ESPECULAÇÃO SOBRE ALGUMAS VANTAGENS DA ASSOCIAÇÃO ENTRE UMA ARQUITETURA COMPUTACIONAL ASSÍNCRONA E A LÓGICA FUZZY

Leo Weber
Professor da Fundação Liberato Salzano
Contato: leoweber@uol.com.br

Resumo

O propósito deste artigo é fazer uma análise sobre vantagens relacionadas com a associação entre um circuito computacional assíncrono e a lógica fuzzy, com características potencialmente significativas.

Palavras-chave: Arquitetura de computadores. Lógica assíncrona. Lógica difusa.

Abstract

The purpose of this article is to do an analysis about advantages related with the association between an asynchronous logic circuit and fuzzy logic, with important potential features.

Keywords: Computer´s architecture. Asynchronous logic. Fuzzy logic.

A arquitetura computacional assíncrona baseada em memória

Desde os primeiros projetos iniciais, com a utilização de relés e válvulas eletrônicas, os computadores conheceram aperfeiçoamentos incessantes no que diz respeito a seus componentes eletrônicos (*hardware*), chegando até os modernos circuitos integrados. No entanto, os princípios de base e a forma como os elementos de um computador estão organizados, sua arquitetura, continuam os mesmos [BRET]. A razão disto está no projeto desenvolvido por Von Neumann, matemático que concebeu a organização das máquinas computacionais. Mesmo com o advento dos microprocessadores, eles continuam sendo compostos por dispositivos de entrada, de saída, unidade de memória, unidade aritmética e unidade de controle, com processamento serial, sincronismo dos eventos a partir de um oscilador (relógio), controle centralizado e comunicação baseada em endereçamentos. O fluxo de dados entre o processador e a memória é o gargalo de um computador seqüencial comandado por relógio. A origem do problema é que a memória foi projetada para acessar uma única localização em cada ciclo de instrução. Enquanto mantivermos este projeto, a maneira de aumentar a velocidade do computador será reduzindo o tempo do ciclo, ou seja, aumentando a freqüência do relógio [HILL].

Historicamente, os circuitos assíncronos foram conhecidos e usados antes dos circuitos síncronos serem desenvolvidos e, por se tratar de um sistema baseado em eventos, não há relógio, onde quem limita a velocidade é o tempo necessário para propagar a informação por seus circuitos [MANO] [UNGE]. Desta forma, há um

benefício potencial de ganho de velocidade, pois cada computação é finalizada no tempo necessário para aquele processamento em particular, e não para o pior caso [BRZO] [TIND].

Há várias concepções de projeto com lógica assíncrona, nas quais inúmeros grupos de pesquisa atuam simultaneamente, como, só para citar, [DAVI], [EBER], [NOWI] e [SUTH]. Uma arquitetura assíncrona passível de implementação é a baseada em uma memória [MANO] [MART] [MYER] [ROTH] [TIND]. Sua simplicidade, a grande capacidade de expansão (é só ligar mais memórias em paralelo para se obter um número maior), o uso de componentes de baixo custo e de fácil obtenção, a possibilidade do uso de diferentes plataformas de *hardware* e a velocidade de processamento definida pela transição das entradas, gera uma excelente relação custo–benefício [*Figura 1*]. A nomenclatura deste tipo de máquina é "n-m-p" [CUES] [MART], onde:

n = variáveis *booleanas* independentes (entradas);
m = variáveis *booleanas* dependentes (saídas);
p = variáveis *booleanas* internas (realimentação ou memórias).

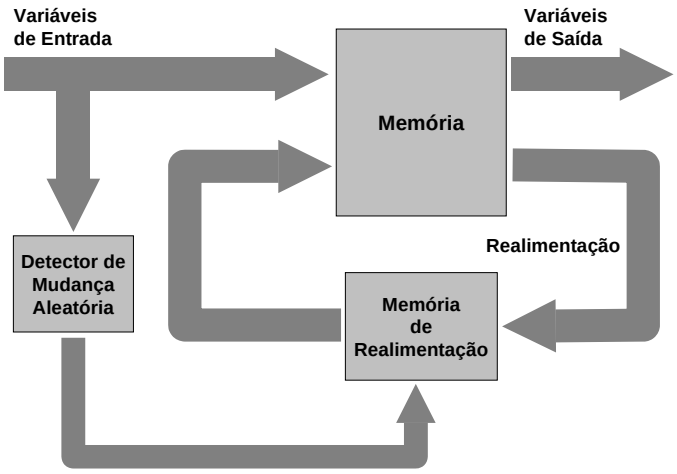

Figura 1: Arquitetura computacional assíncrona baseada em memória

Lógica fuzzy

A lógica *fuzzy* foi estruturada em 1965 pelo Dr. Lofti A. Zadeh (Universidade da Califórnia, Berkeley), pesquisador que cunhou o termo para tratar e representar incertezas. *Fuzzy*, em inglês, significa incerto, duvidoso, difuso, nebuloso. Expressa exatamente os valores com que lida, permitindo representar graus de certeza, ou valores de pertinência, intermediários entre os valores extremos de verdadeiro e falso da lógica clássica (bivalente). A lógica *fuzzy* surge da idéia de mapear variáveis que não tem equivalência matemática definida, como, por exemplo, o grau de certeza sobre se choverá ou não hoje. Para incluir o conceito de verdade parcial, grau de certeza ou grau de pertinência, esta teoria é um conjunto estendido da lógica *booleana* (clássica ou convencional).

Estruturas com lógica *fuzzy* são largamente aplicadas em soluções onde se requer a introdução do conhecimento especializado humano [ZADE], ou ainda, em sistemas de controle e suporte à decisão onde a descrição do problema não pode ser feita de forma precisa [COXE]. O aspecto principal desta abordagem é a capacidade de capturar com

clareza as várias nuances dos conceitos psicológicos utilizados pelos seres humanos em seu raciocínio usual, sem a necessidade de enquadrá-lo obrigatoriamente em modelos por vezes incompatíveis com o grau de difusão apresentado no contexto em questão [MEND].

Por outro lado, a imprecisão associada à descrição de sistemas leva ao tratamento difuso, ou seja, se o conhecimento que possuímos a respeito da estrutura interna do objeto em estudo não é suficiente para formularmos um modelo matemático, podemos nos valer de um método capaz de expressar de uma maneira sistematizada quantidades imprecisas, vagas, mal-definidas [KOSK]. Mas, ao contrário de muitas áreas da ciência contemporânea, a abordagem com lógica *fuzzy* não exige, necessariamente, grande quantidade de conhecimentos matemáticos, pois controladores baseados em regras tomam suas informações do senso comum e da experiência dos operadores [ALTR].

A união das duas concepções

A união de uma arquitetura computacional assíncrona com a lógica *fuzzy*, especula-se, é capaz de enfrentar positivamente importantes questões como as que seguem:

a) potencializar ganho na velocidade de processamento, cuja discussão não é aprofundada neste artigo;

b) desenvolver projetos que fomentem a validação, a flexibilização, o teste e a modularidade, objetos de nossa discussão adiante.

Validação intrínseca de projeto

Existem duas classes de métodos de validação de sistemas digitais. Os métodos pertencentes à primeira classe, chamados de métodos de validação de projeto, são a simulação e a verificação. Os métodos pertencentes à segunda classe, chamados genericamente de teste, são a modelagem e simulação de falhas, geração de testes e projeto visando a facilidade dos testes. Um fator importante para a aplicação desses métodos é a especificação ou descrição do sistema a ser projetado através de linguagens de descrição de *hardware* [DEHO].

O projeto de um sistema digital deve resultar em informações essenciais para a sua implementação: informações estruturais, que descrevem os componentes utilizados e suas interconexões, e informações geométricas, que dão a localização exata dos componentes e rotas sobre o circuito impresso ou uma superfície de silício. De modo geral, o processo de projeto exige abstrações de duas naturezas: trabalhar com diferentes níveis de projeto, visto que um nível de abstração mais elevado permite que o projeto manipule um número menor de componentes do que aquele realmente existente no sistema físico, e realizar o projeto em termos do seu comportamento, que pode ser descrito através de um algoritmo, por exemplo. Este segundo tipo de abstração acrescenta informações de natureza comportamental ao processo de projeto. Em suma, sobre o eixo estrutural temos representados um diferente conjunto de primitivas a partir das quais um sistema digital é composto; sobre o eixo comportamental, identificamos diferentes formalismos para a representação da função de sistemas digitais (considera-se aqui função como sinônimo de comportamento); sobre o eixo geométrico, os níveis de projeto dependem muito da tecnologia a ser adotada [ABRA] [GAJS] [MICH].

Ao longo desse processo de projeto, tanto descrições intermediárias como a descrição final do sistema necessitam ser validadas em relação a uma especificação original do sistema, que dá não só a função esperada deste como também certos parâmetros a serem obedecidos pelo produto final (velocidade, consumo, custo, etc). Existem dois métodos fundamentais que permitem ao projetista validar a descrição de um sistema: simulação e verificação. A simulação corresponde à solução numérica do problema de validação, na qual um modelo do sistema é exercitado por um conjunto de casos de teste com valores definidos para os sinais de entrada do sistema [VERC]. A verificação corresponde, por sua vez, à solução não-numérica do problema de validação, existindo diferentes métodos de verificação [DEMI].

No caso da simulação, um conjunto de vetores de teste é aplicado a um modelo do sistema, e os resultados obtidos são comparados com o resultado esperado. Uma poderosa ferramenta para a programação de arquiteturas computacionais assíncronas baseadas em memória é a TN (Transformada Numérica), ferramenta matemática que opera no campo numérico e que é usada para descrever funções e expressões da álgebra *booleana* e do cálculo proposicional, com o qual se pode resolver e minimizar circuitos lógicos. O funcionamento deste tipo de arquitetura assíncrona faz com que seja necessário gerar um mapa da memória com a microprogramação (*firmware*), onde a TN e suas propriedades auxiliam na busca de todas as alternativas possíveis, demonstrando que, intrinsecamente, o projeto desenrola-se através de um processo de simulação que deve necessariamente cumprir todos os casos para a solução do problema proposto.

Ao contrário da simulação, a verificação procura garantir formalmente a correção do resultado do projeto, seguindo uma de duas abordagens básicas. Na primeira abordagem se busca verificar a correção da descrição de um sistema em relação a uma especificação a ser seguida e na segunda abordagem, procura-se verificar a equivalência entre duas descrições, possivelmente em níveis distintos de abstração, de um mesmo sistema [SHIM]. Ambas sistemáticas fortalecem implementações realizadas com lógica *fuzzy*, pois estas introduzem o conhecimento humano, tomando suas informações do senso comum e da experiência dos operadores, o que, se corretamente tomadas na origem, garantem a verificação formal.

Flexibilidade de projeto

Ao computador associa-se sua ação fundamental, o ato de computar, no qual o cálculo é, por excelência, o processamento da informação. Para que o processamento ocorra adequadamente, deve haver um conjunto de regras que especifique a seqüência de operações a serem executadas para solucionar uma determinada classe de problemas, em um número finito de passos. Este é o conceito de algoritmo. A quantidade de informação algorítmica, ou sua complexidade, é expressa pelo número de bits do menor programa capaz de produzir uma dada mensagem [SIEG]. A complexidade de um algoritmo, incluídos os seus dados de entrada, está relacionada a seu desempenho com respeito à utilização de recursos computacionais, geralmente tempo de máquina e espaço de memória. A notação normalmente utilizada para denotar complexidade é dada por *"big Oh"*, ou seja, **O[f(n)]**, denotando o comportamento assintótico, onde n é denominado o tamanho do problema. Algoritmos são computacionalmente práticos se seu tempo de execução é de ordem polinomial, ou seja, $O(n^c)$, onde n é o tamanho do problema e c uma constante. Algoritmos exponenciais, $O(c^n)$, são computacionalmente

práticos somente para n pequeno. Assim, a classe de problemas denominada P (polinomial) é composta pelos problemas cuja melhor solução demanda algoritmos com complexidade de ordem (até) polinomial. Os problemas cujas melhores soluções até então conhecidas demanda algoritmos com complexidade exponencial constituem a classe de problemas NP (não deterministicamente polinomial).

Uma representação do conhecimento, conforme [BARR], é uma combinação de estruturas de dados e procedimentos de interpretação que, se usados de maneira correta dentro de um programa, levarão a um comportamento que simule o conhecimento dos seres humanos. A adequação heurística de uma representação de conhecimento é determinada pela estrutura da representação e pela eficiência do processo de raciocínio a ela associado. O raciocínio formal é utilizado em geral nas representações baseadas em lógica e consiste na transformação sintática de elementos de conhecimento através da aplicação de regras de inferência. Uma regra de inferência é uma regra sintática que, quando aplicada repetidamente a uma ou mais fórmulas verdadeiras, gera apenas novas fórmulas verdadeiras. Uma importante utilização do raciocínio procedimental se dá quando existe um algoritmo para determinar a solução de um problema relevante ao raciocínio corrente.

O método heurístico é largamente aplicado na representação e modelagem de sistemas. Consiste em se realizar uma tarefa de acordo com a experiência prévia, com regras práticas e estratégias freqüentemente usadas. Uma regra heurística pode ser uma implicação lógica da forma *Se* <*condição*> *Então* <*conseqüência*>. Algoritmos baseados em regras constituem alternativa poderosa de solução de problemas, caso típico do enfoque *fuzzy*, onde uma função real de entrada-saída é aproximada por seções que cobrem regiões da função.

A profundidade de um circuito (*hardware*) é definida como a latência entre as entradas e as saídas do mesmo, controladas pelo número de portas lógicas no caminho mais longo de qualquer entrada para qualquer saída, ou ainda, o número de níveis em uma cadeia causal ligando um objeto a sua origem [STAL]. Em geral, um circuito com uma profundidade pequena opera mais rapidamente do que um circuito com uma grande profundidade [MURD] [PATT]. Existem várias maneiras para se reduzir a profundidade do circuito, mas que normalmente implica aumentar a complexidade de algum outro parâmetro. A redução da complexidade do circuito como um todo pode gerar o aumento da profundidade do circuito. Na solução proposta, arquitetura computacional assíncrona com memória e implementada com lógica *fuzzy*, o projetista do circuito pode optar pela escolha entre complexidade e profundidade, através de algoritmos que maximizem ou minimizem este ou aquele parâmetro.

Em geral, a decomposição de portas lógicas pode ser realizada através de árvores balanceadas ou árvores desbalanceadas. As árvores balanceadas normalmente minimizam a profundidade do circuito e, embora seja possível usar árvores desbalanceadas para produzir circuitos funcionalmente equivalentes, estas últimas maximizam o parâmetro profundidade. No entanto, pode-se preferir uma árvore desbalanceada, pela sua característica de gerar um diâmetro de seção transversal mínimo a cada estágio, o que torna simples dividir a árvore em partes que podem ser espalhadas por um grande número de circuitos separados, típica situação prática encontrada no encapsulamento de *chips*.

Por fim, o uso de transformações nas expressões através de métodos numéricos para resolução de problemas lógicos, caso da TN, constitui ferramenta poderosa no auxílio da opção de projeto desejada. Neste caso, existem duas formas para a resolução de expressões algébricas: ou a expressão é resolvida por meio de deduções algébricas, as quais podem tornar-se bastante complicadas dependendo do grau de complexidade do problema a ser resolvido, ou então as expressões podem ser resolvidas através de meios numéricos, o que pode facilitar consideravelmente a obtenção do resultado.

Projeto visando facilitar o teste

Teste é um dos métodos de validação de um sistema digital, pois através dele é realizado um confronto entre a especificação ou descrição do sistema e a implementação obtida. Esse confronto é executado, de forma geral, pela aplicação de excitações ao circuito e análise das respostas obtidas ou comparação com valores esperados. A falha pode ser conceituada como uma diferença funcional em relação às especificações de projeto. As técnicas de validação de projeto garantiriam a correção, se fossem consideradas condições ideais de fabricação, o que, entretanto, não corresponde a realidade, pois são introduzidas falhas [NARD].

A facilidade ao teste é uma medida qualitativa da capacidade de determinação da existência (detecção) de falhas em um circuito, onde as condições para que os blocos internos cumpram estas metas são dependentes da capacidade de acesso a esses blocos. De um modo geral, os métodos de projeto visando facilitar o teste melhoram a capacidade de se observar, nas saídas, o comportamento de nodos internos do circuito (observabilidade), e melhoram a capacidade de se levar os sinais de entrada do circuito a nodos internos do mesmo (controlabilidade). Em se tratando de circuitos digitais, uma técnica muito difundida é a do caminho de varredura (*scan patch*), que faz uso da varredura de sinais para melhorar a observabilidade e a controlabilidade de nodos internos de circuitos seqüenciais.

Um circuito é testável se um conjunto de padrões de teste pode ser gerado, avaliado e aplicado de tal forma que sejam satisfeitos níveis pré-definidos de desempenho, assinalados em termos de detecção de falhas, localização de falhas e critérios de aplicação de testes dentro de um orçamento de custos e tempo pré-determinado. Quando um circuito seqüencial é testado, a primeira ação executada é trazê-lo a um estado conhecido pela aplicação de uma seqüência de inicialização. Neste caso, o parâmetro predictibilidade, que representa a capacidade de uma rede seqüencial de estabelecer um estado desejado, iniciando de um outro conhecido, tem grande importância. Ora, como a arquitetura assíncrona parte de um projeto onde são traçadas todas as alternativas possíveis, a partir da geração do *firmware*, com o que alia-se a implementação com lógica *fuzzy*, que adota soluções a partir de informações de experiências comprovadas, prospecta-se uma maximização deste parâmetro.

Incremento da modularidade

Os tempos de projeto de sistemas digitais tem-se tornado cada vez mais curtos, pela necessidade de lançamento de novos produtos no mercado. Da mesma forma, o avanço tecnológico vem permitindo circuitos com mais e mais componentes. Prevendo

o gargalo do tempo de projeto, desde a década de 80, de modo mais sistemático, vários trabalhos foram efetuados para aumentar o nível de abstração do projeto, de maneira a aumentar também a produtividade dos projetistas. As linguagens formais para descrição de *hardware* com vistas à sistematização de um processo de projeto, constituem um esforço de padronização que visa normalizar a descrição da circuitaria, através de uma linguagem única que possibilite a fácil migração entre diferentes plataformas tecnológicas [IEEE].

Originalmente, a normalização da linguagem para descrição de *hardware* era voltada para a descrição e simulação de sistemas eletrônicos, e não só para a síntese, o que lhe potencializa maior riqueza que o subconjunto normalmente utilizado e que é fornecido pelos diferentes fabricantes de CAD (*Computer Aided Design*). Assim, confere-se grande capacidade de abstração ao projetista, possibilitando a elaboração de projetos mais complexos, e portabilidade, tornando o projeto praticamente independente de tecnologias de fabricação, podendo beneficiar-se da última fronteira ou migrar para outras técnicas de implementação ou, ainda, reconfigurar a plataforma, reutilizando o mesmo *hardware*. No entanto, esse tipo de projeto apresenta problemas, seja pela sua repetição ou criados pela aplicação de uma linguagem para descrição de *hardware*. A limitação da síntese leva a que muitos problemas devam ainda ser fracionados à mão, não atende restrições sérias, como por exemplo, circuitos voltados ao baixo consumo [RABA] e dificulta a síntese de sistemas de *hardware* e *software* em conjunto [HARR].

Embora facilite a síntese por um lado, de outra forma, a utilização de uma linguagem de descrição de *hardware* também pode tolher a possibilidade de obter vantagens pela adoção de ferramentas numéricas de minimização de sistemas digitais, pois aquelas apresentam pré-definições na geração das estruturas lógicas associadas a cada comando, ditado pelo fabricante da linguagem [CARD]. Aí reside uma possibilidade interessante associada à estrutura computacional proposta, dada sua flexibilidade de programação e minimização oferecida pela aplicação da Transformada Numérica, quanto pela facilidade da aplicação de resolução de problemas por regras de especialistas.

A reutilização da circuitaria de um sistema digital tem-se constituído aspecto de especial interesse no projeto, visto a elevada obsolescência programada de setores da indústria de equipamentos eletrônicos. Os países campeões em números de computadores descartados estudam a questão de perto. Estados Unidos, Japão e Europa já se preocupam com o assunto há mais tempo, inclusive discutindo a criação de leis que obriguem os fabricantes a recolher toda máquina velha fora de uso, que é uma ação indispensável em curto prazo, muito embora paliativa. A questão essencial nesta definição é o conflito existente na arquitetura computacional baseada em *clock*, que, para buscar o aumento do processamento, transporta a geração anterior à obsolescência e aumenta o, assim denominado, lixo digital.

Entende-se, pois, que uma arquitetura computacional assíncrona com ferramenta de minimização lógica aberta, tendo como aliada a aplicação programável de regras de inferência *fuzzy*, potencializa a reutilização da circuitaria, pela característica do dispositivo, barra uma seqüência de obsolescência programada e pode gerar, ainda, uma alternativa concreta de sistema digital capaz de enfrentar os efeitos nocivos do descarte premeditado.

Referências Bibliográficas

[ABRA] ABRAMOVICH, M.; BREUER, M.;FRIEDMAN, A. **Digital Systems Testing and Testable Design.** New Jersey: IEEE Press, 1990.

[ALTR] ALTROCK, C. Von. Fuzzy Logic in Automotive Engineering. **Circuit Cellar,** Vernon, n.88, p.12-27, Nov. 1997.

[BARR] BARR, A.; FEIGENBAUM, E. A. (ed.). **The Handbook of Artificial Inteligence – vol. I-II.** Los Altos: William Kaufmann, 1981.

[BRET] BRETON, P. **Histoire de L'informatique.** Paris: La Decouvert, 1987.

[BRZO] BRZOZOWSKI, J.A.; SEGER, C.J.H. **Asynchronous Circuits.** New York: Springer-Verlag, 1995.

[CARD] CARDOSO, J.M.P.; NETO, H.C. Compilation for FPGA-based reconfigurable hardware. **Design & Test of Computers,** IEEE, vol. 20, no. 2, Mar-Apr 2003, p.65-75.

[COXE] COX, E.D. **Fuzzy Logic for Business and Industry.** Rockland: Charles River Media, 1995.

[CUES] CUESTA, L.; PADILLA, A. G.; REMIRO, F. **Electrónica Digital.** Lisboa: Mc Graw Hill, 1994.

[DAVI] DAVIS, Al; NOWICK, S. M.. **An Introduction to Asynchronous Circuit Design.** Salt Lake City, University of Utah; New York, Columbia University, September 19th, 1997.

[DEHO] DeHON, A.; WAWRZINEK, J. Reconfigurable Computing: What, Why, and Implications for Design Automation. **ACM/IEEE Design Automation Conference,** New Orleand, June 1999, Proceedings… IEEE/Computer Society Press, 1999, p.610-615.

[DEMI] DE MICHELI, G. Hardware Synthesis from C/C++ Models. **Design, Automation and Test in Europe,** Proceedings… Munich, Germany, 9-12 Mar, 1999, IEEE Computer Society Press, Los Alamitos, CA, 1999, p. 382-383.

[EBER] EBERGEN, J. **Circuits Without Clocks: What Makes Them Tick ?** Mountain View, Sun Microsystems Laboratories, May 13th, 1998.

[GAJS] GAJSKI, D. et alii. **High-Level Synthesis: Introduction to Chip and System Design.** New York: Kluwer Academic Publishers, 1992.

[HARR] HARRIS, I.G. Fault models and test generation for hardware-software covalidation. **Design & Test of Computers,** IEEE, vol. 20, no. 4, July-Aug 2003, p.40-47.

[HILL] HILLIS, D. **The Pattern on the Stone.** London: Orion Books, 1998.

[IEEE] The Institute of Electrical and Electronics Engineers, Inc.. **IEEE Standard VHDL, Language Reference Manual, IEEE Std. 1076-1987.** New Jersey: IEEE, 1988.

[KOSK] KOSKO, B. **Neural Networks and Fuzzy Systems.** Englewood Cliffs: Prentice Hall, 1992.

[MANO] MANO, M. M. **Digital Design.** 3th ed. Upper Saddle River: Prentice Hall, 2002.

[MART] MARTINS, W. W. **Esção (n-m-p): Um Computador não-Von Neumann.** São Paulo: Cartgraf, 1985.

[MEND] MENDEL. J.M. Fuzzy Logic Systems for Engineering: A Tutorial. **Proceedings of the IEEE,** v.83, no.3, Mar.1995.

[MICH] MICHEL, P.; LAUTHER, U.; DUZY, P. **The Synthesis Approach to Digital System Design.** New York: Kluwer Academic Publishers, 1992.

[MYER] MYERS, C.J. **Asynchronous Circuit Design.** New York: John Wiley & Sons, 2001.

[MURD] MURDOCCA, M.J.; HEURING, V.P. **Principles of Computer Architecture.** Upper Saddle River: Prentice Hall, 2000.

[NARD] NARDI, A.; SANGIOVANNI-VINCENTELLI, A.L. Logic Synthesis for Manufacturability. **Design & Test of Computers,** IEEE, vol. 21, no. 3, May-June 2004, p.192-199.

[NOWI] VAN BERKEL, C.H.; JOSEPHS, M.B.; NOWICK, S.M. Scanning the Technology: Applications of Asynchronous Circuits. **Proceedings of the IEEE,** v.87, n.2, Feb. 1999, p.223-233.

[PATT] PATTERSON, D.A.; HENNESSY, J.L. **Computer Organization and Design – The hardware/software interface.** 2nd ed. San Francisco: Morgan Kaufmann, 1998.

[RABA] RABAEY, J.; WAN, M. An Energy-Conscious Exploration Methodology for Reconfigurable DSPs. **Design, Automation and Test in Europe,** 1998, Proceedings… IEEE Computer Society Press, Los Alamitos, CA, 1998, p. 341-342.

[ROTH] ROTH Jr., C. H. **Fundamentals of Logic Design.** 4th ed. Boston: PWS Publishing Co., 1995.

[SHIM] SHIMIZU, K.; DILL, D.L. Using formal specifications for functional validation of hardware designs. **Design & Test of Computers,** IEEE, vol. 19, no. 4, Jul-Aug 2002, p.96-106.

[SIEG] SIEGFRIED, T. **The Bit and the Pendulum.** New York: John Wiley & Sons, 2000.

[STAL] STALLINGS, W. **Computer Organization and Architecture.** Upper Saddle River: Prentice Hall, 2000.

[SUTH] SUTHERLAND, I. E.; LEXAU, J. K. Designing Fast Asynchronous Circuits. **ASYNC 2001 Seventh International Symposium on Advanced Research in Asynchronous Circuits and Systems,** Salt Lake City, Proceedings, March 11-14, 2001.

[TIND] TINDER, R.F. **Engineering Digital Design.** 2th ed. San Diego: Academic Press, 2000.

[UNGE] UNGER, S.H. **Asynchronous Sequential Switching Circuits.** New York: Wiley-Interscience, 1969.

[VERC] VERCAUTEREN, S.; LIN, B.; De MAN, H. Constructing Application-Specific Heterogeneous Embedded Architectures from Custom HW/SW Applications. **ACM/IEEE Design Automation Conference,** Las Vegas, Proceedings… ACM Press, 1996, p. 521-526.

[ZADE] ZADEH, L.A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. **IEEE Transactions on Systems, Man and Cybernetics,** New York, v.3, n.1, Jan. 1973, p.28-44.