

Uma metodologia de reconfigurabilidade utilizando a Plataforma de Prototipação Chameleon

Manoel Eusebio de Lima, Remy Eskinazi Sant'Anna,
Abel Guilhermino da Silva Filho e Paulo Romero Martins Maciel

Centro de Informática da Universidade Federal de Pernambuco
GRECO – Grupo de Engenharia da Computação
Caixa Postal 7851 - 50740-540 - Recife - PE - Brasil
{ mel, res, agsf, prmm@cin.ufpe.br }

Resumo. *A computação reconfigurável, não obstante ser uma área de pesquisa recente, vem apresentando um progresso bastante promissor. O desenvolvimento de plataformas que combinam hardware reconfigurável com elementos de controle programáveis, tais como DSP's ou microcontroladores prometem grande aplicabilidade em diversas áreas no futuro. Neste artigo, objetivamos descrever uma metodologia codesign para a plataforma Chameleon utilizando estas características, assim como o uso de Redes de Petri na estimativa de métricas no cálculo dos módulos de reconfiguração. Uma descrição da plataforma é apresentada, bem como a metodologia de reconfiguração. Um pequeno exemplo é discutido com resultados.*

Palavras Chave. *Reconfiguração Dinâmica, Redes de Petri, FPGA.*

1 Introdução

Devido ao crescente melhoramento dos dispositivos reconfiguráveis, notadamente nas características de integração do número de portas lógicas, velocidade e consumo de potência, a complexidade dos sistemas digitais que podem ser implementados também pode ser aumentada proporcionalmente. Isto justifica, por outro lado, o crescente interesse no estudo de novas metodologias de projeto visando redução de custo, melhor performance e menor *time-to-market*. Hardware/software codesign é uma destas tecnologias que tem somado à disponibilidade de ambientes de projeto suportando este tipo de abordagem desde a especificação até a prototipação final do sistema.

De uma maneira geral, implementações em hardware, tem como vantagem explorar concorrência e o aumento da velocidade de processamento dos dados. Infelizmente, aplicações implementadas de tal maneira derivam alto custo de prototipação e pouca flexibilidade para o projeto e mudanças funcionais. Implementações em software por sua vez, apresentam vantagens tais como flexibilidade e baixo custo de implementação de funções complexas. No entanto, ele apresenta limitações tais como dificuldades de explorar paralelismo e aplicações em alta velocidade.

Uma abordagem *codesign* por outro lado, oferece alternativas mais flexíveis para a escolha das soluções de problemas encontrados em ambos tipos de implementações, permitindo a utilização de critérios de escolha para implementação dos componentes de hardware e software baseados em restrições (função de custo), tais como: área de silício, velocidade e aplicação. Em metodologias como esta, os componentes de software podem ser representados por microcontroladores ou microprocessadores e componentes de hardware por ASICs ou FPGAs [Hauck1], [Hauck2].

Neste artigo apresentamos a descrição de uma metodologia para prototipação de sistemas digitais baseada na plataforma Chameleon [Lima2] e no ambiente projetos PISH (Projeto Integrado de Software/Hardware) [Barros1]. Esta versão da plataforma Chameleon prevê reconfiguração dinâmica baseada em componentes FPGAs XC400xx da Xilinx. Embora estes componentes exijam reconfiguração completa em cada ciclo de programação, aplicações dinâmicas podem ser satisfatoriamente implementadas com menos hardware, em tempo previsto. Os componentes de software representam o controle de execução dos componentes de hardware, garantindo a passagem de parâmetros e de comunicação entre processos.

2 Visão Geral da Chameleon e PISH

A plataforma Chameleon, que é mostrada na figura 1(a), é composta basicamente de um componente de hardware (FPGA), um componente de software (microcontrolador) e dispositivos de memória (RAM e EPROM). Os componentes de hardware e software compartilham a memória e um canal de comunicação. Esta plataforma baseia-se na metodologia do PISH, ilustrada através da Figura 1 (b). O projeto PISH não é apenas uma metodologia, mas um ambiente para desenvolvimento de sistemas digitais complexos.

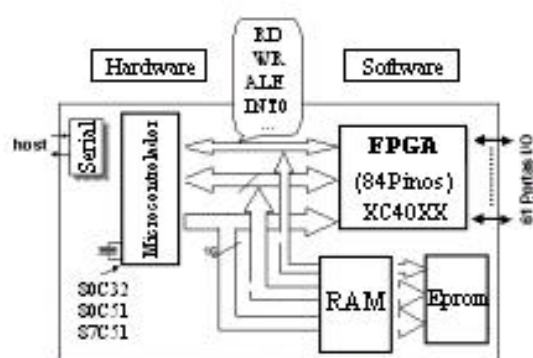


Fig. 1(a) – Plataforma Chameleon

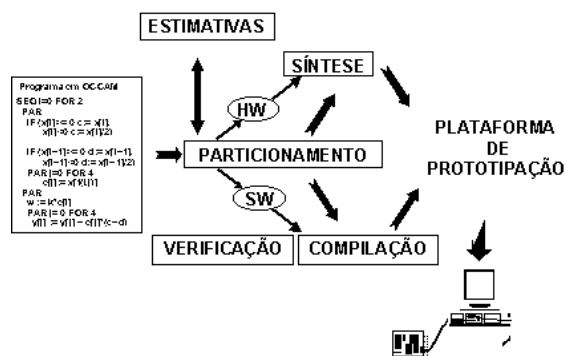


Fig. 1(b) - Metodologia PISH

Esta metodologia visa proporcionar um ambiente completo de projeto a partir da especificação do sistema até sua prototipação. Seu mecanismo de especificação atual usa OCCAM [Barros2] como linguagem, e compreende todas as fases do processo do projeto como mostrado na Figura 1(b).

O particionamento hw/sw é realizado a partir de um mecanismo formal que garante a preservação da semântica da especificação inicial do sistema. A fase de análise usa uma representação em Rede de Petri [Maciel,3] da descrição particionada. Este modelo de Rede de Petri permite análises qualitativas e métricas de computação.

Depois da etapa de particionamento, os processos a serem implementados em hardware são sintetizados, enquanto os processos de software são compilados. Um gerador automático de interfaces [Araújo1,2], gera a interface hw/sw apropriada para ambos os módulos para garantir a correta comunicação depois do particionamento. O gerador cria um arquivo em código C para a partição de software e um arquivo em código VHDL para a partição de hardware. Com isso, os módulos a serem prototipados e suas interfaces podem ser implementados na plataforma de prototipação. Métricas de tempo e área de hardware também são repassadas para a plataforma através de estimadores. Na Plataforma de prototipação Chameleon, os processos de software são executados em um microcontrolador 80C32 de baixo custo, compatível com o 8051 padrão.

Os processos de hardware são implementados em uma FPGA da série XC40XX da Xilinx [Xilinx].

A plataforma apresenta possui dois bancos de memória, sendo 64kbyte de memória de programa e 64kbytes de memória de dados, que provém espaço para armazenamento dos programas de configuração do FPGA e microcontrolador, bem como variáveis e processos de informação para a reconfiguração dinâmica do sistema.

A comunicação entre os componentes de software e hardware é realizada através de um barramento físico de comunicação bi-direcional e sinais de controle gerenciados pelo processador principal. Um protocolo de comunicação garante o correto *handshake* entre processos de software e hardware.

3 Descrição da metodologia aplicada

Uma visão geral da metodologia hardware/software para monitoramento e controle de dispositivos aplicativos será descrita nesta seção.

A metodologia proposta requer mecanismos que gerenciam a plataforma Chameleon, uma Interface Gráfica, *cores* para implementações dos componentes de hardware e software além de dispositivos aplicativos e uma base de dados para armazenamento, como ilustrados na Figura 2.

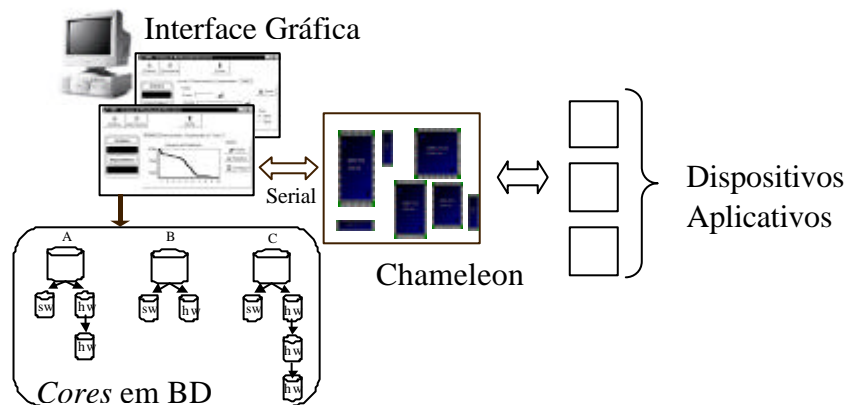


Figura 2 – Metodologia de Reconfigurabilidade

Os dispositivos aplicativos (fig.2) são dispositivos direcionados para uma determinada aplicação e podem ser representados por componentes *off-the-shelf* tais como conversores AD e DA, displays, ports de comunicação e sensores e ASICs.

Um sistema supervisor instalado em PC permitirá realizar a amostragem dos dados para o usuário, em tempo de execução, implementar o *download* dos respectivos *cores* pré-sintetizados da aplicação, além da comunicação serial entre o PC e a plataforma.

Em sistemas tradicionais, poderiam ser necessários múltiplos ASICs para poder tratar cada dispositivo pertencente ao sistema do usuário. Isto acarretaria um aumento de área e custo significativo no desenvolvimento do sistema, além de tornar o mesmo cada vez mais complexo. O uso de dispositivos reconfiguráveis por outro lado tem sido uma solução interessante para estes tipos de aplicações permitindo que diferentes cores equivalentes em funcionalidade a ASICs possam ser implementados facilmente sem modificação do componente de hardware.

3.1 Estratégia de projeto

Com o incremento da complexidade e tamanho de sistema digitas, mais ferramentas de CAD são necessárias para aumentar o processo de validação e conseqüentemente a

prototipação no contexto hardware/software *codesign*. Componentes de hardware continuarão crescendo, assim como o problema de ajustá-los em um único FPGA. Soluções como plataforma Multi-FPGAs [Silva, Lima1], são soluções possíveis, desde que é possível, a partir de uma dada especificação, quebrar o sistema em vários pequenos processos ajustáveis aos FPGAs da plataforma. Esta abordagem exige o uso de uma plataforma com vários componentes de hardware, o que pode elevar o custo da solução.

Este trabalho no entanto, propõe uma metodologia para implementação de pequenos projetos baseadas na reconfigurabilidade dinâmica dos componentes FPGAs. Como mostrado na Figura 3, o fluxo de projeto para o mapeamento dos componentes de hardware e software prevê o uso de apenas uma FPGA para a execução de todos os processos de hardware.

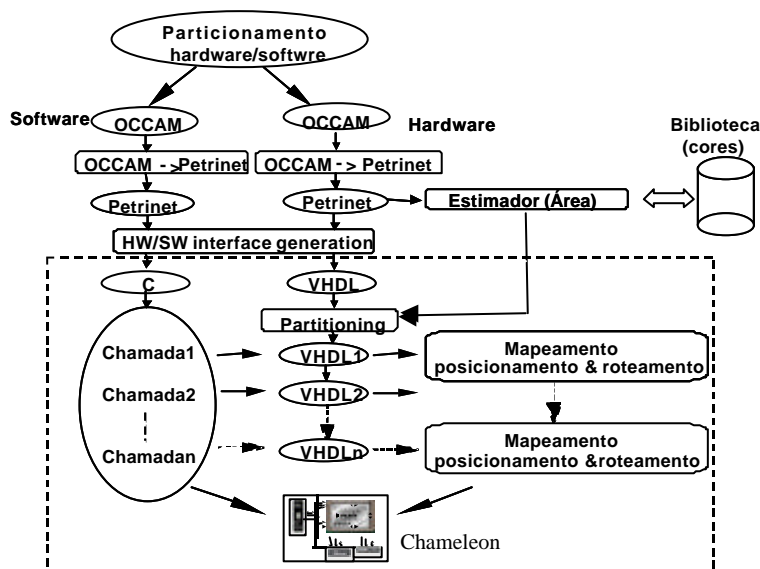


Figura 3 - Mapeamento de componentes

Neste fluxo de projeto, após o particionamento hardware/software, seus respectivos códigos em Occam são transcritos para Redes de Petri. Redes de Petri é um nome genérico para uma família de modelos formais que podem ser representados por diferentes níveis de abstração. A maior atração das Redes de Petri é a forma com que os aspectos básicos dos sistemas concorrentes, distribuídos, assíncronos e não-determinísticos são capturados, tanto do ponto de vista conceitual quanto do ponto de vista matemático. Estas redes demonstram ser um modelo conveniente para expressar as noções fundamentais dos sistemas concorrentes, o que contribui fortemente para o desenvolvimento de uma rica teoria de modelagem e análise de sistemas [Murata, Reisig]. Os fundamentos teóricos em Redes de Petri permitem diversas abordagens para a análise de propriedades qualitativas de sistemas. Os métodos de análise qualitativa utilizando este tipo de formalismo são basicamente divididos em três grupos: métodos estruturais, os baseados na geração do espaço de estados e as reduções [Murata]. Em muitas aplicações, os aspectos temporais têm importância fundamental, o que nos leva aos modelos temporais. Dentre estes, destacamos as classes de modelos determinísticos e de modelos estocásticos que possibilitam a análise desempenho dos sistemas.

Neste trabalho em particular, utilizaremos as Redes de Petri Temporizadas onde os tempos (duração) são associados às transição. A seguir uma definição formal do modelo é apresentada.

Redes de Petri Temporizadas – Seja a tupla $N_t=(N,D,C)$ uma rede de Petri temporizada, onde $N=(P,T,I,O,M_0)$ é uma rede de Petri, $D:T \rightarrow R^+ \cup 0$ é uma função que associa uma

→

duração a cada transição da rede. $C:T$ $(0 \leq R \leq 1)$ é uma função que associa a cada transição uma probabilidade de escolha, onde $\sum_{t \in T_c} c(t) = 1$. T_c é um conjunto de transições em conflito.

Entre as informações estruturais de mais importância no modelo estão os invariantes de lugar [Murata]. Os invariantes de lugar são conjunto de lugares nos quais o somatório de marcas permanece constante. Para subclasses de redes como os grafos marcados [Murata], é possível obter a captura de relações de precedência causal entre as transições que são saídas dos lugares suportes dos invariantes de lugar, o que possibilita a análise do compartilhamento de unidades funcionais pelos processos [Maciel 4,5]. Estimativas mais precisas da área de hardware, que fazem uso de estimativas iniciais obtidas através dos invariantes, são calculadas a partir da construção do espaço de estados da rede temporizada (modelo intermediário) estendida pela alocação das operações às unidades funcionais [Maciel 1].

Componentes sequenciais de código (processos) também podem ser obtidos pela análise dos invariantes de lugar, dado que todas as transições que são saídas dos lugares suportes destes invariantes têm uma relação de precedência. Como resultado, estes processos sequenciais são convertidos em VHDL, na forma de processos ou VHDL estrutural. A granularidade da análise da rede especifica o número de códigos FPGAs a serem configurados. Esta metodologia permite portanto a captura da sequência e dos parâmetros que devem ser transferidos entre processos de hardware durante a execução da aplicação. Após o cálculo das estimativas e particionamento dos processos de hardware, códigos C e VHDL são gerados e mapeados em suas respectivas plataformas. O código C resultante, corresponde a um processo de controle que gerencia a execução da comunicação entre processos de hardware.

Essa nova abordagem exige que todas as reconfigurações do FPGA e toda a passagem de parâmetros entre processos de hardware seja feito pelo componente de software. Este mecanismo de chamada dos componentes de hardware pelo componente de software é feito pelo microcontrolador, através de rotinas que têm acesso a rotina de reconfiguração do FPGA e dos componentes de hardware previamente sintetizados (Figura 4). Estas rotinas em software sincronizam todo o chaveamento entre processos através de *flags* e *buffers* necessários para a execução correta da aplicação.

Assim, devido a critérios de tempo e complexidade de execução de algoritmos, várias estratégias de configuração e reconfiguração da FPGA, associadas ao componente de software podem ser avaliadas, mesmo antes de sua implementação, visando usufruir melhor os recursos existentes e conseguir assim uma melhor performance. Cada nova aplicação poderá ou não utilizar recursos já existentes na base de dados, ou seja, utilizar *cores* já disponíveis na biblioteca, ou sintetizar novos códigos não existentes.

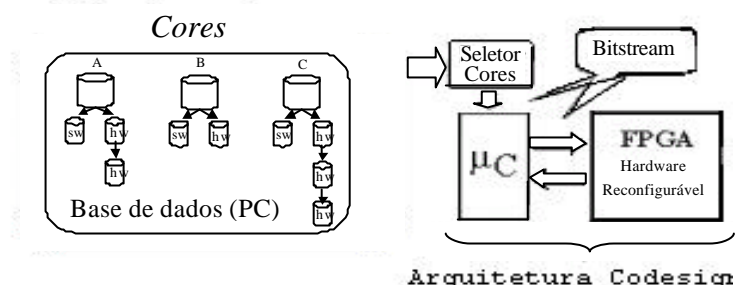


Figura 4 – Estratégia de Projeto

O sistema prevê ainda a confecção de uma biblioteca de *cores* para cada tipo de aplicação a ser definida pelo projetista. Assim, cada nova aplicação, uma vez devidamente

sintetizada pode ser guardada e reutilizada para posterior utilização, sem a necessidade de uma nova avaliação e síntese de seus componentes.

Cada nova especificação de uma função (*core*), por sua vez, deve ser devidamente particionada, mapeada, compilada, simulada e finalmente posicionada e roteada adequadamente para as tecnologias FPGA e microcontrolador do sistema como explicado anteriormente.

3.2 Configuração do sistema

Os arquivos de configuração dos componentes de hardware e software resultantes do processo de particionamento são sintetizados pelas ferramentas de síntese e simulação da Xilinx (*Foundation Series* e *XACT*) e compilados no ambiente da Keil [KEIL] respectivamente.

O componente de hardware é sempre configurado e reconfigurado pelo microcontrolador, através do modo de configuração *Asynchronous Peripheral* [XILINX], isto ocorre devido a disponibilidade de recursos, como barramento paralelo e memória já existentes na plataforma.

Diferentemente de uma abordagem baseada em EPROM, a metodologia sugerida neste projeto visa um armazenamento dos componentes de software e hardware em RAM, permitindo assim uma maior flexibilidade e um número maior de aplicações sem depender do limite da EPROM interna da plataforma. A Figura 5 ilustra os como os códigos dos componentes de hardware e software seriam armazenados em RAM e o circuito adicional necessário para permitir a execução dos processos a partir dela.

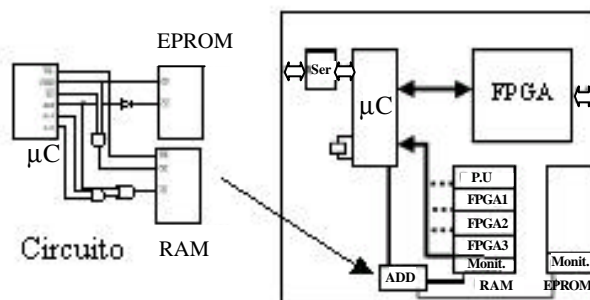


Figura 5 – Armazenamento de cores em RAM

A necessidade de se ter vários componentes de configuração da FPGA exige uma capacidade de memória que deve ser suficiente para armazenar o número de reconfigurações necessárias, caso seja solicitado pela aplicação. Como cada reconfiguração da FPGA possui tamanho fixo para determinada família, o espaço necessário para armazenamento do código de hardware é dado pelo número de possíveis reconfigurações vezes o tamanho do código da FPGA. Por outro lado, o tamanho do código de software depende da aplicação e não possui tamanho determinado.

Assim, pode-se dizer que o tamanho mínimo de memória RAM é dado por:

$$\text{Tam_RAM} = n * \text{Cod_conf_FPGA} + \text{Cod_software} + \text{Cod_Monitor}$$

Cod_conf_FPGA = tamanho do código de configuração da FPGA.

Cod_software = tamanho do código do componente de software

Cod_Monitor = tamanho do código do programa monitor (fixo)

N = número de configurações do FPGA

A transferência e armazenamento adequado de cada nova aplicação são gerenciados por um monitor residente, que tem a função de gravá-los nas posições pré-definidas da RAM e permitir sua execução a partir da RAM.

Qualquer aplicação pode ser interrompida, quando desejado pelo usuário, através do canal de comunicação serial. Esta interrupção é controlada pelo sistema supervisor no PC, que gerencia a implementação de todas as aplicações na plataforma.

Cada nova aplicação a ser prototipada deve ser adequadamente selecionada através do Sistema Supervisor. Informações sobre a nova aplicação selecionada são enviadas para o componente de software (microcontrolador), que por sua vez se encarrega de direcionar a execução do programa para o monitor em RAM. Este monitor possui rotinas que tratam do carregamento de dados de acordo com a área do sistema, códigos de software e hardware, bem como, da configuração do FPGA e chaveamento de contexto entre aplicações e o próprio monitor.

A seguir será apresentado um exemplo para validação desta metodologia proposta.

4 Exemplo: Máquina de Vender Refrigerante

Um pequeno exemplo, será usada como demonstração do procedimento. Esta máquina foi originalmente descrita em Occam e seu código particionado no PISH. As métricas após o particionamento foram estimadas, assim como, foram geradas as interfaces entre componentes de hardware/software. A especificação do problema em Redes de Petri é representado por 148 lugares e 141 translações. As métricas usadas neste exemplo são as áreas dos componentes de hardware. Estas áreas foram obtidas da tabela de equivalência da Xilinx [Xilinx], apresentada na tabela 1, a qual indica o número de portas equivalentes por função implementada nos componentes da família FPGA Xilinx XC40XX, utilizadas como componentes de hardware na plataforma de prototipação em estudo.

Function	Gate Count
2-Input NAND	1
2-to-1 Multiplexer	4
3-Input XOR	6
2-Bit Carry-Save Full Adder	9
D FlipFlop	6
D Flip-Flop with Set or Rreset	8
D Flip-Flop with Reset and Clock Enable	12

Tabela 1: Número de portas lógicas equivalentes

O código em C resultante da partição tem 235 linhas e o código VHDL 357 linhas. O código em C foi compilado para o microcontrolador 80C32. O resultado do particionamento do hardware exigiu a utilização de dois FPGAs X4005E. Nesta metodologia, os dois códigos (tabela 2) serão implementados na mesma FPGA, sob controle do componente de software.

	FPGA1	FPGA2
Gates Count	3715	3074
CLBs Count(%)	242(60%)	202(50%)

Tabela 2. Número de CLBs para a implementação em hardware

A tabela 3 abaixo mostra a estimativa de área durante o particionamento hardware/software obtido pela ferramenta de estimativa baseada em Redes de Petri, e sua implementação real no componente Xilinx's XC4005E.

	Métricas	Multiplexadores	Unidades Funcionais	Controle
	Registradores			
Estimado	240	224	4291	1026
Implementado	152	498	4291	1230

Tabela 3. Comparação entre o hardware estimado e o hardware implementado

5 Conclusões

Este trabalho propõe uma metodologia codesign para prototipação rápida de sistemas digitais de baixo custo em um ambiente que permite reconfiguração dinâmica dos componentes de hardware. Isto permite redução de custos de implementação desde que um único FPGA será usado como componente de hardware. O sistema prevê o uso de estimadores de área de componentes de hardware baseados em Redes de Petri, que analisam em estágio prévio a possibilidade de implementação ou não de seus componentes na plataforma de prototipação. Neste modelo os componentes de software são controladores de sequência de execução dos processos de hardware. Um exemplo preliminar desenvolvido manualmente foi desenvolvido e apresentado. Uma versão automática, prevendo geração automática do controle em software e dos processos de hardware está sendo desenvolvida.

6 Referências Bibliográficas

- [Araújo1 2000] Araújo, C. C., Silva, D. S. Silva, E. Barros, M. E De Lima, P. Maciel, "Co-Synthesis and Prototyping Supporting the Design of Reconfigurable Systems", CORE2000: Reconfigurable Computing – Experiences and Perspectives, São Paulo-Brazil, pp.54-67.
- [Araújo2 1999] Araújo, C.C., E. Barros, "A approach for Interface Generation in the PISH Codesign System ", Proceedings of SBCCI99, Brazil.
- [Lima1 2000] De Lima, M. E., D. S. Silva, D. G. Ramalho, A. V. Burgos, "Chameleon-I: A Rapid Prototyping Multi-FPGA Platform for PISH Codesign System", SBMicro2000 - XV International Conference on Microelectronics and Packaging", pp. 86-91.
- [Lima2 1998] De Lima, M., S. Cavalcante, C. Araújo, H. Saraiva, " Chameleon: A Prototyping Platform for Digital System Design " , XIII SBMicro – International Conference on Microelectronics and Packaging (ICMP'98), pp. 71-77.
- [Brown 1996] Brown S., J. Rose, "FPGA and CPLD Architectures: A Tutorial", *IEEE Design & Test of Computers*, pp. 42-57.
- [Hauck1 1998] – Hauck, S. – "The future of reconfigurable systems", Keynote Address, 5th Canadian conference on field programmable devices, Montreal.
- [Hauck2 1998] – Hauck, S. – "The holes of FPGAs in reprogrammable systems", proceedings of IEEE, vol. 86, No 4, pp 615-638.
- [Silva 2000] Silva, D. S., "Desenvolvimento de uma Plataforma de Prototipação Rápida Usando um Sistema Multi-FPGAs", MSc Thesis, Centro de Informática – UFPE, Brazil.
- [Maciel1 2000] Lima, M. E; Barros, E.; Maciel P. M.; Silva, D. S., Rosenstiel, W., "Resource Sharing Estimation by Petri Nets in PISH Co-design System", HPC'2000, pp.371-376.
- [Maciel3 2000] Maciel P. R., E. Barros, M. de Lima, D. S. da Silva, "Resource Sharing Estimation by Petri Nets in PISH Codesign System", HPC 2000 Special Track on Petri Nets and Performance Evaluation in HPC2000, Washington.
- [Maciel4 1999] Maciel P., E. Barros, W. Rosenstiel. A Petri Net Model for Hardware/Software Codesign, Design Automation for Embedded Systems Journal, Kluwer Academic Publishers, Vol-4, pp 243-310.
- [Maciel5 1999] Maciel, P., E. Barros, W. Rosenstiel, Estimating Functional Unit Number in PISH Codesign System by Using Petri Nets, IEEE 12^o Symposium on Integrated Circuits and System Design, Natal, Brazil.
- [Xilinx1 1989] Xilinx, "Count Capacity Metrics for FPGAs", XAPP059, V1.1.
- [Keil 1997] – KEIL Software inc. professional development kit manuals.
- [Barros1 1997] Barros, E. et al., "Hardware/Software Codesign in the PISH project", proceedings of the II Brazilian Workshop on Hardware/Software Codesign.
- [Barros2 1994] Barros, E.; and Sampaio, A. "Towards provably correct hardware/software partitioning using occam", pp.210-217.
- [Reisig 1982] Reisig, W. , Petri Nets: An Introduction, Springer-Verlag.
- [Murata 1989] Murata T., Petri Nets: Properties, Analysis and Applications, Proceeding of The IEEE.