

IMPLEMENTAÇÃO DE UM ALGORITMO GENÉTICO EM HARDWARE.

Felipe Rampin Godoy, Suely Cunha Amaro Mantovani. – Engenharia Elétrica – Departamento de Engenharia Elétrica – Faculdade de Engenharia – Campus de Ilha Solteira.

Em meio a grande complexidade na qual estão envolvidos os atuais projetos de circuitos digitais, houve a necessidade do desenvolvimento de novas tecnologias capazes de reduzir o tempo de realização do projeto e o seu custo final. O aparecimento dos dispositivos lógicos programáveis ou Programmable Logic Devices- PLDs e, os primeiros trabalhos com Inteligência Artificial (AI) trouxeram uma ampla perspectiva de pesquisas nesta área. Cresce nos anos 80 o interesse dos pesquisadores por teorias biológicas que vêm gerando algoritmos que estão sendo aplicados na resolução de otimização de sistemas em geral adaptação e aprendizagem, tais como Redes Neurais ou Artificial Neural Networks-ANNs, Algoritmos Evolutivos baseados na Evolução Natural de Darwin como Algoritmos Genéticos, Programação Genética, Estratégia Evolutiva, etc.[1],[2] [3]

Para o desenvolvimento de projetos surge o Hardware Evolutivo (EHW) cuja a metodologia é capaz de executar grande quantidade de tarefas desde reconhecimento de padrões, síntese de circuitos digitais, à controle adaptativo. Hardware Evolutivo possui essa denominação por permitir uma mudança em sua arquitetura, estrutura ou funções de maneira dinâmica e autônoma de forma a adaptar-se as mudanças no meio, ou a falhas. Esse processo de adaptação ou reconfiguração de sua arquitetura torna-se viável com o uso dos algoritmos evolutivos e o desenvolvimento de dispositivos programáveis como Field Programmable Gate Arrays (FPGAs), que possibilitam a modificação via programação.

Este trabalho visa a implementação de um algoritmo evolutivo, neste caso o algoritmo genético (GA) [4] em hardware, definido como EHW do tipo intrínseco, e assim realizar síntese de circuitos digitais combinacionais. A principal vantagem da realização do GA em hardware ao invés da simulação em *software* com o mesmo objetivo[5],[6],[7] é a velocidade do processamento importante para aplicações em tempo real.

O processo de síntese é iniciado pela descrição de um problema ou de sua especificação funcional, como uma tabela verdade onde se busca uma topologia, escolha de tipos de componentes adequados e a otimização do circuito (topologia mínima)

A Síntese de Circuitos Digitais combinacionais foi uma das primeiras aplicações da eletrônica Evolutiva encontradas na literatura. A característica desta classe de projetos, onde são poucas as restrições, torna o desempenho dos sistemas evolutivos satisfatórios, quando comparados às ferramentas convencionais de projeto.

O procedimento de projeto para circuitos combinacionais tem evoluído através de técnicas como mapa de Karnaugh [8] - largamente usados para minimizar funções lógicas, ou os métodos de Quine-McCluskey - adequados a implementação computacional, aos atuais softwares comerciais tais como Expresso.

Neste trabalho a metodologia para desenvolvimento deste projeto usando esta técnica, constitui-se das seguintes etapas:

- A primeira etapa consiste na simulação de um algoritmo genético clássico para resolver um problema qualquer de otimização, visando adquirir o conhecimento e medir a sensibilidade dos operadores e dos parâmetros do algoritmo, como ilustrado na figura 1. Para isso, pode-se usar uma linguagem qualquer de programação de alto nível, como a do Matlab v.7.21.
- Na segunda etapa é proposta uma representação para os circuitos em forma cromossômica (string binário), visando compor uma população de indivíduos (ou circuitos) e a implementação de um algoritmo genético em uma linguagem de descrição de hardware, por exemplo AHDL visando programar (reconfigurar) dispositivos lógicos programáveis através de ambientes de software como Max-plusII do fabricante Altera.

A programação dos operadores e circuitos randômicos, do GA é realizada conforme proposta de uma arquitetura mostrada na figura 2. O circuito funciona com o sincronismo de um relógio como em um microprocessador. O resultado final é um string de bits que deve

corresponder ao melhor circuito encontrado para realizar uma função especificada por uma tabela verdade.

- Na última etapa realiza-se testes com a montagem em placa e a reconfiguração do circuito em plataforma altera como a família Flex10K (altera) escolhida por comportar médios circuitos (10.000 a 250.000 portas) e apresentar maior disponibilidade de memória.

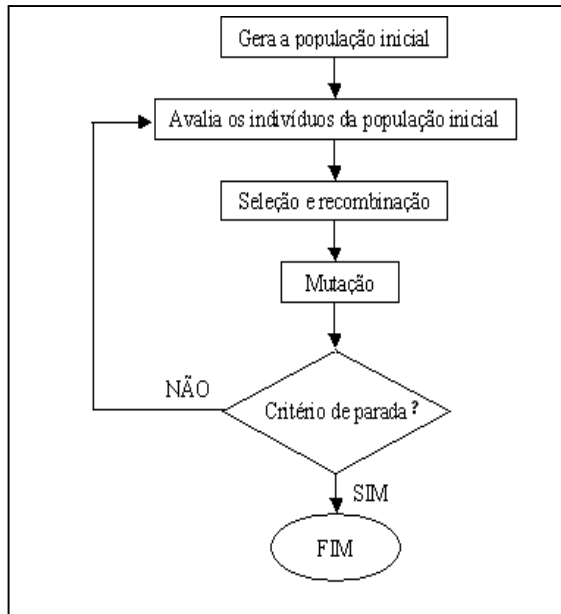


Figura 1: Fluxograma para um Algoritmo Genético Clássico.

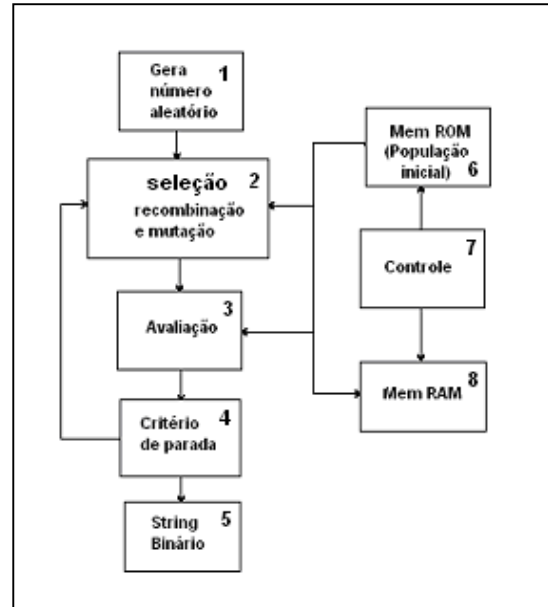


Figura 2: Diagrama de blocos para o GA implementado em linguagem HDL

Inicialmente os strings são gerados de maneira aleatória e submetidos aos processos de avaliação, seleção, recombinação e mutação, até que ocorra a convergência do algoritmo, para o resultado desejado.

O processo de avaliação do algoritmo (ou função objetivo), definido de acordo com o problema, é efetivado pelo acerto da tabela verdade do circuito que se deseja implementar, sendo responsável pelo desempenho de cada string, que é usado no processo de seleção, e como critério de parada do sistema. Como o algoritmo se baseia na teoria da seleção natural, apenas os indivíduos que possuem maior proximidade da função especificada inicialmente, devem continuar no processo (guardados em memória RAM). Desta forma o processo de seleção é iniciado sorteando alguns strings, de maneira aleatória, e verificado seus desempenhos. Dentre os sorteados, aquele que melhor for enquadrado pela função objetivo é passado a diante. Na etapa de recombinação ocorre a troca de informação binária entre dois strings a partir de um ponto, obedecendo a uma taxa de recombinação obtida aleatoriamente, que define se haverá a troca de material, ou se ao final do processo o string possuirá a mesma configuração. O string resultante é submetido ao operador de mutação onde ocorre a troca de um único bit e de maneira análoga ao processo de recombinação. O ponto onde ocorrerá ou não essa troca é definido aleatoriamente. Em todos o processo, a geração dos números aleatórios representam um papel extremamente importante para a eficiência do algoritmo.

O exemplo a seguir mostra a otimização de um circuito onde se busca a melhor solução de uma representação binária que possui a maior variação possível de zero para um, ou de um para zero, dentro de um string de oito bits.

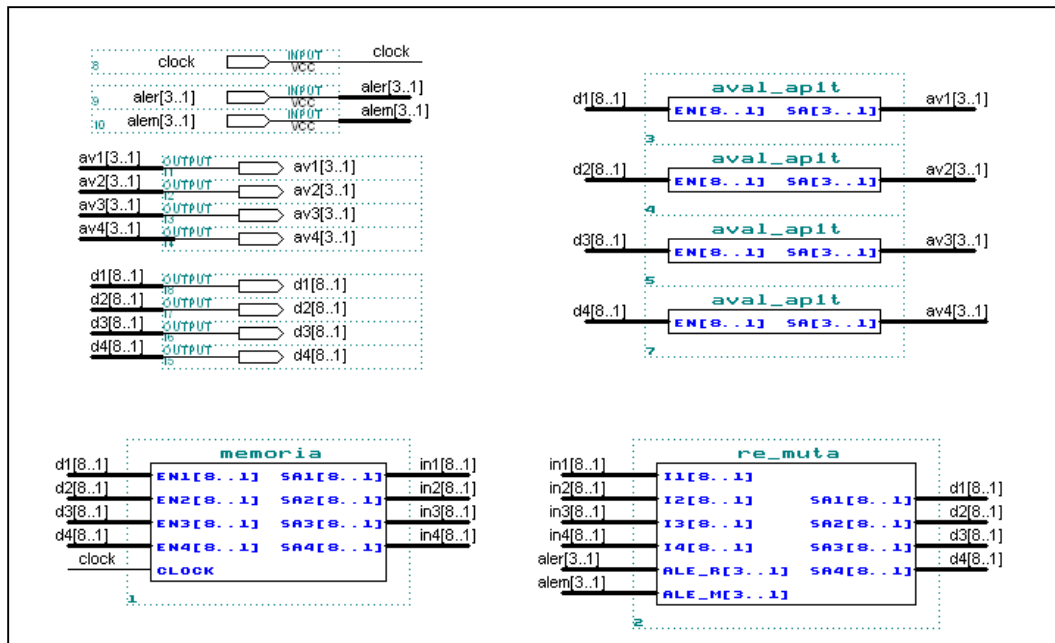


Figura 3: Programa em esquemático no simulador Max-Plus II

A Figura 3, mostra o esquemático implementado que engloba a função avaliação, os processos de recombinação e mutação, e uma memória que armazena os resultados (quatro indivíduos) das novas gerações.

As simulações a seguir relacionam o comportamento da função avaliação (aval_ap1), recombinação e mutação (re_muta).

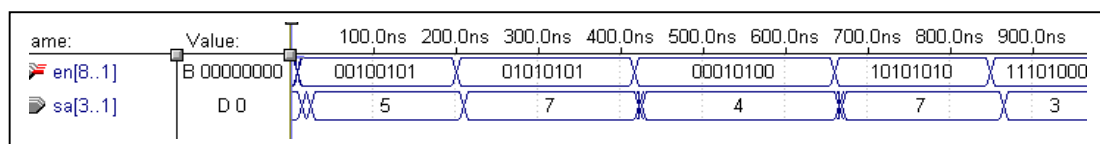


Figura 4: Simulação da função avaliação

Na figura 4, conta-se a variação binária, de '0' para '1' ou de '1' para '0', no indivíduo, e após o término da varredura do string, tem-se o número de variações totais. O vetor binário **en**, representa o indivíduo de entrada para o processo de avaliação e **sa** o desempenho de cada string avaliado.

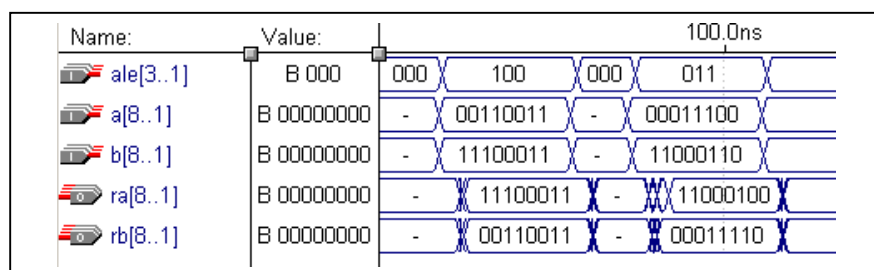


Figura 5: Processo de recombinação

A figura 5 mostra a recombinação, onde os vetores **a** e **b**, representam os strings de entrada, o vetor **ale** corresponde ao ponto onde se iniciará a troca de material binário e os vetores **ra** e **rb** são os strings resultantes do processo de recombinação, para uma taxa de recombinação, **tc** de 100%.

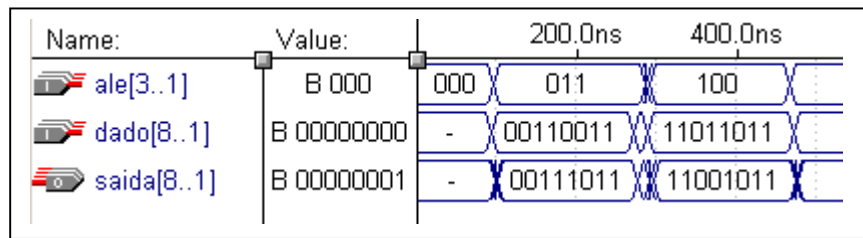


Figura 6: Processo de mutação

A simulação do processo de mutação é vista na figura 6, onde o vetor **ale** representa o bit onde haverá a troca de informação, **dado** é o string de entrada e **saida** é o resultado.

Os resultados preliminares do algoritmo genético implementado em hardware no dispositivo MAX7000S, EPM7128SLC84-7 mostram-se satisfatórios para este exemplo. A idéia de implementar algoritmo genético em *hardware* é importante não somente para projetos de circuitos digitais propriamente ditos, mas também em sistemas de controle e realimentação e reconhecimento de falhas gerando elementos redundantes.

Referências Bibliográficas

- [1] Z. Michalewicz, "Genetic Algorithm + Data Structures = Evolution Programs", 2nd ed New York: Springer-Verlag, 1996.
- [2] R. S. Zebulum, M. A. Pacheco, M. Vellasco. "Evolvable System in Hardware Design: Taxonomy, Survey and Applications", LNSC – vol 1259, 1997.
- [3] T. Higuchi, M. Iwata, I Kajitani., H Iba., Y. Hirao, T Furuya. and B. Manderick "Evolvable Hardware and its Application to Pattern Recognition and Fault - Tolerant Systems", Lecture Notes in Computer Science, vol. 1062, pp. 118-135, 1996.
- [4] Tommi Rintala, "Hardware Implementation Of GA", acessado em janeiro 13, 2003 <http://www.uwasa.fi/cs/publications/2NWGA/node60.html>.
- [5] S. C. A. Mantovani J. R. Oliveira, "Síntese de Circuitos Digitais por Evolução de Circuitos".In: Anais do XXXVII Simpósio Brasileiro de Pesquisa Operacional, CD-V1, p. 1820-1831, São João Del Rei- Minas Gerais, 2004.
- [6] J. L. Sushil. "Genetic Algorithms as a Computational Tool for Design" , PhD thesis, Department of Computer Science, Indiana University, August 1993.
- [7] E.F. Goulart, So., S. C. A. Mantovani, "EHW Aplicado a Síntese de Circuitos Digitais Usando Representação por Portas Lógicas".In: Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional, CD, Goiânia -GO, 2006.
- [8] I.V. Idoeta, F.G. Capuano. " Elementos de eletrônica digital". 13ª edição. Livros Érica Editora Ltda, 1988.