NML DESIGNER: uma Ferramenta para Projeto e Simulação de Circuitos Nanomagnéticos

Thiago R B S Soares, Isaias F. Silva, Omar P Vilela Neto

NML DESIGNER: uma Ferramenta para Projeto e Simulação de Circuitos Nanomagnéticos

Thiago R. B. S. Soares*, Isaias F. Silva*, Omar P. Vilela Neto*
*Department of Computer Science
Federal University of Minas Gerais, Belo Horizonte, Minas Gerais
Email: [thiagorbss, isaiasfaria, omar]@dcc.ufmg.br

***Resumo*—Com a tecnologia CMOS chegando próximo do seu limite de miniaturização, aliado ao aumento do consumo de energia, alternativas para substituí-la vem sendo investigadas nas ultimas décadas. Dentre as diversas alternativas, QCA (*Quantum dot Cellular Automata*) e suas variações destacam-se pela alta frequência e baixo consumo. *Nanomagnetic Logic* (NML) é uma evolução do QCA eletrônico que funciona a partir da interação do campo magnético entre nano ímãs. Para o contínuo avanço da tecnologia, ferramentas que auxiliam o desenvolvimento de circuitos são necessários. Nesse artigo é apresentado o NML Designer, uma ferramenta para projeto e simulação de circuinos NML.**

I. Introdução

Com a tecnologia CMOS chegando próximo do seu limite de miniaturização [1], aliado ao aumento do consumo de energia, alternativas para substituí-la vem sendo investigadas nas ultimas décadas. Dentre as diversas alternativas, QCA (*Quantum dot Cellular Automata*) e suas variações destacam-se pela alta frequência e baixo consumo.

Automatos celulares com pontos quanticos (QCA) surgiram na década de 90 como uma possível alternativa aos transistores de silício que já mostravam indícios de seus limites. A unidade básica de QCA é uma célula em escala nanométrica que apresenta quatro pontos quânticos e dois elétrons livres. Devido ao formato da célula (quadrado) e da força de repulsão dos elétrons, uma célula QCA apresenta apenas dois estados estáveis. Para os primeiros circuitos QCA [2] foram usados como pontos quânticos ilhas de alumínio e serviram para provar a funcionalidade da tecnologia. Entretanto, pontos quânticos feitos a partir de ilhas de alumínio funcionam apenas em temperaturas muitos baixas (0,1K a 4K) e a produção em larga escala é inviável devido ao pequeno tamanho das células.

Nanomagnetic Logic (NML) é uma evolução do QCA eletrônico que funciona a partir da interação do campo magnético entre nano ímãs. A unidade básica de NML são ímãs pequenos o suficiente para apresentarem um único domínio magnético e assim terem seus polos bem definidos nas extremidades da célula. Com isso, seus vetores de magnetização são bem definidos ao ponto de terem apenas dois estados estáveis de magnetização. Ao contrário de QCA, que apresenta células com formato único, uma célula NML é retangular e ainda é possível mudar seu formato para criar novas células com funcionamento específico [3].

Assim, uma célula NML pode ser representada como um retângulo e seu vetor de magnetização como um seta que pode ser direcionada para cima ou para baixo como apresentado na Figura 1 A). Um circuito NML pode ser dividido em zonas e cada zona está em um estado diferente. Os estados são *Hold, Switch e Reset*. Esses estados são determinados por três sinais iguais e periódicos com diferenças de 120 graus entre cada. A borda de subida do sinal de clock representa a presença de campo magnético externo. Esse campo leva os ímãs daquela zona para um meta estado de alta energia onde seu vetor de magnetização fica na direção do eixo "difícil". Assim, os ímãs na zona de *Hold* matem seu estado de magnetização, ímãs em *Switch* tem sua magnetização calculada em função da vizinhança e os ímãs em *Reset* ficam em um meta estado com magnetização "nula"como apresentado na Figura 1 C.

Para analisar um circuito nanomagnético normalmente se usa o OOMMF [4], mas apenas circuitos muito pequenos podem ser simulados em tempo viável. Para circuitos QCA o QCADesigner [5] é o mais utilizado e para NML o ToPoliNano [6] apresenta uma ferramenta interessante e promissora apesar das suas limitações. ToPoliNano automaticamente gera o *layout* físico do circuito partindo de uma descrição VHDL. O circuito então é simulado baseado em um modelo comportamental. O modelo é baseado em resultados obtidos na literatura ou validados usando o OOMMF para simular as portas lógicas básicas.

Nesse artigo é apresentado uma ferramenta capaz de simular circuitos NML onde o projetista posiciona as células uma a uma até montar um circuito. Por mais que seja demorado a criação de projetos pequenos, há uma liberdade maior no momento do desenho do circuito, possibilitando ao projetista criar projetos da maneira que lhe for mais conveniente. Nesse artigo será discutido alguns trabalhos presentes na literatura na seção de Trabalhos Relacionados. Na seção NML Designer será apresentado a ferramenta e seu funcionamento, na seção de Testes será feito uma comparação de desempenho entre o ToPoliNano e NML Designer, na Conclusão será discutido os resultados obtidos e por fim alguns trabalhos futuros serão apresentados.

II. Trabalhos Relacionados

Por NML ser uma tecnologia nova existem poucas ferramentas que auxiliem no projeto e simulação de circuitos nanomagnéticos. ToPoliNano [6] é a única até o presente momento capaz de simular o funcionamento de circuitos NML. A ferramenta é um trabalho em progresso e por isso apresenta algumas limitações.

ToPoliNano foi desenvolvida em C/C++ e seu projeto segue um fluxo de trabalho apresentado na Figura 2. O ***Logic***

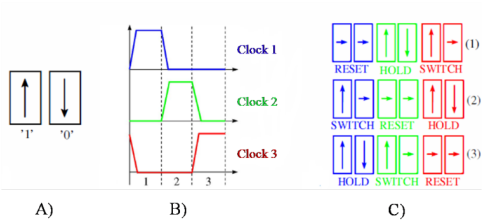

Figura 1. A) Representação de um ímã e seu vetor de magnetização. B) Sinais de clock. C) Ímãs em três zonas e seus estados.

Synthesizer analiza o código VHDL e ele é mapeado nas portas lógicas disponíveis na biblioteca (portas da maioria, portas AND ou OR, inversores e cruzamentos de fio). Essa é a primeira restrição da ferramenta. O código VHDL aceito pela ferramenta é restrito às portas lógicas AND e OR e em forma de componentes a serem inseridas no código e não como construtores básicos da linguagem. Dessa forma o VHDL é bastante restrito e projetar circuitos simples como um somador de 1 bit torna-se uma tarefa árdua.

O ***Parser*** pega o arquivo gerado na etapa anterior e o traduz em um grafo que representa a *netlist* do circuito. O ***Place & Route*** cria o *layout* do circuito baseado na *netlist* onde as portas lógicas são posicionadas e então conectadas para formar o circuito final.

Na etapa ***Simulation*** o circuito é simulado usando um modelo comportamental. Esse modelo é baseado numa aproximação de três estados onde cada ímã tem apenas três possíveis estados (lógico '0', lógico '1' e RESET). Esse modelo é validado usando as portas lógicas básicas presente na literatura. Essa abordagem garante a corretude do comportamento do circuito e ainda permite a simulação rápida de milhares de ímãs.

Esse algoritmo de simulação é executado numa matriz de posições onde cada posição pode ter ou não um ímã. O algoritmo percorre cada coluna da matriz duas vezes determinando o valor dos ímãs de uma zona. Um ímã tem seu valor determinado pelos ímãs próximos e são considerados apenas os que estão na horizontal e vertical. Para cada zona é feito outro passeio na matriz de forma que todas as entradas sejam testadas no circuito. A determinação das zonas é sempre no mesmo sentido (esquerda para direita ou o inverso) e em cada zona é permitido no máximo 5 ímãs [7].

Por ter muitas etapas automatizadas, o projetista perde o controle do circuito gerado. Muitas vezes é necessário criar circuitos com a menor quantidade de ímãs possível e não há garantia disso na ferramenta ToPoliNano. Assim, surge a necessidade de projetar circuitos posicionando os ímãs um a um, assim como no QCADesigner [5]. NML Designer, assim como ToPoliNano, usa um algoritmo de simulação comportamental baseado numa aproximação de três estados porém com maior flexibilidade de projeto e mais rápido.

III. NML Designer

NML Designer surgiu das limitações apresentadas em ToPoliNano e teve como inspiração o QCADesigner. NML Designer é atualmente uma ferramenta web onde o usuário pode criar seu circuito posicionando os ímãs um por um e

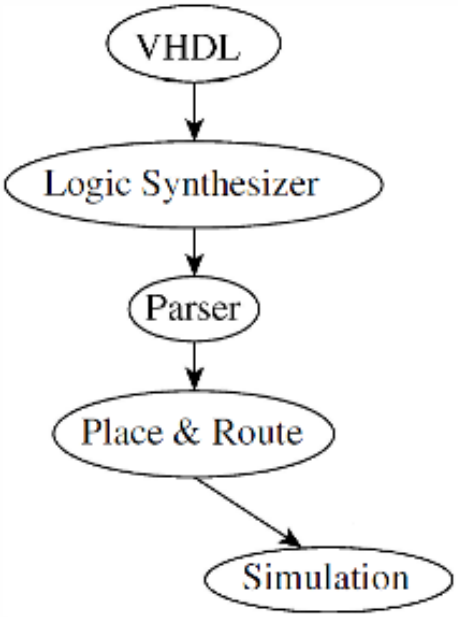

Figura 2. Fluxo de projeto da ferramenta ToPoliNano

validá-lo através de uma simulação. A ferramenta pode ser dividida em duas partes independentes: a interface e o algoritmo de simulação. O algoritmo de simulação foi desenvolvido em Java e a interface em HTML5, JavaScript e CSS. Tais partes serão explicadas a seguir.

A. Algoritmo de Simulação

Ao contrário do algoritmo de simulação do ToPoliNano (que usa uma matriz), o NML Designer usa um grafo como estrutura para representar o circuito. Cada vértice do grafo representa um ímã e as arestas conectam ímãs próximos. As arestas tem peso e seu valor pode ser 1 ou -1 dependendo do arranjo dos ímãs, como apresentado na Figura 3.

Para determinar o valor de todos os ímãs para um determinado conjunto de entrada é feito um passeio no grafo partindo de todas as entradas até que todas as saídas tenham algum valor. O algoritmo respeita as restrições das zonas de clock, assim é garantido a sincronização da informação. Ainda é possível utilizar as portas AND e OR de duas entradas definidas em [3] e o cruzamento de informação é permitido através de um ímã especial.

O próprio algoritmo de simulação pode ser dividido em duas etapas: a leitura dos arquivos de entrada e alocação de memória e a rotina que realiza o passeio no grafo determinando os valor dos ímãs. Essa divisão é importante atualmente pois o gargalo de desempenho do algoritmo é a etapa de leitura de arquivos e alocação de memória.

B. Interface Web

Para essa versão do NML Designer foi desenvolvido uma interface Web com diversas funcionalidades. A interface permite abrir e salvar projetos no formato XML, posicionar os ímãs, mudar a zona de um ímã, adicionar uma porta AND de duas entradas e assim sucessivamente. Todas as opções

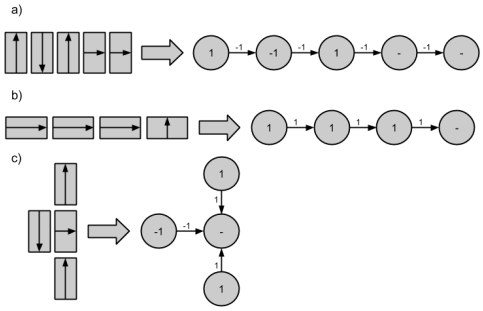

Figura 3. Representação do circuito no grafo.

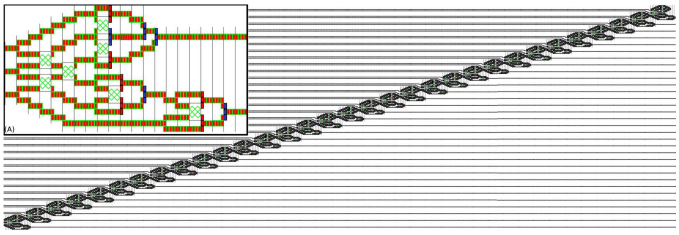

Figura 4. Em foco um somador de 1 bit. Na imagem maior o somador de 32 bits.

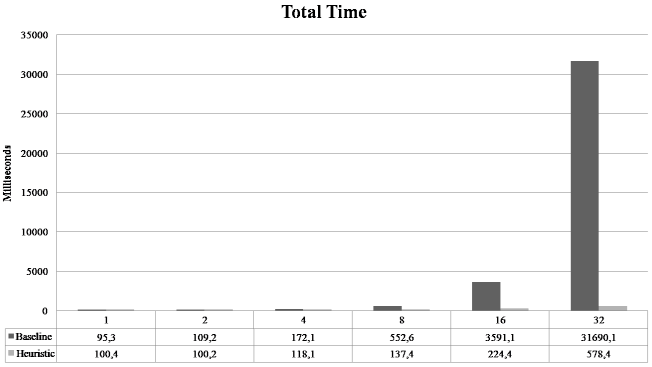

Figura 5. Média do tempo gasto em 10 execuções para cada somador.

estão listadas no menu lateral do site. É importante dizer que a interface por ser facilmente mudada para outra linguagem, basta que ela respeite as restrições de entrada e saída do algoritmo de simulação.

IV. Testes

Para os testes de desempenho foram gerados somadores de 1, 2, 4, 8, 16 e 32 bits baseados no apresentado em [6]. A Figura 4 ilustra um somador de 32 bits. Tal circuito é representado num matriz com mais de 900 mil posições e tem aproximadamente 146 mil ímãs. O gráfico da Figura 5 apresenta o tempo gasto pelo *baseline* (ToPoliNano) e pela *heurística* (NML Designer).

Note que para o somador de 32 bits, o maior circuito testado, o NML Designer demorou pouco mais de meio segundo enquanto o ToPoliNano demorou mais de 30 segundos. Tal fato se dá pelo ToPoliNano ter complexidade computacional $O(n^3)$, enquanto o algoritmo de simulação do NML Designer ser, no pior caso, $O(n^2)$.

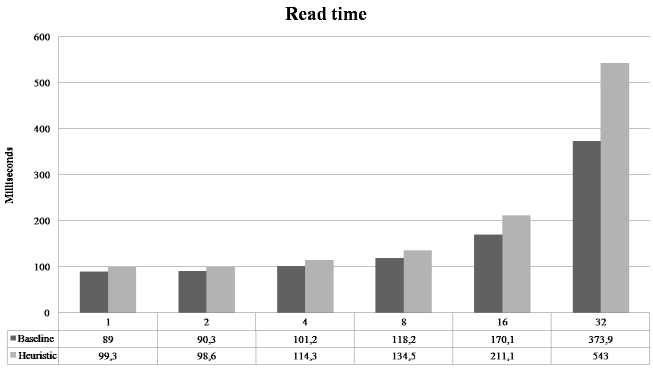

Figura 6. Média do tempo gasto na leitura e alocação de memória

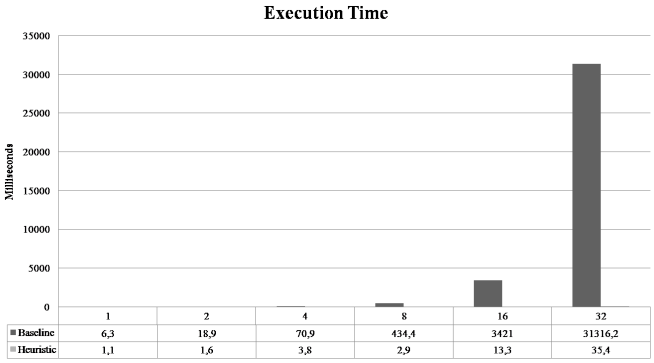

Figura 7. Média do tempo gasto para calcular o valor de todos os ímãs.

Na Figura 6 é apresentado o tempo gasto com leitura e alocação de memória e na Figura 7 é apresentado o tempo gasto com a determinação do valor dos ímãs para os dois algoritmos. Como dito anteriormente, a leitura e alocação de memória é o gargalo de desempenho do NML Designer.

V. Conclusão

Por NML ser uma tecnologia nova há poucas ferramentas para projeto e simulação de seus circuitos. ToPoliNano é uma alternativa interessante mas tem muitas limitações, o que a torna pouco atraente quando se deseja, por exemplo, projetar um circuito otimizando a área. NML Designer realizar uma simulação comportamental igual ao apresentado na literatura [6] e de forma mais eficiente além de dar ao projetista total liberdade de posicionamento dos ímãs e atribuição das zonas de clock. Atualmente a versão web é acessível pelo site www.nanocomp.dcc.ufmg.br/temp/NMLDesigner/demo/index.php.

VI. Trabalhos Futuros

Para trabalhos futuros será desenvolvido outro algoritmo de simulação mais preciso e que garanta mais robustez ao projeto. Em paralelo será desenvolvido algoritmos evolucionários para descobrir circuitos como somadores e multiplexadores com a menor quantidade de células.

Referências

[1] R. K. Cavin, P. Lugli, and V. V. Zhirnov, "Science and engineering beyond moore's law," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1720–1749, 2012.

[2] G. Snider, A. Orlov, I. Amlani, G. Bernstein, C. Lent, J. Merz, and W. Porod, "Experimental demonstration of quantum-dot cellular automata," *Semiconductor Science and Technology*, vol. 13, no. 8A, p. A130, 1998.

[3] M. Niemier, E. Varga, G. H. Bernstein, W. Porod, M. T. Alam, A. Dingler, A. Orlov, and X. S. Hu, "Shape engineering for controlled switching with nanomagnet logic," *Nanotechnology, IEEE Transactions on*, vol. 11, no. 2, pp. 220–230, 2012.

[4] M. J. Donahue and D. G. Porter, *OOMMF User's guide*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1999.

[5] K. Walus, T. J. Dysart, G. Jullien, R. A. Budiman *et al.*, "Qcadesigner: A rapid design and simulation tool for quantum-dot cellular automata," *Nanotechnology, IEEE Transactions on*, vol. 3, no. 1, pp. 26–31, 2004.

[6] M. Vacca, S. Frache, M. Graziano, and M. Zamboni, "Topolinano: a synthesis and simulation tool for nml circuits," in *Nanotechnology (IEEE-NANO), 2012 12th IEEE Conference on*. IEEE, 2012, pp. 1–6.

[7] G. Csaba and W. Porod, "Behavior of nanomagnet logic in the presence of thermal noise," in *International Workshop on Computational Electronics*, 2010, pp. 1–4.