Um Método para a Geração de Sistemas Embutidos Orientados a Aplicação Baseados em SoCs

Fauze Valério Polpeta[1], **Antônio Augusto Fröhlich**[1]

[1]Universidade Federal de Santa Catarina
PO Box 476 - 88049-900 Florianópolis - SC, Brazil
{fauze,guto}@lisha.ufsc.br
http://www.lisha.ufsc.br/~{fauze,guto}

Abstract. *This paper outlines a strategy for automating the design of embedded systems including their hardware and software components. We focus in the* Hardware Mediator *construct, a portability artifact that was originally proposed to enable the port of component-based operating systems to very distinct architectures. Besides giving rise to a highly adaptable system-hardware interface, these mediators are approached as a co-design artifact that can be used to enable the generation of customized* system-on-a-chip *instances and the associated run-time support system considering the specific requirements of the target application.*

Resumo. *Este artigo apresenta uma estratégia para a automatização do projeto de sistemas embutidos abrangendo componentes de hardware e software. Nosso foco está direcionado para o artefato de software denominado* Mediador de Hardware*; mecanismo de portabilidade originalmente proposto para permitir o porte de sistemas operacionais baseados em componentes para diferentes arquiteturas de hardware. Além de dar origem a uma interface software-hardware altamente adaptável, mediadores de hardware são aqui apresentados como um artefato de software-hardware co-design utilizado na geração de plataformas de hardware baseadas em* SoC*s e do respectivo sistema operacional para aplicações dedicadas.*

1. Introdução

Sistemas embutidos têm se tornado cada vez mais complexos, inviabilizando o uso de estratégias de desenvolvimento que decorram em longos períodos de desenvolvimento. Sobretudo, se considerarmos que a competitividade do setor mercadológico de sistemas computacionais embutidos não permite "atrasos". Neste contexto, os dispositivos de lógica programável, ou simplesmente PLDs (do inglês *Programmable Logic Devices*), têm emergido como base para que projetos complexos de hardware sejam embutidos em uma única pastilha de silício. Estes projetos de hardware, quando embutidos em um único *chip*, são referenciados como SoCs (do inglês *System-on-a-Chip*) e são dotados de inúmeras vantagens se comparados às placas de circuito tradicionais. Dentre estas vantagens cabe destacar as relacionadas a fatores como nível de integração, consumo de energia, manutenabilidade e outras métricas de desenvolvimento.

Alguns sistemas embutidos podem ser inteiramente implementados em hardware usando a tecnologia dos SoCs. Porém, o quão mais complexa a aplicação deste sistema, grande é a probabilidade de que ele necessitará de algum tipo de suporte operacional em tempo de execução. Esta talvez seja a razão pela qual muitos grupos têm concentrado esforços

no desenvolvimento de processadores de propósito geral sintetizáveis como Leon2 e OpenRisc [Mattsson and Christensson, 2004]. Mesmo assim, sistemas para suporte em tempo de execução são freqüentemente negligenciados por metodologias e ferramentas de desenvolvimento de SoCs, ficando por vezes restritos a entrega de simples rotinas de escalonamento ou resumidas interfaces software-hardware. A situação se agrava ao considerarmos que um dos principais propósitos de um sistema operacional é garantir a portabilidade das aplicações e sistemas operacionais tradicionais não conseguem lidar com o dinamismo dos SoCs.

Neste artigo discutimos o uso de *Mediadores de Hardware* [Polpeta et al., 2004] como um artefato de *co-design*. O emprego da metodologia Projeto de Sistemas Orientados a Aplicação (AOSD) [Fröhlich, 2001] no contexto sob o qual mediadores de hardware foram originalmente propostos—interface software-hardware—agrega a este mecanismo de portabilidade novo significado em projetos de sistemas embutidos. Mediadores podem ser considerados pontos de partida para se descrever plataformas de hardware que atendem, conjuntamente com o sistema operacional, os requisitos de aplicações dedicadas. As seguintes seções descrevem de forma introdutória a metodologia AOSD, os principais conceitos de mediadores de hardware e como estes são utilizados na geração de sistemas embutidos baseados em SoCs. Na seqüência, um estudo de caso experimental, toma como base uma aplicação implementada sobre o sistema operacional EPOS [Fröhlich and Schröder-Preikschat, 1999], o qual tem nos mediadores de hardware seu mecanismo de portabilidade e também a base para a geração de hardware.

2. Projeto de Sistemas Orientados a Aplicação

Projeto de Sistema Orientados a Aplicação (AOSD) [Fröhlich, 2001] propõe estratégias para se proceder com uma engenharia de domínio no sentido de definir componentes que representem entidades significativas do domínio de sistemas operacionais. Aplicando análise de variabilidade segundo *Projeto Baseado em Famílias* [Parnas, 1976], AOSD viabiliza a modelagem de abstrações independentes que são então organizadas como membros de famílias. Estas abstrações, apesar de realizarem criteriosamente a separação de conceitos que as distinguem entre si, ainda estão sujeitas a dependências que emanam do ambiente onde serão aplicadas. Conseqüentemente, abstrações que incorporam dependências de ambiente tem uma pequena chance de serem reutilizadas em novos cenários. Considerando que um sistema operacional orientado a aplicação está intimamente comprometido com tais mudanças, tais dependências fariam com que sua modelagem fosse depreciada.

A fim de reduzir dependências de ambiente e permitir que abstrações do sistema operacional sejam usadas em diferentes cenários, AOSD agrega ao processo de decomposição o conceito chave da *Programação Orientada a Aspectos* [Kiczales et al., 1997]: a separação de aspectos. Através deste conceito pode-se identificar as variações de cenários que, ao invés de dar origem a novos membros de famílias, definem na verdade aspectos de cenário. Por exemplo, ao contrário de se modelar um novo membro para uma família de mecanismos de comunicação que esteja apto a operar em um ambiente *multithreading*, pode-se modelar *multithreading* como um aspecto de cenário que, quando ativado, bloquearia o mecanismo de comunicação (ou algumas de suas operações) quando em uma seção de código definida como crítica.

Com base nestes conceitos, AOSD tem guiado a engenharia do domínio de sistemas operacionais (veja a Figura 1), modelando componentes que estão fundamentados em três elementos básicos de AOSD: famílias de abstrações independentes de cenário; adaptadores de cenário e interfaces infladas.

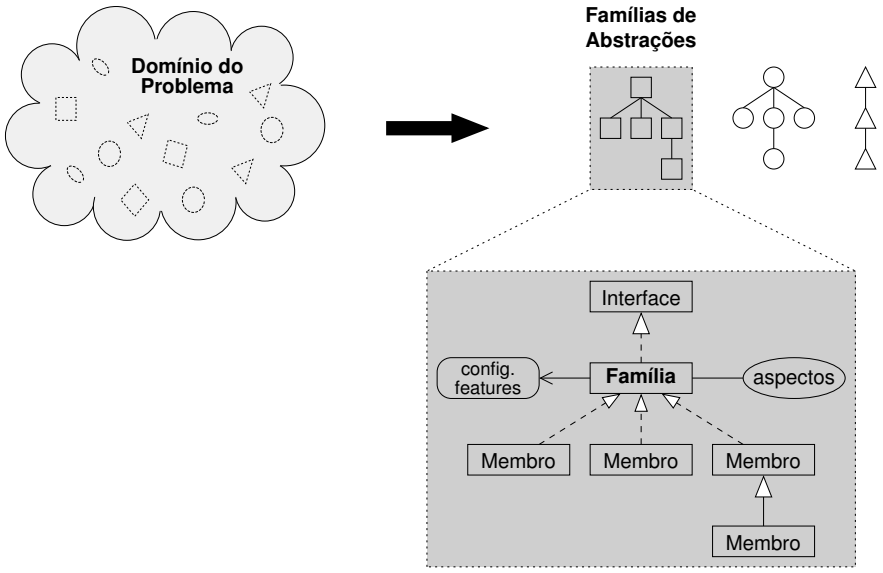

Figura 1: Visão da decomposição de domínio orientada a aplicação.

Famílias de abstrações independentes de cenário

Durante a decomposição de domínio, abstrações são definidas a partir de entidades do domínio e agrupadas em famílias de acordo com suas propriedades comuns. Ao mesmo tempo, a separação de aspectos é usada para que estas abstrações sejam modeladas como componentes independentes do cenário em que serão posteriormente aplicadas.

Adaptadores de cenário

Como citado anteriormente, AOSD permite caracterizar as dependências de cenário que devem ser fatoradas como *aspectos*; mantendo assim, as abstrações do sistema operacional independentes do cenário em que serão empregadas. Entretanto, para que esta estratégia tenha resultado, alguns mecanismos devem ser utilizados para que os aspectos fatorados sejam aplicados sobre as abstrações de maneira transparente. O caminho tradicional para se garantir isto é através do uso de *aspect weavers*, embora os adaptadores de cenário [Fröhlich and Schröder-Preikschat, 2000] tenham as mesmas potencialidades sem que uma ferramenta externa seja usada. Estes adaptadores adaptam uma abstração intermediando sua comunicação através de clientes independentes de cenário que perfazem as adequações necessárias sem gerar *overhead*.

Interfaces infladas

Interfaces infladas sumarizam as propriedades de todos os membros de uma família, criando uma visão única da família como se esta fosse um "super componente". Isto permite aos programadores de aplicação escreverem suas aplicações baseando-se em uma interface bem definida e deixando a decisão de qual membro da família será usado até o momento em que o sistema será gerado. A associação de interfaces infladas a um membro específico da família pode assim ser feita automaticamente por ferramentas de configuração que identificam quais propriedades da família foram utilizadas no intuito de escolher o mais simples membro da família que possui estas propriedades e que deve então ser agregado ao sistema operacional.

3. Mediadores de Hardware

Um sistema operacional projetado de acordo com as premissas de Projeto de Sistemas Orientados a Aplicação pode ser sumariamente visto como conjuntos de componentes de software que podem ser configurados, adaptados e integrados com o objetivo de dar origem a instâncias de sistemas operacionais altamente "customizadas" e específicas ao cenário de aplicação para o qual foram geradas. Entretanto, apesar dos benefícios alcançados pelo emprego da engenharia de software, esta classe de sistema operacionais é passiva da mesma necessidade de portabilidade presente em sistemas operacionais convencionais.

Estratégias tradicionais de portabilidade, focadas principalmente em camadas de abstração de hardware (HAL) e máquinas virtuais (VM) não condizem com os propósitos de AOSD. Frutos de um processo de engenharia de sistema e não de uma engenharia de domínio, estas estratégias usualmente compõem um camada de abstração monolítica que não leva em consideração os requisitos da aplicação. Esta modelagem pode ser um problema quando as plataformas a serem abstraídas são baseadas em SoCs. A diversidade arquitetural e de dispositivos nestas plataformas leva-nos a concluir que as técnicas tradicionais de especificação de interfaces software-hardware estão muito aquém do almejado "*plug-and-play*" [Neville-Neil and Whitney, 2003]. Adicionalmente, estes SoCs, quando implementados em dispositivos de lógica programável como FPGAs (do inglês *Field Programmable Gate Arrays*), estão sujeitos a modificações arquiteturais em curtos espaços de tempo [Rutenbar et al., 2001], fato que agrava ainda mais a portabilidade das abstrações do sistema operacional e da própria aplicação para estas plataformas.

Objetivando a portabilidade das abstrações do sistema operacional para virtualmente qualquer arquitetura, um sistema projetado de acordo com os conceitos de AOSD faz uso do artefato *Mediadores de Hardware* [Fröhlich, 2001, Polpeta and Fröhlich, 2004]. A principal idéia deste artefato de portabilidade não é definir camadas universais de abstrações de hardware, mas sim manter um *contrato de interface* entre as abstrações do sistema e o hardware. Cada componente de hardware é abstraído por um único mediador, garantindo assim a portabilidade sem criar dependências desnecessárias. De fato, mediadores são implementados através de construções meta-programadas e, tão logo o contrato de interface com o sistema é atendido, os mediadores se dissolvem no código gerado para as abstrações que os utilizam. Em linhas gerais, um mediador de hardware entrega a funcionalidade de um componente de hardware através de uma interface orientada ao sistema operacional.

Um importante elemento de mediadores de hardware são as denominadas *configurable features*, que permitem que algumas propriedades dos mediadores possam ser "ligadas" e "desligadas" de acordo com as necessidades das abstrações do sistema. No entanto, estas propriedades não estão restritas somente a *flags* de ativação ou desativação. A implementação de *configurable features* usando *Programação Genérica* [Musser and Stepanov, 1989] permite também que algumas destas propriedades sejam implementadas em software através de estruturas e algoritimos que fazem por suprir sem *overhead* significativo uma capacidade não avaliada no hardware. Um exemplo é caracterizado pela geração de códigos `CRC` por uma *configurable feature* implementada junto ao mediador de um dispositivo de comunicação.

Tal como abstrações em AOSD, mediadores de hardware são organizados em famílias cujos membros representam entidades do domínio de hardware. A Figura 2 ilustra a família de mediadores `NIC` (do inglês *Network Interface Cards*) que, por sua vez, possui como membros os mediadores `91C911`, `MPC8XX` e `RT8139`. Esta modelagem garante que o sistema irá

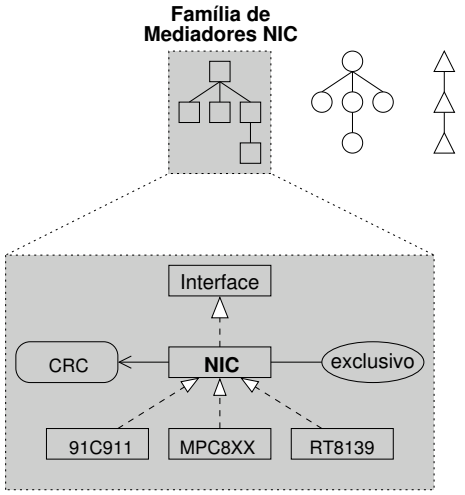

Figura 2: Família de mediadores de hardware para interfaces de rede.

somente incluir código-objeto dos mediadores usados pelo sistema operacional e que, de fato, são requisitos da aplicação. Seguindo a mesma estratégia de modelagem de componentes em AOSD, os aspectos não funcionais ou propriedades ortogonais aos mediadores são fatorados como *aspectos de cenário* que podem ser aplicados aos membros das famílias que os requerem. A título de exemplo, famílias como `UART` (i.e., interface serial) e `NIC` devem freqüentemente operar em modo exclusivo de acesso; isto poderia ser alcançado através da "aplicação" de um aspecto de controle de acesso sobre estas famílias.

4. Geração de SoCs Usando Mediadores de Hardware

Embora projetados originalmente como uma interface software-hardware altamente adaptável, mediadores de hardware podem ser utilizados também na geração de instâncias de hardware orientadas a aplicação. Especificamente, no contexto de lógica programável, onde componentes de hardware são descritos usando linguagens de descrição de alto-nível como VHDL e VERILOG e podem implementar fisicamente os componentes de hardware em uma PLD após um processo conhecido como síntese. É a associação de mediadores de hardware com estas descrições, também conhecidas como IPs (do inglês *Intellectual Property*), que permite-nos inferir quais componentes de hardware são necessários para suportar uma dada aplicação. Mesmo porque, sob a ótica de AOSD, mediadores de hardware são instanciados somente quando identificados como requisitos da aplicação.

Neste sentido, um sistema operacional projetado de acordo com AOSD pode fazer uso de mediadores de hardware não só para compor uma interface entre suas abstrações de sistema e o hardware, mas também para ditar quais componentes constituirão este hardware. Isto é, tão logo um mediador for instanciado para estabelecer uma interface para com um componente de hardware, o IP associado a este mediador será selecionado em um repositório para integrar-se a uma descrição de hardware que será sintetizada em uma PLD. Tal descrição, na forma de um SoC, é composta unicamente dos componentes de hardware necessários para suportar o sistema operacional e conseqüentemente a aplicação. A Figura 3 ilustra a idéia de se usar mediadores para se inferir componentes de hardware. Além da informação relacionada à associação de mediadores de hardware e IPs, a base de dados *Info* contém a informação necessária para configurar os IPs de acordo com os requisitos da aplicação e também para o processo de síntese.

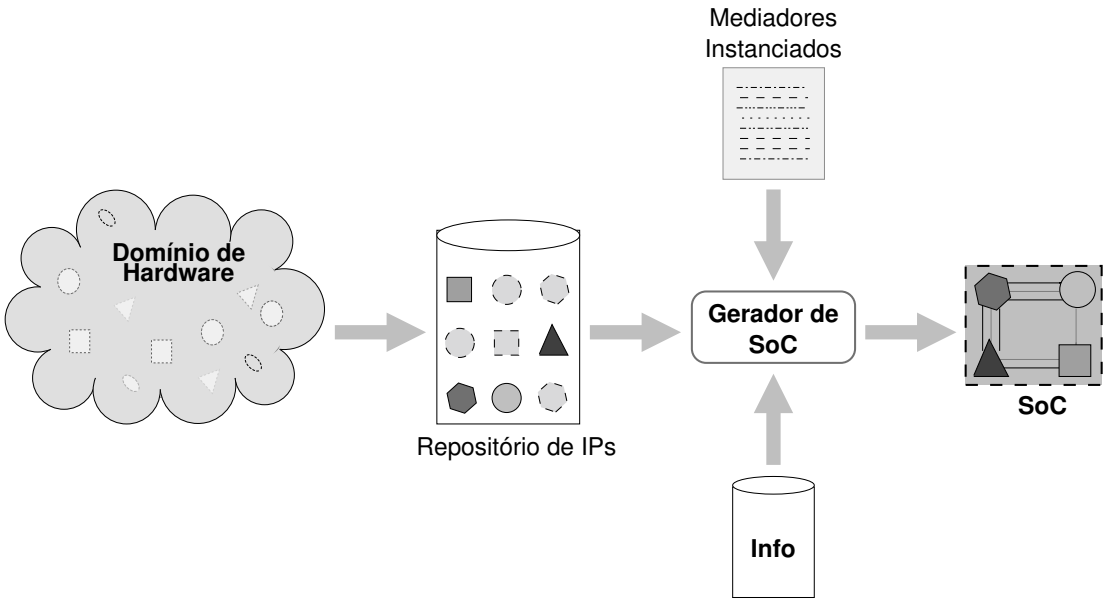

Figura 3: Uso de mediadores de hardware para inferir quais IPs serão instanciados no SoC.

Para exemplificar o uso de mediadores na geração de SoCs, reconsidere a Figura 2 apresentada na seção 3. Como explicado anteriormente, esta figura ilustra a família NIC de mediadores de hardware. A partir de uma dada aplicação que utiliza abstrações do sistema operacional para estabelecer comunicação através de uma interface de rede, pode-se inferir que um membro da família `NIC` será instanciado; mas, a decisão de qual dos membros especificamente é tomada pelo programador. Esta situação caracteriza o que chamamos de *seleção combinada de IP*. Os dispositivos de hardware são inferidos considerando requisitos da aplicação e também através de decisões pontuais do programador da aplicação.

Outro cenário, nomeado *seleção discreta de IP*, está relacionado à seleção de componentes de hardware considerando unicamente os requisitos da aplicação — nenhuma decisão explícita do programador precisa ser tomada. Um bom exemplo para este cenário pode ser deduzido a partir do modelo de gerenciamento de memória a ser implementado pelo sistema operacional em questão. Uma vez que o programador da aplicação usa abstrações do sistema operacional que implementam um modelo *paginado*, uma unidade de gerenciamento de memória (`MMU`) será selecionada para síntese. Inversamente, se um modelo *flat* é adotado a plataforma não irá dispor de uma `MMU`.

Uma terceira situação, referenciada como *seleção explícita de IP*, representa a possibilidade que o programador tem de escolher os componentes de hardware que serão instanciados no sistema. Neste caso, mesmo se o mediador para um dado IP estiver "oculto" por meio de abstrações do sistema, o programador pode explicitamente selecionar os componentes de hardware que deseja ter em seu SoC. De fato, esta estratégia de seleção é sempre tomada quando o programador inicialmente especifica qual arquitetura o sistema adotará, ou seja, SPARCV8, OR32, entre outras.

Por outro lado, a uso de mediadores como artefato na geração de SoCs não está somente relacionado com a seleção de IPs. O elemento *configurable features* explicado anteriomente pode ser também aplicado sobre componentes de hardware no intuito de configurar os IPs para que suportem adequadamente a aplicação e também para "ligar" ou "desligar" funcionalidades

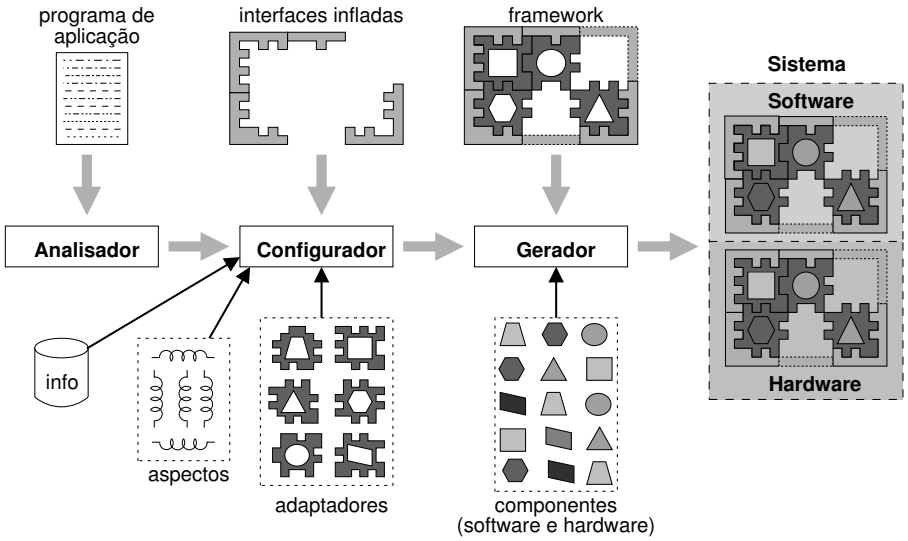

Figura 4: Ferramentas envolvidas na geração automática do sistema operacional e da plataforma de hardware baseada em um SoC.

que em um momento podem estar sendo realizadas pelo hardware e em outro pelo software. Por exemplo, uma *configurable feature* em um mediador de uma porta serial (i.e., `UART`) poderia ativar a geração de códigos CRC, os quais, em outro caso, poderiam ser gerados pelo próprio hardware. Logo, uma *configurable feature* do hardware seria ativada com objetivo de fazer com que as funcionalidades relacionadas a geração de código CRC fossem sintetizadas no SoC.

Outro exemplo interessante da aplicação de *configurable features* em componentes de hardware é a possibilidade da exploração hierárquica de memória tal como apresentado em [Viana et al., 2003]. De acordo com a aplicação final a ser executada sobre a plataforma de hardware pode-se dimensionar adequadamente a memória cache do processador para que o tempo de acesso a dados seja o menor possível.

5. Estudo de Caso: Sistema EPOS

O sistema EPOS (*Embedded Parallel Operating System*) provê suporte operacional adequado para aplicações computacionais dedicadas. Toma a metodologia *Projeto de Sistemas Orientados a Aplicação* como base para o desenvolvimento de famílias de componentes que podem ser adaptados para atender requisitos de aplicações em diferentes cenários. Com o objetivo de manter a portabilidade de suas abstrações de sistema—a princípio, nenhuma das abstrações de sistema interage diretamente com o hardware—e também para viabilizar a geração de SoCs orientados à aplicação, o sistema EPOS faz uso de mediadores de hardware.

Uma aplicação escrita para o sistema EPOS pode ser submetida a uma ferramenta que procura por referências às interfaces infladas. Através destas referências, pode-se então inferir quais famílias e posteriormente quais membros foram utilizados pela aplicação e que, conseqëntemente, irão compor o sistema operacional. Esta ferramenta, denominada *Analisador*, gera uma saída na forma de declarações parciais de interfaces de componentes, incluindo métodos, tipos e constantes que foram utilizadas pela aplicação.

Esta primeira especificação produzida pelo Analisador alimenta uma segunda ferramenta: o *Configurador*, que por sua vez consulta uma base de dados com o objetivo de criar

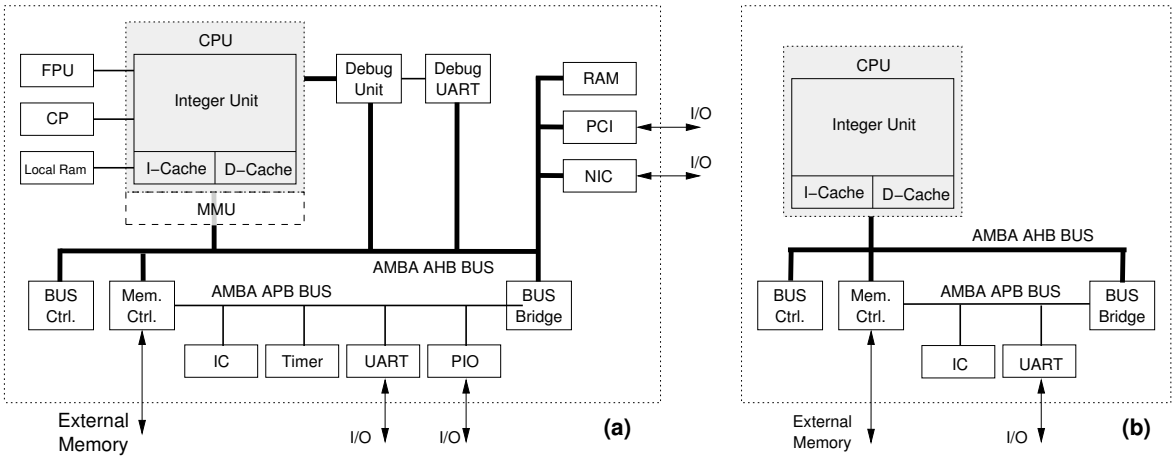

Figura 5: Diagrama de blocos da arquitetura de SoC LEON2 (a) e o SoC gerado para uma aplicação experimental.

uma descrição da configuração do sistema. Esta base de dados contém as informações de cada componente no repositório, bem como as regras de dependência e composição que são usadas pelo Configurador para construir e resolver uma árvore de dependências entre componentes. A saída do configurador consiste de um conjunto de asserções que são usadas para associar interfaces infladas às abstrações do sistema e também para a ativação de aspectos de cenário eventualmente identificados como requisitos da aplicação ou do cenário de execução no qual o sistema está inserido. Quanto aos componentes de hardware, o Configurador produz uma lista do mediadores que foram instanciados e especifica quais destes mediadores promovem a síntese de IPs e quais, especificamente, são estes IPs.

A última etapa no processo de geração é realizada pelo *Gerador*, que traduz as asserções produzidas pelo Configurador em parâmetros de um *framework* meta-programado estaticamente e dispara, subseqüentemente, a compilação do sistema operacional. Adicionalmente, sempre que um SoC precisa ser gerado, o Gerador, tomando como referência as informações extraídas da base de dados pelo Configurador, produz um arquivo de configuração para a síntese do hardware. Escrito em linguagem de descrição de hardware, este arquivo e os IPs selecionados são passados para uma ferramenta de síntese que perfaz a síntese do SoC. Uma visão de todo o processo realizado pelas ferramentas é apresentada na Figura 4.

5.1. Resultados

Com o objetivo de avaliar as potencialidades do método proposto para a geração de sistemas embutidos baseados em SoCs utilizamos a arquitetura de SoC *Leon2* [Inc., 2004]. Esta arquitetura foi desenvolvida com o propósito de dar origem a um SoC configurável" já que sua organização modular permite-nos especificar quais IPs constituirão o SoC. Quanto à lógica necessária para conectar os IPs, esta fica implicitamente definida em seu código-fonte através de asserções de programação que definem como os componentes serão conectados a um *framework* de barramento to tipo AMBA [ARM, 2003]. A Figura 5.a apresenta um diagrama de blocos da arquitetura LEON2. Além da CPU, baseada na arquitetura SPARCV8, vários periféricos são disponibilizados para serem instanciados tão logo a aplicação final e o sistema operacional os requisite.

Os experimentos consistitiram em avaliar uma aplicação a ser executada em um *kit* de desenvolvimento baseado em uma FPGA Xilinx Virtex2, caracterizando um cenário específico

para o qual suporte operacional e um SoC foram gerados. A aplicação consistiu de duas *threads* — TX e RX — sendo executadas em um ambiente cooperativo e objetivando estabelecer comunicação de dados através de uma interface serial. Para isto a abstração de sistema *serial_comunicator* foi utilizada e esta, com base em uma decisão do programador, fez uso do mediador `Leon2_UART` da família `UART`. A fim de identicar o recebimento de dados na interface serial, o mecanismo de interrupções foi utilizado; conseqüentemente, mediador e IP do controlador de interrupções (IC) foram instanciados. Já o modelo de gerência de memória usado na aplicação foi baseado em um espaço de endereçamento *flat*; portanto, os componentes de uma `MMU`, ou seja, mediador e IP, não foram instanciados no sistema. A Figura 5.b apresenta o diagrama de blocos do SoC gerado para esta aplicação.

Objetivando tornar evidente a expressividade destes dados é importante traçar um paralelo com sistemas operacionais tradicionais que foram portados para a plataforma LEON2. Usualmente estes sistemas são gerados no intuito de abstrair todas funcionalidades do SoC mesmo quando a aplicação final não utiliza algumas destas funcionalidades. A ausência de uma engenharia de componentes e do pouco ou nenhum uso de técnicas de engenharia de software afetam não só o tamanho e desempenho do sistema final, mas também custos de produtividade e manutenabilidade. A título de comparação, as imagens de portes tradicionais dos sistemas UCLINUX [Wurm, 2003] e ECOS [Massa, 2002] para a platforma LEON2 possuem respectivamente 2,7 MB e 470 KB e foram portadas para um SoC que ocupa aproximadamente 60% de uma FPGA Virtex2, modelo este, usado no experimento. Já o sistema EPOS gerado neste experimento teve uma imagem de 2.94 Kbytes para um SoC que ocupou 29% da FPGA Virtex2.

6. Conclusão e Trabalhos Futuros

Neste artigo conjeturamos sobre a geração de sistemas embutidos orientados a aplicação baseados em SoCs. Propomos o uso de *mediadores de hardware* como um artefato de *co-design* que pode ser empregado no sentido de permitir que um sistema operacional orientado a aplicação seja gerado em conjunto com a plataforma de hardware que irá suportá-lo. Fazendo uso da metodologia AOSD, associamos mediadores de hardware a componentes de hardware sintetizáveis (i.e., IPs) que, integrados na forma de um SoC, atendem tal como o sistema operacional, os requisitos da aplicação final do sistema. Os resultados apresentados são simples, mas comprovam a idéia apresentada e o conceito de automatização envolvido. Como citado na Seção 4, a possibilidade de se utilizar *configurable features* em componentes de hardware no intuito de melhor adaptá-los à aplicação tem nos incentivado a explorar uma série de fatores que, dentre os quais cabe citar, escalabilidade de processadores, hierarquia de memória, consumo de energia, entre outros. Tão logo os resultados forem obtidos pretendemos apresentá-los à comunidade científica.

Referências

ARM (2003). *The de facto Standard for On-Chip Bus*. Advanced RISC Machines Limited, online document edition. http://www.arm.com/ products/ solutions/ AMBAHomePage.html.

Fröhlich, A. A. (2001). *Application-Oriented Operating Systems*. Number 17 in GMD Research Series. GMD - Forschungszentrum Informationstechnik, Sankt Augustin.

Fröhlich, A. A. and Schröder-Preikschat, W. (1999). High Performance Application-oriented Operating Systems – The EPOS Approach. In *Proceedings of the 11th Symposium on Computer Architecture and High Performance Computing*, pages 3–9, Natal, Brazil.

Fröhlich, A. A. and Schröder-Preikschat, W. (2000). Scenario Adapters: Efficiently Adapting Components. In *Proceedings of the 4th World Multiconference on Systemics, Cybernetics and Informatics*, Orlando, U.S.A.

Inc., G. (2004). *LEON2 XST User's Manual*. Gaisler Research Laboratory, 1.0.22 edition.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J.-M., and Irwin, J. (1997). Aspect-Oriented Programming. In *Proceedings of the European Conference on Object-oriented Programming'97*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242, Jyv skyl, Finland. Springer.

Massa, A. (2002). *Embedded Software Development with eCos*. Prentice Hall PTR, 1 edition.

Mattsson, D. and Christensson, M. (2004). Evaluation of synthesizable CPU cores. Technical report, Chalmers University Of Technology.

Musser, D. R. and Stepanov, A. A. (1989). Generic Programming. In *Proceedings of the First International Joint' Conference of ISSAC and AAECC*, number 358 in Lecture Notes in Computer Science, pages 13–25, Rome, Italy. Springer.

Neville-Neil, G. and Whitney, T. (2003). SoC: Software, Hardware, Nightmare, Bliss. *ACM Queue*, 1(2):24.

Parnas, D. L. (1976). On the Design and Development of Program Families. *IEEE Transactions on Software Engineering*, SE-2(1):1–9.

Polpeta, F. V. and Fröhlich, A. A. (2004). Hardware Mediators: a Portability Artifact for Component-Based Systems. In *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, volume 3207 of *LNCS*, pages 271–280, Aizu,Japan. Springer.

Polpeta, F. V., Fröhlich, A. A., Junior, A. S. H., and D'Agostini, T. S. (2004). Portabilidade em Sistemas Operacionais Baseados em Componentes de Software. In *First Brazilian Workshop on Operating System*, Salvador, Brazil. SBC - Brazilian Computer Society.

Rutenbar, R. A., Baron, M., Daniel, T., Jayaraman, R., Or-Bach, Z., Rose, J., and Sechen, C. (2001). (When) Will FPGAs kill ASICs? In *Proceedings of the 38th conference on Design automation*, pages 321–322. ACM Press.

Viana, P., Barros, E., Sandro Rigo, R. A., and Araujo, G. (2003). Exploring Memory Hierarchy with ArchC. *Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing - SBAC-PAD 2003*.

Wurm, M. (2003). uClinux for Sparc-MMUless with Ethernet MAC. Technical report, Graz University of Technology.