura 39 mostra o esquema do circuito com a introdução de duas falhas que inutilizam o circuito em sua função principal. A primeira falha simula o rompimento do resistor de coletor. A segunda falha corresponde a uma perda quase total das características do resistor de emissor – saindo de uma resistência de 1K para apenas 0,1ohm.

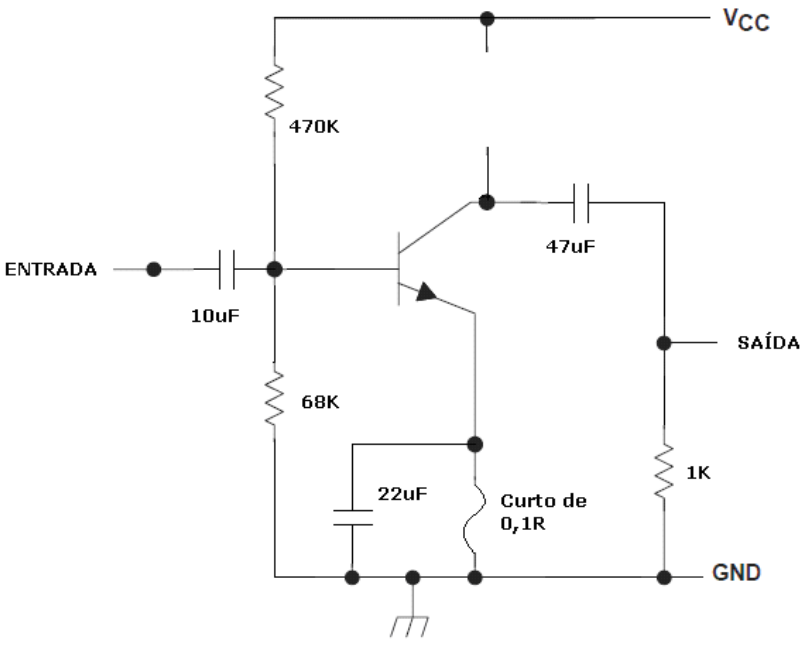


Figura 39 - Circuito pré-amplificador após a introdução de falhas

Para este experimento, o circuito com defeito foi modelado como se fosse um componente (tipo caixa-preta), dele foram disponibilizados apenas os nós do coletor e do emissor (dos componentes defeituosos). Os componentes adicionais disponibilizados para o trabalho do algoritmo genético foram: 4 resistores de 1K2 e um transistor com a mesma especificação do transistor do circuito.

A representação do cromossoma deste experimento é mostrada na Figura 40.

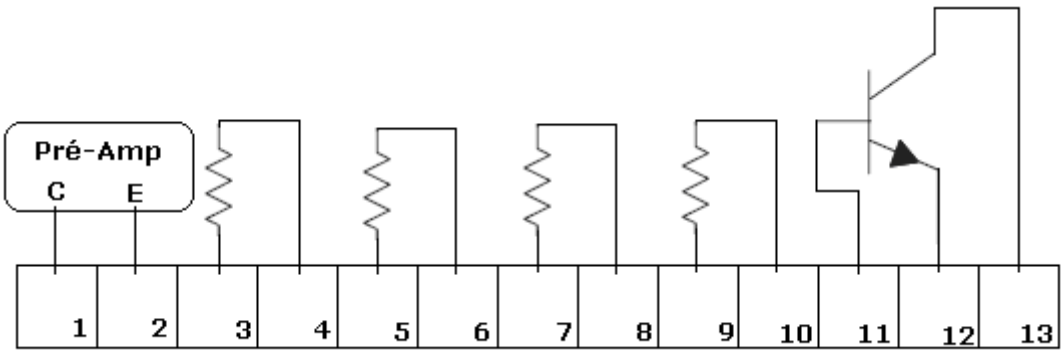


Figura 40 - Representação da estrutura do cromossoma para adaptação a falha

O Gráfico 5 mostra as curvas de entrada e saída. A curva tracejada em cor mais forte e marcadas com sinal "+" corresponde a saída do circuito reparado pelo processo de adaptação. O valor de pico foi sensivelmente reduzido em 14% – de 0,22 para 0,19. O circuito foi recuperado de uma situação de completa inutilidade

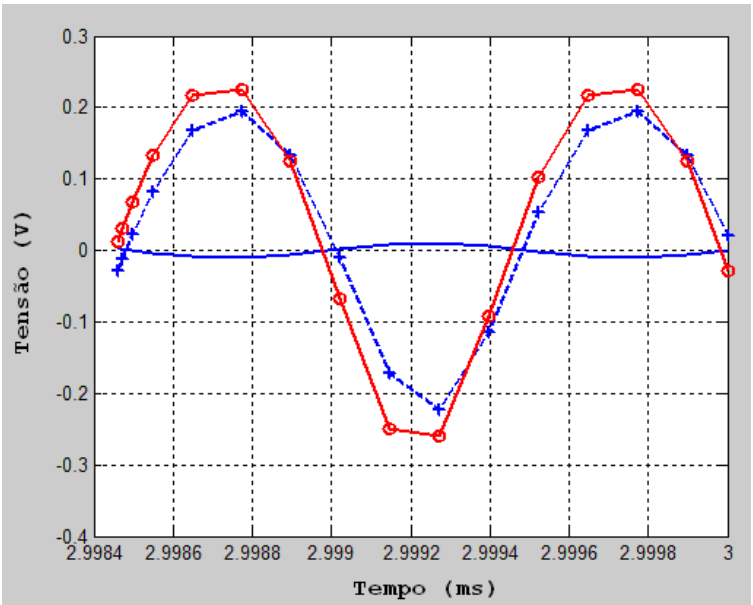


Gráfico 5 - Curvas de Entrada, Saída Padrão e Saída após a adaptação evolucionária
A entrada aparece em cor escura, a saída padrão está marcada com "o"
e a saída aparece marcada com "+"

A Figura 41 apresenta o resultado final do processo de recuperação de falha.

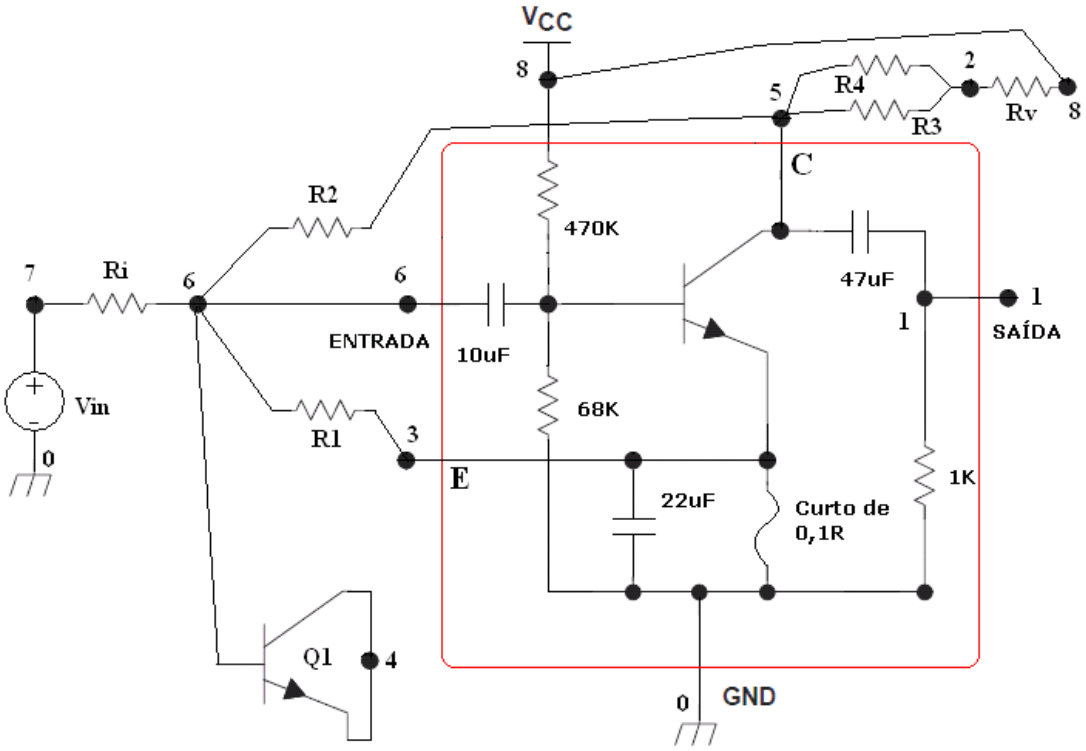


Figura 41 - Circuito pré-amplificador adaptado por evolução genética

Podemos observar que o circuito evoluído tem características bem diferentes do circuito original e sua topologia não pode ser chamada de convencional. Os dados de configuração do algoritmo genético para esta evolução são vistos na Tabela 8. A solução encontrada difere em sua topologia daquela apresentada por Sinohara, 2001.

Número de Gerações	20
Número de Indivíduos	50
Taxa de Crossover	0,7
Taxa de Mutação	0,1

Tabela 8 - Parâmetros do algoritmo genético usado no processo adaptativo do pré-amplificador

3.4 Adaptação a Falha em circuito Inversor (Modelo 7404)

O circuito apresentado na Figura 42 é o modelo funcional da porta inversora 7404 obtida do “data sheet” do fabricante do componente. Este mesmo circuito foi usado por Sinohara, 2001 na avaliação da capacidade de reparo plataforma intrínseca PAMA (Santini, 2001).

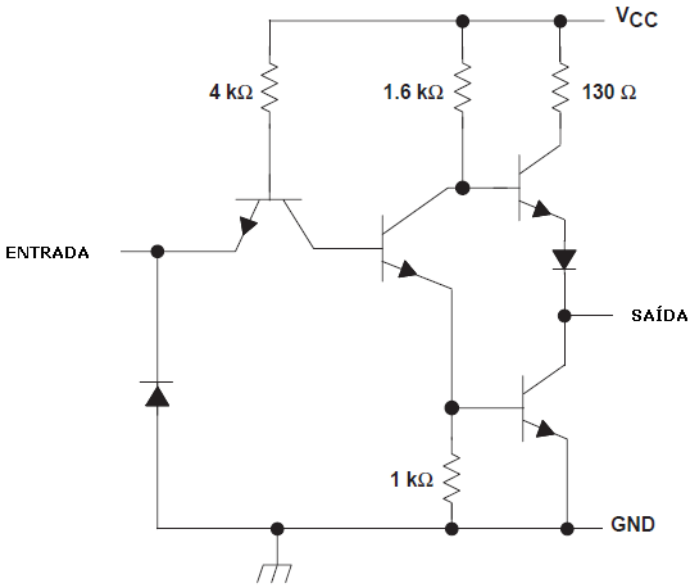


Figura 42 - Circuito modelo do inversor 7404

O Gráfico 6 mostra as curvas de entrada e saída. A linha sólida representa a entrada e a outra representa a saída padrão.

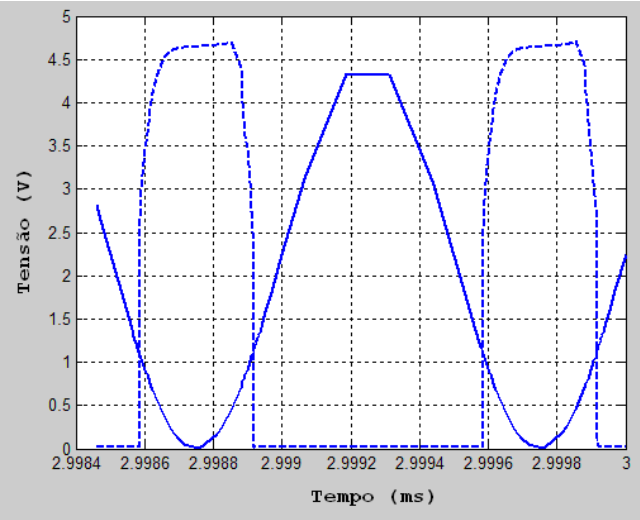


Gráfico 6 - Curvas de entrada e referência do circuito inversor 7404

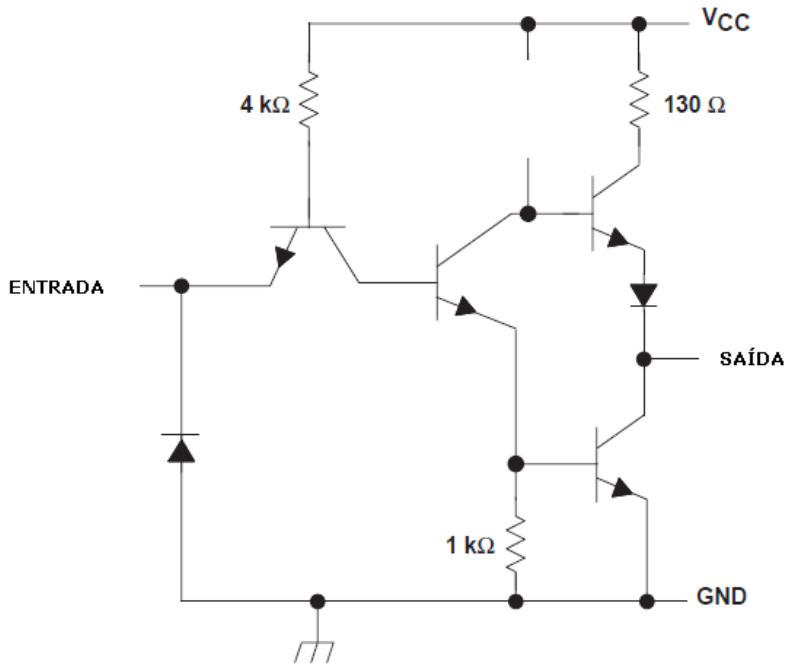


Figura 43 - Diagrama do Circuito inversor com defeito

Para a adaptação deste circuito, com ausência do resistor de 1K6, disponibilizamos 4 resistores de 1K2 e um transistor do mesmo modelo dos transistores internos do circuito (Figura 44).

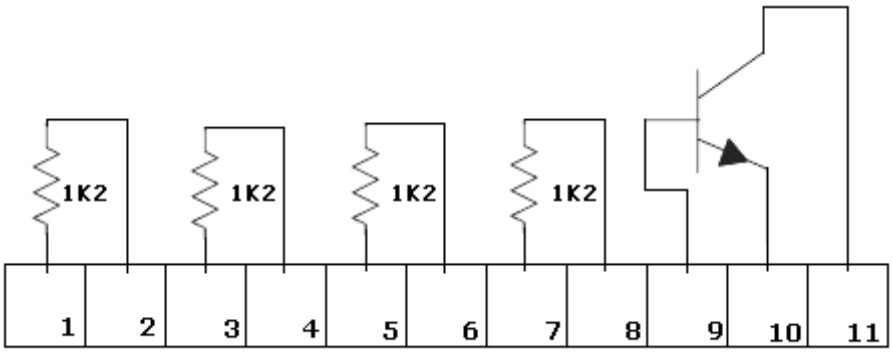


Figura 44 - Representação Cromossomial para adaptação do circuito Inversor Lógico.

Neste modelo de evolução permitimos que os componentes se ligassem a qualquer nó interno do circuito, aumentando as possibilidades de adaptação a falhas em quaisquer componentes do circuito e também a alterações de condições externas como temperatura ou carga na saída.

O Gráfico 7 mostra as curvas de entrada (linha escura sólida), de saída (linha mais clara sólida) e a marcação dos valores da curva de saída (cruzes escuras)

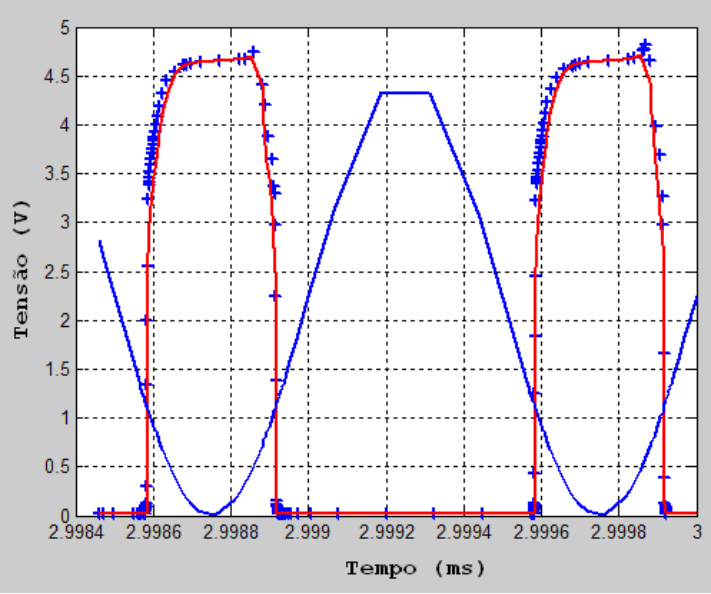


Gráfico 7 - Curvas de Entrada, Saída e Referência

O circuito adaptado (Figura 45) foi gerado sem o uso de um dos resistores, pois o mesmo foi ligado somente a seus próprios terminais. Nestes casos, para evitar circuitos não-simuláveis, retiramos o componente do arranjo antes de enviar para simulação no PSpice.

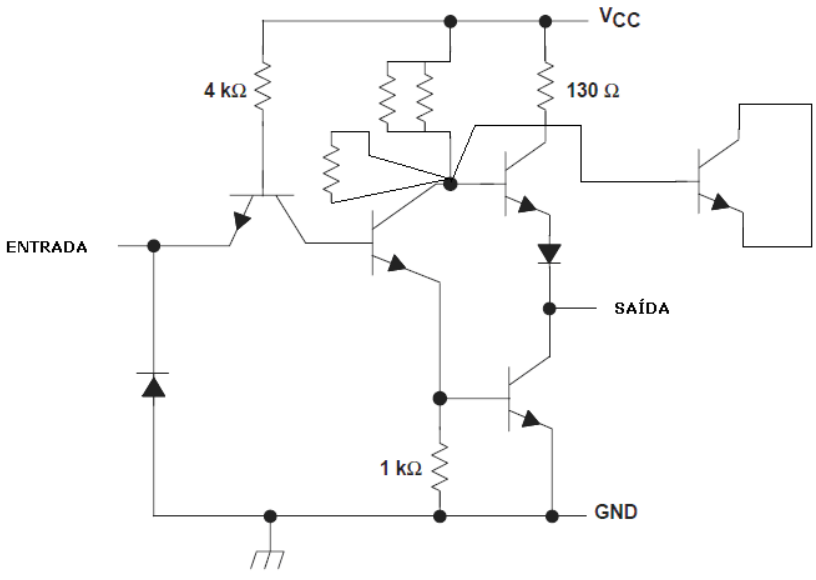


Figura 45 - Circuito adaptado após a introdução da falha

Ao todo foram comparados 266 pontos da curva de saída. Os parâmetros do algoritmo genético são apresentados na Tabela 9.

Número de Gerações	20
Número de Indivíduos	20
Taxa de Crossover	0,7
Taxa de Mutação	0,1

Tabela 9 - Parâmetros do Algoritmo Genético do Inversor

3.5 Adaptação a falha em circuito com controle PID

O circuito da Figura 46 apresenta o diagrama eletrônico didático de uma planta controlada por um controlador PID. Neste experimento será provocada uma falha na malha interna do PID de forma que a saída do processo (nó 22) fique fora de especificação. A plataforma será usada para adaptar o circuito e tentar corrigir a falha de maneira satisfatória.

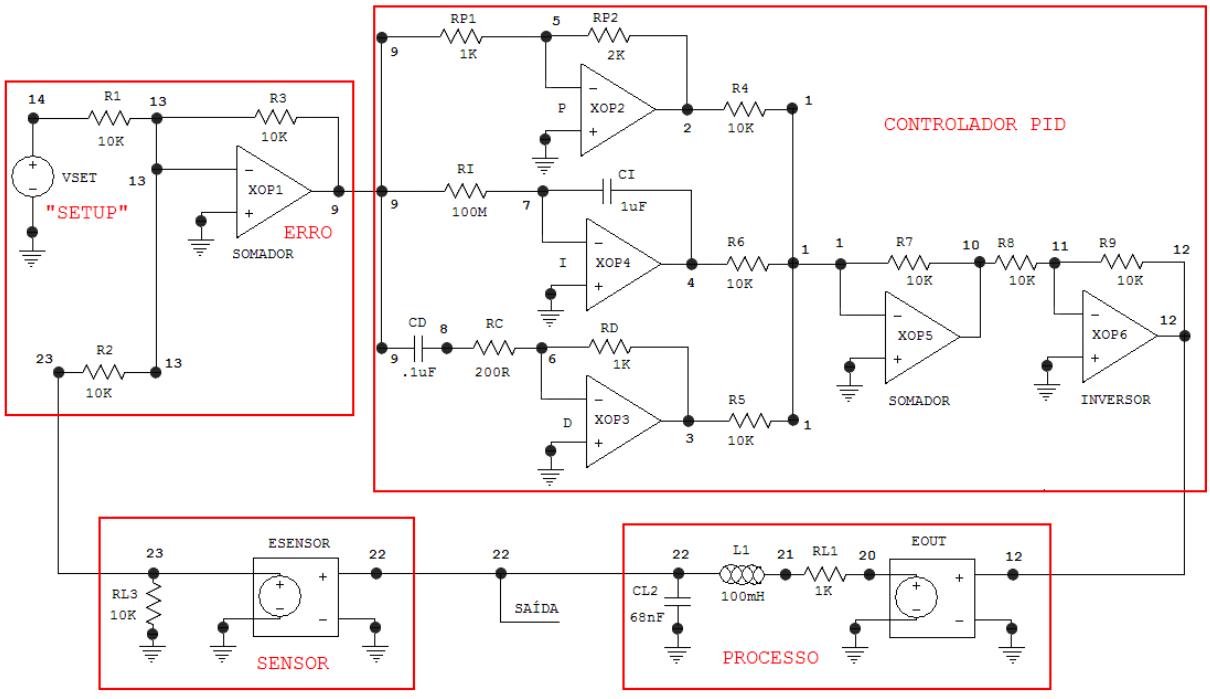


Figura 46 - Diagrama da planta controlada com PID

O Gráfico 8 mostra o comportamento do sistema em regime normal de trabalho com a resposta que foi considerada como padrão para esta análise.

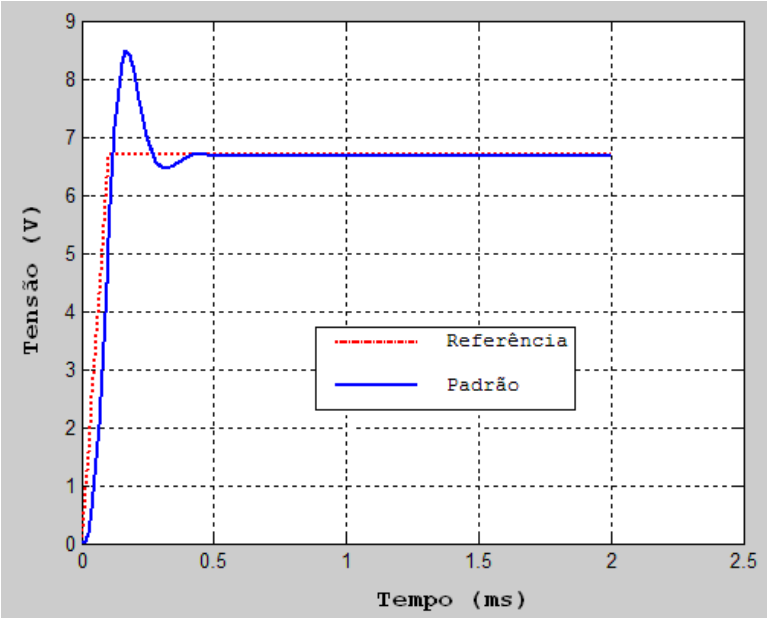


Gráfico 8 - Curvas de referência e padrão de Saída

O defeito introduzido no circuito é apresentado na Figura 47 e foi gerado pela alteração do resistor "RD" – o valor nominal foi aumentado em 50 vezes, o que causou um comportamento oscilatório indesejável. O Gráfico 9 mostra o efeito detectado na saída (nó 22).

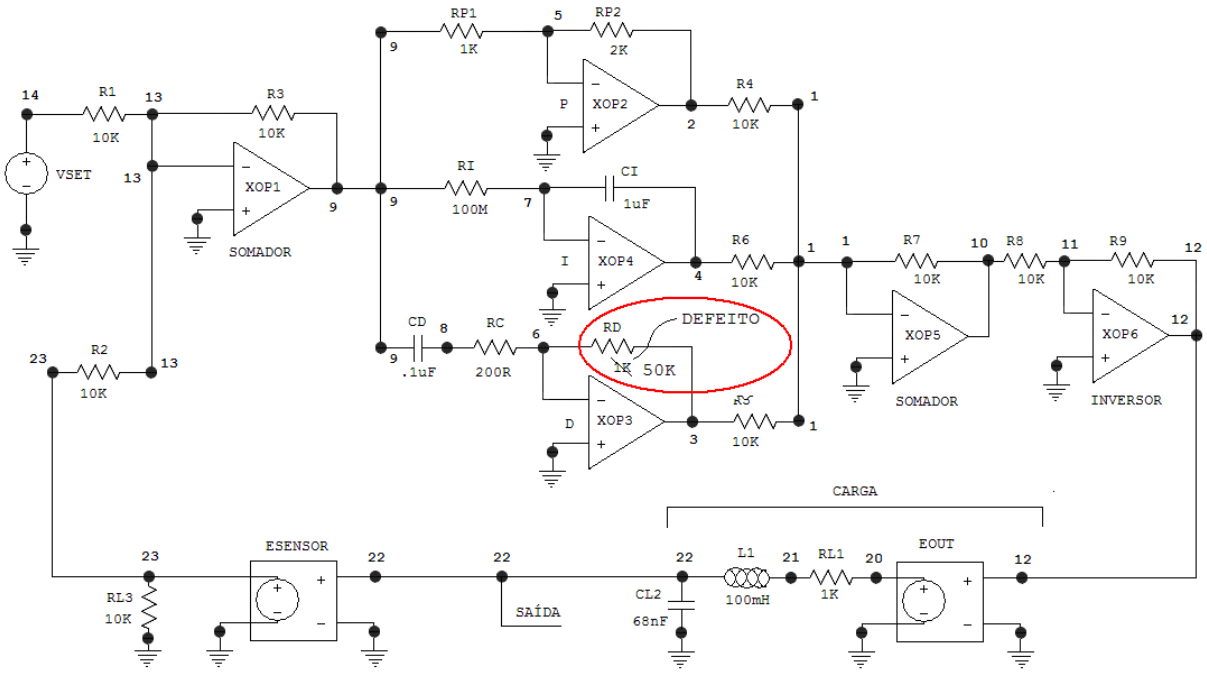


Figura 47 - Circuito de controle com defeito introduzido em RD

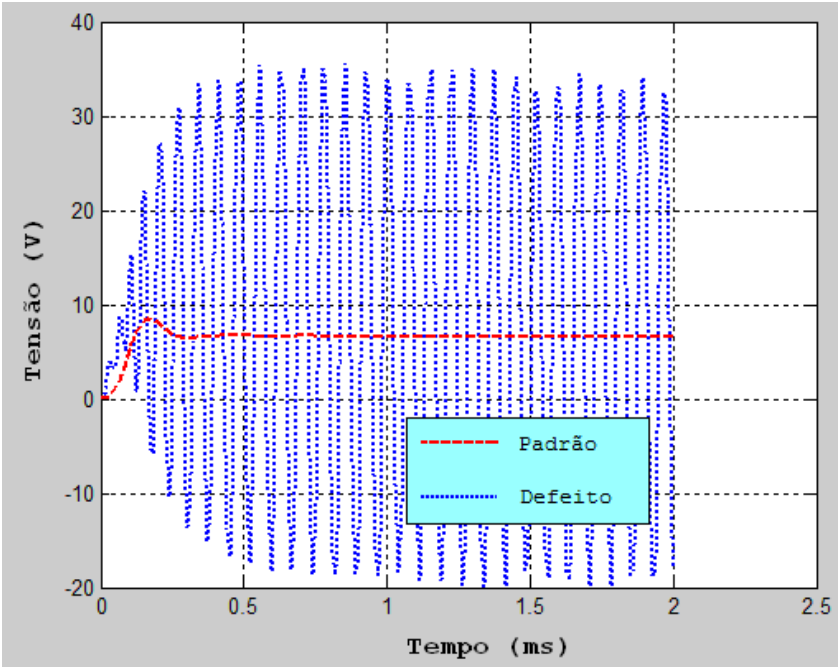


Gráfico 9 - Curva de Saída Padrão e Curva de Saída com Defeito no controle PID

Neste experimento disponibilizamos a malha interna do PID – localizada entre os nós 1 e 9 (incluindo o "*ground*" – nó 0) para a plataforma conectar componentes extras disponibilizados para adaptação do circuito. A Figura 48 mostra o detalhe do circuito que será afetado pela ação de correção da plataforma.

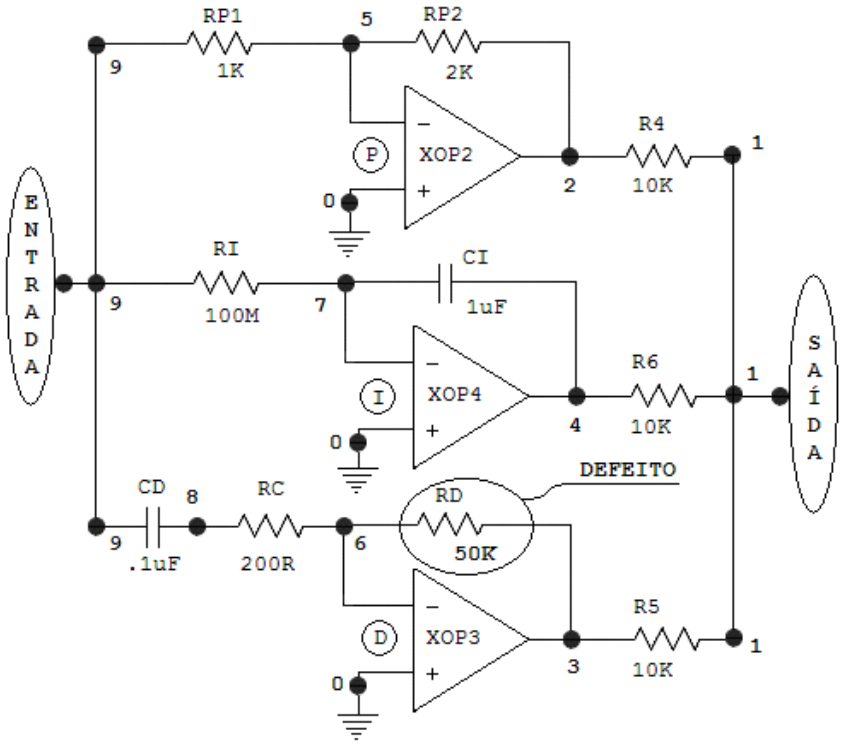


Figura 48 - Diagrama da malha disponível para correção

A Figura 49 mostra a representação cromossomial que será usada no reparo deste circuito. Foram disponibilizados três resistores (1K, 2K e 10K) e dois capacitores (1uF e .1uF) – totalizando 10 terminais de componentes que poderão se ligar aos nós entre 0 e 9.

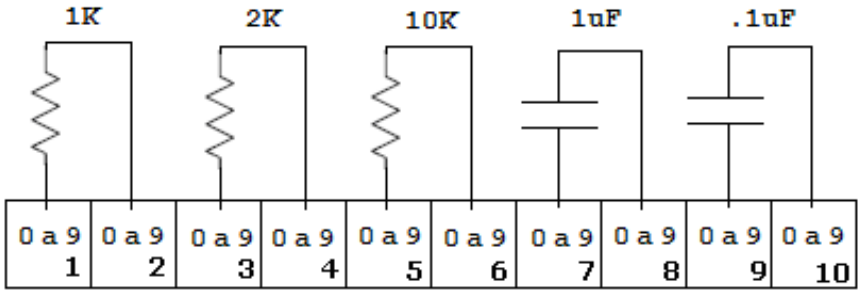


Figura 49 - Cromossoma usado na reparação do Controle PID

A Figura 50 mostra o circuito com os reparos após 30 gerações. A Figura 51 apresenta o cromossoma que corresponde a esta representação.

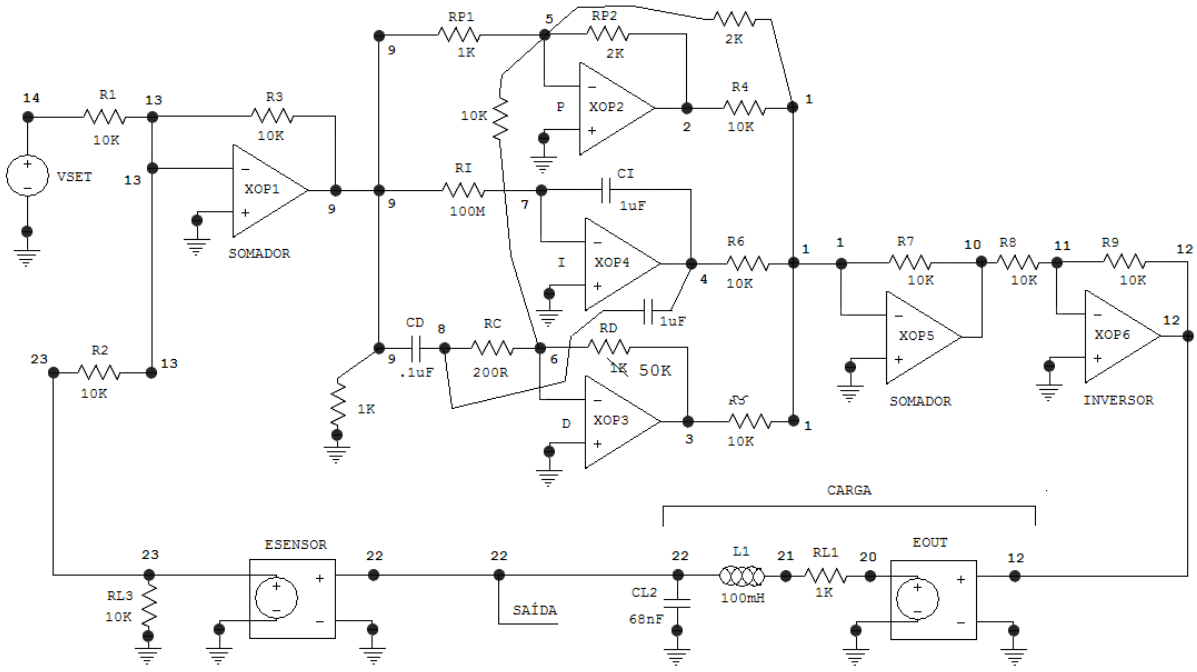


Figura 50 - Diagrama do circuito reparado

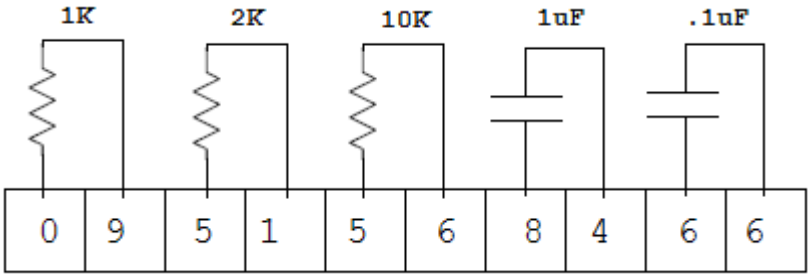


Figura 51 - Representação do cromossoma de reparo

O Gráfico 10 mostra a resposta do circuito reparado. Cabe ressaltar que a adaptação conseguiu obter uma resposta razoável capaz de manter a funcionalidade básica do sistema de controle, recuperando-o de uma situação problemática.

Os componentes foram escolhidos por representarem uma boa chance de reparo de qualquer uma das partes envolvidas na malha a ser corrigida.

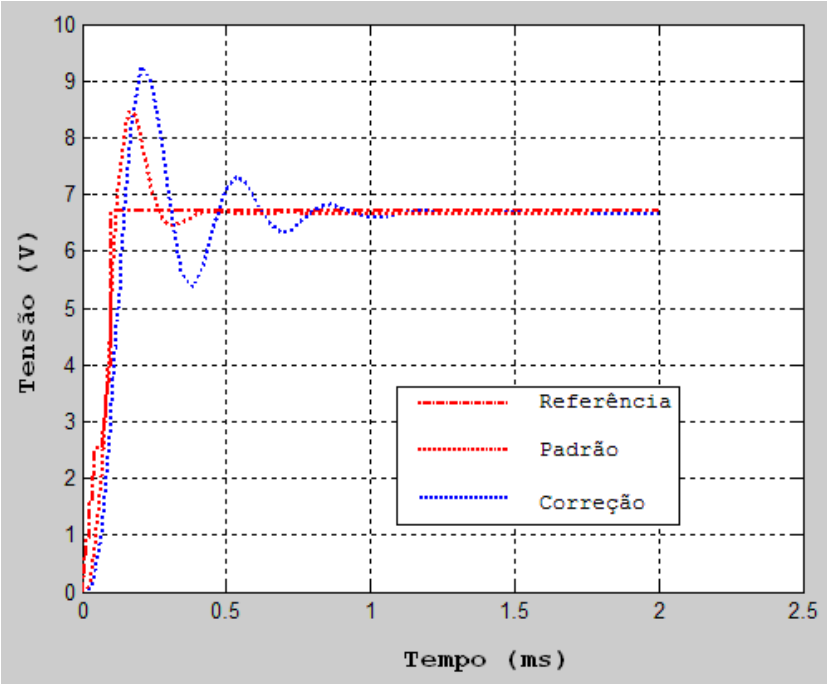


Gráfico 10 - Curva do Circuito PID recuperado da falha

O resultado da evolução do circuito foi conseguido com 30 indivíduos por geração. Vale observar que uma resposta bem mais próxima da ideal poderia ser conseguida, uma vez que havia no cromossoma um resistor de 1K que se fosse aplicado em paralelo com o resistor defeituoso de 50K conseguiria aproximar-se do valor original de 1K. Ainda que esta possibilidade exista, o número de gerações teria que ser aumentado, e o tempo de simulação seria comprometido, retardando a reparação do circuito.

A Tabela 10 mostra os parâmetros usados na execução do GA que resultou no reparo do circuito.

Número de Gerações	30
Número de Indivíduos	30
Taxa de Crossover	0,8
Taxa de Mutação	0,2

Tabela 10 - Parâmetros do algoritmo genético

Este fato aponta para a necessidade de maiores estudos para se identificar as características do genótipo a ser disponibilizado e dos limites de ciclos de evolução para cada situação específica a ser efetivamente implementada em circuitos reais. Esta plataforma foi capaz de resolver de forma satisfatória e rápida o problema apresentado.

4 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou a arquitetura de uma plataforma de desenvolvimento para evolução de circuitos analógicos adaptativos cujo objetivo é favorecer o estudo de algoritmos genéticos e circuitos eletrônicos evolucionários (síntese e adaptabilidade).

Com esta plataforma podem-se estudar alternativas de circuitos e subsistemas e testá-los em software (através de uma interface do MATLAB com o PSpice) de maneira prática e estruturada.

Foi dada ênfase à modularidade e capacidade de expansão de suas partes constitutivas. Também foi levada em conta a aderência às possibilidades materiais e de sistemas atualmente em uso pela instituição, de forma que alunos da graduação, iniciação científica, Pós-Graduação e Mestrado possam utilizá-la com o objetivo de acelerar o processo de aprendizado e/ou expandi-lo.

Verificou-se a utilização da plataforma no desenvolvimento de soluções e na síntese de circuitos segundo especificações de entrada e saída.

Foram testados circuitos com defeitos inviabilizantes e a plataforma foi capaz de adaptar soluções que restabeleceram a operação dos circuitos.

A plataforma se mostrou flexível e de fácil uso para iniciantes no tema e também de grande ajuda para os mais experientes, pois contém elementos que permitem maior organização e documentação do trabalho que esta sendo realizado.

A plataforma implementada neste trabalho é constituída de elementos (ou módulos) que permitem fácil integração (por meio de troca de arquivos) e até substituição integral por outras soluções.

Atualmente o MATLAB já possui complementos de simulação de circuitos. Estes módulos não foram estudados aqui, mas estão disponíveis comercialmente. Programas escritos em Java, C#, Delphi ou outras linguagens comerciais podem fazer uso das interfaces de comunicação do MATLAB e complementá-lo com programas mais poderosos e que utilizam outros módulos externos como interfaces de programação de "*firmware*" ou configuração de circuitos externos.

Programas especiais podem ser escritos para contemplar as funcionalidades do GAOT e desta forma ampliar as possibilidades de desenvolvimento de novos algoritmos e estratégias de adaptação.

Componentes como: FPGA, FPAA, GAL, PAL e outros, podem ser programados "*In Circuit*" pelo envio de arquivos de configuração binários. A plataforma poderia evoluir

circuitos para estes componentes e enviá-los diretamente para avaliação e testes. Esta funcionalidade daria à plataforma uma gama muito mais ampla de possibilidades.

A seguir descrevemos algumas das possibilidades com relação custo-benefício bastante favorável neste momento, sem necessidade de grandes recursos além dos já disponíveis atualmente.

Seria interessante acrescentar um algoritmo automático de criação de circuitos simuláveis para a geração da população inicial. Isto resultaria em melhora do desempenho uma vez que na plataforma atual, em alguns casos, a primeira população tem que ser fortemente corrigida pelo algoritmo de avaliação, o que não é uma operação favorável ao processo de evolução. Alternativamente, um algoritmo mais simples que verificasse se existe pelo menos um caminho completo entre entrada e saída já poderia fornecer uma informação importante ao algoritmo de avaliação.

Com o uso do GAOT seria possível fazer a avaliação paralela (simulação) de circuitos distribuída entre diferentes instâncias do PSpice. Estas instâncias poderiam estar num mesmo computador (com múltiplos processadores) ou em computadores diferentes. Em cada caso o GAOT pode ser alterado para disponibilizar a população atual em arquivo, e instruído a ficar em "*hold*" logo após chamar a função de avaliação para o primeiro indivíduo. Ao término da avaliação, o programa externo encerra sua execução devolvendo o valor da avaliação do primeiro indivíduo. A função de avaliação do GAOT (escrita no MATLAB), apenas fará a leitura sucessiva dos resultados disponibilizados no mesmo arquivo. Um programa externo deverá ser escrito e sua função será a de distribuir em "*threads*" ou "*pipes*" as solicitações de simulação para cada instância. Este programa usará o arquivo contendo a população como entrada e escreverá o resultado de cada simulação no registro correspondente até que a avaliação de todos os indivíduos esteja completa.

Nesta versão da plataforma, o algoritmo genético tem liberdade para fazer cortes, e subseqüente cruzamento, sem levar em conta a natureza do componente que está sendo "cortado", ou seja, o mecanismo pode fazer um cruzamento levando apenas parte das ligações de um componente para formar um herdeiro. Isto pode não ser conveniente e merece ser melhor estudado.

Para a operação de mutação também deve haver cuidado, uma vez que um circuito não simulável poderá ser gerado pela simples troca de nó em um dos terminais de componente.

Em ambos os casos, seria interessante fazer o refinamento destas operações genéticas em conjunto com práticas de construção de circuitos a fim de se garantir que o circuito herdeiro seja minimamente simulável.

REFERÊNCIAS

Al-Naqi, A.; Erdogan, A. T.; Arslan, T.; "**Fault Tolerant Three-Dimensional Cellular Genetic Algorithms with Adaptive Migration Schemes**". 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), 2011, p. 352 – 359.

Amaral, J. F. M.; Amaral, J. L. M.; Santini, C.; Tanscheit, R.; Vellasco, M. M. R.; Pacheco, M. A. C.; "**Towards Evolvable Analog Fuzzy Logic Controllers**". Proc. 2002 NASA/DoD Conference on Evolvable Hardware, IEEE Computer press, July 2002, p. 123 – 128.

Amaral, José Franco do. "**Síntese de Sistemas Fuzzy por Computação Evolucionária**". Rio de Janeiro, 2003, 146 p. Tese de Doutorado (Programa de Pós-Graduação em Engenharia Elétrica - PUC-Rio).

Amaral, J. L. M.; Amaral, J. F. M.; Morin, D.; Tanscheit, R.; "**An immune fault detection system with automatic detector generation by genetic algorithms**". Seventh International Conference on Intelligent Systems Design and Application (ISDA), 2007, p. 283 – 288.

Bäck, T.; Hoffmeister, F.; Schwefel, H.; "**A Survey of Evolution Strategies**". Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, p. 2 – 9, 1991.

Baluja, Shumeet; "**Structure and Performance of Fine-Grain Parallelism in Genetic Search**". In: Chambers, L. Pratical Handbook of Genetic Algorithms: New Frontiers, vol. 2. CRC Press , 1995.

Barker, W.; Halliday, D. M.; Thoma, Y.; Sanchez, E.; Tempesti, G.; Tyrrell, A. M.; "**Fault Tolerance Using Dynamic Reconfiguration on the POEtic Tissue**". IEEE Transactions on Evolutionary Computation, Vol. 11, no. 5, October, 2007. p. 666 – 684.

Barrera, Julio; Flores, Juan J.; "**Search of initial conditions for dynamics systems using intelligent optimization methods**". CERMA 2007, Electronics, Robotics and Automotive Mechanics Conference. P. 348 - 353

Benkhelifa, E.; Pipe, A.; Dragffy, G.; Nibouche, M.; "**Towards Evolving Fault Tolerant Biologically Inspired Hardware Using Evolutionary Algorithms**". IEEE Congress of Evolutionary Computation (CEC 2007), 2007, p. 1548 – 1554.

Fernando, P. R.; Katkoori, S.; Keymeulen, D.; Zebulum, R.; Stoica, A.; "**Customizable FPGA IP Core implementation of a general-purpose genetic algorithm engine**". IEEE Transactions on Evolutionary Computation, Vol. 14, No. 1, february 2010, p. 133 – 149.

Fogel, L. J.; Owens, A. J.; Walsh, M. J.; "**Artificial Intelligence Through Simulated Evolution**". John Wiley, New York, 1966

Goldberg, D.; "**Genetic Algorithms in Search, Optimization and Machine Learning**". Adison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.
Gong Jian; Menfei Yang; "**Robustness of Evolvable Hardware in the Case of Fault and Environmental Change**". Proceedings of the 2009 IEEE International Conference on Robotics and Biometrics, December 2009, p. 1849 – 1853.

Greenwood, G. W.; Tyrrel, A. M.; "**Introduction to Evolvable Hardware: A Practical Guide for Designing Self-Adaptive Systems**". IEEE Press Series on Computational Intelligence. David B. Fogel Series Editor. 2007. ISBN: 978-0-471-71977-9.

Higuchi, T., Iwata, M., Keymeulen, D., Sakanashi, H., Murakawa, M., Kajitani, I., Takanashi, E., Toda, K., Salami, M., Kajihara, N., Otsu, N.; "**Real-World Applications of Analog and Digital Evolvable Hardware**", IEEE Transactions on Evolutionary Computation, Vol. 3, no. 3, Setembro 1999.

Holland, J. H.; "**Adaptation in Natural and Artificial Systems**". University of Michingan Press, Ann arbor, USA, 1975.

Horrocks, D.H.; Khalifa, Y. M. A.; "**Genetically derived filter circuits using prefferred value components**". IEE Colloquium on Analogue Signal Processing, 1994, p. 4/1 – 4/5.

Houck, C. R.; Joines, A. J.; Kay, M. G.; "**A Genetic Algorithm for Function Optimization: A Matlab Implementation**". ACM Transactions on Mathmatical Software, 1996. Disponível para "download" no site: http://bookmark.sunfinedata.com/www-content/articles/language/matlab/2001927-9051, acesso em Fevereiro, 2012.

HuaQu Yang; LiGuang Chen; ShaoTeng Liu; HaiXiang Bu; Yuan Wang; JinMei Lai; "**A Flexible Bit-Stream Level Evolvable Hardware Plataform Based on FPGA**". 2009 NASA/ESA Conference on Adaptive Hardware and Systems, p. 51 – 56.

Keymeulen, D.; Stoica, A.; Zebulum, R.; Katkoori, S.; Fernando, P.; Sankaran, H.; Mojarradi, M.; Daud, T.; "**Self-Reconfigurable Analog Array Integrated Circuit Architeture for Space Applications**". 2007 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2007), 2007, p. 83 - 90.

Keymeulen, D.; "**Self-Repairing and Tuning Reconfigurable Electronics for Space**". 2010 IEEE 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010, p. 1.

Koza, J.R.; Bennett III, F.H.; Andre, D.; Keane, M.A.; Dunlap, F.; "**Automated synthesis of analog electrical circuits by means of genetic programming**", IEEE Transactions on Evolutionary Computation. Vol.1, No.2, July 1997, p.109-128.

Koza, J.R.; Keane, M.A.; Streeter, J. M.; "**The Importance of Reuse and Development in Evolvable Hardware**", Proceedings of the 2003 NASA/DoD conference on Evolvable Hardware, 2003.

Koza, J.R.; "**Survey of Genetic Algorithms and Genetic Programming**", Computer Science Department, Stanford University, 2004. http://www.cs-faculty.stanford.edu/~koza/.

Lohn, J. D.; Colombano, S. P.; "**A Circuit representation technique for automated circuit design**". IEEE Transactions on Evolutionary Computation, Vol. 3, No. 3, september 1999, p. 205 – 219.

Lohn, J. D.; Haith, G. L. Colombano, S. P.; Stassinopoulos, D.; "**Towards evolving electronic circuits for autonomous space applications**". Aerospace Conference Proceedings, 2000 IEEE, Vol. 5, p. 473 – 486.

Lovay, M.; Arregui, A.; Gonella, J.; Peretti, G.; Lubaszewski, M; "**Fault Tolerant Amplifier Systms Using Evolvable Hardware**". Proceedings of the Argentine School of Micro-Nanoelectronics, Technology and Applications (EAMTA), 2010, p. 50 – 55.

Mesquita, A.; Salazar, F.A.; Canazio, P.P.; "**Chromossome representation through Adjacency Matrix in Evolutionary Circuits Synthesis**". Proc. 2002 NASA/DoD Conference on Evolvable Hardware, IEEE Computer press., July 2002, p. 102-109.

Nettleton, D.J.; Garigliano, R.; "**Evolutionary Algorithms and Dialogue**". In: Chambers, L. Pratical Handbook of Genetic Algorithms: New Frontiers, vol. 2. CRC Press , 1995.

Otero, A.; Salvador, R.; Mora, J.; Sekanina, L.; "**A Fast Reconfigurable 2D HW Core Architeture on FPGAs for Evolvable Self-Adaptive Systems**". 2011 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), 2011, p. 336 – 343.

Pacheco, M. A., "**Algoritmos Genéticos: Princípios e Aplicações**", ICA: Núcleo de Pesquisa em Inteligência Computacional Aplicada, 1999, PUC-RIO.

Poli, R.; Langdon, W. b.; McPhee, N. F.; "**A Field Guide to Genetic Programming**". Published via http://lulu.com and freely available at http://gp-field-guide.org.uk, 2008 (With contributions by J. R. Koza).GPbib. ISBN 978-1-4092-0073-4.

Ren Yu; Zhao Chengyao; "**Optimal PID Controller Design in PMSM Based on Improved Genetic Algorithm**". 2010 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), 2010, Vol. 2, p. 123 – 126.

Sá, L. B.; "**Síntese de Filtros com Baixa Sensibilidade utilizando Técnicas Evolutivas**". Rio de Janeiro, 2009, 121 p., Tese de Doutorado (COPPE – UFRJ).

Salazar, F. A.; Mesquita, A.; "**Synthesis of Analog Circuits Using Evolutionary Hardware**". Sixth Brazilian Simposium on Neural Networks, 2000, p. 101 – 106.

Santini, C. C.; "**Desenvolvimento de uma plataforma reconfigurável analogical para a evolução intrínseca de circuitos**". Rio de Janeiro, 2001, 107 p. Dissertação de Mestrado (Departamento de engenharia Elétrica - PUC-Rio).

Santini, C.; Zebulum, R.; Pacheco, M.; Vellasco, M.; Szwarcman, M.; "**PAMA - Programmable Analog Multiplexer Array**", The Third NASA/DoD Workshop on Evolvable Hardware, 2001, p. 36 – 42.

Santini, C.C.; Amaral, J.F.M.; Pacheco, M.; Vellasco, M.; Szwarcman, M.; "**Evolutionary Analog Circuit Design on a Programmable Analog Multiplexer Array**". Proc. of the IEEE International Conference on Field Programmable Technology (FPT), December, 2002, p. 189 – 196.

Santini, C.C., Amaral, J.F.M., Pacheco, M., Vellasco; Tanscheit, R.; "**Evolvability and Reconfigurability**". Proceedings of the IEEE International Conference on Field Programmable Technology (FPT), 2004, p. 105 – 112.

Sinohara, Helio Takahiro. "**Reparos e Ajustes Automáticos de Circuitos Eletrônicos Através de Eletrônica Evolucionária**". Rio de Janeiro, 2001, 84 p. Dissertação de Mestrado (Departamento de engenharia Elétrica - PUC-Rio).

Schultz, A. C.; Grefenstette, J. J.; De Jong, K. A.; "**Adaptive Testing of Controllers for Autonomous Vehicles**". Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology, July 1992, p. 158 – 164.

Sommerhage, Frank; "**PSpice binary import**". MathWorks File Exchangesite – Matlab Central. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/21452-pspice-binary-import/content/readdat.m>. Acesso em: 23 dez. 2011.

Stoica, A.; Zebulum, R.; Keymeulen, D.; "**Mixtrinsic Evolution**". Proceedings of the Third International Conference on Evolvable Systems (ICES '00). London, UK: Springer-Verlag, 2000, p. 208 – 217.

Stoica, A.; Andrei, R.; "**Adaptive and Evolvable Hardware – A Multifaceted Analysis**". 2007 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2007), 2007.

Stoica, A.; Keymeulen, D.; Mojarradi, M.; Zebulum, R.; Daud, T.; "**Progress in the Development of Field Programmable Analog Arrays for Space Applications**". Aerospace Conference, 2008, p. 1 – 9.

Szász, Cs.; Chindris, V.; "**Self-healing and Artificial Immune Properties Implementation upon FPGA-Based Embryonic Network**". 2010 IEEE International Conference on Automation Quality and Testing Robotics (AQTR), 2010, Vol. 2, p. 1 – 6.

Tang, K. S.; Man, K. F.; Kwong, S.; He, Q.; "**Genetic Algorithms and their Applications**". IEEE Signal Processing Magazine, Nov 1996, p. 22 – 37.

Tawdross, P.; König, A.; "**Mixtrinsic Multi-Objective Reconfiguration of Evolvable Sensor Electronics**". 2007 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2007), 2007.

Xiangzhong Meng; Baoye Song; "**Fast genetic algorithms used for PID parameter optimization**". Proceedings of the IEEE International Conference on Automation and Logistics, August 2007, p. 2144 – 2148.

Xin Yao; Higuchi, T.; "**Promisses and Challenges of Evolvable Hardware**". IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol. 29, No. , February 1999, p. 87 – 97.

Xilinx; "**Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite**". Disponível em: <http://www.xilinx.com/support/documentation/white_papers/wp374_Partial_Reconfig_Xilinx_FPGAs.pdf>. Acesso em: 18 jan 2012.

Zebulum, R.; Pacheco, M. A.; Vellasco, M.; "**Synthesis of CMOS operational amplifiers through Genetic algorithms**". Proceedings of the XI Brazilian Symposium on Integrated Circuit Design, 1998. p. 125 – 128.

Zebulum, R.; Pacheco, M. A.; Vellasco, M.; "**Comparison of different evolutionary methodologies applied to electronic filter design**". The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. p. 434 - 439.

Zebulum, R.; Pacheco, M. A.; Vellasco, M.; "**Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms**". Boca Raton, Florida: CRC PRess, 2001. ISBN 0-8493-0865-8.

Zebulum, R.; Mojarradi, M.; Stoica, A.; Keymeulen, D.; Daud, T.; "**Self-Reconfigurable Analog Arrays: Off-the Shelf Adaptive Electronics for Space Applications**". 2007 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2007), 2007, p. 529 – 563.