UM NOVO MÓDULO EQN PARA A FERRAMENTA FLEXMAP

JOÃO JÚNIOR DA SILVA MACHADO; JULIO S. DOMINGUES JUNIOR; LEOMAR SOARES DA ROSA JR; FELIPE DE SOUZA MARQUES
Universidade Federal de Pelotas
{jjdsmachado, jsdomingues, leomarjr, felipem}@inf.ufpel.edu.br

1. INTRODUÇÃO

A microeletrônica atingiu diversos avanços nas últimas décadas, uma das áreas que mais evoluiu foi a de concepção de circuitos integrados. Entretanto, mesmo com todo o avanço nos processos físicos e químicos, projetar circuitos digitais ainda não é uma tarefa trivial. Neste sentido, se torna essencial o uso de ferramentas para auxiliar os projetistas. Estas ferramentas ajudam tanto na automatização de alguns processos, quanto na diminuição de erros humanos inseridos no processo de síntese dos circuitos.

No desenvolvimento de um circuito integrado, o processo de síntese é uma das etapas realizadas. A síntese de um circuito pode ser dividida em três etapas: descrição de alto-nível, síntese lógica e síntese física. O projeto de um circuito digital começa geralmente com uma descrição mais abstrata do circuito. Essa descrição não possui preocupação com o detalhamento de algumas características necessárias no momento da implementação. Nesta etapa, normalmente o circuito é descrito em uma linguagem de descrição de *hardware*, como por exemplo, *Verilog Hardware Description Language* – (*VHDL*, 2013).

Num segundo momento, a síntese lógica é aplicada. Como resultado, é gerado uma descrição mais detalhada com informações sobre a tecnologia e os elementos que serão utilizados para fabricar o circuito. A síntese lógica pode ser dividida em diversas etapas. Em uma dessas etapas, são aplicadas otimizações independentes da tecnologia na qual o circuito será fabricado, assim como otimizações referentes à tecnologia alvo. Por fim, a síntese física preocupa-se com os aspectos geométricos e de fabricação do circuito, como, por exemplo, roteamento e posicionamento das células, e dimensionamento dos transistores.

Uma das etapas mais importantes da síntese lógica é o mapeamento tecnológico, pois ele define as principais características estruturais do circuito. O processo de mapeamento pode ser dividido em três etapas: decomposição, identificação de padrões e cobertura lógica (MARQUES, 2007).

Na etapa de decomposição, será feito a preparação da estrutura de dados (descrição sujeito) para o mapeamento. O circuito será descrito de acordo com o que é requerido pela estrutura de dados, que muitas vezes aceita apenas operadores primitivos ou um conjunto restrito de operadores lógicos (MARQUES, 2007). Após esta etapa, segue a etapa de identificação de padrões, que é responsável por tentar encontrar porções equivalentes dentro do circuito, com os elementos da biblioteca de células da tecnologia alvo. Uma das desvantagens na etapa de identificação de padrões é o fato de que algumas áreas podem resultar em casamentos duplicados. Em outras palavras, uma mesma área pode ter casamento com mais de uma célula, o que resulta em aumento de área no circuito final.

Tentando resolver esses problemas, aplica-se a etapa de cobertura lógica, a qual é responsável por eleger um conjunto de casamentos de forma que o circuito seja implementado da melhor forma. A cobertura tenta satisfazer uma função objetivo, onde esta função normalmente é a minimização de critérios como área, consumo de potência e atraso.

Dentro do escopo de mapeamento tecnológico, existem diversas ferramentas. Uma dessas ferramentas que merece destaque é a ABC (ABC, 2013), desenvolvida pela Universidade de Berkeley, na Califórnia. Esta ferramenta apresenta bons resultados e possui os métodos mais atuais quando se trata de mapeamento tecnológico. A ferramenta, assim como seu código fonte, é disponibilizada livremente. Entretanto, caso o usuário queira adicionar uma simples funcionalidade ou até mesmo um novo fluxo de mapeamento, será necessário uma etapa de programação adicional, modificando seus diversos métodos. Assim, não é possível aplicar e testar variações de fluxos rapidamente.

Neste sentido, a ferramenta *FlexMap* (*FlexMap, 2013*), tenta proporcionar um ambiente flexível para mapeamento tecnológico. Desta forma, a ferramenta permite uma fácil configuração de novos fluxos de mapeamentos, proporcionando ao usuário uma abordagem de código mais genérica, bem como uma interface de fácil acesso, o que viabiliza a rápida prototipação de novos fluxos. A ferramenta ainda está em desenvolvimento e novos módulos estão sendo inseridos.

Este trabalho apresenta um módulo desenvolvido para o *FlexMap.* O módulo desenvolvido realiza a conversão de descrições no formato EQN (*Equation Format*) para o formato AIGER (*Aiger*, 2013). A descrição resultante respeita as regras específicas de cada formato, assim como o circuito final é logicamente equivalente ao original.

2. METODOLOGIA

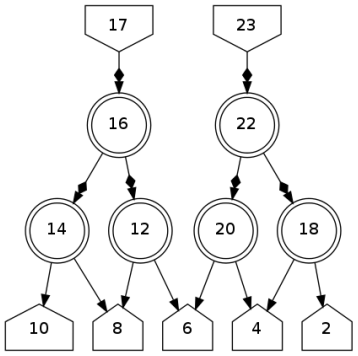
A ferramenta *FlexMap* é focada em mapeamento tecnológico e a proposta da mesma consiste em permitir, através de uma interface de mais alta abstração, um ambiente para síntese configurável, proporcionando a flexibilidade de compor fluxos alternativos sem a necessidade de programação adicional. Dessa forma, possibilita-se uma rápida prototipação de novas metodologias de mapeamento. O *FlexMap* ainda está em desenvolvimento e por isso, existem várias etapas a serem cumpridas dentro do projeto. Uma dessas etapas consiste na definição do tipo de descrição de entrada. Atualmente, a ferramenta apenas aceita circuitos descritos no formato AIGER, o que acaba limitando o escopo da mesma.

Os circuitos são representados em sua maioria por grafos, uma especialização de grafos utilizada é o grafo de portas lógicas *And* e *Inversores - AIG (And-Inverter Graph*). O formato AIGER é um padrão que específica um tipo de *AIG.* Este padrão é bastante utilizado na área de síntese lógica e, por isso, foi definido como descrição padrão para o *FlexMap*. Este formato possui duas versões de descrição, uma versão que é descrita no padrão ASCII (textual) e outra que é descrita em padrão binário. A versão utilizada dentro da ferramenta *FlexMap* é a versão ASCII. A Figura 1 ilustra o padrão AIGER utilizado na descrição de um circuito em formato textual (Figura 1.a) e, também, a representação do mesmo circuito em formato de grafo AIG (Figura 1.b).

O formato AIGER descreve a lógica do circuito apenas utilizando portas-lógicas primitivas do tipo *And e Inversor.* O diferencial deste padrão é que além de ser um padrão bem conhecido e utilizado nas principais ferramentas, como o ABC, também disponibiliza uma vasta documentação. Por outro lado, a proposta do *FlexMap* consiste na disponibilização de um ambiente genérico para mapeamento, o que contradiz a limitação de um único tipo de arquivo de entrada. Neste sentido, diversos padrões foram estudados para desenvolver módulos de entrada. Assim, o primeiro tipo de descrição que foi selecionado foi o padrão *Sysnopsys Equation Format - Eqn*. Este padrão foi desenvolvido pela empresa *Synopsys* e descreve a lógica do circuito através da equação lógica da célula em relação a suas entradas. A Figura 1.c

apresenta um exemplo do padrão *Eqn* na descrição do mesmo circuito anteriormente apresentado.

```
aag 11 5 0 2 6
2
4
6
8
10
17
23
16 13 15
22 19 21
14 8 10
12 6 8
20 4 6
18 2 4
```




```
INORDER = pi_2 pi_4 pi_6 pi_8 pi_10
OUTORDER = po_17 po_23;
po_17 = [16];
po_23 = [22];
[16] = ((2*4)+(4*6));
[22] = ((6*8)+(8*10));
```

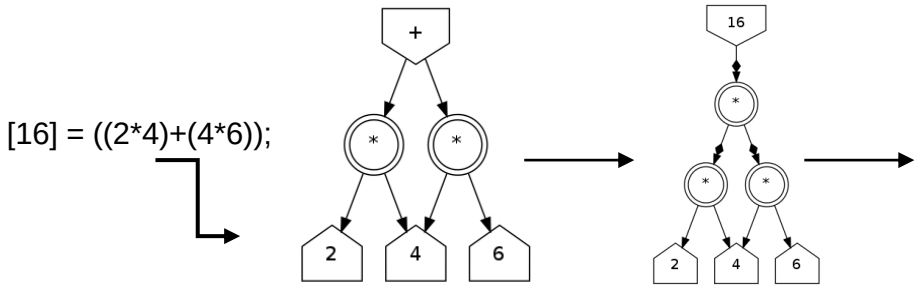
(a) – AIG versão ASCII (b) Grafo AIG (c) Eqn

Figura 1: Exemplos de Descrição.

O módulo desenvolvido carrega a descrição Eqn executando uma rotina para cada linha do arquivo, ou seja, para cada equação uma rotina é invocada. Essa rotina tem como objetivo, transcrever a lógica do Eqn para uma árvore de operadores e, quando necessário, aplicar algumas transformações. Ao final da rotina, a árvore estará respeitando as propriedades do formato AIGER. Assim, como última etapa, é preciso somente descrever a estrutura de cada árvore para gerar o arquivo de saída no padrão AIGER.

A Figura 2 apresenta as etapas do método para apenas uma linha do arquivo de exemplo na Figura 1.c. As transformações são necessárias, visto que o padrão AIGER permite que a lógica esteja descrita com operadores *And e Inversores,* o que não acontece no padrão Eqn. Essa transformação é feita baseada no teorema de *DeMorgan.* Através de *DeMorgan*, é possível apenas substituir nodos com a operação do tipo *Or,* por nodos do tipo *And*, com suas saídas e entradas negadas.

Outra transformação aplicada é a decomposição das equações com três ou mais entradas em diversas sub-equações de duas entradas. Essa transformação se deve à restrição do padrão AIGER, a qual exige que a lógica seja descrita em operadores binários (porta lógica *And* com duas entradas). Como última etapa, após as transformações necessárias, todas as árvores estão respeitando as propriedades do formato AIGER. Então, aplicando uma busca por amplitude em cada árvore, é possível descrever a estrutura de cada árvore, e gerar assim, o arquivo de saída no formato AIGER.




```
aag 5 2 0 1 3
2
4
6
17
8 2 4
10 4 6
16 9 13
```

(a) - Eqn (b) Grafo padrão EQN (c) Grafo padrão AIGER (d) AIG versão ASCII

Figura 2: Exemplos de Descrição

3. RESULTADOS E DISCUSSÃO

Com o objetivo de validar o módulo desenvolvido, utilizou-se um conjunto composto por 99 circuitos presentes no *benchmark* ISCAS'85 (ISCAS, 2013). Estes circuitos são bem conhecidos e utilizados dentro da área de síntese lógica. Com o objetivo de validar o módulo desenvolvido, os circuitos foram transcritos tanto no *FlexMap,* quanto na ferramenta ABC.

Através da funcionalidade para verificar equivalência lógica, presente na ferramenta ABC (comando *CEC)*, foi possível validar o módulo desenvolvido. Esta verificação foi efetuada comprovando que as duas descrições, tanto do *FlexMap* quanto do ABC, são equivalentes. Esta equivalência é aplicada de forma booleana, onde não se leva em conta a estrutura em que o circuito está descrito, mas sim, a função lógica que está sendo representada em ambos os circuitos. Todos os circuitos utilizados demonstraram resultados corretos quanto à equivalência lógica.

A integração deste módulo no projeto do *FlexMap*, possibilita uma integração com outras ferramentas desenvolvidas no grupo de pesquisa. Da mesma forma, será possível integrar o *FlexMap* com outras ferramentas da área, como por exemplo, o ABC. Esta integração pode ser fundamental para o desenvolvimento de novas técnicas de síntese lógica, pois permite o desenvolvimento iterativo com fluxos nas duas ferramentas.

4. CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um novo módulo, preenchendo uma das lacunas identificadas no projeto da ferramenta *FlexMap*. Este módulo foi validado e acrescentado a ferramenta. Dessa forma, o módulo desenvolvido permite a integração com outras ferramentas, como por exemplo, a ferramenta ABC. A adição de novos módulos no *FlexMap* é desejável, pois aumenta a capacidade de integração da ferramenta com o fluxo de desenvolvimento de circuitos integrados atualmente utilizado pela indústria de microeletrônica.

Como trabalhos futuros, pretendem-se desenvolver novos módulos que contemplem padrões de linguagens de descrições de *hardware,* como por exemplo, o formato *BLIF* (*Berkeley Logic Interchange Format*) e o formato *VHDL*.

5. REFERÊNCIAS BIBLIOGRÁFICAS

MARQUES, F. S.; Rosa, L. S. ; Ribas, R. P. ; Sapatnekar, S. S.; Reis, A. I. . ***DAG based library-free technology mapping***. In: 17th *Great Lakes Symposium on VLSI*, 2007, *Stresa-Lago Maggiore. Proceedings GLSVLSI. New York: ACM Press*, 2007.

ABC - BERKELEY, U. C. ABC: ***A System for Sequential Synthesis***. Acessado em 7 de Outubro de 2013. Online. Disponível em: http://www.eecs.berkeley.edu/alanmi/abc/abc.html.

GACI – UFPEL - ***FlexMap*: Uma novo *Framework* para Mapeamento Tecnológico**. Acessado em 7 de Outubro de 2013. Online. Disponível em: http://inf.ufpel.edu.br/gaci/?page_id=597.

Aiger Format - ***The AIGER And-Inverter Graph (AIG) Format***. Acessado em 7 de Outubro de 2013. Online. Disponível em: http://fmv.jku.at/aiger/.

ISCAS *High-Level Models*. **ISCAS *Benchmarks*** – Acessado em 8 de Outubro de 2013. Online. Disponível em: http://web.eecs.umich.edu/~jhayes/iscas.restore/

VHDL - ***Verilog Hardware Description Language*** - Acessado em 10 de Outubro de 2013. Online http://www.ics.uci.edu/~jmoorkan/vhdlref/