PROJETO CONJUNTO HARDWARE/SOFTWARE

HARDWARE/SOFTWARE CODESIGN

Ricardo PANNAIN*
Frank Herman BEHRENS**

ABSTRACT

Hardware/software codesign is not a new approach. It has been used since the first microprocessors and microcomputers came out to the market. Advances in design, synthesis and verification of integrated circuits technology, and the increase of the necessity towards to developed integrated hardware/software systems, are driving hw/sw designers to a new metodology. In this paper we introduce some concepts about this approach and discuss some problems in the development of this systems.

KEY-WORDS: hardware/software codesign, ASIC's - application-specific integrated circuits, real-time embedded systems.

RESUMO

O projeto de sistemas que integram software e hardware não é uma abordagem nova, mas nos últimos anos ganhou mais força devido ao aparecimento de um número cada vez maior de aplicações que necessitam um sistema integrado hardware/software, à necessidade de diminuição de custos de projetos e de testes e ao avanço tecnológico em algumas áreas, tais como, síntese lógica e métodos formais de descrição, que facilitam o desenvolvimento destes sistemas. Neste artigo procuramos introduzir alguns conceitos sobre metodologias para este enfoque de projeto, assim como mostrar as dificuldades encontradas no desenvolvimento do mesmo.

PALAVRAS-CHAVE: integração hardware/software, circuitos integrados de aplicação específica, sistemas de tempo real integrados.

1. INTRODUÇÃO

Um sistema integrado HARDWARE/SOFTWARE é aquele onde se tem a co-existência de hardware e software, cada qual com uma função específica, e que se comunicam através de interfaces, bem definidas e de alto desempenho [1 - 7]. A maioria destes sistemas, consiste de um processador de propósito geral, memória e um circuito integrado de aplicação específica (ASIC's- *Application Specific Integrated Circuits*) interligados. Exemplos deste conceito podem ser vistos em [8 - 10].

A abstração para este modelo de sistema, mostrada na Figura 1, consiste na divisão em duas partes que se interagem. Estas partes são chamadas de domínios,

(*) Engenheiro Eletricista, mestre em Engenharia Elétrica e doutorando em Engenharia Elétrica pela UNICAMP. Atualmente é Coordenador e professor do Curso de Engenharia de Computação do I. I. da PUCCAMP e professor do Departamento de Ciência da Computação do IMECC/UNICAMP.
(**) Engenheiro Eletricista, mestre e doutor em Engenharia Elétrica. Atualmente é professor do Curso de Engenharia de Computação do I. I. da PUCCAMP.

portanto temos o domínio do software e o domínio do hardware. O domínio do software é formado pelas camadas de programas de aplicação, do sistema operacional, dos *drivers* de entrada e saída de dados e do barramento de entrada e saída de dados. O domínio do hardware é formado pelas camadas dos circuitos de aplicação específica, da interface entre este circuito e o barramento de entrada e saída e do barramento propriamente dito.

Ainda neste tipo de abstração, é possível dois tipos de interação: a interação vertical, intra-domínio, e a interação horizontal, inter-domínio. Na interação vertical uma camada interage com as suas camadas vizinhas, ou seja, no caso do domínio do software, os programas interagem com o sistema operacional, este com os *drivers*, que por sua vez interagem com o barramento. No caso do domínio do hardware, os circuitos interagem com a interface de barramento, que por sua vez interage com o barramento. Na interação horizontal, algumas camadas de um domínio interagem com determinadas camadas do outro, por exemplo, a camada de programas (domínio do software) interage com a de circuitos (domínio de hardware) através do envio e recebimento de mensagens; a camada de *drivers* (domínio do software) interage com a de interface (domínio do hardware) através de ações de escrita e leitura de registradores. O barramento é o meio físico utilizado para transações entre o sistema que executa o software e o sistema que comporta o circuito de aplicação específica. É importante destacar ainda que, a interação no nível mais baixo, mostra as transações da CPU, levando-se em conta como o hardware de aplicação decodifica, interpreta e reage a estas transações. No próximo nível de abstração, a decodificação de endereços e as interrupções ficam transparentes, permitindo considerar apenas transferências de dados entre os *drivers* e o hardware. Na interação no nível mais superior, detalhes de sistema operacional e drivers de dispositivos são transperentes.

Exemplos destes sistemas são encontrados em instrumentação médica, controle de processos, automatização de veículos e sistemas de comunicação e redes. Estes sistemas também são chamados de sistemas de tempo real integrados (*real-time embedded systems*), pois têm restrições de tempo em suas ações.

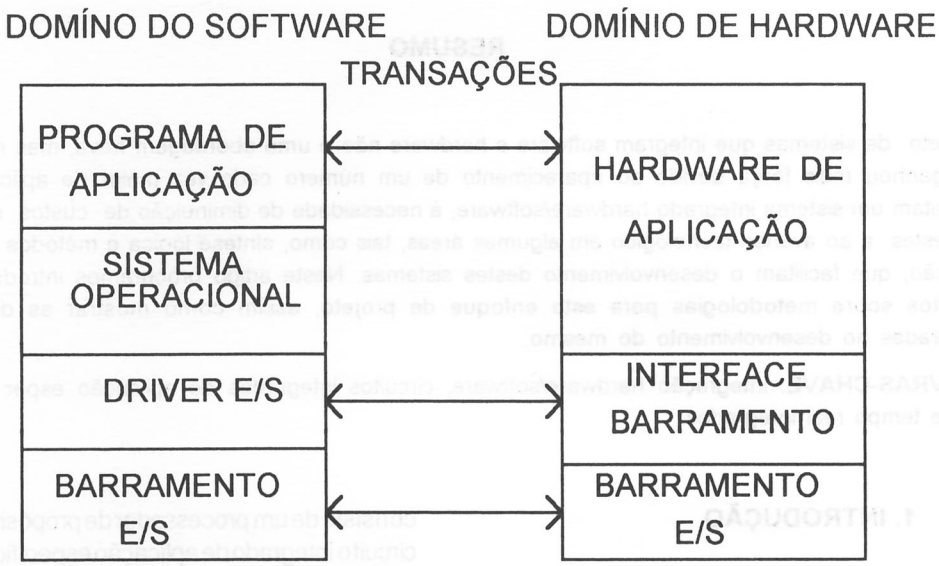

Figura 1 - Modelo de interação hardware/software

Atualmente, os projetos destes sistemas são feitos sem ferramentas adequadas e a metodologia de projeto utilizada é totalmente *ad hoc*. Por isso, para que esta tecnologia seja amplamente aceita na área industrial, é necessário ainda muito trabalho na definição de metodologias específicas para este tipo de projeto. Estas metodologias devem permitir a especificação do sistema, sua partição e principalmente a especificação das interfaces entres as várias partes (software e hardware), além de seus projetos.

O desenvolvimento de ambientes integrados de ferramentas computacionais (os chamados *frameworks*)

[6], que atendam às metodologias desenvolvidas, é imprescindível para o sucesso de um projeto. Este ambiente deve integrar ferramentas de especificação de sistemas com características de definição de interfaces entre hardware e software, ferramentas de particionamento que ajudem a definir qual parte será implementada em software e qual será implementada em hardware, ferramentas de síntese de circuitos, ferramentas de simulação que comportem a simulação conjunta do hardware e do software, ferramentas de teste e outras. Vários esforços estão sendo feitos para propiciar um ambiente integrado que supra os requisitos para este tipo de projeto, com a preocupação principal de definir metodologias e ferramentas destinadas à especificação e descrição funcional de um sistema hardware/software.

Atualmente só existem ambientes integrados de auxílio a projetos, para sistemas de áreas específicas, como por exemplo processamento digital de sinais (DSP´s).

A seguir apresentaremos uma primeira abordagem de metodologia de projeto [1] [2] e discutiremos alguns aspectos práticos para este tipo de projeto.

2. UMA METODOLOGIA PARA HARDWARE/SOFTWARE CODESIGN

Realizar um projeto conjunto hw/sw significa projetar um sistema onde haja a co-existência entre o hardware e o software, ou seja, eles são os componentes do sistema e portanto o seu funcionamento conjunto visa alcançar os objetivos propostos na sua especificação. Com esta definição, temos várias estruturas que se encaixam como hw/sw *codesign*, tanto os ditos aceleradores, onde algumas funções de software são implementadas em hardware, como sistemas onde a concepção do hardware e do software são desenvolvidas em paralelo. Ambos os casos têm uma arquitetura semelhante a mostrada na Figura 2.

A metodologia geralmente utilizada para os aceleradores baseia-se na análise dos pontos críticos da especificação inicial do sistema, normalmente implementada em software, propondo uma partição que será sintetizada em hardware, o qual desempenhará as funções a ele atribuídas de forma mais eficiente do que se permanecesse sob forma de software.

No segundo caso, onde desde a especificação inicial leva-se em conta tanto o subsistema hardware como o software, a metodologia deve ser mais específica, possibilitando maior sucesso no resultado final. Esta metodologia deve possibilitar a co-simulação hardware/software, isto é, a possibilidade de verificar a funcionabilidade de descrições mistas de hardware/software. Ferramentas de síntese devem permitir que rotinas escritas em uma linguagem de programação possam ser implementadas em hardware. Estas metodologias devem também fornecer não só alternativas de particionamento, como um formalismo para descrição de todas estas alternativas.

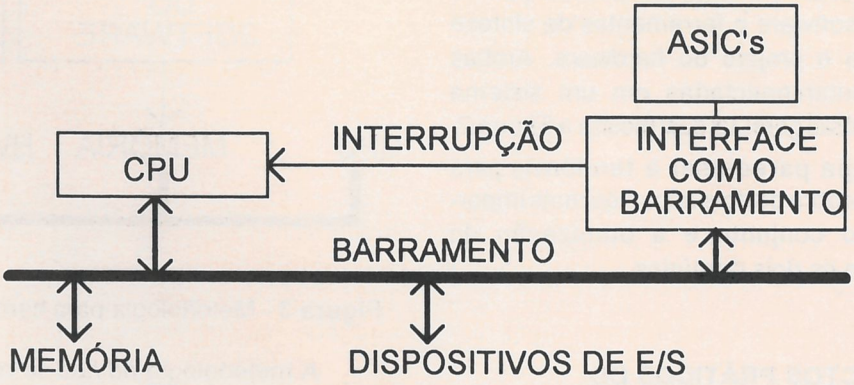

Figura 2 - Arquitetura geral de um sistema hardware/software codesign

O enfoque descrito em [1], é uma metodologia baseada num modelo para simulação e síntese a nível de sistema, que permite um entendimento detalhado do seu comportamento e possui a capacidade de gerar alternativas de projeto, através de co-simulação e co-síntese. Uma característica importante na co-simulação é a ligação do comportamento da simulação de hardware dentro de um ambiente de software. A co-síntese envolve duas características de projeto, interrelacionadas, que são: a escolha do melhor particionamento hardware/software e a escolha do nível de concorrência do controle.

A partição hardware/software é definida pelo conjunto de funções de aplicação implementadas com o hardware de aplicação específica. O particionamento hw/sw leva em consideração explorar ao máximo os recursos do hardware de aplicação específica, visando atingir os objetivos do projeto. A concorrência do controle de um sistema é difinida pelo comportamento e interação entre seus processos. Algumas vezes, para se encontrar um nível de concorrência apropriado, é necessário redefinir os processos através da divisão ou união de alguns deles, objetivando atingir as metas de desempenho do sistema.

Na Figura 3 podemos ver a metodologia em discussão, que usa um ambiente de co-simulação para a descrição mista hardware/software, produzindo algo funcionalmente correto, mas às vezes não atendendo os objetivos das especificações ou mesmo sendo algo não realizável fisicamente, dependendo da tecnologia em questão. A co-síntese modifica a partição hardware/software e o nível de concorrência do controle, visando sempre atingir os objetivos de comportamento e desempenho do sistema.

Como resultado da co-síntese, obtêm-se as especificações do software e hardware para imple-mantação. Na implementação, usam-se técnicas padrão para o desenvolvimento de software e ferramentas de síntese comportamental para o projeto do hardware. Ambas partes são, então, interconectadas em um sistema computacional de propósito geral, tal qual mostra a Figura 2.

Esta metodologia parece ser a tendência para hw/sw *codesign*, pois cobre os dois aspectos mais importantes: a simulação conjunta e a otimização do particionamento entre os dois domínios.

3. ASPECTOS PRÁTICOS DO PROJETO CONJUNTO HW/SW

Codesign é uma prática que vem sendo executada desde os primórdios dos projetos dos primeiros computadores digitais, porém, de uma forma não estruturada [3]. Conceitualmente, é impossível o projeto de uma máquina digital programável, tal como a CPU de um computador, sem a intervenção direta dos usuários de programação, que em última análise, especificam os requisitos de software que esta máquina deve apresentar. Sem dúvida alguma, a indústria que projeta e fabrica microprocessadores e computadores aprendeu, às custas de tentativa e erro, como compatibilizar o projeto do hardware destas máquinas com o projeto do software que são executados nas mesmas. A isto denomina-se de metodologia *ad hoc* de projeto. Não obstante, percebe-se que inicialmente se objetivava o desenvolvimento de sistemas de computação (sistemas de hw/sw) de aplicação genérica.

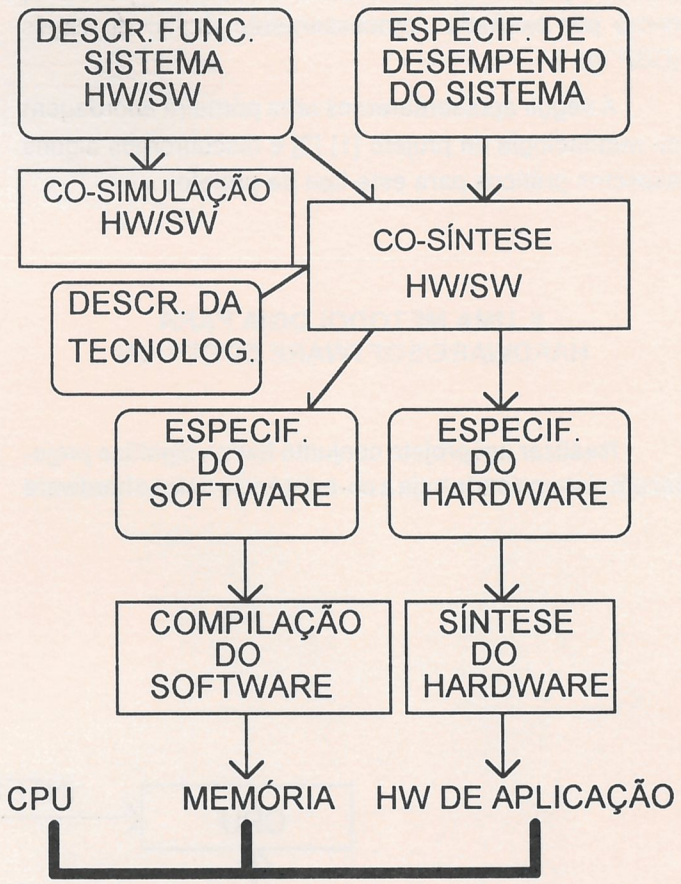

Figura 3 - Metodologia para hardware/software codesign

A metodologia *ad hoc* se refere principalmente ao fato do grupo de projetistas de hardware/software (desde gerentes de projeto até os engenheiros projetistas de *staff*) serem desenvolvidos e treinados *in-house*, pelas condições de competitividade do fabricante. Exemplos

disto, tem-se a Intel, Motorola, IBM, Compaq, Sun, que estão presentes desde os primeiros desenvolvimentos de microprocessadores e computadores e, certamente, possuem em seus *staffs* a reserva de conhecimentos *ad hoc*, à nível gerencial, de projeto e de tecnologias desenvolvidas. Este panorama não é de todo ruim, se considerarmos a grande competitividade deste grupo de fabricantes.

Por outro lado, a tecnologia gerada pelo desenvolvimento dos sistemas de computação modernos é tal, que pode passar da esfera dos fabricantes primários para a esfera dos usuários, que têm interesse em desenvolvimentos específicos e não genéricos. Esta é uma tendência moderna, observada nos últimos cinco anos.

Tecnologias consolidadas de fabricação estão sendo postas à disposição de usuários, que atualmente podem especificar seus produtos baseados não só em sistemas de computação de propósito geral, como também em módulos específicos. A diretriz para este tipo de sistemas sempre foi: o hardware é de propósito geral, cabendo ao software a função de particularizar a aplicação.

O advento dos ASICs, uma tecnologia madura e bem disseminada, propiciou o aparecimento de um novo campo de desenvolvimento, qual seja, hardware específico segundo a especificação do usuário. A pergunta que a partir daí ficou pendente foi: em que condições utilizar hardware específico com vantagens competitivas sobre o hardware de propósito geral e qual a influência desta solução sobre o software?

Por outro lado, não parece sensato reprojetar uma máquina computacional toda a vez que alguma solução específica é buscada. Sensato seria aplicar o conceito de re-utilização de tecnologias (*technology reuse*) [3], que aproveita desenvolvimentos existentes e incorpora aos mesmos novas características. Desta forma, poder-se-ia atender à demanda de novos produtos, baseados em sistemas computacionais existentes, que contivessem em hardware, partes (algorítmos, funções, etc) antes implementadas em software.

Estas partes em hardware seriam implementadas sob forma de circuitos integrados (ASICs), se comunicando com a CPU de sistemas de propósito geral (microcomputadores, estações de trabalho), à maneira que um co-processador aritmético o faz com a CPU principal. Não se descarta a possibilidade de refabricar uma determinada CPU (8086, 486, Pentium, PowerPC ou Spark), integrando juntamente com a mesma as alterações ou complementos de hardware sob especificação do usuário. A forma final de implementação pode ser decidida com base no estudo de custo-benefício de cada opção: para um produto que se imagina um volume de produção entre pequeno-médio (5.000 a 50.000 unidades) e uma vida útil de poucos anos (2 a 3 anos), não se viabiliza economicamente a integração acima referida, e sim, a utilização de uma solução de CPU com ASIC's periféricos.

O próximo problema que se apresenta agora é como otimizar um desenvolvimento desta natureza (CPU de propósito geral + ASIC + software), garantindo o sucesso da implementação, a qualidade do produto final (confiabilidade) e prazos de desenvolvimento compatíveis com a competitividade da indústria (conceito de *time-to-market*). Particularmente a este último aspecto, para sistemas inicialmente inexistentes, deseja-se sobretudo abreviar ao máximo o tempo de desenvolvimento para obtenção de um protótipo do produto final, conceito identificado internacionalmente como Prototipagem Rápida (*Fast Prototyping*) [3] [4].

Nos primórdios da tecnologia para o desenvolvimento de ASIC's, a solução foi a definição e implantação de ambientes integrados de projeto (CAD - *computer aided design*) e de engenharia (CAE - *computer aided engineering*), que num extremo permitem o gerenciamento do projeto - prazos, metas, alocação de recursos, avaliação de custos - e no outro extremo, garantem a consistência de informações deste projeto em suas várias fases - especificação, partição, projeto, integração e testes.

Este panorama sugere que uma abordagem semelhante pode ser realizada para implemetar o projeto conjunto de hw/sw, incorporando também o conceito de prototipagem rápida. Tudo se inicia pela forma que será utilizada para especificar o sistema como um todo. O sucesso de um pacote de ferramentas integradas para *codesign* está diretamente relacionado à disponibilidade de um formato de descrição preliminar do sistema, que permita inicialmente o teste de consistência da especificação, por simulação por exemplo, a partição do sistema e o posterior mapeamento desta descrição em tecnologias disponíveis.

Particularmente, a partição do sistema deve ser o momento mais crítico do projeto, pois está implicitamente considerada a co-existência de hardware e software ainda num formato único e, de alguma forma, por meio de algorítmos adequados, será necessário decidir qual parte será implementada em hardware e qual parte em software. Convém salientar que a escolha da opção por software constitui a solução trivial do problema e a

escolha da opção por hardware caberá somente quando se detetar requisitos de desempenho na especificação do sistema, ou quando outros fatores além de desempenho assim o exigir.

4. CONCLUSÕES

Pensa-se em *codesign* como uma nova metologia de projeto de sistemas digitais, que pressupõe como usuário não somente o grande fabricante de computadores e equipamentos relacionados, mas também, e principalmente, o eventual usuário que desenvolve novos produtos configurados sobre sistemas de propósito geral, que detem o conhecimento de alguma especialidade (instrumentação, aquisição de dados, controle industrial em tempo real, etc) e observa alguma oportunidade de mercado para produtos altamente competitivos.

Dado o contexto acima, fica claro que é altamente desejável a disponibilidade de um ambiente que permita [3]:

- a especificação em alto nível do sistema alvo;
- a verificação de consistência desta especificação por meio de simulação (nos primeiros momentos da especificação, se possível);
- uma análise de alternativas de partição hw/sw;
- um mapeamento da especificação segundo tecnologias disponíveis para uso;
- o gerenciamento do desenvolvimento de cada partição;
- garantia de transparência entre os desenvovimentos das partes de hardware e de software, que eventualmente venham a ser realizados concorrentemente;
- finalmente, a especificação de testes sobre o protótipo obtido.

A idéia de *codesign* pode ser resumida, em termos da tecnologia atual, como engenharia concorrente de hardware e software, com a opção adicional de que se reserva especial atenção para o momento da partição do sistema, pois neste caso há espaço para questionar se algum software poderia ser melhor implementado em hardware.

Existem atualmente ferramentas de desenvolvimento excelentes em ambos os domínios de hardware e software. Falta uma ferramenta integradora (*framework*) [6] que observe e gerencie os aspectos conjuntos dos dois domínios e realize, parcial ou integralmente, a partição do sistema, sem utilizar a metodologia *ad hoc*. Inicialmente, pode-se conceber tal sistema como baseado em premissas de conhecimento de especialistas na área, nos moldes de um sistema especialista (*expert system*). Entretanto, deve-se procurar métodos formais que permitam o reconhecimento automático de estruturas, dirigindo por fim a opção de implementação.

Para circuitos em hardware, existem linguagens de descrição em estágio de evolução bastante avançado e de uso amplamente aceito pela indústria e comunidade científica. São os casos das linguagens VHDL (*Very High-level Hardware Description Language*) [11] e Verilog [1] [12], que permitem a descrição de circuitos digitais de grande complexidade, em alto nível, de forma independente da tecnologia de implementação, permitindo a simulação desta descrição também em alto nível, com possibilidade posterior de mapeamento em diversas tecnologias de *ASIC's*, tais como *gate arrays*, *standard cells* e dispositivos lógicos programáveis (PLDs), esta última opção particularmente interessante ao cenário brasileiro, devido à flexibilidade e baixo investimento [13]. Interessante notar que o VHDL possui bibliotecas de modelos de componentes comerciais, que inclui a maioria das CPUs existentes, permitindo desta forma compatibilizar o projeto de um módulo de hardware com uma dada CPU escolhida, inclusive a nível de simulação.

Para os módulos em software, não existe o equivalente VHSDL (Very High-level Software Description Language) [3], mas sim as linguagens normalmente utilizadas para escrever programas (por exemplo, linguagem C). Alguns autores consideram pouco importante esta não-padronização da descrição do software: necessita-se, na verdade, de métodos sistemáticos para gerenciar a diversidade de linguagens, ao invés de se definir estilos unificados de projeto de software.

Aparentemente, está longe o dia de se ter um sistema completo e geral para *codesign*. Tudo indica que esta diretriz será atingida aos poucos, conquistando áreas de aplicação inicialmente específicas e particularizadas. Com a implementação dos primeiros ambientes, a prática indicará os aspectos ineficientes da metodologia que necessitarão ser otimizados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Thomas, E. D.; Adams, J. K. and Schimit, H.; A Model and Methodology for Hardware-Software Codesign. IEEE Design & Test of Computer, september 1993.

[2] Gupta, R. K.; Coelho, C. N. and Micheli, G. De; Program Implementation Schemes for Hardware-Software Systems. Computer, january 1994.

[3] _____, _____. A D & T Roundtable - Hardware/Software Codesign. IEEE Design & Test of Computer, march 1993.

[4] Wolf, W. ; Hardware-Software Codesign. IEEE Design & Test of Computer, september 1993.

[5] Tuck, B.; Technologies Move Toward Hardware/ Software Codesign. Computer Design, june 1992.

[6] Kumar, S.; Aylor J. H.; Johnson, B. W. and Wulf W. A.; A Framework for Hardware/Software Codesign. Computer, december 1993.

[7] Woo, N. S. and Dunlop A. E.; Codesign from Coespecification. Computer, january 1994.

[8] Carro, L.; Burger, R.; Junqueira A. and Suzim A.; Exploring Parallelism in HW/SW Codesign with a new RISC Architecture. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[9] Carro, L.; Klauck L. and Suzim A.; HW/SW Parallel Operation in a Codesign Environment. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[10] Barros, E.; Lima, M. de and Sampaio, A.; From Hardware/Software Partitioning to Layout Sysnthesis: A Transformational Approach. Anais do VIII Simpósio de Concepção de Circuitos Integrados, novembro 1994.

[11] Armstrong, J. R. ; Chip-level Modeling with VHDL, Prentice Hall, 1989.

[12] _____ , _____ Verilog - Ferramenta proprietária da Cadence Inc (manuais disponíveis).

[13] Behrens, F. H. ; Prototipagem Rápida de Módulos de Hardware para um Ambiente de Projeto Conjunto de HW-SW. Revista do Instituto de Informática - PUCCAMP, vol. 3 nº 2, 1995.