ENSINO SOBRE ALOCAÇÃO DE ESTADOS EM FSM COM O EMPREGO DA AHDL

Marcos Ikeguchi Ohira[1], Fábio Pieroni Zanella[2] e Alexandre César Rodrigues da Silva[3]

Resumo *— Com o rápido aumento da complexidade dos circuitos e da densidade de integração dos CIs, as ferramentas automáticas de síntese tornaram-se necessárias para o projeto de circuitos digitais. Circuitos seqüenciais são modelados por máquinas de estados finitos (FSM) onde os estados são caracterizados somente por nomes simbólicos. Alocação de estado é o mapeamento do conjunto de estados (nomes simbólicos) de uma FSM para um conjunto de códigos binários com o objetivo de minimizar a área do circuito combinacional necessário para realizar a FSM. Comparou-se alguns algoritmos de alocação de estados utilizados no ambiente de projeto Max+ Plus II da Altera. Vários códigos de linhas foram modelados na linguagem de descrição de hardware AHDL e implementados em FPGA. Este trabalho está sendo desenvolvido junto ao curso de graduação do DEE-FEIS-UNESP na linha de pesquisa de sistemas digitais, com o objetivo de inicializar o aluno no uso de ferramentas de síntese digitais.*

Palavras-Chave — Máquinas de estados finitos, alocação de estado, circuitos combinacionais, AHDL, FPGA e ferramentas de síntese digitais.

INTRODUÇÃO

As máquinas de estados, também chamadas de máquinas seqüenciais, são sistemas que respondem em estados diferentes de acordo com um dado de entrada. Caso o sistema esteja em determinado estado, ele ficará no mesmo estado a não ser que o sinal de entrada, denominado de relógio, se altere [1].

Com relação às dependências de entrada, saída, estado atual e próximo estado, as máquinas podem ser classificadas em dois modelos: Mealy e Moore.

Uma máquina seqüencial é denominada máquina de Moore, se suas saídas forem dependentes apenas dos estados e não da entrada. Uma máquina seqüencial é denominada máquina de Mealy, se suas saídas forem dependentes tanto do estado atual como da entrada. Para ambos os modelos de máquinas, o próximo estado é dependente do estado atual e da entrada.

Uma outra característica muito importante das máquinas seqüenciais é a dependência de um clock. Em geral, uma referência de tempo é muito utilizada em sistemas digitais. Quanto à dependência de um clock, as máquinas seqüenciais podem ser classificadas como: síncronas e assíncronas.

As máquinas síncronas são máquinas cuja mudança de estado acontece de acordo com um clock.

Máquinas assíncronas mudam de estado independente de um clock de referência.

Além das diferenças citadas entre máquinas síncronas e assíncronas, na questão de implementação das duas máquinas, as síncronas são muito mais fáceis em relação às assíncronas.

Conhecidas as propriedades das máquinas de estados, um bom exemplo de máquina de estado é o flip-flop, este tem dois estados para uma entrada, em suma, é um sistema (ou máquina de estado) muito simples. Com uma combinação de flip-flops, pode-se obter sistemas extremamente complexos.

Alocação de Estados

As alocações de estados são métodos para dispor estados de máquina, dependendo da sua aplicação, em combinações diferentes.

Para cada tipo de aplicação existe uma alocação ótima e um tipo de preocupação diferente como, por exemplo, custo lógico do circuito e o comportamento do circuito para cada tipo de elementos de memória. Portanto, a escolha do tipo de alocação é muito importante e isso se dá somente sob tentativa dentre todos os algoritmos de alocação.

Antes de se escolher o tipo de alocação a ser utilizada deve-se, inicialmente, conhecer a quantidade de estados e quantidade de flip-flops necessários para o circuito.

Para uma máquina com n flip-flops a quantidade total de estados para essa máquina é representada pela equação 2^n, e o número mínimo de flip-flops necessários para representar os s estados da máquina é representado pela equação $\log_2 s$.

Tomar conhecimento das quantidades de estados e quantidade de flip-flops são informações iniciais para se determinar o algoritmo de alocação ótimo, porém, como já mencionado, para se determinar o melhor algoritmo de alocação deve-se utilizar todos os algoritmos e verificar o melhor. Como isso é um trabalho muito exaustivo, tem-se algumas considerações a serem feitas para se atingir o tipo de algoritmo com um tempo razoavelmente menor, como, por exemplo, determinar uma condição inicial para a máquina (preset ou reset), fazer as minimizações de estados, etc [2]-[3]-[6].

[1] Marcos Ikeguchi Ohira, UNESP – Ilha Solteira, Av. Brasil, 56, CEP: 15385-000, Ilha Solteira, SP, Brasil, ikeguchi@dee.feis.unesp.br
[2] Fábio Pieroni Zanella, UNESP – Ilha Solteira, Av. Brasil, 56, CEP: 15385-000, Ilha Solteira, SP, Brasil, zanella@dee.feis.unesp.br
[3] Alexandre César Rodrigues da Silva, UNESP – Ilha Solteira, Av. Brasil, 56, CEP: 15385-000, Ilha Solteira, SP, Brasil, acrsilva@dee.feis.unesp.br

© 2003 ICECE

Tipos de Algoritmos de Alocação de Estados

- **Simplest:** É o algoritmo de alocação mais simples, pois ele utiliza a quantidade de flip-flop mínima que é obtido da equação $\log_2 s$, sendo s o menor número inteiro obtido. Esse tipo de alocação é muito simples para ser utilizado, porém não é muito usual, pois o custo no circuito, principalmente em relação ao circuito de controle, fica muito alto.
- **Decomposed:** É um algoritmo mais apurado em relação ao Simplest, pois este tem o estado inicial em reset e cada um de seus bits pode representar uma operação da máquina. O bit mais significativo (MSB), após a máquina sair da condição de reset, é sempre "1", pois indica que a máquina está em operação, e os outros bits indicam os estados da máquina.
- **One-Hot:** O mais utilizado algoritmo de alocação. Este utiliza apenas um bit por estado de máquina, isso garante que o circuito de controle da máquina se torne muito mais simples, com isso diminui o custo do circuito. Utilizando o algoritmo One-Hot não é necessário um circuito de controle na sua saída [1]. A desvantagem que esse tipo de alocação apresenta ocorre quando se tem uma máquina com muitos estados, pois o circuito da máquina irá utilizar uma quantidade grande de flip-flops aumentando assim o custo do circuito. Este algoritmo também é usado pelo AHDL, que será apresentado a seguir.
- **Almost One-Hot:** Assim como o nome já diz, esse algoritmo de alocação é uma variação do One-Hot, pois ao contrário do One-Hot, o estado inicial do Almost One-Hot é o reset, isso apresenta duas vantagens, pois diminui um flip-flop no circuito, fazendo com que o custo do circuito diminua e facilite a inicialização do circuito, pois é mais simples iniciar uma máquina à partir do reset [3].

Altera Hardware Description Language (AHDL)

A AHDL é uma linguagem muito potente, versátil e de fácil utilização, pois permite descrever um projeto de várias formas, como, por exemplo, na forma hierárquica com compartilhamento entre arquivos.

Como se trata de uma linguagem, todo o circuito, ou projeto, são feitos de forma descritiva que pode ser feita no próprio editor de texto do software MAX+plus II [4], ou em qualquer outro editor de texto, desde que utilize o código ASCII. Contudo, para que se possa simular o circuito descrito, deve-se compilar o arquivo texto, para isso utiliza-se o compilador do software MAX+plus II. Em face disso, a utilização do editor de texto existente no próprio software é muito vantajosa, pois no mesmo software é possível escrever o circuito, compilar e fazer as simulações.

Além de todas essas vantagens, um circuito implementado em AHDL, pode ser transformado em um símbolo para ser utilizado na forma esquemática (.gdf) [4].

Implementação de Máquinas de Estado em AHDL

Uma máquina de estados pode ser facilmente implementada utilizando a AHDL, pois pode utilizar equações Booleanas ou tabelas verdade para fazer a implementação, sendo que as alocações podem ser feitas tanto pelo próprio usuário quanto pelo software.

Para se implementar uma máquina de estados deve-se seguir os seguintes passos:

- Construir uma tabela de próximo estado;
- Declarar a máquina de estados (na estrutura Variable Section);
- Adicionar a tabela verdade (ou de próximo estado), ou se for o caso, as equações booleanas (estrutura Logic Section).

Esses passos são seguidos para que o software faça as alocações. Se o usuário for fazê-las, a única diferença é que se deve declarar os estados e suas alocações.

As máquinas de estados utilizam flip-flops como elemento de memória e para serem implementadas em AHDL é necessário que o clock, reset e enable sejam devidamente designados, visto que o flip-flop depende desses parâmetros para funcionamento, ou seja, a entrada de clock, enable e reset devem ser devidamente declaradas quando necessário, caso contrário, a máquina não funcionará de maneira correta.

Como já mencionado, a implementação de uma máquina de estado pode ser feita de duas formas: alocação direta, na qual o próprio usuário faz as alocações, e a indireta, na qual o software faz as alocações.

Nas alocações indiretas o software faz todas as minimizações de estados necessárias e implementa um tipo de alocação de estado que são basicamente dois tipos, o Simplest e o One-Hot. A escolha do algoritmo utilizado é ditada pela família de FPGA (*Field Programable Gate Array*) utilizado. Para FPGA's da família MAX, o software, por definição, faz a alocação do tipo Simplest e, também, tem-se a possibilidade da utilização do algoritmo One-Hot, através da seleção da opção One-Hot State Machine Encoding. Para os FPGA's da família FLEX a alocação utilizada é o One-Hot, independentemente da opção One-Hot State Machine Encoding [5].

Quanto ao tipo de máquina a ser implementada (Mealy ou Moore) as mudanças são muito pequenas, devido à versatilidade da linguagem [4].

METODOLOGIA

Neste trabalho estudou-se métodos de alocação de estados e a linguagem de descrição de hardware (AHDL), que junto ao docente da área de sistemas digitais do DEE-FEIS-UNESP, está sendo disseminada nas disciplinas de Circuitos Digitais e Projetos de Sistemas Digitais.

Essas duas teorias (alocação e AHDL) estão sendo ensinadas aos alunos de graduação com o intuito de motivar o aprendizado, pois, mostra de uma maneira prática, que um mesmo circuito digital, mais precisamente máquinas de

estados finitas síncronas, podem ter variados comportamentos, dependendo de sua alocação de estados, e com a linguagem AHDL os alunos aprendem que um circuito digital pode, também, ser implementado em uma linguagem de alto nível.

Como estudo inicial utilizou-se diferentes técnicas de alocação de estado (Simplest, One-Hot e Almost-One-Hot), cujos códigos foram implementados na linguagem de descrição AHDL.

Escolheu-se para o estudo das alocações de estados os códigos de linha AMI (Alternated Mark Inversion), o código HDB1 (High-Density Bipolar First-Order Coding) e o código HDB3 (High-Density Bipolar Third-Order Coding) devido a sua importância em transmissão de dados em banda base através de um par de fios (par telefônico) que é a maneira mais simples, prática e econômica de comunicação de dados em perímetros urbanos ou distâncias de alguns quilômetros.

Estes códigos foram todos simulados no ambiente Max + Plus II e para o mesmo código foram utilizadas as alocações citadas.

RESULTADOS

Embora cada alocação de estado não altere o funcionamento de cada código de linha utilizado, os resultados obtidos no report file, gerado pelo MAX+plus II, têm diferenças entre si, pois o tipo de alocação altera os custos das funções lógicas, definidos como o número de portos de entradas das portas lógicas em seu formato primitivo, o número de células utilizadas (para uma implementação em FPGA), a quantidade de memória e tempo de síntese. Tais parâmetros foram utilizados para comparação entre as implementações.

Todos os códigos foram implementados no componente programável FPGA MAX7000E, Device: EPM 7128ELC84-7, e as simulações comprovam seu perfeito funcionamento.

As simulações referentes a cada código estão apresentadas na Figura 1.

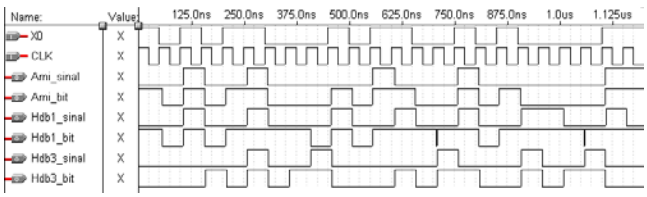

FIGURA 1.
SIMULAÇÃO DOS CÓDIGOS IMPLEMENTADOS.

A partir do Report File gerado pelo software Max+Plus II e das alocações, foram obtidos os custos totais para cada código, cujos resultados estão apresentados na Tabela I.

TABELA I.
CUSTOS OBTIDOS PARA CADA CÓDIGO VARIANDO-SE O TIPO DE ALOCAÇÃO.

Código	Alocação	Custo Lógico	Células Utilizadas	Memória Utilizada	Tempo de Compilação
AMI	Simplest	012	03	3,622 K	00:00:07
	One-Hot	012	04	3,670 K	00:00:03
	Almost One-Hot	012	03	3,499 K	00:00:04
HDB1	Simplest	028	04	3,688 K	00:00:04
	One-Hot	057	06	3,786 K	00:00:04
	Almost One_hot	058	06	6,197 K	00:00:02
HDB3	Simplest	340	14	3,687 K	00:00:09
	One-Hot	268	34	5,598 K	00:00:10
	Almost One-Hot	159	34	3,888 K	00:00:06

CONCLUSÃO

Com esse projeto concluiu-se que determinados tipos de alocações utilizados apresentaram vantagens em diferentes aspectos, por exemplo, para a alocação do tipo Almost One-Hot foi obtido, em geral, a menor tempo de compilação em relação a todas as implementações realizadas. Na alocação Simplest obteve-se a menor quantidade de células utilizadas.

Os resultados apresentados na Tabela I mostram que não existe um método de alocação ideal para otimizar um circuito, pois notou-se que para cada situação e condição (por exemplo células de FPGA utilizadas) há uma alocação específica que torna o circuito otimizado.

O ensino da linguagem AHDL no curso de graduação proporciona ao discente um conhecimento diferenciado, pois através da AHDL, o aluno pode descrever um circuito digital de forma alternativa e muito versátil, unido à facilidade de aprendizado.

Pode-se salientar que para o ambiente utilizado (MAX+plus II) a linguagem AHDL é uma linguagem de alto nível muito poderosa, visto que é uma linguagem desenvolvida pela ALTERA, fabricante do próprio MAX+plus II.

AGRADECIMENTOS

Os autores agradecem à CAPES e à PI Componentes por todo o suporte oferecido durante a realização do projeto, e também aos docentes da área de sistemas digitais do DEE-FEIS-UNESP.

REFERÊNCIAS

[1] COMER, D. J. Introduction to State Machine, *Digital Logic and State Machine Design*. 3.ed., 1995, p. 208-229.

[2] COMER, D. J.; General State Machine Architecture, *Digital Logic and State Machine Design*. 3.ed., 1995, p. 296-301.

[3] WAKERLY, J. F. State Assignment, *Digital Design Principles and Practices.*, p. 387-390.

[4] MAX+PLUS II AHDL, ver. 6.0,1995.

[5] FLEX 8000. *State Machine Encoding*, ver. 1, 1994, p. 187 - 189.

[6] STONHAM, T. J. State Assignment, *Digital Logic Techniques*, 3.ed., 1996, p. 109-110.

[7] FRANÇA, J. L. *Manual para Normalização de Publicações Técnico-Científicas*. 4. ed., 1998, 213p.