SÍNTESE TOPOLÓGICA EVOLUCIONÁRIA E OTIMIZAÇÃO NUMÉRICA DE PARÂMETROS EM ESTRUTURAS CMOS

Nilton Gavião Menezes

DISSERTAÇÂO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____
Prof. Antonio Carneiro de Mesquita Filho, Dr.d'État

_____
. Prof. José Franco Machado do Amaral, D. Sc.

_____
Prof. Jorge Lopes de Souza Leão, Dr.Ing.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2007

MENEZES, NILTON GAVIÃO

Síntese Topológica Evolucionária e Otimização Numérica de Parâmetros em Estruturas CMOS [Rio de Janeiro] 2007

IX, 59p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2007)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Algoritmo Evolucionário
2. Otimização Numérica

I. COPPE/UFRJ II. Título (série)

Dedicatória

À minha esposa Sônia Maria e minhas filhas Ana Beatriz e Ana Carolina.

Agradecimentos

- Ao Professor Dr. Antonio Carneiro de Mesquita Filho;
- Ao Instituto de Pesquisas da Marinha, especialmente ao Comandante CMG João Alberto Vianna Tavares;
- Ao meu amigo Felippe.
- Ao professor Dr. José Franco Machado do Amaral
- Ao professor Dr. Jorge Lopes de Souza Leão
- A minha família pelo incondicional apoio e compreensão.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SÍNTESE TOPOLÓGICA EVOLUCIONÁRIA E OTIMIZAÇÃO NUMÉRICA DE PARÂMETROS EM ESTRUTURAS CMOS

Nilton Gavião Menezes

Março/2007

Orientador: Antonio Carneiro de Mesquita Filho

Programa: Engenharia Elétrica

O projeto de circuitos baseados exclusivamente em transistores CMOS culmina no seu aspecto topológico e nas dimensões específicas dos parâmetros *W/L* de cada um dos transistores envolvidos. A utilização de algoritmos evolucionários para sínteses e otimização automáticas, normalmente, exige a codificação binária da topologia do circuito, em conjunto com os parâmetros *W/L* de cada transistor, gerando muitas vezes, estruturas de tamanho exageradamente grande. Circuitos de baixa complexidade podem necessitar de uma quantidade tão grande de bits para sua representação que o espaço de soluções pode exceder até o número estimado de átomos existentes no universo. A proposta em questão utiliza técnicas evolucionárias para a síntese topológica do circuito e numéricas para a otimização de seus parâmetros, numa tentativa de minimizar o esforço computacional necessário.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TOPOLOGIC EVOLUTIONARY SYNTHESIS AND NUMERICS PARAMETERS OPTIMIZATION FOR CMOS STRUCTURES

Nilton Gavião Menezes

March/2007

Advisor: Antonio Carneiro de Mesquita Filho

Program of: Electrical Engineering

The design of circuits based exclusively in CMOS transistors, culminates in the topological aspect and specific dimensions of the parameters *W*/*L* of each transistor involved. The use of evolutionary algorithms for automatic synthesis and optimization usually requires the binary encoding of the circuit topology and *W*/*L* parameters of each transistor, often creating extremely large chromosome structures. The representation of low-complexity circuits may require such a large number of bits that the solutions space dimensions may exceed the estimated number of atoms existing in the universe. In this work the evolutionary topological synthesis of CMOS-only circuits and numerical techniques for optimization of device parameters are proposed as an attempt to reduce the computing efforts needed.

.

Índice:

Índice de Figuras:

1. Introdução

Inspirada pelo princípio Darwiniano (Darwin, 1859) da evolução das espécies e na genética, a Computação Evolucionária tem procurado utilizar estes princípios na geração de novos métodos e algoritmos para a solução de diversos problemas da engenharia.

A Eletrônica Evolucionária (Higuchi,1999), área derivada da Computação Evolucionária, utiliza-se de métodos evolucionários para síntese automática de circuitos em estruturas programáveis. A pesquisa de novas arquiteturas reprogramáveis, especificamente voltadas à síntese evolucionária de circuitos, tem sido objeto de grande interesse nesta área (Stoica, 2000).

Analogamente ao que acontece com os seres vivos, o mecanismo de síntese / otimização em eletrônica evolucionária compreende indivíduos com codificação genética e aptidões específicas que se cruzam, gerando uma diversidade de novos indivíduos que em algumas situações apresentam alterações genéticas, comumente conhecidas por mutações.

As técnicas utilizadas para execução de tais tarefas em eletrônica evolucionária são os chamados algoritmos evolucionários (De Garis,1993).

O conceito de hardware evolucionário surgiu a partir da existência de plataformas reconfiguráveis de hardware como, por exemplo, o FPGA (Field Programmable Gate Array) (Thompson,1996; Miller, 1997), em conjunto com mecanismos de pesquisa, otimização e adaptação dos chamados algoritmos evolucionários. Suas implementações mais populares são os Algoritmos Genéticos (Holland,1975; Goldberg,1989) e a Programação Genética (Koza, 1992; De Garis,1992).

A utilização direta de uma plataforma de hardware programável (Sanches,1996; Zebulum, 1998) para avaliação do mérito de cada indivíduo,

caracterizado por um cromossomo, em um determinado algoritmo evolucionário é chamada de evolução intrínseca e sua estrutura pode ser visualizada na figura 1-1.

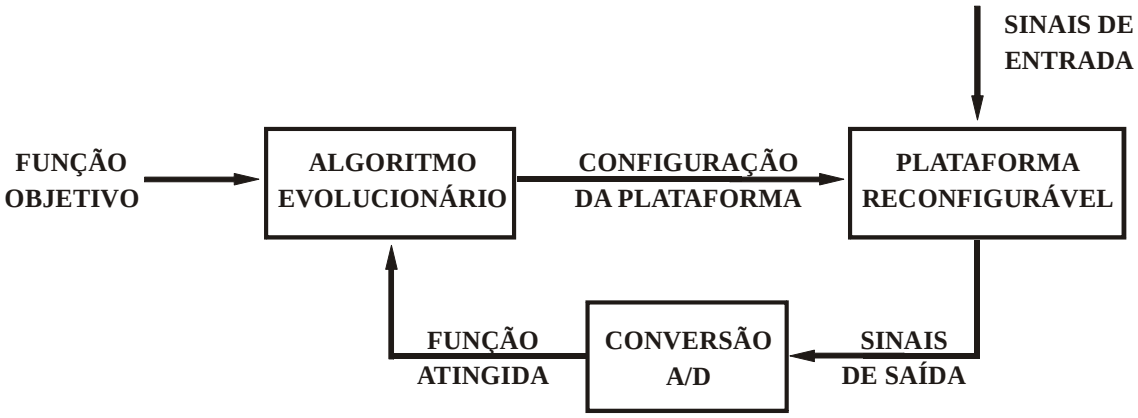

Figura 1-1 – Evolução Intrínseca

Quando em um algoritmo evolucionário, na ausência de uma plataforma de hardware reconfigurável, se utiliza um simulador de circuitos para avaliar o mérito de cada indivíduo/cromossomo executa-se a chamada evolução extrínseca e sua estrutura pode ser visualizada na figura 1-2.

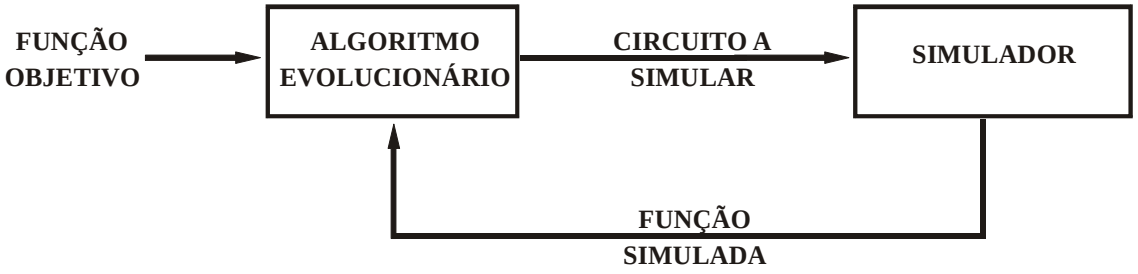

Figura 1-2 – Evolução Extrínseca.

A utilização de algoritmos evolucionários para síntese e otimização automáticas de circuitos baseados exclusivamente em transistores CMOS, normalmente exige a codificação binária da topologia do circuito, em conjunto com os parâmetros *W/L* de

cada transistor, gerando muitas vezes, estruturas de tamanho exageradamente grande. Circuitos de baixa complexidade podem necessitar de uma quantidade tão grande de bits para sua representação que o espaço de soluções pode exceder até o número estimado de átomos existentes no universo.

Pretende-se neste trabalho, encontrar novos mecanismos de síntese e otimização de circuitos eletrônicos, compostos exclusivamente de transistores CMOS, que possibilitem uma ampla varredura do espaço de soluções com o menor esforço computacional possível. Para desempenhar esta tarefa, uma nova representação cromossomial de topologia de circuitos eletrônicos é apresentada como alternativa às soluções ora existentes.

Este trabalho está organizado em cinco capítulos cujo conteúdo é relacionado a seguir.

O capítulo um, apresenta uma breve introdução definindo o objetivo e a estrutura do trabalho em si.

O capítulo dois fornece os subsídios necessários para os processos de síntese topológica de circuitos apresentando uma visão geral sobre algoritmos evolucionários, que fazem uso dos métodos evolucionários para desempenhar suas funções e suas respectivas variações, que vão desde o algoritmo genético tradicional (Holland,1975), sua variação STEADY-STATE (Linden, 2006), até os moldes da programação genética (Koza, 1992).

O capítulo três fornece subsídios para o desenvolvimento da tarefa de sintonia de parâmetros de transistores CMOS em circuitos com topologia conhecida, apresentando uma breve apresentação do que vem a ser otimização numérica e a explanação de duas técnicas amplamente conhecidas, “Downhill Simplex” (Nelder & Mead,1965) e “Particle Swarm Optimization (PSO)” (Kennedy & Eberhart,1995).

O capítulo quatro utiliza-se dos dois capítulos anteriores para dar forma ao sistema total, composto de um módulo de síntese topológica, onde uma nova representação cromossomial é proposta, e um módulo de otimização numérica de

parâmetros *W/L* com a finalidade de tornar o circuito evoluído, plenamente sintonizado. A implementação computacional destes processos, bem como, os parâmetros necessários à execução do sistema como um todo, finalizam este capítulo.

O capítulo cinco apresenta alguns resultados obtidos com o sistema desenvolvido.

O capítulo seis apresenta as conclusões e sugestões de trabalhos futuros.

2. Algoritmos Evolucionários

2.1. Introdução

Os algoritmos evolucionários surgiram quando John Henry Holland, então pesquisador da universidade de Michigan, inspirou-se na teoria da evolução das espécies de Darwin para criar um algoritmo matemático destinado a otimizar sistemas complexos. Holland estudou formalmente a evolução das espécies e propôs um modelo computacional que, quando implementado, poderia oferecer boas soluções para problemas extremamente difíceis que eram insolúveis computacionalmente, até aquela época (Holland,1975).

Os algoritmos evolucionários funcionam mantendo uma população de estruturas que evoluem de forma semelhante à evolução das espécies. A estas estruturas são aplicados os chamados operadores genéticos, tais como, cruzamento, (também conhecido como “crossover”) e mutação, entre outros (Zebulum,1999). Cada indivíduo recebe uma avaliação de seu mérito que representa a sua qualidade como solução do problema em questão e baseado nesta avaliação os operadores genéticos são aplicados de forma a simular o que ocorre na natureza. Indivíduos com maior aptidão tendem a sobreviver e gerar novos indivíduos que ocuparão o lugar de indivíduos menos aptos.

Um dos ramos mais importantes dos algoritmos evolucionários são os chamados algoritmos genéticos (Holland,1975; Goldberg,1989). Os algoritmos genéticos são especialmente utilizados em problemas de otimização numérica onde se deseja usualmente, encontrar mínimos ou máximos de funções.

2.2. Algoritmo Genético Tradicional

A estrutura de um algoritmo genético tradicional pode ser visualizada na figura 2.2-1.

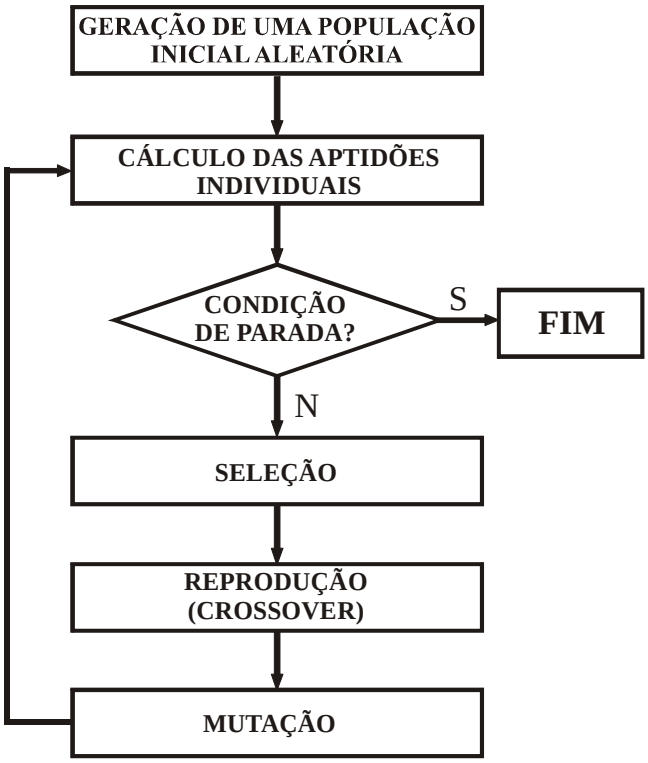

Figura 2.2-1 – Algoritmo Genético Tradicional.

Define-se primordialmente, a estrutura adequada de representação cromossomial de cada indivíduo, codificando-o binariamente. Esta é uma fase especialmente importante porque a eficiência do algoritmo genético depende de uma boa representação cromossomial.

Segue-se a esta fase, a geração de uma população inicial aleatória de indivíduos que serão avaliados um a um para verificação das suas aptidões individuais. Com base nas aptidões individuais, selecionam-se convenientemente, os indivíduos da população para em seguida, cruzá-los (cruzamento, figura 2.2-2), entre si, gerando descendentes com características herdadas exclusivamente de seus genitores. Indivíduos com maior aptidão terão maior chance de sobreviver e participar

de cruzamentos gerando novos indivíduos, enquanto os de menor aptidão terão maior chance de serem eliminados.

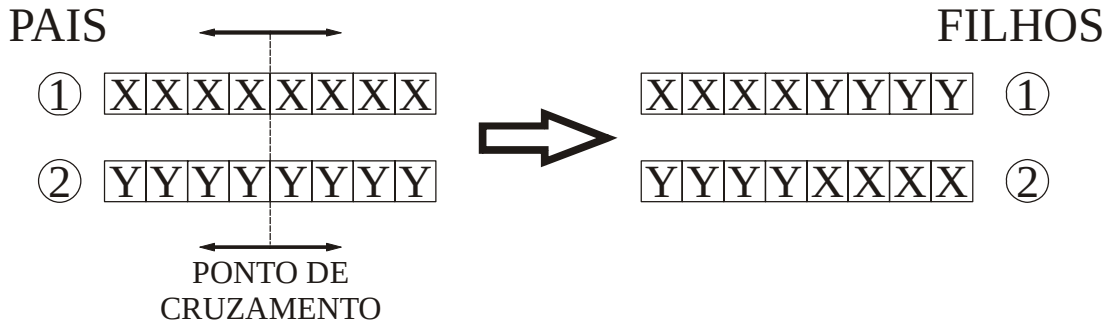

Figura 2.2-2 – Cruzamento

Tal como ocorre na natureza, ao ser gerado por cruzamento, uma porcentagem da população de novos indivíduos poderá sofrer mutações (fig. 2.2-3) que são, na verdade, pequenas alterações em seus códigos genéticos.

Figura 2.2-3 - Mutação

Este processo evolucionário, que descreve uma geração ou torneio, deve repetir-se até que seja encontrado um ou mais indivíduos que atendam às especificações de aptidão ou então seja atingido o número máximo de gerações ou torneios previamente determinados.

2.3. Algoritmo Genético STEADY-STATE

Uma vez que a cada torneio/geração, toda a população deve ser avaliada, a técnica tradicional anteriormente abordada apresenta-se pouco atraente quando a avaliação de aptidão individual exige um grande esforço computacional. Numa tentativa de minimizar tal esforço, uma variante dos algoritmos genéticos, denominada steady-state, tenta contornar este inconveniente, evoluindo e conseqüentemente avaliando, poucos indivíduos a cada torneio/geração e fazendo com que a população, evolua progressivamente para o objetivo desejado.

A simplicidade de implementação computacional desta variante dos algoritmos genéticos, aliada a sua eficiência, que por muitas vezes é superior ao modelo tradicional, torna-a bastante atraente.

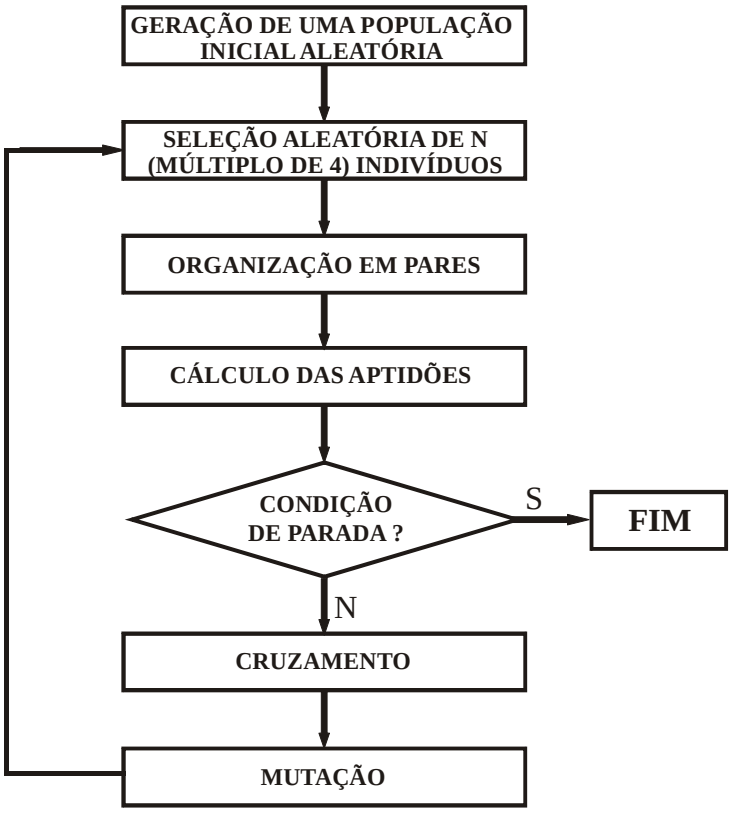

Figura 2.3-1 – Algoritmo Genético Steady-State

A figura 2.3-1 mostra o fluxograma de um algoritmo genético steady-state.

A partir de uma codificação cromossomial adequada, gera-se a população inicial que será evoluída, tal como no modelo tradicional. No entanto, antes de dar início ao processo evolucionário, há que se escolher adequadamente o número de indivíduos que participarão de cada torneio.

Deve-se ter em mente que os indivíduos serão sempre separados em pares. Após a avaliação de ambos os indivíduos do par serão revelados, o melhor indivíduo do par, que será preservado na população e participará na geração de um novo indivíduo através da operação de cruzamento, e o pior indivíduo do par, que será irremediavelmente descartado. Para que haja cruzamento, existe a necessidade da existência de outro par com as mesmas propriedades do anterior de modo que os dois melhores indivíduos entre os dois pares se cruzem, gerando dois novos indivíduos que ocuparão os lugares dos piores indivíduos dos dois pares. Desta forma, o número de indivíduos participantes de cada torneio deve ser sempre um múltiplo de quatro (dois pares). Usualmente, apenas quatro indivíduos participam de cada torneio, uma vez que não existe comprovação irrefutável de melhoria de desempenho no aumento de indivíduos por torneio STEADY-STATE (Linden,2006).

A figura 2.3-2 Ilustra a seleção, o cruzamento e a mutação (quando houver) de um torneio STEADY-STATE.

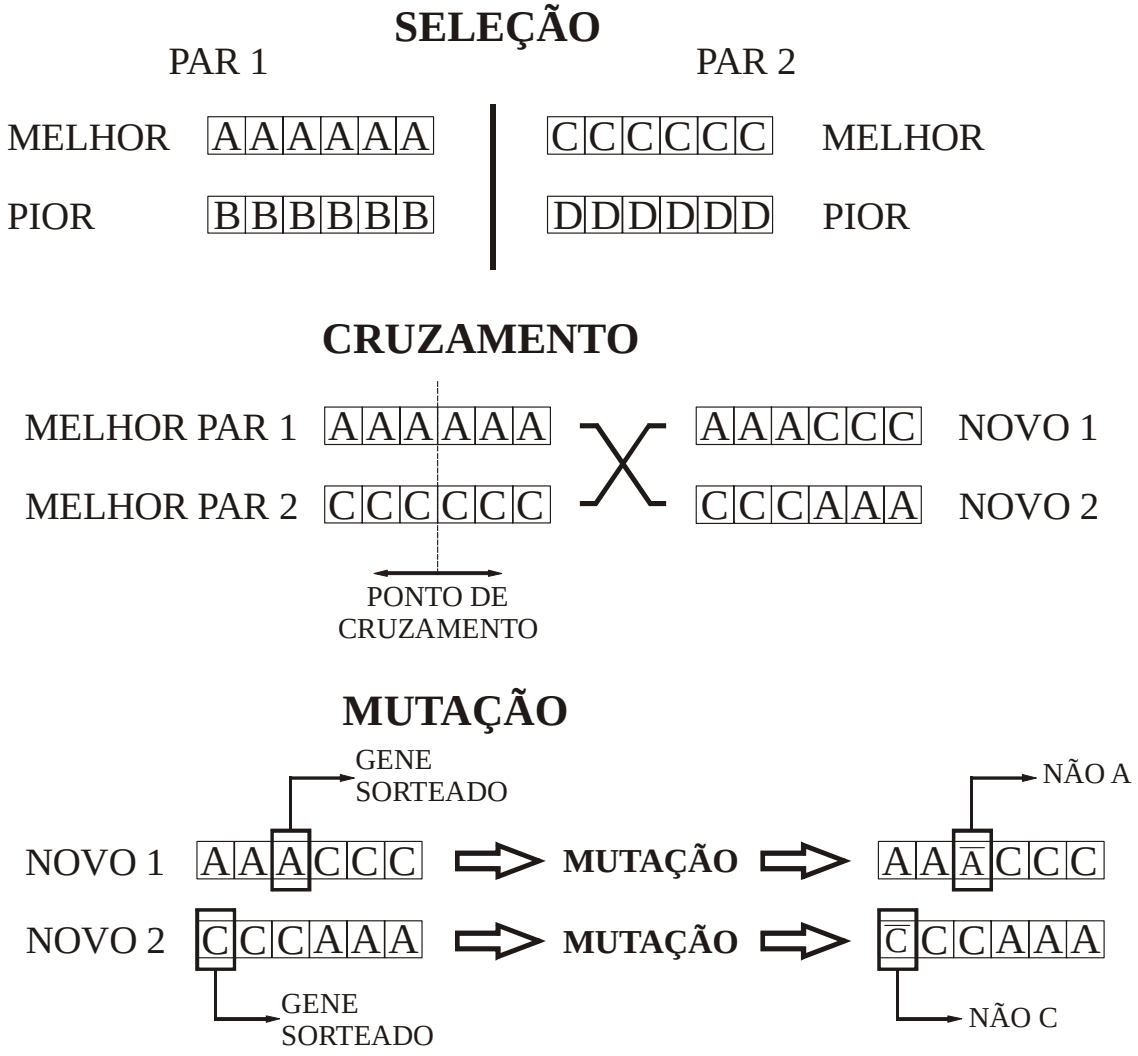

Figura 2.3-2 – Detalhe da seleção, cruzamento e mutação de um torneio

2.4. Programação Genética

Em 1992, John Koza surgiu com uma proposta que estendia a utilização dos algoritmos genéticos para aplicações em que seria possível representar estruturas de tamanho variável, com o objetivo de representar e evoluir programas de computadores. Em lugar de um cromossomo simples, codificado binariamente e de tamanho fixo, foram criadas árvores de construção sintática onde os programas são representados. Estas árvores são constituídas de dois tipos de representação nodal distintas, onde os vértices internos das árvores representam funções dos seguintes tipos:

- operações aritméticas (soma, subtração, etc.);

- funções matemáticas (sin, cos, etc.);

- funções booleanas (E, OU, etc.);

- operadores condicionais (if... then... else);

- operadores de loop (DO... WHILE).

Os demais nós, que são os vértices externos, representam números constantes ou variáveis das funções. Como exemplo, a expressão $\sqrt{x^3 + y^2}$ pode ser representada pela árvore representada na figura 2.4-1 (Zebulum, 2002).

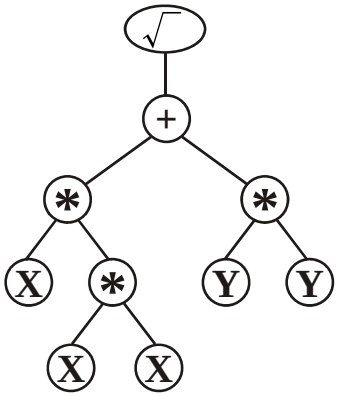

Figura 2.4-1 – Representação da árvore $\sqrt{x^3 + y^2}$

As técnicas de manipulação e a aplicação dos operadores dos algoritmos genéticos continuam, com certas nuances, sendo válidos em programação genética. A utilização de programação genética para a solução de problemas, via de regra, envolve os seguintes passos:

- a determinação dos tipos de terminais necessários;

- a determinação dos tipos de funções envolvidas;

- a determinação da forma de avaliação dos indivíduos;

- estabelecimento dos parâmetros de controle necessários à execução;

- definição dos critérios de término de execução do processo.

Como se pode perceber, são passos similares aos utilizados em algoritmos genéticos, exceto pelo fato de haver terminais e funções a serem definidas. Um torneio

de programação genética pode ser observado na figura 2.6-2, onde os “strings” de representação binária são substituídos por árvores sintáticas.

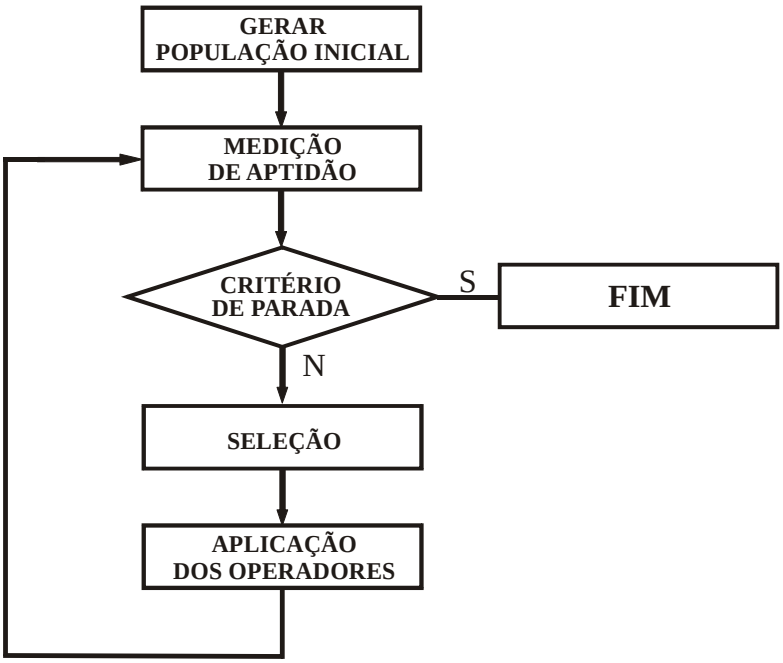

Figura 2.6-2 - Torneio de programação genética

Finalizando, o fluxo normal de programação genética apresenta-se como a seguir.

População Inicial: são geradas árvores aleatórias, de sintaxe válida (funções e terminais adequadamente posicionados) e de tamanho máximo definido a priori

Medição da aptidão: tal como em algoritmos genéticos, um valor escalar deve representar o quão apto um indivíduo é, segundo uma determinada especificação funcional.

Seleção: probabilística favorecendo os mais aptos como nos algoritmos genéticos

Operadores: além dos conhecidos operadores de cruzamento e mutação dos algoritmos genéticos, existem outros operadores tais como permutação, edição,

deleção, encapsulamento e destruição, descritos em detalhe por Koza (1992) e que serão omitidos por não serem do escopo deste trabalho.

Critério de parada: Como nos algoritmos genéticos, definido quando atingido o número máximo de iterações ou descoberto um indivíduo que satisfaça as especificações do projeto.

A codificação de topologias de circuitos em cromossomos, bem como a aplicação de algoritmos evolucionários para sua síntese, normalmente, obedece a premissas que as classificam no âmbito da programação genética, em lugar dos algoritmos genéticos (Zebulum, 2002).

3. Métodos de Otimização Numérica

3.1. Introdução

A busca por soluções de problemas relacionados a determinadas funções matemáticas, cuja aplicação de conceitos puramente analíticos seja ineficiente ou exija um esforço exagerado, sempre motivou o homem a procurar ferramentas alternativas para contornar ou tornar mais brando o esforço necessário. Tarefas, como encontrar raízes, máximos ou mínimos de funções, podem tornar-se árduas e até mesmo impossíveis, se levadas a cabo analiticamente. Os métodos numéricos, como alternativa, apresentavam-se igualmente insatisfatórios uma vez que, ao homem é extremamente custosa a realização de operações repetitivas.

O surgimento de máquinas capazes de executar, de forma rápida, operações repetitivas, motivou um grande investimento na busca de métodos numéricos cada vez mais eficientes e precisos. Do pós-guerra até meados de 1965 foram produzidos excelentes trabalhos nesta área até que John A. Nelder e Roger Mead desenvolveram um método denominado “Downhill Simplex” (Nelder & Mead,1965) que de forma simples e hábil, era capaz de encontrar rapidamente mínimos de funções, sem o prévio conhecimento de suas derivadas ou outras funções matemáticas “sofisticadas”, utilizando-se apenas da avaliação do valor da função em determinados pontos Este método, embora rápido e preciso, quando aplicado a funções não contínuas ou que apresentem mínimos locais e global distintos, apresenta o inconveniente de não garantir o encontro de mínimos globais, uma vez que sua procura limita-se a encontrar o caminho por onde a função decresce, a partir de um determinado ponto inicial. Sua eficiência diminui à medida que o número de dimensões aumenta.

Por volta de 1995, James Kennedy e Russell Eberhart desenvolveram um método de otimização denominado “Particle Swarm” (Kennedy & Eberhart,1995), inspirados segundo eles, no comportamento social de pássaros em bando ou de peixes em cardume. Esta técnica compartilha diversas similaridades com métodos

evolucionários computacionais tais como os algoritmos genéticos e são capazes de pesquisar e encontrar tanto mínimos locais quanto globais. Como desvantagem em relação ao método “Downhill Simplex”, o esforço computacional é consideravelmente maior, tendo em vista, a necessidade da avaliação da totalidade da população a cada iteração do algoritmo.

Para este trabalho onde um dos principais objetivos reside em otimizar os parâmetros W’s e L’s de transistores de determinados circuitos, ambos os métodos foram implementados com o objetivo, de avaliar o comportamento da função que os define.

3.2. Downhill Simplex (ou Nelder-Mead)

O método Downhill Simplex, como diz o próprio nome, otimiza funções dentro de um espaço chamado “Simplex”, definido como uma figura geométrica de *N* dimensões, *N+1* vértices com todos os seus segmentos de reta, faces de polígonos e etc. Em duas dimensões, um simplex é definido por um triângulo e em três dimensões um tetraedro, não necessariamente regular.

Para dar início ao processo de otimização é necessário escolher um ponto de partida. O algoritmo então deverá percorrer um caminho de descida (downhill) através de uma topografia *N*-dimensional até encontrar um mínimo local ou até mesmo global. O método deve começar não apenas com um simples ponto, mas com *N+1* pontos que definem o “simplex” inicial. Se existir um ponto inicial P0 então os outros *N* pontos podem ser expressos pela equação 3.2-1(Wright,1995):

$$\mathbf{P}_i = \mathbf{P}_0 + a_i \, \mathbf{e}_i \qquad (3.2\text{-}1)$$

Onde $\mathbf{e}_i$ são os *N* vetores unitários e a_i são constantes que caracterizam o tamanho escalar de cada um dos vetores que definem o “simplex”.

O método então, promove uma série de passos que, em sua maioria, movem o mais alto ponto (onde o valor da função é mais alto), através da face oposta do

“simplex” para um suposto ponto de valor mais baixo. Estes passos são chamados reflexão, expansão, contração para dentro / para fora e encolhimento global.

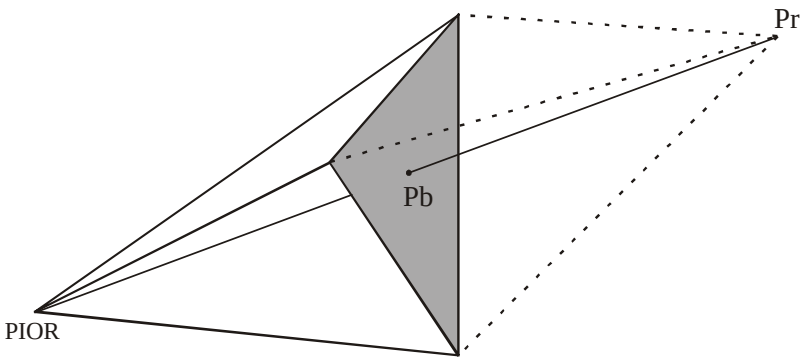

Figura 3.2-1 – Reflexão

O primeiro passo do algoritmo é determinar matematicamente o ponto **Pr** que é uma reflexão do pior ponto do simplex passando pelo ponto **Pb**, que é a média dos pontos do “simplex” exceto o pior ponto conforme pode ser observado na figura 3.2-1. Quando o valor da função em **Pr** está entre o segundo pior ponto e o melhor ponto do simplex, o pior ponto é então substituído pelo ponto refletido **Pr**. O novo “simplex” aparece definido pelas linhas tracejadas na figura 3.2-1.

Quando o valor da função no ponto **Pr** é igual ou menor que o valor do menor (melhor) ponto do “simplex”, ou seja, uma melhor estimação do mínimo, o valor da função no ponto **Pe** é checado para ver se a função decresce na direção de **Pr.**

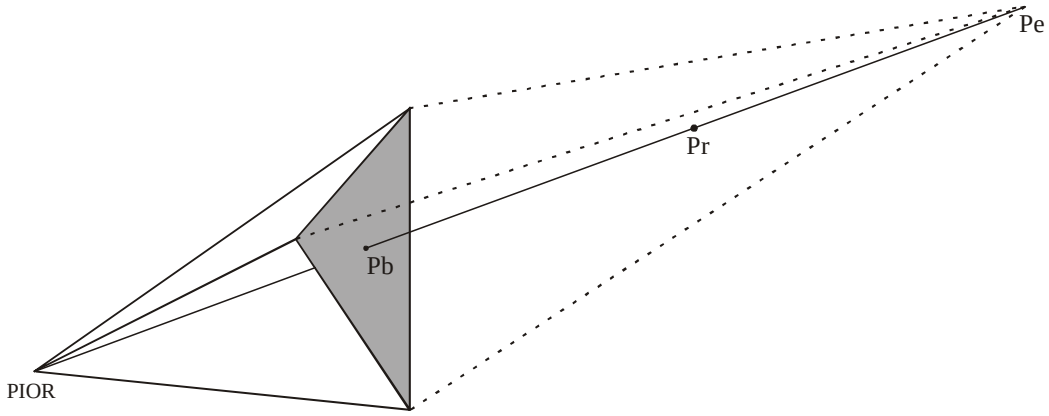

Figura 3.2-2 – Expansão

O melhor dos pontos, **Pr** ou **Pe**, substitui o pior ponto do simplex.

Quando o valor da função no ponto **Pr** é maior ou igual ao valor do pior ponto, o valor do ponto **Pcd** deve ser checado para saber se a função é menor entre o pior ponto e o ponto **Pb**.

Quando o valor da função no ponto **Pr** é maior ou igual ao valor do segundo pior ponto e menor que o valor do pior ponto do “simplex”, o valor da função em **Pcf** é checado para saber se a função é menor entre os pontos **Pr** e **Pb**.

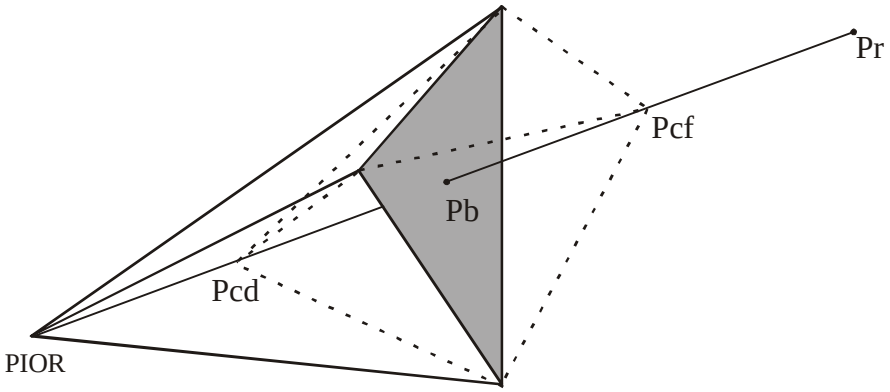

Figura 3.2-3 – Contração para dentro ou para fora

De acordo com o valor da função nos pontos **Pcd** e **Pcf** em relação ao pior ponto do simplex, o melhor (menor valor) entre **Pcd** e **Pcf** substitui o pior ponto. Foi efetuada a contração para dentro ou para fora.

Quando o valor da função nos pontos **Pcd** e **Pcf** é maior que o maior (pior) valor do “simplex”, procede-se então a operação de encolhimento global que nada mais é que contrair todo o “simplex” na direção do ponto de menor valor (melhor ponto).

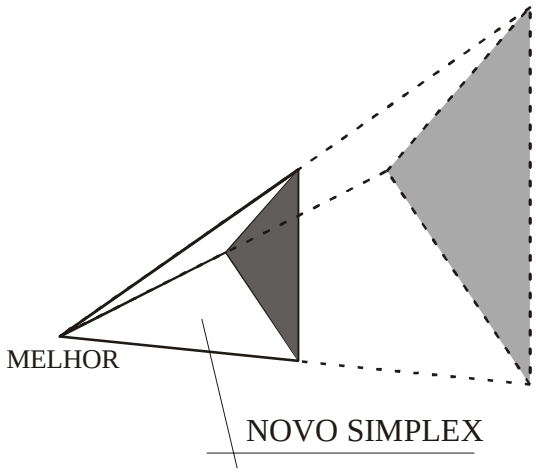

Figura 3.2-4 – Encolhimento Global

À medida que o número de dimensões aumenta, a eficiência do algoritmo degrada devido à necessidade do grande número de avaliações da função a cada operação de encolhimento global. A esta altura, outras técnicas de otimização tornam-se uma alternativa interessante.

3.3. Particle Swarm (PSO)

Introduzido por James Kennedy e Russel Eberhart em 1995, o algoritmo "Particle Swarm Optimization (PSO)" surgiu de experiências com algoritmos modelados a partir da observação do comportamento social de determinadas espécies de pássaros (Reynolds,1987, Heppner & Grenander, 1990). As partículas se comportam como os pássaros à procura de alimento ou do local de seus ninhos, utilizando o aprendizado próprio e o aprendizado do bando. O PSO é composto de partículas representadas por vetores que definem a velocidade atual de cada partícula e de vetores de localização, atualizados segundo sua velocidade atual, seu aprendizado pessoal e o aprendizado adquirido pelo bando (Prado & Saramago, 2005).

A figura 3.3-1 mostra o fluxograma de execução do algoritmo PSO.

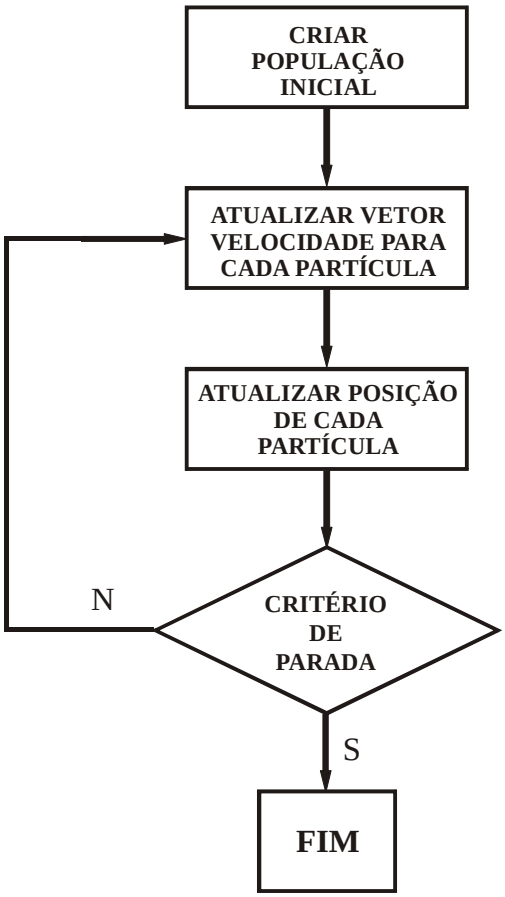

Figura 3.3-1 – Fluxograma do algoritmo PSO

O primeiro passo do algoritmo é gerar as *N* partículas que formarão o “swarm” ou o “enxame” com suas respectivas posições. Pode-se também neste momento, arbitrar velocidades iniciais para cada partícula.

O algoritmo manter-se-á ativo, atualizando os vetores de velocidade e posição ciclicamente até que seja atingido qualquer critério de parada (número máximo de iterações atingido, partícula com aptidão desejada, etc.).

O vetor velocidade de cada partícula deve ser atualizado pela equação 3.3-1 (Prado & Saramago, 2005).

$$v_{k+1}^{i} = w.v_{k}^{i} + c1.r1.(p^{i} - x_{k}^{i}) + c2.r2.(p_{k}^{s} - x_{k}^{i}) \qquad (3.3\text{-}1)$$

Onde:

v_k^i é a velocidade atual da partícula

p^i é a melhor posição encontrada pela partícula *i*

p_k^s é a melhor posição dentre todas as partículas na iteração *k*

w é um parâmetro que representa a inércia da partícula e controla a sua capacidade de exploração do espaço de soluções. Um valor alto determina uma busca global enquanto um valor baixo determina uma busca local. Usualmente estes valores oscilam entre 0,7 e 1,4.

c1 e c2 são os chamados parâmetros de confiança e definem o quanto uma partícula confia em si (*c1*) ou no bando (*c2*). Usualmente ambos assumem o valor igual a dois.

r1 e r2 são números aleatórios compreendidos entre zero e um.

Para o cálculo da posição futura de cada partícula no algoritmo é utilizada a equação 3.3-2 (Prado & Saramago, 2005).

$$x_{K+i}^i = x_k^i + v_{k+1}^i \tag{3.3-2}$$

Onde:

x_{K+i}^i é a posição de cada partícula *i* na iteração *k*+1

v_{k+1}^i é o vetor velocidade da partícula

Embora seja classificado como um algoritmo evolucionário, pode-se perceber que determinadas características presentes nos algoritmos genéticos, tais como, a sobrevivência do mais apto ou a utilização de operadores do tipo cruzamento ou mutação, não estão presentes neste método de otimização.

4. A Síntese e Otimização

Com o auxílio das técnicas, brevemente descritas nos capítulos 2 e 3 e uma adequada representação cromossomial para descrição de circuitos desenvolvida para esta aplicação, o sistema final com todos os seus módulos, é descrito neste capítulo.

4.1 Topologia

Como ponto de partida para o desenvolvimento deste trabalho, foi definido que o circuito a ser evoluído seria composto exclusivamente de transistores CMOS, uma vez que tais estruturas podem representar virtualmente, qualquer tipo de componente eletrônico. Definiu-se ainda, que a síntese topológica, evolucionária, seria executada separadamente da otimização numérica dos parâmetros W's e L's que dimensionam os transistores e que ambos os processos, completamente independentes, deveriam interagir convenientemente, de forma a gerar circuitos plenamente sintonizados.

As representações sintáticas de construção de circuitos ora existentes (Mesquita et al, 2002), embora bastante eficientes, despertaram grande interesse na criação de uma nova estrutura com maior facilidade de aplicação dos operadores genéticos. Como alternativa, a representação cromossomial a ser utilizada para execução deste trabalho deveria apresentar entre outras, as seguintes características:

- permitir a aplicação de operadores genéticos com pouco ou nenhum cuidado especial;

- permitir a representação de circuitos de tamanho variável;

- conservar as características topológicas de cada uma das partes do cromossomo, quando seccionado, para aplicação do operador de cruzamento;

- a aplicação do operador de mutação em um ou mais genes de determinado cromossomo deveria ser executada de forma simples e direta, sem cuidados especiais.

Desta forma, a implementação do algoritmo evolucionário para a síntese de circuitos poderia ser desenvolvida de forma simples e segura.

4.2 Parâmetros W e L

A otimização dos parâmetros dimensionais de cada transístor em um circuito é sem dúvida, de suma importância. Circuitos com excelentes topologias podem apresentar-se completamente insatisfatórios sem uma correta sintonia de seus parâmetros. A execução da sintonia de um determinado circuito pode, sem prejuízo, ser executada a partir da definição de sua topologia, quando então, poder-se-á determinar a aptidão específica do indivíduo/circuito.

Esta otimização, puramente numérica, poderia perfeitamente ser executada por um algoritmo genético, porém, como a geração da topologia utiliza-se de uma técnica similar, optou-se por experimentar outras tecnologias igualmente eficientes e até mesmo mais rápidas sob certos aspectos computacionais.

4.3 Representação Cromossomial

A representação cromossomial criada para esta aplicação, assemelha-se ao aspecto convencional do netlist padrão SPICE, que representa muito bem a estrutura topológica de circuitos elétricos, mas apresenta-se sabidamente insatisfatório para utilização direta em sínteses evolucionárias. Uma adequada representação cromossomial deve permitir operações de cruzamento e mutação de forma simples e confiável, além de assegurar que as informações de qualquer parte do circuito permaneçam preservadas após qualquer operação.

Como, neste caso em particular, o circuito a ser evoluído possui apenas transistores do tipo CMOS do tipo N ou tipo P e uma vez que tais estruturas podem representar, virtualmente, qualquer tipo de componente discreto usual (resistores, capacitores, indutores e etc), cada elemento ou transístor será representado por uma estrutura de 3 terminais com o aspecto da figura 4.3-1.

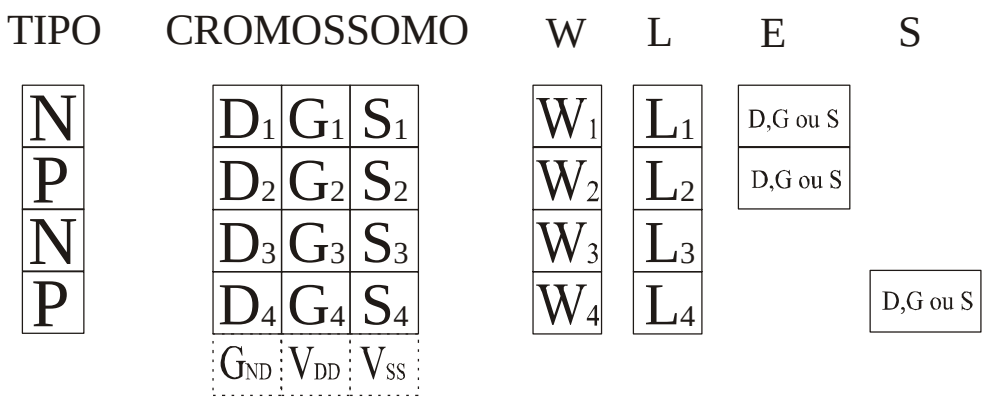

Figura 4.3-1 – Representação Cromossomial Completa

Tipo é representado por um caracter N ou P e define se o transistor em questão é do tipo NMOS ou PMOS. O chamado **CROMOSSOMO**, que representa a topologia do circuito, é uma lista de números inteiros onde os elementos representam os terminais (Dx, Gx e Sx) de cada transistor. Cada terminal contém um valor que determina quantos terminais a frente o terminal atual está ligado. Por exemplo, se o terminal G1 (gate do transistor 1) estiver ligado ao terminal D2 (dreno do transistor 2), então em G1 haverá o número 2, representando serem necessários dois passos a frente para alcançar o dreno do transistor 2. Um valor zero indica que o terminal atual não está ligado a nenhum terminal posterior, por exemplo, se na ligação descrita anteriormente o dreno do transistor 2 não estiver ligado a mais nenhum terminal, então ele conterá o valor zero, indicando o fim do nó.

Gnd, VDD e VSS não são na verdade, representados e servem apenas para referência de distância numérica quando o terminal em questão estiver ligado a qualquer ponto de alimentação. Como exemplo, se o terminal D3 estiver ligado a Vdd, ele terá o valor sete que é o número de passos necessários para chegar a Vdd.

W e **L** são números reais contendo as dimensões dos parâmetros em questão.

A entrada principal do circuito, representada por **E**, onde será aplicada a fonte de sinal, é sempre um terminal qualquer do primeiro transistor do circuito (Dreno, Gate

ou Source). A entrada secundária, utilizada apenas para sínteses com excitação diferencial, é sempre um terminal qualquer do segundo transistor e a saída do circuito, Representada por **S**, onde a carga deve ser ligada, é sempre um terminal qualquer do último transistor do circuito.

O terminal BULK de cada transistor não é representado, assumindo-se assim que para transistores do tipo P ele será sempre ligado a Vdd e para transistores do tipo N ele será sempre ligado a Vss quando a síntese utilizar fontes simétricas ou a Gnd quando a síntese utilizar fonte de alimentação simples.

Ilustrando melhor, o circuito da figura 4.3-2 tem sua representação cromossomial mostrada na figura 4.3-3.

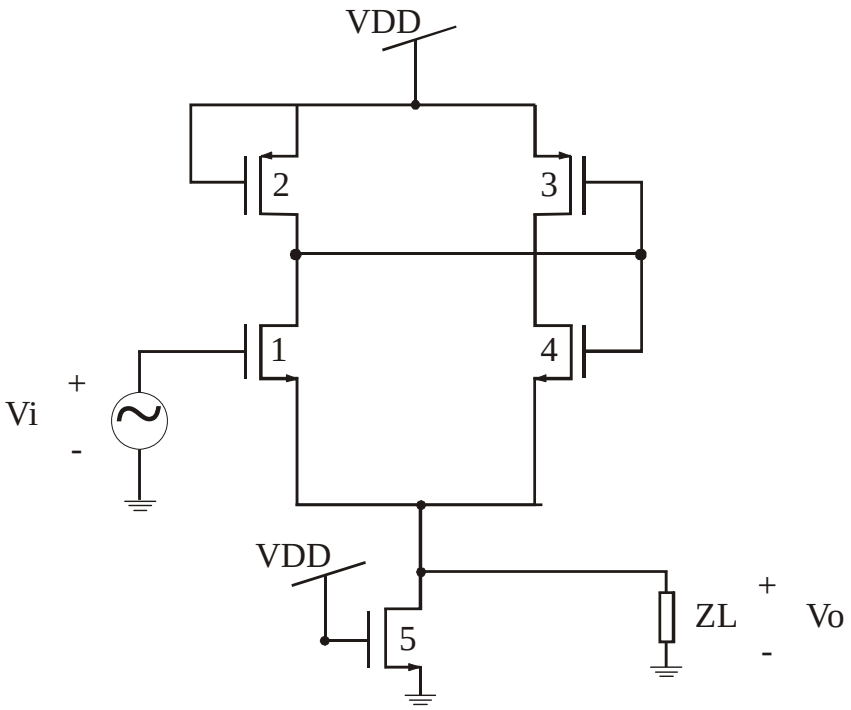

Figura 4.3-2 – Circuito CMOS

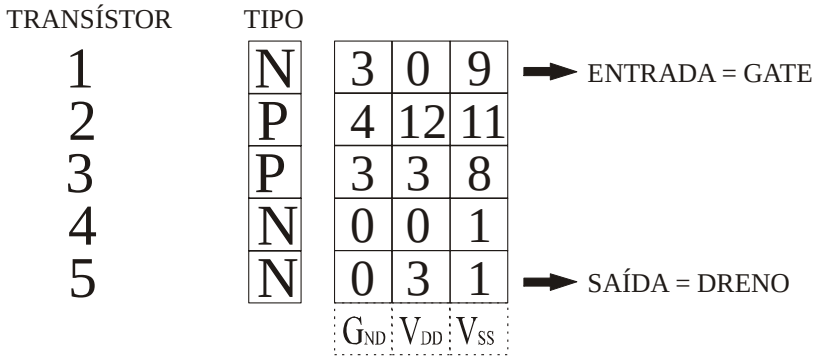

Figura 4.3-3 – Representação cromossomial do circuito da fig. 4.2-2

O transistor onde a fonte de sinal é ligada deve ser sempre o primeiro a ser representado e o transistor onde será aplicada a carga deve ser sempre o último. Detalhando, a construção da tabela cromossomial, parte do transistor 1 começando pelo seu Dreno, que tem o valor 3 significando que este terminal está ligado ao terminal Dreno do transistor 2 (Três casa a frente). O valor 4 no Dreno do transistor 2 significa que ele está conectado ao terminal Gate do transistor 3 que por sua vez está ligado ao terminal Gate do transistor 4 devido ao seu valor 3. O valor 0 no terminal Gate do transistor 4 significa que não há mais ligações neste nó. Perceba que as ligações de um determinado nó, partem sempre do transistor menos significativo (menor valor de referência) em direção aos mais significativos (maior valor de referência). O valor 0 no Gate do Transistor 1 significa que ele não está ligado a nenhum outro terminal no interior do circuito, somente a fonte de sinal é aplicada nele. O valor 12 no Gate do transistor 2, excede o tamanho do cromossomo em duas unidades significando que está conectado a Vdd.

Como se pode notar, qualquer circuito pode ser representado como uma seqüência de números inteiros.

4.3.1. Regras na operação de cruzamento

A operação de cruzamento, quando executada com cromossomos de mesmo tamanho e mesmo ponto de cruzamento, gerando novos indivíduos com os mesmos tamanhos dos pais, não exige qualquer cuidado adicional. Basta definir o ponto de corte, que pode ser em qualquer posição de ambos os cromossomos genitores e executar o cruzamento, gerando os dois novos cromossomos filhos da forma tradicional, ilustrados na figura 4.3.1-2. Todos os valores de Dn, Gn e Sn serão automaticamente válidos.

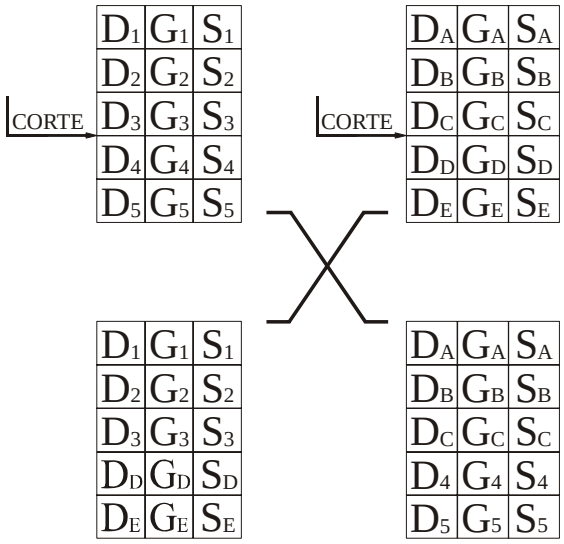

Figura 4.3.1-1 – Cruzamento em cromossomos de mesmo tamanho e mesmo ponto de corte

A operação de cruzamento em cromossomos de tamanhos diferentes exige um certo cuidado adicional, tendo em vista, a possibilidade de ocorrência de referências inválidas nos seus cromossomos filhos, observe o exemplo da figura 4.3.1-2

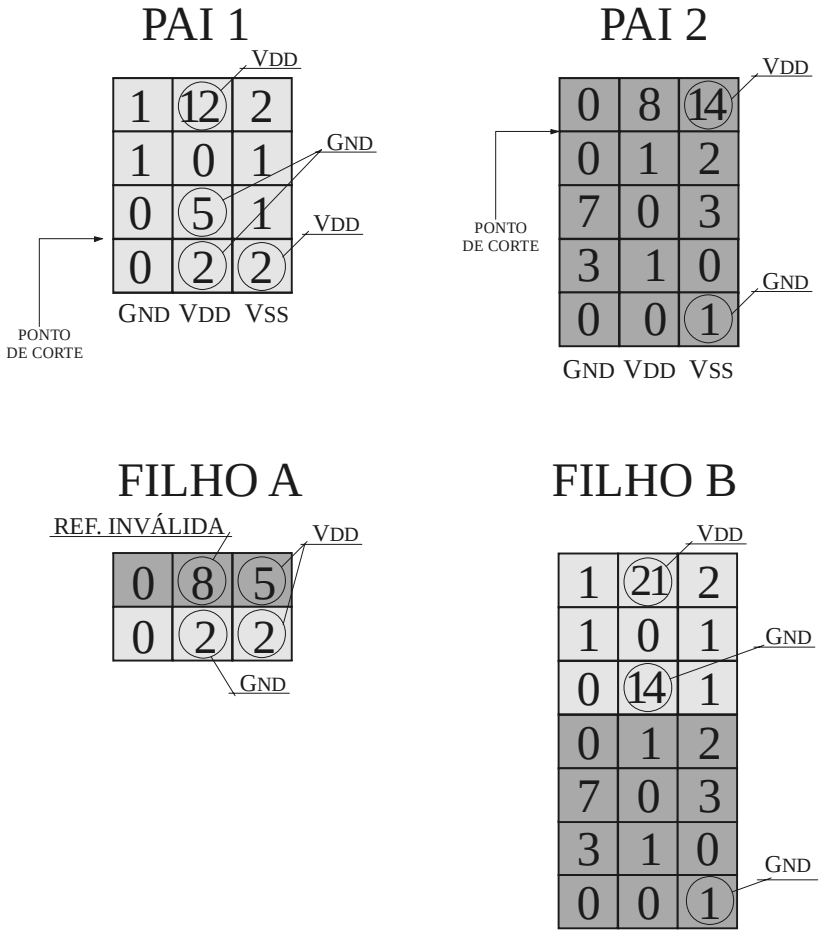

Figura 4.3.1-2 – Cruzamento em cromossomos de tamanhos diferentes em pontos diferentes

Como ponto de partida, há que se identificar os terminais ligados à alimentação (Gnd, Vdd ou Vss) nos cromossomos pais de forma a transferi-los convenientemente aos seus filhos gerados, garantindo assim, que os terminais ligados à alimentação de qualquer transistor dos circuitos, mantenham-se conservados.

O próximo e último passo é identificar os terminais que após o cruzamento, apresentem referências inválidas, ou seja, que contenham um ponteiro para um terminal inexistente, localizado além do fim do próprio cromossomo, excetuando-se os de alimentação, previamente ajustados. O cromossomo *Filho A,* da figura 4.3.1-2, apresenta em seu terminal *G* do transistor 1, um exemplo de terminal com ligação inválida. Para corrigir tais anomalias, basta que se reduza os referidos ponteiros de três unidades repetitivamente, até que seja atingido um terminal válido no interior do

circuito. Vale ressaltar que os terminais “fantasmas” de alimentação, não fazem parte do circuito, são apenas referências e não devem ser considerados nesta operação de redução. O cromossomo filho A, passaria a ter o aspecto da figura 4.3.1-3.

Figura 4.3.1-3 – Cromossomo filho A após o ajuste dos terminais inválidos

A operação de mutação pode ser executada em qualquer terminal de um cromossomo filho, em qualquer quantidade, tomando-se apenas o cuidado de não gerar referências inválidas.

4.4. Implementação do projeto

Os módulos independentes de síntese topológica e otimização paramétrica, foram desenvolvidos em Visual Basic Versão 6.0, utilizando-se tecnologia de soquete (winsock - Microsoft) para comunicação entre os processos de síntese e otimização. Basicamente, a estrutura do projeto envolve dois módulos com finalidades diferentes. Um módulo chamado *módulo principal* que tem a finalidade de executar as seguintes tarefas:

- gerar/controlar os diversos parâmetros da síntese evolucionária e da otimização numérica dos parâmetros;

- gerar convenientemente topologias aleatórias com parâmetros *W e L*, também aleatórios;

- executar a síntese topológica baseada em técnicas evolucionárias;

- ativar / controlar a execução dos diversos módulos de otimização, avaliando comparativamente as aptidões dos circuitos de acordo com o mérito desejado e apresentando gráfica e interativamente os resultados atingidos durante a execução de todo o processo de síntese / otimização.

Para otimização dos parâmetros de cada circuito evoluído, foi desenvolvido um segundo módulo, chamado *módulo otimizador*, que executa a tarefa de avaliar e otimizar iterativamente, os circuitos provenientes do módulo principal. Os dados topológicos juntamente com os parâmetros W, L e de controle da otimização a ser executada, são recebidos via soquete (rede) e submetidos ao método desejado (Downhill-Simplex ou PSO). Para avaliação eletrônica da aptidão do circuito sendo otimizado, o módulo otimizador necessita externamente de um programa simulador de circuitos SPICE, que deve ser instalado e configurado previamente nos computadores onde os *módulos de otimização* estiverem sendo executados. Ao término do processo de otimização, o *módulo otimizador* devolve ao *módulo principal*, também via soquete,

o resultado alcançado pelo processo de otimização desejado, de acordo com as premissas contidas nos parâmetros de controle recebidos.

Os circuitos a serem sintetizados são representados por suas curvas de transferência DC e AC desejadas, descritas em arquivos texto que são lidos pelo *módulo principal* implementado para calcular a aptidão de cada indivíduo na população.

A técnica evolucionária utilizada para a síntese topológica do circuito é do tipo STEADY-STATE, onde a cada torneio, quatro indivíduos da população são escolhidos aleatoriamente, separados em dois pares de forma que o melhor indivíduo de cada par será utilizado para geração de dois novos indivíduos que tomarão os lugares dos piores indivíduos dos pares.

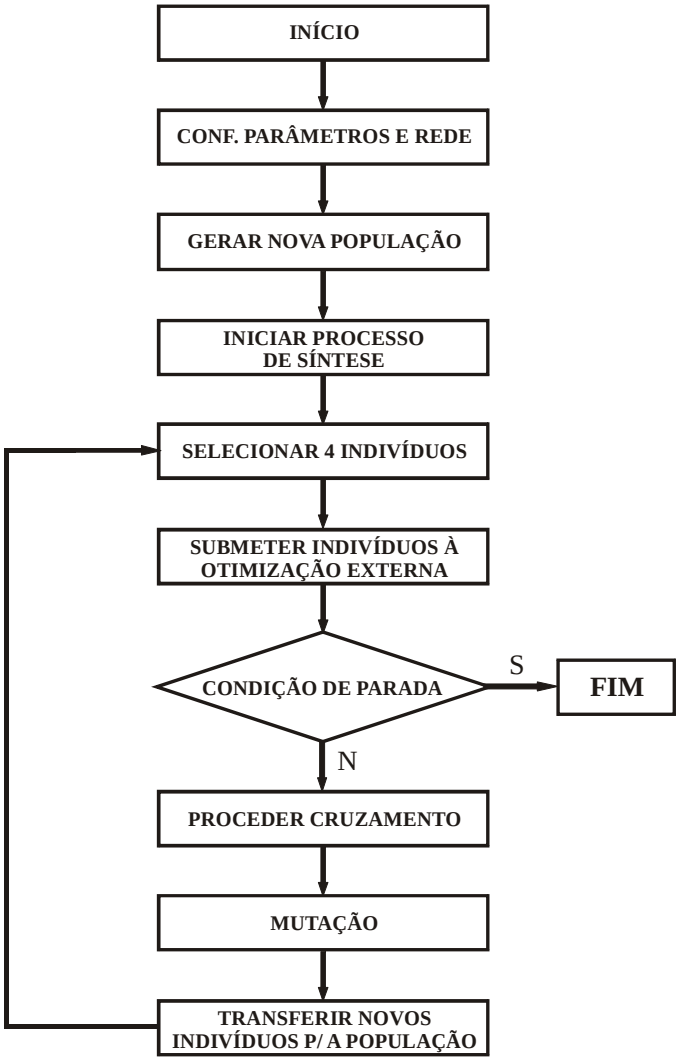

Figura 4.4-1 Fluxograma do módulo principal

Por conveniência, indivíduos não avaliados terão como aptidão, o valor “menos um” (-1). Um valor de aptidão nulo representará um indivíduo defeituoso, por exemplo, com o terminal de entrada ligado a uma das alimentações do circuito. Estes indivíduos merecem especial atenção e seu tratamento será discutido adiante. Valores de aptidão no intervalo (0,1], representarão a aptidão relativa do indivíduo válido.

Além da definição das curvas DC e AC desejadas, há que se configurar adequadamente outros parâmetros para a síntese, relacionados a seguir.

Configuração de rede

Sem módulo otimizador não há como executar a síntese, uma vez que a simulação eletrônica dos indivíduos/circuitos é feita exclusivamente pelos processos otimizadores. Deve-se, portanto, conectar o módulo de síntese a pelo menos um módulo de otimização disponível. Os endereços IP dos nós otimizadores disponíveis devem ser preenchidos no espaço reservado para tal, clicando-se em seguida no botão abaixo do mesmo. Quando uma conexão for bem sucedida, um ícone representando um computador aparecerá indicando que o nó foi conectado com sucesso.

Uma porta aleatória local entre 6000 e 6999 será alocada ao soquete no momento da sua conexão, portanto, há que se ter cuidado na reserva de tal espaço no sistema operacional.

Vale observar que no mesmo computador onde a síntese topológica será executada, pode haver também, uma instância do módulo otimizador e que isto não afetará significativamente, a performance do sistema, uma vez que, o torneio STEADY-STATE precisa aguardar o término de todas as quatro avaliações / otimizações para poder prosseguir seu fluxo.

A figura 4.4-2 apresenta a tela de configuração de rede do módulo principal.

Figura 4.4-2 Tela de configuração de rede do módulo principal

Parâmetros da Síntese / Otimização

Para a execução dos processos de síntese e otimização, existem parâmetros que devem ser preenchidos a “priori” para então, dar início a estes processos. A figura 4.4-3 apresenta o aspecto da tela de configuração de parâmetros.

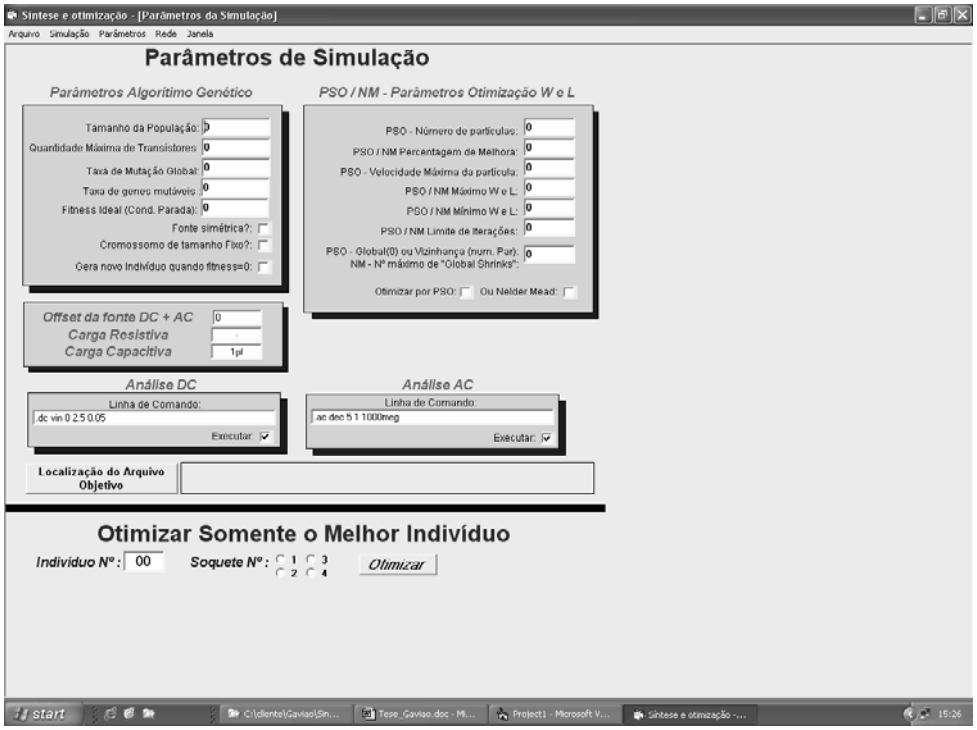

Figura 4.4-3 - Tela de configuração dos parâmetros

Parâmetros Algoritmo Genético

Tamanho da população: define o número de indivíduos/circuitos que participarão dos torneios de síntese e consequentemente otimização.

Quantidade máxima de transistores: define o número máximo de transistores que cada um dos indivíduos/circuitos poderá ter.

Taxa de mutação global: define em porcentagem, a probabilidade de um indivíduo gerado através de cruzamento, sofrer mutação.

Taxa de genes mutáveis: define em porcentagem, a quantidade de genes/terminais que serão mutados em uma operação de mutação.

Fitness Ideal (condição de parada): Um número real entre zero e um que define o fim do processo de síntese/otimização.

Fonte Simétrica: Quando habilitado, indica se os indivíduos da população poderão conter terminais ligados a uma fonte de alimentação negativa (VSS), além da positiva (VDD).

Cromossomo de tamanho fixo: quando habilitado, todos os indivíduos/circuitos terão o número de transistores indicado em "Quantidade máxima de transistores" explanado anteriormente.

Gera novo indivíduo quando fitness=0: define se um indivíduo com aptidão zero, por exemplo, sem ligação à fonte de alimentação, será descartado e substituído por um novo, gerado aleatoriamente ou se ele permanecerá inválido na população.

PSO / NM - Parâmetros Otimização W e L

PSO – Número de partículas: Define para o(s) módulo(s) de otimização, quando configurado(s) para utilizar PSO, a quantidade total de partículas que serão geradas aleatoriamente, além da originária recebida do módulo principal. Este parâmetro não é utilizado quando a otimização for NM.

PSO/NM – Porcentagem de melhora: define em porcentagem, a melhora mínima desejada do indivíduo a ser otimizado. Quando alcançada, a otimização será interrompida e o indivíduo devolvido imediatamente, com seus novos valores, ao

módulo principal. Um valor zero (0) neste parâmetro, força a execução da rotina de otimização até ser atingido o número máximo de iterações previamente configurado. Caso não haja melhora, o indivíduo original será devolvido com seus valores originais. Este parâmetro é utilizado por PSO e NM.

PSO – Velocidade máxima da partícula: Define a velocidade máxima da partícula PSO durante a execução da rotina de otimização. Este parâmetro não é utilizado quando a otimização for NM.

PSO/NM – Máximo W e L: define o valor máximo que os parâmetros W e L de cada transistor do circuito podem assumir durante a otimização. Este parâmetro é utilizado por PSO e NM.

PSO/NM – Mínimo W e L: define o valor mínimo que os parâmetros W e L de cada transistor do circuito podem assumir durante a otimização. Este parâmetro é utilizado por PSO e NM.

PSO/NM – Limite de iterações: define o número máximo de iterações que podem ocorrer para tentar otimizar um determinado indivíduo. Este parâmetro é utilizado por PSO e NM.

PSO - Global(0) ou Vizinhança (num. Par) NM - Nº máximo de "Global Shrinks": Quando PSO, define o tipo de abordagem para o desenvolvimento dos enxames PSO. Quando NM, define a quantidade máxima de operações do tipo “global shrink” que poderá ocorrer durante a execução da rotina NM.

Otimizar por PSO / Otimizar por NM: Define o tipo de otimização a ser executada. A execução de ambos simultaneamente não é permitida, porém, caso nenhuma opção seja marcada, o módulo otimizador limitar-se-á a avaliar a aptidão do indivíduo sem executar qualquer otimização.

Demais Parâmetros de configuração

Offset da fonte DC+AC; Carga Resistiva e Carga Capacitiva: Parâmetros utilizados para a simulação elétrica do circuito pelo(s) módulo(s) de otimização.

Análise DC e Análise AC: Quando habilitadas, utilizam as linhas de comando preenchidas para execução da análise DC e/ou AC na simulação elétrica do circuito. A medida de aptidão de cada indivíduo levará em conta o(s) tipo(s) de análise(s) desejada(s). Ao menos uma das análises deve ser habilitada sempre.

Localização do arquivo objetivo: local e nome do arquivo que contém a descrição da(s) curva(s) de transferência DC e/ou AC desejada(s).

A seção onde aparece o título "***Otimizar somente o melhor indivíduo***" pode ser utilizada quando, durante a execução da síntese automática, um determinado indivíduo da população desperte especial interesse na execução apurada de sua sintonia (otimização dos parâmetros W's e L's), sem alteração da sua respectiva topologia. A otimização desejada será executada utilizando-se os parâmetros de configuração definidos nesta própria página, devendo-se escolher "a priori" o número do indivíduo na população e o soquete do módulo otimizador desejado. Deve-se interromper o módulo de síntese automática para utilização deste recurso.

Controle da simulação

Após a definição dos parâmetros desejados para síntese / otimização descritos anteriormente, deve-se preferencialmente, salvá-las através do menu "***Arquivo***". Para dar início ao processo de síntese / otimização, deve-se abrir o menu "***Simulação***". A figura 4.4-4 mostra o aspecto da janela que se abrirá.

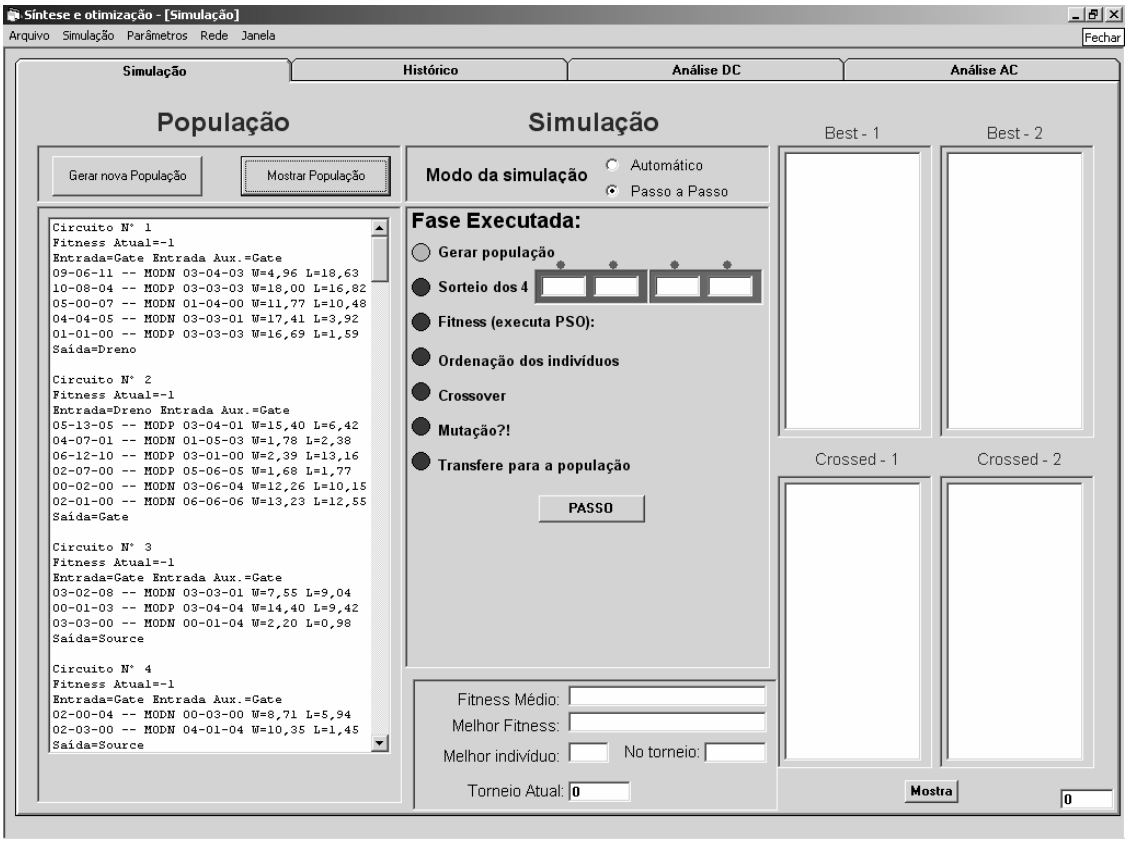

Figura 4.4-4 - Janela Simulação e GUIA Simulação

A janela, apresentada acima, possui quatro "*guias*" descritas como:

Simulação

Antes de iniciar o processo, é necessário gerar uma nova população de indivíduos com as características definidas anteriormente nos parâmetros do sistema, clicando no botão "***Gerar nova população***". Este comando atua como um "*reset*" de todo o processo de síntese / otimização, descartando todo e qualquer dado ou variável do processo anterior.

O botão "***Mostrar População***" transfere para o "*listbox*" localizado a esquerda da janela, os detalhes atualizados de todos os indivíduos presentes na população. Este recurso pode ser utilizado a qualquer momento, não interferindo no andamento do programa.

Sempre que houver pelo menos um módulo otimizador conectado, surgirão as opções de execução automática ou passo a passo, localizadas no centro e ao alto da tela, bem como o botão denominado "***Passo***" que ao ser clicado muda sequencialmente o estado atual do processo de síntese / otimização. Pode-se acompanhar os diversos estados através do "led" verde que se acende ao lado de cada estado.

Os "*listboxes*" localizados a direita em conjunto com o botão "***Mostrar***" servem para a verificação da correta execução dos operadores de cruzamento e mutação, se for o caso, nos indivíduos de um determinado torneio. Foram usados para depuração durante o desenvolvimento e não influenciam na execução do processo.

Histórico

Esta guia apresenta a evolução gráfica das aptidões média e do melhor indivíduo do processo de síntese sendo executado. Pode ser atualizado automaticamente ou conforme desejado bastando “clicar” na opção desejada. Seu aspecto pode ser visualizado na figura 4.4-5.

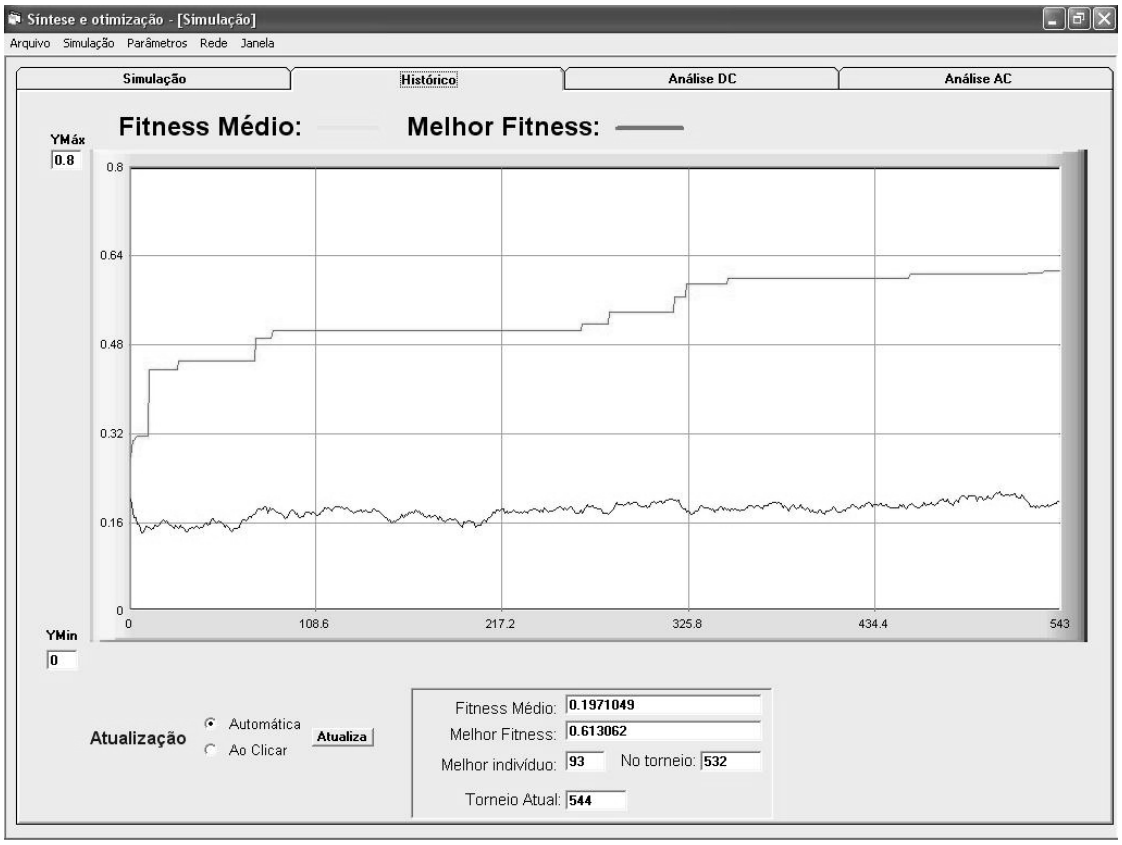

Figura 4.4-5 - Janela Simulação e GUIA Histórico

Análise DC

Esta guia apresenta graficamente as curvas de transferência DC desejada e a melhor atingida até o momento. A figura 4.4-6 mostra o aspecto de ambas.

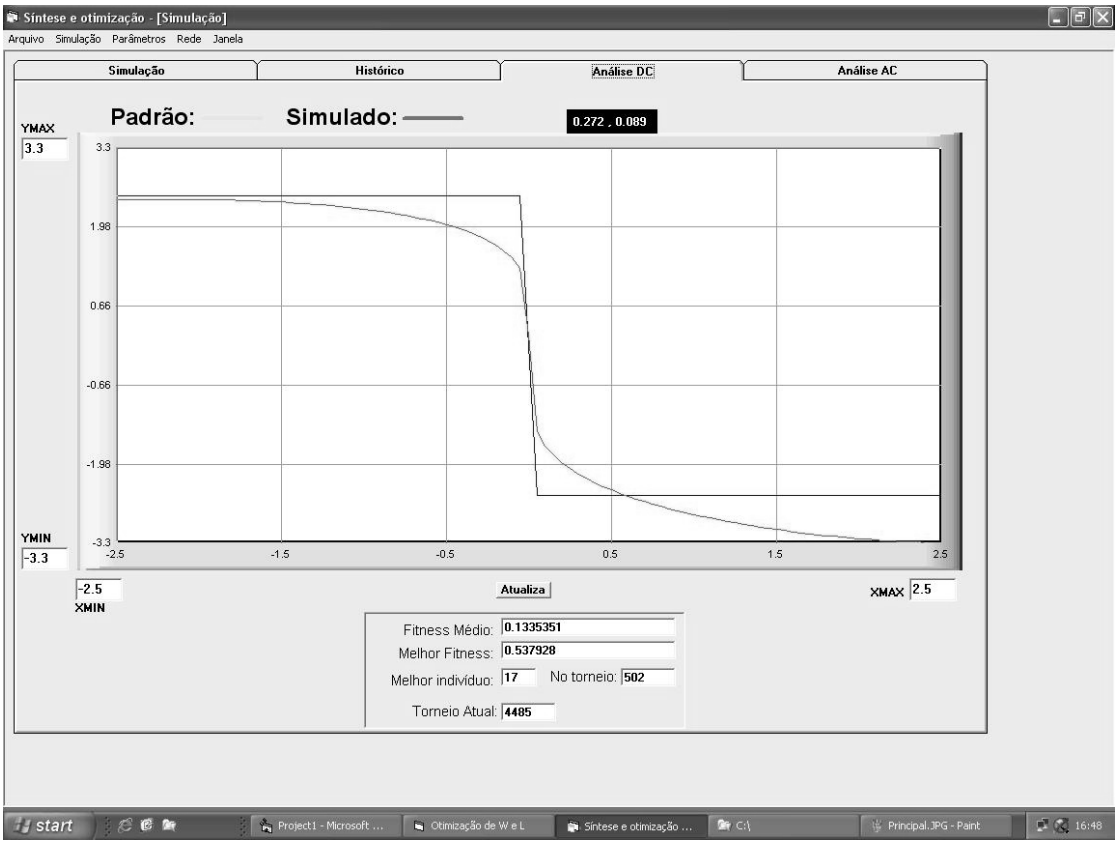

Figura 4.4-6 – Janela de Simulação e GUIA Análise DC

Análise AC

Esta guia apresenta graficamente as curvas de transferência AC desejada e a melhor atingida até o momento. A figura 4.4-7 mostra o aspecto de ambas

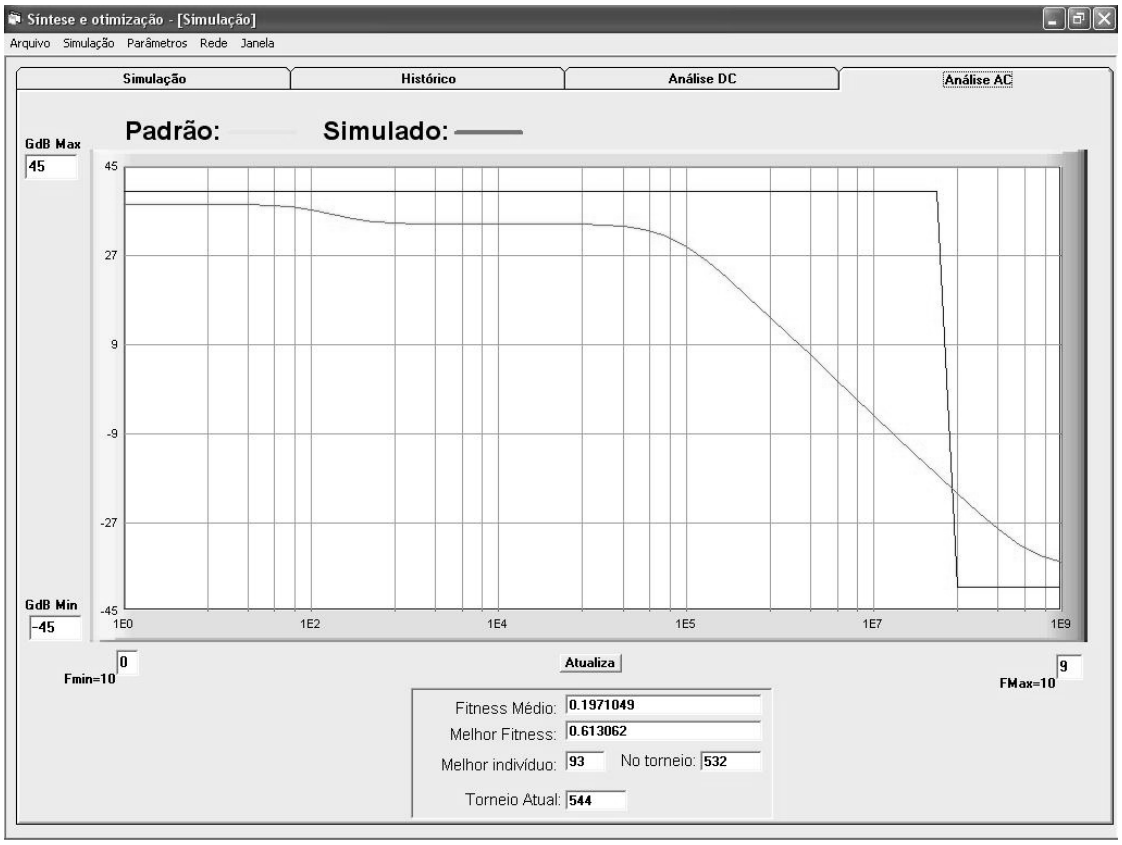

Figura 4.4-7 – Janela de Simulação e GUIA Análise AC

4.5. O Módulo de Otimização

O programa desenvolvido para executar a otimização numérica de parâmetros W e L de circuitos CMOS tem como característica fundamental, ser totalmente controlado por um programa externo e o fluxo de sua execução pode ser visualizado na figura 4.5-1.

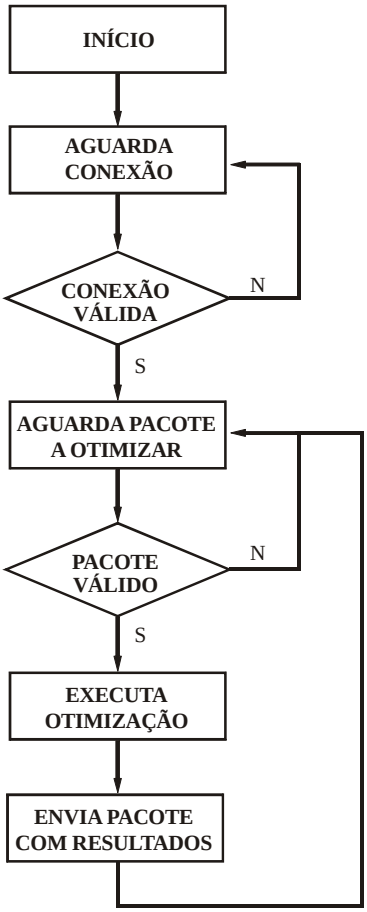

Figura 4.5-1 - Fluxograma do módulo de otimização

Ao ser iniciado, o módulo entra no estado de espera aguardando um pedido de conexão. Seu aspecto pode ser visto na figura 4.3.3-2 onde aparece o seu endereço IP que deve ser anotado para configuração de ambiente de rede, no módulo que executa a síntese topológica.

O módulo de otimização permanecerá neste estado até que haja um pedido de conexão externa e que a negociação de conexão seja bem sucedida.

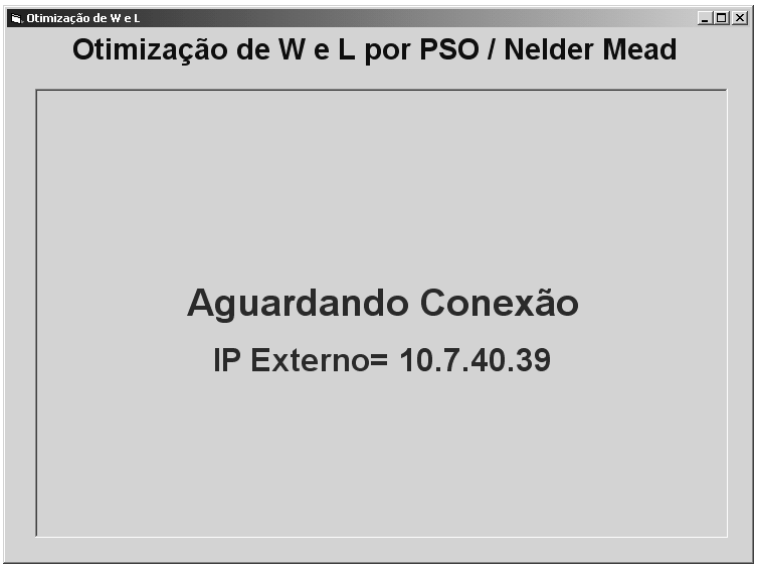

Figura 4.5-2 - Aspecto inicial do módulo de otimização

Ao conectar-se ao programa externo, o módulo de otimização muda de estado, passando a ter o aspecto da figura 4.5-3. A partir deste momento, o soquete de comunicação entra em espera aguardando a chegada de um pacote específico contendo dados com a topologia do circuito a ser otimizado, o valor atual de aptidão do circuito, os parâmetros W's e L's a serem otimizados e o tipo de otimização desejada entre outros parâmetros necessários à execução da rotina.

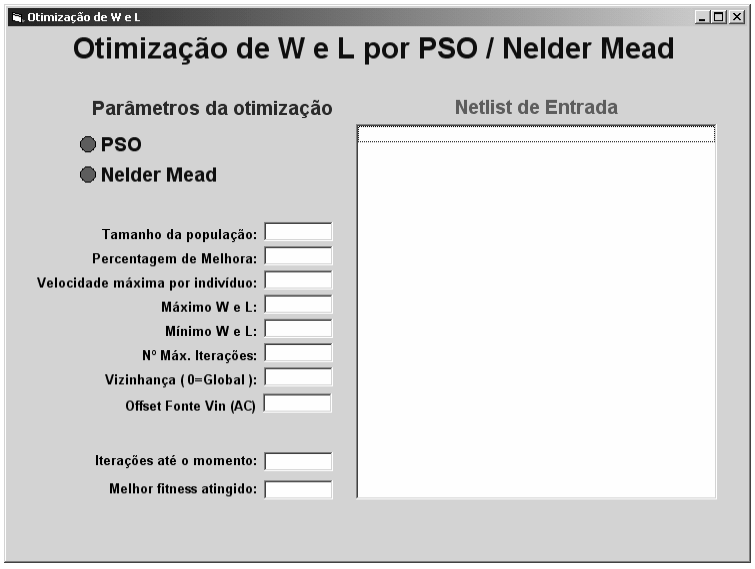

Figura 4.5-3 - Módulo de otimização conectado com sucesso

Ao receber um pacote válido, o programa entra no estado de execução, onde a otimização definida no pacote recém-recebido é processada. Este pacote contém os seguintes dados:

- parâmetros relativos à execução da otimização desejada;

- os dados topológicos do circuito a ser otimizado com seus parâmetros *W/L* ;

- o valor atual de aptidão do circuito que será utilizado como meta a ser atingida pelo método de otimização desejado;

- os dados do arquivo texto contendo os objetivos das curvas de transferência DC e AC a serem atingidos..

Os indivíduos gerados pela rotina de otimização, ora chamados de *partículas* para que não sejam confundidos com os cromossomos de topologia, têm sua origem de acordo com o tipo de otimização solicitada.

A rotina de otimização tem início e a cada iteração do método numérico desejado, o indivíduo com melhor valor de aptidão conseguida é revelado.

Na otimização por Downhill-Simplex (ou Nelder-Mead), a distribuição das partículas utilizadas para execução da otimização é feita gerando-se uma partícula extra para cada dimensão da partícula recebida, acrescida de um valor aleatório. Entenda-se por dimensão, cada um dos parâmetros *W* ou *L* do circuito proveniente do módulo principal. Estas partículas em número de *n+1, onde n* é o número de dimensões da partícula recebida.

Na otimização por Particle Swarm (PSO), a distribuição das partículas se dá aleatoriamente e em número determinado pelos parâmetros recebidos.

O cálculo da aptidão de cada partícula durante a execução da otimização, é feito baseando-se nos dados do arquivo objetivo recebido do módulo principal e os dados do arquivo gerado pelo simulador de circuitos associado a cada partícula, segundo a média da soma das raízes das diferenças dos quadrados de cada ponto e normalizado para um valor entre 0 e 1. Este cálculo é realizado para cada uma das análises desejadas (somente DC, somente AC ou DC e AC).

O módulo permanece neste estado, não aceitando pedidos pendentes, até que seja terminado todo o processo de otimização. A figura 4.5-4 mostra um exemplo do momento em que o estado de execução chega ao fim.

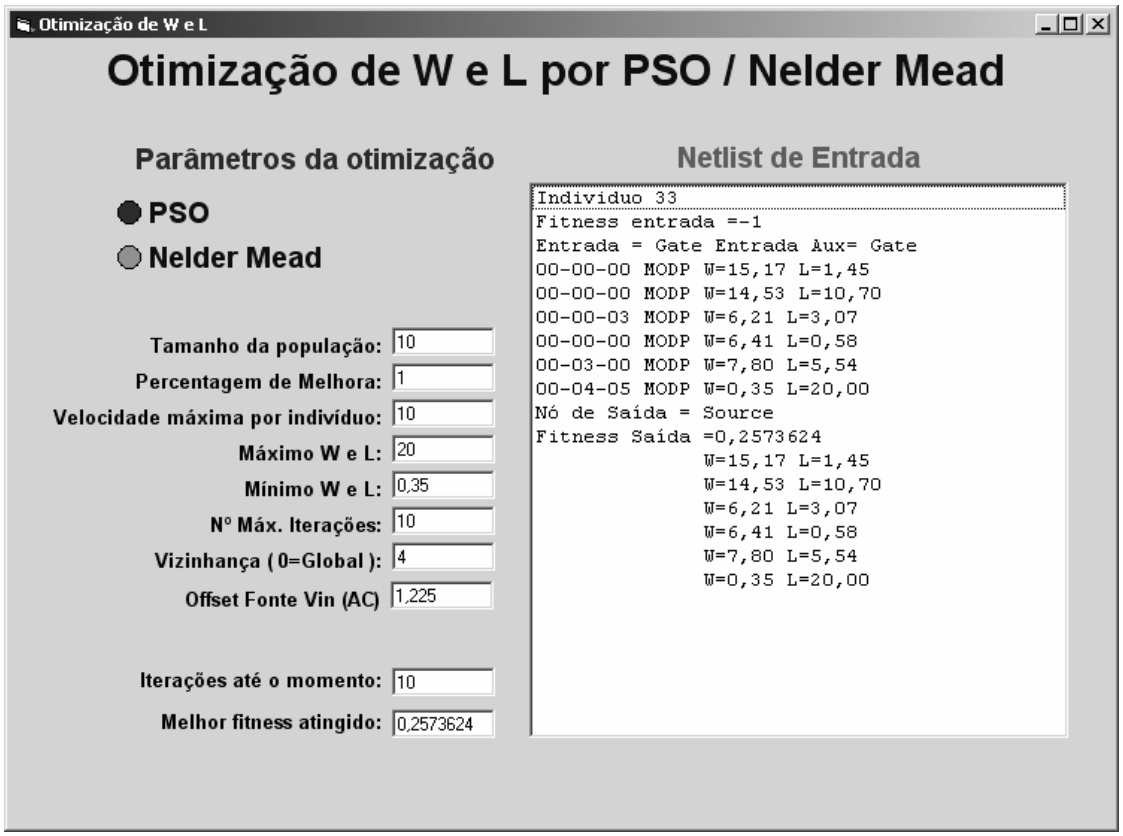

Figura 4.5-4 - Final da execução de uma rotina de otimização

Após o estado de execução o módulo entra finalmente no estado de envio do pacote de resultados, devolvendo ao módulo principal, os parâmetros otimizados e o valor de aptidão alcançado pela rotina de otimização e retornando ao estado de espera de pacote a otimizar.

5. Resultados obtidos

Para validação de qualquer trabalho desenvolvido é mister que sejam obtidos resultados que proporcionem subsídios para a correta avaliação do seu conteúdo. Este capítulo apresentará alguns resultados obtidos a partir de determinados circuitos eletrônicos de características popularmente conhecidas.

Como condição principal de parada, o tempo médio de uma hora e meia foi arbitrado como limite para execução dos processos de síntese e otimização. A aptidão padrão desejada foi configurada em um (1), ou seja, a máxima alcançável possível.

5.1 Primeiro Circuito

O primeiro circuito proposto a ser “evoluído” é um amplificador inversor de 40 dB de ganho e banda passante de 100 MHz, excitando uma carga puramente capacitiva de 1 pF. Para esta “evolução”, será utilizada uma fonte simples de alimentação em lugar de uma simétrica. O ponto de operação escolhido tem como valor, metade do valor de pico da fonte de excitação, que excursionará de 0 V a 2,5 V, ou seja, 1,25 V Para otimização dos parâmetros, será utilizada a técnica Downhill-Simplex (Nelder-Mead).

A seguir, uma breve descrição dos parâmetros utilizados para esta “evolução”.

TOPOLOGIA

População de circuitos: 100.

Quantidade máxima de transistores por circuito: 6

Taxa de mutação Global: 50 %

Taxa de Genes mutáveis: 10 %

Aptidão Ideal (condição de parada): 1

Gerar novo indivíduo quando o cromossomo for defeituoso: sim

Indivíduos (cromossomos) de tamanho variável (conforme determinado anteriormente, máximo de 6 e mínimo de 2)

PARÂMETROS DA OTIMIZAÇÃO

Tipo de Otimização: Downhill Simplex (Nelder-Mead)

Percentagem de melhora: 1%

Valores de W e L mínimos possíveis: 0,35 um

Valores de W e L máximos possíveis: 20 um

Número máximo de iterações por otimização: 10

Número máximo de operações de “encolhimento global” permitido por otimização: 4

Foram utilizados para esta “evolução”, 4 computadores executando a sintonia dos parâmetros.

Após aproximadamente 1 hora e 30 minutos de execução, o sistema foi interrompido, apresentando os seguintes resultados:

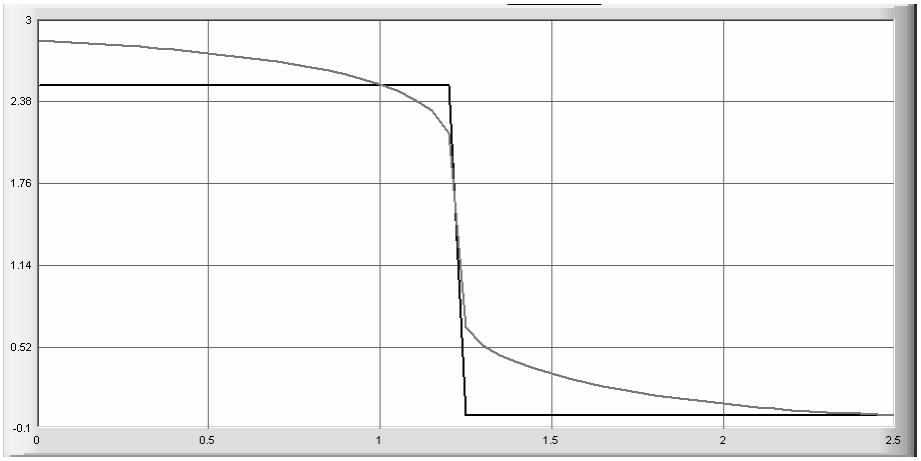

Figura 5.1-1 – Curva de transferência DC

Uma análise visual da curva de transferência DC nos revela que o melhor circuito atingido, no torneio 532 após 544 torneios, tem seu ponto de operação praticamente idêntico ao objetivo. A faixa dinâmica observada na análise DC é menor que o objetivo e impõe uma certa restrição em relação ao ideal desejado. A figura 5.1-1 mostra a curva de transferência DC atingida.

Com referência ao ganho em banda passante do amplificador (Análise AC), pode-se observar na figura 5.1-2 que a freqüência de corte ficou em torno de 10 MHz e o ganho desejado, ligeiramente inferior. Vale ressaltar que a deformação observada na banda passante do circuito evoluído é devida ao ponto de operação do circuito simulado ser ligeiramente diferente do ponto de operação do circuito objetivo.

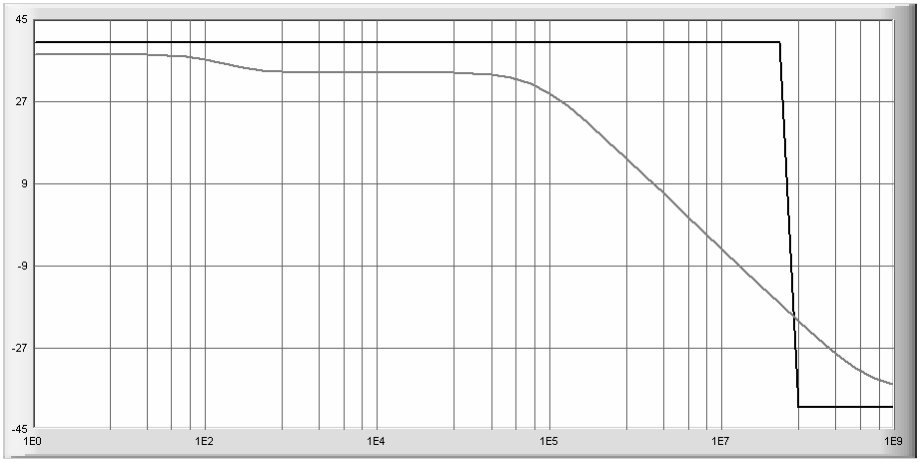

Figura 5.1-2 – Análise AC

Finalmente, o netlist padrão SPICE com os respectivos parâmetros W e L do melhor indivíduo evoluído apresentou-se como a seguir:

hora: 12:43:21
Iteração: 532
Melhor indivíduo: 93
fitness:0.613062

Entrada=Dreno
Entrada Aux=Dreno
3-0-3 PMOS W=18.29 L=17.11
4-4-0 PMOS W=11.17 L=8.76
0-3-0 PMOS W=2.98 L=15.08
5-3-1 PMOS W=12.3 L=14.27
5-4-0 NMOS W=10.18 L=17.77
Saída=Dreno

5.2 Segundo Circuito

O segundo circuito a ser “evoluído” tem características bastantes parecidas com o circuito anterior, ou seja, um amplificador com 40 dB de ganho e banda passante de 100 MHz, a diferença reside no tipo de alimentação, com fonte simétrica de ± 3,3 V. e principalmente, no tipo de otimização dos parâmetros utilizada, que será PSO. O ponto de operação do amplificador deverá ficar em torno de 0 V. com excitação variando de -2,5 V. a +2,5 V.

A seguir, uma breve descrição dos parâmetros utilizados para esta “evolução”.

TOPOLOGIA

População de circuitos: 50.

Quantidade máxima de transistores por circuito: 5

Taxa de mutação Global: 50 %

Taxa de Genes mutáveis: 20 %

Aptidão Ideal (condição de parada): 1

Gerar novo indivíduo quando o cromossomo for defeituoso (ex. entrada em curto com a alimentação): não

Indivíduos (cromossomos) de tamanho variável (conforme determinado anteriormente, máximo de 5 e mínimo de 2)

PARÂMETROS DA OTIMIZAÇÃO

Tipo de Otimização: Particle Swarm (PSO)

Número de partículas:10

Velocidade máxima da partícula: 10

Percentagem de melhora: 1%

Valores de W e L mínimos possíveis: 0,35 um

Valores de W e L máximos possíveis: 20 um

Número máximo de iterações por otimização: 100

Desenvolvimento do enxame de partículas : 0 (maior peso no aprendizado da melhor partícula).

Após aproximadamente 1 hora e 30 minutos de execução, o sistema foi interrompido, apresentando os seguintes resultados:

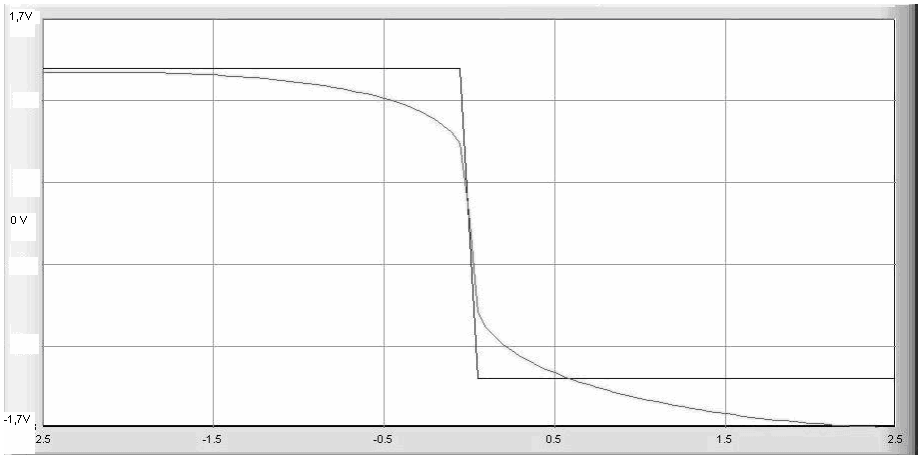

Figura 5.2-1 – Curva de transferência DC

Tal como observado no circuito anterior, uma análise visual da curva de transferência DC nos revela que o melhor circuito atingido no torneio 502, tem seu ponto de operação praticamente idêntico ao objetivo. A faixa dinâmica observada na análise DC é menor que o objetivo e impõe uma certa restrição em relação ao ideal desejado. A figura 5.2-1 mostra a curva de transferência DC atingida.

Com referência ao ganho em banda passante do amplificador (Análise AC), pode-se observar na figura 5.1-2 que a freqüência de corte ficou em torno de 50 MHz e o ganho desejado ficou, ora superior (em freqüências muito baixas), ora inferior mas sempre próximo ao desejado. Mais uma vez, vale ressaltar que a deformação observada na banda passante do circuito evoluído é devida ao ponto de operação do circuito simulado ser ligeiramente diferente do ponto de operação do circuito objetivo.

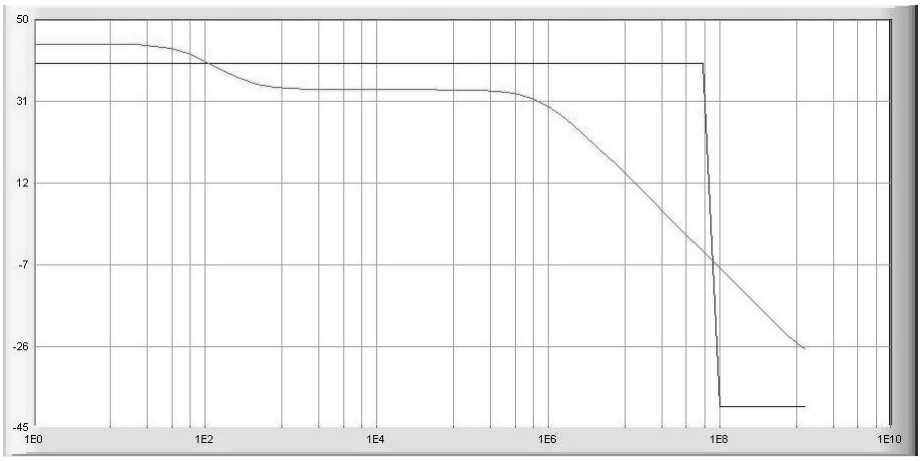

Figura 5.2-2 – Análise AC

Finalmente, o netlist padrão SPICE com os respectivos parâmetros W e L do melhor indivíduo evoluído apresentou-se como a seguir:

hora: 12:35:42
Iteração: 502
Melhor indivíduo: 17
fitness:0.537928

Entrada=Gate
Entrada Aux=Gate
2-3-4 NMOS W=3.01 L=6.35
0-2-0 NMOS W=16.65 L=11.98
1-3-4 PMOS W=19.83 L=13.53
4-3-2 PMOS W=3.99 L=8.6
Saída=Dreno

5.3 Terceiro Circuito

O terceiro circuito proposto, é um filtro passa-baixas com ganho unitário e freqüência de corte 100 Hz excitando uma carga capacitiva de 1 pF. Para esta "evolução", será utilizada uma fonte simples de alimentação em lugar de uma simétrica. O ponto de operação escolhido, tem como valor, metade do valor de pico da fonte de excitação, que excursionará de 0 V. a 2.5 V., ou seja, 1.25 V. . Para otimização dos parâmetros, será utilizada a técnica Particle Swarm (PSO)

A seguir, uma breve descrição dos parâmetros utilizados para esta "evolução".

TOPOLOGIA

População de circuitos: 100.

Quantidade máxima de transistores por circuito: 6

Taxa de mutação Global: 50 %

Taxa de Genes mutáveis: 10 %

Aptidão Ideal (condição de parada): 1

Gerar novo indivíduo quando o cromossomo for defeituoso (ex. entrada em curto com a alimentação): sim

Indivíduos (cromossomos) de tamanho variável (conforme determinado anteriormente, máximo de 6 e mínimo de 2)

PARÂMETROS DA OTIMIZAÇÃO

Tipo de Otimização: Particle Swarm (PSO).

Número de partículas:10

Velocidade máxima da partícula:10

Percentagem de melhora: 1%

Valores de W e L mínimos possíveis: 0,35 um

Valores de W e L máximos possíveis: 20 um

Número máximo de iterações por otimização: 20

Desenvolvimento do enxame de partículas : 0 (maior peso no aprendizado da melhor partícula).

Após aproximadamente 1 hora e 30 minutos de execução, o sistema foi interrompido, apresentando os seguintes resultados:

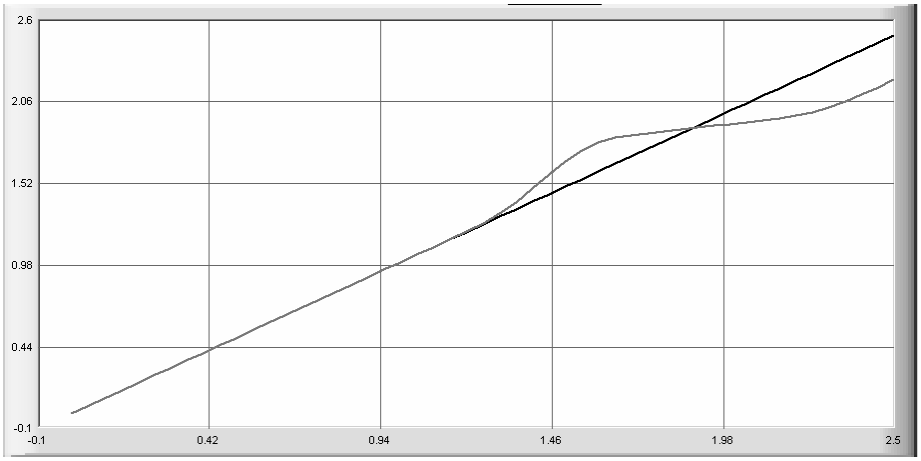

Figura 5.3-1 – Curva de transferência DC

Pela curva de transferência DC atingida, pode-se verificar que a linearidade do filtro evoluído, reside no intervalo de 0 V. a aproximadamente 1,35 V de amplitude do sinal de entrada.

A banda passante do filtro, que pode ser observada na figura da análise AC ficou em aproximadamente 100 Hz, bem próximo ao ideal desejado

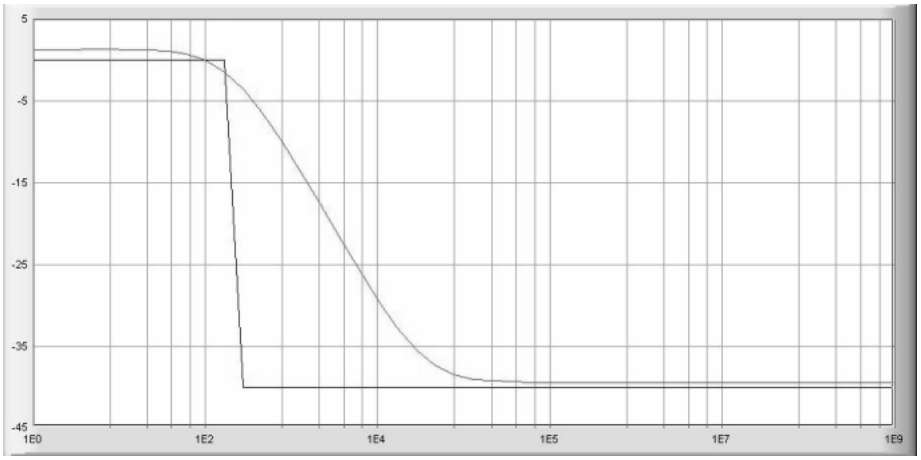

Figura 5.3-2 – Análise AC

Finalmente, o netlist padrão SPICE com os respectivos parâmetros W e L do melhor indivíduo evoluído apresentou-se como a seguir:

hora: 15:39:01
Iteração: 716
Melhor indivíduo: 90
fitness:0.5750843

Entrada=Gate
Entrada Aux=Gate
5-4-3 NMOS W=4.53 L=7.25
5-1-3 PMOS W=17.96 L=14.96
1-3-0 PMOS W=7.64 L=0.55
3-5-4 NMOS W=17.59 L=0.35
4-3-3 PMOS W=19.01 L=0.35
3-3-5 PMOS W=15.6 L=3.43
Saída=Dreno

6. Conclusões e Trabalhos Futuros

Neste trabalho foi proposto um ambiente de síntese de circuitos contendo apenas transistores CMOS com o objetivo de avaliar a eficiência dos processos de síntese evolucionária. Foi observado que a divisão do processo em síntese topológica e otimização numérica dos parâmetros W/L dos transistores permite reduzir consideravelmente o custo computacional do algoritmo.

Para a síntese topológica foi proposta uma nova codificação cromossomial que simplifica a aplicação dos operadores genéticos e permite a detecção de indivíduos inválidos com pouco custo adicional. O algoritmo de síntese proposto se mostrou bastante eficiente em relação aos seguintes pontos.

- Em função da média de tempo reservada, de aproximadamente 1 hora e 30 minutos, para a síntese e otimização dos circuitos nos exemplos apresentados e o valor de aptidão atingido pode-se afirmar que a separação proposta das tarefas envolvidas na síntese aumenta a eficiência do processo.

- Foi observado nos testes efetuados que, para o tipo de funções que regem o comportamento dos parâmetros W's e L's em circuitos CMOS, a otimização por Downhill-Simplex (ou Nelder-Mead) é mais eficiente computacionalmente do que o algoritmo de Particle Swarm (PSO). Entretanto, deve-se assinalar que o Downhill Simplex não é eficiente na busca de mínimos globais.

- A substituição de indivíduos defeituosos, gerados após uma operação de cruzamento ou mutação, por indivíduos válidos melhorou estatisticamente o desempenho do algoritmo.

Como trabalhos futuros, a validação de regras de construção de circuitos em estruturas do tipo FPTA (Pimentel, 2006), revela-se uma opção atraente.

Como exemplo do uso deste tipo de processo evolucionário em outros tipos de aplicação, a mesma estrutura de síntese e otimização, está sendo utilizada no desenvolvimento de um sistema de simulação de assinatura acústica/magnética de navios para utilização em navios varredores de minas.

Referências Bibliográficas:

CLERC M., KENNEDY J., *The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space*, IEEE Transactions on Evolutionary Computation, 2002, 6, 58-73

DARWIN, C., 1859, *On the origin of species by means of natural selection,* or*, The presevation of favored races in the struggle for life***.** London, John Murray.

DE GARIS, H., 1992, *Genetic Programming : GenNets, Artificial Nervous Systems, Artificial Embryos*. Ph.D. dissertation, Brussels University.

DE GARIS, H., 1993, "Evolvable Hardware: Genetic Programming of a Darwin Machine". In: *Artificial Neural Nets and Genetic Algorithms*, R.F. Albretch, C.R. Reeves, N.C. Steele (eds), Springer-verlag, NY.

DE JONG K., A., 1975 *An analysis of the behavior of a class of genetic adaptive systems*, Ph.D. dissertation, Univ. of Michigan, Ann Arbor, Diss. Abstr. Int. 36(10), 5140B, University Microfilms no. 76-9381.

GOLDBERG, D., 1989, Genetic Algorithms in search, optimization and Machine Learning. Addison-Wesley Publishing Company, Inc. Reading, Massachusetts.

HAUPT R. L., HAUPT S. E., 1998, *Pratical Genetic Algorithms,* John Wiley & Sons Inc, United States of America, 1 ed., US.

HEPPNER, F., GRENANDER, U. 1990, A Stochastic non Linear model for coordinated bird flocks, In S.Krasner, Ed. The Ubiquity of Caos, AAAS Publications, Washington, DC.

HIGUCHI, T., IWATA, M., KEYMEULEN, D., SAKANASHI, H., MURAKAWA, M.,KAJITANI, I., TAKANASHI, E., TODA, K., SALAMI, M., KAJIHARA, N., OTSU, N., 1999 *Real World Applications of Analog and Digital Evolvable Hardware*, IEEE Transactions on Evolutionary Computation, Vol3, no. 3, Setembro 1999.

HOLLAND, J., 1975, *Adaptation in Natural and Artificial Systems*. 1 ed. University of Michigan Press, Ann Arbor, USA

KENNEDY, J., EBERHART, R. C., “Particle Swarm Optimization”, In:*Proceedings of the 1995 IEEE Internacional Conference on Nerual Networks*, Perth, Australia, 1995, pp. 1942-1948.

KENNEDY, J., EBERHART, R. C., SHI, Y., 2001, Swarm Inteligence, Morgan Kaufmann Publishers, CA, USA.

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

LAGARIAS J. C., REEDS J. A.., WRIGHT M. H., WRIGHT P. E.,”Convergence Properties of the Nelder Mead Simplex Method in Low Dimensions”, In: SIOPT Volume 9 Issue 1,1998.

LINDEN, R., 2006, *Algoritmos Genéticos: Uma Importante ferramenta de Inteligência Computacional*., 1 ed., Brasport, Rio de Janeiro, Brasil.

MESQUITA, A.; SALAZAR, F.A.; CANAZIO, P.P.,2002, "Chromossome Representation Through Adjacency Matrix in evolutionary Circuit Synthesis". In: *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware*, pp. 102-109, IEEE Computer Press, July.

MILLER J., P. THOMSON, T. FOGARTY, *Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuis: A Case Study*, in Genetic Algorithms and Evolution Strategies in Enginnering and Computer Science, Wiley, 1997.

NELDER J. A. and MEAD R., *A simplex method for function minimization*, Computer Journal 7 (1965), 308-313.

PIMENTEL, A. L. P. 2006, *Estruturas CMOS Programáveis Para Aplicação Em Eletrônica Evolucionária*, Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

POMEROY, P., "An Introduction to Particle Swarm Optimization", In: *http://www.adaptiveview.com*, 15 Setembro de 2003.

PRADO J.R., SARAMAGO S.F.P., 2005, "Otimização por colônia de Partículas". In: *FAMAT em Revista* , Número 4, pp. 87-103, 2005.

REYNOLDS, C. W. 1987, Flocks, Herds and Schools: A Distributed behavioral Model. Computer Graphics, 21(4):25-34

ROJAS, J. E., VIANA, F.A.C., Rade, D. A. and Steffen Jr, V., "Force identification of mechanical systems by using particle swarm optimization". In: *Proceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, Agosto 30-01 Setembro 2004.

SANCHEZ, e., 1996, “Field Programmable Gate Array (FPGA) Circuits”. In: *Towards Evolvable Hardware: The evolutionary engineering approach*, v.1062, Lecture Notes of Computer Science, Springer-verlag, pp.1-18.

SCHWEFEL, H. P. E TAYLOR, L., “Evolution and Optimum Seeking”, John Wiley & Sons Inc, United States of America, pp. 87-88, 1994.

STOICA, A., KEYMEULEN, D., ZEBULUM, R., THAKOOR, A., DAUD, T., KLIMECK, G., JIN,Y., TAWEL, R., DUONG, V., 2000 *Evolution of analog circuits on Field Programmable Transistor Arrays* , Proceedings of the Second NASA DoD Workshop on Evolvable Hardware, pp.99-108, IEEE Computer press., July, 2000.

THOMPSON A., “An evolved circuit, intrinsic in silicon, entwined in physics”, In: *International Conference on Evolvable Systems. Springer-Verlag Lecture Notes in Computer Science*, 1996, 390-405

WRIGHT M. H., 1996, “Direct Search Methods: Once Scorned, Now Respectable”. In: *Proceedings of the 1995 Dundee Biennal Conference in Numerical Analysis*, pp. 191-208, Addison Wesley Longman, Harlow, United Kingdom.

ZEBULUM, R.S.,1999, *Síntese de Circuitos Eletrônicos por Computação Evolutiva.* Tese de D.Sc., PUC-RJ, Rio de Janeiro, RJ, Brasil

ZEBULUM, R. S.; PACHECO, M. A. C.; VELLASCO, M. M. R., 2002, *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*, CRC Press.