Uma Metodologia para Suporte ao Ensino de Projeto de Circuitos Integrados Apoiada em Ferramentas de Código Aberto

Karina R. G. da Silva[1], Cássio L. Rodrigues[2], Elmar U. K. Melcher[3]

[1]Escola de Engenharia Elétrica e de Computação –
Universidade Federal de Goiás (UFG) – Goiânia, GO – Brasil

[2]Instituto de Informática –
Universidade Federal de Goiás (UFG) – Goiânia, GO – Brasil

[3]Departamento de Sistemas e Computação –
Universidade Federal de Campina Grande (UFCG) – Campina Grande, PB – Brasil

karinarg@eee.ufg.br, cassio@inf.ufg.br, elmar@dsc.ufcg.edu.br

Abstract. *This paper presents a methodology to support the teaching of integrated circuit design. With this methodology, the teaching evolutes gradually, without an abrupt contact with the low level of programming required in IC design. Experiments showed a 30% gain in student achievement in relation to a traditional methodology.*

Resumo. *Este artigo apresenta uma metodologia para dar suporte ao ensino de projetos de circuitos integrados. Nesta metodologia, o ensino evolui de forma gradual, sem um contato brusco com a programação de baixo nível exigida nos projeto de circuitos integrados. Experimentos mostraram um ganho de 30% no rendimento dos alunos em relação a uma metodologia tradicional.*

1. Introdução

A microeletrônica no Brasil possui uma história de surgimento, declínio e ressurgimento. De acordo com [Filho 2009], em 1950 se iniciou a fabricação de transistores e diodo para atender ao mercado de consumo e se desenvolveu a tecnologia para a fabricação de Germânio. Na década de 60 o governo já havia percebido a importância da microeletrônica para o Brasil e investiu em Pesquisa e desenvolvimento. Além disso, se iniciaram vários desenvolvimentos na área, tais como montagem de componentes semicondutores por filiais de empresas transnacionais, o Banco Nacional de Desenvolvimento (BNDES) estimulou a criação de laboratório de pesquisas, se criou a lei da informática, houve o surgimento de empresas da área de eletrônica, entre outros.

Atualmente o setor de microeletrônica vem ressurgindo, com a tentativa de definir os componentes semicondutores como um dos quatro setores prioritários para a política industrial [Filho 2009]. O Governo Brasileiro vem investindo fortemente no setor de Microeletrônica, através do Programa CI-Brasil, do Ministério da Ciência e Tecnologia, como pode ser visto em [Ci-brasil].

A falta de recursos humanos com formação na área de microeletrônica é um dos principais entraves ao desenvolvimento deste setor. Isto porque durante muito tempo o desenvolvimento de Circuitos Integrados (CIs) não foi um conhecimento prioritário nas disciplinas ministradas nas universidades. Como conseqüência, temos também deficiências de metodologias e ferramentas que dão suporte ao professor e ao estudante nessa área. Em resposta a estas adversidades, um dos programas que vem investindo no sentido de formar e constituir conhecimento na área de projetos de circuitos digitais é o programa Brazil-IP [Brazil-ip]. Esse programa tem como objetivo principal a capacitação de pessoas para atuação na cadeia de produção de circuitos integrados.

Do ponto de vista acadêmico, uma das grandes dificuldades no ensino de projeto de CI é fazer com que o aluno se familiarize com o baixo nível de abstração em que os CIs operam. Para desenvolver um CI, o projeto do mesmo é desenvolvido a partir de Linguagens de Descrição de Hardware (HDLs, *Hardware Description Languages)*, tais como SystemC e SystemVerilog . Uma vez pronto, o projeto em HDL é sintetizado em um código de mais baixo nível por meio de ferramentas. No entanto, para que esta síntese seja possível, o dispositivo sendo implementando deve estar no nível denominado de RTL (Register transfer Level). Nesse nível de abstração, é necessário que a programação seja realizada usando máquina de estados, relógio (clock) para sincronizar as funções e sinais para a comunicação de todos os módulos que compõe o sistema. Essa não é uma forma usual de programação na área de informática.

Considerando-se a dificuldade imposta pelo nível RTL em que o projeto de um CI é descrito, este artigo apresenta uma metodologia para apoiar o ensino de projeto de CIs. Nesta metodologia, o ensino evolui de forma gradual, sem um contato brusco com a programação de mais baixo nível. O conceito chave deste contato gradual é a adoção da verificação funcional como ponto de partida do projeto. A verificação funcional visa assegurar que projeto do CI está de acordo com as suas especificações. Com a metodologia em questão, denominada de VeriSC [Silva 2007], primeiro o aluno trabalha e amadurece o ambiente de verificação funcional do CI, para em seguida trabalhar com RTL do CI propriamente.

Por não precisar ser sintetizado, o ambiente de verificação é desenvolvido em um nível de abstração mais alto, chamado de nível de transação. O nível abstração da transação é aquele permitido pela linguagem de programação adotada no projeto. O CI opera em nível de abstração mais baixo, chamado de nível de sinais. A idéia da verificação funcional é fornecer estímulos para um Modelo de Referência e para o projeto RTL do CI e comparar se os resultados produzidos são iguais. Como o Modelo de Referência, geradores de estímulos e comparadores trabalham com transações e o projeto RTL do CI trabalha com sinais, o ambiente de verificação também possui elementos responsáveis por traduzir informações do nível de transação para o nível de sinais e vice-versa.

Da construção do ambiente de verificação funcional para o projeto em RTL, o aluno desenvolve gradualmente os elementos que traduzem a informação de mais alto nível do ambiente de verificação para a informação de mais baixo nível exigida pelo projeto RTL do CI. A vantagem desta abordagem é permitir ao aluno focar o seu raciocínio e sua reflexão primeiro nas funcionalidades do CI, sendo que para isso ele tem a sua disposição ferramentas e linguagens de mais alto nível abstração, tais como C e C++. Uma vez que o aluno tem um entendimento destas funcionalidades a partir da construção de um modelo de referência, o aluno pode focar o raciocínio na comunicação com o CI, que requer uma mudança de nível de abstração. Nesta mudança de nível de abstração, o aluno pode resolver também alguns problemas relacionados à sincronização com outros módulos. Por último, depois de ter um entendimento das funcionalidades do CI e das entradas de baixo nível deste CI, a programação em RTL do mesmo se inicia. Desta forma, a metodologia evita o contato direto com a programação em RTL, com uso de máquina de estados e relógio (clock) para sincronizar a comunicação dos módulos que compõe o sistema.

O restante deste artigo está dividido em três seções. Na Seção 2 é explicada a metodologia proposta; a Seção 3 é dedicada à apresentação de informações da aplicação da metodologia; os trabalhos futuros e considerações finais são discutidos na Seção 4.

2. O Projeto de um CI e a Metodologia Proposta

Um projeto de Circuito Digital pode ser dividido em fases, em que cada uma delas tem um nível de abstração diferente. Essas fases podem ser definidas como mostrada na Figura 1. Na especificação do hardware e especificação da verificação funcional, são gerados documentos para a fase de implementação e a fase posterior de verificação. A próxima fase é a implementação do *testbench*, que é o ambiente de verificação funcional do projeto em RTL do CI, doravante designado por DUV (*Design Under Verification*). A próxima fase é a de implementação do DUV,

que é implementado no nível de registradores. Em seguida, vem a fase de verificação funcional, que é fase onde são conduzidos os testes no DUV, através de inserção de estímulos e simulação. A fase de síntese é a transformação do RTL em *netlists*. Na simulação pós-síntese são realizadas simulações para identificar se foram inseridos erros na fase de síntese. Em seguida, os dados são prototipados na arquitetura alvo.

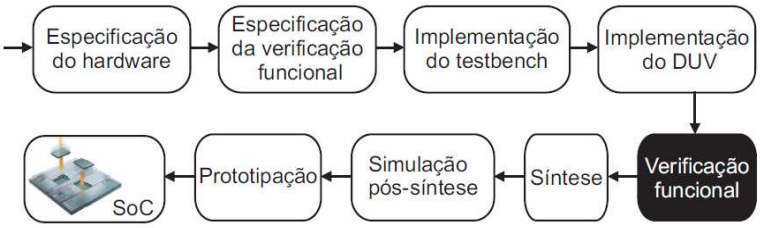

Figura 1: Fases de desenvolvimento de um CI.

O *testbench* da metodologia VerisC pode ser descrito como mostrado na Figura 2. O Source é responsável por gerar estímulos para o sistema. O Modelo de Referência implementa as funcionalidades do DUV no nível o de transações. O TDriver obtém as informações geradas pelo Source, faz a tradução para o nível de sinais e executa o protocolo de comunicação entre Source e DUV. O TMonitor converte os dados de sinais para o nível de transação e executa a comunicação entre o DUV e o Checker. O Checker compara se os resultados produzidos pelo DUV e pelo Modelo de Referência são iguais. O módulo denominado de DUV é o próprio dispositivo sendo implementado, e ele pode ser tanto o projeto inteiro, como parte do projeto sendo implementado.

A parte cinza na Figura 2 é o *testbench*. Esse *testbench* é implementado no início do projeto e fica pronto para receber o DUV antes mesmo de sua implementação. Isto facilita a depuração do DUV, uma vez que o ambiente de testes já pode ser usado desde a primeira versão do DUV. Outra vantagem é que o estudante não precisa descer no nível baixo de forma de onda para captar os erros, pois esse ambiente de testes faz a comparação dos dados automaticamente e mostra se os dados estão certos ou errados. Mais detalhes da metodologia podem ser obtidos em [silva 2007].

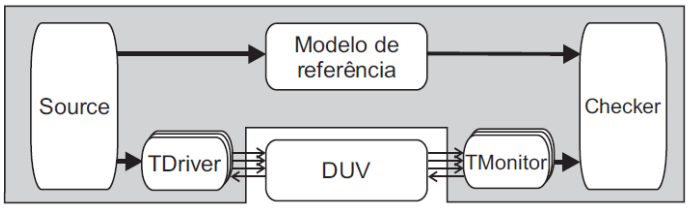

Figura 2: Organização do testebench.

O alto custo das ferramentas existentes na indústria pode ser outro entrave no ensino de projeto de CIs. Para contornar esta dificuldade, a metodologia proposta possui suporte ferramental apoiado em ferramentas de código aberto. Por exemplo, existe uma ferramenta para gerar parte do *testbench* automaticamente para o estudante. Essa ferramenta é denominada eTBc e pode ser baixada gratuitamente de [LAD]. A ferramenta eTBc gera grande parte do código. Ela recebe um arquivo denominado de TLN que define as interfaces de comunicação entre os módulos e em seguida gera o código.

3. Aplicação da Metodologia

A metodologia já foi usada em vários projetos acadêmicos e não acadêmicos. A metodologia VeriSC e a ferramenta eTBc foram utilizadas na verificação de um decodificador de vídeo MPEG4 [Brazil-ip]. A metodologia foi empregada também para a verificação de um decodificador de áudio MP3 e um processador 8051, além de outros projetos parceiros, desenvolvidos em universidades federais [Brazil-ip].

Outros resultados foram obtidos com testes realizados com estudantes em cursos de verificação. O experimento foi realizado com estudantes como mesmo grau de conhecimento e a mesma experiência. Os estudantes tinham pouca experiência em desenvolvimento Orientado a Objetos e quase nenhuma experiência com a linguagem C++. Devido aos estudantes terem pouca experiência com hardware e não terem utilizado nenhuma outra metodologia de verificação anteriormente, foi utilizado um simples conversor PCM/DPCM, que consiste em calcular a diferença de duas amostras de áudio digital subseqüentes, realizando a operação de saturação no resultado dessa diferença.

O primeiro grupo de alunos que realizou o curso de verificação, empregou uma metodologia tradicional para verificar o seu projeto. Esse grupo não fez um *testbench* hierárquico, eles implementaram somente um *testbench* depois de haver implementado o dispositivo RTL em um dos blocos do DUV. O segundo grupo de alunos, realizou o segundo curso ministrado utilizando a metodologia VeriSC e implementou 3 *testbenches*. Um *testbench* para o DUV completo, um *testbench* para o bloco diferença e um *testbench* para o bloco de saturação. Eles empregaram a abordagem hierárquica da metodologia para a realização dessa verificação funcional.

Os resultados experimentais indicaram que houve um aumento de 30% da produtividade com os estudantes que utilizaram a metodologia VeriSC e empregaram a abordagem hierárquica. Observações realizadas com os estudantes durante o curso revelaram que essa melhor produtividade foi devida aos seguintes fatores:

- Elevado grau de reuso de código do *testbench* em VeriSC e o oposto na metodologia tradicional.
- Menos erros de compilação devido à abordagem incremental do VeriSC.
- Menos erros de execução, devido à mesma razão.
- Menos tempo para depurar o RTL, uma vez que o *testbench* já estava pronto antes mesmo do início da implementação do RTL.

4. Trabalhos Futuros

A metodologia proposta trabalha com a decomposição hierárquica do projeto, sendo que cada parte do projeto deve ser decomposto em blocos que possam ser usados para executar uma determinada funcionalidade de forma independente. Uma vez que esses módulos tenham sido implementados, é necessário que cada um deles seja verificado separadamente. Para esta decomposição, o suporte ferramental tem-se mostrado adequado. Como parte dos trabalhos em andamento e futuros, o suporte ferramental para ajudar o aluno na parte de composição dos blocos está sendo construído.

5. Referências

Brazil-ip, http://www.brazilip.org.br/. Acesso em 09.08.2010.

Filho, J. E. R. (2009), História da Microeletrônica no Brasil. Disponível em <http://www2.desenvolvimento.gov.br/arquivo/publicacoes/sti/indBraOpoDesafios/coletanea/Semicondutores/Ripper.pdf>.

Ci-brasil, http://www.ci-brasil.gov.br/. Acesso em 09.08.2010.

Silva, K., R., G., Melcher, E. U. K., Maia, I. and Cunha, H. do N. (2007) A methodology aimed at better integration of functional verification and rtl design. *Design Automation for Embedded Systems*, 10(4):285–298.

LAD, http://www.lad.dsc.ufcg.edu.br. Acesso em 09.08.2010.