Nome: Glauber Móddo Cabral RA: 069271

Resumo de artigo

UMEZAWA, Y.; SHIMIZU, T. *A formal verification methodology for checking data integrity*. In: DATE '05: Proceedings of the conference on Design, Automation and Test in Europe. Washington, DC, USA: IEEE Computer Society, 2005. p. 284–289. ISBN 0-7695-2288-2.

A verificação formal de *chips* exige muitos recursos computacionais, podendo falhar pela falta dos mesmos. A verificação lógica, embora muito utilizada nos últimos tempos, tornou-se difícil de ser completada devido à complexidade dos chips; como solução, têm sido propostas metodologias híbridas para a verificação dos mesmos. Nestas soluções, um ponto crítico é a seleção do que será verificado por cada uma das técnicas. Propôs-se considerar três requisitos nesta decisão: (1) propriedades definidas formalmente são difíceis de verificar com técnicas lógicas; (2) a complexidade das propriedades é condizente com a suportada pelas ferramentas de verificação formal, sendo que a descrição das mesmas não deve tomar muito tempo na verificação do modelo; (3) a metodologia de verificação formal deve ser bem documentada e compartilhada pelas equipes de forma sistemática durante o projeto.

Aplicou-se a metodologia no projeto de *chips* para servidores, com altos requisitos de disponibilidade e confiabilidade. Os componentes internos destes elementos são protegidos por bits de paridade e detecção de estados internos inválidos; os erros nestes mecanismos devem ser detectados e reportados. Como os pontos de verificação são muitos, a simulação lógica tomaria muito tempo; optou-se, então, por verificar a integridade dos dados no circuito. As propriedades de integridade de dados são simples e facilmente descritas, podendo ser subdivididas em módulos sempre que forem grandes demais para as ferramentas. Por serem voltadas a dados, estas propriedades precisam ser exaustivamente testadas e a verificação formal comporta as combinações de estados a serem checados.

A verificação da integridade de dados baseou-se em três propriedades: (1) *habilidade de detecção de erros*, devendo-se verificar se todos os valores inválidos são detectados e reportados; (2) *validade do estado interno*, onde deve-se checar se a integridade do estado interno dos componentes se mantém quando a integridade das entradas é válida; (3) *integridade dos dados de saída*, propriedade que verifica se a integridade dos dados nas saídas dos componentes é mantida quando a integridade dos dados de entrada é garantida. Essa nova abordagem permitiu a aplicação da metodologia de forma sistemática durante o projeto do chip.

Adotou-se o seguinte fluxo de projeto. Os projetistas de lógica produziriam código RTL verificável, em Verilog, com métodos simples de injeção de erros em cada ponto de verificação de integridade. Os métodos deveriam ser bem definidos em relação às portas de entrada e controlados de forma independente para cada entidade a ser testada. Os mesmos projetistas gerariam a especificação da integridade dos dados em códigos PSL, linguagem amplamente suportada pelas ferramentas de verificação, ou em texto puro.

Os engenheiros de verificação formal deveriam utilizar os códigos PSL ou produzí-los de acordo com as especificações em texto puro, sem utilizar características da linguagem dependentes de ferramentas. De posse dos códigos, deveriam executar a verificação do modelo; caso as propriedades fosem grandes demais, deveriam dividi-las, manualmente, em propriedades menores verificáveis pelas ferramentas. Os resultados da verificação do modelo seriam reportados aos projetistas de lógica para correções no projeto.

Utilizou-se duas ferramentas de verificação formal, tendo sido descritas mais de 2 mil propriedades em linguagem PSL que demoravam cerca de 20 horas para serem verificadas. Foram encontrados 7 erros lógicos, dos quais quatro dificilmente seriam detectados por simulação lógica, embora facilmente encontrados pela verificação formal.

A implementação de injeção de erros nos códigos PSL implicou, basicamente, na inserção de um multiplexador no *chip*. Esta alteração aumentou a área do mesmo em menos de 2% e implicou em uma perda de apenas 4% do tempo total de execução, não causando impactos consideráveis na performance final do *chip*. Obteve-se, ainda, significativo ganho em produtividade.