AMBIENTE DE TESTES DE ESTRUTURAS CMOS

PROJETADAS PARA ELETRÔNICA EVOLUCIONÁRIA

Carlos Philippe Emmanuel Vasconcellos Duarte

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. Marco Aurélio Cavalcanti Pacheco, Ph.D.

_____

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

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2006

DUARTE, CARLOS PHILIPPE EMMANUEL VASCONCELLOS

Ambiente de Testes de Estruturas CMOS Projetadas para Eletrônica Evolucionária [Rio deJaneiro] 2006

XI, 88p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Algoritmo Evolucionário

2. Proposta de Barramento

Dedicatória

À minha esposa Maria Cristina e minha filha Jaqueline.

Agradecimentos

- Ao Prof. Dr. Antonio Carneiro de Mesquita Filho;
- Ao Instituto de Pesquisas da Marinha, em especial aos Comandantes CMG(EN) João Roberto Vasconcellos Martins, CF João Alberto Vianna Tavares e CF(EN) João Luís Marins;
- Ao amigo Antonio Luiz Pimentel Guimarães;
- À minha família, pelo apoio e compreensão;
- À UFRJ.

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

AMBIENTE DE TESTES DE ESTRUTURAS CMOS PROJETADAS PARA ELETRÔNICA EVOLUCIONÁRIA

Carlos Philippe Emmanuel Vasconcellos Duarte

Março/2006

Orientador: Antonio Carneiro de Mesquita Filho

Programa: Engenharia Elétrica

A arquitetura do sistema de barramento e o projeto das estruturas necessárias à sua implementação em um circuito integrado do tipo FPTA a ser utilizado como plataforma de hardware reconfigurável em processos evolucionários intrínsecos são discutidos. O circuito é composto de transistores programáveis interligados em uma estrutura de barramentos locais e globais visando a escalabilidade do circuito. Para avaliar as diversas arquiteturas de barramento propostas, foi desenvolvido um ambiente de teste que implementa um algoritmo genético na variante “steady state” utilizando o simulador Spice em uma arquitetura de rede.

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

ENVIRONMENT FOR TESTING OF CMOS STRUCTURES DESIGNED FOR EVOLUTIONARY ELECTRONIC

Carlos Philippe Emmanuel Vasconcellos Duarte

March/2006

Advisor: Antonio Carneiro de Mesquita Filho

Program of: Electrical Engineering

The architecture of the bus system and the project of necessary structures to its implementation in an integrated FPTA (Field Programmable Transistor Array) circuit to be used as a reconfigurable hardware platform in evolutionary intrinsic systems will be discussed. The circuit is constituted of programmable transistors interconnected in a structure of local and global buses aiming the scalability of the circuit. In order to evaluate the various bus architectures, a test environment was developed which implements a genetic algorithm in the “steady state” variant by using the “Spice” simulator in a network.

Índice:

Índice de Figuras:

1. Introdução

1.1. Breve Histórico da Computação Evolucionária

A evolução das espécies pela seleção natural é um dos mais envolventes temas da ciência moderna. Inspirado pelo princípio Darwiniano 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 computação evolucionária é classicamente dividida em quatro sub-áreas principais:

- *Algoritmo Genético (Genetic Algorithm),*
- *Programação Genética (Genetic Programming),*
- *Estratégia Evolutiva (Evolution Strategies) e*
- *Programação Evolucionária (Evolutionary Programming)*

A essas sub-áreas estão relacionadas ainda as linhas de pesquisa em (Eberbach,2000) Vida Artificial (*Artificial Life*), Sistemas classificatórios (C*lassifier Systems*), Computação e Bio-informática baseadas em DNA (*DNA-based Computing and Bio-informatics*), Robótica Evolucionária (*Evolutionary Robotics*) e Hardware Evolucionário (*Evolvable Hardware*), Otimização utilizando Enxame de partículas *(Particle Swarm)* e Colônias de Formigas (*Ant Colony System*).

No seguinte histórico foram relacionadas cronologicamente, algumas obras para ilustrar o avanço desse campo de pesquisa. Com o surgimento de novas plataformas computacionais mais poderosas prevê-se uma utilização cada vez maior desse tipo de ferramenta.

- 1953 N. Barricelli “Symbiogenesis” experiments no IAS (Institute of Advanced Studies) (Barricelli,1953)

- 1957 Alex S. Fraser and Barricelli publicaram independentemente sobre Computação Evolucionária; Fraser já usava diploides, alelos e crossover (Barricelli,1957).
- 1962 G.E.P. Box, G.J. Friedman, W. W. Bledsoe, H.J. Bremermann independentemente desenvolveram algoritmos inspirados em evolução para otimização e aprendizado de máquina, mas o trabalho não teve seguimento. Friedman (1959) tentou desenvolver um código para computador, sem obter sucesso, demonstrando que a variação e a seleção de indivíduos podem funcionar somente se a variação estiver apta a produzir filhos viáveis (Bremermman,1962).
- 1962 John Holland, na Univ. of Michigan, começou a publicar e ensinar *Adaptive Systems*, e em 1965 foi o primeiro que descreveu a importância do crossover (Holland,1962).
- 1965 Ingo Rechenberg, Berlin, desenvolveu uma estratégia de evolução (*Evolution Strategy)* sem população ou crossover através de uma mutação nos pais para produzir um filho, mantendo o melhor indivíduo (Rechenberg,1965).
- 1966 L.J. Fogel, A.J. Owens e M.J. Walsh publicaram o primeiro livro sobre computação Evolucionária, "*Inteligência Artificial Através de Evolução Simulada"*, introduzindo como estratégia de evolução a *programação evolucionária* baseada em máquinas de estado finitas mantendo o melhor dos pais e um filho (Fogel,1966).
- 1970 A. S. Fraser and D. Burnell publicou um segundo livro, "*Computer Models in Genetics"*, abrangendo uma década de trabalho (Fraser,1970).
- 1975 John Holland publicou "*Adaptation in Natural and Artificial Systems"* consistindo em uma análise introdutória com aplicações na biologia, controle, e inteligência artificial. Considerado um dos trabalhos de maior importância nesse campo estabeleceu a base teórica formal para a otimização

evolucionária com a introdução de *esquemas*, construção de blocos e soluções parciais (Holland,1975).

- 1975 Kenneth De Jong's em sua tese "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", segundo Holland, introduziu um set de funções de teste com propriedades bem definidas, demonstrando a larga aplicabilidade dos Algoritmos Genéticos (De Jong,1975).

Essas pesquisas iniciais tiveram pouca repercussão devido à falta de plataformas computacionais de alto desempenho. A partir de 1980 com o advento da tecnologia VLSI, dos microprocessadores e o conseqüente aumento da capacidade computacional das máquinas, pôde-se verificar um novo impulso nessa área de pesquisa. A partir de então, os trabalhos passam a ter maior aplicabilidade na solução de problemas complexos, para os quais os métodos tradicionais não eram eficientes. Assim, podem ser citados;

- 1985 N.L. "Cramer, *A Representation for the Adaptive Generation of Simple Sequential Programs"* (Cramer,1985), um sistema adaptativo para geração de curtas funções seqüênciais.
- 1987 C. Fujiki and J. Dickinson publicaram baseado em GAs um código em lisp para solução do problema dilema do prisioneiro (Fujiki et al,1987).
- 1988 (patent filed) /1989 (publicação) John Koza, em Stanford, introduced *genetic programming*, uma aplicação da computação evolucionária para representação em árvore binária de programas em LISP visando a solução de problemas de engenharia (Koza,1989).

Uma grande área de aplicação da Computação Evolucionária, que se desenvolveu recentemente é a da Eletrônica Evolucionária (Higuchi,1999) que consiste na utilização de circuitos integrados programáveis em conjunto com os métodos evolucionários visando a síntese automática de circuitos.

Nesse campo a pesquisa tem se orientado principalmente para a busca de arquiteturas reprogramáveis especificamente voltadas à síntese evolucionária (Stoica,2000) (Zebulum,1997).

As pesquisas iniciais (Holland,1962) – (De Jong,1975) na área de Eletrônica Evolucionária, utilizavam algoritmos totalmente baseados em simulações em software que despendiam muito tempo para propósitos práticos. Isso porque um grande número de iterações é requerido pelo algoritmo evolucionário, gerando um circuito, por vezes difícil de ser implementado em hardware. Por isso é necessário impor restrições relacionadas à tecnologia de fabricação utilizada para que os circuitos gerados pelo processo evolucionário possam ser diretamente mapeados no hardware. A evolução em hardware pode reduzir o tempo para alcançar a resposta de um circuito de dias para segundos (Thompson,1996). Os FPGAs (Field Programmable Gate Arrays) (Sanchez,1996) e os FPAAs (Field Programmable Analog Arrays) (Zebulum,2000),são exemplos de plataformas apropriadas para a eletrônica evolucionária.

Uma característica importante do hardware evolucionário é o nível utilizado de fracionamento dos componentes dos circuitos, a chamada granulosidade. Uma limitação conhecida dos FPAAs (Zebulum,2000) é o seu baixo nível de granulosidade, basicamente o amplificador operacional. Analogamente as FPGAs, que se aplicam essencialmente à síntese de circuitos digitais, possuem também um baixo nível de granulosidade utilizando como componentes básicos as portas lógicas. Os FPTAs (Field Programmable Transistor array) (Langeheine,2001) (Zebulum,2000), surgiram como proposta de uma plataforma versátil para evolução em hardware, sendo o seu nível de granulosidade indo até o transistor, o que permite evoluções tanto na área digital quanto, e principalmente, na área analógica.

Relativo ao Hardware Evolucionário, tem-se o seguinte histórico:

- 1993 Tetsuya Higuchi *et al*., *Evolvable Hardware with Genetic Learning: A first step towards building a Darwin machine* (Higuchi,1993).

- 1994 P. Marchal *et al*, Embryological Development on Silicon Artificial Life IV (Marchal,1994).
- 1995, O primeiro Workshop Internacional - EVOLV95 in Lausanne ;

  Utilizando FPTA (Field Programmable Transistor Array), objeto deste trabalho, podemos destacar:
- University of Heidelberg – Kirchoff Institute for Physics.
    - Circuito integrado FPTA consistindo de um array de 256 células de transistores programáveis (Langeheine,2001).
- Jet Propulsion Laboratory
    - Circuito integrado FPTA com oito transistores e vinte e quatro chaves (Stoica,1999).
- PUC – RJ
    - PAMA – Programmable Analog Multiplexer Array (Santini,2001).

1.2. Objetivos

Neste trabalho são descritas as fases iniciais do projeto de parte de um circuito integrado do tipo FPTA a ser utilizado como plataforma de hardware reconfigurável em processos evolucionários. O circuito a ser fabricado em tecnologia CMOS, deverá ser composto de transistores programáveis e possuir o maior grau de liberdade para interconexão dessas células. Para isso fez-se necessário o projeto de um barramento composto de chaves analógicas, acionadas por bits (cromossomo) como alternativa aos esquemas de interconexão do tipo Norte-Sul-Leste-Oeste (NSEW) utilizados nas implementações atuais (Stoica,1999) (Langeheim,2001).

O projeto e avaliação de células de transistores programáveis é discutida e apresentada em (Guimarães,2006), sendo a parte relativa ao projeto e avaliação das arquiteturas de barramentos propostas, o objetivo deste trabalho.

Como etapa preliminar do projeto de uma arquitetura de barramentos, foi necessário o desenvolvimento de um Ambiente de Teste e Avaliação de estruturas de

interligação de células compostas por transistores programáveis. A proposta foi baseada na arquitetura da PAMA (Programmable Analog Multiplexer Array) (Santini,2001), desenvolvida na PUC-RJ, cuja função é de evoluir circuitos analógicos genéricos baseados em componentes discretos, sem que seja necessário o uso de simuladores.

O ambiente desenvolvido implementa um Algoritmo Genético na variante Steady-State, utilizando o simulador SPICE em arquitetura em rede, para permitir maior velocidade de execução. Uma máquina principal fica responsável por executar o algoritmo e distribuir os arquivos para serem simulados nas máquinas auxiliares. Foram utilizadas oito máquinas para receberem os arquivos do computador principal.

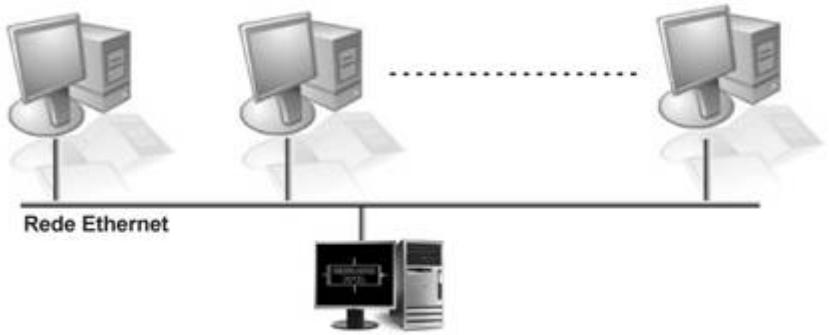


Figura 1-1–Ambiente de simulação

Esta mesma ferramenta poderá futuramente ser facilmente adaptada para evolução intrínseca Figura 1-2.

As estruturas são baseadas na tecnologia CMOS AMS 0.35µm (Austriamicrosystems).

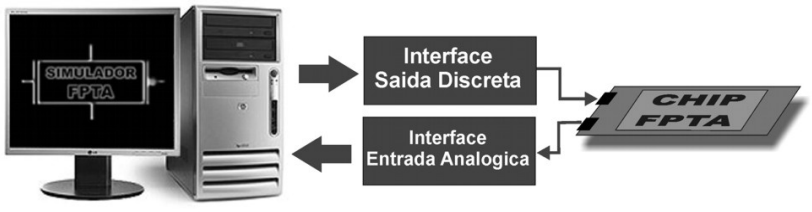


Figura 1-2 - Arquitetura de Hardware para Evolução Intrínseca

1.3. Estrutura do Trabalho

Esta dissertação está dividida em cinco capítulos adicionais, descritos a seguir:

O capítulo 2 abrange os conceitos fundamentais, de onde se buscou a inspiração para que circuitos possam ser criados, copiando o modo como a vida natural age nos indivíduos, porém na busca de um processo mais acelerado.

Introduz os conceitos de algoritmo genético, eletrônica evolucionária e plataformas para evolução, que nortearam a especificação do projeto para circuito evolucionário. Faz algumas considerações sobre o uso da estrutura proposta e descreve a metodologia utilizada para validação da proposta.

O capítulo 3 apresenta as especificações gerais de projeto e considerações sobre o barramento.

O capítulo 4 apresenta a proposta do circuito, descrevendo as estruturas utilizadas em separado e a descrição detalhada do projeto. São apresentados alguns testes iniciais para verificação de performance da estrutura.

O capítulo 5 descreve o ambiente de simulação que foi criado para testes das estruturas de hardware e como foi implementado o algoritmo para esse ambiente.

O capítulo 6 apresenta os resultados obtidos com o ambiente de simulação e por fim, o capítulo 7 apresenta a conclusão e sugere novas direções para continuação da pesquisa.

2. A Síntese de Circuitos Usando Eletrônica Evolucionária

2.1. Conceitos Fundamentais

O ser humano utiliza um poderoso conjunto de dispositivos sensores-atuadores incluindo olhos, ouvidos, mãos e cérebro que compartilham uma característica essencial: eles são produto de mutações aleatórias e cruzamentos genéticos bem sucedidos, como descrito pela teoria darwinista da evolução das espécies.

Segundo (Konner, 1998):

"A neuroanatomia em qualquer espécie -- especialmente em uma espécie dominada pelo cérebro, como a nossa -- é produto de um processo evolutivo relativamente desordenado, oportunista, simultâneo e apenas parcialmente integrado, passado ao longo de bilhões de anos, que deu origem a órgãos desiguais, evoluídos em animais diferentes, em eras diferentes, e com propósitos diferentes"

Essa sentença singular, que captura os princípios básicos da evolução natural, também contém todas as características necessárias para descrever uma má engenharia em termos humanos. Entretanto, entre o homem e a natureza, a última é a mais engenhosa. Organismos biológicos, uma das mais intrincadas estruturas conhecidas pelo homem, exibem comportamento guiado pela cooperação paralela de um grande número de elementos relativamente simples: as células. As tarefas impostas a muitos sistemas computacionais atuais têm exigido o desenvolvimento de técnicas cada vez mais complexas. Como a inteligência humana oferece uma limitação natural para o nível de complexidade desses sistemas, a engenharia vem utilizando, em uma escala cada vez maior, a natureza como fonte de inspiração no projeto de sistemas de software e hardware.

A vida natural é dominada por dois grandes sistemas estocásticos. Um sistema está no indivíduo e é chamado de aprendizado, enquanto o outro diz respeito à evolução da população responsável pelas múltiplas gerações. Como dito acima,

cientistas têm buscado na natureza, inspiração para suas pesquisas e a eletrônica evolucionária não foge à regra. Sipper e colegas (Sipper *et al*, 1997) desenvolveram também um modelo unificado para identificação dos aspectos biológicos que podem ser replicados em sistemas de engenharia. O modelo, conhecido como Filogenia**,** Epigenesia e Ontogenia (FEO), procura classificar o tipo de relevância biológica de um sistema segundo três direções distintas.

2.1.1. Inspiração Biológica

2.1.1.1. Filogenia

Darwin notou que a seleção evolucionária nas espécies ocorre randomicamente e que a sobrevivência ou a extinção de cada organismo é determinada pela sua habilidade de se adaptar ao meio ambiente. Cada vida existente tem uma manifestação física ou uma aparência que se deve ao seu fenótipo, o qual é determinado pelo genótipo, codificado em forma de DNA.

Mutações, que são erros no processo de duplicação do DNA que ocorrem durante a divisão celular e o cruzamento entre cromossomos (parte integrante do DNA), resultam na evolução das espécies, aumentando assim o surgimento de novas espécies bem como sua contínua adaptação às variações do meio ambiente. Portanto Filogenia é a história evolucionária de uma espécie, sendo encontrada em sistemas em que vetores binários (genoma) evoluem ao longo do tempo, através de um algoritmo evolucionário (AE).

Outro exemplo de exploração de mecanismos filogenéticos são programas de vida artificiais, utilizados para simular processos evolucionários, em que indivíduos (programas) procuram sobreviver em um ambiente de recursos limitados (memória e/ou tempo de processamento).

2.1.1.2. Epigenesia

Teoria biológica segundo a qual a constituição de seres vivos se inicia a partir de células não-estruturadas e se faz mediante sucessiva formação e adição de partes que não existiam na unidade inicial. Considerando que um ser humano adulto é formado por cerca de $6x10^{13}$ células, formadas a partir de um genoma composto por $3x10^{9}$ bases nitrogenadas, não possuindo, portanto, informação suficiente para descrever todo o organismo, fica claro que o aumento na complexidade é promovido por um mecanismo extracelular. Ou seja, o desenvolvimento de um organismo deve ser influenciado pelo ambiente, que atua como fonte de informação adicional. Na engenharia, esse conceito encontra contrapartida em problemas de aprendizagem, surgindo daí sua aplicação em Inteligência Artificial (IA).

2.1.1.3. Ontogenia

O processo de desenvolvimento de organismos multicelulares é conhecido como ontogenia. Isso inclui as sucessivas divisões da célula mãe, o zigoto, onde cada nova célula formada possui uma cópia do genoma original, e a especificação da sua posição no conjunto. Sistemas de inspiração ontogenética devem implementar processos correspondentes à divisão celular -- que permite a formação de um sistema multicelular, que trabalha paralelamente e à diferenciação celular que permite a uma célula, que possui todo o genoma do organismo, assumir uma função específica (p. ex.: rodar um trecho do código).

2.1.1.4. Taxinomia dos Sistemas Eletrônicos Evolucionários

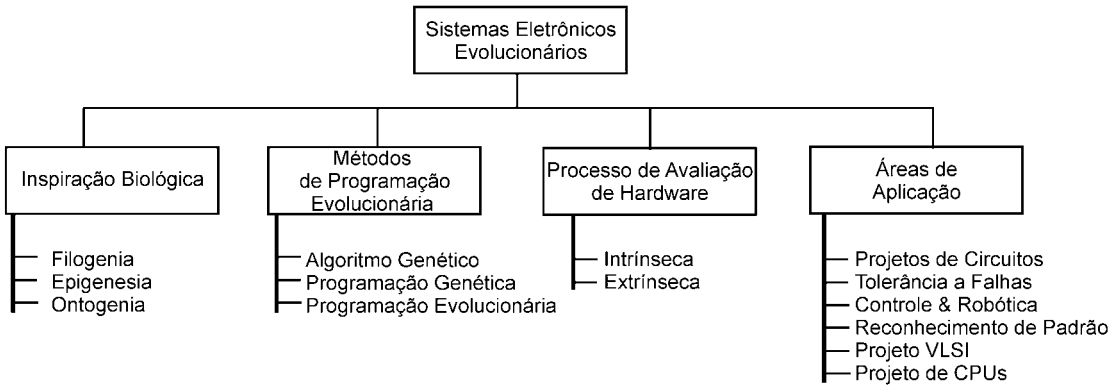


Figura 2-1 – Taxinomia dos sistemas evolucionários (ROMAHI,1998)

2.1.2. Métodos de Programação Evolucionária

Embora exista um número significativo de métodos para a implementação de projetos evolucionários, três principais são usados em hardware evolucionário. O Algoritmo Genético, popularizado por Holland, seguiu o método natural, aplicando os conceitos de mutação e cruzamento (crossover) a um genótipo codificado de maneira binária. A Programação Genética introduzida por De Garis (1993) e John Koza (1992), estende a idéia do algoritmo genético para o domínio dos programas de computador. Isso é feito representando cromossomos como árvores cujos nós das ramificações (galhos) são funções e suas folhas variáveis ou constantes. Finalmente, a Programação Evolucionária difere dos demais principalmente na codificação de cromossomos, assim como nos operadores genéticos utilizados. Cromossomos neste caso tendem a ser adotados como números inteiros ou reais.

2.1.3. Processo de Avaliação de Hardware

Existem duas principais metodologias chamadas de extrínseca e intrínseca, atribuídas ao De Garis(1993) ou on-line e off-line segundo Yao(1996).

O método extrínseco refere-se ao caso onde o processo evolucionário é feito efetivamente em software. Esse método se caracteriza por executar o processo evolucionário em simuladores (usando pacotes como o SPICE), uma vez que atinja a função desejada o hardware pode então ser de fato construído.

Evolução intrínseca refere-se à execução do processo evolucionário diretamente no hardware, cromossomos ou genótipos são carregados no circuito e são avaliados através da aplicação de sinais de entrada ao hardware e leitura da resposta a esses sinais. A principal vantagem desse processo é a não necessidade de desenvolver modelos do circuito, tornando o processo muito mais rápido.

É importante notar que existe sobreposição entre esses métodos, porque é possível ter plataformas híbridas onde os dois processos convivam durante a vida útil do sistema, evoluindo o circuito de acordo com a variação do ambiente ou corrigindo alguma avaria que possa ter sofrido.

2.2. Algoritmo Evolucionário

Dado um problema, o conjunto de todas as possíveis soluções é chamado de espaço problema ou espaço de busca. O algoritmo evolucionário é um processo iterativo que consiste de uma população de indivíduos, cada um representado por uma seqüência finita de símbolos, conhecido como genoma, codificando uma possível solução no espaço problema. O algoritmo evolucionário encontra aplicação onde o espaço de busca é demasiadamente grande para ser exaustivamente pesquisado.

A Figura 2-2 apresenta os passos de um algoritmo evolucionário típico. Inicialmente a população é gerada aleatória ou heuristicamente, definindo-se um critério de aptidão (fitness) para avaliar cada genoma. Os indivíduos mais aptos se reproduzem, sendo que os genes dos descendentes sofrem recombinação e mutação. O processo se repete até algum critério de parada ser atingido.

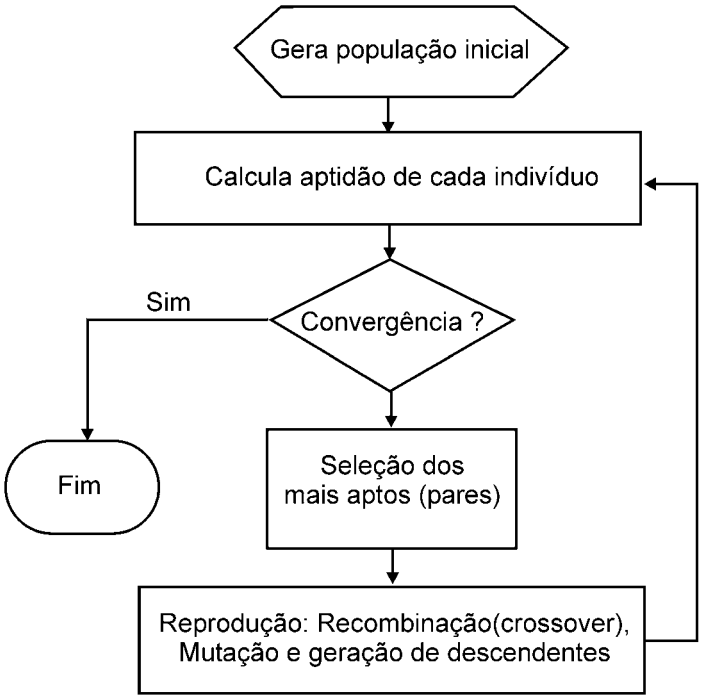


Figura 2-2 – Algoritmo Evolucionário básico

2.2.1. Fundamentos Matemáticos

A maioria dos trabalhos procura encontrar regras empíricas para melhorar seu desempenho, não havendo uma teoria geral que explique de forma completa como e porque a técnica funciona. Algumas hipóteses são bem aceitas como é o caso do teorema de Esquema proposto por Holland em1975. Esquema é um padrão de valores de genes que podem ser representados em codificação binária. Um cromossomo pode conter um ou mais padrões. Com o passar das gerações, as soluções boas tendem a compartilhar certas partes de seu cromossomo segundo o teorema do Esquema a ser discutido a seguir.

2.2.1.1. Representação de um Esquema

$$H = 11*$$

onde: * pode assumir valor de “0” e “1” (don’t care)

H é um padrão que descreve todos os cromossomos do espaço 2^3, cujos dois primeiros bits são iguais a 1, não importando os demais. Portanto, 111 e 110 são cromossomos que pertencem a $H = 11*$.

2.2.1.2. Número de Esquemas

Seja o espaço de busca ou espaço problema K^L onde:

K = número de elementos do alfabeto de representação

L = comprimento do cromossomo

$$\text{Total de Esquemas} = (K+1)^L$$

2.2.1.3. Ordem de um Esquema

Ordem ou especificidade $\vartheta(H)$ = número de posições fixas (diferentes de *) presentes no Esquema.

H = 0 1 1 * 1 * * O(H) = 4
H = 0 * * * * * * O(H) = 0

2.2.1.4. Comprimento de um Esquema

$\delta(H)$ = distância entre a primeira e a última posição específica, diferente de *, no Esquema.

H = 0 1 1 * 1 * * δ(H) = 4
H = 0 * * * * * * δ(H) = 0

2.2.1.5. Indivíduos pertencentes a um Esquema

Um indivíduo pertence a um Esquema se para todas as L posições o símbolo do indivíduo é igual ao símbolo do Esquema, exceto nas posições onde o símbolo do Esquema é don't care (*).

Um Esquema possui $2^{L-O(H)}$ indivíduos

$Exemplo: *1* possui 2^{3-1} indivíduos$

010

011

110

111

2.2.1.6. Esquemas representado por um indivíduo

- Um indivíduo representa 2^L Esquemas.
- Para cada uma das L posições de um indivíduo, define-se um Esquema diferente, usando o símbolo presente no indivíduo ou o símbolo $*$
- Exemplo: 0 1 0 representa os seguintes Esquemas:

010

$*10$

$0*0$

$01*$

$0**$

$*1*$

$**0$

$***$

2.2.1.7. Efeito da Seleção

Esquemas permitem analisar o efeito global da reprodução e dos operadores genéticos

- Seja m(H,t), o número de representantes do Esquema H na população no ciclo t.
- Sabemos que $p_{i=} \frac{f_i}{\Sigma f_j}$ é a probabilidade do cromossomo *i* ser escolhido
- Logo, o número esperado de representantes de H no ciclo seguinte (t+1) é:

$$m(H,t+1) = n * \Sigma^{i \in H} \frac{f_i}{\Sigma^n f_j}$$

- Definindo a aptidão média do Esquema H, como:

$$f(H) = \Sigma^{i \in H} \frac{f_i}{m(H,t)} \text{ então,}$$

$$m(H,t+1) = m(H,t) * n * \frac{f(H)}{\Sigma^n f_j}$$

como $f_{médio} = \Sigma^n \frac{f_j}{n}$ *então*,

$$m(H,t+1) = m(H,t) * \frac{f(H)}{f_{médio}}$$

Podemos dizer que:

1 - Esquemas com aptidão acima da média proliferam;

2 - Esquemas com aptidão abaixo da média tendem a desaparecer.

2.2.1.8. Taxa de Evolução

- Supondo H acima da média de um fator constante C estacionário, a partir de t=0:

$$m(H,t+1) = m(H,t) * \frac{f_{médio} + C * f_{médio}}{f_{médio}}$$

$$m(H,t+1) = m(H,t) * (1+C)$$

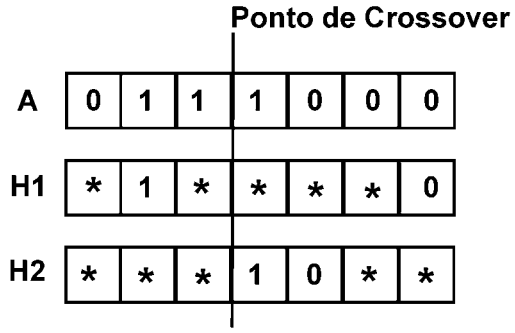
Assim, para qualquer t temos:

$$m(H,t+1) = m(H,0) * (1+C)^t$$

O número de ocorrências nas gerações sucessivas de bons(maus)Esquemas, crese(decresce) exponencialmente.

2.2.1.9. Efeito do Crossover

Para verificar o efeito do crossover nos Esquemas, façamos o cruzamento de A com outro genitor. Podemos notar que H1 será destruído e padrão não será transmitido aos descendentes, a não ser que par genitor de A possa recuperar padrão. H2 sobreviverá e será transmitido a um dos descendentes. Isso se deve ao fato de que o ponto de corte está situado entre dois definidos alelos, que se encontram bem afastados.



2.2.1.10. Probabilidade de Destruição

$$P_d = \frac{\delta(H)}{L-1}$$

A probabilidade de sobrevivência é:

$$P_S = 1 - \frac{\delta(H)}{L-1}$$

Então, considerando a probabilidade do crossover e a recuperação de H após o crossover temos,

$p_s \geq 1 - p_c * \frac{\delta(H)}{L-1}$, portanto:

$$m(H,t+1) \geq m(H,t) * \frac{f(H)}{f_{médio}} \left[1 - p_c * \frac{\delta(H)}{L-1} \right]$$

2.2.1.11. Efeito da Mutação

Seja p_m a probabilidade de uma posição sofrer mutação. $1-p_m$ é a probabilidade de sobrevivência. H tem O(H) posições fixas, assim a probabilidade de sobrevivência do Esquema é: $(1-p_m)^{O(H)}$. Sabendo que $p_m \ll 1$, então:

$$(1-p_{m)^{O(H)}} \approx 1 - O(H) * p_m$$

2.2.1.12. Teorema Fundamental

$$m(H,t+1) \geq \frac{m(H,t) * f(H)}{f_{médio}} \left[1 - p_c * \frac{\delta(H)}{L-1} \right] * \left[1 - O(H) * p_m \right]$$

"Esquemas curtos de baixa ordem e com alta aptidão tendem a proliferar nas gerações sucessivas a uma taxa exponencial."

2.2.2. Algoritmo Genético

A implementação prática de um algoritmo genético requer atenção para os seguintes pontos:

- Escolha da função de aptidão
- Problemas de convergência
- Escolha da técnica de seleção
- Lacuna entre gerações

Os dois principais algoritmos usados são: o generacional, onde há substituição da população a cada geração, e o de estado estacionário (steady state) onde há superposição da população em cada ciclo. O algoritmo generacional, mostrado na Figura 2-3, apresenta alguns problemas significativos como, por exemplo, a possibilidade de indivíduos com aptidão elevada não se reproduzirem e indivíduos duplicados serem selecionados e proliferarem rapidamente na população. Esses problemas podem fazer com que populações de tamanho reduzido percam a diversidade rapidamente. Para minimizar o problema usa-se o elitismo, permitindo que o melhor indivíduo passe para a próxima geração preservando as melhores soluções.

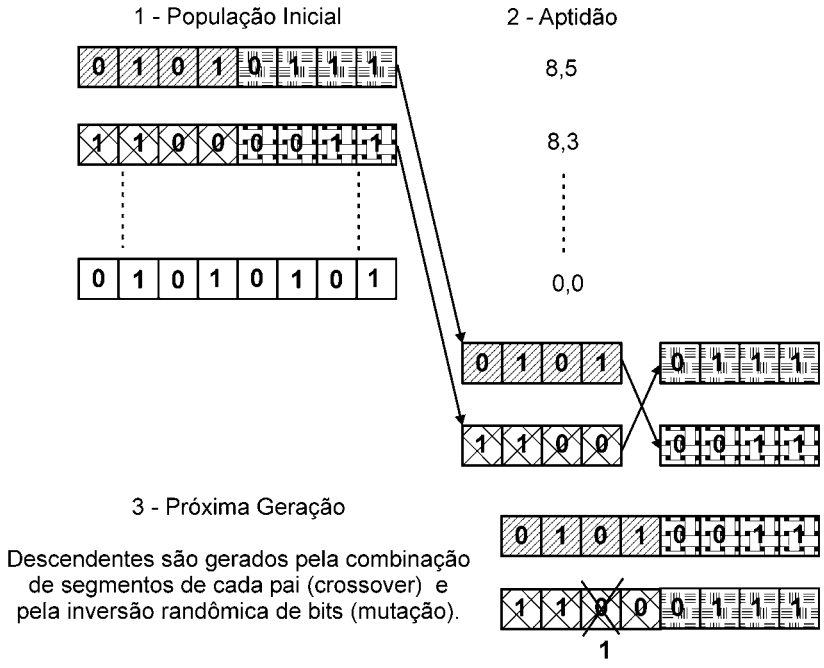


Figura 2-3 – Algoritmo baseado em gerações

2.2.2.1. Função de aptidão

A aptidão (fitness) de um indivíduo depende do desempenho do fenótipo sendo calculada a partir do genótipo através de uma “Função de Aptidão” (Fitness Function). Ela deve retornar um valor numérico único (figura de mérito) e depende do problema a resolver. Ela pode ser óbvia em alguns casos, se o objetivo é encontrar máximos ou mínimos de uma função ou pode ser a própria função a otimizar.

Muitas vezes a função objetivo deve recompensar sub-objetivos e também introduzir funções de penalidade que indicam a qualidade de um cromossomo, neste caso a função aptidão fica sendo uma constante menos a punição.

2.2.2.2. Convergência

A convergência prematura acontece quando uma população de tamanho finita, de indivíduos de alta aptidão, mas não de aptidão ótima, levam o algoritmo genético a convergir para um máximo local e não global. A convergência lenta acontece quando a aptidão dos indivíduos é próxima da média da população, o que costuma ocorrer perto do fim do algoritmo. Uma maneira de tentar resolver esse tipo de problema é modificar o modo de seleção de indivíduos para reprodução.

2.2.2.3. Técnicas de Seleção

Método de seleção é uma tarefa para alocar oportunidades de reprodução (reproductive trials) a cada indivíduo da população. Oportunidade de reprodução é um valor inteiro derivado da aptidão que indica o número de cópias que o indivíduo terá no grupo de reprodutores (mating pool).

- Método geral de seleção
  - Indivíduos da população original são copiados para um grupo de reprodutores (em geral do tamanho da população original).

- Indivíduos com boa aptidão são selecionados muitas vezes para serem copiados no grupo de reprodutores.
- Indivíduos com aptidão média são selecionados menos vezes.
- Indivíduos com baixa aptidão nunca são selecionados.
- Após a formação do Grupo de Reprodutores, pares de indivíduos deste grupo são escolhidos randomicamente para reprodução, sendo tirados do grupo até que este esteja vazio.

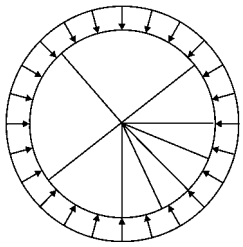
❑ Métodos de seleção baseados na forma de usar a aptidão de cada indivíduo para calcular o valor da oportunidade de reprodução:

- Remapeamento Explícito de Aptidão (Explicit Fitness Remapping)
- Remapeamento Implícito de Aptidão (Implicit Fitness Remapping)

❑ Remapeamento Explícito de Aptidão (Explicit Fitness Remapping)

- Método básico de remapeamento: a aptidão de cada indivíduo é remapeada dividindo o valor da aptidão do individuo pela aptidão média da população.
- As oportunidades de reprodução são alocadas de modo proporcional a este valor remapeado.
- O resultado da divisão deve ser convertido em inteiro de tal modo a não introduzir uma propensão (“bias”), o que requer um método de amostragem.
- Amostragem Universal Estocástica (Stochastic Universal Sampling) ou Seleção por Roleta:
    - ✓ Seja um círculo dividido em n regiões (tamanho da população), onde a área de cada região é proporcional à aptidão do indivíduo. Coloca-se sobre este círculo uma "roleta" com n cursores, igualmente espaçados. Após um giro da roleta a posição dos cursores indica os indivíduos selecionados. Os

indivíduos cujas regiões possuem maior área terão maior probabilidade de serem selecionados várias vezes.



- Seleção por Classificação:
    - ✓ Roleta tem problemas quando há grandes diferenças entre os valores de aptidão.
    - ✓ A Seleção por Classificação: primeiro classifica a população e então atribui a cada cromossomo um valor de aptidão determinado pela sua classificação.
    - ✓ O pior terá aptidão igual a *1*, o segundo pior *2* etc. de forma que o melhor terá aptidão igual a *N* (número de cromossomos na população).
    - ✓ Agora todos os cromossomos têm uma chance de serem selecionados. Entretanto, este método pode resultar em menor convergência, porque os melhores cromossomos não se distinguem muito dos outros.
- Métodos de remapeamento explícito para evitar convergência prematura:
    - ✓ Escalamento de Aptidão (Fitness Scaling): define-se um valor máximo de oportunidades de reprodução (típico 2,0) subtraindo uma quantidade fixa da aptidão e dividindo pela média das aptidões modificadas. Aumenta a razão entre a aptidão máxima e a média.

- ✓ Janelamento de Aptidão (Fitness Windowing): subtrai o valor da mínima aptidão observada nas últimas n gerações e divide pela média das aptidões modificadas.
- ✓ Ordenamento de Aptidão (Fitness Ranking): ordenamento linear ou exponencial dos indivíduos de acordo com seu valor bruto de aptidão. Normaliza a razão entre a aptidão máxima e a média e é menos sensível a indivíduos com valores extremos de aptidão (seleção por classificação é um exemplo).

❑ Remapeamento Implícito de Aptidão (Implicit Fitness Remapping)

- Grupo de reprodução é preenchido sem o passo intermediário de remapear a aptidão.
  - ✓ Seleção por Torneio (Tournament Selection):

    Seleciona aleatoriamente um grupo de n indivíduos da população original e copia o individuo com melhor aptidão no grupo de reprodução. Todos os indivíduos são devolvidos à população original. O processo se repete até encher o grupo de reprodução.
  - ✓ Seleção por Torneio Probabilista (Probabilistic Tournament Selection): idem ao anterior, mas o melhor individuo vence o torneio com probabilidade p ($0,5 < p < 1,0$).

2.2.2.4. Lacunas entre gerações

A Lacuna entre gerações (generation gap) é a proporção dos indivíduos da população que são substituídos em cada geração (taxa substituídos / total). Trabalhos mais antigos usam um valor de 1 para o gap (toda a população é substituída por descendentes em uma geração). Para melhorar isso pode-se utilizar:

- Substituição de estado estacionário (steady-state replacement): somente alguns indivíduos são substituídos em cada geração.

- Elitismo: pelo menos uma cópia sem alterações da melhor solução da geração anterior é passada para a nova população, de forma que a melhor solução possa sobreviver às sucessivas gerações. Este pode ser um modelo melhor do que ocorre na natureza (pelo menos em espécies superiores), onde pais podem auxiliar os filhos, mas também competem com eles.

A Substituição de estado estacionário requer seleção não só de reprodutores, mas de indivíduos a morrer os métodos mais usuais são:

- Seleção de reprodutores por aptidão e seleção de indivíduos a morrer de forma aleatória;
- Seleção de reprodutores de forma aleatória e seleção de indivíduos a morrer por aptidão inversa (mais baixa);
- Seleção de reprodutores por aptidão e seleção de indivíduos a morrer por aptidão inversa.

No algoritmo steady state Figura 2-4, poucos membros reproduzem a cada ciclo e seus descendentes substituem os piores indivíduos, introduzindo assim a idéia da superposição (overlapping) de gerações. Esse método é eficiente se cuidados forem tomados para que não ocorra duplicação de indivíduos.

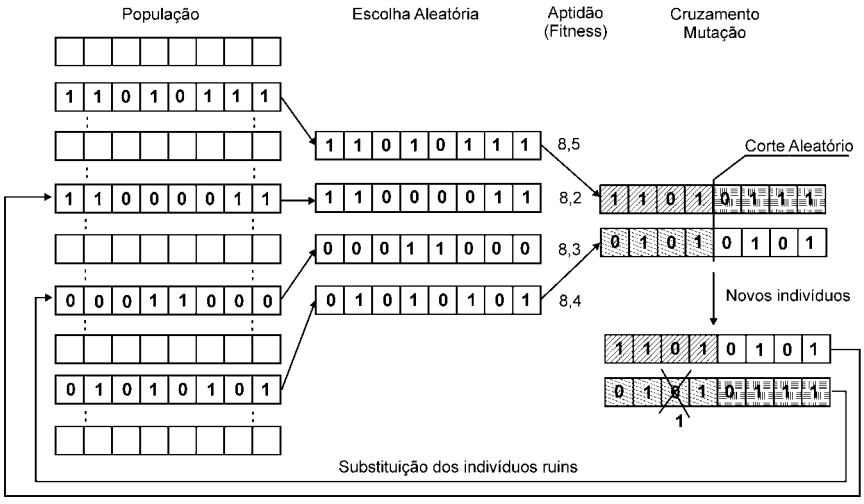


Figura 2-4 – Algoritmo Steady State

2.3. Eletrônica Evolucionária

O eixo filogenético de sistemas de hardware bio-inspirados, forma o núcleo da eletrônica evolucionária (Romahi,1998]. A principal idéia que está por trás da eletrônica evolucionária, é que cada circuito eletrônico factível pode ser representado como um indivíduo ou um cromossomo de um processo evolucionário, o qual realiza operações genéticas padrão sobre o circuito tais como: seleção, crossover e mutação.

O eixo pode ser visto como tendo duas áreas principais, intrínseco e extrínseco como é mostrado na Figura 2-5.

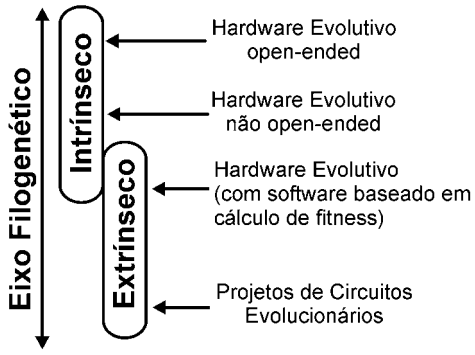


Figura 2-5 – Eixo filogenético

Na base do eixo, existe o método extrínseco puro, ou mais corretamente, projeto de circuito evolucionário. Isso se refere no caso onde algoritmos evolucionários são aplicados ao projeto de circuitos. Assim o processo é realizado na simulação onde somente com a solução final teremos o circuito a ser construído. O passo seguinte está situado no domínio do projeto intrínseco, onde a aptidão (fitness) é testada no hardware real. No entanto, a evolução não é “open-ended”, isto é, existe um objetivo pré-definido e que não interage com o ambiente. O topo do eixo é onde a população de entidades de hardware evoluem em um ambiente “open-ended”, isto é, existe uma interação com o ambiente proporcionando melhorias no circuito.

2.3.1. Projeto Extrínseco

O projeto extrínseco marcou o início do campo da eletrônica evolucionária. É na essência a aplicação de metodologias evolucionárias no projeto de hardware em simuladores. O método consiste em usar simuladores de software para avaliar circuitos evolucionários. No final do processo evolucionário, a solução pode então ser construída convencionalmente ou carregada em um circuito integrado programável.

O principal objetivo é a criação de uma ferramenta automática, baseada em métodos evolucionários para projetos de circuitos eletrônicos, sem a necessidade de alguma espécie de intervenção humana, de tal maneira que a técnica evolucionária possa substituir o projetista de circuito. Utilizando este método, o projetista deverá somente se preocupar com a especificação do sistema, deixando o resto por conta do processo evolucionário.

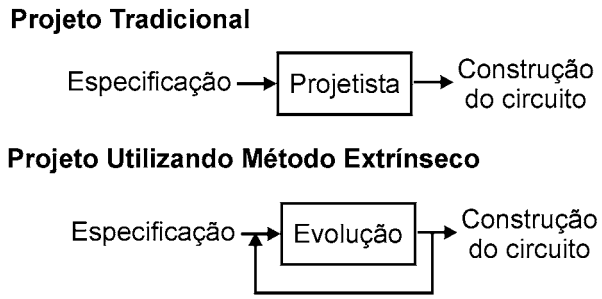


Figura 2-6 – Projeto Extrínseco

2.3.2. Projeto Intrínseco

O projeto intrínseco utiliza uma das principais metodologias usadas no campo da eletrônica evolucionária, sendo que a principal vantagem desse método é não ser necessário o desenvolvimento de modelos de circuito, porque o processo evolucionário se realiza diretamente no circuito. Assim, a solução final irá se comportar exatamente como previsto.

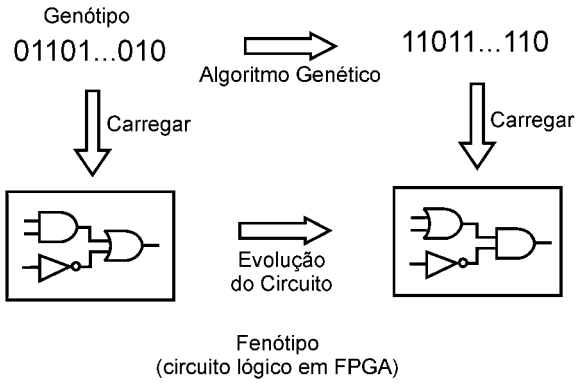


Figura 2-7 – Projeto Intrínseco

2.4. Plataformas para Evolução intrínseca de Circuitos

2.4.1. Plataformas Evolucionárias Digitais

Com o advento do FPGA (Field Programmable Gate Array), a evolução digital intrínseca pode se propagar mais intensamente, por causa das facilidades oferecidas por esse tipo de circuito integrado. A Figura 2-8 demonstra de maneira simplificada as células internas do circuito FPGA. Cada célula contém uma unidade funcional que pode ser configurada para desempenhar alguma função booleana. A saída de cada célula pode ser configurada para ser ligada à uma saída do circuito ou para se conectar à outra célula vizinha de acordo com os bits fornecidos pelo algoritmo.

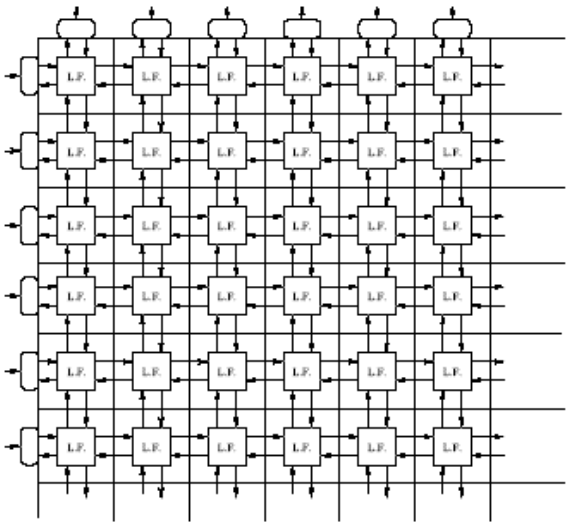


Figura 2-8 – Diagrama de bloco simplificado de um FPGA

É importante notar que nenhuma configuração de células pode causar avaria no circuito, simplificando assim de sobremaneira, a aplicação do algoritmo genético não necessitando de pré-verificação dos bits gerados.

2.4.2. Plataformas Evolucionárias Analógicas

Na síntese de circuitos, os progressos mais notáveis têm ocorrido na área digital. Na área dos circuitos analógicos os avanços têm sido mais modestos. Para a síntese de circuitos digitais são utilizados basicamente a representação por portas lógicas e a representação por funções. No primeiro caso são utilizadas macro-células (AND, NAND, OR, NOR, XOR), no segundo caso são utilizadas funções digitais mais complexas, tais como: unidade lógica aritmética ou um produto de variáveis lógicas.

Ao contrário dos circuitos digitais, onde o comportamento das células é fixado por níveis lógicos discretos, os componentes básicos dos circuitos analógicos variam seu comportamento continuamente com o ponto de operação, produzindo uma vasta gama de resultados diferentes. Há cerca de duas décadas métodos mais modernos de síntese de circuitos analógicos utilizando as técnicas de otimização heurísticas (Sussman,1975) tais como: os sistemas especialistas, o simulated annealing foram introduzidas, culminando mais recentemente com as técnicas da Computação Evolucionária utilizando algoritmos genéticos.

O FPAA (Field Programmable Analog Array), é um circuito integrado (similar ao FPGA), que pode ser configurado para implementar várias funções analógicas, usando um conjunto de blocos analógicos configuráveis (BAC) e uma rede programável de interconexão desses blocos (Figura 2-9).

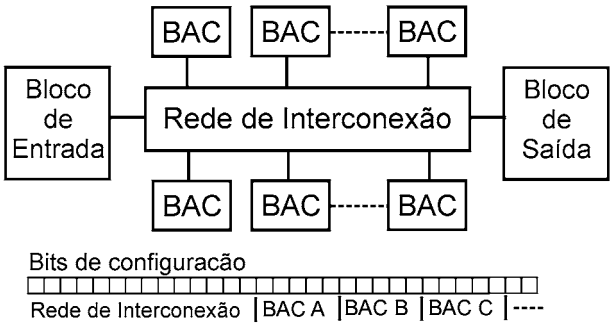


Figura 2-9 – Diagrama de blocos simplificado de um FPAA

Cada BAC pode implementar algumas funções analógicas tais como: amplificação; integração; diferenciação; adição; subtração; multiplicação; comparação; logaritmo e exponencial. O que o torna atrativo para a evolução intrínseca é que o circuito pode ser programado através de bits. No entanto, vale ressaltar que os circuitos comerciais como o Motorola MPAA020 ou o Zetex TRAC020 não foram projetados com o intuito de servir à plataformas evolucionárias. O circuito integrado da Motorola possui estruturas que podem ser danificadas dependendo da configuração de programação. O circuito da Zetex requer suporte externo de chaves e outros componentes dificultando o seu uso. Especificamente para uso em eletrônica evolucionária tem-se o circuito integrado da universidade de Edinburgh (PALMO) voltado para plataformas intrínsecas e que consiste numa série de circuitos integradores programados via PWM (Pulse Width Modulation).

Uma outra proposta para evolução intrínseca é a PAMA (Programmable Analog Multiplexer Array) (Santini,2001) da PUC (Pontifícia Universidade Católica), que introduz o conceito de barramento analógico para interligar componentes discretos através de multiplexadores e demultiplexadores (Figura 2-10).

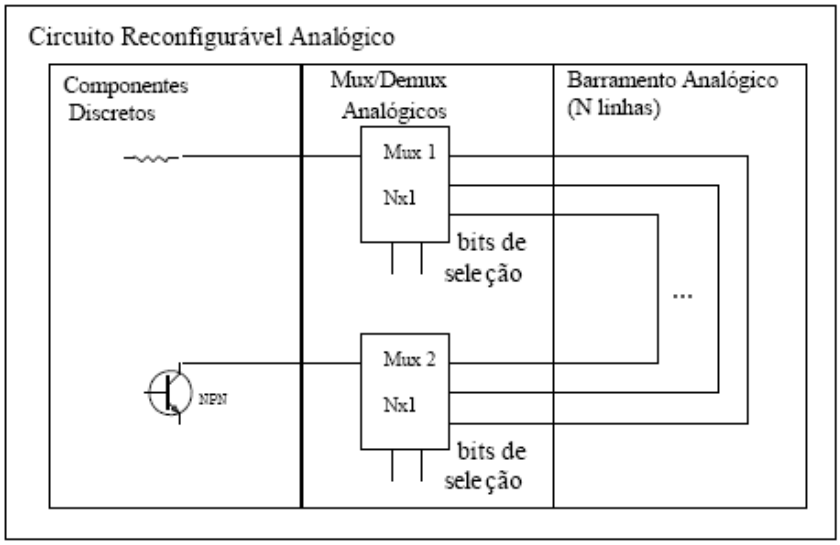


Figura 2-10 – Diagrama de blocos PAMA

Um outro tipo de circuito integrado que surge para uso em ambientes intrínsecos é o FPTA (Field Programmable Transistor Array). Os circuitos FPAAs, como visto anteriormente, interligam blocos analógicos que podem ser amplificadores operacionais, integradores, ou blocos de componentes passivos. Para o FPTA os blocos são compostos por transistores programáveis que se interligam através de chaves analógicas. Dois tipos de circuitos apresentam essas características, o da Universidade de Heidelberg (Kirchoff Institute) Figura 2-11e o da JPL (Jet Propulsion Laboratory – NASA) Figura 2-12.

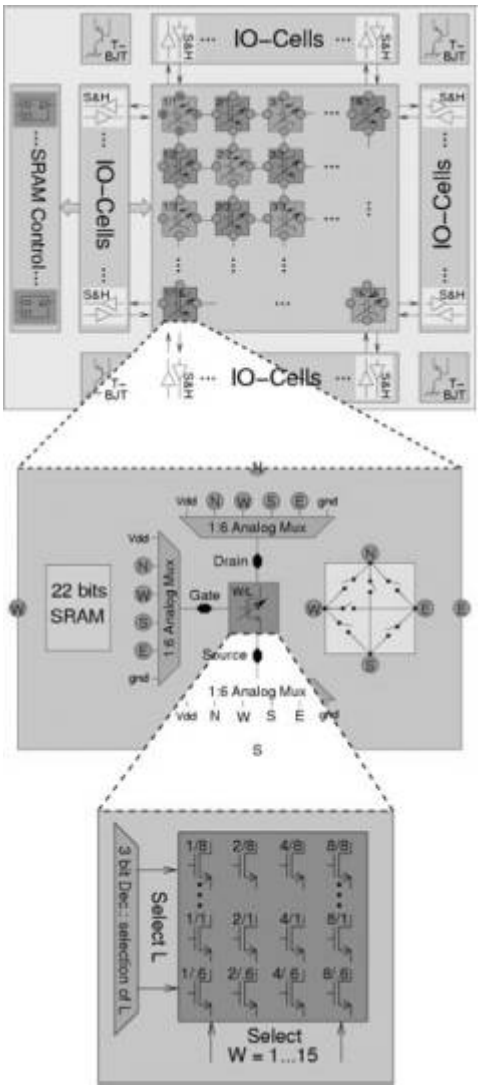


Figura 2-11 - FPTA da Universidade de Heidelberg

O circuito integrado da Universidade de Heidelberg consiste de uma matriz de 256 células de transistores programáveis. Cada célula permite 75 combinações de tamanho (W e L) do transistor. Qualquer terminal da célula pode se conectado a um outro terminal da célula vizinha, seguindo a ordem Norte, Sul, Leste e Oeste através de multiplexadores analógicos. É interessante ressaltar que os transistores não utilizados no interior da célula são desligados, porém os transistores de uma linha da matriz têm em comum a ligação da porta (gate), e portanto na ligação de dois ou mais transistores temos um aumento da capacitância de porta (gate), que influi no desempenho do transistor.

O circuito integrado JPL Figura 2-12 consiste de uma matriz de 6x6 formada por células base. A célula base (Figura 2-12a) é composta por oito transistores e vinte e quatro chaves com algumas ligações pré-estabelecidas, tendo disponíveis duas entradas e duas saídas. A matriz pode ser dividida em células de fronteira e células centrais. As células de fronteira são em número de vinte e recebem sinais de entradas externas (In1 à In20). Estas células podem opcionalmente, serem usadas para passar o sinal de entrada para as células centrais, ou para rotear o sinal de saída para os amplificadores isoladores (buffers) de saída (total de 10 buffers). Todas as células de fronteira têm suas saídas voltadas para as células centrais, exceto as células dos vértices CB1, CB2, CB3 e CB4, cujas saídas vão para seus vizinhos (CB5 à CB12) como mostra a Figura 2-12b. Cada célula pode receber como entrada, as saídas dos seus vizinhos Norte, Sul, Leste e Oeste. As oito células internas são as únicas que apresentam capacitores que podem ser programados. O uso de diferentes razões de W/L para os transistores dá ao circuito maior versatilidade para otimização.

O circuito é dividido em nove grupos de quatro células, apresentando a seguinte razão W/L em μm: 1,2/1,2; 2,4/1,2; 4,8/1,2; 2,4/2,4; 4,8/2,4; 9,6/2,4; 4,8/4,8; 9,6/4,8; 19,2/4,8.

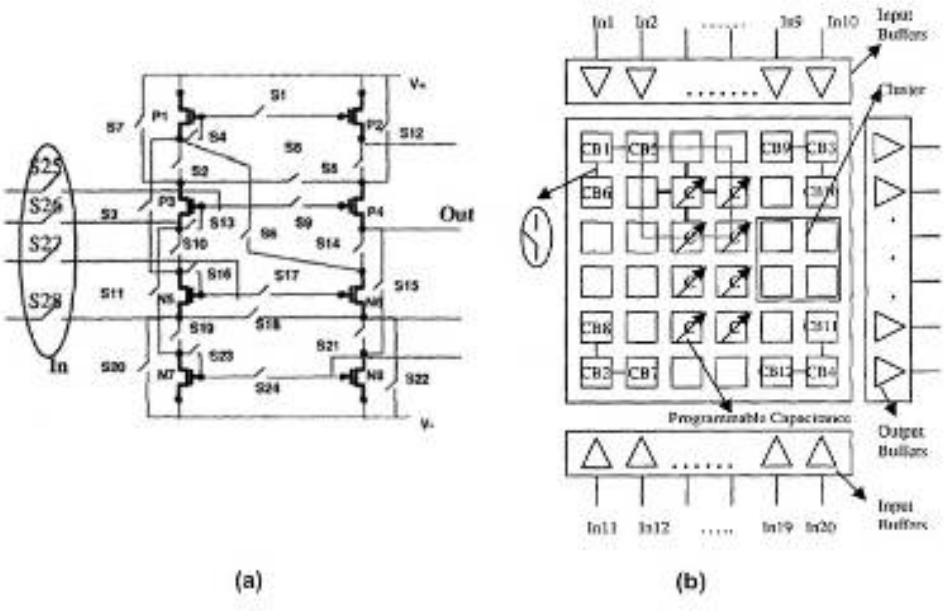


Figura 2-12 – FPTA –JPL a) Célula base b)Vista global

3. Especificações Gerais de Projeto de um Circuito para EHW

Com base no que a literatura fornece sobre plataformas intrínsecas para circuitos analógicos, algumas características podem ser evidenciadas e utilizadas no projeto de um circuito para hardware evolucionário.

- Bloco básico – Do circuito proposto pelo JPL (NASA) o número de oito transistores para definir uma célula foi incorporado no projeto, porém os transistores de tamanho fixo dão pouca flexibilidade para formação de topologias diversas (clássicas ou não). O transistor fixo foi então substituído por transistores programáveis (LANGEHEINE,2001) seguindo a proposta da universidade de Heidelberg, porém com os cuidados de performance para não tornar o circuito extremamente lento.
- Interconexão dos Transistores Programáveis – A interligação dos transistores do circuito da NASA através de chaves e ligações pré-estabelecidas que induzem à estruturas clássicas, limita a possibilidade de formação de topologias diversas. O circuito de Heidelberg interliga os transistores programáveis seguindo uma rota Norte, Sul, Leste e Oeste que também não flexibiliza o circuito quanto à topologias. As limitações atuais, citadas anteriormente como o roteamento Norte, Sul, Leste e Oeste, bom para circuitos digitais não sendo aconselhável para circuitos analógicos, são resolvidas pelo grupo do ICA (PUC) com a proposta de utilização de um barramento analógico.
- Proteção contra avaria da célula programável – Devido à flexibilidade de interligação das células ligações indevidas devem ser evitadas para que não ocorram danos nos transistores.
- *Escalabilidade* – O circuito integrado deverá ser construído, de maneira que com o aumento da complexidade da função objetivo, módulos internos possam ser adicionados aumentando assim, a capacidade de evolução do circuito.

- Área disponível do circuito integrado – Neste caso é um fator limitante para dimensionamento das células e do barramento hierárquico que demanda um grande número de chaves. A área útil disponível do circuito integrado é de 3,24 mm^2 (1,8µm x 1,8 µm) como mostra a figura

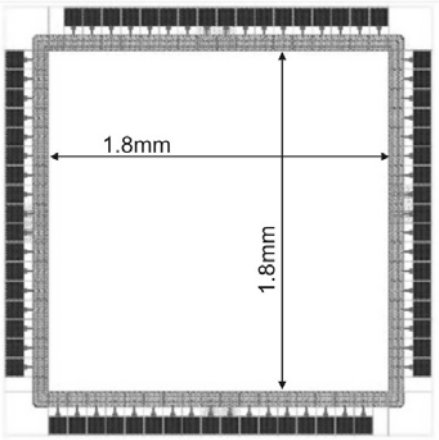


Figura 3-1- Área útil do circuito integrado de 68 pinos

3.1. Considerações sobre o Barramento

O número de nós de alguns circuitos clássicos em conjunto com a quantidade de transistores utilizados, fornece uma direção no número de colunas (vias) do barramento que pode ser utilizado, uma vez que cada coluna acrescentada representa maior quantidade de chaves no circuito.

A célula programável tem um tamanho estimado de 40µmx20µm (0,0008mm^2). Os valores de L e W podem variar de L=0,35µm;W=0,7µm à L=5,5µm;W=10,5µm. Portanto, para que as chaves não causem maiores influências nas interligações do circuito, estas não devem ter valor menor que o maior transistor programado pode alcançar. O tamanho da chave utilizada para o transistor NMOS é de: L=0,35µm;W=5µm e para o transistor PMOS é de: L=0,45µm;W=15µm. A área estimada de ocupação da chave é de 9µmx21µm (0,0002 mm^2).

Podemos notar em um circuito amplificador operacional clássico Figura 3-2, uma incidência de nove nós de conexão entre terminais, e sete transistores.

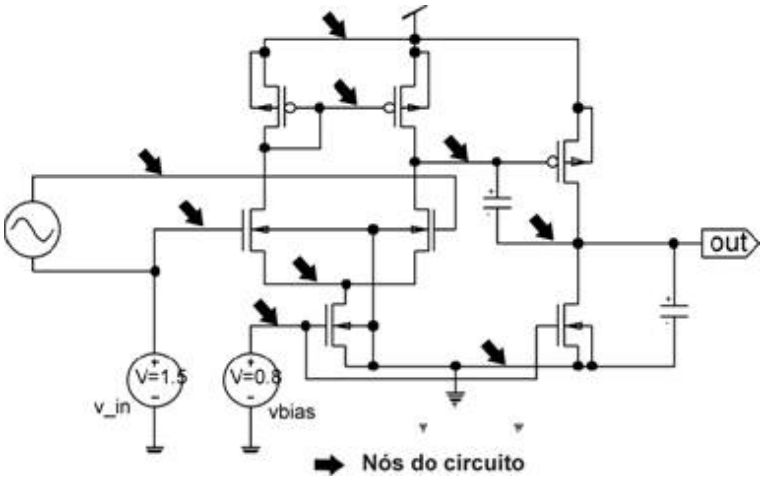


Figura 3-2– Amplificador operacional

No amplificador de transcondutância da Figura 3-3, dez nós e nove transistores são utilizados.

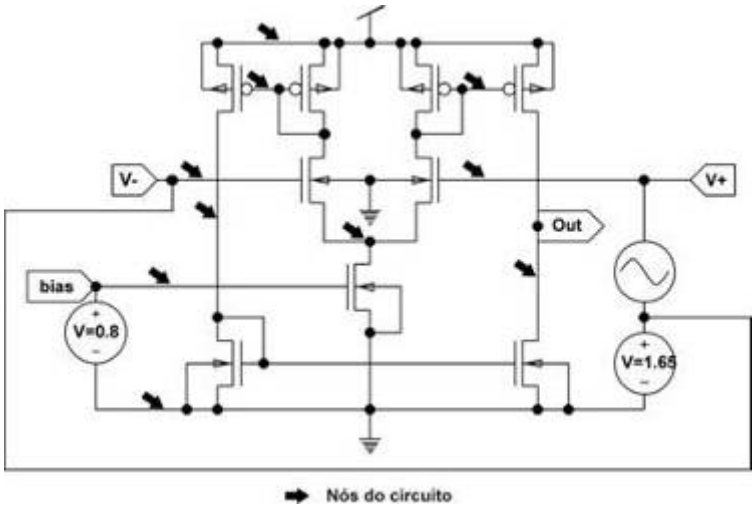


Figura 3-3 – Amplificador de Transcondutância (OTA)

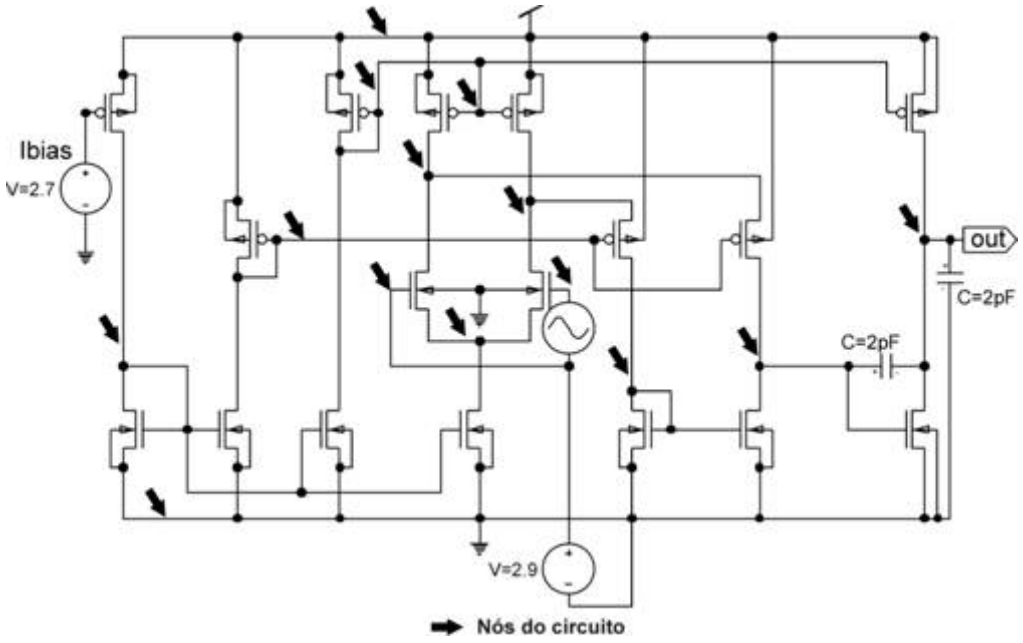


Figura 3-4 – Amplificador Operacional de alto ganho

Em circuitos mais elaborados como o da, Figura 3-4 dezessete transistores e quatorze nós foram utilizados. Geralmente como em alguns circuitos mais usuais são utilizados sete a nove transistores e usando como referência o circuito da NASA (Stoica,1999) com oito transistores, optou-se em formar blocos de oito transistores programáveis, quatro do tipo PMOS e quatro do tipo NMOS. Tomando por base os circuitos clássicos apresentados, em especial o amplificador operacional, oito colunas que representam os nós do circuito, tornam-se insuficientes para suprir todas as ligações. Então numa relação de compromisso em permitir maiores possibilidades de interligação, simplicidade de circuito para decodificação, armazenamento de dados (Latchs) e proteção, optou-se em dezesseis vias ou colunas por bloco formando assim o barramento local. Para Interligar o barramento local a outros blocos, um barramento externo foi acrescentado.

3.2. Metodologia de Validação

Para verificar a influência das chaves no desempenho do circuito, é feita inicialmente uma comparação entre o desempenho de um circuito de referência com topologia clássica, sem chaves, e o desempenho do mesmo circuito transposto para a

arquitetura proposta de barramento. Nesta etapa não é utilizado o ambiente de teste desenvolvido. A seguir a ferramenta de teste desenvolvida é utilizada em processos evolucionários simulados utilizando a configuração de barramento proposta com os transistores programáveis.

No primeiro teste utilizou-se circuitos analógicos clássicos e suas análises extraídas do simulador Spice, fez-se então a transposição para o modelo de barramento proposto, obtendo-se basicamente o mesmo circuito, porém, cada terminal da célula de transistor passou a ter associada a ela uma chave analógica.

Para o segundo teste foi utilizado o simulador para testes das estruturas propostas. Devido à natureza lenta deste processo, optou-se em utilizar nove máquinas disponíveis no laboratório, sendo então necessário desenvolver um programa tendo como função executar um algoritmo genético do tipo “steady state” e gerenciar os arquivos que são enviados para as máquinas auxiliares. Para as máquinas auxiliares foi desenvolvido um programa que recebe os arquivos para execução no simulador e devolve os resultados para o computador principal.

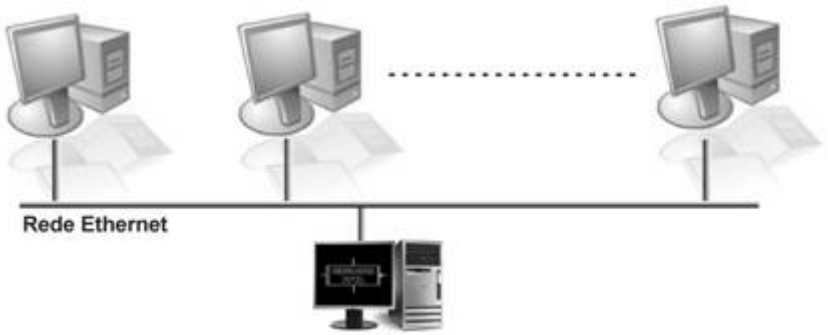


Figura 3-5 – Arquitetura de Hardware utilizada para Evolução Extrínseca

4. Detalhamento do Projeto Arquitetura Geral

4.1. Estruturas Utilizadas

Neste item as estruturas que irão compor o barramento, iniciando com o transistor MOS FET, serão examinadas de forma conceitual. As principais características das portas lógicas, chaves complementares, decodificadores e elementos de memória (Latchs) serão discutidas.

4.1.1. O Transistor MOS FET

Circuitos integrados CMOS (*Complementary Metal-Óxide Semiconductor*) utilizando Transistores de Efeito de Campo *(Field Effect Transistors* - FET) são o resultado de um avanço tecnológico que começou na década de 1930 e se tornou viável somente nos anos de 1960. A partir de 1970, com o surgimento do transistor MOSFET (Metal Oxide Semiconductor Field Effect Transistor), foi possível atingir a complexidade e a densidade dos atuais circuitos VLSI *(Very Large Scale of Integration)*. Atualmente cerca de 75% dos circuitos semicondutores são implementados em tecnologia CMOS, não se prevendo alterações significativas neste panorama nos próximos 10 a 20 anos. As principais vantagens apresentadas pela tecnologia CMOS são o baixo consumo de potência, alta imunidade a ruído, alto nível de integração, simplicidade de projeto e operação confiável em ampla faixa de valores de tensão.

4.1.2. Características do Transistor MOS

O comportamento dos transistores MOS varia segundo o tipo de estrutura usada na sua fabricação. Os três principais modos de operação são:

- Enriquecimento (*enhancement*),
- Depleção (*deplection*) e,
- Enriquecimento-depleção.

Os circuitos lógicos MOS convencionais são em geral implementados apenas com transistores operando no modo enriquecimento. Além disso, a tecnologia CMOS utiliza transistores complementares operando no modo enriquecimento; os transistores NMOS (MOSFET canal N) e PMOS (MOSFET canal P), cujos símbolos esquemáticos são mostrados na Figura 4-1 que se distinguem pelo tipo de portador majoritário responsável, em cada caso, pelo mecanismo de condução.

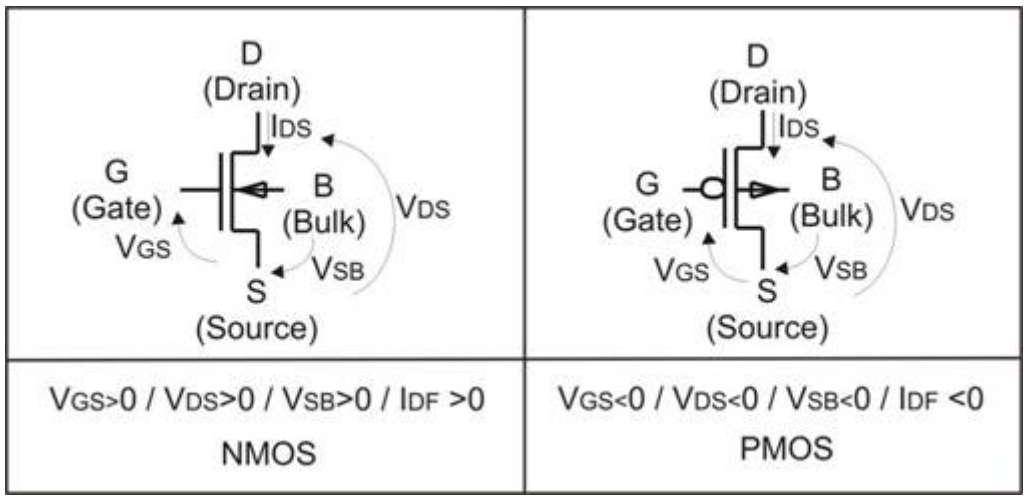


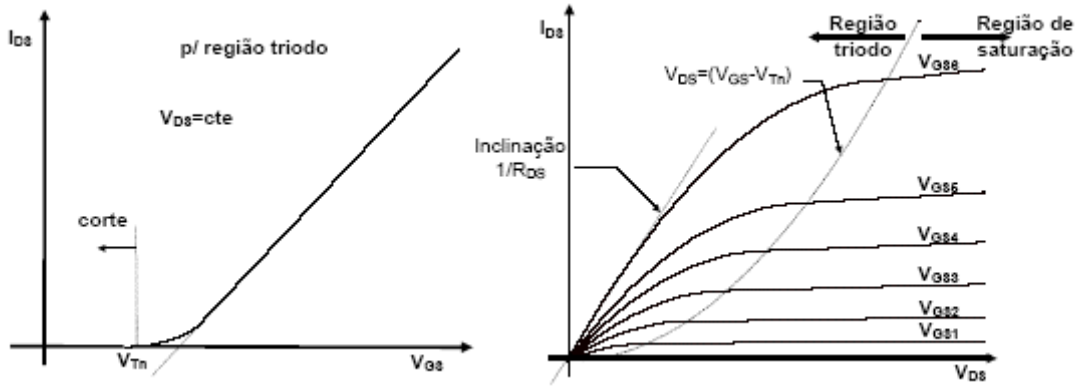
Figura 4-1 – Símbolos dos Transistores MOSFET

Os transistores CMOS podem operar em quatro regiões distintas; corte, sublimiar, triodo ou linear, e saturação.

Na região de corte, a tensão $V_{GS}<0$ não permite a formação do canal de condução caracterizando o comportamento do transistor como chave aberta porque, neste caso, a corrente I_{DS} é nula, independentemente da tensão de dreno. Na região de sublimiar, $0 \leq V_{GS} < V_T$, o transistor apresenta comportamento semelhante ao dos transistores bipolares, sendo a corrente de dreno modelada por equações fortemente não lineares. Nesta região o transistor é usado em aplicações específicas onde a baixa dissipação de potência é requisito essencial de projeto. Na região triodo, $V_{DS} < V_{GS} - V_T$, o transistor se comporta como uma resistência controlada por tensão como é mostrado

na Figura 4-2. A equação que relaciona tensão e corrente num NMOS na região triodo é:

$$I_D \cong K\frac{W}{L}\left[\left(V_{GS}-V_T\right)V_{DS}-\frac{{V_{DS}}^2}{2}\right] \qquad (1)$$



(a) Região de operação linear (b) Característica I_{DS}xV_{DS} completa

Figura 4-2 – Curvas características de NMOS no modo enriquecimento

Na região de saturação, $V_{DS} \geq V_{GS}-V_T$, a corrente I_{DS} dada pela Eq. (1) atinge o valor máximo, para um dado valor de V_{GS}, quando $V_{DS}=V_{GS}-V_T$. A partir deste valor de V_{DS} a corrente de dreno permanece aproximadamente constante caracterizando o transistor como uma fonte de corrente controlada por V_{GS}.

A equação que descreve aproximadamente o comportamento de I_{DS} neste caso é:

$$I_D \cong \frac{K}{2}\frac{W}{L}\left(V_{GS}-V_T\right)^2 \qquad (2)$$

Devido ao efeito da modulação do comprimento do canal, o transistor não se comporta como fonte de corrente ideal, como sugerido pela Eq. (2), sendo a corrente

ainda influenciada pela variação de V_{DS}, para considerar esta dependência a equação anterior é modificada para:

$$I_D \cong \frac{K}{2}\frac{W}{L}\left(V_{GS} - V_T\right)^2 (1 + \lambda V_{DS}) \tag{3}$$

Onde λ é o fator de modulação do comprimento de canal.

As equações anteriores consideram os terminais de fonte e substrato aterrados, porém em algumas estruturas tal como a chave complementar isto não ocorre, fazendo com que seja necessário acrescentar o efeito da diferença de potencial entre fonte e substrato V_{SB} que altera o valor de V_T.

$$\Delta V_{T_n} = \gamma(\sqrt{2\left|\Phi_p\right| + V_{SB}} - \sqrt{2\left|\Phi_p\right|}\ \textit{onde}:$$
$$\gamma = \frac{\sqrt{2\varepsilon_{Si} q N_a}}{C_{ox}} \text{ e } \Phi_p = -\frac{KT}{q} I_n\left(\frac{N_a}{n_i}\right) \tag{4}$$

Este efeito chamado de efeito de corpo, pode ser visto na condutância dreno-fonte G_{DS}, que varia com a tensão V_{SB} como mostrado na Figura 4-3b.

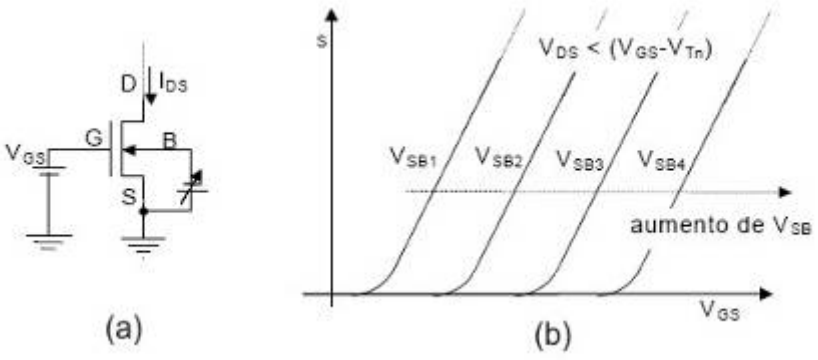


Figura 4-3 – Efeito de corpo em MOSFET a) circuito de polarização, b)G_{DS} x V_{SB}

4.1.3. Inversor CMOS

O inversor CMOS é a porta lógica mais simples implementada com esta tecnologia e será utilizado como componente da estrutura do barramento para o acionamento das chaves complementares, portas, latchs e decodificadores.

É composto por dois transistores NMOS e PMOS, operando de forma complementar.

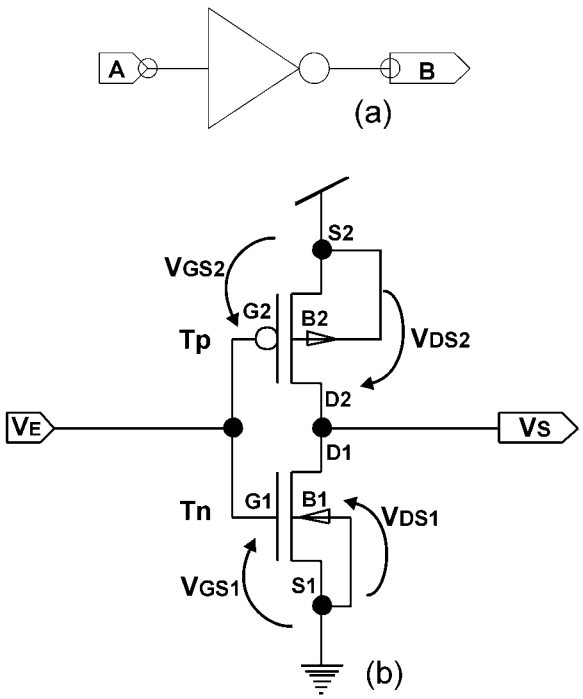


Figura 4-4 Inversor CMOS básico(b) símbolo esquemático– (a)

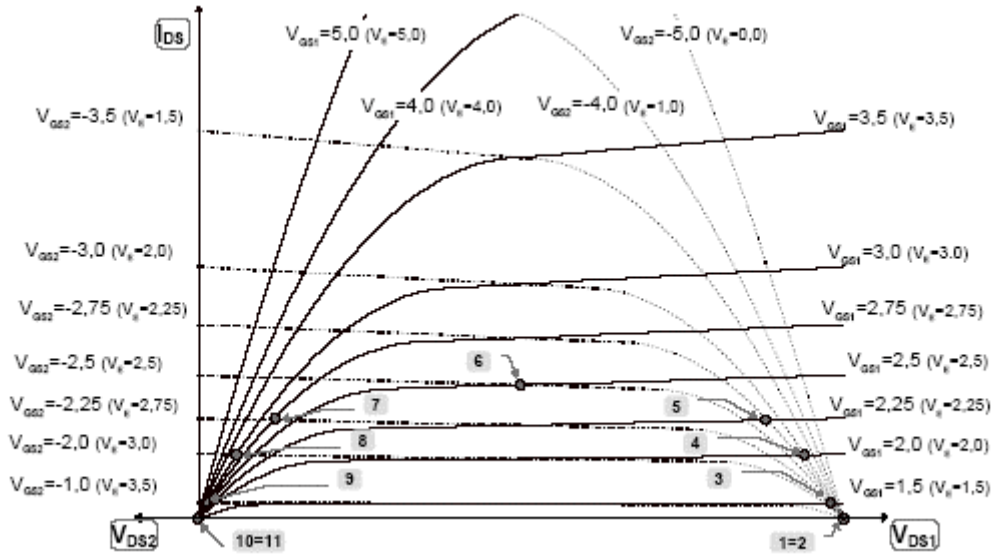


Figura 4-5 – Curvas características de Tn e Tp da figura Figura 4-4

Se a tensão aplicada à entrada do inversor for inicialmente igual a zero, temos a tensão porta-fonte de T_n, V_{GS1}, igual a zero Volts e a tensão porta-fonte do T_P, V_{GS2}, igual a V_{DD}. Portanto o transistor T_n estará cortado e o transistor T_p, em condução na região triodo. Na associação de curvas características mostradas na Figura 4-5, estamos operando no ponto 1, sendo por isso a tensão de saída praticamente igual a V_{DD}. Está situação, que corresponde a região ***A*** da característica de transferência do circuito mostrada na Figura 4-6, permanece até que a tensão de entrada atinja a tensão de limiar do transistor NMOS, V_{Tn} (ponto 2). Quando a tensão de entrada do inversor atinge V_{Tn}, o transistor T_n passa a conduzir na região de saturação permanecendo o transistor T_p na região triodo (pontos 3 e 4 dos gráficos das Figura 4-5 e Figura 4-6) e o circuito passa a operar na região ***B***. Aumentada a tensão de entrada, o transistor T_p passará da região triodo à região de saturação e o circuito passará a operar na região ***C***, onde ambos os transistores estão saturados (pontos 5, 6 e 7). Nesta região o inversor apresenta um ganho entre entrada/saída grande e pode mesmo ser empregado como amplificador. Um pequeno aumento da tensão de entrada faz o transistor T_n entrar na região triodo, e o circuito passará a operar na região ***D*** (pontos 8 e 9). Finalmente quando a tensão de entrada for maior que (V_{DD}-

V_{Tp}) teremos o corte do transistor T_p , ponto 10, permanecendo o transistor T_n na região triodo (região ***E***). Como T_p está cortado a saída fica praticamente igual a zero Volts.

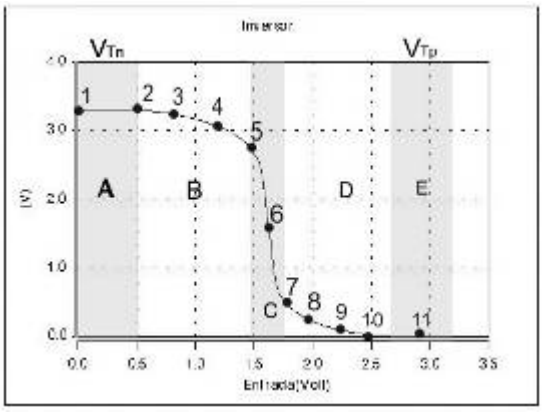


Figura 4-6-Característica de transferência de um inversor CMOS

4.1.4. Chave CMOS Complementar

São usados dois transistores complementares em paralelo para compensar as variações de resistências fonte-dreno provocadas pela variação das tensões V_{GS} e V_{SB} dos transistores. Na Figura 10 temos as curvas de resistência de ambos os transistores e da resistência equivalente da chave em função da tensão de entrada; estas curvas ilustram bem a operação dos dois transistores. Note que a resistência equivalente da associação tem seu valor máximo próximo da metade da excursão da entrada. Outra razão para o uso de dois transistores é permitir que tensões com valores próximos tanto de zero como de VDD possam ser transmitidas pela chave. Neste trabalho foram usados os dois símbolos Figura 4-7b. O símbolo da Figura 4-7b a direita indica uma chave previamente selecionada para os casos de testes que serão apresentados em outro capítulo.

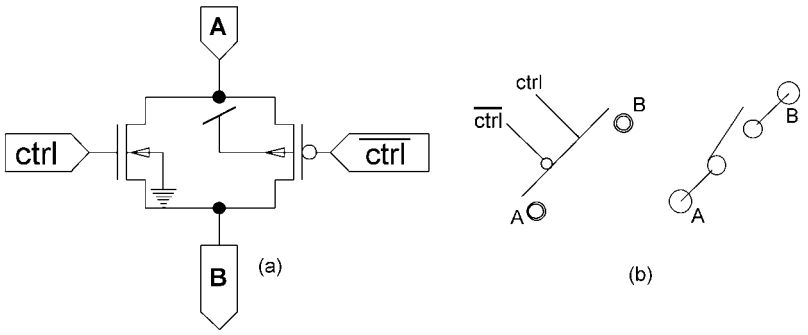


Figura 4-7 – Chave CMOS complementar

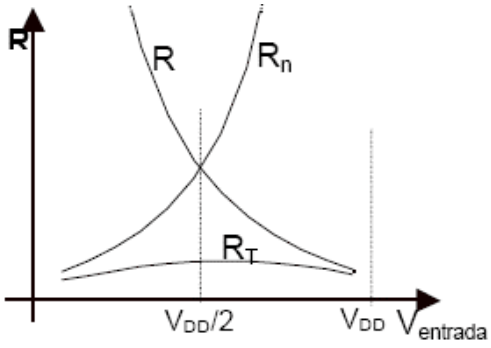


Figura 4-8 – Variação da resistência de uma chave CMOS complementar em função da tensão de entrada

4.1.5. Porta NAND

As portas lógicas NAND podem ser consideradas generalizações do circuito inversor. Esta porta NAND de quatro entradas com saída complementar é usada no decodificador 4x16 do barramento.

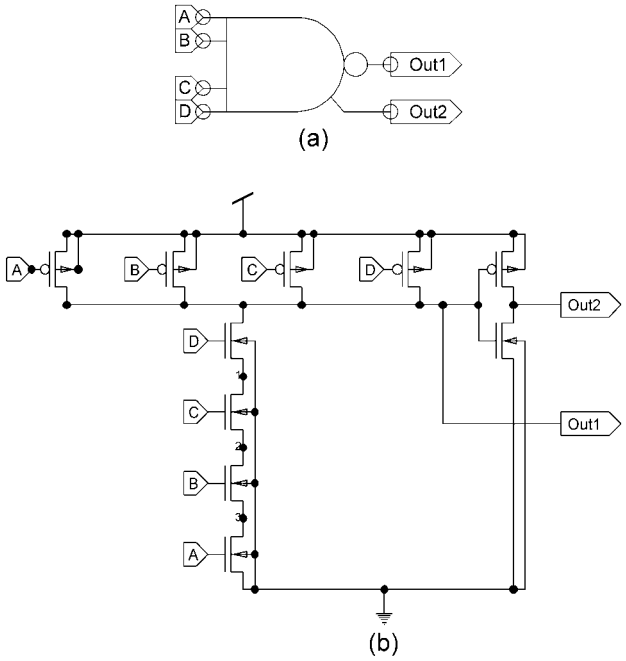


Figura 4-9 – Porta NAND de 4 entradas

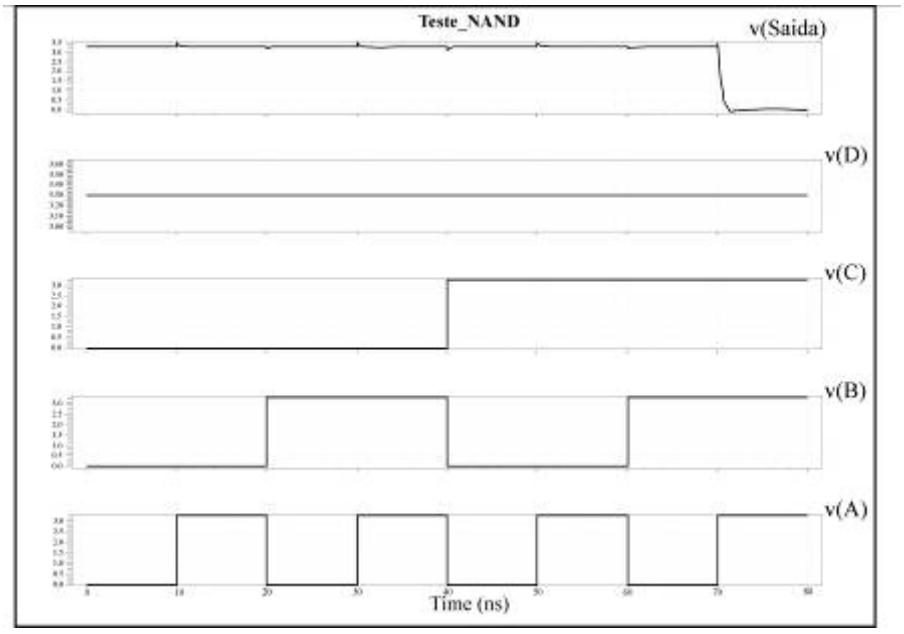


Figura 4-10 –Simulação da porta NAND

4.1.6. Decodificador 4x16

Este decodificador é utilizado no barramento local para selecionar uma das quinze colunas ao terminal da célula. É composto de inversores e porta NAND de quatro entradas.

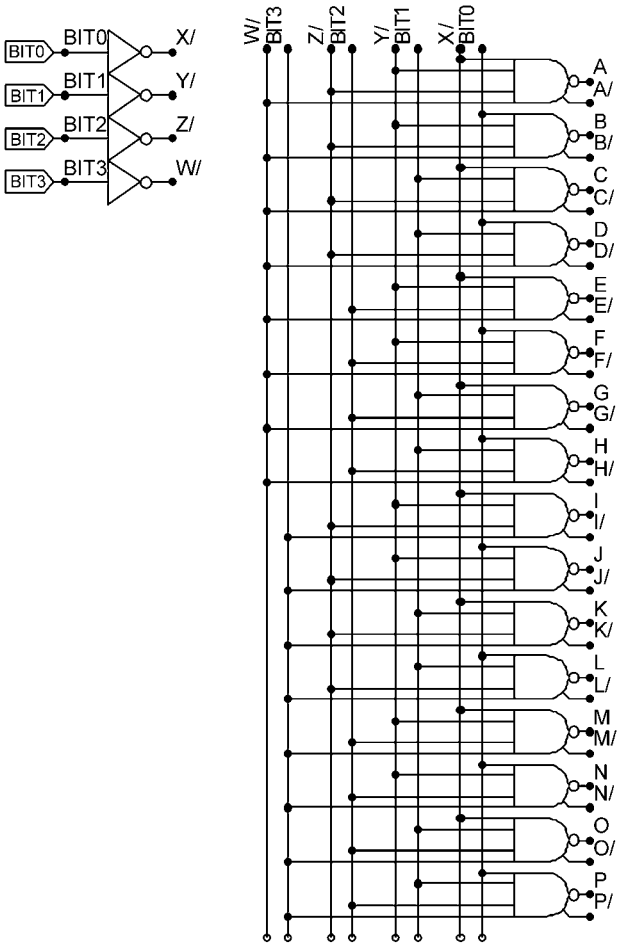


Figura 4-11 – Decodificador 4x16

A Figura 4-12 mostra a simulação do decodificador 4x16, com apenas alguns sinais de saída e com a entrada V(D) ou BIT3 (Figura 4-11) aterrada, para fins de redução do gráfico.

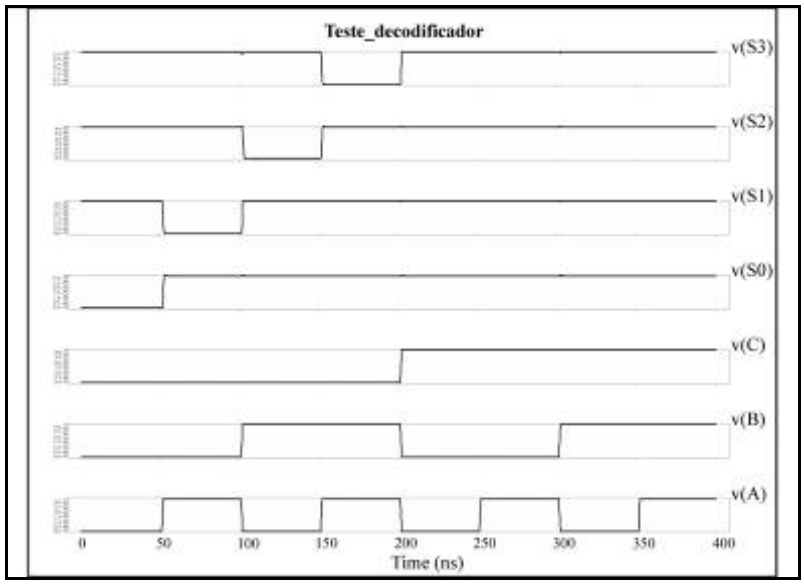


Figura 4-12 - Simulação do decodificador 4x16

4.1.7. Decodificador com controle

Este decodificador é utilizado para ativar os Latchs das células e os Latchs das chaves internas ao bloco.Também é utilizado para selecionar os blocos e os Latchs das chaves do barramento externo. A entrada ENABLE em “1” coloca as saídas de A à P em nível alto e por conseguinte as saídas A/ à P/ em nível baixo.

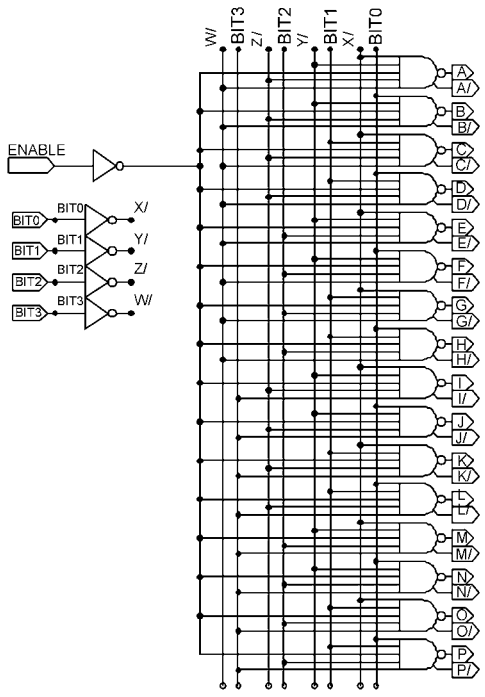


Figura 4-13 – Decodificador 4x16 com controle

4.1.8. Chave SPDT um polo duas posições

A chave SPDT é utilizada no circuito de proteção das células programáveis. Colocando o sinal Ctrl em nível alto o sinal In1 passa para a saída e com o sinal Ctrl em nível baixo a entrada In2 passa para a saída.

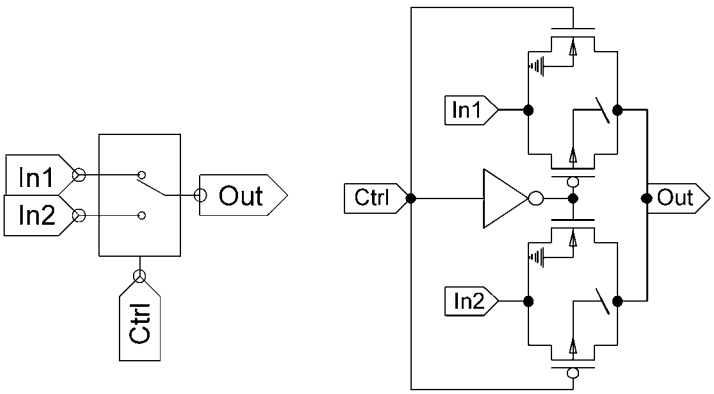


Figura 4-14 – Chave (SPDT) um polo duas posições

4.1.9. Latch

O Latch foi utilizado para armazenar o valor de programação das chaves, fornecido pelo algoritmo genético. O circuito abaixo equivale a um bit e possui uma saída complementar que evitaria o uso de inversor na chave complementar, como pode ser visto na Figura 4-15.

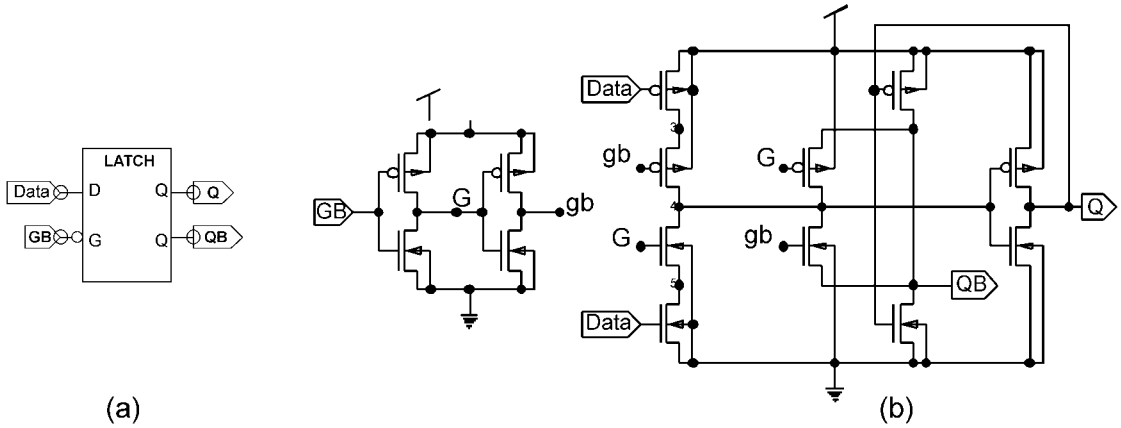


Figura 4-15– Latch 1 bit e saída complementar

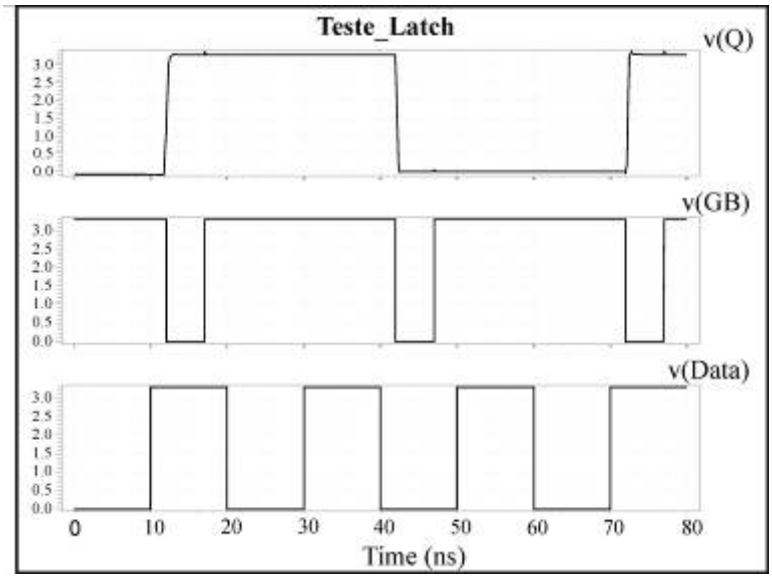


Figura 4-16 – Simulação do Latch

4.2. Proposta de Barramento

Como vimos anteriormente na seção 3, o circuito integrado é subdividido em blocos. Cada Bloco é composto por oito células programáveis de transistores, quatro do tipo NMOS e quatro do tipo PMOS, que se conectam através de um barramento local de quinze vias, de acordo com a Figura 4-17.

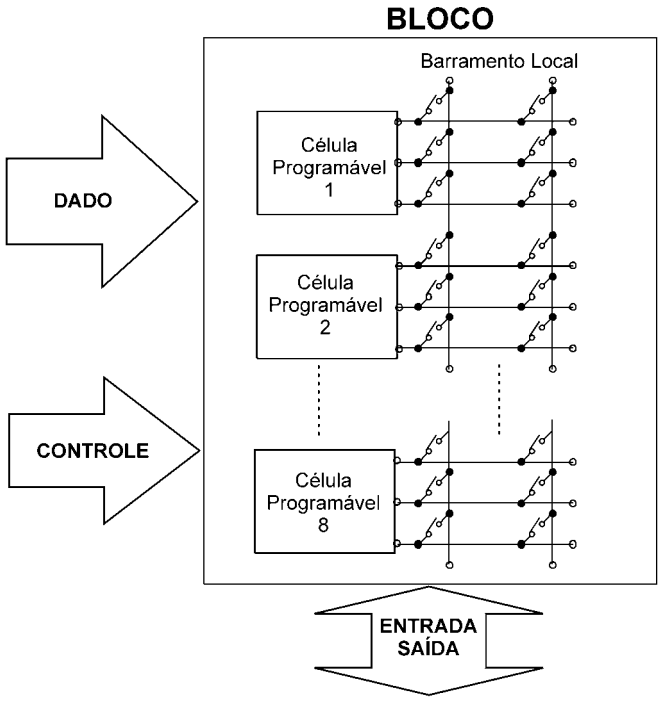


Figura 4-17 – Visão geral do Bloco proposto para o circuito integrado

Cada bloco é programado por uma via de dados (bits do cromossomo) e uma via de controle para seleção dos Latchs respectivos. Oito colunas do barramento local são disponibilizadas para interligação com outros blocos e também como entradas e saídas do circuito. A figura Figura 4-18 fornece uma idéia geral da arquitetura proposta onde podemos ver os blocos sendo interligados a outros blocos através de um barramento externo ao bloco e este barramento sendo interligado à outros barramentos externos de maneira hierárquica.

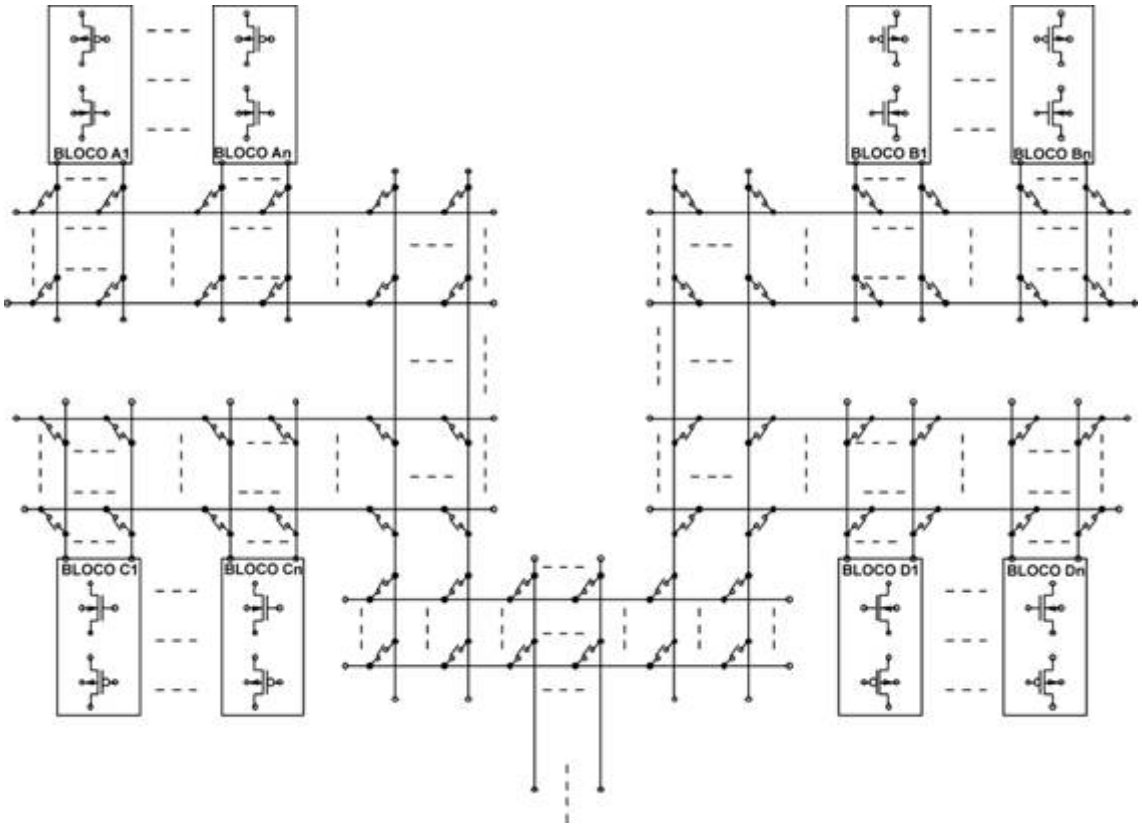


Figura 4-18 – Visão geral do barramento

A seguir será feito um detalhamento dos circuitos que compõe o barramento e sua programação.

4.2.1. Barramento local

As células programáveis necessitam de seis bits de programação e estes bits ficam armazenados em latchs. Cada Latch possui uma entrada “habilita” que é utilizada para “armazenar” o dado na saída.

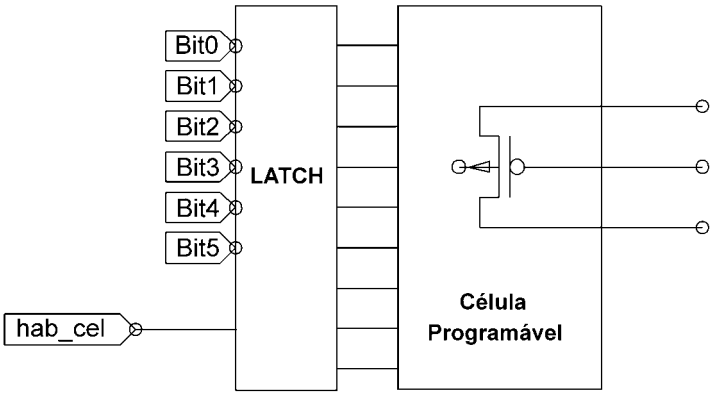


Figura 4-19 – Programação da célula

Cada célula disponibiliza três terminais de conexão do transistor equivalente (dreno, porta, fonte). Estes terminais equivalem às linhas da matriz do barramento.

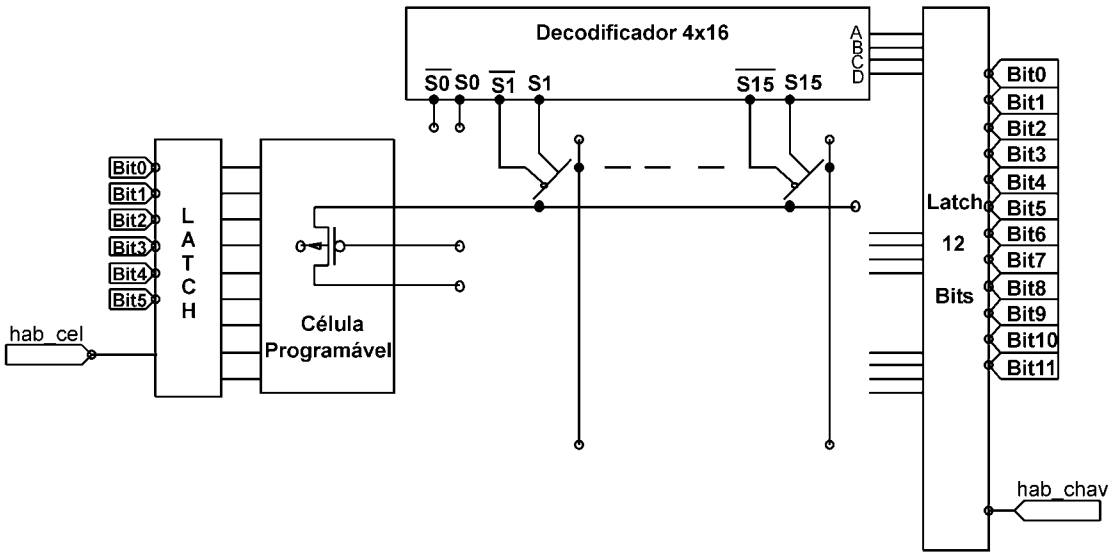


Figura 4-20 – Detalhamento da programação do barramento local

Para cada linha da célula é colocado um decodificador, no caso da Figura 4-20 o decodificador é de 4x16. A saídas S0 do decodificador não são utilizadas para que com a combinação da entrada A,B,C,D em zero, não seja acionada nenhuma chave

do barramento. Sendo um decodificador de quatro entradas, para as três linhas da célula se faz necessário um latch com doze bits. Portanto, para uma célula temos seis bits de programação da célula e doze bits de programação para as chaves. Dois sinais de habilitação são utilizados por célula para armazenar os dados provenientes do algoritmo nos latchs.

4.2.2. Circuito de Proteção

O fato de usar um decodificador para seleção das colunas funciona também como uma proteção, uma vez que impossibilita o curto-circuito na alimentação. Para evitar que uma célula seja avariada por ter o terminal dreno ligado à tensão de alimentação, V_{DD}, e o terminal de fonte ligado à terra, provocando assim uma corrente em excesso, um circuito de proteção é sugerido (Figura 4-21). A colocação destes circuitos será avaliada posteriormente, pois implica em quantidade de transistores e, por conseguinte, diminuição da área útil. Proteção por tratamento de software poderá ser utilizada também.

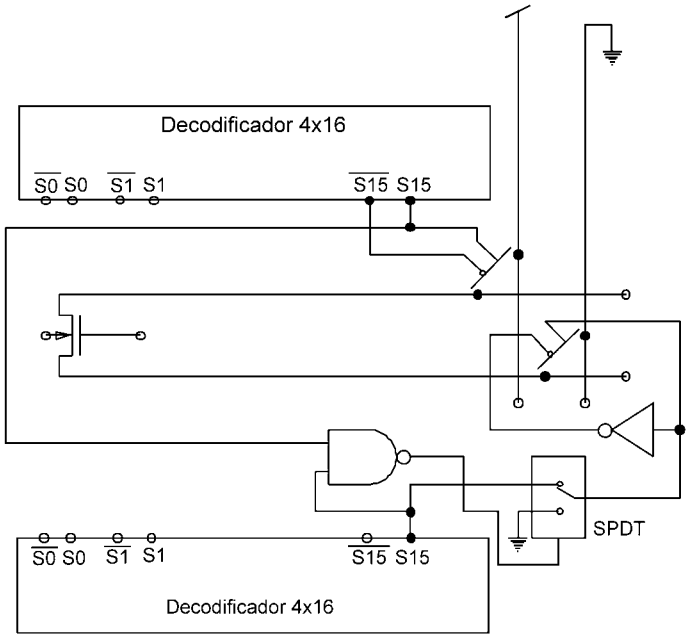


Figura 4-21- Circuito de proteção da interligação das células programáveis

O circuito de teste da Figura 4-22, foi utilizado para simularmos o efeito da proteção dos transistores programáveis e a simulação é apresentada na Figura 4-23. O circuito detecta as saídas S15_s e S15_d em nível alto e deixa somente a saída S15_s atuar evitando que a chave do dreno seja acionada colocando o transistor em risco.

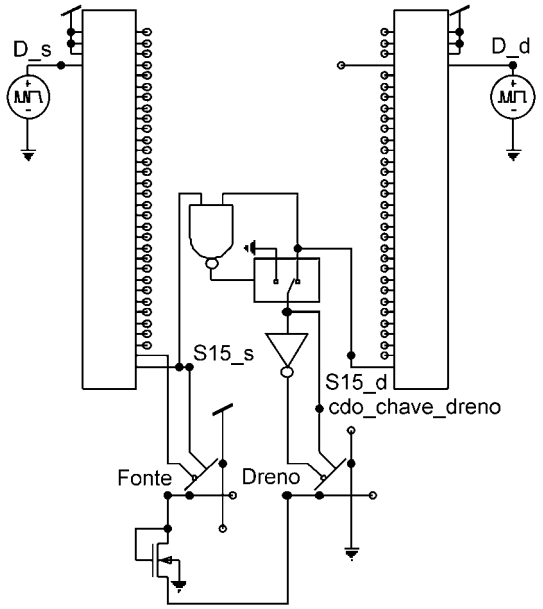


Figura 4-22 – Circuito de teste utilizado para simulação da proteção

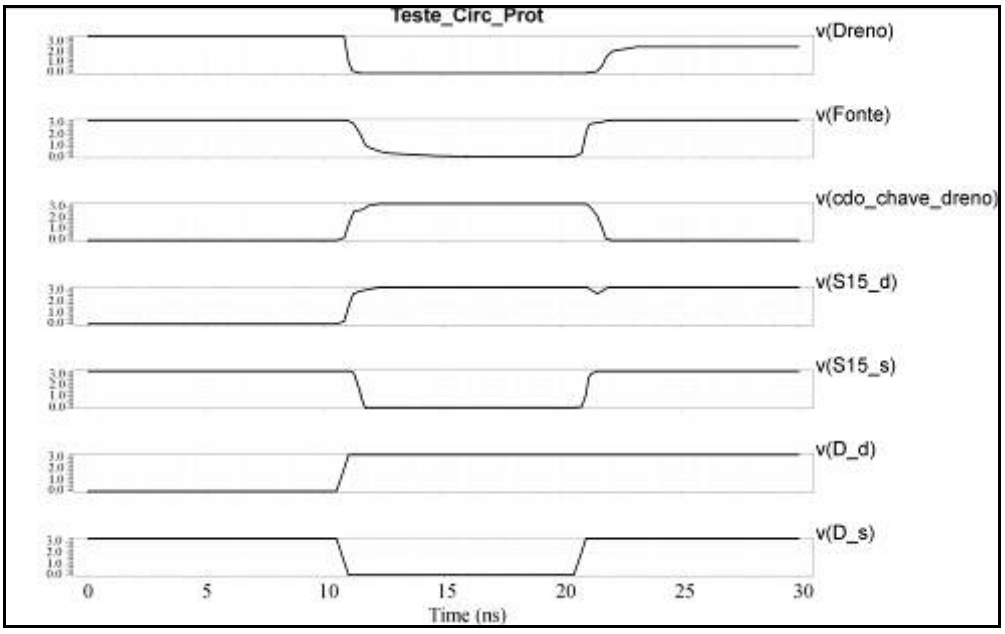


Figura 4-23 – Simulação do circuito de proteção

4.2.3. Descrição do bloco – Célula programável + Barramento local

Como podemos ver na Figura 4-24, os bits de programação das chaves e das células vão para uma via de dados (cromossomo) de doze bits. Os sinais hab_cel, hab_chav do decodificador são provenientes dos latch´s, como visto na Figura 4-20. Os quatro bits deste decodificador vão para uma via de controle.

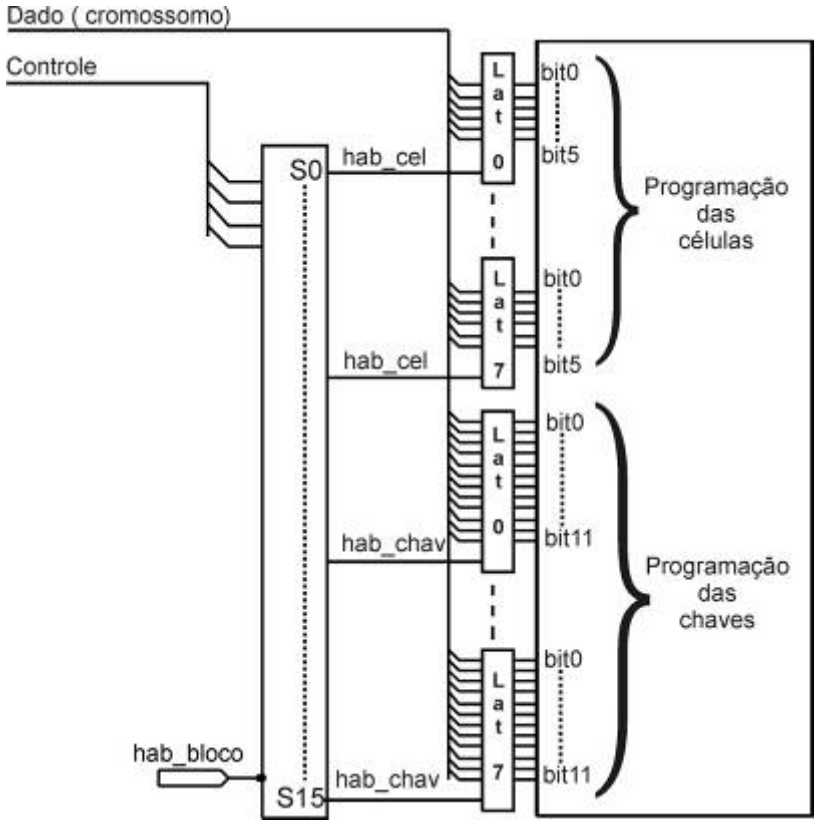


Figura 4-24 - Programação de um bloco contendo oito células programáveis

4.2.4. Barramento externo ao bloco

Seguindo o mesmo tipo de programação do bloco, com a diferença de que as chaves não estão ligadas a um decodificador, portanto, todas têm que ter o seu controle armazenado no latch

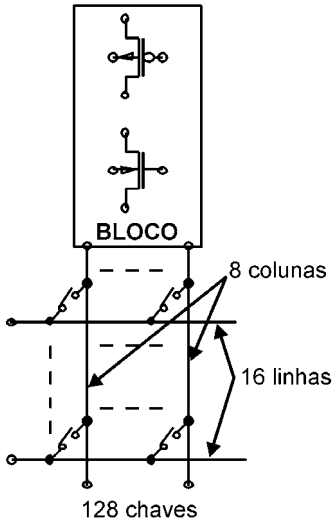


Figura 4-25– Chaves do barramento externo ao bloco

Cada grupo de 128 chaves do barramento externo (16x8 – 16 linhas do barramento externo com as 8 colunas disponibilizadas pelo bloco) estão ligadas a 16 latchs Figura 4-26). O dado de entrada destes latchs está conectado ao mesmo barramento de dados do bloco. A seleção dos latchs é gerada pelo decodificador 4x16 que tem sua entrada conectada ao barramento de controle, do mesmo bloco.

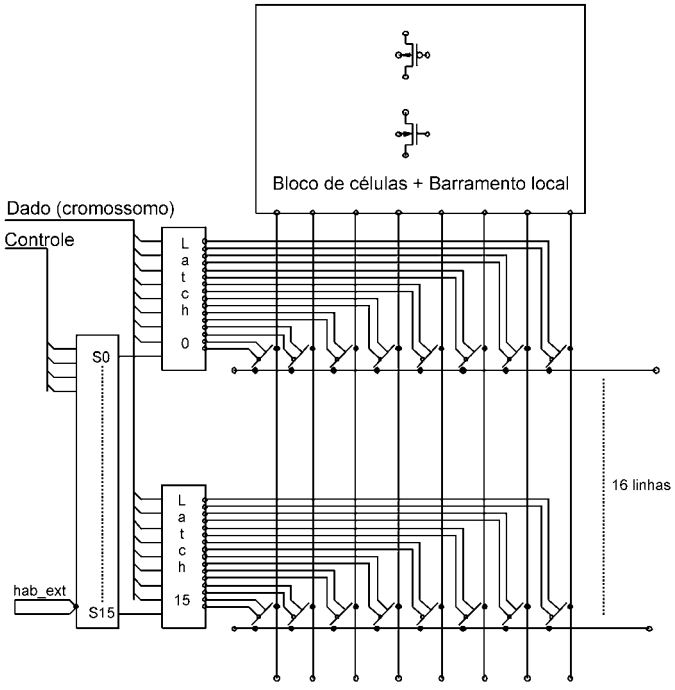


Figura 4-26– Barramento externo - programação

4.2.5. Proposta para o circuito integrado

A Figura 4-27 mostra um diagrama de blocos da proposta para o circuito integrado que é composto de 16 blocos com 8 células programáveis cada, dois barramentos externos de 16 vias que são disponibilizados nos pinos do circuito integrado com possibilidade de interligação entre os barramentos por chaves internas.

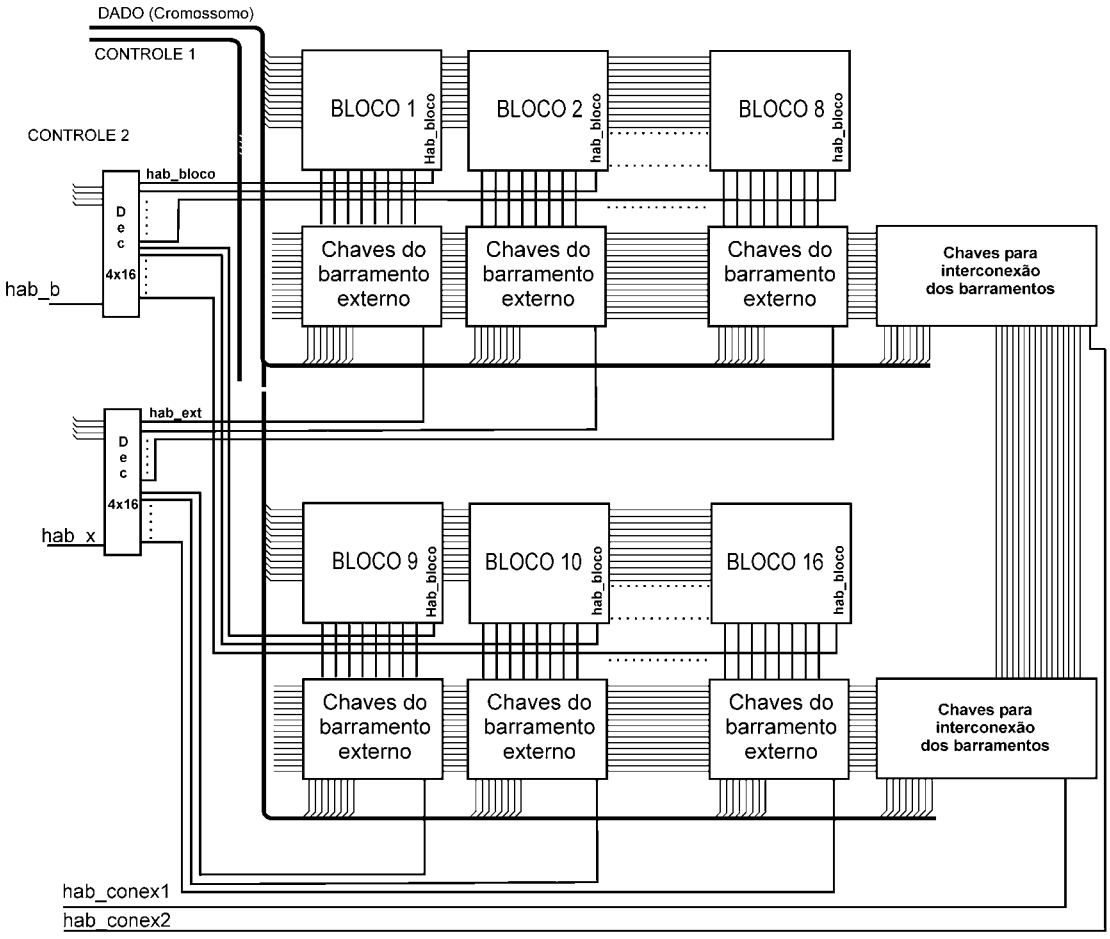


Figura 4-27– Visão geral dos blocos e chaves do barramento externo

Uma via de dados de 12 bits e duas vias de controle com 4 bits cada, dois sinais hab_b e hab_x e dois sinais para interconexão do barramento, hab_conex1 e hab_conex2 em conjunto com a via de dados e de controle serão utilizados para a programação do circuito integrado. A quantidade de pinos definida (Figura 3-1) para o circuito integrado é de 68 pinos. Este tipo de arquitetura permite que evoluções

independentes possam ser realizadas, pois ao disponibilizarmos dois barramentos de 16 vias, sinais de entrada e saída podem ficar separados nos barramentos externos.

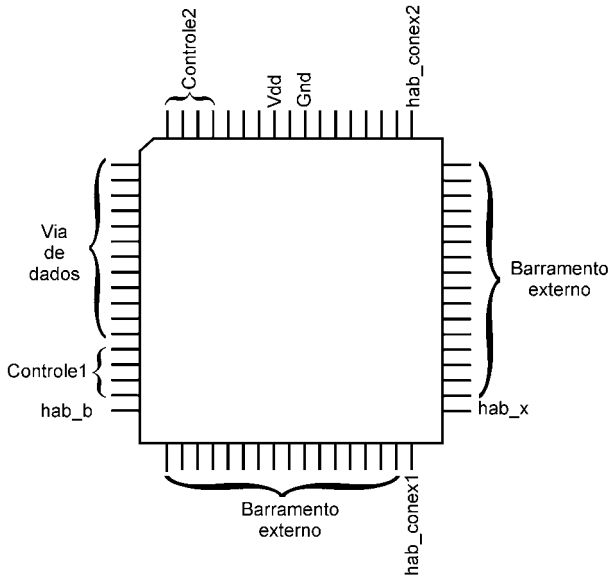


Figura 4-28 – Proposta de distribuição de sinais

Para a programação do bloco (Figura 4-30) coloca-se o valor do cromossomo que irá configurar o transistor programável, nas linhas de dado. Os sinais das linhas de controle 1 selecionam a célula de transistores que se deseja configurar e as linhas de controle 2 seleciona o bloco a ser programado. Após estabilização dos sinais aplica-se um pulso na entrada hab_b fazendo com que os dados sejam armazenados nos latchs respectivos. Terminada a programação dos transistores inicia-se a programação do barramento local utilizando o mesmo procedimento sendo que os sinais do controle 1 são direcionados aos latchs das chaves do barramento local.

O cromossomo representando um bloco, incluindo as chaves do barramento externo, é formado por 272 bits dispostos da maneira indicada na Figura 4-29. O primeiro grupo de 48 bits representa a programação dos oito transistores configuráveis, o bloco seguinte corresponde aos bits das chaves que conectam os terminais dos transistores ao barramento local. Os 128 bits seguintes são responsáveis pelas chaves do barramento externo como mostra a Figura 4-29 com o nome de “Linhas do barramento externo”.

48 bits Bloco1 Transistores programáveis								96 bits Bloco1 Chaves barramento local								128 bits Linhas do barramento externo 16x8															
6 bits	6 bits	6 bits	6 bits	6 bits	6 bits	6 bits	6 bits	12 bits	12 bits	12 bits	12 bits	12 bits	12 bits	12 bits	12 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16

Figura 4-29 – Formato do cromossomo para um bloco

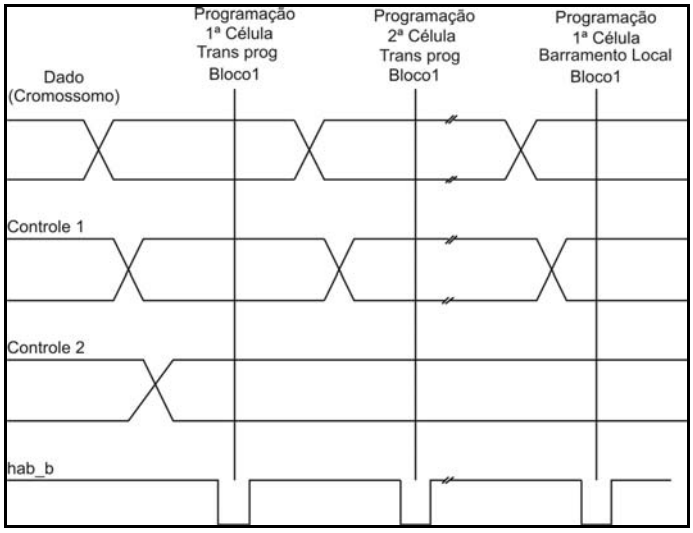


Figura 4-30 – Programação de um bloco do circuito integrado

Para programar o barramento externo coloca-se nas linhas de dado o valor do cromossomo respectivo e aciona-se através das linhas de controle 1 o latch da primeira linha do barramento (16 linhas-8 colunas). O controle 2 seleciona quais conjuntos de chaves (barramento externo) a ser programado. Após estabilização dos sinais aplica-se um pulso na entrada hab_x para armazenar o dado nos latchs respectivos.

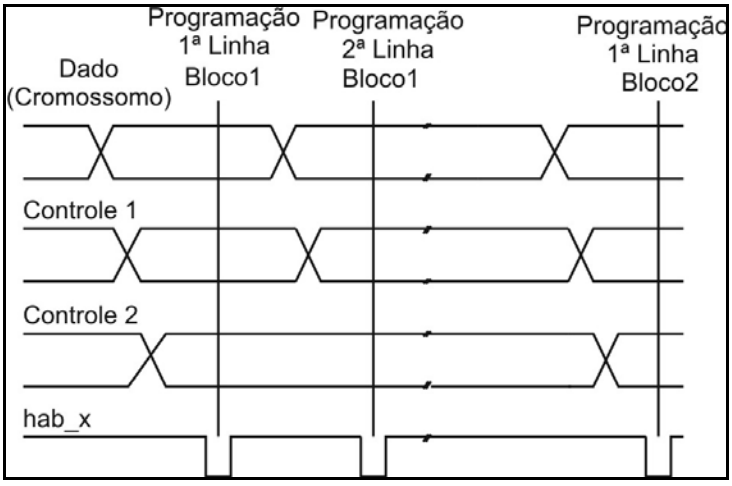


Figura 4-31 – Programação do barramento externo

A programação do barramento de interconexão, visto na Figura 4-32 como um bloco denominado “chaves para interconexão dos barramentos”, faz-se aplicando o valor relativo às chaves do cromossomo nas linhas de dado. O controle 1 seleciona qual linha do barramento de interconexão (16 linhas – 8 colunas) será programada e após estabilização dos sinais aplica-se um pulso na entrada hab_conex1 ou hab_conex2, para armazenar os dados nos latchs respectivos.

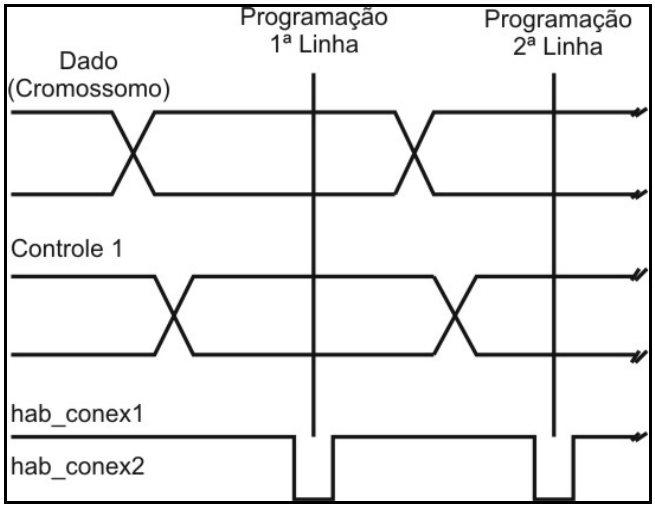


Figura 4-32 – Programação do barramento de interconexão

4.3. Testes Preliminares

Os testes a seguir foram realizados apenas com o intuito de verificar se as chaves exerceriam muita influência em alguns circuitos clássicos. Como foi citado anteriormente, o amplificador operacional e o amplificador de transcondutância na sua forma clássica foram simulados no SPICE e suas curvas de ganho e transferência DC foram obtidas. Na Figura 4-33 é mostrado o amplificador operacional e na Figura 4-36 é mostrado o amplificador de transcondutância transpostos para o tipo de barramento proposto com as chaves necessárias já acionadas e o tamanho dos transistores e valores das fontes idênticos aos do circuito original. O tamanho da chave utilizada para o transistor NMOS foi de: L=0,35µm;W=5µm e para o transistor PMOS: L=0,35µm;W=15µm.

Os circuitos transpostos foram simulados obtendo-se para o amplificador operacional uma variação de aproximadamente 13% na análise AC (Figura 4-34) e na análise DC (Figura 4-35) a excursão não apresentou diferença significativa. Para o amplificador de transcondutância (Figura 4-37) a variação da análise AC ficou em torno de 18% e a análise DC não apresentou variação significativa em relação ao original.

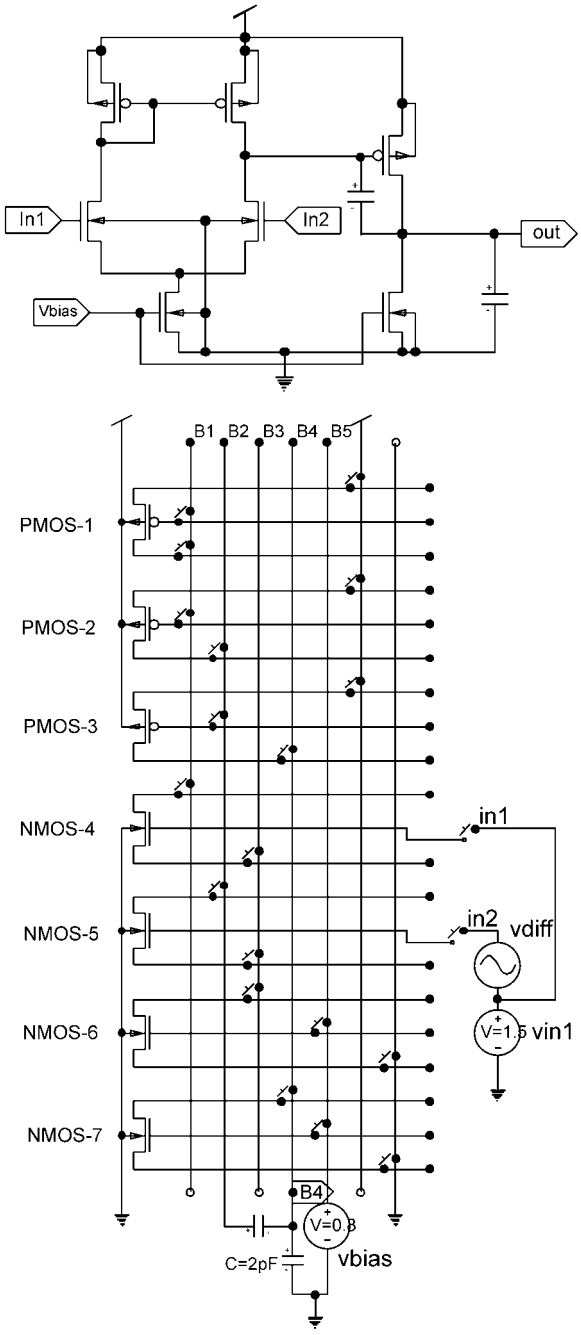


Figura 4-33 – Amplificador operacional transposto para o modelo de barramento

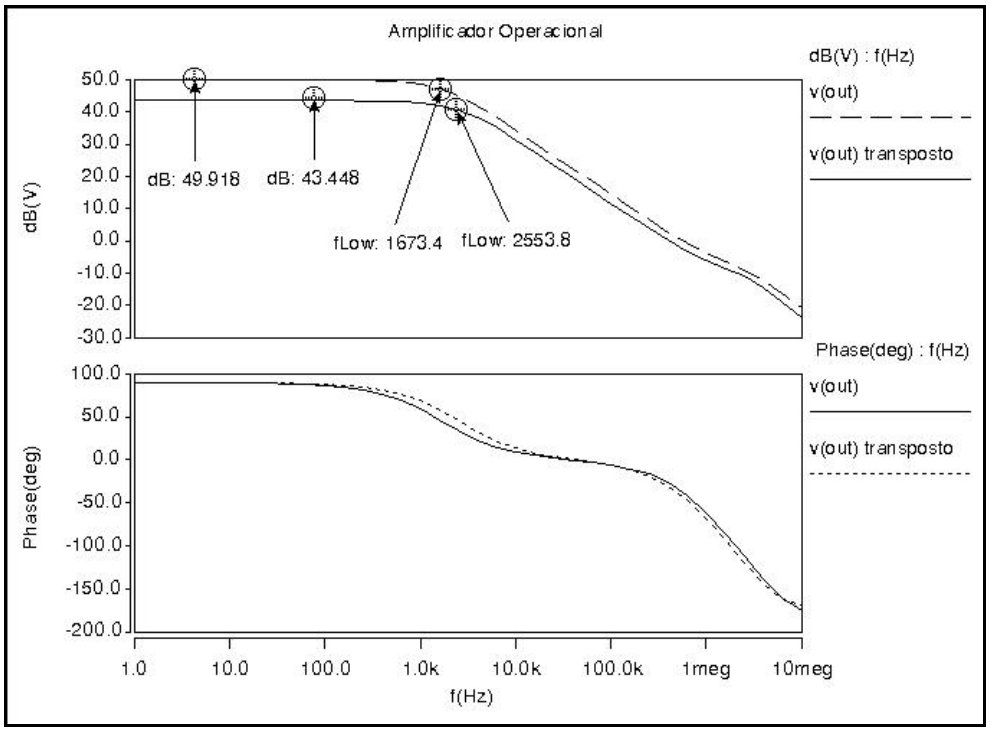


Figura 4-34 – Ganho e Fase do amplificador operacional normal e transposto

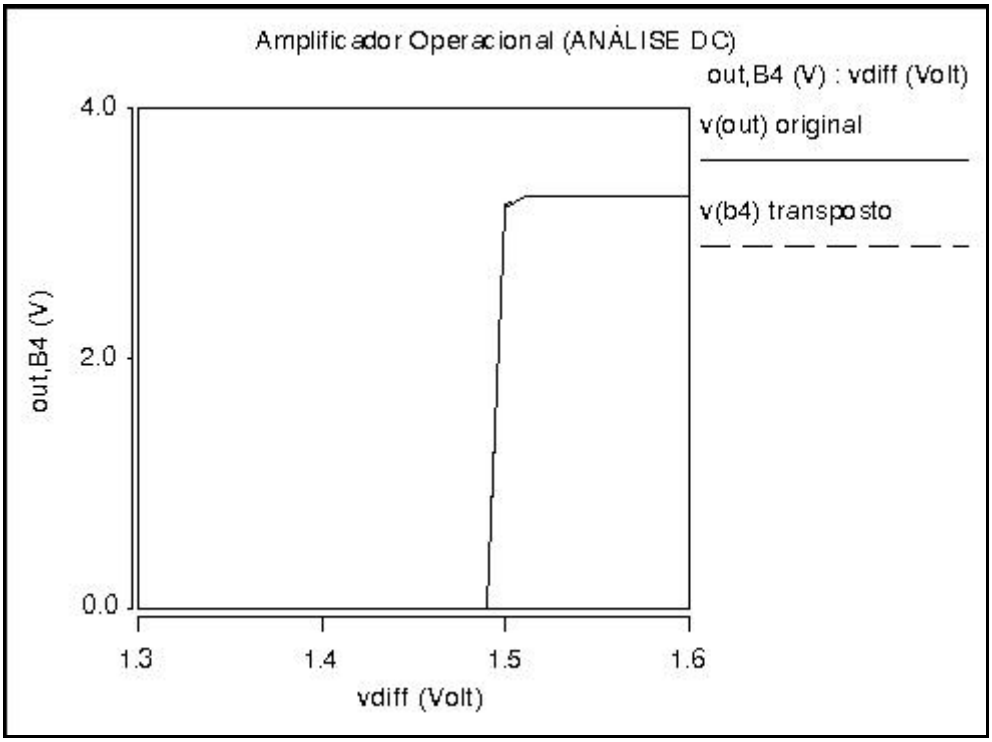


Figura 4-35 – Transferência DC do amplificador operacional normal e transposto

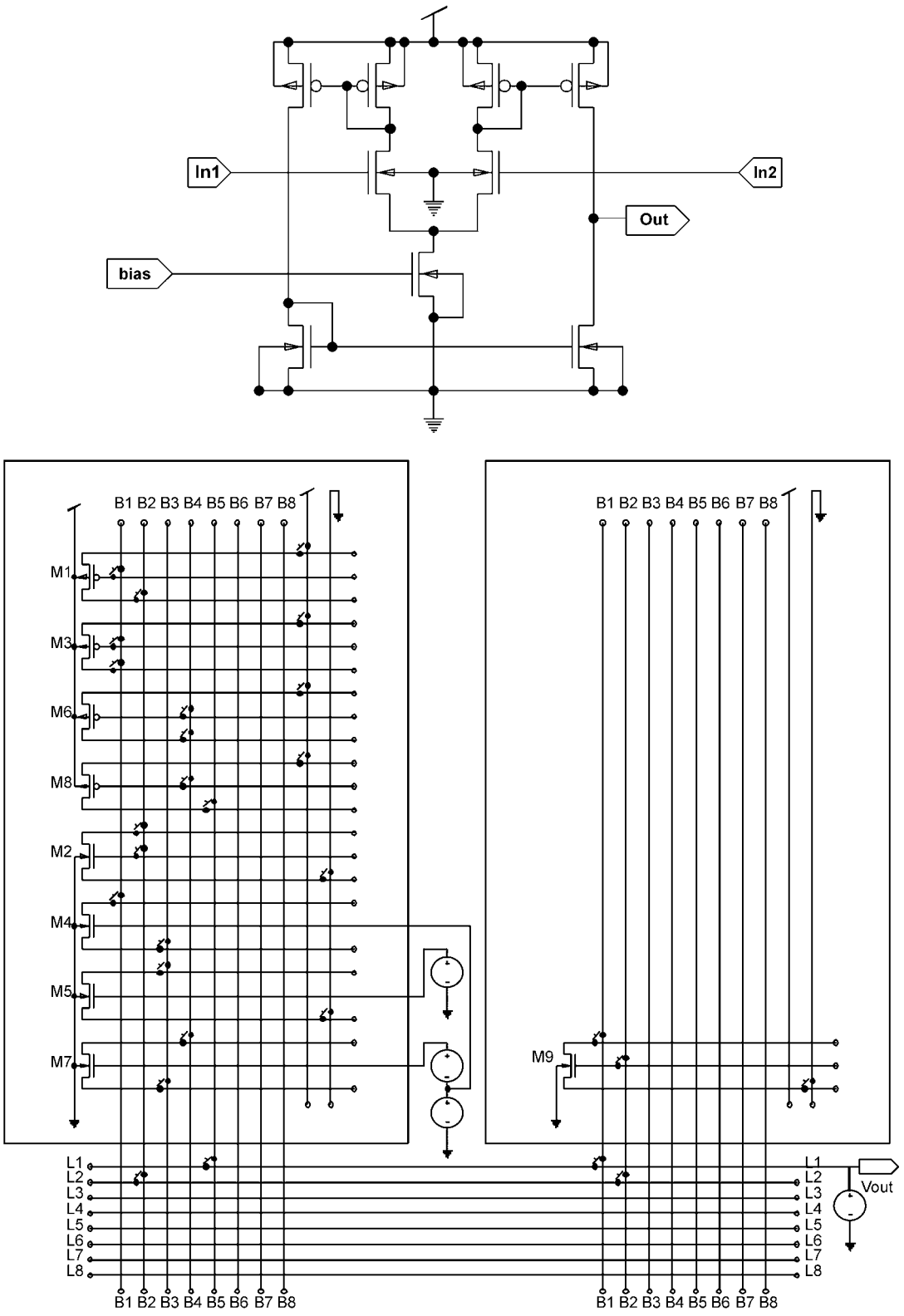


Figura 4-36 – Amplificador Operacional de Transcondutância transposto para o modelo de barramento

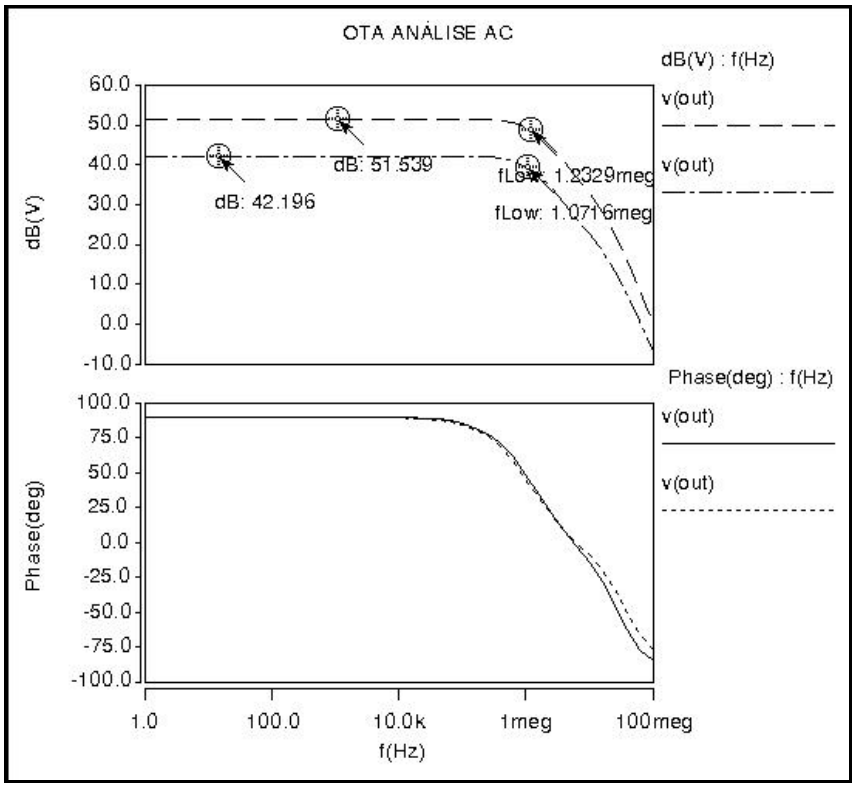


Figura 4-37– Ganho e fase do amplificador operacional de transcondutância normal e transposto

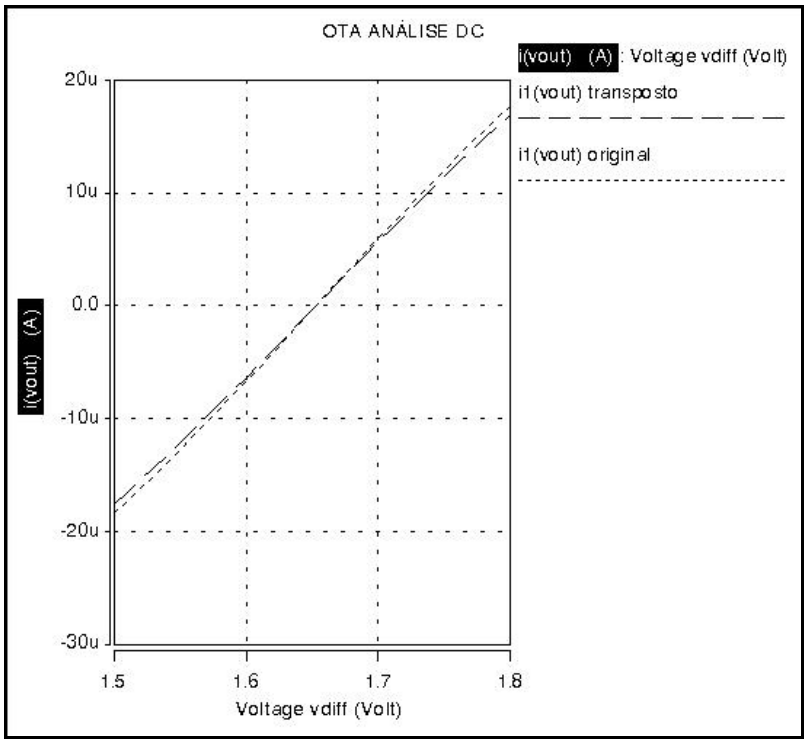


Figura 4-38 – Transferência DC do amplificador operacional de transcondutância normal e transposto

5. Ambiente de Simulação

5.1. Descrição da Ferramenta

O programa foi desenvolvido utilizando o sistema operacional Windows XP e o ambiente de desenvolvimento Microsoft Visual Basic 6. Uma vez definida a função objetivo o programa tem como finalidade executar o algoritmo genético do tipo “Steady state” para gerar indivíduos (cromossomo) que são substituídos nos arquivos do hardware sob teste no simulador SPICE. O simulador fornece as curvas de saída correspondente aos bits do cromossomo que foram devidamente substituídos e então é calculada a aptidão (fitness) de cada indivíduo. Sendo esse um processo demasiadamente lento, pois, várias iterações são realizadas até atingir uma aproximação desejada da função objetivo, optou-se para melhorar a performance da ferramenta, desenvolver um programa utilizando nove máquinas ligadas em rede Ethernet. O computador principal é responsável pela execução do algoritmo, distribuição dos arquivos para as máquinas que estiverem disponíveis em rede, compilar os resultados e apresentar os gráficos em uma interface homem máquina (IHM) amigável. Possibilita também efetuar troca de parâmetros do algoritmo, armazenamento de arquivos para que se possa interromper uma evolução e continuá-la mais adiante. Às máquinas auxiliares cabe a tarefa de receber os arquivos do computador principal executar a simulação no SPICE e retornar o resultado.

5.1.1. Programa Principal

O programa manipula os arquivos utilizados pelo simulador SPICE para converter o cromossomo fornecido pelo algoritmo em bits de programação das chaves de configuração do circuito e extrair a função de saída para calcular a aptidão dos indivíduos do torneio.

O arquivo de entrada utilizado para simulação do SPICE, como podemos ver um trecho na Figura 5-1, possui algumas frases que fornecem ao programa o ponto de

busca para localização do que será substituído, no caso, "*Main circuit" é uma delas. No exemplo da Figura 5-1, que representa o circuito que está sendo simulado, a palavra "bit" corresponde propositalmente às chaves do barramento. De acordo com o cromossomo fornecido pelo algoritmo genético, os caracteres "bit0" ao "bit119" são substituídos por V_{DD}, se o bit do cromossomo for "1", ou GND se o bit do cromossomo for "0".

```
* Main circuit: Barramento
.ac DEC 5 1 10MEG
Xcelula_1 bit0 bit1 bit2 bit3 bit4 bit5 bit6 bit7 bit8 bit9 bit10 bit11
+ bit12 bit13 bit14 bit15 bit16 bit17 bit18 bit19 bit20 bit21 bit22 bit23 N56 N55
+ N54 N29 N28 N42 Gnd Vdd celula
Xcelula 2 bit24 bit25 bit26 bit27 bit28 bit29 bit30 bit31 bit32 bit33 bit34
+ bit35 bit36 bit37 bit38 bit39 bit40 bit41 bit42 bit43 bit44 bit45 bit46 bit47
+ N20 N19 N34 N81 N80 N76 Gnd Vdd celula
.dc vin1 0 3.3 0.1
XDecoder_3x8_1 N73 N68 N67 N72 N69 Vdd Gnd out bit48 bit49 bit50 bit51 bit52
+ bit53 bit54 bit55 bit56 N56 N55 N54 Gnd Vdd Decoder_3x8
XDecoder_3x8_2 N65 N66 N84 N32 N83 N26 N82 N22 bit57 bit58 bit59 bit60 bit61
+ bit62 bit63 bit64 bit65 N56 N55 N54 Gnd Vdd Decoder_3x8
XDecoder_3x8_3 N73 N68 N67 N72 N69 Vdd Gnd out bit66 bit67 bit68 bit69 bit70
+ bit71 bit72 bit73 bit74 N29 N28 N42 Gnd Vdd Decoder_3x8
XDecoder_3x8_4 N65 N66 N84 N32 N83 N26 N82 N22 bit75 bit76 bit77 bit78 bit79
+ bit80 bit81 bit82 bit83 N29 N28 N42 Gnd Vdd Decoder_3x8
XDecoder_3x8_5 N73 N68 N67 N72 N69 Vdd Gnd out bit84 bit85 bit86 bit87 bit88
+ bit89 bit90 bit91 bit92 N20 N19 N34 Gnd Vdd Decoder_3x8
XDecoder_3x8_6 N65 N66 N84 N32 N83 N26 N82 N22 bit93 bit94 bit95 bit96 bit97
+ bit98 bit99 bit100 bit101 N20 N19 N34 Gnd Vdd Decoder_3x8
XDecoder_3x8_7 N73 N68 N67 N72 N69 Vdd Gnd out bit102 bit103 bit104 bit105
+ bit106 bit107 bit108 bit109 bit110 N81 N80 N76 Gnd Vdd Decoder_3x8
XDecoder_3x8_8 N65 N66 N84 N32 N83 N26 N82 N22 bit111 bit112 bit113 bit114
+ bit115 bit116 bit117 bit118 bit119 N81 N80 N76 Gnd Vdd Decoder_3x8
.include modn.mod
.include modp.mod
.options post
.print dc v(out) .print ac v(out)
```

Figura 5-1 – Arquivo de entrada do SPICE

Da mesma maneira é tratado o arquivo texto de saída do SPICE, onde são extraídos os resultados das análises.

A função objetivo segue o mesmo padrão do arquivo texto de saída do SPICE, cujos valores serão utilizados para o cálculo da aptidão do indivíduo.

Ao iniciar o programa, uma tela de apresentação com uma barra de menu na parte superior da janela (Figura 5-2), permite navegar pelo programa. Duas outras telas compõem a IHM, uma tela de configuração de rede e uma outra tela subdividida em pastas para configuração do algoritmo e acompanhamento da sua evolução e também, para apresentação dos gráficos resultantes das análises.

5.1.1.1. Tela de apresentação

A Figura 5-2 mostra o detalhe do menu, onde o comando “Arquivo” permite realizar as operações de:

- novo – Criar um arquivo para poder armazenar os dados referentes à simulação que será iniciada.
- abrir – Abrir um arquivo já existente para análise ou continuação da simulação.
- salvar – Guardar as informações pertinentes até então geradas.
- apagar – Excluir um arquivo existente.

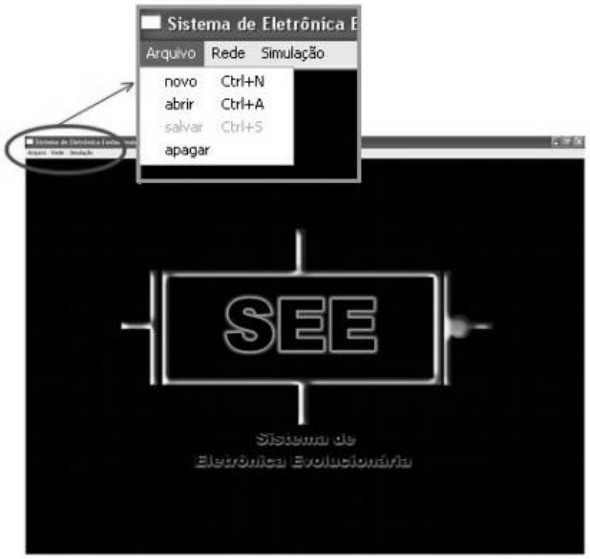


Figura 5-2 – Tela de apresentação

5.1.1.2. Tela de configuração e acompanhamento das conexões de rede

O segundo comando do menu (Rede), abre uma tela de configuração e acompanhamento do estado das conexões de rede. Como podemos ver na Figura 5-3, a tecla “Carregar Arquivo de Configuração” abre uma janela onde um arquivo texto pré-estabelecido é carregado com os endereços (IP – Internet Protocol) das máquinas que irão participar da simulação. Os IPs também podem ser inseridos, diretamente no campo destinado à sua visualização.

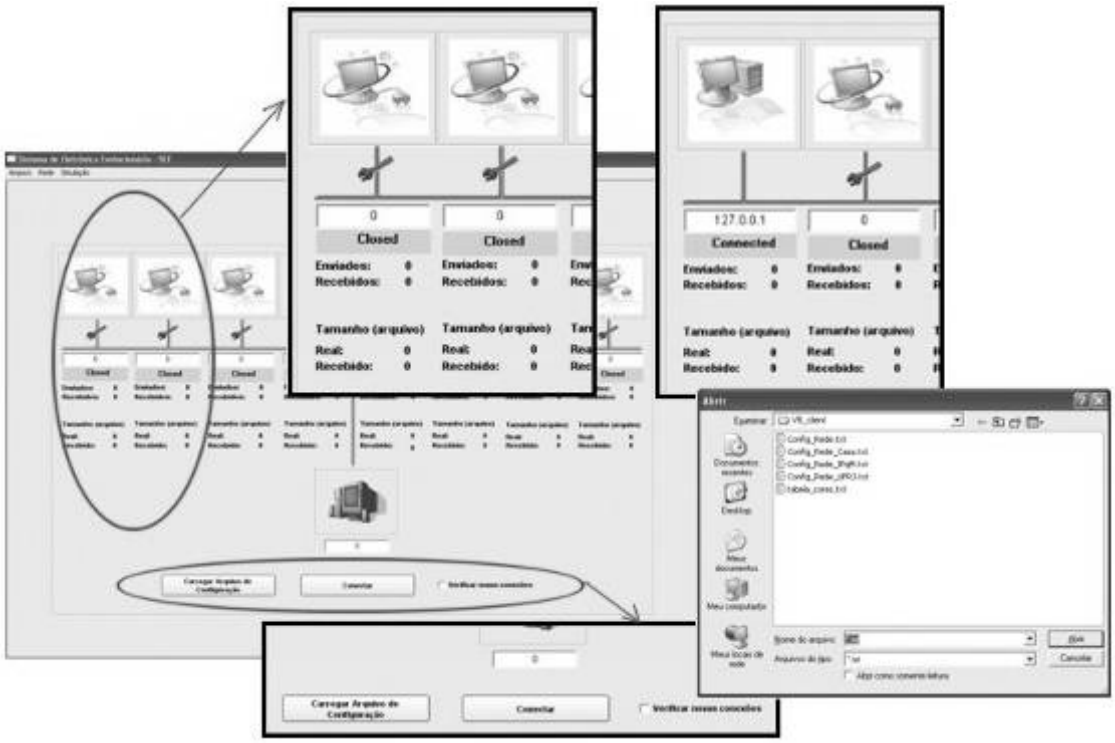


Figura 5-3 – Tela de configuração das conexões de rede

O formato do arquivo texto de configuração segue a maneira usual da notação utilizado para escrever o IP, acrescentado de aspas como mostra o exemplo abaixo.

"146.164.96.51"
"146.164.96.58"
"146.164.96.53"
"146.164.96.54"
"146.164.96.55"
"146.164.96.56"
"146.164.96.57"
"146.164.96.62"

As figuras mostram o estado da conexão, como pode ser visto no detalhe superior direito. Abaixo de cada figura os endereços (IPs) das máquinas e o acompanhamento dos arquivos enviados e recebidos. O comando para checar conexões, “Verificar novas conexões”, uma vez acionado, faz uma varredura de todos os endereços para atualizar o estado das conexões de rede.

5.1.1.3. Tela de configuração e acompanhamento do algoritmo genético

O terceiro comando “Simulação” (Figura 5-4) abre uma tela contendo seis pastas: Análise DC, Análise AC, Análise Transiente, NetList, Parâmetros e

Observações. A lateral direita da tela é fixa para todas as pastas, são informações do tipo:

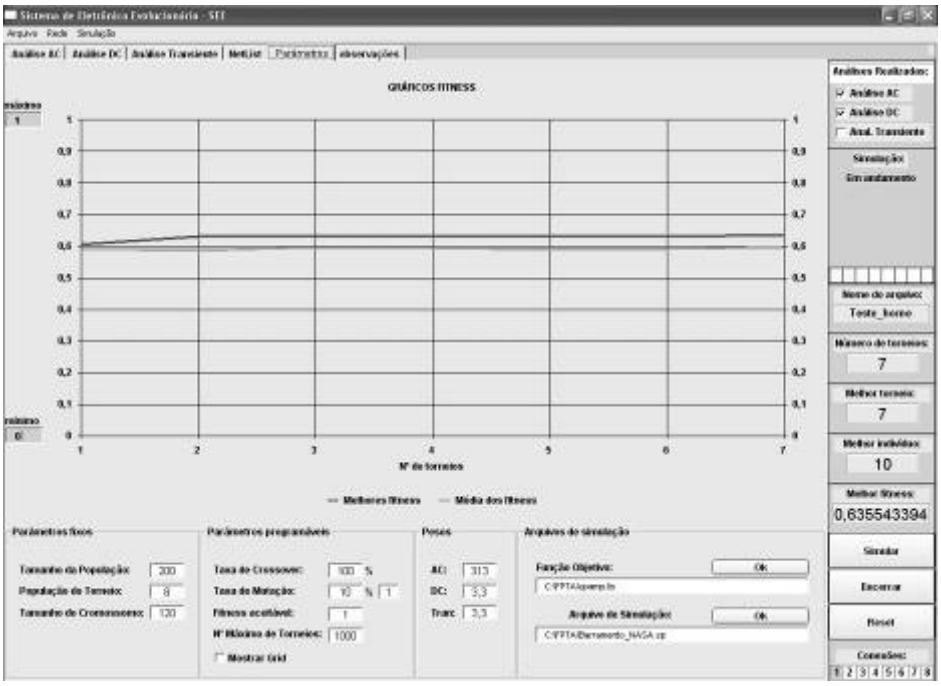


Figura 5-4 – Tela de Simulação - Parâmetros

- Análises Realizadas – Indica quais análises estão sendo verificadas pelo simulador SPICE (Análise AC, Análise DC e Análise Transiente).
- Simulação – Indica se o programa está em andamento e a barra inferior, onde cada divisão é um indivíduo participante do torneio. A cor clara significa que a aptidão está sendo calculada. A cor mais escura indica que sua aptidão já foi calculada em sorteio anterior.
- Nome do Arquivo – Nome do arquivo escolhido no comando “Arquivo” na barra de menu.
- Número de torneios – Quantidade de torneios realizados.
- Melhor torneio – Torneio onde se obteve a melhor aptidão.
- Melhor indivíduo – O número do indivíduo que possui melhor aptidão da população sorteada.
- Melhor fitness – Maior aptidão verificada até o momento.

- Comando Simular – Inicia execução
- Comando Encerrar – Encerra execução e armazena os dados.
- Comando Reset – Limpa os dados carregados de configuração.
- Conexões – As conexões que estiverem ativas aparecem na cor verde.

A seguir será descrito o conteúdo das pastas da tela “Simulação”

❑ Pasta Análise AC

O campo à esquerda desta pasta (Valores – função objetivo) mostra a listagem dos valores da função objetivo. No campo “Gráficos” o gráfico da função objetivo é apresentado na parte superior e o gráfico da função resultante da evolução é apresentado abaixo.

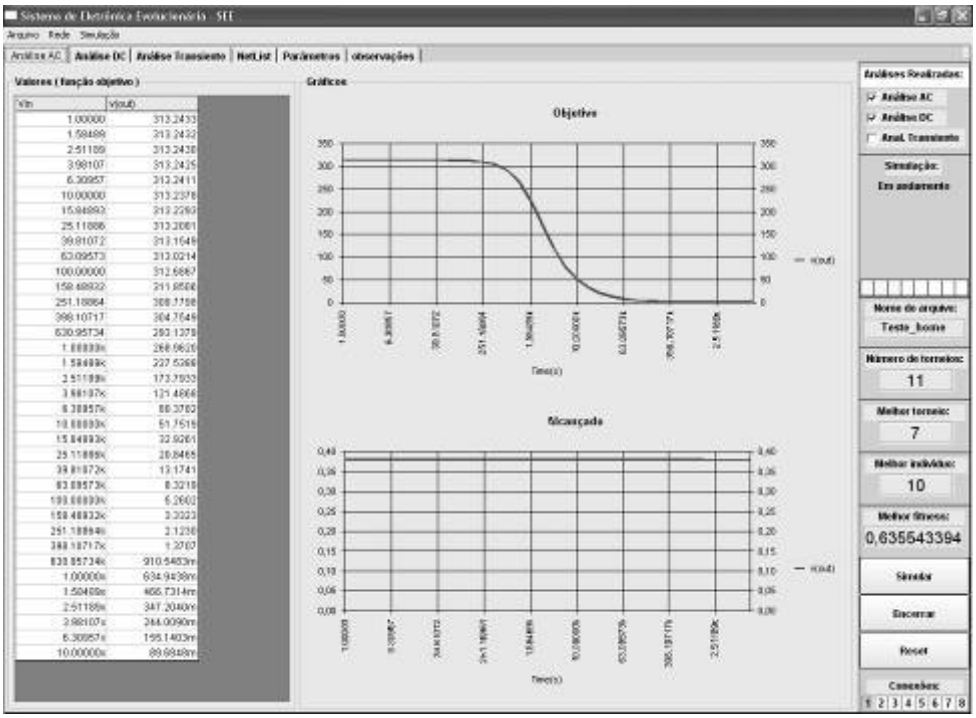


Figura 5-5 – Tela de simulação – Análise AC

❑ Pasta Análise DC - Segue a mesma orientação da Análise AC

❑ Pasta Análise Transiente - Segue a mesma orientação da outras análises

❑ Pasta NetList – mostra o arquivo utilizado para entrada do simulador, que é a descrição do circuito que está sendo simulado.

- Pasta Parâmetros – Esta pasta apresenta na parte central (Figura 5-4) um gráfico das melhores aptidões na cor verde e na cor vermelha a média das aptidões. O gráfico apresenta o comportamento do algoritmo a cada torneio realizado. Para melhor visualização do gráfico, os valores mínimo e máximo podem ser alterados. Na parte inferior da pasta estão presentes os parâmetros que são utilizados no algoritmo e alguns comandos para alteração dos parâmetros e carga dos arquivos de simulação.
  - Parâmetros fixos
    - Tamanho da população – número de indivíduos que irão participar da simulação.
    - População de torneio – número de indivíduos que serão sorteados a cada torneio.
    - Tamanho do cromossomo – número de bits de acordo com o hardware que está sob análise.
  - Parâmetros programáveis
    - Taxa de crossover – permite programar o algoritmo para realizar ou não cruzamento a cada torneio.
    - Taxa de mutação – permite programar o algoritmo para efetuar ou não mutação a cada torneio.
    - Fitness aceitável – valor onde atingiu a função objetivo.
    - Nº máximo de torneios – ao atingir o número programado de torneios o programa termina.
    - Mostrar grid – uma vez acionado, aparece na tela uma tabela com os indivíduos participantes do torneio, a aptidão e os bits do cromossomo.
  - Pesos
    - AC – Será utilizado para cálculo da aptidão da análise AC.
    - DC – Será utilizado para cálculo da aptidão da análise DC.

- Tran - Será utilizado para cálculo da aptidão da análise de transiente

➢ Arquivos de simulação

- Comando da função objetivo – Utilizado para carregar a função objetivo
- Comando do arquivo de simulação – utilizado para carregar o arquivo de entrada do SPICE onde se encontram os bits que serão substituídos pelo cromossomo gerado pelo algoritmo.

❑ Pasta Observações – utilizado para inserir alguns comentários que possibilite, por exemplo, o registro de alguma mudança introduzida nos parâmetros do algoritmo, durante a simulação.

5.1.2. Programa Auxiliar

Ao iniciar o programa uma tela de apresentação (Figura 5-6) mostra o endereço (IP), a porta de rede que está sendo utilizada e aguarda um pedido de conexão do computador principal.

Figura 5-6 – Tela de apresentação

Sendo a conexão estabelecida, a tela de apresentação é substituída por uma tela de acompanhamento do tráfego de arquivos (Figura 5-7) e a indicação do estado da simulação. A conexão sendo encerrada por ação normal no computador principal ou por falha, a tela de apresentação substitui a atual e fica novamente aguardando pedido de conexão.

O programa recebe o arquivo a ser simulado, inicia o simulador SPICE, aguarda o término da simulação reduz o arquivo texto de saída enviando-o para o computador principal.

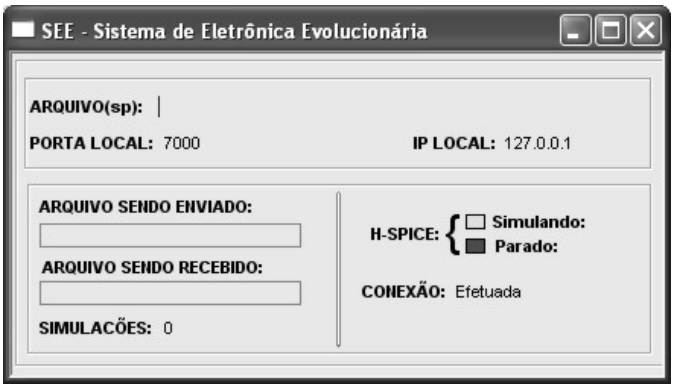


Figura 5-7 – Tela de acompanhamento da simulação

5.2. Algoritmo Genético Implementado

O algoritmo genético implementado é do tipo "substituição do estado estacionário" (steady state) e foi desenvolvido para poder utilizar até oito computadores auxiliares ligados em rede. O cálculo da aptidão é efetuado com a seguinte fórmula:

$$f = \frac{1}{1+\varepsilon}$$

onde o erro ε é dado por:

$$\varepsilon = \mathrm{W}_{\mathrm{AC}}\varepsilon_{\mathrm{AC}} + \mathrm{W}_{\mathrm{DC}}\varepsilon_{\mathrm{DC}} + \mathrm{W}_{\mathrm{Tran}}\varepsilon_{\mathrm{Tran}}$$

onde ε é o erro total, $\varepsilon_{\mathrm{AC}}$ é o erro proveniente da análise AC, $\varepsilon_{\mathrm{DC}}$ é o erro da análise DC, W é peso atribuído às análises. O termo do erro correspondente à análise AC é:

$$\varepsilon_{\mathrm{AC}_k} = \sum_{i=1}^{N_f} \left| G_k(\omega_i)_{obtido} - G\ (\omega_i)_{desejado} \right|$$

$\varepsilon_{\mathrm{AC}_k}$ é o erro correspondente ao k^n indivíduo da população, N_f é o número de freqüências de amostragem ω_i, $G_k(\omega_i)_{\mathrm{obtido}}$ and $G(\omega_i)_{\mathrm{desejado}}$ são respectivamente, o valor

da função de transferência obtido por k^n indivíduo na freqüência de amostragem i^n e o valor desejado correspondente.

Os passos de execução do algoritmo são descritos a seguir (Figura 5-8), com oito computadores ligados em rede.

- Geração da população cujo tamanho é especificado na tela Simulação na pasta Parâmetros.
- Sorteio de indivíduos que participarão do torneio e enviados aos computadores auxiliares para simulação.
- O programa controla a partir desse momento um registro de controle dos indivíduos que estão no processo de simulação.

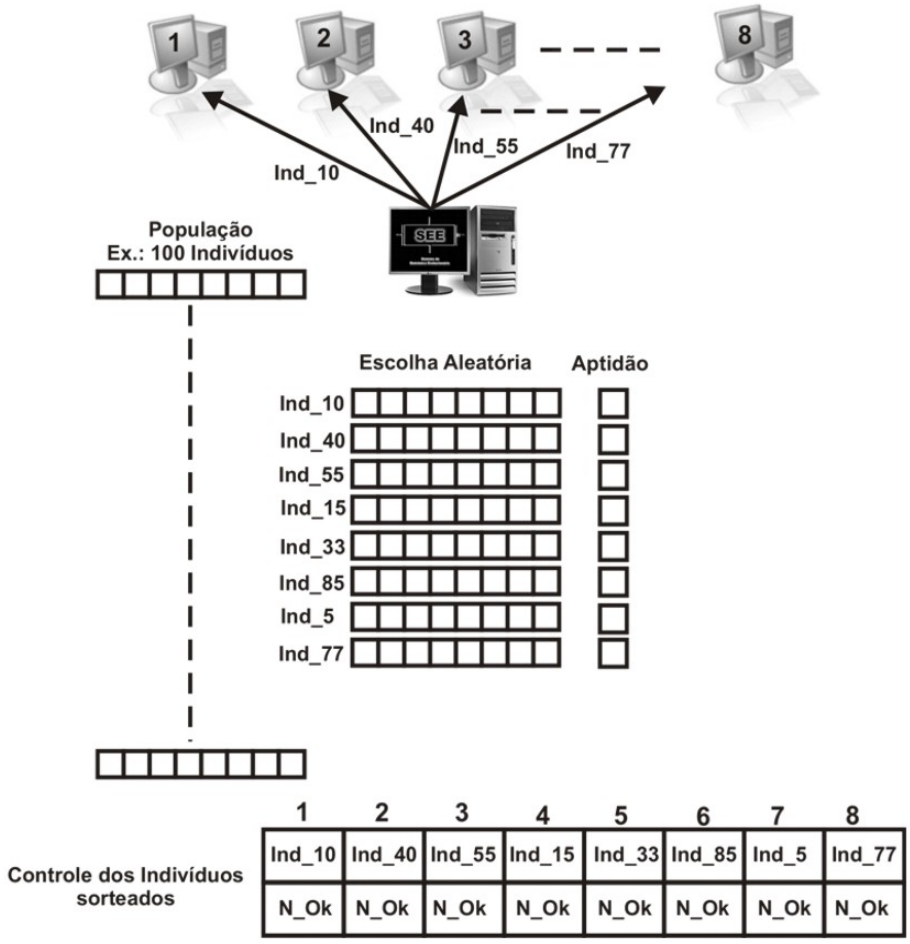


Figura 5-8 – Início de processamento do algoritmo genético

A medida que quatro arquivos são recebidos pelo computador principal (Figura 5-9), os passos do algoritmo passam a ser :

- Calcula as aptidões dos indivíduos.
- Cruzamento dos mais aptos, mutação de acordo com o que foi especificado na tela “Simulação pasta parâmetros” e substituição dos indivíduos ruins pelos neo-indivíduos.
- Sorteio de mais quatro indivíduos
- Atualização do registro de controle dos indivíduos
- Envio dos arquivos para as máquinas auxiliares
- Atualização das telas da IHM
- Caso tenha recebido mais quatro arquivos volta ao cálculo da aptidão seguindo o ciclo do algoritmo.

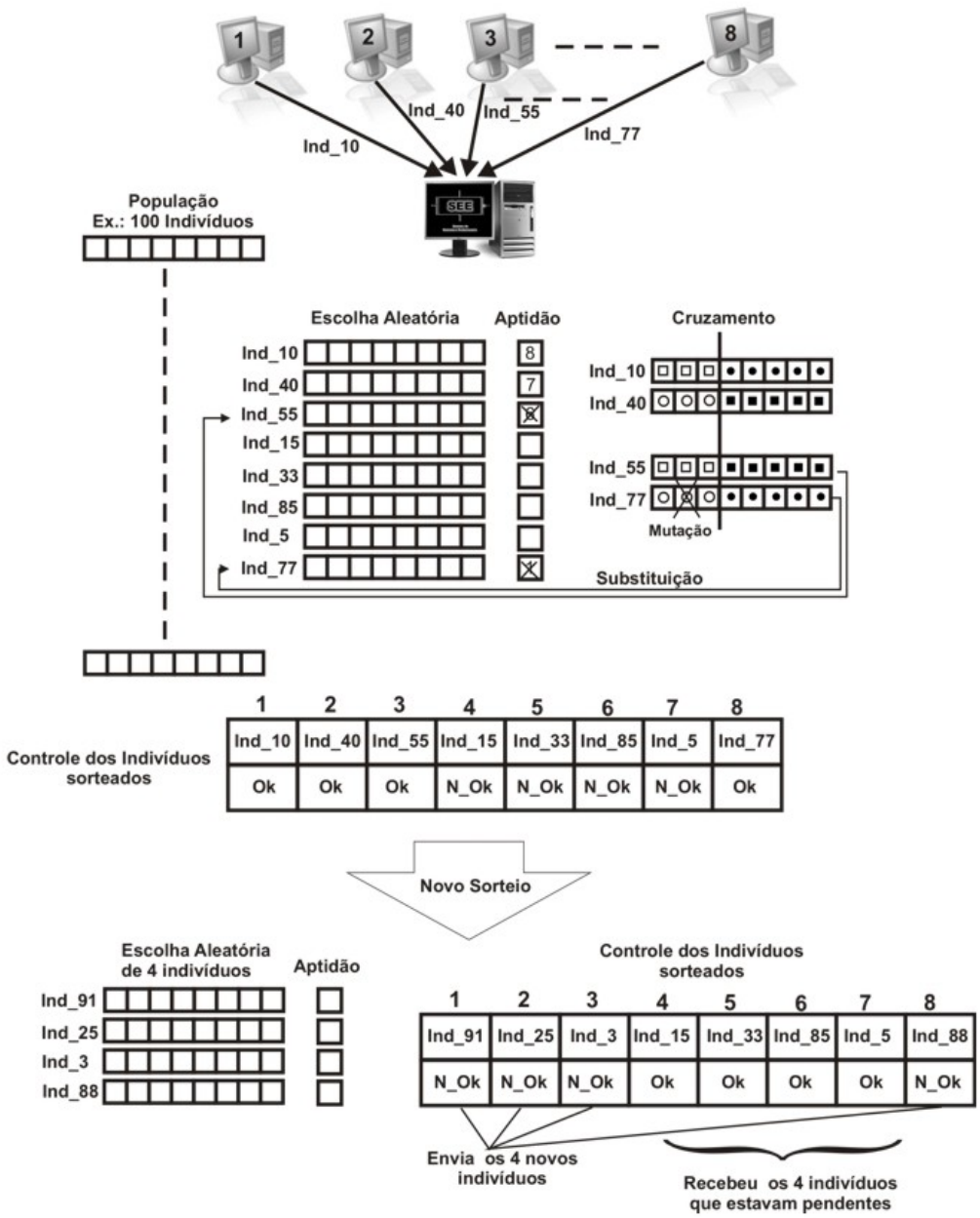


Figura 5-9 – Passos do Algoritmo Genético

6. Resultados

De acordo com os testes apresentados no capítulo 3.3, a variação de 13% na análise AC do amplificador operacional e a variação em torno de 18% na análise AC do amplificador de transcondutância, estão dentro do percentual aceitável, uma vez que a tolerância do processo de fabricação é de 20%.

Os testes realizados utilizando a ferramenta desenvolvida tiveram como finalidade avaliar a flexibilidade oferecida pelo barramento local em conjunto com diferentes estruturas de transistores programáveis. As funções objetivo de um inversor digital e de um amplificador operacional foram as entradas para que o processo evolucionário, utilizando o hardware sob teste, pudesse traçar o comportamento das estruturas indicando a viabilidade ou não do que fora proposto.

Dois tipos de análises foram simultaneamente usadas para cálculo de aptidão de cada indivíduo. No caso do circuito inversor, a análise DC e análise de transientes. No caso do amplificador operacional, a análise AC e análise de transferência DC. Isso se fez necessário, tomando como exemplo o amplificador operacional, para evitar que o processo evolucionário alcançasse um circuito com o ganho especificado, porém com uma excursão de sinal muito pequena, o que inviabilizaria a topologia alcançada.

6.1. Circuito Inversor

As figuras abaixo mostram a evolução do teste para dois tipos de estrutura de transistores programáveis e um tipo de barramento local. Na parte superior das análises DC e de transiente é apresentada a função objetivo e na parte inferior dos gráficos aparecem as funções alcançadas pelo algoritmo genético. Um outro gráfico mostra as melhores aptidões e a média das aptidões.

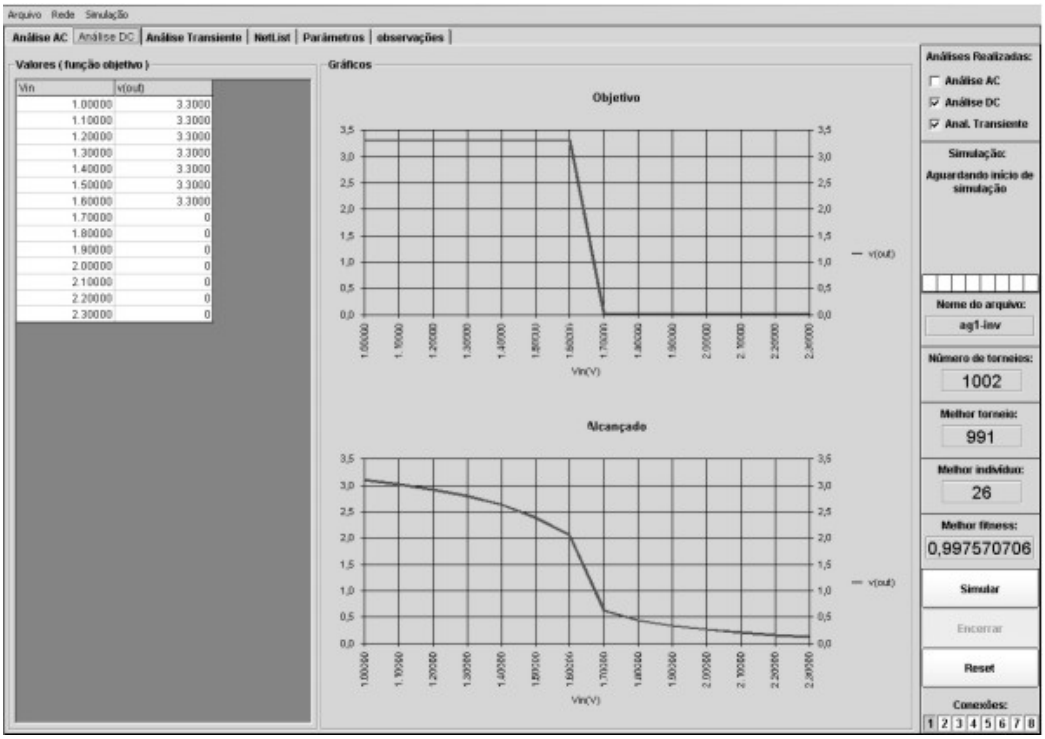


Figura 6-1 – Análise DC do circuito inversor estrutura 1 dos transistores programáveis

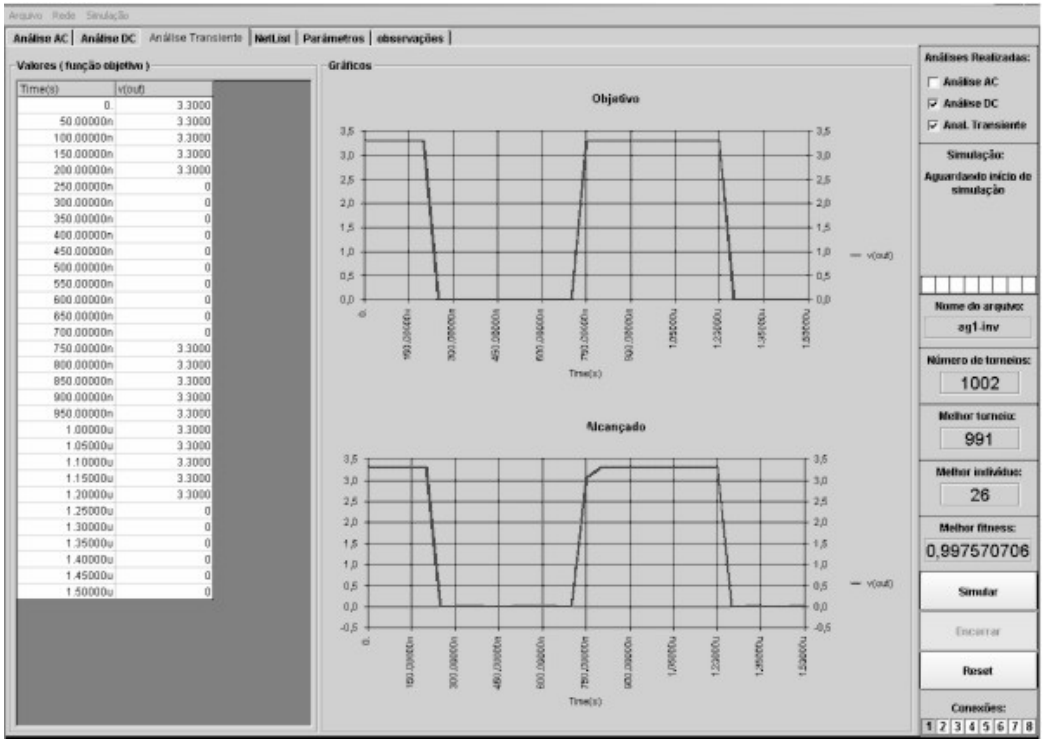


Figura 6-2 – Análise de transiente do circuito inversor estrutura 1 dos transistores programáveis

A Figura 6-3 mostra o comportamento do algoritmo através das melhores aptidões e da média das aptidões referentes ao circuito da Figura 6-1.

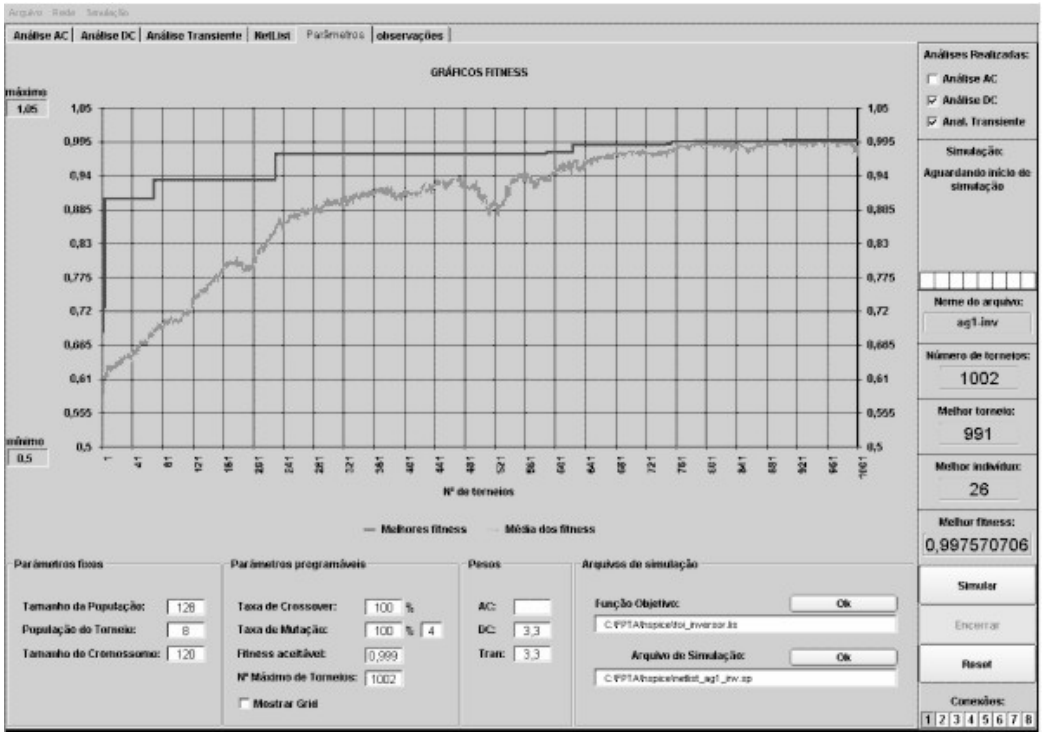


Figura 6-3 Comportamento do algoritmo genético para o circuito da Figura 6-1

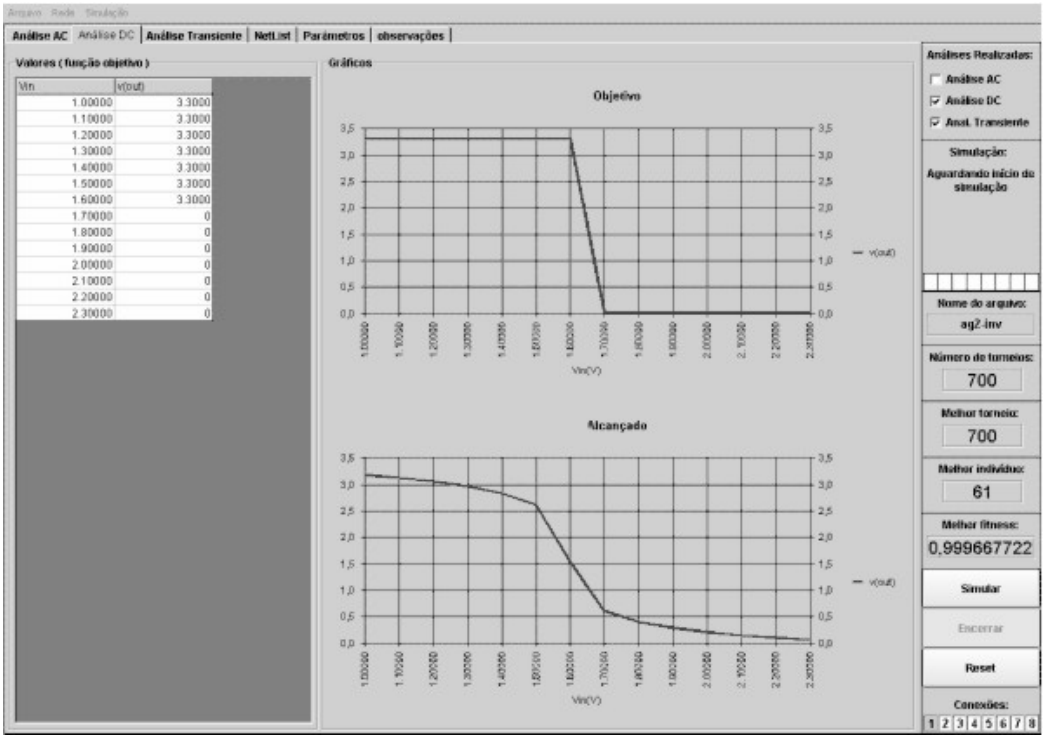


Figura 6-4 – Análise DC circuito inversor estrutura 2 dos transistores programáveis

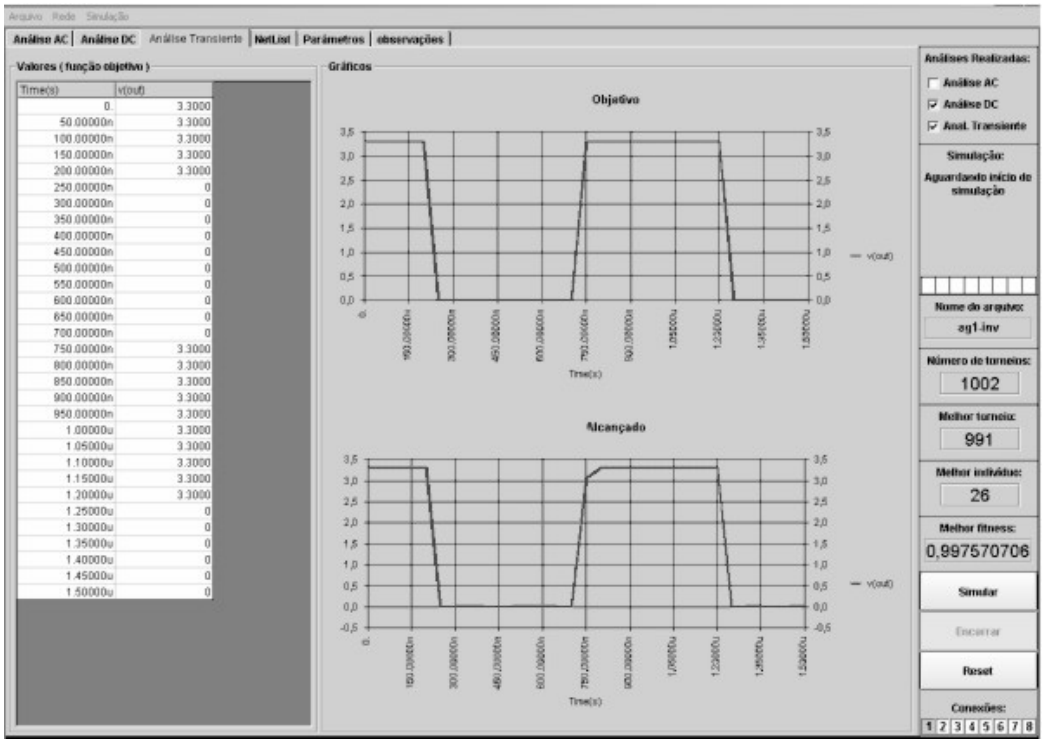


Figura 6-5 - Análise de transiente do circuito inversor da estrutura 2 dos transistores programáveis

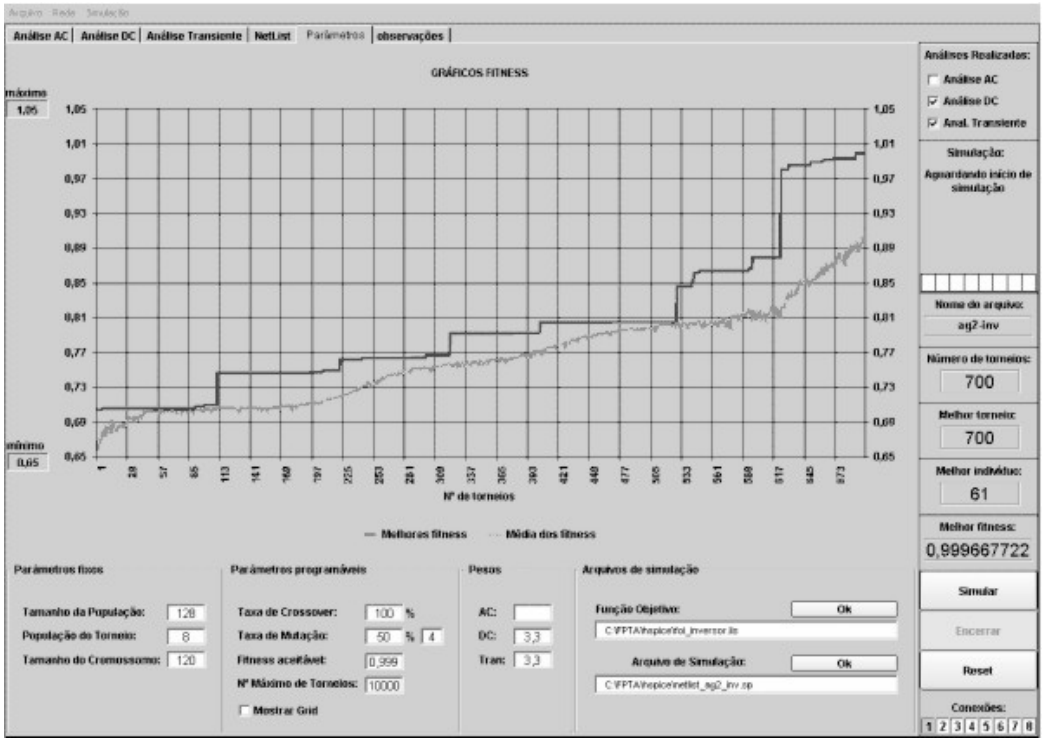


Figura 6-6 – Comportamento do algoritmo genético para o circuito inversor da estrutura 2 dos transistores programáveis

6.2. Circuito amplificador operacional

As figuras abaixo apresentam as análises para o teste cuja função objetivo representa a função de um amplificador operacional. Como visto anteriormente, os gráficos situados na parte superior mostram as funções objetivo e na parte inferior as funções alcançadas.

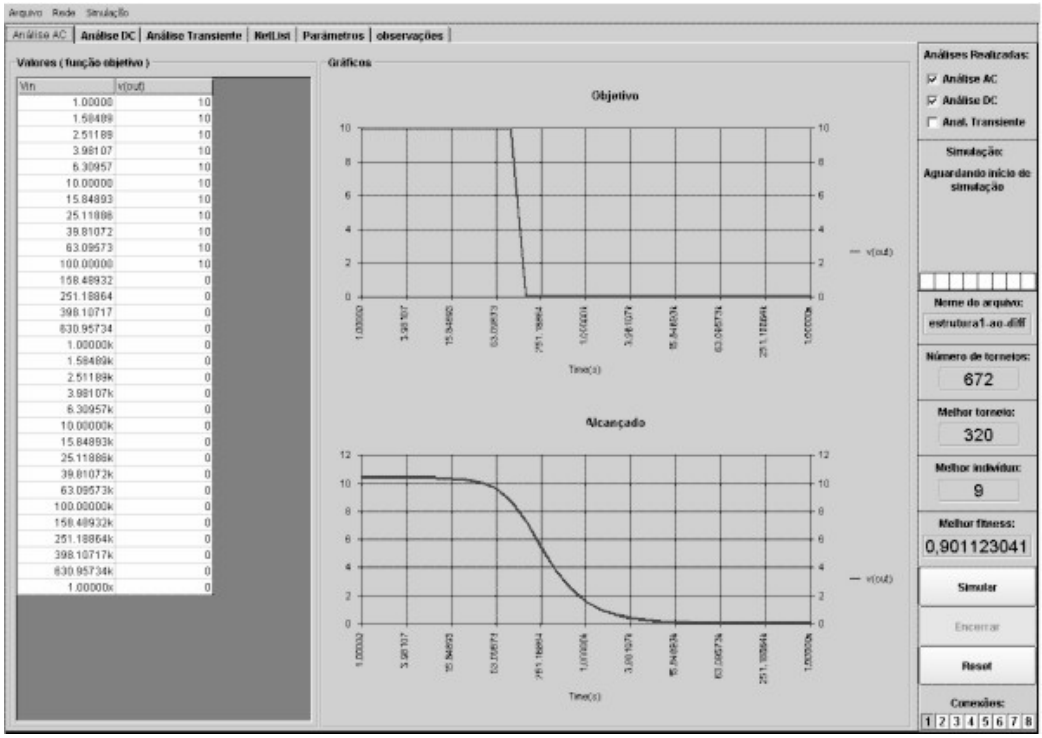


Figura 6-7 – Análise AC do amplificador operacional para estrutura 1 dos transistores programáveis

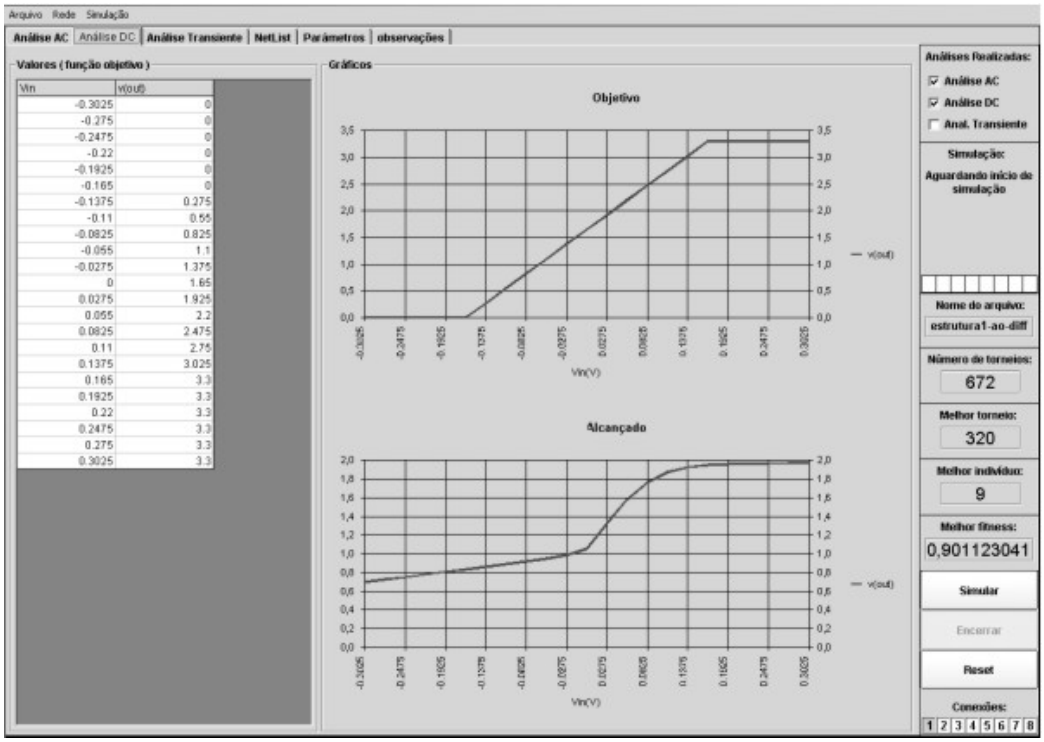


Figura 6-8 - Análise DC do amplificador operacional para estrutura 1 dos transistores programáveis

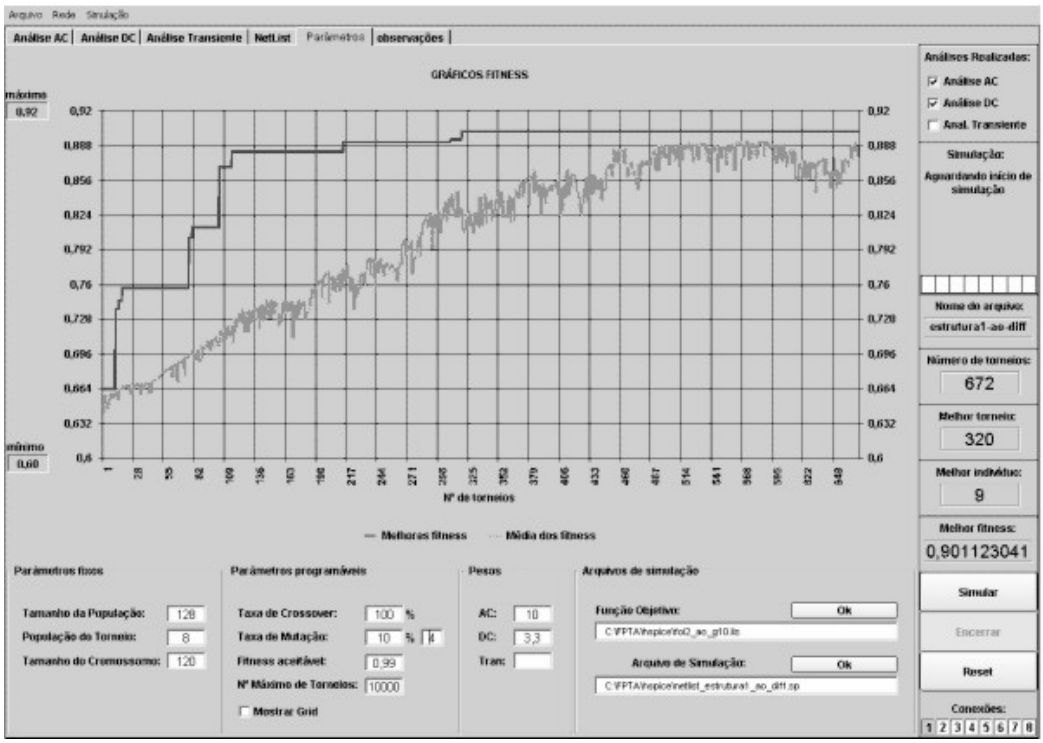


Figura 6-9 - Comportamento do algoritmo genético para o amplificador operacional da estrutura 1 dos transistores programáveis

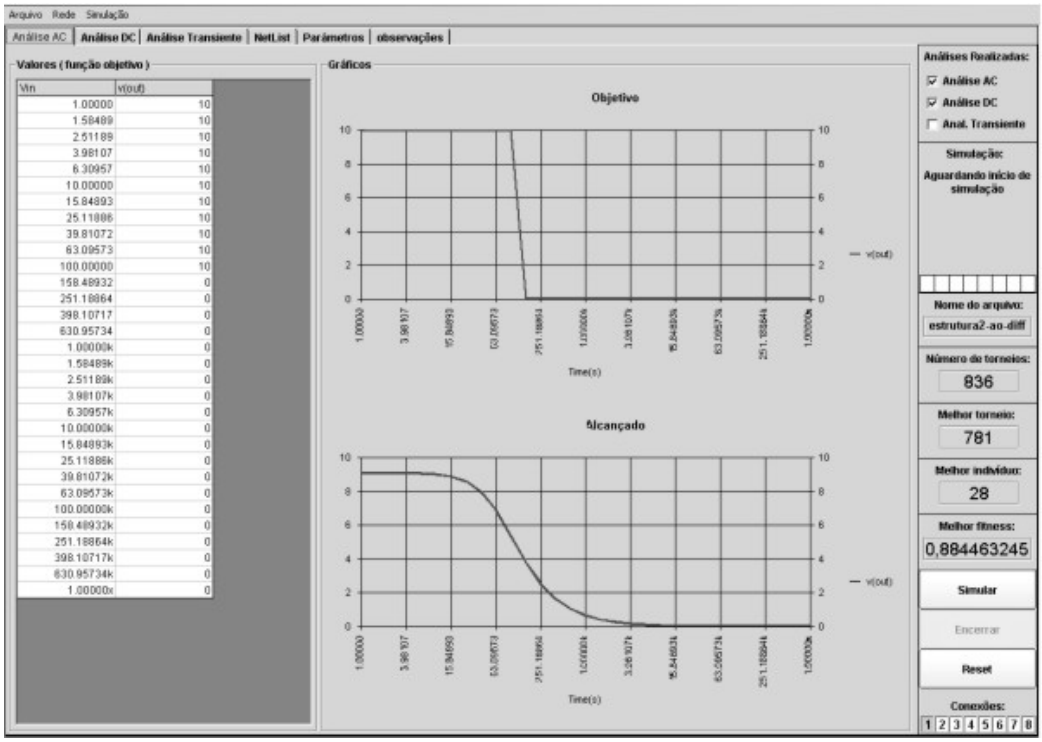


Figura 6-10 - Análise AC do amplificador operacional para estrutura 2 dos transistores programáveis

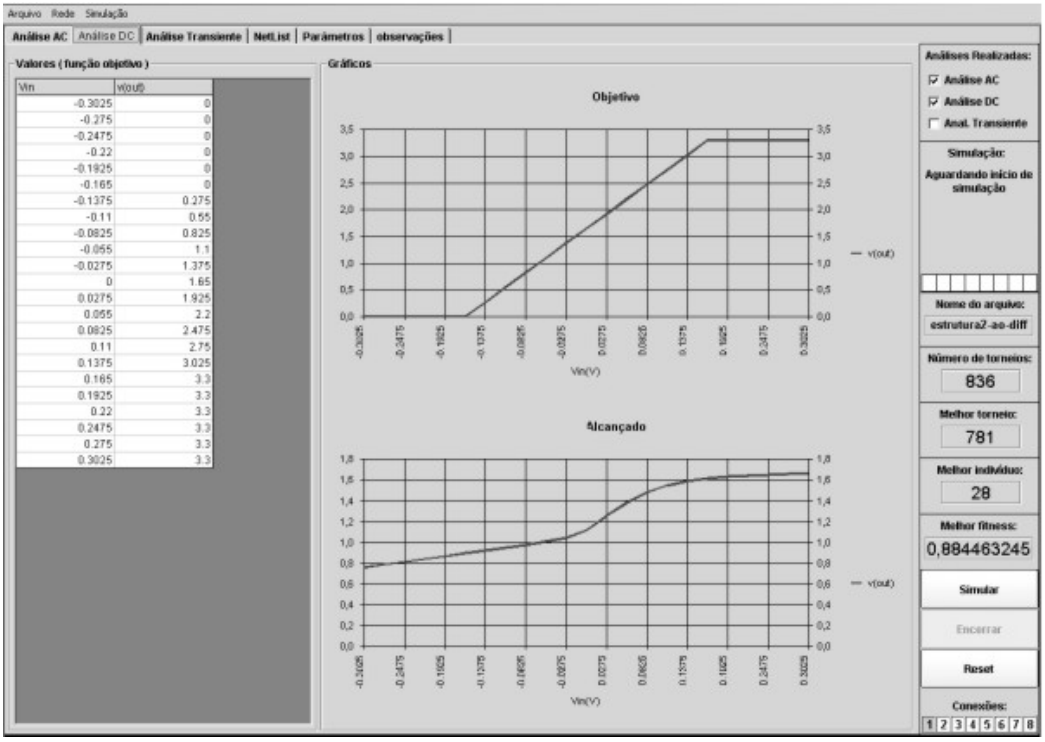


Figura 6-11 - Análise DC do amplificador operacional para estrutura 2 dos transistores programáveis

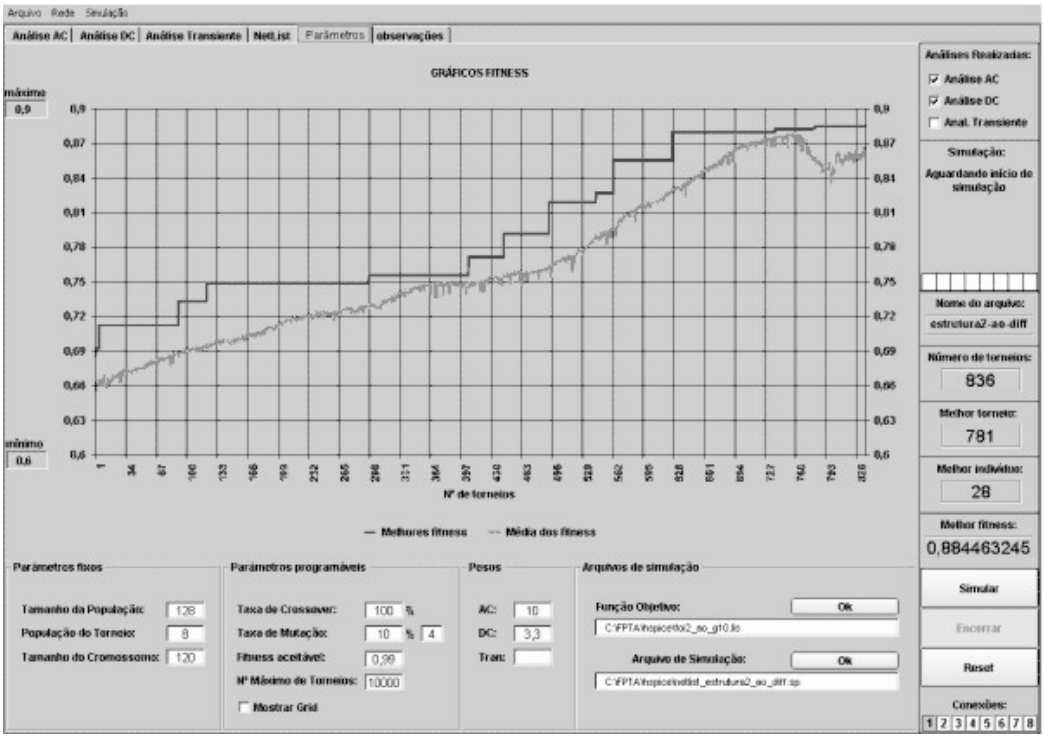


Figura 6-12 - Comportamento do algoritmo genético para o amplificador operacional da estrutura 2 dos transistores programáveis

7. Conclusão

A eletrônica evolucionária utilizando plataformas extrínsecas e intrínsecas apresenta algumas vantagens e desvantagens.

A utilização de ambiente extrínseco demonstra ser uma poderosa ferramenta de desenvolvimento para circuitos digitais e analógicos, contudo alguns inconvenientes podem ser evidenciados.

Na evolução extrínseca, a avaliação de desempenho dos projetos é realizada em software e para isso, são usados simuladores que utilizam modelos físicos dos componentes. Esses modelos são utilizados em projetos convencionais e, portanto, as equações utilizadas são otimizadas para as faixas mais comuns de operação dos dispositivos. Isso pode representar sérias restrições para processos evolucionários porque configurações de circuito não convencionais interessantes, podem ser descartadas pelo algoritmo evolucionário devido à imprecisões de modelo.

Um outro item a considerar diz respeito à capacidade de processamento necessário para executar as simulações implicando em sérias limitações de velocidade para evolução do circuito.

Em relação às plataformas intrínsecas, a modelagem não é necessária uma vez que o circuito é avaliado diretamente em hardware o que resulta em ganhos consideráveis na velocidade do processo evolutivo, além de permitir a exploração de efeitos dinâmicos, tais como: diferenças de propagação devido às portas dos transistores; interferências devido à proximidade de componentes e flutuações nas linhas de alimentação.

Apesar de ter sido desenvolvido no âmbito de ambientes tradicionais de simulação, o projeto da estrutura de barramento proposta neste trabalho a ser utilizada em um circuito integrado do tipo FPTA para uso como plataforma evolucionária intrínseca, teve como meta principal garantir a funcionalidade das estruturas propostas. Para isso, como etapa preliminar, foram efetuados testes com as estruturas propostas, visando verificar a degradação provocada pelas chaves que efetuam as interconexões dos componentes com o barramento em circuitos convencionais. Isso foi feito por transposição de circuitos convencionais, tais como amplificadores operacionais e de transcondutância para a forma de barramento, sem alteração tanto do tamanho dos transistores quanto aos valores das fontes utilizadas. Os resultados obtidos permitiram verificar que o desempenho dos circuitos testados não sofreu variações significativas que pudessem impactar o uso em hardware evolucionário.

Numa segunda etapa foram testados blocos do circuito proposto, com as estruturas dos transistores programáveis, usando a ferramenta desenvolvida para o processo evolucionário. Os testes demonstraram um comportamento satisfatório da estrutura de barramento permitindo atingir os resultados desejados nos processos evolucionários efetuados.

Na estrutura proposta, o teste de dois blocos de 8 transistores programáveis cada, com barramento externo de 16 vias requer um cromossomo de 544 bits,

equivalente a um espaço de busca na ordem de 10^{163}. Portanto, considerando-se o tempo de processamento extremamente elevado das evoluções extrínsecas, um programa mais extenso de testes só poderá ser realizado com a fabricação do circuito integrado em seqüência a este trabalho.

Sugere-se também em paralelo com o projeto de layout do circuito integrado, a implementação em forma discreta de uma versão reduzida do circuito composta por dois blocos de 8 transistores programáveis interligados por barramento. Os testes efetuados nesta estrutura podem auxiliar decisões relativas a desempenho e otimização de espaço no circuito final.

Uma outra sugestão é dar continuidade ao projeto da ferramenta de software utilizando uma rede de computadores específica para esse tipo de aplicação (GRID), tornando uma ferramenta poderosa para evoluções extrínsecas.

Referências:

BARRICELLI, N, A, 1957, *Symbiogenetic evolution processes realized by artificial methods*. Methodos, 9(35-36),

BREMERMANN, H. J., 1962 *Optimization through evolution and recombination in Self-Organizing Systems,* M. C. Yovits *et al.,* Eds. Washington,DC: Spartan.

CRAMER, N., L., 1985 *A representation for the adaptive generation of simple sequential programs*. In J.J. Grefenstette, editor, Proceedings of an International Conference on Genetic Algorithms and Their Applications, Carnegie-Mellon University, July 24-26, 1985.

DE GARIS, H., 1993 *Evolvable hardware: genetic programming of a darwin machine*. In: International Conference on Artificial Neural Networks and Genetic Algorithms, Innsbruck., 1993

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.

EBERBACH E., 2000 *Expressing Evolutionary Computation, Genetic Programming, Artificial Life, Autonomous Agents and DNA-Based Computing in $-Calculus* - Revised Version, Proc. 2000 Congress on Evolutionary Computation CEC'2000, San Diego, CA, 2000, 1361-1368.

FOGEL, L., J., 1966 A., J., OWENS and M., J., WALSH, *Artificial Intelligence Through Simulated Evolution*.

FRASER A.,S., 1970 BURNELL D., *Computer Models in Genetics*.

FUJIKI, C., DICKINSON, J., 1987 *Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma*. In J. J. Grefenstette, editor, Genetic Algorithms and their Applications: Proceedings of the second international conference on Genetic Algorithms, pages 236-240, MIT, Cambridge, MA, USA, 28-31 July 1987. Lawrence Erlbaum Associates.

GOLDBERG, D. E., 1985 *Genetic algorithms and rule learning in dynamic system control*, in Proc. 1st Int. Conf. on Genetic Algorithms and Their Applications. Hillsdale, NJ: Lawrence Erlbaum, pp. 8–15.

GUIMARÃES, A., L., P., 2006 *Estruturas CMOS Programáveis para Aplicação em Eletrônica Evolucionária* UFRJ Março 2006

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.

HIGUCHI T., T. NIWA, and T. TANAKA, 1993 *Evolving hardware with genetic learning: A first step toward building a Darwin machine*, in Animals to Animats 2: Proc. Second Int. Conf. Simulation of Adaptive Behavior, J.-A. Meyer, H. L. Roitblat, and S. W. Wilson, Eds: MIT Press, 1993, pp. 417--424.

HOLLAND J. H., 1962 *Outline for a logical theory of adaptive systems*, J. Assoc. Comput. Mach., vol. 3, pp. 297–314.

HOLLAND J. H., 1975 *Adaptation in Natural and Artificial System*". Ann Arbor, MI: Univ. of Michigan Press.

HOLLAND J. H. and J. S. REITMAN, 1978 *Cognitive systems based on adaptive algorithm*" in Pattern-Directed Inference Systems, D. A. Waterman and F. Hayes-Roth, Eds. New York: Academic.

HOLLAND J., H., 1975 *Adaptation in Natural and Artificial Systems*. AnnArbor, MI: Univ. of Michigan Press.

KOZA, J., R., 1989 *Hierarchical genetic algorithms operating on populations of computer programs*, in Proc. 11th Int. Joint Conf. on Artificial Intelligence, N. S. Sridharan, Ed. San Mateo, CA: Morgan Kaufmann,1989, pp. 768–774.

KOZA, J.R. 1992 *Genetic programming: on the programming of computers by means of natural selection*. Massachusetts: MIT Press, 1992.

KONNER, M. 1998 *A piece of your mind*. Science, New York, v. 281, p. 653-654, Jan. 1998.

LANGEHEINE A., J., BECKER, J., FOLLING, S., MEIER, K., SCHEMMEL, J., 2001 *CMOS FPTA chip for intrinsic hardware evolution of analog electronic circuits*, Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on 12-14 July 2001 Page(s):172 – 175

MARCHAL P., 1994 *Embryological development on silicon*, in Artificial Life IV, R. Brooks e P. Maes, Eds: MIT Press, 1994, pp. 365--366.

RECHENBERG, I., 1965 *Cybernetic solution path of an experimental problem,"Royal Aircraft Establishmen"t*, Library translation No. 1122, Farnborough, Hants., U.K., Aug.

ROMAHI, Y., 1998 *Engineering Systems from Nature Evolutionary Electronics* Dissertation

SANCHEZ, E. 1996 *Filed Programmable Gate Array (FPGA) Circuit*, in Towards Evolvable Hardware, pp. 1-18, E. Sanchez e M. Tomassini (editors), Springer-Verlag LNCS 1062.

SANTINI, C, S., 2001 *Desenvolvimento de uma plataforma reconfigurável analógica para a evolução intrínseca de circuitos* Dissertação, Departamento de Engenharia Elétrica Pontifícia Universidade Católica do Rio de Janeiro Julho 2001

SIPPER, M. 1995 *An introduction to artificial life. Explorations in Artificial Life*, p. 4-8, Sep. 1995. (Special Issue of AI Expert)

SIPPER, M. 1997 *A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems*. IEEE Transactions on Evolutionary Computation, v. 1, n. 1, p. 83-97, 1997.

SIPPER, M., RONALD, E.M.A. 2000 *A new species of hardware*, IEEE Spectrum, vol. 37, no. 3, pp. 59-64, março de 2000

STOICA, A., KEYMEULEN, D., TAWEL, R., SLAZAR-LAZARO, C., LI, W., 1999 *Evolutionary Experiments with a fine-grained reconfigurable architecture for analog and digital CMOS circuits*, Proceedings of the First NASA DoD Workshop on Evolvable Hardware, pp.76-84, IEEE Computer press., July, 1999.

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.

STOICA 2001 *Reconfigurable VLSI Architectures for Evolvable Hardware: From Experimental Field Programmable Transistor Arrays to Evolution-Oriented Chips* IEEE Transactions on very large scale integration (VLSI) systems, VOL. 9, NO. 1, February 2001

SUSSMAN, G. J., STALLMAN, R. M., 1975 *Heuristic Techniques in Computer-Aided Circuit Analysis*, *IEEE transactions on Circuits and System*s, vol. 22, 1975.

THOMPSON A., 1996 *An evolved circuit, intrinsic in silicon, entwined in physics*, in *Int. Conf. Evolvable Systems*: (Lecture Notes in Computer Science) Springer-Verlag, 1996, pp. 390–405.

ZEBULUM R., S., 2000 *The Design Process of an Evolutionary Oriented Reconfigurable Architecture* IEEE 2000

ZEBULUM, R. S., PACHECO, M. A., VELLASCO, M., 1996 *Evolvable Systems in Hardware Design: Taxonomy, Survey and Applications*, em Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), LNCS 1259, pp. 344-358, 1997. Conferência realizada em Tsukuba, Japão, Outubro,1996.

ZEBULUM, R.S., 2000 *A Flexible Model of a CMOS Field Programmable Transistor Array Targeted for Hardware Evolution*, ICES 2000

YAO, X., HIGUCHI, T., 1996 *Promises and Challenges of Evolvable Hardware*, Proceedings of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES96), LNCS 1259, pp. 55-80, 1997. Conferência realizada em Tsukuba, Japão, Outubro, 1996.