COM2AHDL: FERRAMENTA CAD DESENVOLVIDA PARA O ENSINO DE CIRCUITOS DIGITAIS

Alexandre César Rodrigues da Silva[1], Vanderley Balieiro Júnior [2] e Gracieli Sartório Cardoso [3]

Resumo *— A rápida evolução dos sistemas eletrônicos tem motivado o uso, cada vez mais, de ferramentas CAD no ensino de disciplinas envolvendo laboratórios. Na prototipagem rápida, empregando FPGA, torna-se imprescindível o uso de ferramentas de síntese. Muitas linguagens de descrição de hardware compõem os ambientes de projeto disponibilizados para as Universidades através de Programas Universitários. O objetivo deste trabalho foi habilitar estudantes da disciplina Circuitos Digitais I, ministrada na Faculdade de Engenharia de Ilha Solteira, a projetar e desenvolver ferramentas computacionais que pudessem ser utilizadas em conjunto com os ambientes de projeto comerciais. Uma das ferramentas desenvolvidas, denominada COM2AHDL, gera um modelo AHDL para circuitos combinacionais, cuja descrição inicial é feita através da Tabela Verdade que descreve o comportamento do circuito. Esta iniciativa, além de permitir a sinergia entre disciplinas, possibilitou que alunos desenvolvessem suas próprias ferramentas para gerar modelos em alto nível de abstração e com interface para ferramentas comerciais.*

Palavras chave — Ferramentas CAD, Linguagens de Descrição de Hardware, AHDL, Síntese lógica .

INTRODUÇÃO

A rápida evolução da indústria de microeletrônica tem desafiado os atuais currículos acadêmicos em manterem-se adequados de modo a fornecer os conhecimentos necessários aos seus alunos. Os laboratórios tradicionais, dos cursos de graduação em circuitos digitais, têm utilizados software de simulações para o auxílio no projeto e modelagem de pequenos sistemas digitais clássicos. Esses modelos de sistemas têm sido utilizados para explanar ou reforçar muitos dos conceitos e idéias contidas em livros textos. Na descrição desses modelos empregava-se, até então, a captura de esquemático.

Atualmente, muitos desses cursos estão se alicerçando com base no uso da tecnologia de FPGA (Field Programmable Gate Array) e nos ambientes das avançadas ferramentas CAD (Computer Aided Design).

Estes sofisticados ambientes de síntese lógica estão sendo disponibilizados para as universidades através de programas universitários. Dessa forma, tem-se a oportunidade de introduzir os estudantes no uso de HDLs (linguagens de descrição de hardware) e na metodologia de projetos hierárquico (top-down e button-up).

Neste artigo descreve-se a experiência desenvolvida no Curso de Circuitos Digitais I ministrado na Faculdade de Engenharia de Ilha Solteira – UNESP, onde os alunos foram motivados a desenvolverem suas próprias ferramentas para serem utilizadas em conjunto com os sofisticados ambientes comerciais.

Uma dessas ferramentas, denominada COM2AHDL (Combinacional para AHDL), traduz a especificação contida na tabela verdade, descrevendo o comportamento de um circuito combinacional, para um modelo na linguagem de descrição AHDL (Altera Hardware Description Language). Essa iniciativa, além de permitir a sinergia entre disciplinas, possibilitou que alunos desenvolvessem suas próprias ferramentas computacionais para gerarem modelos em alto nível de abstração integráveis com o ambiente comercial Max+Plus II da Altera.

Através do arquivo "report file", gerado no processo de compilação do modelo, os estudantes podem extrair informações relevantes para o entendimento da teoria de circuitos digitais.

METODOLOGIA DE PROJETO

A metodologia de projeto utilizada consiste, basicamente, em 4 passos, sendo que utilizou-se dois conjuntos de ferramentas, a COM2AHDL e a TAB2VHDL [1] em conjunto com o ambiente Max+Plus II. Os passos são os seguintes:

- **Descrição**: Faz-se uma especificação abrangente de como o sistema projetado deve funcionar. Essa descrição deve ser feita, tanto quanto possível, através de linguagens de descrição de hardware. Neste estágio de projeto já é possível efetuar simulações com o objetivo de detectar, antecipadamente, erros de projetos ou má interpretação da especificação inicial. O trabalho humano torna-se mais racional quando emprega-se o diagrama de transição de estados (circuitos seqüenciais) ou a tabela verdade (circuitos combinacionais);
- **Implementação**: O objetivo é decidir sobre a estrutura da implementação e quais blocos funcionais serão utilizados. Isto requer a partição do projeto em blocos menores cujo comportamento pode ser descrito em

[1] Alexandre César Rodrigues da Silva, DEE – FEIS - UNESP, Av. Brasil, 56, 15.385-000, Ilha Solteira, SP, Brasil, acrsilva@dee.feis.unesp.br
[2] Vanderley Balieiro Júnior, DEE – FEIS - UNESP, Av. Brasil, 56, 15.385-000, Ilha Solteira, SP, Brasil, balieiro@dee.feis.unesp.br
[3] Gracieli Sartório Cardoso, DEE – FEIS - UNESP, Av. Brasil, 56, 15.385-000, Ilha Solteira, SP, Brasil, gracieli@dee.feis.unesp.br

termos de hardware. Esta ação resulta em duas espécies de blocos: os blocos disponíveis em dispositivos físicos e os blocos que o próprio projetista necessita descrever. Quando o projeto é implementado, novamente utiliza-se a simulação para verificar se erros não foram introduzidos nesse processo;

- **Síntese**: A síntese do sistema projetado é feita através de um programa que lê a especificação, descrita em alguma linguagem de descrição ou através de esquemático, e produz um *netlist* otimizado e mapeado dentro da tecnologia destino. As ferramentas de síntese, em geral, fornecem arquivos de relatório contendo parâmetros como tempo de compilação, quantidade de células utilizadas, tempo de atrasos, funções booleanas sintetizadas, e muitas outras informações;
- **Layout**: O layout é também conhecido por *Place* e *Route*. Este passo produz uma descrição exata do dispositivo físico final. A descrição do layout diz exatamente como as partes do dispositivo são usadas e como os fios são conectados dentro do dispositivo.

A ferramenta apresentada nesse trabalho insere-se na fase de descrição do projeto onde, a partir da tabela verdade que descreve o comportamento do circuito combinacional, gera-se um modelo AHDL, a partir do qual o projeto pode ser sintetizado por ferramentas de síntese comercial.

A ferramenta COM2AHDL gera, automaticamente, uma descrição AHDL que seria cansativo quando escrita a mão pelo projetista. Além disso, o arquivo de entrada serve como documentação do projeto. A Figura 1 apresenta a tela inicial do programa COM2AHDL.

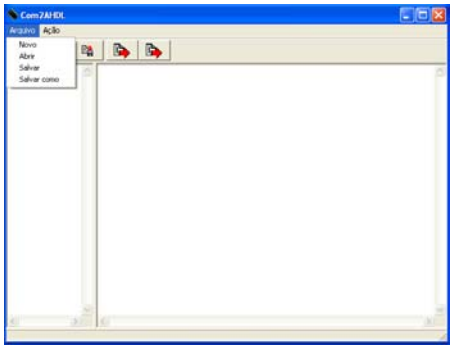

FIGURA. 1
TELA INCIAL DO PROGRAMA COM2AHDL.

O COM2AHDL foi inicialmente programado na linguagem C e posteriormente transcrito em C^{++} Builder oferecendo, dessa forma, uma interface mais amigável com o estudante.

FERRAMENTA COM2AHDL

Um simples exemplo de um circuito codificador BCD8421 para 7 segmentos é utilizado para ilustrar o emprego da ferramenta desenvolvida. O circuito contém um barramento de entrada denominado BCD8421 de dimensão 4, ou seja, pode-se representar essa entrada na forma de vetor como BCD8421[3..0], onde o bit de posição 0 representa o menos significativo e o de posição 3 o mais significativo. Do mesmo modo, o circuito tem um barramento de saída, denominado SEGMENTO que na forma de vetor é represento por SEGMENTO[6..0], ou seja, consiste em um barramento de dimensão 7. A Figura 2 apresenta o esquemático que representa o circuito codificador.

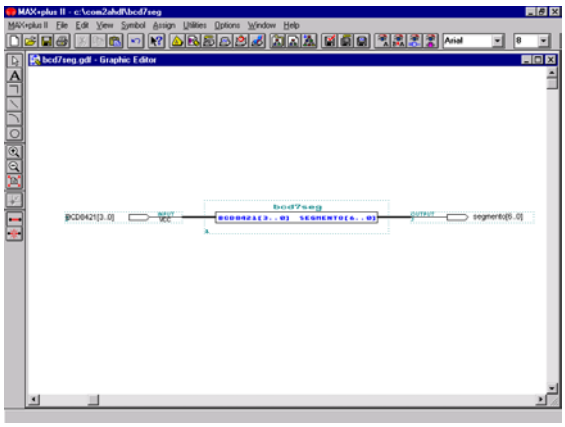

FIGURA. 2
ESQUEMÁTICO DO CODIFICADOR BCD8421 PARA 7 SEGMENTOS.

No programa COM2AHDL transcreve-se, na janela da esquerda, a tabela verdade que descreve o funcionamento do circuito a ser projetado e, após selecionar a opção adequada, é gerado, na janela da direita, uma descrição em AHDL que está pronta para ser sintetizada no ambiente Max + Plus II. A Figura 3 apresenta a descrição do codificador BCD8421 para 7 segmentos e o modelo AHDL gerado.

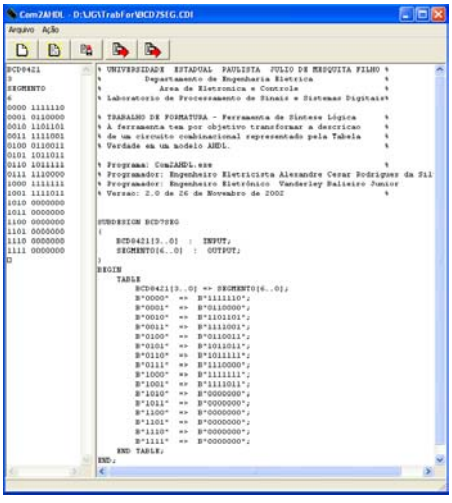

FIGURA. 3
TABELA VERDADE DO BCD8421 PARA 7 SEGMENTOS E O MODELO AHDL.

Com o modelo AHDL gerado a ferramenta de síntese automaticamente executa a minimização das funções lógicas e a síntese lógica a nível de porta lógicas. Em geral, muitas das portas de nível intermediárias são compartilhadas entre as várias saídas do sistema. Pode-se notar que no modelo descrito na linguagem de descrição AHDL não se utilizou equações lógicas, apenas descreveu-se o comportamento que se espera do circuito.

O estudante pode também simular o modelo e analisar o resultado da síntese lógica. A Figura 4 apresenta o modelo em estudo como entrada do ambiente de projeto Max + Plus II.

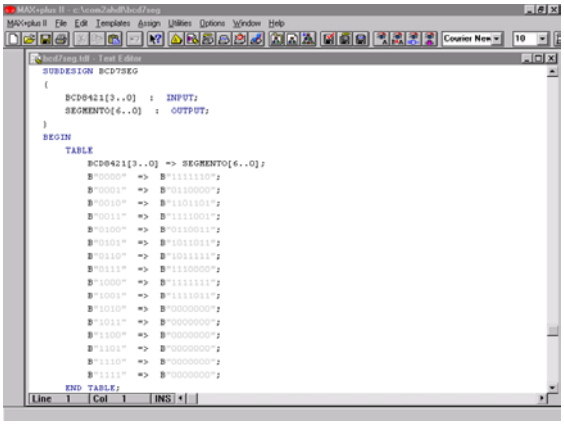

FIGURA. 4
MODELO AHDL DO BCD8421 PARA 7 SEGMENTOS NO MAX+PLUS II.

Para se verificar o perfeito funcionamento do modelo, este pode ser simulado. A Figura 5 apresenta o resultado da simulação.

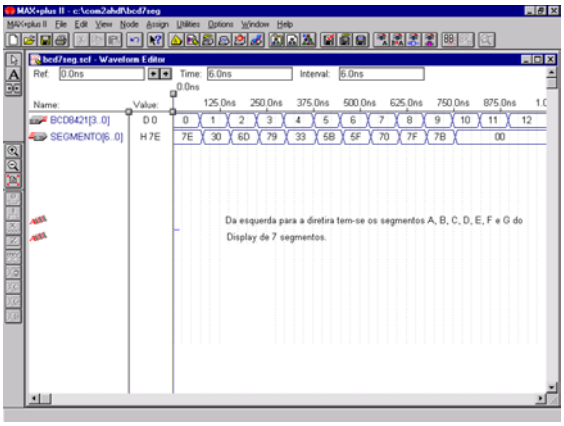

FIGURA. 5
RESULTADO DA SIMULAÇÃO DO CODIFICADOR BCD8421 PARA 7 SEGMENTOS NO AMBIENTE MAX+PLUS II.

Pode-se notar, pelo resultado da simulação apresentado na Figura 5, que o modelo funcionou como o previsto. No barramento de saída, representado na notação hexadecimal, foram acionados os segmentos correspondentes aos números BCD8421 tidos no barramento de entrada em cada instante de tempo.

Da análise do arquivo "report file", gerado pelo ambiente de síntese, o aluno pode extrair as funções combinacionais de cada função de saída. Essas funções, por terem sido otimizadas por algoritmos de minimização, são apresentadas na sua forma mínima. A Figura 6 apresenta parte do arquivo de relatório onde pode-se notar que a função que representa o SEGMENTO1 é dada por:

SEGMENTO1 = LCELL(_EQ002 $!BCD84211);
_EQ002 = !BCD84210 & BCD84211 & BCD84212 &
!BCD84213 # BCD84210 & !BCD84211 & !BCD84212 &
!BCD84213 # !BCD84211 & BCD84212 & BCD84213;

Nesta representação o símbolo & representa a operação AND, o símbolo # representa a operação OR e o símbolo ! representa a operação NOT. Note que BCD8421n, com n variando entre 0 e 3, representam as variáveis do sistema, ou seja, os sinais de entrada.

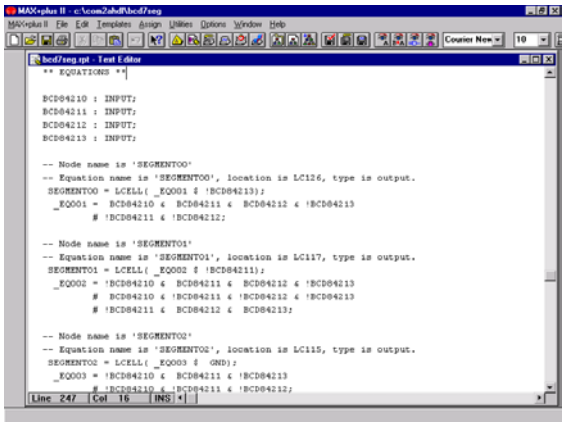

FIGURA. 6
ARQUIVO DE RELATÓRIO APRESENTANDO AS FUNÇÕES COMBINACIONAIS DAS FUNÇÕES DE SAÍDAS.

A LINGUAGEM DE DESCRIÇÃO AHDL

A AHDL (Altera Hardware Description Language) é uma linguagem modular de alto nível empregada para se modelar lógica combinacional complexa, operações em grupos, máquinas de estados e tabelas verdades. A especificação é feita através de uma entrada textual que, posteriormente, é compilada e simulada, podendo então ser utilizada para programar componentes FPGA da Altera. Esta linguagem permite que os projetistas criem projetos hierárquicos que podem incorporar outros arquivos de projeto. A representação simbólica de um projeto AHDL é criado automaticamente no processo de compilação e síntese que pode ser incorporado dentro de uma descrição gráfica (captura de esquemático).

Um modelo AHDL contém partes denominadas de "Sections" e "Statements". Todo modelo deve conter uma *Design Section* ou uma *Subdesign Section* combinada com uma *Logic Section*. Algumas *Sections* e *Statements* são opcionais. Descreve-se a seguir as seções e declarações aceitas no AHLD:

- **Title Statements** (opcional): gera comentários para o arquivo de relatório (report file) gerado pelo compilador;
- **Constant Statements** (opcional): especifica um nome simbólico que pode ser substituído por uma constante;
- **Function Prototype Statements** (opcional): declara os portos de uma macrofunção ou primitiva e a ordem em que estes portos devem ser declarados na linha de referência;
- **Define Statements** (opcional): define uma função matemática que retorna um valor que está baseado em um argumento;
- **Parameters Statements** (opcional): declara um ou mais parâmetros que controlam a implementação de uma função parametrizável. Um valor default pode ser especificado para cada parâmetro;
- **Design Sections** (necessário): especifica pinos, células lógicas, opções lógicas e atribuição de dispositivos. Também especifica quais os pinos estão ligados juntos na placa. Esta seção é necessária se for a única seção do modelo;
- **Assert Statements** (opcional): permite que o projetista teste condições de uma expressão qualquer e relate o resultado;
- **Subdesign Sections** (necessário): declara os portos de entradas, de saídas e os bidirecionais de um modelo AHDL. Essa seção é necessária a não ser que o modelo consista somente da seção Design;
- **Variable Sections** (opcional): declara uma variável que representa uma informação interna;
- **Logic Section** (necessário): define as operações lógicas do modelo. Essa seção é necessária, a menos que o modelo consista somente da seção Design.

Considere como exemplo o circuito lógico apresentado na Figura 7 na forma de esquemático. O modelo AHDL para o circuito pode ser descrito como:

```
SUBDESIGN  FUNÇÃO
(
    A0, A1, B          :  INPUT;
    Out1, Out2         :  OUTPUT;
)
BEGIN
   Out1  =  A1 & !A0;
   Out2  =  Out1 # B;
END;
```

A seção *Subdesign* descreve os portos de entradas e de saídas e especifica os seus nomes que podem ser referenciados. O *Subdesign* também tem um nome que deve ser o mesmo nome do arquivo contendo o modelo AHDL.

A seção *Logic*, que descreve as operações lógicas do circuito utiliza declarações de atribuições e expressões relacionais.

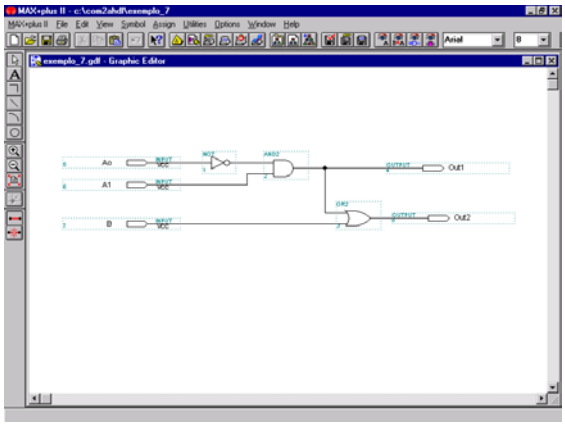

FIGURA. 7
CIRCUITO LÓGICO REPRESENTANDO UMA FUNÇÃO BOOLEANA.

A linguagem também disponibiliza a lógica condicional, onde é possível escolher diferentes comportamentos dependendo do valor de uma entrada lógica. Essa lógica condicional é implementada através das declarações IF e CASE.

A AHDL oferece uma grande facilidade em converter padrões de entradas em valores de saídas. Isto é feito pelo mapeamento das entradas em padrões de saídas através do uso da tabela verdade.

A declaração *Truth Table* contém um cabeçalho que descreve quais entradas são mapeadas para quais saídas e em que ordem. No exemplo apresentado a seguir o padrão de saída para todos os possíveis padrões da entrada, denominada IN[1..0], é descrito através de declaração *Truth Table*.

```
SUBDESIGN  DECODER
(
    IN[1..0]          :  INPUT;
    A, B, C, D        :  OUTPUT;
)
BEGIN
  TABLE
     IN[1..0]  =>  A,  B,  C,  D;
        H "0"  =>  1,  0,  0,  0;
        H "1"  =>  0,  1,  0,  0;
        H "2"  =>  0,  0,  1,  0;
        H "3"  =>  0,  0,  0,  1
  END TABLE;
END;
```

Utilizou-se da declaração *Truth Table* nos modelos gerados pelo programa COM2AHDL.

Dezenas de projetos de circuitos combinacionais, que são deixados como atividade para os alunos, foram implementados em FPGA utilizando-se a ferramenta desenvolvida. Em todos os projetos constatou-se o perfeito funcionamento, tanto através da simulação quanto em teste de bancada.

EXEMPLO DE PROJETO

Projeto 1: Um técnico de laboratório possui 4 produtos químicos (A, B, C e D) e 2 recipientes para guardá-los. É conveniente remover os produtos químicos de um recipiente para outro de tempos em tempos. É perigoso manter B e C juntos a menos que A também esteja presente. C e D não podem estar juntos com A. O projeto consiste em gerar um circuito lógico para indicar as situações perigosas de armazenamento dos produtos químicos [2].

A Figura 8 apresenta a descrição do projeto como entrada para a COM2AHDL e o modelo AHDL gerado. A simulação do modelo comprovou o seu perfeito funcionamento.

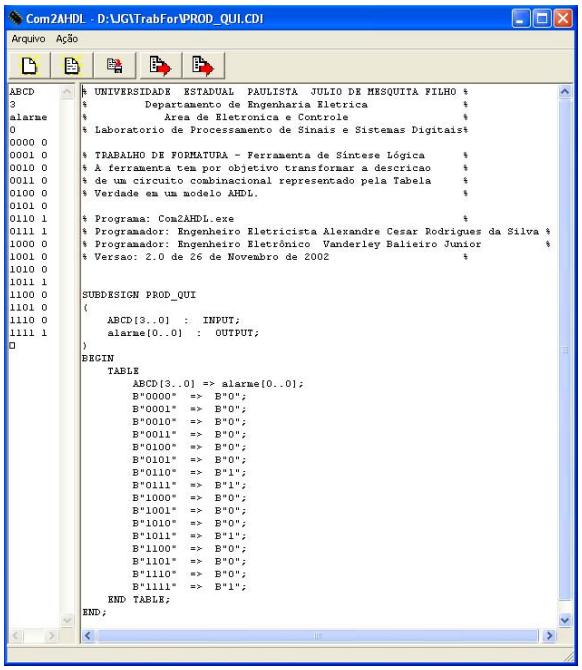

FIGURA. 8
TABELA VERDADE E O MODELO AHDL GERADO PELA FERRAMENTA COM2AHDL.

Uma outra ferramenta, que se utiliza dos mesmos passos de projeto apresentado neste trabalho, está sendo testada para gerar o modelo VHDL para um circuito combinacional.

CONCLUSÃO

Este trabalho apresentou uma ferramenta que, em conjunto com as disponíveis comercialmente, permite que o projeto de um circuito combinacional seja implementado em FPGA a partir da descrição da tabela verdade que descreve o funcionamento do mesmo.

A iniciativa do desenvolvimento desse trabalho surgiu da necessidade de alunos do curso de Circuitos Digitais I, ministrado na Faculdade de Engenharia de Ilha Solteira, criarem as suas próprias ferramentas para o auxílio ao desenvolvimento de projeto de circuito digitais.

A ferramenta desenvolvida permitiu a automatização do projeto de circuitos digitais e se tornou extremamente conveniente, pois retirou do projetista a tarefa da descrição do modelo. Dessa forma, cabe aos alunos apenas a análise crítica das soluções, deixando para a máquina as cansativas tarefas sujeito a erros.

AGRADECIMENTOS

Os autores agradecem o apoio recebido do PPGEE da Faculdade de Engenharia Elétrica de Ilha Solteira – UNESP, da CAPES e da P.I. Componentes.

REFERÊNCIAS

[1] L. O. Tancredo, "TAB2VHDL: Um Ambiente de Síntese Lógica para Máquinas de Estados Finitos", *Tese de Mestrado*, PPGEE – FEIS – UNESP, Setembro, 2002.

[2] I. Bonatti, M. Madureira, "Introdução à Análise e Síntese de Circuitos Lógicos", *Editora da Unicamp*, Campinas, 1990.

[3] Z. Salcic, A. Smailagic, "Digital Systems Design and Prototyping: Using Field Programmable Logic and Hardware Description Languages", Kluwer Academic Publishers, Second Edition, 2000.