Otimizando a Exploração de Paralelismo Através da Interação entre Arquiteturas e Compiladores

Autores

Marcio Merino Fernandes
Eduardo Marques

1. Introdução

O contexto deste trabalho é o projeto de microprocessadores otimizados para o processamento de aplicações específicas. Esta estratégia é de grande interesse principalmente na área de sistemas embarcados projetados na forma de um SoC (System-on-Chip), e implementados com hardware reconfigurável. Atualmente, técnicas de hardware e software para o desenvolvimento de sistemas embarcados despertam grande interesse, uma vez que essa plataforma tem sido cada vez mais utilizada [1] [2]. O uso de arquiteturas do tipo VLIW [18] tem-se mostrado eficiente em projetos desse tipo, servindo como base para sistemas paralelos de granularidade baixa. As próximas seções deste documento discutem com maiores detalhes estes e outros conceitos relacionados com o projeto de pesquisa apresentado.

Pode-se dizer que existem duas classes de microprocessadores: uma dedicada ao processamento de uso geral, tipicamente encontrada em servidores ou computadores pessoais (PCs), e outra voltada ao processamento especializado, na forma de sistemas embarcados ou SOCs [2]. Historicamente, a maior motivação para o desenvolvimento de processadores de uso geral é a busca pelo aumento de desempenho na execução de processamento. Nos últimos 20 anos isso tem sido possível através de uma combinação de arquiteturas cada vez mais elaboradas, e freqüências de clock dobrando a cada 18 meses (*Lei de Moore*). Entretanto, nos últimos anos tem se verificado uma desaceleração na taxa de aumento de freqüências de clock, uma tendência reconhecida pelo fabricante de microprocessadores Intel. Isso sinalizou o fim da chamada “corrida de gigahertz” no mercado de microprocessadores, fazendo com que fabricantes passem a competir em termos de *processor cores* e *threads* [2]. Em contraste, o projeto de processadores para sistemas embarcados compreende todo o espectro de parâmetros como desempenho,consumo de energia, custo e complexidade de projeto [6]. Implementações de SoCs possuem algumas vantagens sobre implementações para uso geral, sendo particularmente interessante a possibilidade de especializar o processador para uma determinada função. Isso permite em muitos casos a obtenção de ganhos de desempenho através da exploração do paralelismo inerente da aplicação, com o auxílio de um modelo de arquitetura que se adapte bem à função. Dessa forma, ganhos de desempenho tornam-se possíveis sem um aumento significativo (e às vezes desnecessário) na velocidade de clock, e do consumo de energia, um parâmetro cada vez mais importante para a categoria crescente de dispositivos portáteis. No projeto de microprocessadores para SoCs tem-se a vantagem óbvia da disponibilidade de um conjunto bem definido de requisitos. Entretanto, o espaço de possíveis soluções e arquiteturas é bastante vasto, exigindo flexibilidade no projeto e implementação desta classe de microprocessadores. O uso de computação reconfigurável [3] está se consolidando como uma alternativa importante para o desenvolvimento de sistemas embarcados, especialmente na forma de SoCs [4]. Com esse modelo, sistemas de alto desempenho podem ser construídos utilizando-se um ou mais microprocessadores, na forma de *softcores* [7], e blocos de hardware customizado. Um modelo de arquitetura com crescente interesse para o projeto de processadores de alto desempenho são as chamadas *arquiteturas reconfiguráveis de granularidade baixa* (CGRAs – Coarse *Grained Reconfigurable Architectures*).

CGRAs tipicamente possuem um conjunto de elementos de processamento (PEs), cada um possuindo uma ou mais unidades funcionais (FUs), e dispositivos de armazenamento de dados (registradores ou memória local). As facilidades oferecidas por uma plataforma de hardware reconfigurável possibilitam uma grande variação de possibilidades em termos de funcionalidade e interconexão de elementos de processamento [5]. O uso adequado de CGRAs permite ganhos de desempenho significativos na execução de *loops* e outras funções computacionalmente intensivas, as quais frequentemente determinam o desempenho global de aplicações em diversas áreas. Entretanto, apesar dos diversos modelos de arquitetura propostos ao longo dos anos, ainda existe pouco suporte de *compilação* para o projeto e exploração eficiente dessa classe de arquiteturas. Uma alternativa interessante é o uso de um modelo de arquitetura *modular* e *parametrizável*, o que pode facilitar análise e experimentos em busca da melhor solução para a aplicação em questão [5] [8].

Arquiteturas do tipo VLIW (Very Long Instruction Word) podem ser utilizadas como base para o desenvolvimento de CGRAs. Isso se deve às possibilidades de parametrização desse tipo de arquitetura, bem como ao estado atual das técnicas de compilação para exploração de paralelismo a nível de instruções [9]. Nesse tipo de arquitetura, grupos de unidades funcionais pertencentes a um mesmo *chip* operam em paralelo, executando um *scheduling* de instruções estaticamente definido em tempo de compilação. A exploração mais agressiva de paralelismo leva a necessidade de organizar grupos de unidades funcionais em clusters [10] , uma estratégia apropriada para implementação utilizando-se hardware reconfigurável. Esse tipo de arquitetura oferece suporte para a execução de código otimizado através de técnicas como *software pipelining* [11], reconhecidamente eficiente na aceleração de *loops*. Neste contexto, alguns trabalhos recentes tem alcançado resultados positivos com a utilização de arquiteturas VLIW para o desenvolvimento de CGRAs**.**

A arquitetura Adres inclui um processador VLIW acoplado a um CGRA, este responsável pela execução de *kernels* críticos para o desempenho de aplicações [8]. Ambos os módulos possuem componentes comuns, permitindo assim o compartilhamento de recursos em suas diversas configurações utilizando FPGAs. A arquitetura ARRIVE é constituída por um processador RISC (ARM7) acoplado a um CGRA, similar a um processador VLIW/SIMD [12]. Esse modelo permite a exploração eficiente do paralelismo a nível de instruções encontrado em aplicações do tipo DSP (digital signal processing). Outra arquitetura VLIW reconfigurável exibindo ganhos de desempenho significativos é apresentada em [13]. Adotando uma abordagem mais genérica, uma metodologia formal para o projeto de processadores VLIW reconfiguráveis é apresentado em [14]. Essa metodologia utiliza técnicas de compilação avançadas para a identificação de trechos críticos do programa (em termos de desempenho), e geração de unidades funcionais dedicadas, as quais operam conjuntamente a um processador VLIW. Técnicas de compilação referentes ao sistema de memórias para arquiteturas do tipo CGRA é o foco do trabalho apresentado em [5]. Em outro trabalho, o compilador DRESC utiliza técnicas para a aceleração de *loops (software pipelining)* em arquiteturas CGRAs [15].

2. Objetivos

Levando-se em conta o contexto apresentado na introdução deste documento, em particular no que diz respeito ao uso de arquiteturas VLIW para o projeto de processadores reconfiguráveis de granularidade baixa, serão apresentados a seguir os problemas de pesquisa que o projeto proposto pretende abordar, bem como os objetivos primários a serem atingidos nesse sentido.

· **P1:** Existe uma grande demanda por processadores customizáveis, voltados a aplicações específicas, fora do domínio convencional de servidores e computadores pessoais. Processadores desse tipo são tipicamente utilizados em sistemas embarcados, na forma de SoCs. Atualmente, existe uma demanda crescente para

projetos de sistemas multimídia e de realidade aumentada [25] empregando processadores desse tipo.

· **P2:** Na maioria dos casos, o uso adequado de processadores customizáveis de alto desempenho é viável apenas caso existam técnicas de compilação capazes de gerar código otimizado para a arquitetura alvo. Nesse sentido, o processo de interação entre compilador e arquitetura deve apoiar-se nas características dos programas de aplicação a serem utilizados.

· **P3:** Em muitos casos é suficiente concentrar-se apenas em alguns trechos do código da aplicação, os quais são responsáveis por uma fração significativa do tempo total de execução. Esse é o caso típico de alguns *loops*, ou operações de acesso à memória, que exibem bom potencial para exploração de paralelismo.

· **P4:** Plataformas de execução utilizando hardware reconfigurável constituem-se em uma alternativa viável e interessante para a implementação de processadores customizáveis.

· **P5:** Arquiteturas VLIW oferecem boas possibilidades para o projeto de processadores em hardware reconfigurável, já que permitem a replicação de elementos de processamento relativamente simples. Além disso, possuem um conjunto de técnicas de compilação bem desenvolvidas.

O problema a ser abordado por este projeto de pesquisa situa-se no contexto das premissas P1 a P5, concentrando-se diretamente na premissa P2. Dessa forma, é possível defini-lo da seguinte maneira:

Problema de Pesquisa: *Melhorar as técnicas existentes para a definição da arquitetura de processadores baseados em CGRAs, a serem utilizados no domínio de aplicações multimídia, e de suporte a sistemas de realidade aumentada.*

Uma vez definido o problema a ser abordado, o projeto proposto deverá concentrar-se nos seguintes **objetivos primários** (**Oi**):

· **O1:** Definir uma arquitetura reconfigurável de granularidade baixa (CGRA), baseada no modelo VLIW. A arquitetura deverá ser genérica, parametrizável, e adequar-se bem ao gerador de código do compilador a ser adotado pelo projeto.

· **O2:** Desenvolver e adequar técnicas de compilação existentes para a geração de código otimizado para a arquitetura definida em O1.

· **O3**: Desenvolver técnicas de compilação para mapeamento *automático* e *otimizado* de loops em estruturas customizadas de hardware reconfigurável. Para isso, serão utilizadas técnicas como *software pipelining e loop unrolling, entre outras* [26].

· **O4**: Desenvolver soluções de arquitetura e técnicas de compilação para otimização de acesso à memória da arquitetura definida em O1.

· O avanço no estado da arte das técnicas para a exploração das possibilidades de projeto de processadores para sistemas embarcados possui grande potencial para impulsionar o desenvolvimento de sistemas apoiados nesse modelo.

· O estado da arte das técnicas de compilação para o tipo de arquitetura abordado pelo projeto ainda possui limitações no que diz respeito à utilização de técnicas agressivas para a execução otimizada de loops e acesso a memória. Isso se deve principalmente às dificuldades para se explorar eficientemente um amplo espaço de opções para a organização da arquitetura.

· O Brasil possui algumas limitações, principalmente econômicas, para a implantação de uma indústria de microeletrônica, o que causa dependência externa na área de sistemas embarcados, entre outras. O domínio de metodologias avançadas que viabilizem localmente o projeto e implementação de processadores pode diminuir essa dependência econômica (pagamento de *royalties*).

3. Desenvolvimento

Para que se alcancem os objetivos primários definidos anteriormente, foi definida uma série de tarefas (**Ti**), as quais compõem a base da metodologia a ser empregada para a condução do projeto.

Esse processador será utilizado como arquitetura alvo do compilador, na execução otimizada de loops, e outros trechos de programa que exibam paralelismo a nível de instruções suficiente e identificável pelo compilador. Dessa forma, busca-se resolver uma das maiores dificuldades encontradas para a aplicação de técnicas como software pipelining, que é a necessidade de suporte de hardware especial, não encontrado em processadores convencionais [16][17]. Além disso, a arquitetura deverá oferecer recursos para a execução dos demais trechos do programa, possivelmente através de um de seus elementos de processamento operando de forma escalar. Deve ser observado que muitas das técnicas a serem desenvolvidas possuem suas raízes nas arquiteturas VLIW [18][9], porém fazendo-se uso da flexibilidade adicional oferecida por uma arquitetura parametrizável, implementada com tecnologia de hardware reconfigurável.

O detalhamento do projeto do processador deverá ser no *nível de arquitetura,* incluindo elementos como unidades funcionais, arquitetura do conjunto de instruções, registradores, memória local, memória compartilhada, topologia de interconexão entre componentes, etc. Dessa forma, poderão ser usados como parâmetros de ferramentas de simulação, a serem utilizadas para a obtenção de estimativas e resultados experimentais. A eventual implementação de algumas instâncias da arquitetura genérica depende do resultado de pedidos de financiamento para a compra de equipamentos, especialmente placas de FPGA e ferramentas de EDA [27], os quais serão encaminhados no momento oportuno.

O ponto central deste projeto é um *compilador* capaz de gerar código otimizado para uma arquitetura parametrizável do tipo CGRA-VLIW. É fundamental que o compilador ofereça recursos para a identificação de *paralelismo a nível de instruções*, especialmente em estruturas de repetição (*loops*). Na prática isso se exige *análises de dependências de dados e de controle sofisticadas*. Além disso, é desejável que o compilador possua um gerador de código (*back-end*) para a plataforma de hardware reconfigurável a ser adotada, ou facilidades para a implementação desta. A princípio, a opção a ser adotada é o Compilador Galadriel-Nenya [19], atualmente utilizado pelos autores como parte de um projeto de cooperação internacional Brasil-Portugal. Alternativamente, poderá ser adotada a infra-estrutura Trimaran, um conjunto de ferramentas de compilação e simulação para arquiteturas VLIW [20].

Conforme colocado na descrição da tarefa T2, considera-se a possibilidade de efetuarem-se algumas mudanças internas no compilador a fim de se *identificar* o paralelismo a nível de instruções [9] disponível no código de aplicações, gerando grafos de fluxo de dados (DAGs) e código intermediário com maior potencial para a exploração dos recursos de paralelismo da arquitetura. A especificação completa dessas tarefas depende das características internas do compilador a ser adotado.

A princípio serão implementadas as otimizações para geração de código conhecidas como *software pipelining* [11] e *loop unrolling* [21]. Dependendo dos resultados obtidos, outras otimizações [26] poderão ser implementadas, uma vez que a infra-estrutura de compilação para isso é basicamente a mesma, principalmente no que se refere às fases de análise do compilador (tarefa T3). Deve ser observado que esta tarefa é altamente dependente da arquitetura alvo do compilador, o que caracteriza a *interação* no desenvolvimento conjunto da arquitetura e do compilador. Mais especificamente, o paralelismo disponível na arquitetura pode ser efetivamente utilizado apenas se o compilador for capaz de gerar código otimizado para seus recursos. Por outro lado, diversas otimizações avançadas dependem de recursos de arquitetura não convencionais, o que é facilitado quando se utiliza uma arquitetura flexível e parametrizável, conforme propõem este projeto.

Uma vez gerado código intermediário otimizado (tarefa T3), este deverá ser compilado para execução no processador VLIW definido na tarefa T1. Em um compilador convencional a geração de código seria para o conjunto de instruções da arquitetura alva, que geralmente é fixo. No caso do projeto proposto, a arquitetura alvo pode se adaptar às necessidades de processamento do código intermediário otimizado, em particular no que diz respeito ao paralelismo, e também à necessidade do uso de registradores para armazenar valores intermediários [16]. Por outro lado, é possível que determinados trechos de computação sejam mapeados diretamente para estruturas de hardware customizado, operando conjuntamente com o processador VLIW. É importante ser ressaltado que *não se pretende desenvolver técnicas básicas para o mapeamento de código intermediário em estruturas de hardware*. Caso esta opção seja colocada em prática, será através da utilização do compilador Galadriel-Nenya [19], que já oferece recursos para isso (ver tarefa T2).

Pretende-se com esta tarefa fazer uma análise *quantitativa* de desempenho do código otimizado gerado (T4). Além disso, também será feita uma análise *qualitativa* do código, verificando-se possíveis oportunidades para o desenvolvimento de otimizações adicionais. O objetivo desta tarefa é se concentrar unicamente nos trechos de código otimizado, evitando-se assim interações que afetem o trabalho de análise. Inicialmente serão utilizados loops simples, na forma canônica. Em uma segunda etapa serão selecionados loops e outros trechos de código presentes em alguns benchmarks relevantes para esta área de pesquisas. Entre eles, podemos citar os seguintes:

O *benchmark* MediaBench é um conjunto de programas e trechos de código tipicamente encontrados em aplicações multimídia. Já o MiBench contém *kernels* de aplicações de sistemas embutidos, similar ao *benchmark* EEMBC, padrão de referência usado pela industria de microprocessadores, mas praticamente inacessível às instituições acadêmicas. O Perfect Club Benchmarks, apesar de ser voltado à aplicações científicas, possui um bom conjunto de *loops* exibindo características importantes para este tipo de trabalho. A princípio a avaliação de desempenho será feita de modo estático, ou seja, analisando-se parâmetros como

número de instruções executadas em cada ciclo (IPC). Conforme mencionado na descrição da tarefa T1, caso se obtenha financiamento para a aquisição de placas de FPGA, será possível a obtenção de resultados experimentais mais detalhados através da execução desses benchmarks em uma implementação real de algumas instâncias da arquitetura proposta. De qualquer modo, esse não é um requisito indispensável para que se obter uma avaliação confiável sobre a efetividade das técnicas a serem desenvolvidas ao longo do projeto.

4. Resultados

Este projeto está relacionado com outros trabalhos de pesquisa atualmente em andamento. Em particular, existe uma forte cooperação entre os autores deste artigo e pesquisadores da Universidade Técnica de Lisboa, a qual é apoiada formalmente pelo CNPq, através de um projeto internacional (**Convênio: CNPq/Grices Processo Número: 49.0884/2004-0).**

Um outro pedido de financiamento para este projeto foi submetido recentemente ao CNPq, encontando-se em fase de análise. Estima-se um tempo total de execução igual a 36 meses, com previsão de início para 1-Março-2007, com algumas tarefas sendo executadas em paralelo. Em alguns casos isso é simplesmente uma opção, como no caso das tarefas T1 e T2: os primeiros estudos para a definição da arquitetura podem ocorrer em paralelo com a avaliação do compilador a ser adotado. Por outro lado, outras tarefas devem necessariamente ser executadas em paralelo, como é o caso de boa parte das tarefas T1 e T4: neste caso o refinamento da arquitetura deve ocorrer conjuntamente com as técnicas de compilação para geração de código, já que muitas vezes essas técnicas exigem recursos de arquitetura que não foram anteriormente previstos. Similarmente, parte da tarefa T3 deve ocorrer simultaneamente com a tarefa T4, uma vez que ajustes nos módulos de análise do compilador devem ocorrer devido às particularidades do gerador de código. A tarefa T5 deverá ser executada na fase final do projeto, juntamente com T6, que trata da finalização da análise experimental. Deve ser notado que avaliações experimentais serão conduzidas a partir dos primeiros resultados obtidos na execução da tarefa T4, já que isso é de fundamental importância para guiar o processo de desenvolvimento e tomada de decisões referentes à arquitetura e técnicas de compilação correspondentes.

5. Considerações Finais

Ao final da execução deste projeto espera-se que alguns dos seus resultados possam contribuir para avançar o estado da arte na área do problema de pesquisa que ele aborda: “*Melhorar as técnicas existentes para a definição da arquitetura de processadores baseados em CGRAs, a serem utilizados no domínio de aplicações multimídia, e de suporte a sistemas de realidade aumentada”.* Mais especificamente, existe a perspectiva de avanços nos seguintes pontos:

o Heurísticas para reduzir o espaço de busca de soluções adequadas para o projeto de processadores

customizados.

o Aprimoramentos em técnicas de compilação existentes, e possível desenvolvimento de novas técnicas.

o Exploração mais eficiente dos recursos de hardware disponíveis em arquiteturas reconfiguráveis.

o Definição de algumas instâncias da arquitetura VLIW genérica que apresentem bom desempenho na execução de aplicações multimídia ou de realidade aumentada. Estas podem ser apropriadas para a implementação não apenas em hardware reconfigurável, mas também utilizando tecnologia VLSI.

A área e o problema abordado pelo projeto proposto são de grande interesse atual para atividades relacionadas com o projeto de microprocessadores, e ferramentas de software correspondentes (*tools chain*). De acordo com a experiência do proponente, resultados positivos deste projeto podem resultar na concessão de **patentes**, tanto no Brasil como no exterior. Além disso, podem facilitar a aproximação para futuros trabalhos em cooperação com centros de pesquisa dos grandes fabricantes de microprocessadores, e tecnologias correlatas.

Referências Bibliográficas

[1] RAU, B., SCHLANSKER, M. Embedded computing: New directions in architecture and automation, *Tech. Rep. HPL-2000-115*, Hewlett Packard Laboratories. Sept. 29, 2000.

[2] FLYN, M. J., HUNG, P. Microprocessors Design Issues: Thoughts on the Road Ahead*. IEEE Micro,* Mai-Jun 2005.

[3] COMPTON, K., HAUCK, S. Reconfigurable Computing: A Survey of Systems and Software. *ACM Computing Surveys*, v. 34, 2002.

[4] EDENFELD, D., KHANG, A. RODGERS, M., ZORIAN, Y. 2003 Technology Roadmap for Semiconductors. *Computer* v. 37, 2004.

[5] LEE, J., CHOI, K., DUTT, N. Compilation Approach for Coarse Grained Reconfigurable Architectures. *IEEE Design and Test of Computers*. Jan-Feb, 2003.

[6] WONG, S., VASSILIADIS, S., COTOFONA, S. Embedded Processors: Characteristics and Trends, *Technical Report CE-TR-2004-03*, Delft, The Netherlands, May 2004.

[7] LYSECKY, R. VAHID, F. A Study of the Speedups and Competitiveness of FPGA Soft Processor Cores using Dynamic Hardware/Software Partitioning, *Proc. Design Automation and Test in Europe Conference (DATE'05),* March, 2005;

[8] MEI, B., LAMBRECHTS, A., VERKEST, D., MIGNOLET, J., LAUWEREINS, R. Architecture Exploration for a Reconfigurable Architecture Template. . *IEEE Design and Test of Computers*. Mar-Apr, 2005.

[9] RAU, B., FISHER, J. Instruction-level parallel processing: History, Overview, and Perspective. *The Journal of Supercomputing*, July, 1993.

[10]FERNANDES, M., LLOSA, J., TOPHAM, N. Distributed Modulo Scheduling. *HPCA-5 – Fifth International Symposium on High Performance Computer Architecture*, Orlando, USA, 1999.

[11]RAU, B. Interative Modulo Scheduling. *The International Journal of Parallel Programming,* February, 1996.

[12]ZABEL, M., KHOLER, S., ZIMMERLING, M., PREU²ER, T., SPALLEK, R. Design Space Exploration of Coarse Grain Reconfigurable DSPs. *Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs*, Puebla City, Mexico, 2005.

[13]JONES, A., HOARE, R., KUSIC, D., FAZEKAS, J., FOSTER, J. An FPGA-based VLIW Processor with Custom Hardware Execution. *Proceedings of FPGA´2005*, Monterrey USA, 2005.

[14]POZZI, L. Methodologies for the Design of Application Specific Reconfigurable VLIW Processors. PhD Thesis. Politechnico di Milano, 2000.

[15]MEI, B., VERNALDE, S., VERKEST, D., De MAN, H., LAUWEREINS, R. Exploiting Loop Level Parallelism on Coarse Grained Reconfigurable Architectures Using Modulo Scheduling. *Proceedings of Design Automation and Test in Europe*, 2003.

[16]RAU, B., LEE, M., TIRUMALAI, P., SCHLANSKER, M. Register Allocation for Software Pipelined Loops. *Proceedings of the ACM SIGPLAN´92 - Conference on Programming Language Design and Implementation*, 1992.

[17]RAU, B., TIRUMALAI, P., SCHLANSKER, M. Code Generation Schema for Modulo Scheduled Loops. *Proceedings of the MICRO-25, The 25th Annual International Symposium on Microarchitecture*, 1992.

[18]FISHER, J. Very Long Instruction Word Architectures and the ELI-512. *Proceedings of the 10th Annual International Symposium on Computer Architecture*, 1983.

[19]CARDOSO, J., NETO, H. Compilation for FPGA-Based Reconfigurable Hardware, IEEE *Design & Test of Computers Magazine*, March/April, 2003.

[20]CHAKRAPANI, L., GYLLENHAAL, J., HWU, W., MAHLKE, S., PALEM, K., R., RABBAH. Triamaran: An

Infrastructure for Research in Instruction-Level Parallelism. *Lecture Notes in Computer Science*, v. 3602, pg. 32-41, 2005.

[21]CARDOSO, J., DINIZ, P. Modeling Loop Unrolling: Approaches and Open Issues. *Proc. Of Int'l Workshop on Systems, Architectures, Modeling, and Simulation (SAMOS IV),* Samos, Grécia, July 2004, LNCS 3133, Springer Verlag.

[22]BERRY, M., CHEN, D., KOSS, P., KUCK, D. The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers, *Technical Report 827*, CSRD, Univ. of Illinois at Urbana-Champaign, Nov. 1988

[23]LEE, C., POTKONJAK, M., MANGIONE-SMITH, W. Media- Bench: a Tool for Evaluating and Synthesizing Multimedia and Communication Systems, *Proceedings . of Int. Symposyum. On Microarchitecture,* Dec. 1997

[24]GUTHAUS, M., RINGENBERG, J., ERNST, D., AUSTIN, T., MUDGE, T., BROWN, R. MiBench: A free, commercially representative embedded benchmark suite. *IEEE 4th Annual Workshop on Workload Characterization*, Austin, TX, December 2001.

[25]SMITH, R., PIEKARSKI, W., WIGLEY, G. Hand Tracking for Low Powered Mobile AR User Interfaces. *Proceedings of 6th Australasian User Interface Conference (AUIC2005),* Newcastle, Australia, 2005.

[26]BACON, D., GRAHAM, S., SHARP, O. Compiler Transformations for High Performance Computing. *ACM Computing Surveys*, pg. 345-420*,* Dec. 1994*.*

[27]Altera Corporation. Nios II Development Kit, Stratix II Edition. http://www.altera.com/products/devkits/altera/kit-niosii-2S30.html. Acesso em 15-05-2006.