ALGORITMO DE VERIFICAÇÃO ESTRUTURAL APLICADO NO PROCESSO DE MAPEAMENTO TECNOLÓGICO NA FERRAMENTA FLEXMAP

JOÃO JÚNIOR DA SILVA MACHADO[1]; JULIO SARAÇOL DOMINGUES JUNIOR[2] FELIPE DE SOUZA MARQUES[3]; LEOMAR SOARES DA ROSA JUNIOR[4]

[1]Universidade Federal de Pelotas – jjdsmachado@inf.ufpel.edu.br
[2]Universidade Federal do Pampa – Campus Bagé – juliodomingues@unipampa.edu.br
[3]Universidade Federal de Pelotas – felipem@inf.ufpel.edu.br
[4]Universidade Federal de Pelotas – leomarjr@inf.ufpel.edu.br

1. INTRODUÇÃO

Devido a evolução da área de microeletrônica, o projeto de circuitos integrados (CI) *Very Large-Scale Integration* (VLSI) é considerado cada vez mais exigente, uma vez que é preciso desenvolver circuitos que apresentem, ao mesmo tempo, maior desempenho e menor consumo de energia. Existem diversas etapas que compõem a síntese de um CI, contudo, é na etapa de síntese lógica que gera-se uma descrição mais detalhada do circuito, contendo informações referentes a tecnologia e os elementos que serão utilizados para fabricar o circuito. Uma das etapas realizadas durante a síntese lógica é o processo de mapeamento tecnológico, o qual consiste em encontrar um conjunto de células interconectadas para implementar um dado circuito digital, visando minimizar uma função objetivo, que modela características como área, atraso, potência, etc.

O mapeamento tecnológico divide-se em Decomposição, Casamento e Cobertura (MARQUES, 2007). Este trabalho possui foco voltado para a etapa de casamento de padrões, a qual é responsável por verificar a equivalência entre os subgrafos da descrição sujeito que representa o circuito com as células da biblioteca *standard-cell*. Cada padrão é expresso por uma codificação, chamada de assinatura. A quantidade de comparações necessárias para identificação de padrões está diretamente relacionada ao tamanho do circuito e, tipicamente, este número está na casa de centenas de milhões. Neste sentido, investigar e propor novas metodologias para esta etapa pode viabilizar otimizações consideráveis no tempo de execução de uma ferramenta de síntese lógica.

Existem diversas formas para se efetuar o processo de verificação de equivalência de células, como por exemplo, verificação estrutural, verificação booleana e verificação por restrição. O objetivo do respectivo trabalho é proporcionar na ferramenta FlexMap (FLEXMAP, 2016) o processo de equivalência de células utilizando a abordagem de verificação estrutural. A ferramenta FlexMap foi desenvolvida na Universidade Federal de Pelotas e possui algumas estratégias e estruturas de dados que permitem a configuração de diferentes fluxos de síntese para realizar o processo de mapeamento tecnológico. Desta forma, é possível desenvolver dentro do ambiente FlexMap, fluxos alternativos de mapeamento e aplicar diferentes métodos de síntese, utilizando a estrutura de dados e o algoritmo de cobertura lógica desejados. A motivação para este trabalho, se deu em virtude de uma limitação existente no algoritmo de casamento de padrões implementado no FlexMap. Este algoritmo baseia-se numa metodologia Booleana, capaz de tratar apenas de funções compostas por no máximo 7 entradas. O algoritmo proposto neste trabalho consegue suprir esta limitação, proporcionando a utilização de funções com até 16 entradas. A implementação desta técnica, no contexto da ferramenta FlexMap, permite uma aceleração nos atuais algoritmos de mapeamento tecnológico disponíveis na ferramenta.

Como forma de avaliação do algoritmo desenvolvido, diversos experimentos foram efetuados comparando os resultados do método proposto com o método de verificação Booleana disponível no FlexMap. Os resultados mostram que não houveram diferenças significativas entre ambas abordagens, considerando o nível de otimização atingido. Além disso, o método proposto foi no mínimo 130 vezes mais rápido que o método Booleano.

2. METODOLOGIA

Em geral, as abordagens de mapeamento tecnológico são aplicadas utilizando estruturas de dados que representam o circuito. Uma boa estrutura de dados é elemento chave para o sucesso do mapeamento, uma vez que ela define as principais características do método e está diretamente relacionada com o consumo de memória (MARQUES, 2007). Neste trabalho, utilizou-se grafos do tipo DAG como estrutura de dados, mais precisamente, grafos do tipo AIG (do inglês, *And-Inverter-Graph*).

O algoritmo desenvolvido pode ser dividido em três etapas. Inicialmente o circuito é decomposto utilizando como descrição sujeito o formato AIGER, formato este utilizado para descrever grafos do tipo AIG. Após isso, aplica-se a abordagem de cortes-*k* proposta por Cong em (CONG; DING, 1992), o qual consiste em decompor a descrição sujeito em subgrafos, particionando assim o grafo que representa o circuito. Esta técnica permite uma melhor aplicação da etapa de casamento de padrões, já que cada subgrafo pode ser melhor verificado, possibilitando assim que se identifique um maior número de equivalências lógicas com os elementos disponíveis na biblioteca de células.

Após enumerar todos os cortes-*k*, realiza-se a assinatura destes cortes, utilizando a abordagem *bottom-up*. O método que efetua o assinalamentos dos cortes, aplica uma etapa intermediária, a qual é responsável por identificar as profundidades lógicas de todos os nodos que fazer parte do corte. Este processo é uma etapa muito importante, pois garante que grafos isomórficos sejam assinalados considerando a mesma estrutura, reduzindo desta forma o problema de dependência estrutural (CHATTERJEE et al., 2006).

Em um segundo momento, efetua-se o assinalamento das células presentes na biblioteca de células. Este processo consiste em gerar uma árvore para cada função presente na biblioteca de células. A partir da árvore resultante, o processo de assinalamento é efetuado, aplicando as mesmas abordagens e métodos utilizados para assinalar os subgrafos do circuito, o que irá gerar como resultado, uma assinatura estrutural que irá representar a respectiva função lógica.

Por fim, após ter todos os cortes-*k* da descrição sujeito e todas as funções da biblioteca de células gerados e assinalados, realiza-se a detecção dos melhores cortes de cada nodo. O critério para eleger a eficácia de um corte é o custo atrelado a este, o qual é calculado através da função custo que está sendo utilizada. Neste trabalho, utilizou-se a abordagem proposta por Valavan em (MANOHARARAJAH et al., 2006) denominada Fluxo de Área. Essa abordagem consiste em minimizar a área do circuito, computando os custos das melhores áreas, as quais serão consideradas no momento de aplicar a cobertura do circuito. Com todos os cortes-*k* avaliados, realiza-se então a cobertura do circuito, selecionando os melhores cortes e descartando aqueles que são englobados por inteiro por outro corte. Após isso, com o circuito mapeado, gera-se a descrição de saída do circuito, utilizando o formato EQN (do inglês, Equation Format), o qual descreve a função lógica de uma função booleana através de uma equação.

3. RESULTADOS E DISCUSSÃO

Com o objetivo de verificar a eficiência do algoritmo desenvolvido, diversos experimentos foram realizados, utilizando como base, o *benchmark* MCNC (YANG, 1988), o qual é um conjunto composto por 40 circuitos e bastante utilizado e referenciado na área de síntese lógica. Além disso, foi utilizado duas configurações distintas para função custo, ambas baseadas na abordagem proposta por Valavan (MANOHARARAJAH et al., 2006). A primeira função custo (*f1*) representa o computo do custo de área acumulado, obtido através da biblioteca de células. Já a segunda função custo (*f2*), segue a abordagem tradicional do fluxo de área proposto por Valavan.

Os experimentos realizados consistiram em comparar a abordagem de verificação estrutural, proposta neste trabalho, com a abordagem de verificação Booleana já implementada no FlexMap. Para isso, utilizou-se duas bibliotecas de células distintas. Uma biblioteca composta por 80 funções *P-class*, as quais representam todas as funções com até três entradas, rotulada na Tabela 1 como *3P.genlib* e, uma segunda biblioteca de células composta por 3984 funções *P-class*, as quais representam todas as funções com até quatro entradas, rotulada na Tabela 2 como *4P.genlib*. As tabelas abaixo representam o número de elementos lógicos necessários para mapear cada circuito do *benchmark* MCNC nas respectivas bibliotecas utilizadas, bem como o tempo de execução de cada abordagem para os respectivos circuitos.

Circuitos	Ver. Estrutural		Ver. Booleana	
	f1	*f2*	*f1*	*f2*
9symml	211	168	211	165
alu2	630	563	630	515
apex6	645	579	645	572
b1	13	11	13	4
c8	251	199	251	180
cc	64	55	62	39
cht	307	217	307	207
cm150a	68	45	68	16
cm151a	32	21	32	8
cm152a	31	24	31	20
cm162a	52	31	52	31
cm163a	50	30	50	30
cm42a	18	17	18	17
cm82a	21	13	14	6
cm85a	38	30	34	30
cmb	52	46	52	44
cu	66	60	66	56
decod	30	30	30	30
example2	326	291	326	279
frg1	659	607	659	606
frg2	1791	1560	1791	1408
i1	37	30	36	29
i2	232	223	231	221
k2	2289	1991	2289	1976
lal	144	112	142	101
ldd	112	93	112	93
majority	17	9	17	9
mux	123	96	123	92
pcle	62	54	62	54
pcler8	78	69	78	69
pm1	60	43	60	41
sct	153	124	151	113
tcon	40	32	40	8
term1	670	547	668	479
ttt2	508	430	506	380
unreg	112	95	112	95
vda	1020	891	1020	888
x1	1571	1407	1571	1385
x2	57	48	57	45
z4ml	217	185	217	170
Total	**12857**	**11076**	**12834**	**10511**
Tempo (s)	23	2915	3082	21

Tabela 1. Resultados *3P.genlib*

Circuitos	Ver. Estrutural		Ver. Booleana	
	f1	*f2*	*f1*	*f2*
9symml	211	142	211	132
alu2	630	528	630	436
apex6	645	536	645	509
b1	13	11	13	4
c8	251	174	251	140
cc	64	41	62	33
cht	307	170	307	142
cm150a	68	31	68	16
cm151a	32	15	32	7
cm152a	31	22	31	20
cm162a	52	35	52	29
cm163a	50	34	50	25
cm42a	18	17	18	17
cm82a	21	10	14	6
cm85a	38	26	34	28
cmb	52	38	52	33
cu	66	56	66	49
decod	30	30	30	30
example2	326	274	326	248
frg1	659	520	659	363
frg2	1791	1345	1791	1150
i1	37	28	36	23
i2	232	129	231	125
k2	2289	1836	2289	1716
lal	144	103	142	80
ldd	112	90	112	79
majority	17	7	17	5
mux	123	65	123	64
pcle	62	53	62	26
pcler8	78	62	78	62
pm1	60	43	60	35
sct	153	119	151	99
tcon	40	32	40	8
term1	670	471	668	369
ttt2	508	362	506	312
unreg	112	95	112	94
vda	1020	833	1020	806
x1	1571	1268	1571	921
x2	57	42	57	34
z4ml	217	167	217	139
Total	**12857**	**9860**	**12834**	**8414**
Tempo (s)	496	511	127671	128758

Tabela 2. Resultados *4P.genlib*

Uma vez que a abordagem Booleana obteve um número inferior de células necessárias para mapear cada circuito, em todos os experimentos realizados, aplicou-se o teste de hipótese *T de Student*, o qual utiliza conceitos estatísticos para avaliar se uma nova solução gerada pode ou não ser aceita. Os resultados da avaliação do respectivo teste, demonstraram que a diferença entre ambas as abordagens não são significativas, sendo portanto uma solução válida. Entretanto, os métodos Booleanos possuem em geral um custo computacional mais elevado, o que pode ser observado nos resultados obtidos através dos experimentos. Neste quesito, o algoritmo desenvolvido obteve um tempo de execução muito melhor que o algoritmo Booleano em todos os experimentos realizados, chegando a ser 138 vezes mais rápido que o algoritmo Booleano para a função custo *f2*, por exemplo. A razão para isto, é que os métodos Booleanos necessitam encontrar a forma canônica da função, o que implica em uma maior complexidade que a abordagem de verificação estrutural.

4. CONCLUSÕES

O presente trabalho teve como objetivo principal, desenvolver um algoritmo para a etapa de casamento de padrões. De acordo com os resultados obtidos, pode-se observar que o algoritmo desenvolvido apresentou resultados satisfatórios. Além disso, o método proposto possui um tempo de execução bem abaixo do apresentando pelo método Booleano.

A partir do desenvolvimento deste algoritmo, a ferramenta FlexMap, poderá considerar diferentes arranjos de células no momento do mapeamento. A importância deste trabalho se deve ao fato de que além de otimizar os resultados de mapeamento, o algoritmo desenvolvido irá contribuir para obter uma ferramenta acadêmica com diferentes fluxos de mapeamento, permitindo comparação entre os métodos e metodologias disponíveis na literatura.

5. REFERÊNCIAS BIBLIOGRÁFICAS

CHATTERJEE, S.; MISHCHENKO, A.; BRAYTON, R. K.; WANG, X.; KAM, T. Reducing Structural Bias in Technology Mapping. **In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. 25, no. 12, pp. 2894-2903, Dec. 2006.

CONG, J.; DING, Y. An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. **In Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD '92)**. IEEE Computer Society Press, Los Alamitos, CA, USA, 48-53.

FLEXMAP. **FlexMap Um novo Framework para Mapeamento Tecnológico**. Acessado em 08 de ago. de 2016. Online. Disponível em: http://inf.ufpel.edu.br/gaci/?page_id=597.

MANOHARARAJAH, V.; BROWN, S. D.; VRANESIC, Z. G. Heuristics for Area Minimization in LUT-Based FPGA Technology Mapping. **In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. 25, no. 11, pp. 2331-2340, Nov. 2006

MARQUES, F. S.; ROSA JR, L. S.; RIBAS, R. P.; SAPATNEKAR, S. S.; REIS, A. I. 2007. DAG based library-free technology mapping. **In Proceedings of the 17th ACM Great Lakes symposium on VLSI (GLSVLSI '07)**. ACM, New York, NY, USA, 293-298.

YANG, S. Logic Synthesis and Optimization Benchmarks. **Published at 1989 MCNC International Workshop on Logic Synthesis**.