High-Performance Timing Simulation of Embedded Software

J. Schnerr, O. Bringmann, A. Viehl, and W. Rosenstiel, "High-performance timing simulation of embedded software," in **Design Automation Conference**, 2008. DAC 2008. 45th ACM/IEEE, 2008, pp. 290-295. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4555825

Autor do resumo: Roberto Pereira; RA: 089089; e-mail: robertop.ihc@gmail.com

Neste artigo, os autores apresentam uma abordagem híbrida para a simulação de precisão de ciclos (*cycle-accurate*) em softwares embarcados, fundamentam e justificam a proposta, e realizam alguns experimentos básicos para demonstrar a viabilidade da abordagem. A pesquisa está contextualizada na tendência das funcionalidades de um sistema serem efetuadas cada vez mais por cooperação, interconexão e distribuição de componentes em vez da utilização de componentes isolados. Esta tendência de mudança no produto implica, também, em mudanças de paradigma no projeto: o impacto de um componente no sistema como um todo **deve** ser considerado, e para isso, é preciso uma modelagem abrangente e uma análise e simulação da integração de sistema. Para os autores, o ponto-chave na análise de desempenho de sistemas embarcados é saber *qual é o comportamento de timing* (*timing behavior*) de um sistema e *como ele pode ser determinado*. Assim, o artigo apresenta uma breve discussão sobre a análise de desempenho em sistemas embarcados distribuídos e, então, propõe uma abordagem híbrida que **combina características** tanto da *abordagem analítica* quanto da *simulada* com o intuito de melhorar a velocidade da simulação sem ocasionar grande perda na precisão dos resultados.

Para os autores, as análises estáticas de pior/melhor tempo de execução (WCET/BCET) oferecem resultados muito pessimistas quando se considera além dos blocos básicos — e que são reforçados quando se leva em conta efeitos de concorrência no uso da cache de diferentes aplicações em arquiteturas multi-core. A abordagem proposta aplica um esquema de *back-annotation* dos valores WCET/BCET determinados estaticamente dos blocos básicos do código binário gerado a partir do código-fonte em C e busca reduzir o pessimismo característico da análise estática WCET/BCET.

Basicamente, a abordagem consiste em: **1)** traduzir o código-fonte C para um código binário específico para o processador desejado (utilizando *cross-compilation*). **2)** Ler, por meio da uma ferramenta desenvolvida para realizar a anotação (*back-annotation*), o código gerado e a descrição das características do processador que será considerado na simulação. **3)** Decodificar o código-fonte e traduzi-lo numa representação intermediária composta por uma lista de objetos (instruções intermediárias). **4)** A partir da lista de objetos construir uma lista de blocos básicos do programa. **5)** Calcular estaticamente o número de ciclos gastos por cada bloco básico — considerando a descrição do *pipeline* do processador em questão. **6)** Encontrar correspondências entre o código-fonte C e o código binário. **7)** Inserir código para a geração de ciclos (análise estática) que contará os ciclos que os blocos básicos gastarão para serem executados. **8)** Finalmente, inserir código para a correção dinâmica da geração de ciclo. A correção dinâmica é necessária porque não é possível determinar estaticamente todos os impactos da arquitetura na quantidade de ciclos gastos (por exemplo, *cache hit\miss* e previsão de desvio exercem grande influência).

Para testar a velocidade de execução e a precisão do código traduzido, dois filtros e dois programas partes de rotinas de decodificação de áudio, foram compilados com um compilador C em objetos para o processador *Infineon Tricore*, e também para gerar código SystemC anotado. Como referências para medidas foram utilizados o Tricore TC20GP *evaluation board* e o simulador do conjunto de instruções (ISS) Tricore. Foram utilizados 2 tipos de anotações no código SystemC: “1”- que gera os ciclos depois da execução de cada bloco básico e “2” - que além disso, adiciona ciclos ao contador de ciclos quando for necessário (há comunicação com HW). Na comparação da velocidade de execução dos códigos SystemC com processador Tricore, o tipo “1” foi cerca dez vezes mais lento, enquanto o tipo “2” apresentou variações (de cerca de 4 vezes mais lento para cerca de 4 vezes mais rápido). E ao comparar a velocidade de execução do SystemC com a velocidade de execução de um ISS convencional, a abordagem proposta pelos autores apresentou melhora de até 91% na velocidade. Além disso, também foram realizadas comparações da precisão de ciclo (*cycle-accuracy*): o desvio do contador de ciclos dos programas traduzidos (considerando questões de memória e previsão de desvio) em relação ao obtido pelo TC20GP *evaluation bord* variou de 4 a 7% e foi praticamente a mesma variação com relação ao ISS convencional.

Assim, o artigo evidencia que a abordagem é válida e promissora, pois os resultados mostraram uma melhora no desempenho e uma boa precisão do modelo de software embarcado temporizado (*timed*). Uma das vantagens é que a abordagem possibilita uma execução rápida do código anotado, além do código SystemC gerado poder ser utilizado em ambientes de simulação SystemC. A desvantagem\dificuldade fica por conta da análise da correspondência entre o código binário e o código-fonte C, pois ela está sujeita às modificações de estrutura e otimizações feitas pelo compilador, podendo ser necessário utilizar técnicas de recompilação para conseguir identificar essas correspondências.