UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação

Dissertação

Geração Automática de Redes de Transistores Dedicada a Dispositivos FinFET com Gates Independentes Baseada em Composição Funcional

Renato Souza de Souza

Pelotas, 2016

Renato Souza de Souza

Geração Automática de Redes de Transistores Dedicada a Dispositivos FinFET com Gates Independentes Baseada em Composição Funcional

Dissertação apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal de Pelotas, como requisito parcial para à obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Leomar Soares da Rosa Junior

Coorientador: Prof. Dr. Felipe de Souza Marques

Pelotas, 2016

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

S719g Souza, Renato Souza de

Geração automática de redes de transistores dedicada a dispositivos FinFET com gates independentes baseada em composição funcional / Renato Souza de Souza ; Leomar Soares da Rosa Junior, orientador ; Felipe de Souza Marques, coorientador. — Pelotas, 2016.

66 f.

Dissertação (Mestrado) — Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2016.

1. Vlsi design. 2. Tecnologia cmos. 3. Tecnologia FinFet. 4. (ig) FinFET. 5. Double gate. I. Rosa Junior, Leomar Soares da, orient. II. Marques, Felipe de Souza, coorient. III. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Renato Souza de Souza

Geração Automática de Redes de Transistores Dedicada a Dispositivos FinFET com Gates Independentes Baseada em Composição Funcional

Dissertação aprovada, como requisito parcial, para obtenção do grau de Mestre em Ciência da Computação, Programa de Pós-Graduação em Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da defesa: 04 março de 2016

Banca Examinadora:

.....

Prof. Dr. Leomar Soares da Rosa Junior (Orientador)
Doutor em Microeletrônica pela Universidade Federal do Rio Grande do Sul

.....

Prof. Dr. Felipe de Souza Marques (Coorientador)
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

.....

Prof. Dr. Júlio Carlos Balzano de Mattos
Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

.....

Prof. Dr. Rafael Iankowski Soares
Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul

.....

Profª. Drª. Cristina Meinhardt
Doutora em Ciência da Computação pela Universidade Federal do Rio Grande do Sul

Agradecimentos

A Universidade Federal de Pelotas, pela oportunidade de realizar minha pós-graduação em Computação.

Aos professores por proporcionar-me acesso ao conhecimento e também por seus esforços para ofertar um ensino público de qualidade.

Aos meus orientadores Leomar e Felipe, pelas orientações, apoio e confiança. O acolhimento de vocês novamente fez toda a diferença.

Ao amigo Vinicius Possani, por ter me ajudado a decidir o tema de pesquisa. Sua ajuda foi fundamental para minha formação.

Ao Mayler Martins, que mesmo longe, abdicou-se do seu tempo para me ajudar no desenvolvimento do trabalho.

Aos amigos, Ricardo, Roger, Lucas, João, Maicon, Regis e Gustavo pelos bons momentos vividos durante esta pós-graduação. O companheirismo nos momentos de estudos e lazer foram essenciais.

A minha família, minha mãe, meu irmão e minha namorada, pelo amor, incentivo e apoio incondicional.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado!

“Conhecimento serve para encantar as pessoas,
Não para humilhá-las.”
Mario Sergio Cortella

Resumo

SOUZA, Renato Souza. **Geração Automática de Redes de Transistores Dedicada a Dispositivos FinFET com Gates Independentes Baseada em Composição Funcional**. 2016. 66f. Dissertação (Mestrado em Computação) – Programa de Pós-Graduação em Computação. Universidade Federal de Pelotas, Pelotas, 2016.

A tecnologia FinFET é amplamente reconhecida como a principal alternativa para resolver os problemas ocasionado pela redução do canal do transistor. Esta tecnologia consiste em uma nova abordagem para a construção de um transistor em três dimensões. Assim, o *gate* do transistor mantem contato com três faces do canal, proporcionando um controle maior do fluxo dos elétrons no canal. A estrutura padrão de um transistor FinFET é conhecida como *Single Gate* (SG) FinFET. Contudo, algumas variações desta estrutura foram propostas. Uma destas variações estruturais é conhecida como *Independent Gate* (IG) FinFET, onde um transistor (IG) FinFET pode ser implementado com dois *gates*. Consequentemente, explorar os agrupamentos de um transistor (IG) FinFET *double gate* acaba por tornar-se um meio interessante para reduzir o número de transistores em um circuito. Neste contexto, este trabalho propõem-se um método alternativo para a geração de redes de transistores dedicada a dispositivos (IG) FinFET *double gate*. O método baseia-se em uma metodologia, especialmente desenvolvida para síntese lógica, chamada de Composição Funcional. Utilizando-a, é possível controlar o número de transistores associados em paralelo ou em série. Sendo assim, simplifica a procura por padrões de arranjos promissores para explorar o potencial dos dispositivos (IG) FinFET *double gate*. Os experimentos realizados demonstram que o método proposto é capaz de gerar redes de transistores, com um número reduzido de dispositivos (IG) FinFET *double gate*, quando comparado com os métodos dedicados a este mesmo propósito.

Palavras-chave: VLSI design, tecnologia CMOS, tecnologia FinFET, Transistor, MOSFET, (IG) FinFET, *double gate*.

Abstract

SOUZA, Renato Souza. **Using Functional Composition to Automatically Generate IG FinFET Transistor Networks**. 2016. 66f. Dissertação (Mestrado em Computação) – Programa de Pós-Graduação em Computação. Universidade Federal de Pelotas, Pelotas, 2016.

The FinFET technology is widely recognized as the leading alternative to solve problems minimization of short-channel effects. This technology consists in the construction of a transistor in three dimensions. Thus, the channel of the FinFET transistor is rounded by the gate in such a way that there is a contact of three faces of the channel with three sides of the gate. The standard structure of a FinFET transistor is known as single-gate (SG) FinFET. However, some variations of this structure have been proposed. One of these structural changes is known as Independent-Gate (IG) FinFET, where a transistor (IG) FinFET can be implemented with two gates. This way, explore the potential provided by (IG) FinFET transistors becomes a powerful strategy to decrease the transistor count in logic gates. This way, explore the potential provided by (IG) FinFET transistors becomes a powerful strategy to decrease the transistor count in logic gates. The method is based on a methodology developed specifically for logic synthesis, known as Functional Composition. It methodology allows controlling the number of associated transistors in parallel or in series. Thus, it simplifies the search for patterns of promising arrangements to explore the potential of devices (IG) FinFET double gate. The experiments have demonstrated that the proposed method is able to generate optimized IG FinFET transistor networks, when compared to methods dedicated to this same purpose.

Keywords: VLSI design, CMOS technology, FinFET technology, MOSFET, (IG) FinFET, double gate.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AIG	*And-Inverter Graph*
(IG) FinFET	*Independent Gate FinFET*
(SG) FinFET	*Single Gate FinFET*
CMOS	*Complementary Metal-Oxide-Semiconductor*
DARP	*Defense Advanced Research Projects Agency*
DG	*Double Gate*
FC	*Functional Composition*
FinFET	*Field-Effect Transistor*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
QCA	*Quantum Dots Cellular Automata*
ROBDD	*Reduced and Ordered Binary Decision Diagram*
SOP	*Soma de Produtos*
VLSI	*Very-Large-Scale Integration*

Sumário

1 INTRODUÇÃO

A indústria de microeletrônica tem apresentado grandes avanços nas últimas décadas, em especial, na evolução no desenvolvimento de circuitos VLSI (*Very-Large-Scale Integration*). Estes circuitos eletrônicos possuem um grande número de transistores embarcados em um único chip, chegando, nos dias atuais, na casa de bilhões de transistores. Assim, temos, a cada dia, circuitos cada vez mais complexos.

Essa evolução, basicamente, se deve ao grande avanço na escala da tecnologia CMOS (*Complementary Metal-Oxide-Semiconductor*), que é hoje a tecnologia mais largamente utilizada na fabricação de circuitos integrados. Tal avanço, proporcionou a construção de transistores menores, atingindo, assim, um melhor desempenho e uma alta capacidade de integração. Porém, o contínuo dimensionamento dos transistores não é algo trivial em virtude dos limites dos materiais utilizados e do processo de fabricação (KING, 2005). Dentre esses limites podemos citar as consequências elétricas da redução do canal do transistor e as dificuldades no processo de litografia durante a fabricação do circuito integrado.

Em termos elétricos, um dos principais desafios consiste na minimização da corrente de fuga. Isto é, esse consumo de energia acontece quando o circuito encontra-se em estado de equilíbrio, ou seja, não ocorre nenhuma transição em suas entradas ou saídas. Essa corrente basicamente acontece devido a redução do canal do transistor, pois a estrutura planar do *gate* dos transistores MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*) não permite um controle total e eficaz da movimentação dos elétrons no canal dos transistores (POSSANI, 2015). Em virtude desta característica do transistor MOSFET, pesquisas apontaram para a adoção de novas estruturas de transistores, surgindo, assim, a tecnologia FinFET (*Field Effect Transistor*) (HUANG, LEE, *et al.*, 1999).

A tecnologia FinFET propõem uma nova abordagem para construir um transistor. Um transistor FinFET é construído em três dimensões, onde o *gate* mantem contato com três áreas (regiões) ao redor do canal do transistor. Deste modo, é possível ter um controle de forma rápida e segura do fluxo de elétrons no canal do transistor. Além disso, o processo de fabricação de um transistor FinFET é compatível com o modo de fabricação do transistor MOSFET, tornando-o uma alternativa viável para atuar como substituto.

1.1 Motivação

Entre as tecnologias emergentes, a tecnologia FinFET oferece características interessantes em escala nanométrica, devido a possibilidade de um controle maior do canal do transistor. Estudos afirmam que o transistor FinFET apresenta vantagens em termos de desempenho e eficiência energética, tanto no consumo dinâmico, quanto no estático (SKOTNICKI, HUTCHBY, *et al.*, 2005) (NOWAK, RAINEY, *et al.*, 2002) (NOWAK, ALLER, *et al.*, 2004) (DOYLE, ARGHAVANI, *et al.*, 2002) .

Outro ponto muito importante, o qual já foi citado, é o fato do processo de fabricação do transistor FinFET ser compatível com o do transistor MOSFET. (HUANG, LEE, *et al.*, 1999). Em consequência, as principais fabricantes de circuitos VLSI avaliam um pequeno aumento, entre 2% a 5%, no custo total da produção de circuitos com transistores FinFET (SYNOPSYS, 2012). Em comparação com o transistor MOSFET, o consumo de potência estática pode ser reduzido em mais de 90% e também é possível obter um aumento de 37% em desempenho utilizando a metade da potência dinâmica (SYNOPSYS, 2012). Sendo assim, as principais fabricantes de circuitos integrados adaptam-se aos novos fluxos de produção para o desenvolvimento de circuitos com transistores FinFET.

A Intel foi uma das pioneiras na fabricação de circuitos integrados com transistores FinFET de 22 nanômetros em 2011. Com este feito, a Intel provou que era possível obter o mesmo desempenho, com uma menor dissipação de potência, do que um transistor MOSFET de 32 nanômetros. Ou, ainda, podia-se obter maior desempenho com o mesmo consumo de energia de uma solução em 32 nanômetros. Essas informações são apresentadas na Figura 1.

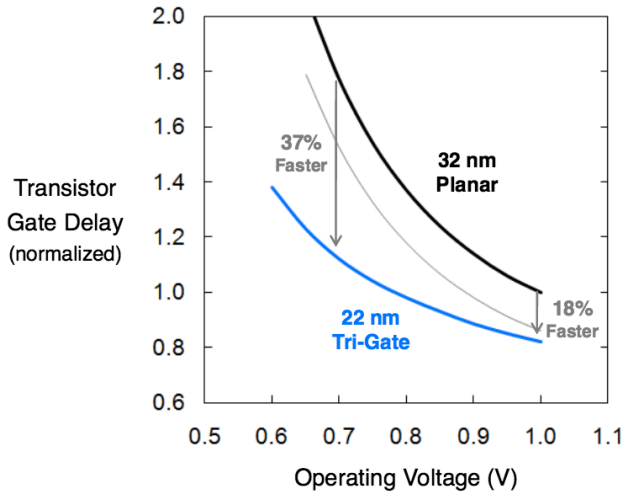


Figura 1 - Gráfico comparativo que mostra a relação de atraso entre as tecnologias da Intel de 32 nm MOSFET e a de 22 nm FinFET. (INTEL, 2011)

Devido ao panorama atual da indústria, muitos trabalhos estão sendo realizados, tanto em síntese física quanto em síntese lógica. Porém, ainda existem desafios e possibilidades de projeto que podem ser exploradas. Como exemplo, equipes de projetos de circuitos integrados estão tendo que utilizar suas ferramentas atuais, as quais são voltadas para projeto com transistores MOSFET, em projetos de circuitos com transistores FinFET. Com isso, acabou-se criando novos desafios para essas equipes, pois suas técnicas e ferramentas não lhes permitem projetar um circuito ideal com transistores FinFET. Estes desafios acarretam no atraso do tempo de entrega do projeto. (SYNOPSYS, 2012)

Além dos avanços em termos de desempenho, consumo de energia e integração dos dispositivos, a tecnologia FinFET, em particular, também possibilita uma maior capacidade de exploração no espaço de projeto referente ao modo de construção do transistor. Um transistor FinFET pode ser construído de duas formas. Com um único *gate* ao redor do canal do transistor, conhecido por *Single-Gate* (SG) FinFET. Ou, ainda, com dois *gates* independentes alinhados em paralelo, conhecido como *Independent-Gate* (IG) FinFET. Nesse sentido, os dois *gates* da estrutura (IG) FinFET podem ser explorados com diferentes objetivos de acordo com as restrições e as necessidades do projeto. Alguns trabalhos mostram a possibilidade de controlar cada um dos *gates* independentes com um sinal de entrada diferente. Através dessa técnica é possível reduzir o número total de transistores necessários para implementar uma determinada função lógica (CHIANG, KIM, *et al.*, 2005) (CHIANG,

KIM, *et al.*, 2006) (DATTA, GOEL, *et al.*, 2007) (WANG, 2010) (ROSTAMI e MOHANRAM, 2011). Portanto, explorar os agrupamentos em um transistor (IG) FinFET se torna um meio interessante para reduzir o número de transistores em um circuito.

Existem diferentes métodos disponíveis na literatura dedicados a reduzir o número de transistores em uma rede. Alguns métodos são baseados em técnicas de fatoração de expressões Booleanas (SENTOVICH, 1992) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012) e outros em otimização sobre estruturas de grafos (KAGARIS e HANIOTAKIS, 2007) (POSSANI, SOUZA, *et al.*, 2012) (POSSANI, CALLEGARO, *et al.*, 2015). Todos os métodos podem obter soluções satisfatórias para redes com transistores MOSFET e também com transistores (SG) FinFET, pois eles baseiam-se na minimização do número de literais para implementar uma função lógica. Porém, Possani (2015) demonstrou, através de diversos estudos de caso, as deficiências e limitações dos métodos disponíveis na literatura quando utilizados em redes com dispositivos (IG) FinFET (POSSANI, 2015). Em especial, o autor apontou à existência de uma mudança de paradigma que acabou sendo introduzida dentro da síntese lógica devido ao uso dos transistores (IG) FinFET *double gate*. O autor demonstrou que, ao considerar transistores (IG) FinFET *double gate*, os métodos de minimização de transistores não são capazes de fornecer a melhor solução. Neste sentido, Possani (2015) propõe dois métodos alternativos para geração automática de redes de transistores baseadas no uso de dispositivos (IG) FinFET *double gate*. Os dois métodos são baseados em grafos. O primeiro tem o intuito de encontrar padrões de arranjos os quais podem ser agrupados. O segundo método utiliza uma árvore lógica e também um técnica de defatoração de expressões Booleanas. Os dois métodos têm por objetivo explorar o potencial dos transistores (IG) FinFET *double gate*, com o intuito de minimizar o número total de transistores na rede final automaticamente gerada.

Apesar de Possani (2015) demonstrar que os métodos proposto são capazes de gerar redes de transistores (IG) FinFET *double* gate otimizadas, e com um baixo custo em tempo de execução, ainda existe uma lacuna a ser explorada. Devido ao fato dos métodos apresentados por Possani (2015) serem baseados em grafos, acredita-se que através do uso de uma metodologia especialmente desenvolvida para síntese lógica, conhecida como Composição Funcional (*Functional Composition*

- *FC*) (REIS, 2009) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012), é possível obter resultados melhores. A Composição Funcional apresenta uma grande flexibilidade para criar algoritmos com bons resultados para diversas aplicações. Quando aplicada na síntese de funções Booleanas, a metodologia de Composição Funcional apresenta bons resultados (MARTINS, DA ROSA JUNIOR, et al., 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012). Além disso, o uso da técnica de Composição Funcional possibilita ter o controle de critérios específicos durante a otimização (MARTINS, DA ROSA JUNIOR, *et al.*, 2010). Um destes critérios, por exemplo, é o controle do número de transistores associados em paralelo ou em série, tornando esta metodologia uma excelente técnica para realizar os agrupamentos dos transistores (IG) FinFET.

1.2 Objetivos

De uma maneira geral, os principais desafios para a mudança de tecnologia estão relacionados às etapas de síntese física no fluxo de projeto de um circuito integrado. Deste modo, a indústria acaba dedicando-se para solucionar problemas químicos, físicos, geométricos e elétricos. Porém, existe uma demanda por soluções eficientes relacionadas com a fase de síntese lógica do projeto de um circuito integrado. Nesse sentido, considerando a evolução dos circuitos VLSI e a existência de novas possibilidades de pesquisa introduzidas pela adoção e utilização da tecnologia FinFET, este trabalho tem como objetivo apresentar um método alternativo de geração automática de redes de transistores dedicado a utilização de (IG) FinFET. O método proposto irá aplicar a metodologia alternativa proposta por Martins, conhecida como Composição Funcional (MARTINS, DA ROSA JUNIOR, *et al.*, 2010), no intuito de fornecer soluções mais otimizadas do que as técnicas atualmente disponíveis na indústria e na academia.

2 BACKGROUND

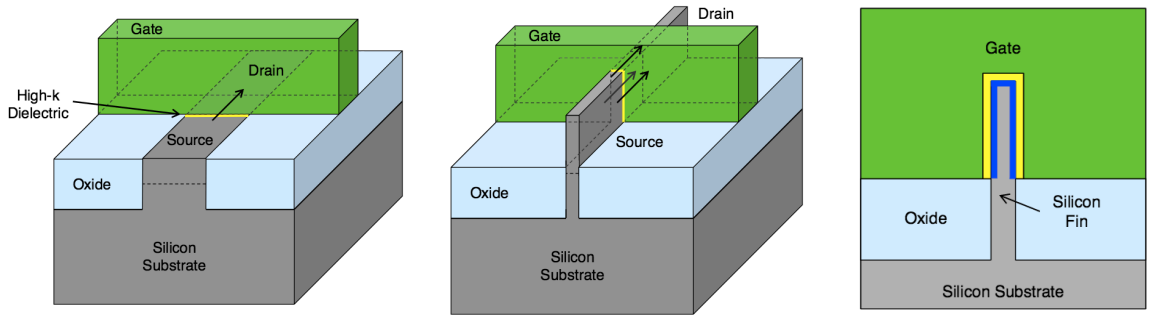
Este capítulo apresenta uma breve discussão sobre a tecnologia FinFET e, também, uma rápida revisão bibliográfica sobre trabalhos relacionados, os quais propõem novas técnicas e implementações de portas lógicas baseadas no uso de transistores FinFET.

2.1 Transistor FinFET

Por volta do ano de 1996, a indústria de microeletrônica fabricava transistores com tecnologia MOSFET de 250 nanômetros. Porém, para a grande maioria dos pesquisadores, ter transistores MOSFET para tecnologias abaixo de 100 nanômetros, seria inviável. Assim, a *Defense Advanced Research Projects Agency* (DARP) começou a financiar pesquisas inovadoras para o sucessor do transistor MOSFET. Uma equipe da Universidade de Berkeley, na Califórnia, liderada pelo professor e pesquisador Chenming Hu, concebeu uma proposta de duas novas estruturas de transistores que poderiam suceder o transistor MOSFET. Uma delas foi a estrutura FinFET. Assim, em 1997, o grupo de pesquisa de Chenming Hu recebeu recursos para demonstrar, experimentalmente, a tecnologia FinFET, dando origem ao primeiro artigo sobre o assunto (HUANG, LEE, *et al.*, 1999).

Como já foi citado anteriormente, a principal diferença estrutural do transistor FinFET para o MOSFET reside no canal. O canal do transistor FinFET é envolvido pelo *gate* de forma que exista contato entre três partes do *gate* com o canal. O resultado é apresentado pelo corte da Figura 2(c), diferentemente do transistor MOSFET, ilustrado pela Figura 2(a), onde a fonte e o dreno se comportam como canais internos na camada de silício. No caso do transistor FinFET, fonte e dreno, se projetam para fora da camada de silício, como se fosse uma barbatana (*"fin"*, em inglês), atravessando o *gate* do transistor. Isso pode ser notado através da Figura 2(b), a qual apresenta de forma abstrata a estrutura modelo de um transistor

FinFET. Essa estrutura também é conhecida como *Single-Gate* (SG) FinFET, devido a construção de um único *gate* ao redor de todo o canal do transistor.



(a) (b) (c)

Figura 2 - Transistor MOSFET em (a), a estrutura padrão de um transistor FinFET em (b) e corte na vertical de um transistor FinFET em (c). (INTEL, 2011)

Devido ao fato do canal do transistor FinFET ser envolvido pelo *gate*, a tensão aplicada ao *gate* acaba distribuindo-se por uma superfície maior. Sendo assim, permite-se um controle maior do canal do transistor. Consequentemente, é possível obter um maior controle do fluxo de corrente no transistor e, também, fazer com que a tensão no substrato de silício não influencie na corrente enquanto o transistor encontra-se no estado desligado. Portanto, é possível obter uma baixa dissipação de potência estática, mais conhecida como *leakage,* e um aumento importante na capacidade de isolamento, quando comparado a um transistor MOSFET. Assim o transistor acaba consumindo menos energia quando está desligado.

Outra característica muito interessante encontrada no transistor FinFET é a possível obtenção de um bom desempenho com um baixo consumo de energia total. Basicamente, isso ocorre devido à grande área de inversão do transistor, fazendo com que ocorra um maior fluxo de corrente quando o transistor está conduzindo. Além disso, é possível aumentar o número de *fins* no transistor, com o objetivo de aumentar a capacidade de drenagem e, consequentemente, aumentar a precisão e o controle do transistor. A Figura 3 ilustra a estrutura de um transistor FinFET com múltiplos *fins*.

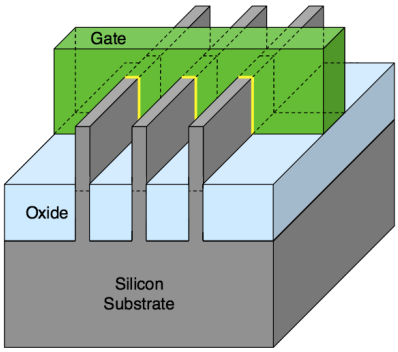


Figura 3 - Estrutura de um transistor FinFET com múltiplos *fins.* (INTEL, 2011)

Sem dúvida, essa nova abordagem de transistor conduziu uma renovação para a indústria de microeletrônica, viabilizando a continuidade da tecnologia CMOS. Apesar da tecnologia de transistores FinFET estar bem consolidada na indústria nos dias atuais, existe ainda um grande conjunto de desafios e possibilidades de projetos, relacionados a tecnologia FinFET, os quais podem ser explorados.

2.2 Variações e Inovações Sobre Transistores FinFET

Na seção anterior foi possível obter um conhecimento essencial sobre a tecnologia FinFET. A estrutura apresentada anteriormente, é considerada a estrutura padrão de um transistor FinFET, a qual é conhecida como transistor (SG) FinFET. É possível encontrar na literatura, trabalhos que apresentam algumas variações desta estrutura de transistor (SG) FinFET. Algumas dessas variações estruturais são conhecidas como *Independent Gate* (IG) FinFET ou *Double Gate* (DG) FinFET (WANG, 2010) (ROSTAMI e MOHANRAM, 2011) (MISHRA, MUTTREJA e JHA, 2011) (CAKICI, MAHMOODI, *et al.*, 2005).

Um transistor (IG) FinFET basicamente é construído por meio da remoção da parte superior do *gate* do transistor (SG) FinFET (LIU, MATSUKAWA, *et al.*, 2007). Portanto, removendo a extremidade do *gate* do transistor (SG) FinFET da Figura 4(a) é possível obter um transistor (IG) FinFET, representado pela Figura 4(b). Através desta estrutura é possível utilizar cada *gate* do transistor com diferentes propósitos de acordo com as restrições e necessidades de projeto no desenvolvimento de um circuito integrado. Assim, pode-se obter reduções em área, potência dinâmica ou capacitância do circuito.

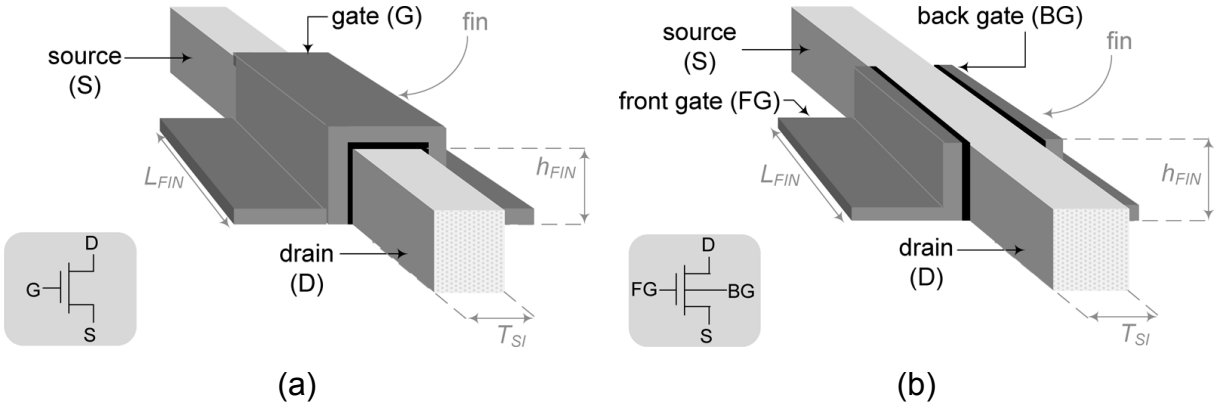


Figura 4 - Estrutura (SG) FinFET em (a) e estrutura (IG) FinFET em (b). (ALIOTO, 2011)

Uma das soluções possíveis encontradas na literatura consiste em utilizar um dos *gates* para realizar o controle da tensão de *threshold* dinamicamente, obtendo, assim, uma redução na dissipação de potência estática e um aumento no desempenho do transistor (MUTTREJA, AGARWAL e JHA, 2007).

Outro possível proposito é conectar cada um dos *gates* em um sinal diferente, podendo, assim, implementar tanto um arranjo de dois sinais em paralelo *(a+b),* quanto um arranjo em série *(a*b)* (MUTTREJA, AGARWAL e JHA, 2007) (WANG, 2010). Basicamente, isso é realizado com o objetivo de minimizar o número total de transistores de uma determinada função lógica, obtendo, desta forma, uma redução em área (CHIANG, KIM, *et al.*, 2006) (CHIANG, KIM, *et al.*, 2005) (WANG, 2010) (ROSTAMI e MOHANRAM, 2011) (POSSANI, 2015).

À vista disso, nas próximas seções serão brevemente apresentados e discutidos os principais trabalhos que estão diretamente ligados à exploração dos transistores (IG) FinFET. Alguns trabalhos demonstram que é viável a implementação de uma rede de transistores com dois *gates* independentes (IG) FinFET. Outro, apresenta métodos alternativos para geração automatizada de redes de transistores baseada em dispositivos (IG) FinFET.

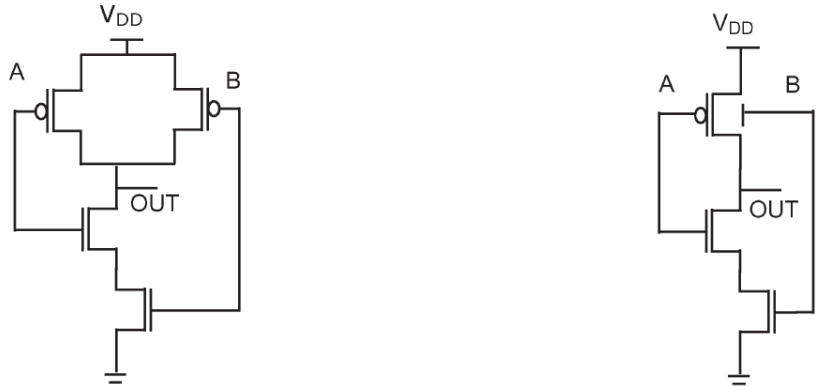
Os trabalhos apresentados são fundamentais para se obter uma melhor compreensão dos avanços e, principalmente, dos desafios que a tecnologia FinFET vem ocasionando para a síntese lógica e para a síntese física.

2.2.1 Chiang et al. 2005 (CHIANG, KIM, *et al.*, 2005)

Chiang et al. propuseram uma nova implementação de portas logicas *NAND* e *NOR* de duas entradas, utilizando transistores (IG) FinFET. Devido ao fato de um

transistor (IG) FinFET possuir dois *gates* independentes, os autores aproveitaram esta possibilidade para ligar cada sinal de entrada em um *gate* diferente do transistor (IG) FinFET. Assim, com apenas um transistor (IG) FinFET é possível ter dois sinais de entrada. Com isso, é possível reduzir a área e a dissipação de potência da porta lógica. O objetivo dos autores com esse agrupamento foi realizar um arranjo em paralelo entre os dois sinais, como exemplo *(a+b)*. Isso foi possível devido ao fato de, que para realizar o arranjo em paralelo, o *threshold* do transistor não necessita ser elevado.

O transistor proposto foi chamado pelos autores de (IG) FinFET *regular-*V_T. A Figura 5(a) ilustra a representação tradicional de uma porta lógica *NAND* de duas entradas, onde é possível notar que os dois transistores P-FinFET estão associados em paralelo. Já a implementação proposta por Chiang et al. é apresentada pela Figura 5(b), onde é possível notar que os dois transistores que estavam em paralelo foram agrupados em apenas um único transistor (IG) FinFET.



(a) (b)

Figura 5 - Implementação tradicional de uma porta *NAND* de duas entradas em (a) e implementação proposta pelos autores com agrupamento de transistores em paralelos em (b). (CHIANG, KIM, *et al.*, 2005)

Os autores demonstraram que, utilizando a implementação desta porta lógica *NAND* de duas entradas, é possível obter reduções em torno de 40% em potência estática e de 33% em potência dinâmica. Foi possível notar, também, uma melhoria de 10% em termos de desempenho e uma redução na capacitância de entrada. Isso tudo comparado com uma implementação tradicional de um *NAND* de duas entradas. No caso da implementação proposta para a porta lógica *NOR* de duas entradas, as reduções obtidas foram de aproximadamente 20% para a

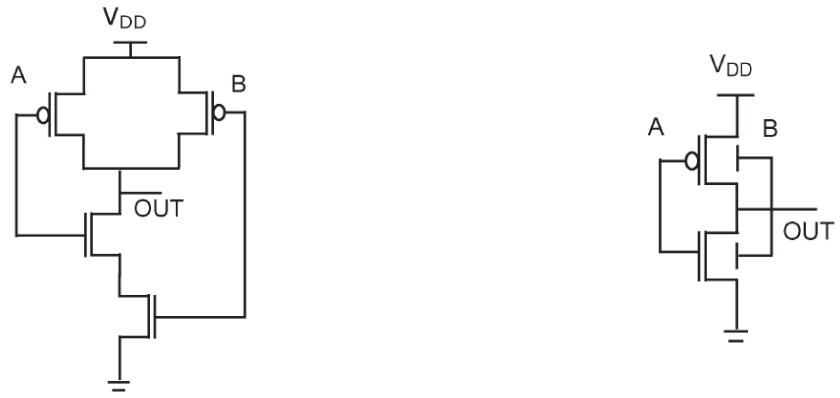
potência estática e 17% na potência dinâmica. Porém, a solução proposta para a *NOR* é aproximadamente 2% mais lenta que a implementação tradicional. Os experimentos foram realizados utilizando o valor para V_{DD} = *1.0V*.

Concluindo, através da abordagem proposta, é possível obter redução em área, potência e capacitância total de uma porta lógica.

2.2.2 Chiang et al. 2006 (CHIANG, KIM, *et al.*, 2006)

Em 2006, Chiang et al. propuseram uma nova implementação de transistores no qual também é possível agrupar dois transistores em série utilizando apenas um transistor (IG) FinFET. Esse agrupamento de transistores em série também é possível de ser realizado devido ao ajuste da função de *threshold* do transistor, análogo ao agrupamento em paralelo proposto por Chiang et al. em 2005 (CHIANG, KIM, *et al.*, 2005). Basicamente, o transistor irá conduzir o sinal quando os dois *gates* estiverem ativos. Caso apenas um dos *gates* estiver ativo, a corrente acaba sendo consideravelmente baixa, assim o transistor não conduz o sinal. Este tipo de transistor foi denominado pelos autores de transistores (IG) FinFET *high-*V_T.

Portanto, utilizando o transistor (IG) FinFET *regular-*V_T (CHIANG, KIM, *et al.*, 2005) e o transistor (IG) FinFET *high-*V_T (CHIANG, KIM, *et al.*, 2006), os autores demostraram que é possível realizar uma nova implementação para a porta lógica *NAND* de duas entradas com apenas dois transistores (IG) FinFET, ilustrado pela Figura 6(b). Já na implementação tradicional se faz necessário quatro transistores FinFET, como pode ser visto na Figura 6(a).



(a) (b)

Figura 6 - Implementação tradicional de uma porta *NAND* de duas entradas em (a) e solução otimizada com os transistores (IG) FinFET *high-*V_T *(série)* e um (IG) FinFET *regular-*V_T *(paralelo)* em (c). (CHIANG, KIM, *et al.*, 2006)

Os autores mostraram que é possível reduzir a área e a capacitância em duas vezes com um aumento no desempenho de 19%, utilizando um valor de V_{DD} = *0.6V* para porta *NAND* de duas entradas e utilizando os transistores FinFET *high*-V_T e um FinFET *regular*-V_T.

Também, através de simulações elétricas, foi possível demostrar a viabilidade da implementação de um porta lógica com um número maior de entradas, como exemplo a *NAND* de seis entradas. No caso do uso da tecnologia MOSFET a implementação se torna indesejável, pois se faz necessário seis transistores em série. Isto ocasiona um grande atraso causado pela resistência dos seis transistores associados no *stack*. Porém, utilizando a implementação proposta pelos autores, uma porta *NAND* de seis entradas contém apenas três transistores (IG) FinFET *high*-V_T associados em séries e três transistores (IG) FinFET *regular*-V_T associados em paralelo, como mostra a Figura 7.

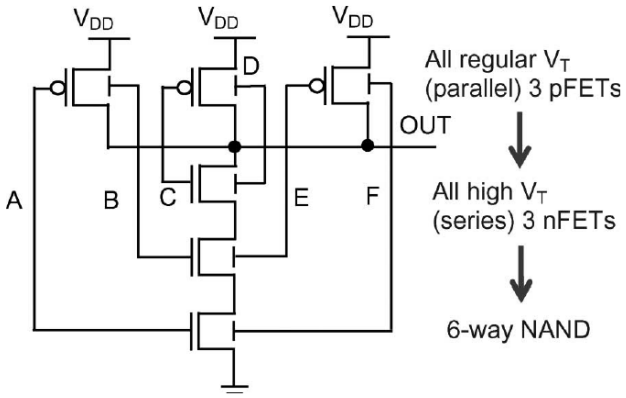


Figura 7 - Nova implementação da porta lógica NAND de seis variáveis com apenas seis transistores (IG) FinFET. (CHIANG, KIM, *et al.*, 2006)

Com isso, o número total de transistores em uma rede pode ser amplamente reduzido através do uso dos transistores propostos por Chiang et. al. devido aos agrupamentos em série e em paralelo.

2.2.3 Datta et al. (DATTA, GOEL, *et al.*, 2007)

Datta et al. em 2007 apresentaram diferentes implementações de portas lógicas *NAND*, *NOR* e Inversor *low power* com transistores (IG) FinFET. Algumas das implementações utilizam os dois *gates* para realizar os agrupamentos em paralelo, com o intuito de minimizar o número de transistores. Outras, utilizam um dos *gates* como *bias* para controlar o *threshold*.

Foram utilizados diversos circuitos do *benchmark* ISCAS85 com base em duas bibliotecas de células proposta pelos autores. A primeira biblioteca contém apenas três portas lógicas, *NAND*, *NOR* e Inversores. As três portas foram implementadas com diferentes tamanhos e usando apenas um único transistor (SG) FinFET. Já na segunda biblioteca foram utilizadas além das mesmas três portas da primeira biblioteca, mais cinco células *low power*. Todas as células foram implementadas com transistores (IG) FinFET.

Os autores compararam as células com transistores (IG) FinFET com as soluções baseadas com transistores (SG) FinFET. Os resultados mostraram que utilizando as células com os transistores (IG) FinFET foi possível obter uma redução considerável na dissipação de potência e na área dos circuitos.

2.2.4 Wang (WANG, 2010)

Wang apresentou um algoritmo para efetuar a síntese de circuitos, o qual explora o uso dos dois *gates* de um transistor (IG) FinFET. O autor também considerou a possibilidade de realizar agrupamentos entre dois sinais em série ou em paralelo usando apenas um transistor (IG) FinFET, ajustando a função do *threshold* do transistor.

A ideia inicial do algoritmo é utilizar como entrada um conjunto de mintermos da função e deriva-la para um dos planos da rede (ou plano *pull-up* ou plano *pull-down)*. Assim, todas as combinações possíveis de agrupamentos dos mintermos são geradas, dando origem a um conjunto chamado pelo autor de *Power Set* (PS). Após, todos os possíveis agrupamentos do conjunto PS, os quais podem ser implementados utilizando um transistor (SG) FinFET ou (IG)FinFET, são separados em um conjunto chamado SPS. Por fim, são realizadas uniões e intersecções disponíveis entre os elementos do conjunto SPS. O resultado é mapeado em arranjos em série e em paralelo. Deste modo, é possível obter um dos planos da rede. O outro plano é facilmente construído de forma dual.

O autor apresentou quatro novas soluções estruturais. Foi apresentada uma nova solução para uma porta majoritária, para um multiplexador de duas entradas, para uma *XOR* de três entradas e para um comparador. As novas soluções obtidas pelo método proposto, foram comparadas com CMOS padrão e também com *Pass Transistor Logic* (PTL). As redes que utilizaram transistores (IG) FinFET apresentaram uma maior redução em área e dissipação de potência. Já as redes

utilizando transistores (SG) FinFET apresentaram um aumento de 52% na dissipação de potência, quando comparadas com as redes (IG) FinFET. Entretanto, as redes com (SG) FinFET apresentaram um menor atraso do que as outras redes.

Embora o artigo apresente um algoritmo, o autor não deixou claro como o algoritmo foi implementado e, também, como as redes utilizadas nos resultados foram construídas.

2.2.5 Rostami et al. (ROSTAMI, M & MOHANRAM, K., 2011)

Os autores apresentam um transistor *high-*V_{th} que implementa um arranjo de dois sinais em série, semelhante ao apresentado em outros trabalhos (WANG, 2010) (CHIANG, KIM, *et al.*, 2006). Apresentam, também, um transistor *low-*V_{th} que implementa um arranjo de dois sinais em paralelo, semelhante a um *regular-*V_T apresentado em (CHIANG, KIM, *et al.*, 2006).

A partir desta possibilidade de obter esses arranjos, os autores propõem duas técnicas de defatoração que tem como objetivo encontrar bons agrupamentos série e paralelo para utilizar os transistores (IG) FinFET. A técnica proposta tende a reduzir o número de transistores da rede, desta forma, resultando em uma redução de área e consumo da rede. Com isso, foram propostas três bibliotecas de células *low power*, que foram desenvolvidas com transistores *high-*V_{th} e *low-*V_{th}. A primeira biblioteca consiste de portas convencionais como *NOT, AND2, OR2, NAND3, NOR3*, entre outras, onde todas foram implementadas apenas com transistores (SG) FinFETs. A segunda biblioteca contém 41 células, que é composta por células desenvolvidas através de agrupamentos de transistores paralelos e do desligamento de um dos *gates* propostas em (MUTTREJA, AGARWAL e JHA, 2007) (DATTA, GOEL, *et al.*, 2007). A terceira e última, integra as outras duas bibliotecas, totalizando 135 células, que foram desenvolvidas utilizando transistores (SG) FinFETs e também (IG) FinFETs *high-*V_{th} e *low-*V_{th} através da técnica de defatoração propostas pelos autores. Os autores apresentaram, através de alguns experimentos, uma redução significativa em consumo de energia e também em área dos circuitos.

Porém, os autores não deixaram claro como os arranjos das células que compõem as bibliotecas foram gerados. Se os arranjos foram gerados automaticamente pela técnica de defatoração proposta ou se foram gerados manualmente.

2.2.6 Alioto (ALIOTO, 2011)

Alioto apresenta uma interessante discussão sobre questões relacionadas ao projeto, *layout* e densidade de células que utilizam transistores FinFET. Neste trabalho, o autor estende significativamente as análises realizadas em trabalhos anteriores, nos quais foram consideradas células simplistas de um único dispositivo FinFET e também circuitos extremamente simples (ALIOTO, 2009) (ALIOTO, 2010).

Então, o autor utiliza uma biblioteca composta por Inversores, *NAND2, NAND4, NOR2, NOR4 e AOI*. Também é apresentado o *layout* do *carry* de um somador de 32-bit. Todos os *layouts* das células foram construídos utilizando transistores (SG) FinFET, (IG) FinFET, células mistas (MT) que possuem tanto o transistor (SG) quanto o (IG) FinFET e, também, transistores MOSFET para meio de comparação. Os *layouts* foram desenhados considerando as regras de tecnologias para 32, 45 e 65 nanômetros. Diferentes alturas para os *fins* também foram consideradas.

Os resultados apresentados pelo autor mostram que as análises anteriores com base em um único dispositivos não são necessárias para se ter uma ideia sobre a densidade real da célula, devido às restrições de espaçamento adicionais impostas pela estrutura. A análise também mostrou que as células com transistores (SG) FinFET, considerando uma altura para os *fins* baixa, apresentam uma densidade no *layout* comparável com a MOSFET. Caso seja considerado uma altura para os *fins* moderadamente mais alta, é possível obter uma significante melhora na densidade do *layout*. Porém, as células com transistores (IG) FinFET apresentaram um grande aumentos na densidade do *layout*. Basicamente isso ocorre devido ao fato de ter que dispor os contatos separados dos dois *gates* em um transistor.

Com isso, acaba-se gerando espaços indesejáveis no *layout* devido as regras do projeto. Portanto, células com transistores (IG) FinFET dificilmente serão utilizadas em modelos reais. No caso das células MT, que contêm tanto transistores (SG) FinFET quanto (IG) FinFET, com alturas dos *fins* moderados, os resultados apresentaram uma densidade comparável com a MOSFET. Entretanto, as células MT com altura de *fins* mais elevadas apresentaram uma melhoria significativa na densidade.

2.2.7 Possani (POSSANI, 2015)

Possani inicialmente apresenta uma ampla analise com diversos estudos de casos provando e demonstrando a existência de uma mudança de paradigma que está sendo introduzida dentro da síntese lógica devido ao uso dos transistores (IG) FinFET *double gate*. O autor discute que os métodos de geração automática de redes de transistores existentes na literatura, com o intuito de minimizar o número total de transistores, baseiam-se na minimização do número de literais que implementam a função lógica. Contudo, ao se considerar transistores (IG) FinFET *double gate*, esta abordagem de minimizar o número de literais não funciona corretamente para se obter uma rede menor.

Um dos estudos de casos apresentado pelo autor demonstra que apenas minimizar o número de literais em uma expressão Booleana não garante uma rede com número mínimo de transistores (IG) FinFET *double gate*. Este estudo de caso é apresentado a seguir.

Considere a função lógica *f*, a qual pode ser descrita de duas maneiras distintas. Uma delas é composta por 14 literais e a outra por 15 literais, como apresentada na Equação (1) e Equação (2), respectivamente.

$$f = (!a + ((!c + !d) \,.\, (b + (c + d)))) \,.\, ((a \,.\, !b) + ((!c + d) \,.\, (c + (!d \,.\, (a + !b))))) \quad (1)$$

$$f = ((!b \,.\, a) \,.\, ((!d \,.\, c) + (d \,.\, !c))) + ((!d \,.\, !c) \,.\, ((!b \,.\, !a) + (b \,.\, a))) + (!a \,.\, (d \,.\, c)) \quad (2)$$

Caso essas duas formas fatoradas fossem utilizadas para construir redes de transistores compostas por transistores MOSFET ou (SG) FinFET, as redes obtidas a partir da Equação (1) e Equação (2), seriam compostas por 14 e 15 transistores, respectivamente. Porém, considerando dispositivos com transistores (IG) FinFET *double gate*, o resultado não seria o mesmo.

Como pode ser notado na Figura 8(b), a Equação (1) é composta por 9 transistores (IG) FinFET *double gate*. No entanto, com a Equação (2) é possível construir uma rede com 8 transistores (IG) FinFET *double gate*, como mostra a Figura 8(c). O autor representa os transistores, (SG) FinFET, (IG) FinFET (paralelo) e (IG) FinFET (série) de acordo com as notações ilustrados na Figura 8(a), respectivamente.

Neste caso o autor demonstra que a quantidade de literais presentes na função não está diretamente relacionada com a quantidade de transistores (IG) FinFET *double gate* presentes na rede.

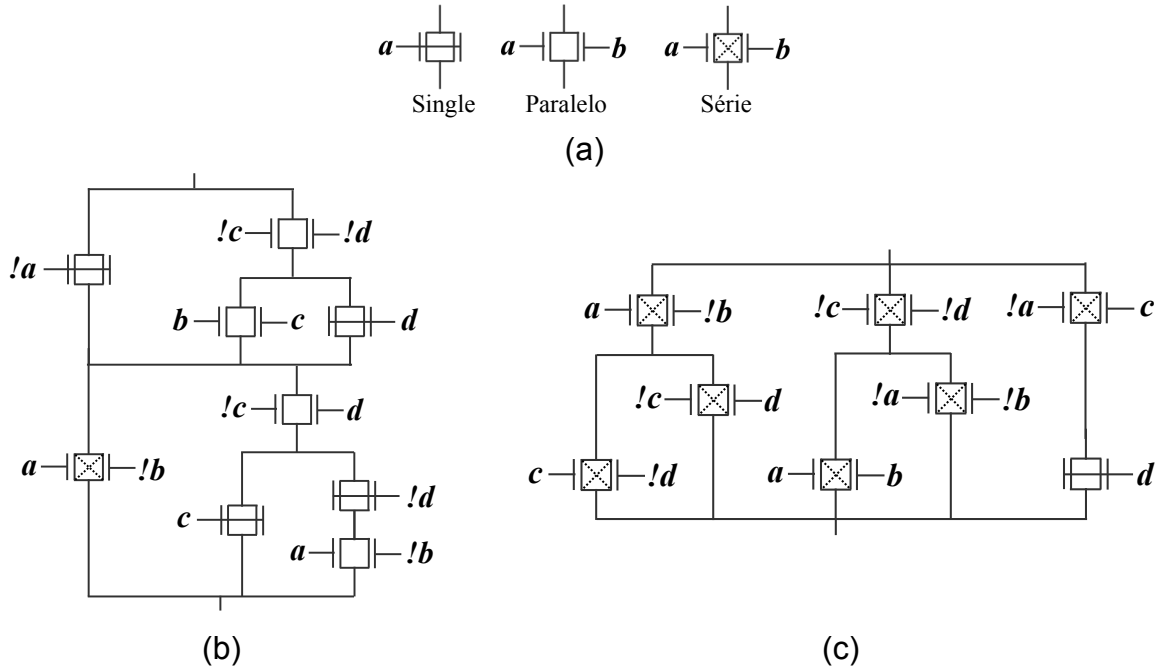


Figura 8 - (a) representação dos transistores (SG) FinFET, (IG) FinFET (paralelo) e (IG) FinFET (série) ,(b) um plano da rede obtida a partir da Equação (1), (c) e uma plano da rede obtida a partir da Equação (2). (POSSANI, 2015)

Neste sentido, o autor propõe dois métodos alternativos para geração automática de redes de transistores baseadas no uso de dispositivos (IG) FinFET *double gate*. Um dos métodos é baseado em grafos, com o intuito de encontrar padrões de arranjos os quais podem ser agrupados. Já o segundo método utiliza técnicas de defatoração em expressões Booleanas. Os dois métodos tem por objetivo explorar o potencial dos transistores (IG) FinFET *double gate*, com o intuito de minimizar o número total de transistores na rede final gerada automaticamente.

Os resultados apresentados mostram que os dois métodos propostos são capazes de fornecer células otimizadas com transistores (IG) FinFET *double gate*, com um baixo custo no tempo de execução. Além disso, o autor mostra, através de experimentos, que os métodos convencionais de geração de redes de transistores existentes na literatura não são ideais para serem utilizados em células baseadas em transistores (IG) FinFET *double gate*.

3 COMPOSIÇÃO FUNCIONAL

Composição Funcional consiste em uma metodologia dedicada à síntese lógica proposta em 2009 (REIS, 2009). Algumas versões adaptadas e/ou evoluidas foram propostas em 2010 (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (FIGUEIRO, 2010) e 2012 (MARTINS, M., RIBAS, R., AND REIS, A, 2012).

Diferente de algumas abordagens encontradas na literatura, como, por exemplo, a Decomposição Funcional (ASHENHURST, 1959) (CURTIS, 1962), a Composição Funcional realiza uma associação *bottom-up* de funções Booleanas. Essa característica, resulta em alguns fatores que auxiliam no processo de síntese. Como exemplo, os custos inicias das funções são conhecidos; as operações lógicas realizadas são simples; as sub-funções geradas resultam em implementações sub-ótimas; e, também, um custo de controle pode ser facilmente definido.

A Composição Funcional é baseada nos seguintes princípios: (1) a representação da função lógica por um par funcional/estrutural; (2) o uso de funções iniciais com um custo conhecido; (3) a associação entre funções simples para criar funções complexas; (4) o controle dos custos obtidos usando uma ordem parcial que permite o uso de programação dinâmica; (5) o uso de um conjunto de sub-funções “permitidas” para reduzir o tempo de execução e memória. Esses princípios são discutidos nas próximas subseções.

3.1 Representação de Pares Funcional/Estrutural

A Composição Funcional utiliza um par composto por {*funcionalidade, estrutural*} para representar funções Booleanas. Esse par é basicamente uma estrutura de dados que contém uma representação funcional e uma representação estrutural da mesma função Booleana.

A representação funcional necessita ser uma representação canônica, pois assim é possível evitar as dependências estruturais. Deste modo, a Composição

Funcional pode ser considerada um método Booleano. Exemplos de representação funcional são: estruturas de ROBDD (*Reduced and Ordered Binary Decision Diagram*) (BRYANT, 1986); tabela verdade; ou tabelas verdades representadas como inteiros (REIS, 2009).

A representação estrutural pode ser um nodo raiz de uma árvore de operadores ou uma cadeia de caracteres com a forma fatorada que representa a função Booleana. Além disso, algumas outras informações podem ser armazenadas, como a quantidade de literais, profundidade lógica, número de transistores e propriedades série/paralelo da função (REIS, 2009).

A Figura 9 ilustra um exemplo de um par funcional/estrutural, onde a representação funcional é feita pelo valor de uma tabela verdade representado por um número inteiro, considerando o bit mais significativo à esquerda. A parte estrutural é representado por um expressão.

1011_2 (*funcional*)	!a + b (*estrutural*)

Figura 9 - Exemplo de um par funcional/estrutural.

3.2 Funções Iniciais

A Composição Funcional gera novas funções a partir das associações de funções conhecidas. Portanto, um conjunto de funções iniciais são necessárias para que o algoritmo comece. O conjunto de funções iniciais deve ter duas características. Primeiro, o par funcional/estrutural deve ser simples de ser calculado. Em segundo, as funções iniciais devem apresentar um custo de cada uma das funções, pois assim é possível realizar o cálculo do custo para as próximas funções derivadas. O conjunto das funções iniciais pode variar de acordo com o propósito especifico no qual será utilizada a Composição Funcional. A Figura 10 apresenta um exemplo de conjunto de funções inicias para uma função que contem duas variáveis, utilizando o mesmo tipo de par funcional/estrutural ilustrado na Figura 9.

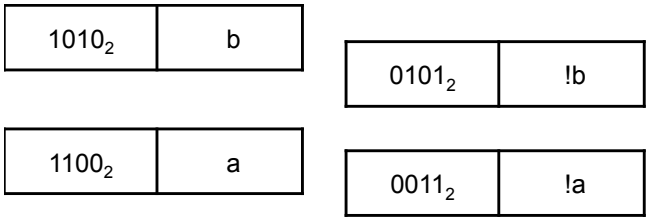


Figura 10 - Funções inicias representadas pelo par funcional/estrutural.

3.3 Associação de Pares Funcional/Estrutural

Como já foi mostrado anteriormente, a Composição Funcional utiliza um par funcional/estrutural para representar uma função. Portanto, quando uma operação lógica (como exemplo, a operação lógica *AND*) deve ser aplicada na função, essa operação acaba sendo aplicada de forma independente na parte funcional e na parte estrutural do par que está associado a função. A principal vantagem dessa associação dos pares é que, para calcular operações entre representações do mesmo tipo, se torna muito mais rápida do que realizar conversões entre parte funcional e estrutural ou vice-e-versa. A Figura 11(a) apresenta a associação de um par funcional/estrutural. O par <F3,S3> é obtido a partir dos pares <F1,S1> e <F2,S2>. O cálculo da parte funcional (F3 = F1 * F2) é independente do cálculo da parte estrutural (S3 = S1 * S2). O conceito pode ser expandido para executar operações complexas, tanto na parte funcional quando na estrutural, como pode ser visto na Figura 11(b).

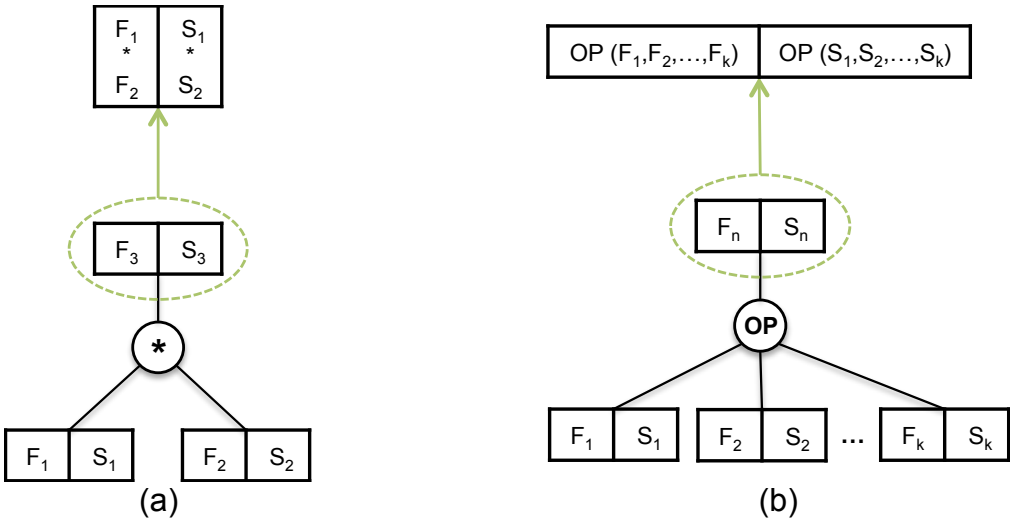


Figura 11 - Exemplo de associação de pares: (a) usando uma operação lógica OR entre dois elementos; (b) utilizando operações complexas entre k elementos.

3.4 Ordem Parcial e Programação Dinâmica

A ideia básica da programação dinâmica consiste em resolver problemas computacionais em que a solução ideal é obtida através de combinações de soluções sub-ótimas, evitando, assim, o recálculo de subproblemas. Deste modo, esta técnica começa resolvendo subproblemas e, em seguida, as soluções desses subproblemas são combinadas para se obter a solução completa. Dentro da Composição Funcional, a programação dinâmica é associada ao conceito de ordem parcial. Ordem parcial é utilizada para classificar os custos das soluções intermediarias, com o intuito de garantir que as implementações dos pares com custo mínimo sejam utilizados para os subproblemas. Assim, para utilizar o conceito de ordem parcial, soluções intermediarias dos subproblemas são classificados em "baldes", em ordem crescente de acordo com o custo estrutural do par funcional/estrutural. Os baldes são calculados na ordem crescente dos custos. Essa ideia é apresentada pela Figura 12.

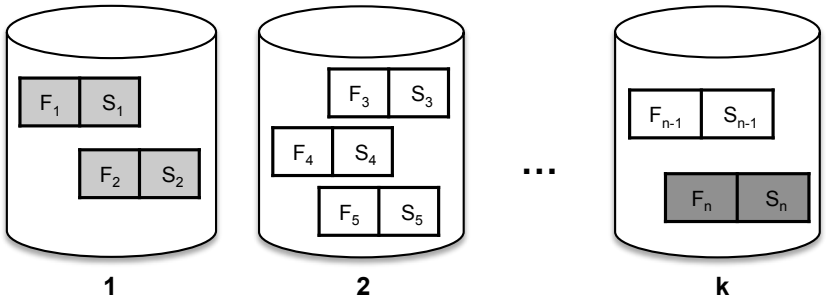


Figura 12 - Baldes da Composição Funcional. Os pares cinza claro são as funções inicias, os pares brancos são as funções intermediarias e o par cinza escuro é a função alvo, a qual foi localizada no k-balde.

3.5 Sub-Funções Permitidas

A Composição Funcional pode se tornar algo muito custoso devido à grande quantidade de funções intermediarias que são criadas através de combinações exaustivas. Portanto, algumas otimizações de desempenho devem ser empregadas com o intuito de tornar a abordagem viável e mais eficiente. Uma dessas otimizações é o uso de sub-funções permitidas. A ideia é utilizar uma tabela que possa ser pré-calculada antes de iniciar o algoritmo. Essa tabela será composta pelas sub-funções permitidas. Isso significa que as funções que não estão presentes

nesta tabela são descartadas durante o processamento. Desta forma, o conjunto de funções que serão manipuladas será reduzido, permitindo assim o controle do tempo de execução do algoritmo. Porém, a quantidade de funções presentes na tabela, pode ou não, influenciar no resultado final do método. Em alguns casos, a Composição Funcional atinge melhores resultados de acordo com a quantidade de sub-funções permitidas. Vários níveis de otimizações podem ser implementados para melhorar o tempo de execução e memória versus a qualidade do resultado (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012). A vista disso, os níveis de otimização podem variar da utilização de um conjunto reduzido de funções até a um esforço exaustivo que inclui todas as funções possíveis.

3.6 Fluxo Geral e Aplicações

A Figura 13 apresenta o fluxo geral do algoritmo contendo os princípios da Composição Funcional. O primeiro passo consiste em analisar a função de entrada. Após, os pares funcional/estrutural iniciais são gerados e comparados com a função de entrada para verificar se a função alvo já foi encontrada. Caso a função alvo não tenha sido encontrada, as funções permitidas são computadas e armazenadas em um conjunto, que são utilizadas para descartar as funções indesejáveis. Posteriormente, os pares funcional/estrutural iniciais são inseridos no primeiro balde. Assim, é possível realizar a associação entre os pares para compor novos pares, que serão inseridos nos próximos baldes, de acordo com seus custos. Esses novos pares serão utilizados para as próximas associações. Dessa forma, o processo continuará até que a função alvo seja encontrada.

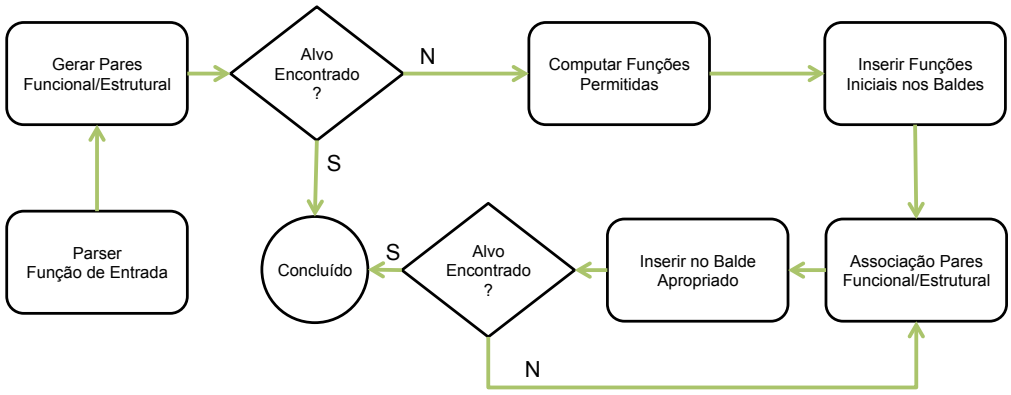


Figura 13 - Fluxograma geral da Composição Funcional (MARTINS, M., RIBAS, R., AND REIS, A, 2012).

A Composição Funcional utiliza essa analogia de baldes para controlar os custos dos pares gerados. Pensando em geração de dispositivos *single gate*, o custo de uma função pode ser associado à quantidade de literais presentes na mesma. Portanto, os pares funcional/estrutural que estão presentes no primeiro balde, apresentam custo um. Então, o balde 1 consiste de pares funcional/estrutural que representam funções com apenas um literal. Para gerar o balde 2, realiza-se a combinação entre os pares funcional/estrutural que compõem o balde 1, através de alguma operação especifica, exemplo *AND* ou *OR*. Assim, o balde 2 será preenchido por pares que representam funções com dois literais. De forma similar, a associação entre os pares dos baldes 1 e 2, preenchem o balde 3. O balde 4, é composto pela associação dos pares dos baldes 1 e 3 e entre os pares do balde 2. Para compor o balde 5, utiliza-se as associações entre os pares do balde 1 e 4 e dos baldes 2 e 3. A Figura 14 ilustra o processo para criar todos os baldes até sub-funções que apresentam custo cinco. Desde modo, a geração do balde *n-literais* pode ser expressa pela Equação 3.

$$B_n = \cup_{i=1}^{\frac{n}{2}} \left((B_i * B_{n-1}) \cup (B_i + B_{n-1}) \right) \mid n \geq 2 \qquad (3)$$

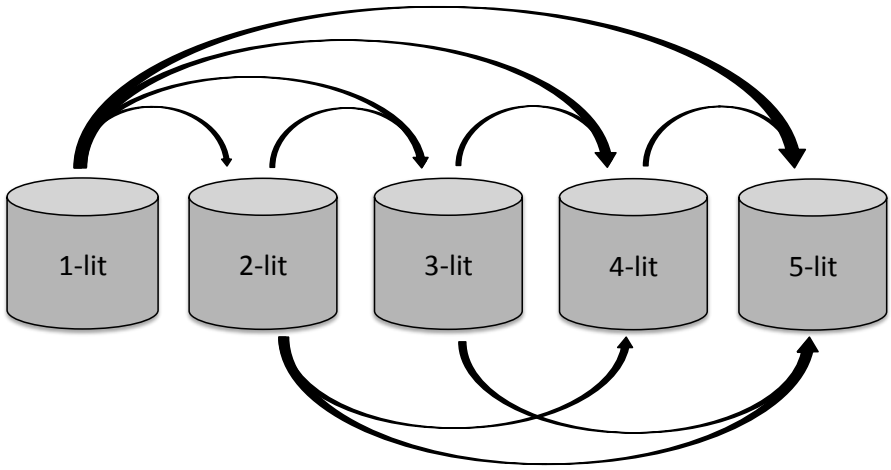


Figura 14 - Geração de sub-funções até o balde 5.

Como aplicações, a Composição Funcional pode ser utilizada para diversos propósitos:

- Fatoração Booleana, incluindo fatoração exata e fatoração considerando operadores (AND, OR, XOR e NOT) (MARTINS, M., RIBAS, R., AND REIS, A, 2012);
- Síntese lógica QCA (*Quantum Dots Cellular Automata*), considerando portas lógicas majoritárias e inversores (MARTINS, M., RIBAS, R., AND REIS, A, 2012);
- Computação de cadeias mínimas de decisão (MARTINS, DA ROSA JUNIOR, *et al.*, 2010);
- Redução na construção de AIG (*And-Inverter Graph*) (MARTINS, DA ROSA JUNIOR, *et al.*, 2010).

Para cada aplicação, o algoritmo é obtido escolhendo adequadamente a representação dos pares funcional/estrutural, as funções inicias, as associações dos pares, uma ordem parcial apropriada associada à programação dinâmica e, também, as funções permitidas, que se adequam ao problema proposto.

4 ESTUDO DE CASO

Este capítulo retrata dois estudos de caso que contribuíram como motivação para a realização deste trabalho.

Em geral, uma rede de transistores pode ser gerada de diversos modos e com diferentes estilos lógicos. Neste trabalho, o objetivo é gerar redes de transistores para implementar portas lógicas baseada em dispositivos (IG) FinFET *double gate*. Para simplificar os estudos de caso, nos exemplos a seguir consideramos apenas um dos planos de uma porta lógica.

Nos dois estudos utilizou-se, para geração das redes de transistores, o método de fatoração proposto por Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012), o qual consiste em um método de fatoração dedicado a dispositivos *single-gate* e baseado na metodologia de Composição Funcional. Portanto, as redes foram geradas tradicionalmente e, quando possível, foram realizados agrupamentos em série ou em paralelo dos transistores, representando, assim, um transistor (IG) FinFET *double gate*. Isto foi feito com o propósito de compreender os impactos introduzidos pelo uso do transistor (IG) FinFET *double gate*, devido ao agrupamento dos transistores em série e/ou paralelo.

As notações para transistores (SG) FinFET, (IG) FinFET paralelo e (IG) FinFET série são ilustradas, respectivamente, na Figura 15. Elas serão utilizadas a partir deste ponto no restante deste trabalho.

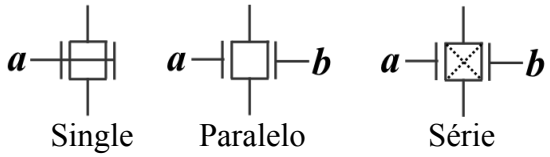


Figura 15 - Notação usada para o transistor (SG) FinFET (*single*), (IG) FinFET paralelo e (IG) FinFET série.

Caso 1: Considere a função *f1*, representada pela Equação (4). Essa equação pode ser fatorada de forma ótima através do método proposto por Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012). A fatoração resulta na função apresentada pela Equação (5).

$$f1 = (!a * c * d) + (!a * b * d) + (a * !b * !c) + (a * !b * !d) \quad (4)$$

$$f1 = (a + (d * (b + c))) * (!a + (!b * (!c + !d))) \quad (5)$$

A rede de transistores obtida a partir da função *f1*, representada pela Equação (5), na qual a rede é composta por dispositivos *single gate*, ou seja, transistores MOSFET e/ou (SG) FinFET, é ilustrada pela Figura 16(a). Nota-se que a função resultante é composta por 8 literais, resultando em uma rede com 8 transistores *single gate*.

No entanto, ao considerar a possibilidade de construir a rede utilizando transistores (IG) FinFET *double gate* deve-se, então, avaliar as possibilidade de realizar todos os agrupamentos possíveis em série e/ou paralelo entre dois transistores. A partir da Figura 16(a) é possível perceber que existem apenas duas possibilidade de agrupamentos. Primeiramente é possível realizar os dois agrupamentos em paralelo que estão indicados pelos círculos pontilhados na Figura 16(a). A rede resultante após os agrupamentos em paralelo, utilizando transistores (IG) FinFET *double gate*, é apresentada pela Figura 16(b). Esta solução possibilita uma redução de dois transistores, quando comparação com a rede inicial da Figura 16(a).

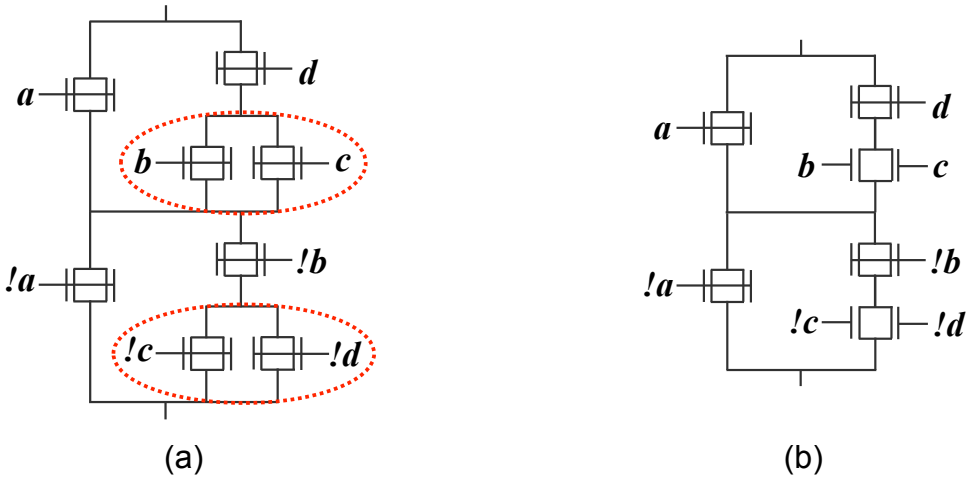


Figura 16 - (a) rede de transistores *single gate* obtida pela Equação (5), indicando os possíveis agrupamentos; (b) rede de transistores (IG) FinFET, após os agrupamentos.

Porém, considerando apenas os caminhos que propagarão o sinal de um terminal a outro da rede da Figura 16(b), é possível verificar que existe outro modo de construí-la. Esses caminhos estão ilustrados pelas linhas tracejadas e pontilhadas da Figura 17(a). Essa outra representação estrutural proporciona novas possibilidades de agrupamento, ilustrado pela Figura 17(b). Os dois novos agrupamentos estão indicados pelos círculos pontilhados e tracejados na Figura 17(b). Neste caso, cada agrupamento resulta em uma combinação em série entre os transistores destacados. Portanto, após realizar os agrupamentos possíveis, a rede resultante é apresentada pela Figura 17(c). Nota-se que a rede é composta por apenas quatro transistores (IG) FinFET *double gate*, o que propicia uma redução de quatro e dois transistores, quando comparada com as redes da Figura 16(a) e Figura 16(b), respectivamente.

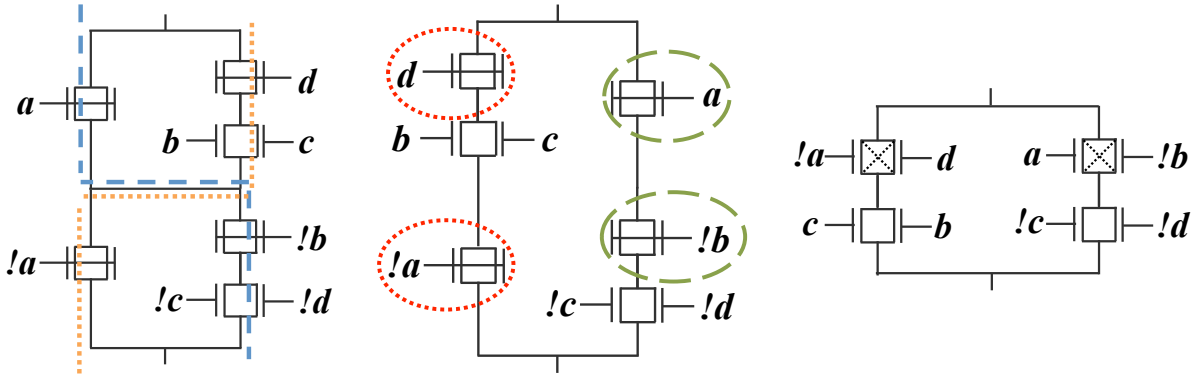


Figura 17 - (a) caminhos sensibilizados presentes na rede da Fig.16(b); (b) indicação dos possíveis agrupamentos em série; (c) rede resultante após os agrupamentos.

A Equação (6), mostra a função que representa a rede de transistores (IG) FinFET *double gate* da Figura 17(c).

$$f1 = ((!a * d) * (c + b)) + ((a * !b) * (!c + !d)) \quad (6)$$

Ao analisar a função representada pela Equação (6), é possível notar que ela é composta por quatro agrupamentos dois a dois. Portanto, se a função representada pela Equação (4) fosse fatorada a partir de agrupamentos *AND* e *OR* dois a dois, existe, assim, a chance de se obter como resultado a função ilustrada pela Equação (6).

Caso 2: Utilizando a função *f2*, representada pela Equação (7), como entrada para a fatoração proposta por Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012), tem-se como resultado a própria função *f2* indicada pela Equação (7).

$$f2 = f * (c + (!d * (!b + !e))) \quad (7)$$

A função *f2* pode ser utilizada para gerar uma rede de transistores com dispositivos *single gate*. Cada literal presente na função é mapeado para um transistor da rede, como mostra a Figura 18(a). Além disso, a função *f2* também pode ser representada através de uma rede com dispositivos (IG) FinFET *double gate*, como mostra a Figura 18(b). Nota-se que a rede da Figura 18(a) é composta por cinco transistores *single gate*. Mesma quantidade de literais presentes na função da Equação (7). Enquanto que a rede composta por dispositivos (IG) FinFET *double gate*, representada pela Figura 18(b), apresenta quatro transistores. Essa redução do número de dispositivos ocorre devido ao agrupamento realizado em paralelo indicado pelo círculo pontilhado na Figura 18(a).

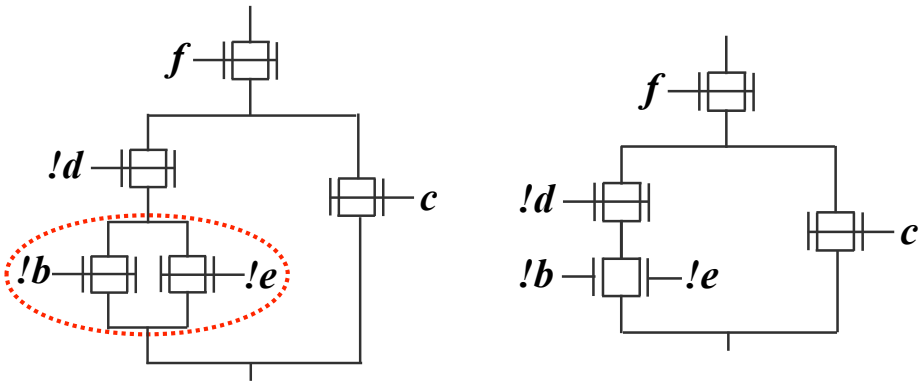
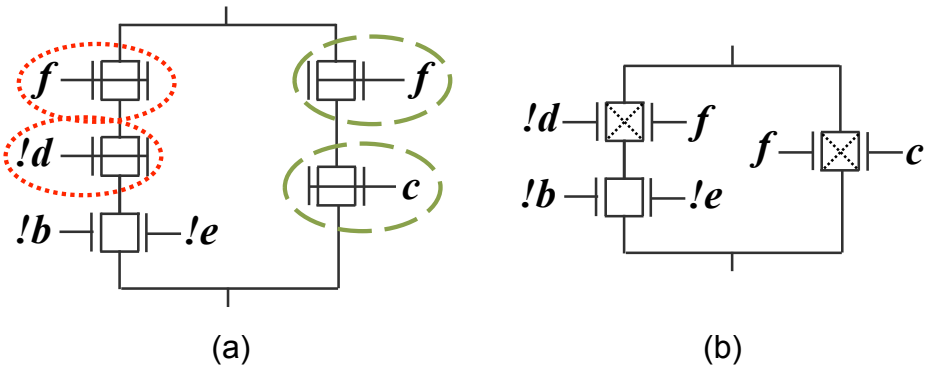


Figura 18 - (a) rede de transistores *single gate* obtida pela Equação (7), indicando o possível agrupamento; (b) rede de transistores (IG) FinFET, após o agrupamento.

Apesar da rede da Figura 18(b) ser composta por um número reduzido de dispositivos, essa ainda não apresenta a solução mais otimizada. Analisando a rede presente na Figura 18(b), é possível notar que a variável *f* está em série com os dois

caminhos sensibilizáveis da rede. Portanto, a duplicação da variável f resulta em uma rede na qual é viável realizar novos agrupamentos dois a dois, o que pode ser visto na Figura 19(a). Realizando os novos agrupamentos indicados pelos círculos pontilhados e tracejados obtém-se a rede apresentada pela Figura 19(b), composta por apenas três transistores (IG) FinFET *double gate*.



(a) (b)

Figura 19 - (a) novos possíveis agrupamentos; (b) rede otimizada referente a função representada pela Equação (8).

A rede da Figura 19(b) apresenta uma redução de um transistor (IG) FinFET *double gate* quando comparada com a solução mostrada na Figura 18(b). A função que representa a rede ilustrada pela Figura 19(b) é descrita pela Equação (8).

$$f2 = (f * c) + ((!d * f) * (!e + !b)) \quad (8)$$

Sendo assim, quando o objetivo é gerar redes de transistores com dispositivos *single gate*, deve-se minimizar o número de literais presentes na função. Então, é possível perceber que a função fatorada da Equação (7), composta por cinco literais, resulta em uma solução mais eficiente do que a função apresentada pela Equação (8), composta por seis literais. Todavia, quando o objetivo é gerar redes de transistores com dispositivos (IG) FinFET *double gate*, a função da Equação (8), apresenta uma melhor solução, mesmo contendo um literal a mais, quando comparada com a função da Equação (7).

Logo, a redução do número de literais em funções Booleanas não pode ser considerado com métrica quando o objetivo é explorar o uso dos transistores (IG) FinFET *double gate*. Esse é um caso semelhante ao apresentado por Possani (POSSANI, 2015) , onde o autor discute sobre as mudanças de paradigma ocasionadas pelo uso dos dispositivos *double gate*.

4.1 Conclusão

Os dois estudos de caso apresentados neste capitulo podem ser considerados como exemplos básicos que reforçam uma mudança de paradigma identificada por Possani (POSSANI, 2015). A ocorrência dessas situações é muito comum entre outras funções Booleanas. Utilizar como métrica a quantidade de literais durante o processo de geração de redes de dispositivos (IG) FinFET *double gate* pode não ser a melhor estratégia a ser tomada. Com isso, se torna claro a necessidade de se desenvolver novas técnicas para a geração de redes com transistores (IG) FinFET *double gate*.

5 MÉTODO PROPOSTO

Este capítulo apresenta o método proposto para a geração de redes de transistores dedicados a dispositivos (IG) FinFET *double gate*. O método é baseado na metodologia de Composição Funcional, devido ao fato de ser possível controlar critérios como o custo das funções e também quantidade de transistores em série e/ou paralelo.

5.1 Definições

Devido ao fato do método proposto ser baseado na metodologia de Composição Funcional, deve-se utilizar como base os cinco princípios apresentados no capítulo 3.

5.1.1 Sobre a Representação de Pares Funcional/Estrutural

A parte funcional do par é representada por uma tabela verdade notada como inteiros (POSSANI, SOUZA, *et al.*, 2012). Assim, torna-se mais fácil e rápido verificar a equivalência lógica e também aplicar operações entre funções. Devido ao fato da representação ser um número inteiro, as operações lógicas são realizadas através de operações *bit* a *bit* dentro da unidade lógica aritmética (ULA) do processador. Com isto, acaba-se evitando a utilização de estruturas mais complexas. A parte estrutural é representada por uma *string*, a qual representa a função Booleana.

5.1.2 Sobre as Funções Iniciais

As funções iniciais são o ponto de partida para a Composição Funcional. A partir das associações entre os pares das funções iniciais é que são geradas as próximas sub-funções que irão compor a função alvo. Além disso, as funções iniciais necessitam ter um custo. No método proposto, as funções iniciais apresentam custo um.

Assim, o método proposto considera como funções iniciais, todas as variáveis presentes na função de entrada. Então, são armazenadas no balde 1, as variáveis com suas devidas polaridades. Também, são armazenadas todas as combinações, dois a dois, entre essas variáveis utilizando as operações *AND* e *OR*. Esta particularidade tem como intuito possibilitar sub-funções capazes de representar um transistor (IG) FinFET *double gate*.

Portanto, o balde 1 é composto pelos pares que representam todas as possíveis funções com um e dois literais, como ilustrado na Figura 20.

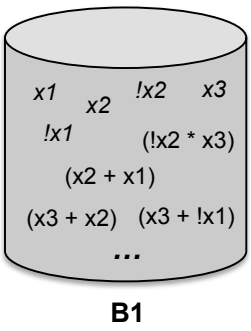


Figura 20 - Exemplo do balde 1 para uma função que contém três variáveis.

Apesar de um transistor (IG) FinFET *double gate* utilizar um agrupamento entre dois sinais, ainda existe um motivo especifico para considerar as funções com apenas um literal. Considere a função descrita pela Equação (9) como exemplo. Utilizando uma variação do método proposto, na qual apenas é considerado no balde 1 os pares que representam funções com dois literais. Neste cenário obtém-se como resultado a função apresentada pela Equação (10). Entretanto, usando o método proposto, que considera no balde 1 as funções com um literal, além das funções com dois literais, podemos chegar no resultado descrito pela Equação (11).

$$out = (!a * !c * !d) + (!a * !b) \quad (9)$$

$$out = ((!c + !b) * ((!b * !a) + (!d * !a))) \quad (10)$$

$$out = (!a * ((!b * !a) + (!c * !d))) \quad (11)$$

É possível notar que em ambos os resultados, tanto para a Equação (10) quanto para a Equação (11), as redes resultantes são construídas com apenas três

transistores (IG) FinFET *double gate*. Isto é ilustrado pelas Figura 21(a) e Figura 21(b).

Porém, se considerarmos os *layouts* de cada uma das redes, é possível perceber uma pequena diferença. A Figura 21(c) e a Figura 21(d) ilustram o *layout* das redes apresentadas na Figura 21(a) e Figura 21(b), respectivamente. É possível perceber que a rede gerada pelo método proposto, ilustrada pela Figura 21(b), apresenta um *layout* como uma área menor em comparação com o *layout* mostrado pela Figura 21(c). Isso acontece devido ao uso das sub-funções com apenas um literal no balde 1. O transistor (SG) FinFET apresenta uma área menor quando comparado a um transistor (IG) FinFET *double gate*. Neste caso, a quantidade de dispositivos FinFET é a mesma para ambos os resultados, 3 dispositivos.

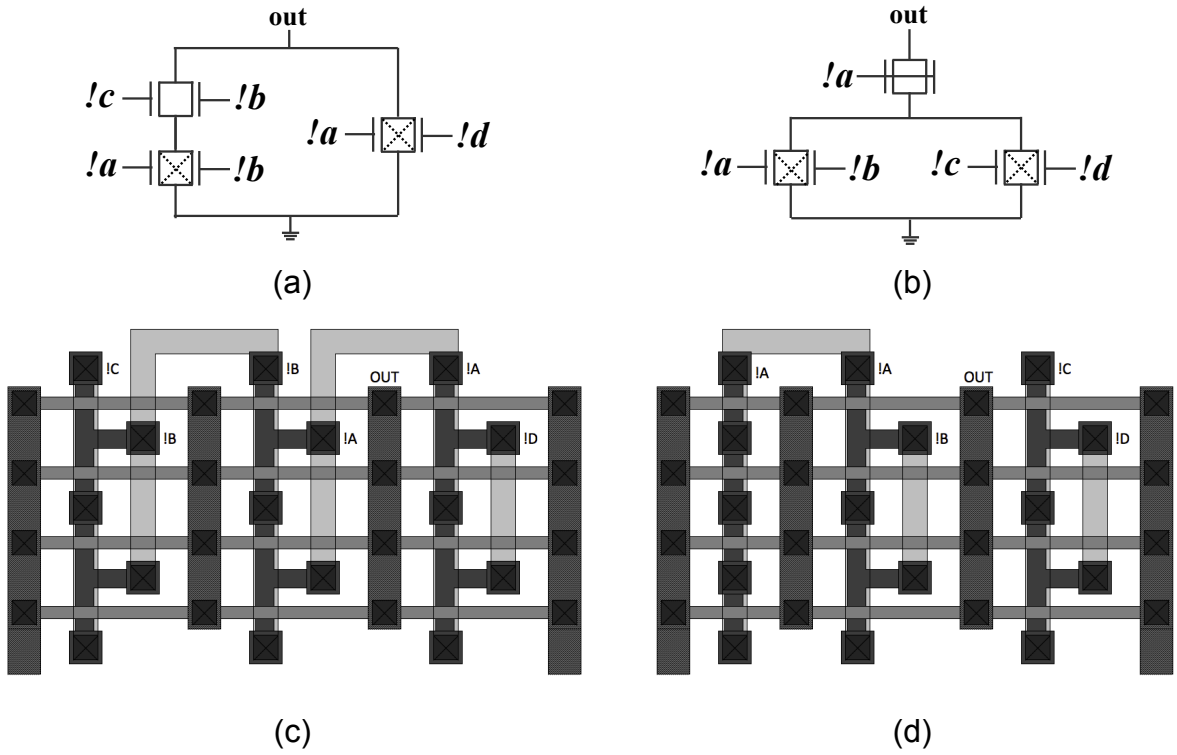


Figura 21 - (a) rede de transistores referente a Equação (10); (b) rede de transistores referente a Equação (11); (c) *layout* da função da Equação (10); (d) *layout* da função da Equação (11).

Este exemplo ilustra e justifica o motivo pelo qual o método proposto considera as funções com apenas um literal no balde 1.

Por fim, é importante mencionar que os *layouts* apresentados nas Figura 21(c) e Figura 21(d) são apenas rascunhos das redes, os quais foram desenvolvidos

de acordo com as regras de desenho apresentadas em (ALIOTO, 2011), ignorando qualquer regra de projeto comercial estabelecida pela indústria.

5.1.3 Sobre a Associação de Pares Funcional/Estrutural

As associações entre os pares funcional/estrutural são realizadas através de operações lógicas *AND* e *OR*. Deste modo, é possível representar os agrupamentos em série e em paralelo, podendo, assim, utilizar os transistores (IG) FinFET *double gate*.

Como critério de otimização do método, não se faz necessário a associação entre todos os pares funcional/estrutural de um mesmo balde. Em outras palavras, não é necessário realizar o produto cartesiano entre os pares do mesmo balde. Como exemplo, para construir o balde 4, é necessário realizar a associação entre os pares que estão no balde 3 com o balde 1. Ainda, é necessário a associação entre os próprios pares que estão no balde 2. A Figura 22 mostra a associação utilizando a operação *AND* entre os pares do balde 2, com o objetivo de construir o balde 4. Este exemplo ilustra o motivo pelo qual não é necessário a realização das associações entre todos os pares de um mesmo balde, pois isto resultaria em pares repetidos. Assim, este tipo de otimização auxilia na diminuição de pares desnecessários, diminuindo a quantidade de associações de n^2 para $\frac{n\,(n-1)}{2}$.

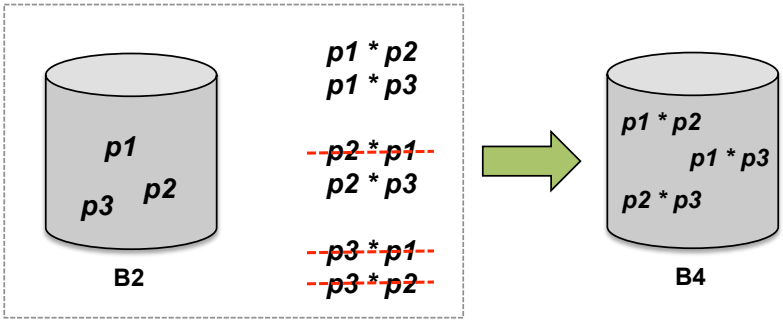


Figura 22 - Otimização na associação entre os pares de um mesmo balde.

5.1.4 Sobre a Ordem Parcial e Programação Dinâmica

Na Composição Funcional é possível obter soluções ótimas utilizando programação dinâmica associada a ordem parcial organizada em termos de custos. Portanto, o método proposto utiliza a associação dos baldes para classificar cada sub-função em um custo. Os baldes são organizados em ordem crescente, começando por um. Cada custo representa o custo real de utilizar um transistor (IG)

FinFET *double gate*. Assim, as funções presentes no balde *n*, utilizam *n* transistores (IG) FinFET *double gate*.

Seguindo essa ideia, para construir o balde 2 é necessário realizar a associação entre os pares funcional/estrutural que apresentam custo um. Portanto, é realizada a associação entre os pares presentes no balde 1. Para criar o balde 3, realiza-se a associação entre os pares do balde 1 com o balde 2. O balde 4 é gerado a partir das associações dos baldes 1 com 3 e do balde 2 com o 2. Assim, a criação do balde *n* é representada pela Equação 3 apresentada no capitulo 3.

5.1.5 Sobre as Sub-Funções Permitidas

O método proposto utiliza uma abordagem exata para gerar as funções. Portanto, todas as sub-funções precisam ser permitidas. Isto se deve ao fato, de que a quantidade de sub-funções permitidas utilizadas pode influenciar no resultado final da Composição Funcional. Portanto, o método proposto utiliza uma tabela que contém todas as sub-funções possíveis. Porém, o número de sub-funções que são geradas cresce exponencialmente em cada balde, tornando-se um método custoso em tempo de execução e uso de memória. Assim, algumas otimizações se tornam necessárias para eliminar com segurança as sub-funções repetidas que não contribuem para encontrar a solução mínima. Neste sentido, a tabela é preenchida de acordo com as sub-funções que são criadas pelo método proposto. Antes de uma sub-função ser gerada e adicionada em um balde, é feito a verificação se essa sub-função já está contida na tabela. Essa verificação é realizada através da associação entre a parte funcional dos pares das sub-funções que estão sendo combinadas. Caso a reposta seja positiva, a sub-função é descartada. Esta estratégia auxilia na diminuição da complexidade do método. Se a resposta for negativa, a sub-função é adicionada no balde e também na tabela. Desse modo, a tabela não contém sub-funções que são logicamente equivalentes, tento assim, apenas a ocorrência com o menor custo estrutural.

Caso seja necessário utilizar como entrada no método proposto um grande número de funções, a tabela é gerada apenas uma vez. Devido ao fato de que as sub-funções são geradas a partir de seu menor custo, é possível gerar todas as funções possíveis para o determinado conjunto que está sendo utilizado como entrada. O cálculo de todas as funções possíveis é feito através do número de entradas das funções presentes no conjunto. Portanto, todas as funções presentes

no determinado conjunto de entrada devem pertencer a mesma classe de funções, ou seja, ter a mesma quantidade de variáveis. Após gerada todas as funções possíveis, basta realizar uma busca na tabela através da parte funcional do par funcional/estrutural para retornar as funções alvo de acordo com as funções presente no conjunto. O único problema neste caso é que isso só se aplica a um conjunto que contém funções com no máximo quatro variáveis. Atualmente se torna inviável gerar e armazenar todas as funções possíveis para funções com mais do que cinco entradas. Devido ao fato de que a complexidade de memória para armazenar a tabela é de 2^{2^n}, onde n é a quantidade de entradas da função. Portanto, considerando funções com 5 entradas, existem mais de 4 bilhões de funções para serem armazenadas na tabela. Considerando que cada estrutura que represente uma função tenha 1 Kilobyte, seria necessário em torno de 4 Terabyte de memória.

5.2 Exemplificando o Método Proposto

Esta seção apresenta um simples exemplo do funcionamento básico do método proposto.

Considere a função f apresentada pela Equação (12) como entrada para o método proposto. Essa função está descrita em uma forma de SOP (Soma de Produtos).

$$f = (!a * c * d) + (!a * b * d) + (a * !b * !c) + (a * !b * !d) \quad (12)$$

Primeiramente é realizado um pré-processamento para verificar a quantidade de variáveis presentes na função de entrada e, também, para analisar se todas as variáveis encontram-se diretas ou negadas. Esta etapa é realizada para coletar as informações para construir o balde 1. Portanto, analisando a função presente na Equação (12) é possível notar que ela é composta por quatro variáveis e todas as variáveis ocorrem em ambas às polaridades. Deste modo, o balde 1 será composto por todos os pares funcional/estrutural que representam as variáveis, e também pelos pares que representam as combinações dois a dois, *AND* e *OR*, entre estas variáveis.

A cada geração de um par, antes dele ser adicionado a um balde, é verificado se este par não é a função alvo. Ou seja, a parte funcional do par da sub-

função tem que ser a mesma da parte funcional do par da função de entrada. Caso a sub-função seja a função alvo, o método proposto retorna como resultado a sub-função. Caso contrário, o método proposto continua a operação de busca. Além disso, a cada geração de um par, este par é adicionado em uma tabela de funções permitidas. Entretanto, deve-se verificar se um par com a mesma funcionalidade encontra-se na tabela. Caso este par já esteja presente na tabela, o par não é adicionado no balde. Caso contrário, o par é adicionado na tabela e também no balde referente ao seu custo.

A Figura 23 ilustra a representação do balde 1 para este exemplo. Com o intuito de deixar o exemplo mais simples de ser compreendido, todas as ilustrações das sub-funções dentro dos baldes serão apresentadas pelas próprias descrições das sub-funções. Porém, o método proposto armazena dentro dos baldes os pares funcional/estrutural.

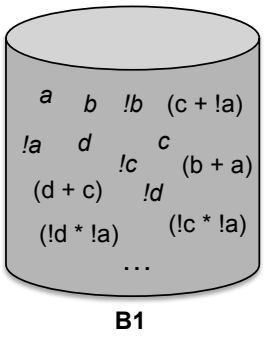


Figura 23 - Representação do balde 1, considerando a função apresentada pela Equação (12).

Após construído o balde 1, o método proposto segue o processo de busca gerando o próximo balde. O balde 2 é composto pelas sub-funções que apresentam custo dois, isto é, as sub-funções que contém apenas dois transistores (IG) FinFET *double gate*. Então, para gerar o balde 2, é usado os pares presente no balde 1. Assim, através das associações entre os próprios pares do balde 1, utilizando as operações *AND* e *OR*, são criados os pares que irão compor o balde 2. A Figura 24 ilustra este processo.

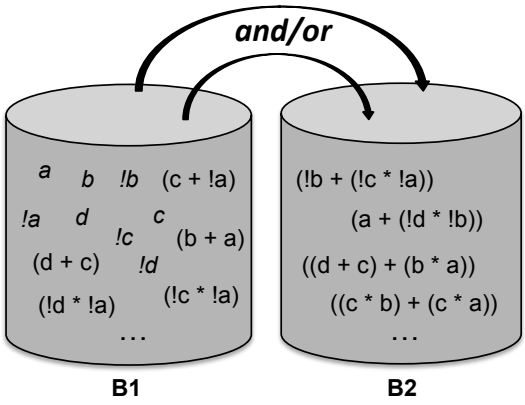


Figura 24 - Geração do balde 2.

Caso, a função alvo não tenha sido alcançada, o método proposto continua o processo de geração dos baldes. O próximo balde a ser construído é o balde 3, que contém as sub-funções que apresentam custo três. Assim, o balde 3 é preenchido através das associações, *AND* e *OR*, entre os pares dos baldes 1 e 2.

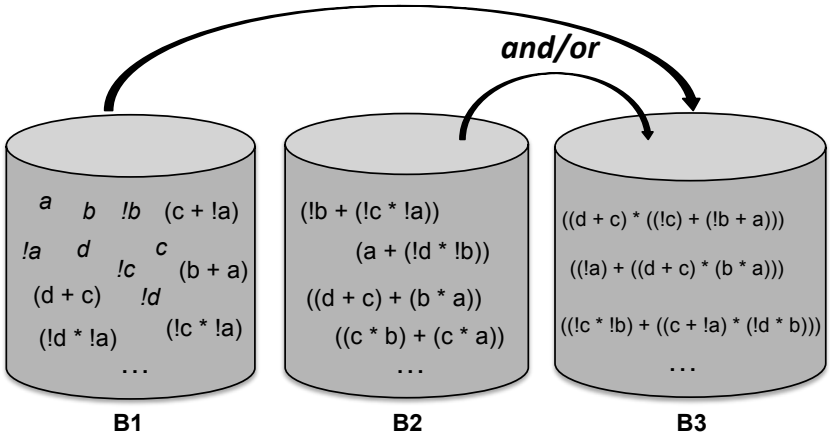


Figura 25 - Geração do balde 3, a partir da associação dos pares presentes no balde 1 e 2.

É importante lembrar que a cada geração de uma sub-função verifica-se se esta sub-função não é a função alvo. Neste caso específico, a função alvo ainda não foi encontrada. Logo, deve-se gerar o balde 4. A criação do balde 4 deve ser composta por pares que apresentam custo quatro. Assim, utiliza-se os pares do balde 1 com o balde 3, como mostra a Figura 26.

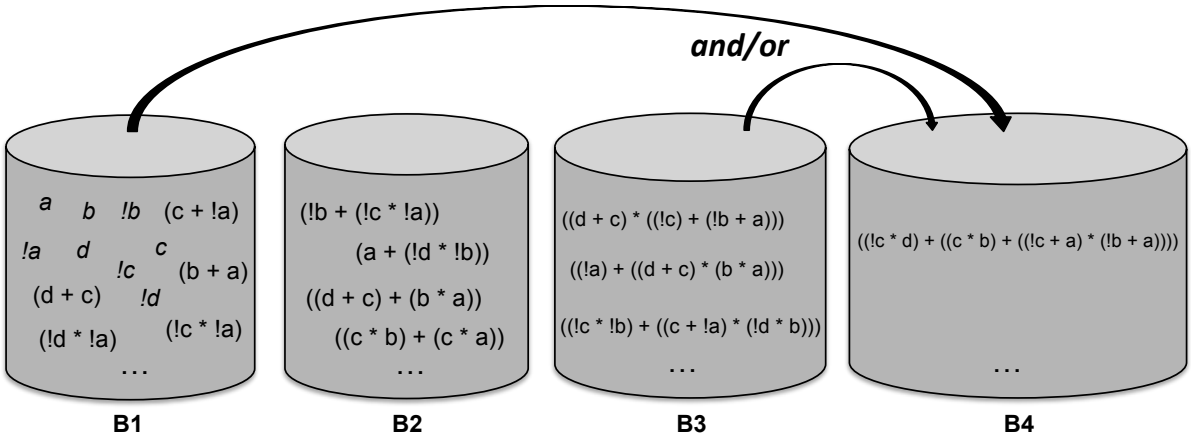


Figura 26 - Geração do balde 4, através da associação entre os pares dos baldes 1 e 3.

Caso não encontrado a função alvo, utiliza-se ainda, a associação entre os próprios pares contidos no balde 2, para completar o balde 4. A Figura 27 ilustra este processo. Neste caso acaba-se encontrando a função alvo.

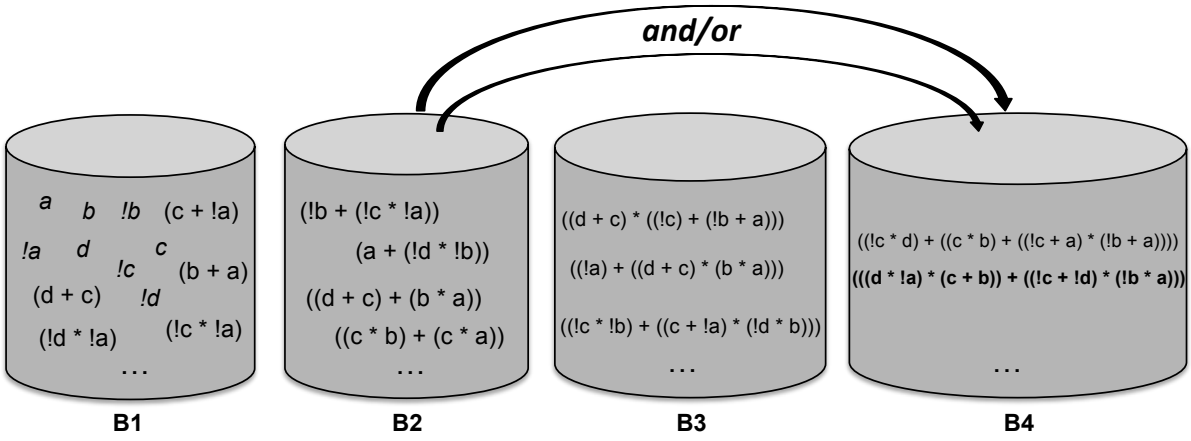


Figura 27 - Construção do balde 4, através da associação entre os pares presentes no balde 2.

Por fim, o método proposto, retorna a função alvo encontrada. Note que a função alvo é composta por quatro agrupamentos dois a dois. Deste modo ela pode ser construída por quatro transistores (IG) FinFET *double gate*.

6 RESULTADOS

Este capítulo apresenta os resultados obtidos pelo método proposto. Para avaliar a eficiência do método proposto efetuou-se alguns experimentos, onde foram utilizadas diferentes classes de funções Booleanas e também um conjunto de circuitos como *benchmarks*. O método proposto foi desenvolvido na linguagem Java, devido ao fato de que os métodos utilizados para comparação também terem sido desenvolvido nessa linguagem. Todos os experimentos foram executados em um computador Intel Core i5 3.2GHz com 8GB de memória RAM.

O primeiro *benchmark* utilizado foi o conjunto *P-class* de 4 entradas, o qual é composta por 3.982 funções Booleanas. Também foi utilizado um subconjunto da classe NPN (negação-permutação-negação) de 5 entradas. Esta classe é composta por 616.125 funções. Porém, foram utilizadas apenas 413 funções dessa classe. Como mencionado anteriormente, usar o método proposto para algumas funções acima de cinco variáveis se tornar inviável, devido ao estouro de memória no processamento. Do subconjunto da classe NPN, somente as 413 funções executaram até o final. Por fim, também foi utilizado um conjunto de 21 circuitos do *benchmark* ACM/SIGDA (IWLS, 2005). Esses *benchmarks* foram utilizados com o intuído de comparação com os resultados apresentados por Possani (POSSANI, 2015).

Os resultados obtidos com o método proposto foram comparados aos resultados obtidos pela técnica de Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012). Além disso, também foram comparados com os resultados gerados pelos dois métodos dedicados a dispositivos (IG) FinFET proposto por Possani (POSSANI, 2015). Vale lembrar que o método proposto por Martins foi projetado para fatoração e geração de redes com dispositivos *single gate*. Para este tipo de dispositivo o método é considerado como um dos métodos estado-da-arte. O motivo para utiliza-lo como comparação é para

demonstrar que esse método não apresenta bons resultados para redes baseadas em dispositivos (IG) FinFET *double gate*. Para realizar uma comparação justa com o método proposto por Martins, a cada rede gerada pelo método, foram realizados todos os agrupamentos possíveis, dois a dois, em série e/ou paralelo, para representar um dispositivo (IG) FinFET *double gate*.

O primeiro experimento foi realizado utilizando a *P-class* 4 entradas. Esse conjunto foi aplicado no método proposto neste trabalho. De modo que os resultados obtidos foram comparados à fatoração proposta por Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012) e os dois métodos proposto por Possani (POSSANI, 2015).

A Tabela 1 apresenta o número total de transistores (IG) FinFET *double gate* e o tempo total de execução obtidos por cada método avaliado, considerando como entrada o conjunto de funções presentes na *P-class* 4 entradas. Como pode ser verificado, o método proposto foi capaz de obter uma redução maior na quantidade de dispositivos (IG) FinFET *double gate* em relação aos outros métodos. Principalmente, obteve-se uma redução em relação aos dois métodos propostos por Possani, os quais são métodos dedicados à dispositivos (IG) FinFET *double gate.* Como era esperado, a fatoração proposta por Martins obteve o pior resultado. Apesar do método proposto apresentar um menor número de dispositivos para este conjunto, obteve o maior tempo de execução para gerar todas as 3.982 funções.

Tabela 1 - Total de número de transistores (IG) FinFET considerando o conjunto de funções da *P-class* 4 entradas.

Benchmark	***P-Class 4 entradas***			
Métodos	(MARTINS, M. et al., 2012)	Baseado em Grafo (POSSANI, V. 2015)	Defatoração (POSSANI, V. 2015)	**Método Proposto**
Número total de dispositivos (IG) FinFET	21.788	22.647	21.764	20.462
Tempo de Execução (s)	21,1	1,47	0,41	51,0

Para esta primeira experiência, dois histogramas são apresentados nas Figura 28 e Figura 29. A Figura 28 mostra os ganhos, perdas e empates em número

de transistores, para cada função, do método proposto quando comparado ao método baseado em grafos proposto por Possani (POSSANI, 2015). Os valores negativos representam a perda e os valores positivos representam o ganho.

Como pode ser visto na Figura 28, um pequeno subconjunto das funções geradas pelo método proposto apresentam um transistor a mais, em cada rede, em relação ao método baseado em grafo. Essa perda, indicada pelo valor -1, equivale a 0,05% de todo o conjunto. Os resultados que deram o mesmo número de transistores correspondem a 54,17% das funções. Já nos ganhos, que vão de 1 até 4 transistores a menos, o método proposto apresentou melhores resultados em 45,78% das funções.

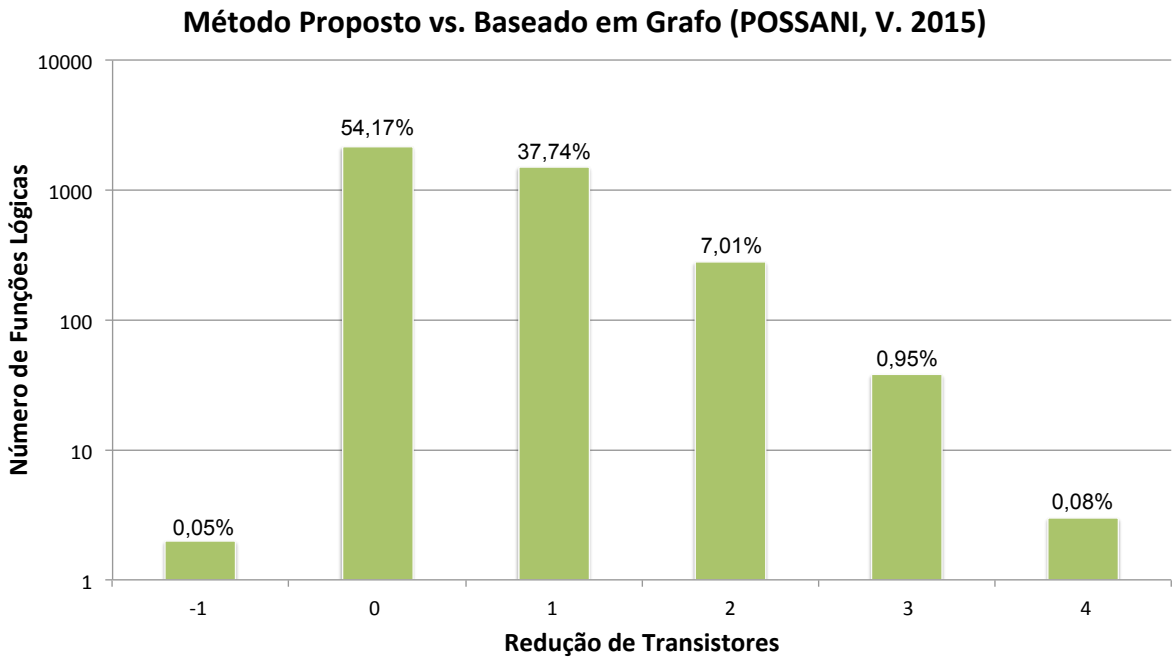


Figura 28 - Aumento e redução dos transistores obtidos pelo método proposto em relação ao método baseado em grafos (POSSANI, 2015), para o conjunto de funções da *P-class* 4 entradas.

A Figura 29 também apresenta algo semelhante à Figura 28. Contudo, a comparação é realizada entre o método proposto e o método de defatoração proposto por Possani (POSSANI, 2015). Através da Figura 29 é possível perceber que o método proposto apresentou bons ganhos em relação ao método de

defatoração. Para este conjunto, o método de defatoração proposto por Possani foi o que obteve melhores resultados entre seus dois métodos.

Os ganhos, que vão de 1 até 3 transistores a menos, correspondem a 33,13% das funções da *P-class* 4 entradas. Por outro lado, cerca de 0,30% das funções do conjunto gerado pelo método proposto apresentaram um transistor a mais. Já os empates, correspondem a 66,57% do conjunto.

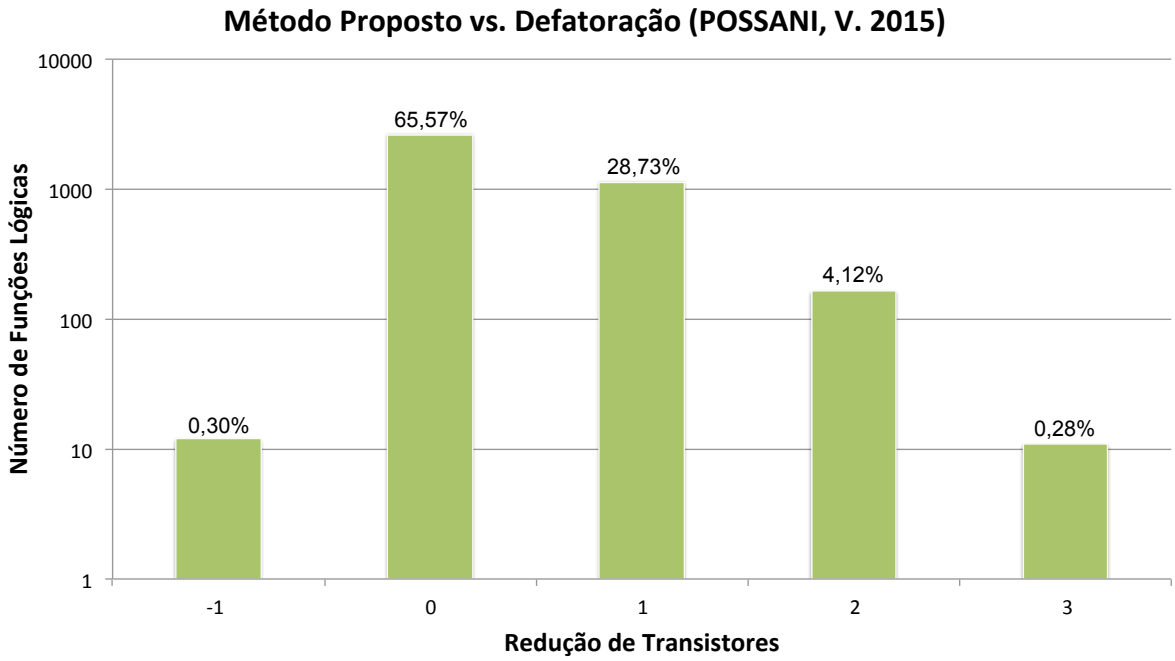


Figura 29 - Aumento e redução dos transistores obtidos pelo método proposto em relação ao método de defatoração (POSSANI, 2015), para o conjunto de funções da *P-class* 4 entradas.

O segundo experimento foi realizado utilizando a classe NPN de 5 entradas. Apenas 413 funções presente nesse conjunto foram possíveis de serem computadas no método proposto. Assim, somente estas 413 funções foram consideradas nos métodos propostos por Possani e por Martins.

A Tabela 2 apresenta os resultados em quantidade de transistores para este subconjunto de funções. Como pode ser visto, o método proposto mostrou uma menor quantidade total de transistores (IG) FinFET *double gate* em comparação aos outros métodos. Neste experimento não foi possível ter acesso ao tempo total de execução que cada um dos métodos, de Martins e de Possani, utilizou para gerar os

resultados para o subconjunto. Os autores não forneceram os tempos para cada função utilizada individualmente. O método proposto gerou os resultados para todo o subconjunto em 2,72 segundos.

Tabela 2 - Total de número de transistores (IG) FinFET considerando um subconjunto de 413 funções da classe NPN 5 entradas.

Benchmark	***Subconjunto de 413 funções da classe NPN-5***			
Métodos	(MARTINS, M. et al., 2012)	Baseado em Grafo (POSSANI, V. 2015)	Defatoração (POSSANI, V. 2015)	**Método Proposto**
Número total de dispositivos (IG) FinFET	2.163	2.106	2.109	1.931

Para este segundo experimento também realizou-se a verificação qualitativa dos resultados. As Figuras 30 e 31 ilustram os empates e os ganhos em número de transistores para as funções geradas pelo método proposto em relação aos resultados obtidos pelos métodos baseados em grafos e em defatoração, respectivamente.

Basicamente, o método proposto apresentou um ganho, de 1 até 2 transistores a menos, em 39,23% das funções, empatando em número de transistores em 60,77% das funções em relação ao método baseado em grafos. Estes resultados são ilustrados pela Figura 30.

A Figura 31 mostra que o método proposto gerou 62,47% da funções com o mesmo número de transistores quando comparado ao método de defatoração. Porém, apresentou um ganho, reduzindo de 1 até 2 transistores, em 37,53% das funções.

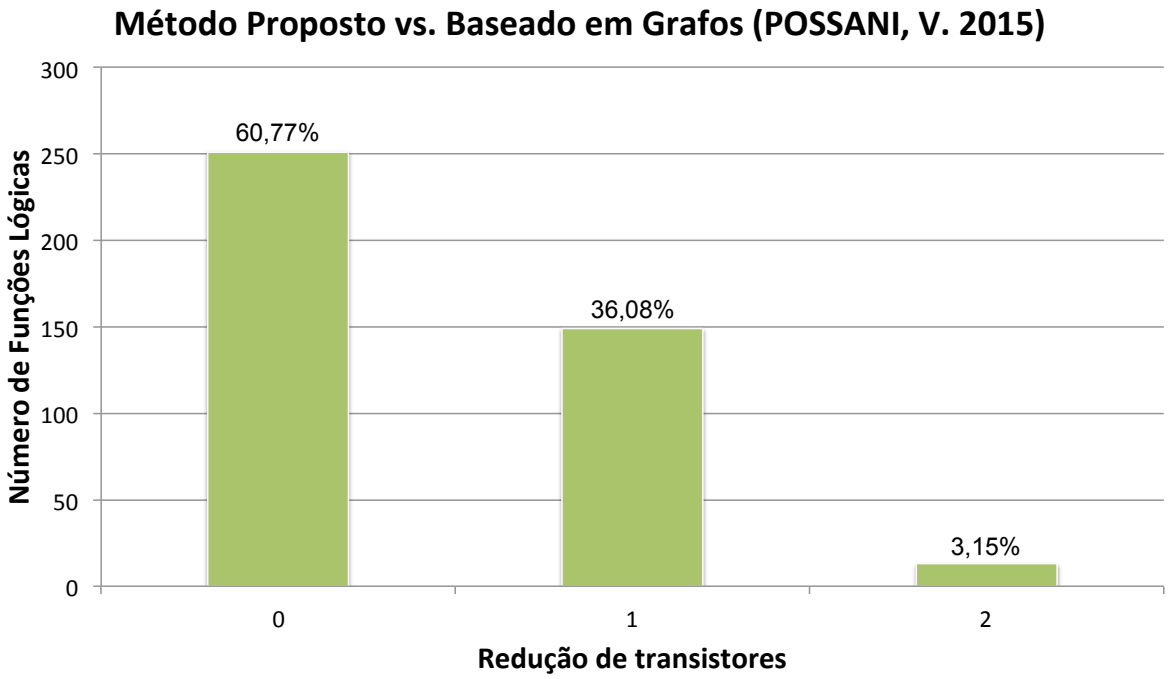


Figura 30 - Redução dos transistores obtidos pelo método proposto em relação ao método baseado em grafos (POSSANI, 2015), para o subconjunto de 413 funções da NPN 5 entradas.

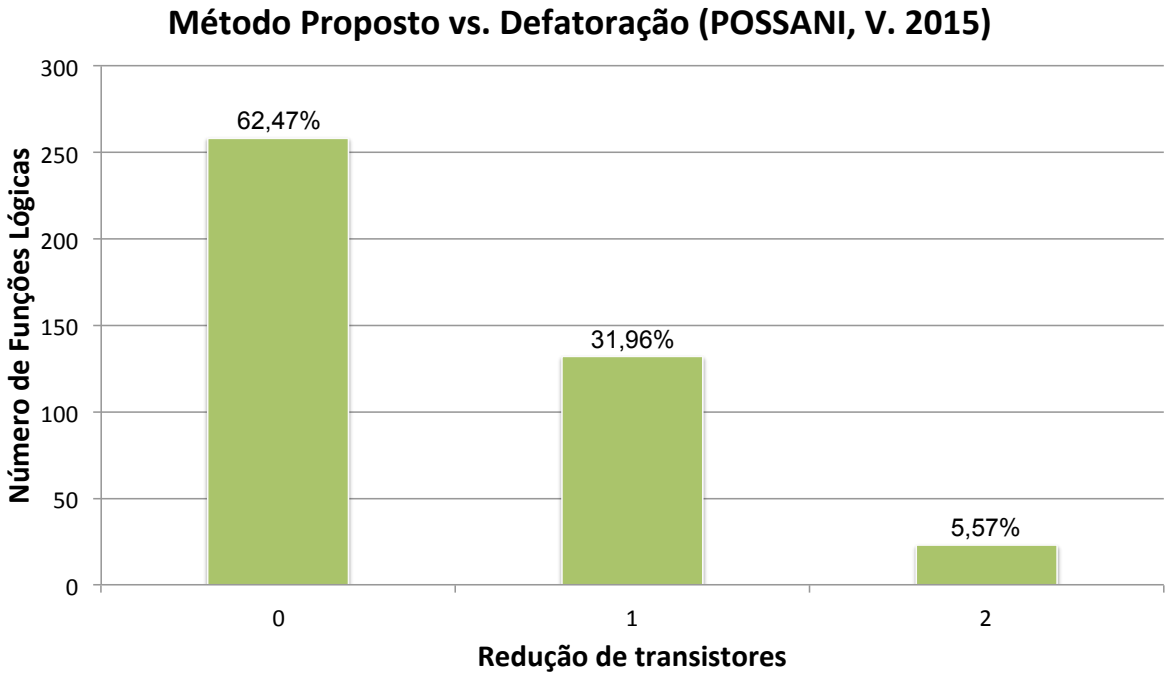


Figura 31 - Redução dos transistores obtidos pelo método proposto em relação ao método de defatoração (POSSANI, 2015), para o subconjunto de 413 funções da NPN 5 entradas

Por fim, o terceiro e último experimento foi realizado sobre o conjunto de circuitos do *benchmark* ACM/SIGDA (IWLS, 2005). Funções com 4 entradas foram extraídas dos circuitos através da realização de uma cobertura de K-cortes, com tamanho máximo 4. Para realizar os cortes, foi utilizada a ferramenta ABC (ABC, 2005) com o seguinte comando "*resyn2 ; if -K k*" (onde k representa o número máximo de entradas para cada K-corte). Desse modo, foram formadas portas lógicas para todas as funções extraídas pela ferramenta ABC. A Tabela 3 apresenta a quantidade total de dispositivos que cada um dos métodos gerou para cada circuito, bem como a redução no número de transistores e o tempo de execução necessário.

Neste experimento, o método proposto foi comparado com o método de fatoração proposto por Martins e o método baseado em grafo proposto por Possani. Como pode ser notado na Tabela 3, a abordagem proposta apresentou uma quantidade menor de transistores (IG) FinFET *double gate* para implementar os circuitos testados.

Quando comparado com o método baseado em grafo, dos 21 circuitos testados, o método proposto apresentou melhores resultados em 9 circuitos, reduzindo o número de dispositivos em 42,86% dos circuitos. Porém, comparando-se o tempo de execução, o método baseado em grafos, gerou as redes para os 21 circuitos em apenas 1,79 segundos. O método proposto foi capaz de fornecer as redes com quantidade reduzida de transistores em 51 segundos.

Tabela 3 - Resultados para geração dos circuitos com cortes k=4.

K = 4					
Circuitos	(MARTINS, M. et al., 2012)	Baseado em Grafo (POSSANI, V. 2015)	**Método Proposto**	Redução de Transistores	
				Martins	Possani
apex6	2200	2192	2184	0,72%	0,36%
apex7	736	735	732	0,54%	0,40%
c8	324	323	322	0,61%	0,31%
cm152a	48	48	48	0%	0%
cm162a	132	132	132	0%	0%
cm163a	130	128	126	3,07%	1,56%
cordic	152	150	148	2,63%	1,33%
cmb	128	128	128	0%	0%
count	410	400	400	2,43%	0%
cu	154	154	154	0%	0%
dalu	3880	3856	3774	2,73%	2,12%
decod	154	154	154	0%	0%
frg1	348	348	348	0%	0%
i2	446	446	446	0%	0%
i5	832	832	832	0%	0%
i7	2134	2130	2130	0,18%	0%
pair	4184	4166	4148	0,86%	0,43%
pcle	302	232	230	23,84%	0,86%
vda	2228	2228	2228	0%	0%
x2	132	132	132	0%	0%
x3	1932	1921	1910	1,13%	0,57%
Número total de dispositivos (IG) FinFET	20.986	20.835	20.706	1,33%	0,62%
Tempo de Execução (s)	0,44	1,79	51,0	-	-

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou um método automático de geração de redes de transistores dedicado a dispositivos (IG) FinFET *double gate*. A motivação para o desenvolvimento do presente trabalho se deu em virtude das análises realizadas, da constatação dos avanços e das novas possibilidades introduzidas na área de síntese lógica devido a introdução dos transistores (IG) FinFET *double gate*. Também foi motivação para a realização deste trabalho a mudança de paradigma relacionada à concepção da rede de transistores apresentada por Possani (POSSANI, 2015), onde ficou demonstrado que os métodos convencionais de geração de redes não apresentam bons resultados quando considerados os dispositivos *double gate*. A solução apresentada neste trabalho é baseada na metodologia de Composição Funcional. Através desta técnica é possível obter um controle, durante a execução, da quantidade de transistores em série e/ou em paralelo a serem implementados na rede, podendo-se alcançar bons resultados dadas as características dos dispositivos *double gate*.

Para avaliar o método proposto neste trabalho, foram realizados experimentos onde os resultados obtidos foram comparados com os métodos dedicados a dispositivos *double gate* propostos por Possani (POSSANI, 2015). Além disso, também foi utilizado para comparação o método de fatoração baseado em Composição Funcional, dedicado a dispositivos *single gate,* proposto por Martins (MARTINS, DA ROSA JUNIOR, *et al.*, 2010) (MARTINS, M., RIBAS, R., AND REIS, A, 2012). Como esperado, o método de fatoração proposto por Martins apresentou os piores resultados dentre os experimentos realizados. O método proposto aqui apresentou os melhores resultados quando comparado aos dois métodos dedicados a dispositivos *double gate* propostos por Possani. Durante os experimentos foram utilizados três *benchmarks* diferentes. O primeiro benchmark utilizado foi a *P-class* de 4 entradas, composto por 3.982 funções, onde o método proposto apresentou

uma redução no número de transistores, comparados aos métodos de grafos e o de defatoração, de 9,65% e 5,98%, respectivamente. No segundo experimento, utilizando um subconjunto de funções da classe NPN de 5 entradas, o método proposto mostrou uma redução de 8,31% comparado ao método de grafos e 8,44% quando comparado ao método de defatoração. Além disso, o método proposto se demonstrou eficiente na geração de redes de transistores dos circuitos do *benchmark* ACM/SIGDA. Neste experimento, 21 circuitos foram investigados, tendo o método proposto apresentado uma redução de 0,62% na quantidade de transistores quando comparado com a solução alcançada pelo método baseado em grafos proposto por Possani. Através dos resultados obtidos, conclui-se que o método proposto pode ser considerado uma forma alternativa para gerar redes de transistores dedicadas a dispositivos (IG) FinFET *double gate*. Apesar dos resultados apresentados pelo método proposto serem promissores, em questão de tempo de execução, o método demonstrou um pior desempenho. Isso se deve a complexidade do método. Mais especificamente, isto se deve ao fato de não ter sido empregada nenhuma heurística para reduzir o número de sub-funções geradas. Além disso, considerando o método proposto, é possível atribuir mais uma aplicação à metodologia de Composição Funcional.

Como trabalhos futuros pretende-se adicionar heurísticas com o intuito de reduzir a complexidade do método proposto, tornando-o mais competitivo em tempo de execução. Por fim, como sequência deste trabalho, pretende-se realizar avaliações elétricas para verificar o desempenho e o consumo de energia das redes geradas pelo método proposto.

REFERÊNCIAS

ABC, B. L. S. A. V. G. **ABC:** A System for Sequential Synthesis and Verification, 2005. Disponivel em: <http://www.eecs.berkeley.edu/~alanmi/abc/ >. Acesso em: Jan 2015.

ALIOTO, M. **Analysis and Evaluation of Layout Density of FinFET Logic Gates**. International Conference on Microelectronics. [S.l.]: [s.n.]. 2009.

ALIOTO, M. **Analysis of Layout Density in FinFET Standard Cells and Impact of Fin Technology**. International Symposium on Circuits and Systems (ISCAS). Paris: IEEE. 2010. p. 3204 - 3207.

ALIOTO, M. **Comparative Evaluation of Layout Density in 3T, 4T, and MT FinFET Standard Cells**. IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. [S.l.]: [s.n.]. 2011. p. 751-762.

ASHENHURST, R. L. **The decomposition of switching functions**. Computation Lab, Harvard University. [S.l.], p. 74–116. 1959.

BRYANT, R. E. Graph-based algorithms for Boolean function manipulation. **IEEE Transactions on Computers**, New York, USA, v. 35, p. 677-691, Ago 1986.

CAKICI, T. et al. **Independent gate skewed logic in double gate SOI technology**. Proceedings IEEE International SOI Conference. [S.l.]: [s.n.]. 2005. p. 83–84.

CHIANG, M. et al. Novel High-Density Low-Power Logic Circuit Techniques Using DG Devices. **IEEE TRANSACTIONS ON ELECTRON DEVICES**, v. 52, n. 10, p. 2339-2342, Oct 2005.

CHIANG, M. et al. High-Density Reduced-Stack Logic Circuit Techniques Using Independent-Gate Controlled Double-Gate Devices. **IEEE TRANSACTIONS ON ELECTRON DEVICES**, v. 53, n. 9, p. 2370- 2377, Sep 2006.

CURTIS, H. **A new approach to the design of switching circuits**. [S.l.]: D. Van Nostrand Company, 1962.

DATTA, A. et al. Modeling and Circuit Synthesis for Independently Controlled Double Gate FinFET Devices. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 26, n. 11, p. 1957-1966, Nov 2007.

DOYLE, B. et al. **Transistor elements for 30 nm physical gate lengths and below**. Intel Tech. J. [S.l.]: [s.n.]. 2002. p. 42-54.

FIGUEIRO, T. . R. R. . A. R. A. **Functional composition applied to aig constructive optimization**. South Brazil Simposium on Microelectronics. [S.l.]: [s.n.]. 2010.

HUANG, X. et al. **Sub 50-nm FinFET:** PMOS. Technical Digest. International Electron Devices Meeting. Washington, DC: [s.n.]. Dec 1999. p. 67 - 70.

INTEL. **Intel's Revolutionary 22 nm Transistor Technology**, May 2011. Disponivel em: <http://download.intel.com/newsroom/kits/22nm/pdfs/22nm-details_presentation.pdf>. Acesso em: Fev 2016.

IWLS, B. **Benchmarks, IWLS**, 2005. Disponivel em: <http://www.iwls.org>. Acesso em: Janeiro 2015.

KAGARIS, D.; HANIOTAKIS, T. A Methodology for Transistor-Efficient Supergate Design. **IEEE Trans. On Very Large Scale Integration (VLSI) Systems**, p. 488-492, 2007.

KING, T.-J. **FinFETs for nanoscale CMOS digital integrated circuits**. Proc. Int. Conf. Computer-Aided Design. [S.l.]: [s.n.]. 2005. p. 207-210.

LIU, Y. et al. Cointegration of High-Performance Tied-Gate Three-Terminal FinFETs and Variable Threshold-Voltage Independent-Gate Four-Terminal FinFETs With Asymmetric Gate-Oxide Thicknesses. **IEEE Electron Device Letters**, v. 28, n. 6, p. 517 - 519, June 2007.

MARTINS, M. et al. **Boolean Factoring with Multi-Objective Goals**. IEEE Int. Conf. on Computer Design. Amsterdam: [s.n.]. 2010. p. 229-234.

MARTINS, M., RIBAS, R., AND REIS, A. **Functional composition:** A new paradigm for performing logic synthesis. 13th International Symposium on Quality Electronic Design (ISQED). [S.l.]: [s.n.]. 2012. p. 236–242.

MISHRA, P.; MUTTREJA, A.; JHA, N. FinFET Circuit Design. **Springer Science+Business Media, LLC**, p. 23-54, 2011.

MUTTREJA, A.; AGARWAL, N.; JHA, N. **CMOS Logic Design with Independent-gate FinFETs**. 25th International Conference on Computer Design. Lake Tahoe, CA: [s.n.]. 2007. p. 560 - 567.

NOWAK, E. et al. **A functional FinFET-DGCMOS SRAM cell**. Tech. Dig. IEDM. [S.l.]: [s.n.]. 2002. p. 411–414.

NOWAK, E. J. et al. Turning silicon on its edge. **IEEE Circuits Devices Mag.**, v. 20, p. 20-31, Jan-Feb 2004.

POSSANI, V. **Exploring Independent Gates in FinFET-Based Transistor Network Generation** . Dissertação do Programa de Pós-Graduação em Computação, UFPel. Pelotas: [s.n.]. 2015. p. 78.

POSSANI, V. et al. Graph-Based Transistor Network Generation Method for Supergate Design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, n. 99, 2015.

POSSANI, V. N. et al. **Boolean Representation Code - An Efficient Method to Represent Boolean Functions**. 12th Microelectronics Students Forum. Brasília : [s.n.]. 2012.

REIS, A. . A. B. R. J. . L. . A. R. P. R. **Fast boolean factoring with multi- objective goals**. Proceedings of the International Workshop on Logic and Synthesis. Berkeley, CA, USA : [s.n.]. 2009.

ROSTAMI, M.; MOHANRAM, K. Dual-Vth Independent-Gate FinFETs for Low Power Logic Circuits. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 30, n. 3, p. 337- 349, Mar 2011.

SENTOVICH, E. **SIS: A system for sequential circuit synthesis**. Technical Report No. UCB/ERL M92/41, EECS Department, University of California. Berkeley. 1992.

SKOTNICKI, T. et al. The end of CMOS scaling. **IEEE Circuits Devices Mag.**, v. 21, p. 16-26, Jan-Feb 2005.

SYNOPSYS. Synopsys. **Synopsys**, 2012. Disponivel em: <www.synopsys.com/COMPANY/PUBLICATIONS/SYNOPSYSINSIGHT/Pages/Art2-finfet-challenges-ip-IssQ3-12.aspx>. Acesso em: Jan 2014.

WANG, M. Independent-Gate FinFET Circuit Design Methodology. **IAENG International Journal of Computer Science**, v. 37, n. 1, p. 50, Feb 2010.