UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDE DAVID FREITAS BRANDÃO

Geração de circuitos a partir de BDDs: simplificações possíveis a partir da decomposição de shannon.

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. André Reis

Porto Alegre
2021

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Brandão, Eduarde David Freitas

Geração de circuitos a partir de BDDs: simplificações possíveis a partir da decomposição de shannon. / Eduarde David Freitas Brandão. – Porto Alegre: PPGC da UFRGS, 2021.

63 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2021. Orientador: André Reis.

1. Diagramas de Decisão Binários (BDDs). 2. Grafos de Ands e Inversores (AIGs). 3. Mapeamento Tecnológico. 4. Decomposição de Shannon. 5. Multiplexadores. I. Reis, André. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Rui Vicente Oppermann
Vice-Reitora: Profª. Jane Fraga Tutikian
Pró-Reitor de Pós-Graduação: Prof. Celso Giannetti Loureiro Chaves
Diretora do Instituto de Informática: Profª. Carla Maria Dal Sasso Freitas
Coordenador do PPGC: Prof. João Luiz Dihl Comba
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

RESUMO

Esta dissertação apresenta uma contribuição para a geração de circuitos digitais a partir de diagramas de decisão binários (BDDs). Existe um método trivial para síntese de circuitos digitais a partir de BDDs onde cada nodo do BDD é mapeado diretamente para um multiplexador em uma correspondência um para um, ou seja: um nodo, um multiplexador. Este método é possível porque cada nodo do BDD é uma decomposição de Shannon que pode ser implementada como um multiplexador. Porém, simplificações podem ser aplicadas no circuito do multiplexador de nodos específicos, dependendo de propriedades da função Booleana representada em cada nodo específico. Esta dissertação cataloga e apresenta simplificações para o circuito do multiplexador baseado em condições apresentadas por nodos de BDDs. Foi também implementado um método para gerar circuitos a partir de BDDs usando a correspondência um para um com os multiplexadores simplificados (conforme o caso) ao invés de utilizar os multiplexadores completos. O método utilizando as simplificações leva a uma redução no número de portas lógicas necessárias para implementar o circuito final, quando comparado com a versão que implementa multiplexadores completos (sem as simplificações). Para os benchmarks apresentados, o método apresenta uma redução de 27.9% no número total de nodos, quando comparado com a versão que não utiliza as simplificações.

Palavras-chave: Diagramas de Decisão Binários (BDDs). Grafos de Ands e Inversores (AIGs). Mapeamento Tecnológico. Decomposição de Shannon. Multiplexadores.

Geração de circuitos a partir de BDDs: simplificações possíveis a partir da decomposição de shannon

ABSTRACT

This dissertation presents a contribution to the generation of digital circuits from binary decision diagrams (BDDs). There is a trivial method for synthesizing digital circuits from BDDs where each node of the BDD is mapped directly to a multiplexer in a one-to-one correspondence, that is: one node, one multiplexer. This method is possible because each BDD node is a Shannon decomposition that can be implemented as a multiplexer. However, simplifications can be applied to the multiplexer circuit of specific nodes, depending on properties of the Boolean function represented in each specific node. This dissertation catalogs and presents simplifications for the multiplexer circuit based on conditions presented by BDD nodes. A method was also implemented to generate circuits from BDDs using one-to-one correspondence with simplified multiplexers (as applicable) instead of using full multiplexers. The method using simplifications leads to a reduction in the number of logic gates required to implement the final circuit, when compared to the version that implements complete multiplexers (without simplifications). For the benchmarks presented, the method presents a reduction of 27.9% in the total number of nodes, when compared to the version that does not use the simplifications.

Keywords: Binary Decision Diagrams (BDDs). And Inverter Graphs (AIGs). Technology Mapping. Shannon Decomposition. Multiplexors.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A indústria de semicondutores alcançou notavelmente uma posição distinta entre outras indústrias nas últimas décadas (ITRS, 2015). Muito disso se deve tanto à velocidade com que as melhorias são alcançadas em seus produtos quanto à capacidade da indústria de diminuir exponencialmente os tamanhos mínimos de recursos usados para fabricar circuitos integrados. Entre uma série de tendências relacionadas a esses fatos, a tendência mais citada é o nível de integração, que geralmente é expresso como a Lei de Moore (MOORE, 1965).

A possibilidade de integração proporcionada pela tecnologia CMOS leva a existência de circuitos com milhões de portas lógicas, o que implica no uso de diferentes ferramentas automáticas de síntese e verificação de circuitos em um fluxo de projeto. Em uma primeira aproximação, pode-se dizer que ferramentas de projeto automático de circuitos eletrônicos (EDA - Electronic Design Automation - Tools, em inglês) se dividem em ferramentas de síntese lógica e ferramentas de síntese física. A seguir são descritas brevemente estas etapas.

As ferramentas de síntese lógica produzem uma rede de portas lógicas interconectadas que em conjunto realizam a função desejada para o circuito descrito inicialmente pelo projetista. Esta descrição inicial é feita em uma linguagem de descrição de hardware tal como Verilog ou VHDL. A descrição em linguagem Verilog é processada e transformada em uma estrutura de dados interna, tipicamente um grafo de ANDs e Inversores (AIG - And Inverter Graph, em inglês), usado em ferramentas atuais do estado da arte tais como o ABC. A estrutura de dados interna é então otimizada para reduzir funções de custo tais como número total de nodos (relacionado com área) ou o caminho mais longo do circuito (relacionado com o atraso crítico). Finalmente (depois da otimização), a estrutura da dados interna é transformada em uma rede de portas lógicas, possivelmente fazendo referência a portas lógicas específicas de uma biblioteca de células.

As ferramentas de síntese física produzem um leiaute de circuito integrado a partir da rede de portas lógicas interconectadas produzida pela síntese lógica. Para isto a síntese física realiza, entre outras etapas, o posicionamento e roteamento de células. A etapa de posicionamento determina a posição dos leiautes das células individuais em um plano. A etapa de roteamento descreve as camadas físicas que realizam as interconexões entre as células.

Esta dissertação foca principalmente na etapa de síntese lógica. A etapa de síntese física não será discutida no texto. A seguir apresentamos a motivação, objetivos e finalmente discutimos a organização da tese resumindo o conteúdo de cada capítulo.

1.1 Motivação

A principal motivação deste trabalho foi uma submissão para o concurso do Workshop Internacional de Síntese Lógica (IWLS). A princípio esta motivação parece muito pontual, mas como se trata de uma conferência que é estado da arte a motivação se justifica.

O concurso do IWLS visava sintetizar circuitos digitais de pequeno porte com área mínima medida em termos de nodos de AIG. A proposta do grupo LogiCS foi baseada em BDDs, pois havia alguns alunos trabalhando com pacotes de BDDs para o qual já existia código pronto, proporcionando agilidade de implementação e oportunidade de aprendizado extra. Foi avaliado em conjunto com o Professor André Reis que esta abordagem baseada em BDDs poderia ter pontos fracos e pontos fortes. Uma primeira desvantagem é que BDDs podem ter sabidamente um tamanho exponencial para alguns circuitos. Além disso, este tamanho depende do ordenamento das variáveis e pode ser computacionalmente infactível encontrar a melhor ordem, ou mesmo uma boa ordem. Outra desvantagem é que nodos de BDDs são multiplexadores que correspondem a um conjunto de três nodos de AIG (conforme será visto no corpo da dissertação), fazendo que a transformação de BDDs em AIGs seja custosa em termos de nodos. Felizmente, BDDs também apresentam algumas vantagens no contexto do concurso do IWLS. Uma primeira vantagem é que a construção dos BDDs a partir das tabelas verdade é muito fácil de implementar, pelo fato de BDDs serem estruturas de dados canônicas. Uma segunda vantagem é o fato de BDDs naturalmente compartilharem subfunções lógicas, por serem grafos canônicos. Isto é uma vantagem no contexto do concurso do IWLS pois os circuitos apresentavam múltiplas saídas (com a ressalva de que as funções para as múltiplas saídas poderiam ter melhores ordens de variáveis que seriam incompatíveis entre si).

A motivação da dissertação surgiu a partir de uma das desvantagens listadas acima, mais especificamente o fato de que nodos de BDDs são multiplexadores que correspondem a um conjunto de três nodos de AIG, fazendo que a transformação de

BDDs em AIGs seja custosa em termos de nodos. Assim, foi vista uma oportunidade de mitigar esta transformação um para três usando versões simplificadas do circuito multiplexador, sempre que possível.

A síntese de circuitos baseados em multiplexadores não é muito comum, mas pode ser usada em contextos específicos tais como circuitos hazzard-free (KUSHNEROV; MEDINA; YAKOVLEV, 2021), como são conhecidos os circuitos que produzem somente transições unidirecionais de sinal de saída. BDDs ainda são usados e importantes em outros contextos, tais como computação quântica (WILLE; HILLMICH; BURGHOLZER, 2021) e resolvedores de fórmulas de satisfiability quantificadas (BRYANT; HEULE, 2021). Embora estes contextos não sejam contextos de síntese de circuitos, são contextos que mostram a importância de BDDs como estruturas de dados.

1.2 Objetivos

O objetivo desta dissertação é apresentar uma contribuição a geração de circuitos digitais a partir de BDDs. A geração de circuitos digitais a partir de BDDs faz a correspondência entre cada nodo de um BDD e um multiplexador de duas entradas. Como multiplexadores correspondem a um conjunto de três nodos de AIG, fazer a transformação direta de BDDs em AIGs pode ser custoso em termos de nodos. Esta dissertação tem como objetivo determinar quais simplificações podem ser feitas neste processo, conforme a topologia do BDD. As possíveis simplificações são catalogadas e seu efeito na área do circuito sintetizado é avaliado. Para tanto, foi implementado um método para gerar circuitos a partir de BDDs usando a correspondência um para um com os multiplexadores simplificados (com menos de três nodos de AIG, conforme o caso) ao invés de utilizar os multiplexadores completos (com três nodos de AIG). O método utilizando as simplificações leva a uma redução no número de portas lógicas necessárias para implementar o circuito final, quando comparado com a versão que implementa multiplexadores completos (sem as simplificações), mitigando a desvantagem apontada na motivação.

1.3 Organização da dissertação

A organização desta dissertação está dividida em seis capítulos. Os próximos capítulos estão organizados da seguinte forma:

Capítulo 2: *Conceitos Básicos* — apresenta os conceitos básicos necessários ao entendimento desta dissertação.

Capítulo 3: *Trabalhos Anteriores* — revisa alguns trabalhos prévios que tem conexão com o trabalho apresentado nesta dissertação.

Capítulo 4: *Reduções Propostas* — cataloga e descreve as reduções propostas para gerar circuitos a partir de BDDs usando multiplexadores e a decomposição de Shannon. É a principal contribuição teórica desta dissertação.

Capítulo 5: *Resultados* — apresenta e discute os resultados obtidos. A ênfase é na avaliação da contribuição de cada simplificação proposta. É a principal contribuição prática desta dissertação.

Capítulo 6: *Conclusão e trabalhos futuros* — apresenta as principais conclusões, resume as contribuições deste trabalho e apresenta possíveis trabalhos futuros.

2 CONCEITOS BÁSICOS

2.1 Sobre este capítulo

Este capítulo resume a base técnica necessária relacionada a este trabalho. São discutidos vários tópicos, incluindo questões da síntese lógica, diversas estruturas de dados para síntese lógica e alguns processos como a decomposição de Shannon.

2.2 Síntese lógica

Neste tópico falaremos sobre síntese lógica. A maioria dos circuitos integrados são projetados através do fluxo de projeto baseado em células. Este fluxo visa fornecer circuitos eficientes em um curto espaço de tempo. Ele compreende várias etapas de síntese, otimização e verificação. A síntese lógica é uma etapa essencial desse fluxo. Seu objetivo é obter automaticamente um circuito lógico pequeno e rápido para uma determinada função Booleana. A captura do circuito é um primeiro passo importante para a síntese lógica.

Um fluxo de projeto baseado em células é composto de uma sequência de processos a serem desenvolvidos pelo projetista para obter a especificação do circuito, ao fim do fluxo esta especificação é enviada para fabricação. Esta especificação é dada principalmente por máscaras que especificam a localização das portas lógicas (compostas de transistores) e dos fios que as conectam. A síntese lógica é uma das etapas iniciais do fluxo, tradicionalmente ela é divida em outras três etapas mais específicas:

- Otimizações independentes de tecnologia: tradicionalmente nesta etapa se transforma a especificação do circuito em formato *Register Transfer Level* (RTL) para uma estrutura multi-nível de rede Booleana como um grafo de ANDs e inversores (AIG (MISHCHENKO; CHATTERJEE; BRAYTON, 2006)) ou um grafo de majoritárias e inversores (MIG (AMARú; GAILLARDON; MICHELI, 2016)), ou uma rede decisões como um BDD (BRYANT, 1986). A partir de uma estrutura independente de tecnologia são realizadas otimizações lógicas que tornam a especificação lógica do circuito mais compacta.
- Mapeamento tecnológico: esta etapa transforma a estrutura independente de

tenologia em uma estrutura correlacionada a biblioteca de células providenciada pela tecnologia do circuito a ser fabricado. A partir desta etapa o projetista passa a ter acesso a especificações físicas básicas do circuito.

- Otimizações dependentes de tecnologia: ao fim da síntese lógica são feitas otimizações considerando especificações físicas do circuito, como consumo de potência, atrasos de propagação e área do circuito. Geralmente alterando um desses objetivos acaba também modificando os outros, o que torna um desafio na implementação de circuitos integrados.

2.3 Funções Booleanas

Uma função Booleana é uma função matemática onde suas variáveis estão limitadas a um conjunto de dois valores. As funções Booleanas são usadas para expor o comportamento de um circuito lógico. Onde ambos pertencem ao conjunto de valores binários $\{0, 1\}$ e os elementos são interpretados como valores lógicos, por exemplo, 0=falso e 1=verdadeiro. Ou seja, dado um conjunto binário $\mathbb{B} = \{0, 1\}$, uma função f de n entradas e uma saída tem a forma $f : \mathbb{B}^k \rightarrow \mathbb{B}^m$, onde k é um inteiro não negativo conhecido como aridade, quando $k = 0$ a função é uma constante, a função Booleana tem múltiplas saídas quando $m > 1$.

2.3.1 Funções Unate e Binate

Uma função unate pode ser implementada utilizando lógica *single-rail* com portas ANDs e ORs, assim diminuindo a quantidade de pinos e interconexões do circuito lógico. A função unate é um tipo especial de função Booleana que tem característica monotônica, ou seja, para todo a_i e b_i em $\{0, 1\}$, se $a_1 \leq b_1, a_2 \leq b_2, ..., a_n \leq b_n$, então $f(a_1, ..., a_n) \leq f(b_1, ..., b_n)$. Uma função Booleana é monotônica se para todas entradas, caso se inverta uma delas de falso para verdadeiro só pode causar a saída trocar também de falso para verdadeiro. Uma função f pode ser:

- Unate positiva sobre uma variável dependente x_i se x_i' não aparece na sua representação em soma de produtos (SOP, do inglês sum of products).
- Unate negativa sobre uma variável dependente x_i se x_i não aparece na sua representação em SOP.

- Unate *vacuos* sobre uma variável dependente x_i se nem x_i ou x_i' não aparece na sua representação em SOP.
- Binate sobre uma variável x_i se não é possível escrever uma SOP em que ambos x_i e x_i' não apareçam. Ou seja, uma função é binate se uma de suas variáveis não é nem unate positiva nem unate negativa.

Infelizmente a maioria das funções Booleanas mais comumente usadas como somadores completos e multiplicadores não são unate. Elas normalmente são binate, implicando que as propriedades unate não podem ser exploradas para resolver eficientemente a maioria desses implementações. Enquanto a maioria das funções são binate, a expansão recursiva de Shannon em funções binate leva a unificar cofatores. Então, se não podemos explorar características unate diretamente em uma função binate, podemos sempre expandí-lo usando a decomposição de Shannon até encontrar cofatores unate e então resolver o problema de forma eficiente. A decomposição de Shannon será detalhadamente exposta na Seção 2.4.1.3 a seguir.

2.3.2 Função simétrica

Uma função Booleana simétrica é aquela na qual uma permutação, ou seja, um reordenamento de suas n variáveis não implicará nenhuma alteração na sua função Booleana. Uma função simétrica Booleana dependerá somente do número de zeros e uns na sua entrada, por isso ela também é conhecida como função Booleana de contagem.

Em uma função Booleana simétrica a ordem das variáveis não altera o tamanho do BDD. A maioria das funções Booleanas não são simétricas e portanto o ordenamento adequado das variáveis do BDD para reduzir o tamanho do grafo é importante na síntese de circuitos a partir de BDDs.

2.4 Representação de funções Booleanas

Existem várias estruturas de dados capazes de representar uma função Booleana. Cada uma delas tem suas respectivas vantagens e desvantagens. Iremos descrever algumas delas a seguir.

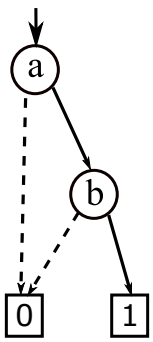
2.4.1 Diagramas de Decisão Binários - BDDs

Diagramas de Decisão Binários ou BDDs são uma das estruturas de dados mais utilizadas para representar e manipular funções Booleanas. Originalmente estudados e propostos por Lee (LEE, 1959) e Akers (AKERS, 1978), mas somente após Bryant (BRYANT, 1986) propor os BDDs reduzidos e ordenados *ROBDDs,do inglês Reduced Ordered BDDs* fez com que estes se tornassem realmente populares. Este modelo de estrutura têm propriedades importantes, que dão forma canônica às funções Booleanas, ao mesmo tempo que mantém o tamanho do grafo viável para muitas funções práticas. Estas características facilitam as manipulação e simplificação de funções Booleanas(MINATO; ISHIURA; YAJIMA, 1990).

Um BDD é um grafo acíclico dirigido (DAG) onde cada nó é controlado por uma variável e possui exatamente duas arestas (aresta 0 e aresta 1) que apontam para o caminho a ser seguido quando a variável assume o valor 0 ou 1. No final de cada caminho, existem dois nós chamados de terminais, que representam as constantes 0 e 1. Cada nodo não terminal de um BDD representa uma função (ou sub-função) Booleana que não é uma constante (BRYANT, 1986).

Considere a figura 2.1, que mostra o BDD para uma função AND de duas entradas. Neste BDD, os terminais zero (quadrado com o valor 0) e um (quadrado com o valor 1) representam os valores lógicos zero e um. Os nodos que não são terminais (nodos redondos) simbolizam a expansão de Shannon da função Booleana: F=x' * fx0 + x * fx1, onde x é o indice de uma variável e fx0 e fx1 são as funções apontadas pelo arco_0 (as vezes representado como tracejado) e pelo arco_1 (representado como uma linha sólida). Note que o único caminho entre a raiz (topo) do BDD e o nodo terminal um se dá através das linhas sólidas que representam os valores $a = 1$ e $b = 1$. Assim o BDD representa de fato uma AND.

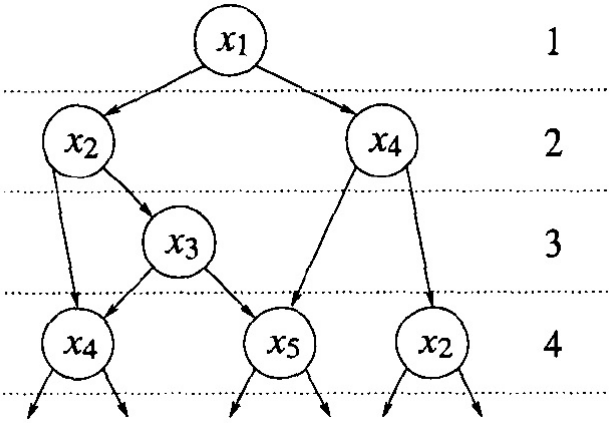
Figura 2.1: ROBDD para a *and* de 2 entradas e função $f(a, b) = a \cdot b$.



2.4.1.1 BDDs Ordenados - OBDDs

Um BDD é ordenado (OBDD) quando as variáveis de entrada são organizadas em ordem fixa para cada um dos caminhos do grafo, e uma variável, só aparece uma única vez em um caminho, sendo assim ela só é avaliada uma vez. O ordenamento das variáveis em um BDD é importante para garantir a canonicidade da estrutura de dados, mas tem um custo em tamanho do BDD. Por esta razão foram propostos os free BDDs (GUNTHER; DRECHSLER, 1999), que abrem mão do ordenamento (e da canonicidade) em busca de uma redução do tamanho. A figura 2.2 apresenta um exemplo parcial de um free BDD. Note que no free BDD apresentado as variáveis x_2 e x_4 podem aparecer em diferentes níveis e existem caminhos onde elas não aparecem na mesma ordem, caracterizando a falta de ordenamento das variáveis em um free DD.

Figura 2.2: Um exemplo (parcial) de Free BDD. Fonte: (GUNTHER; DRECHSLER, 1999).



2.4.1.2 BDDs Reduzidos e Ordenados - ROBDD

Os BDDs reduzidos e ordenados (ROBDDs, do inglês *Reduced Ordered Binary Decision Diagrams*) são os BDDs mais utilizados na prática, de maneira que quase sempre que se refere a um BDD na verdade está se referindo a um ROBDD, conforme proposto por Bryant (BRYANT, 1986). Os ROBDDs são caracterizados por (1) uma ordem fixa das variáveis e (2) a aplicação de duas regras de redução específicas (descritas adiante). As características dos ROBDDs garante que a estrutura de dados é canônica. Na prática, nodos diferentes têm funções Booleanas diferentes; e uma função Booleana corresponde a um e somente um nodo do BDD

(não existem nodos distintos com a mesma função Booleana). A vantagem da utilização de ROBDDs é a sua forma canônica para um função Booleana, dada uma ordem de variáveis fixa. A propriedade canônica é muito útil para procedimentos como a checagem de equivalência de funções lógicas e o mapeamento tecnológico.

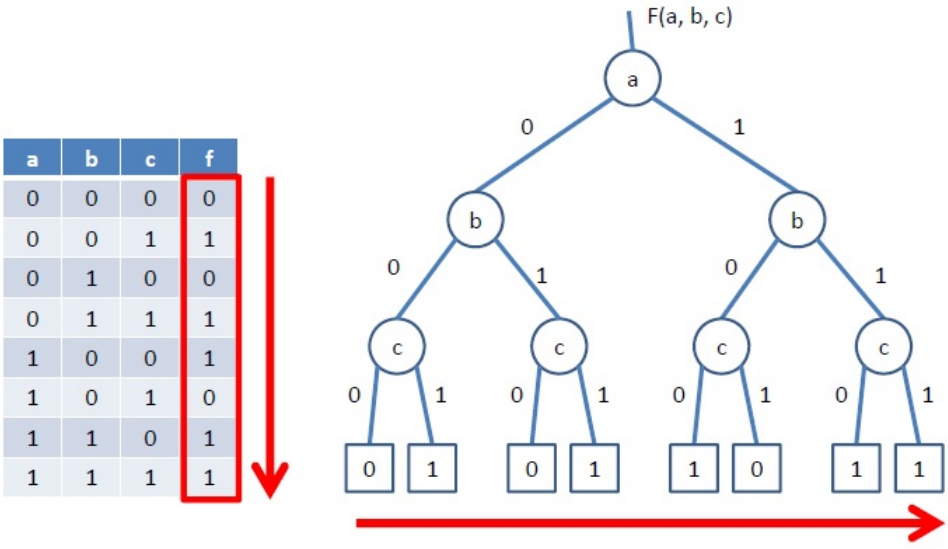
O diagrama de decisão pode ser considerado reduzido quando respeita as duas regras a seguir:

- Regra de partilha: dois nodos com a mesma função Booleana são unificados e um deles é deletado (na prática, nem é criado).
- Regra de eliminação: cada nodo cujos dois arcos filhos apontam para a mesma função Booleana (nodo) deve ser removido, e substituído por um dos dois filhos (que são iguais).

Desta maneira, um BDD pode ser considerado BDD reduzido (RBDD) quando respeita as duas regras expostas acima. Além disso, um BDD é considerado um BDD ordenado (OBDD) quando suas variáveis estão ordenadas de maneira fixa nos caminhos do grafo. Um ROBDD é reduzido (RBDD) e ordenado (OBDD).

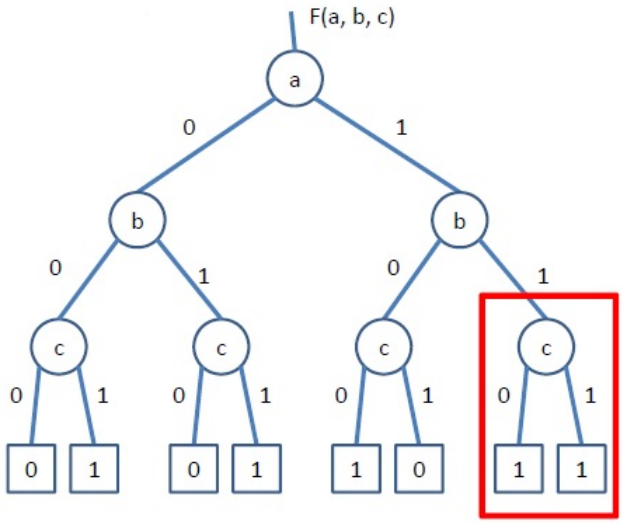
A maneira mais fácil de entender a construção de um ROBDD é perceber que a partir de uma tabela verdade pode ser criada uma uma árvore de decisão binária conforme ilustrado na figura 2.3. Note que a árvore de decisão binária é criada com uma ordem de variáveis fixa (no caso $\{a, b, c\}$, a partir da raiz), o que garante a característica ordenada de um OBDD. Sobre esta árvore podem ser aplicadas as reduções, conforme veremos a seguir.

Figura 2.3: Exemplo de criação de uma árvore de decisão binária a partir de uma tabela-verdade. Fonte: Prof. André Reis.



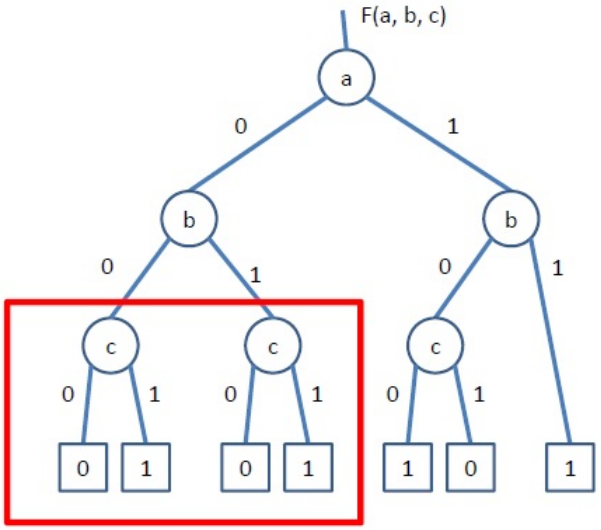
A figura 2.4 ilustra um nodo onde se pode aplicar a regra da eliminação. O nodo dentro do quadrado vermelho pode ser eliminado e substituído pelo nodo terminal um (quadrado contendo 1), já que os dois arcos do nodo apontam para o nodo terminal um (e na prática o nodo não toma nenhuma decisão).

Figura 2.4: Um exemplo de nodo onde se aplica a regra da eliminação. Fonte: Prof. André Reis.



A figura 2.5 mostra um par de nodos que podem ser unificados pela regra da partilha. Note que os dois nodos dentro do quadro vermelho representam a função Booleana correspondente a variável c.

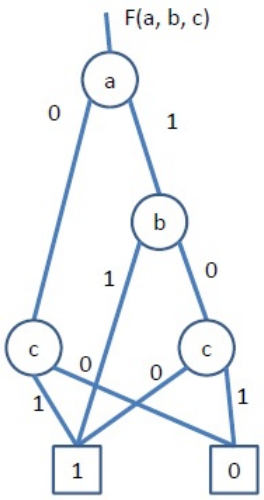
Figura 2.5: Um exemplo de nodo onde se aplica a regra da partilha. Fonte: Prof. André Reis.



A figura 2.6 mostra o ROBDD final que resulta da aplicação de todas as reduções possíveis ao ROBDD, durante o processo de construção a partir da tabela-verdade. Note que a árvore de decisão Booleana original (figura 2.3) possuia sete

nodos não terminais. Destes sete nodos, três foram eliminados, resultando em um ROBDD com quatro nodos não terminais (figura 2.6).

Figura 2.6: ROBDD final depois de aplicadas todas as reduções ao ROBDD. Fonte: Prof. André Reis.



Um ROBDD pode crescer exponencialmente com o número de variáveis de entrada da função Booleana. Ou seja, a escolha do ordenamento das variáveis é de suma importância, especialmente para grandes tamanhos de entradas (MOEINZADEH et al., 2009).

2.4.1.3 Decomposição de Shannon

Uma função booleana pode ser decomposta em funções booleanas mais simples. Uma decomposição bem conhecida de funções booleanas é a decomposição de Shannon (SHANNON, 1949). A equação 2.1 mostra a expressão correspondente à decomposição de Shannon. O significado da Equação 2.1 é que uma função booleana *f(x)* pode ser reescrita usando duas funções mais simples que não dependem da variável x. Essas funções mais simples são o cofator positivo $f_{x=1}$ e o cofator negativo $f_{x=0}$, que não dependem de x pois a variável assume um valor fixo.

$$f(x) = (x \cdot f_{x=1}) + (\overline{x} \cdot f_{x=0}) \tag{2.1}$$

A decomposição de Shannon aparece nos nós de um ROBDD. Por exemplo, o ROBDD na figura 2.1 poderia ter o nó raiz expresso de acordo com a Equação 2.2, onde $f_{a=0} = 0$ e $f_{a=1} = b$.

$$f(a, b) = (a \cdot f_{a=1}) + (\overline{a} \cdot f_{a=0}) \tag{2.2}$$

2.4.2 And-Inverter Graphs

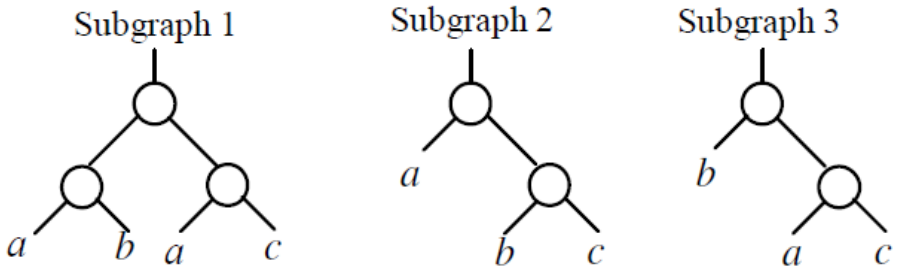
Grafos de Ands e Inversores (AIGs, And-Inverter-Graphs em inglês) são estruturas de dados para representar funções Booleanas. AIGs são grafos dirigidos e acíclicos onde os nodos são sempre portas AND de duas entradas e as inversões são representadas nos arcos (geralmente se desenha um arco tracejado para indicar a negação do sinal). A figura 2.7 apresenta três diferentes AIGs para a função f=abc. O subgrafo 1 pode ser representado pela equação 2.3. O subgrafo 2 pode ser representado pela equação 2.4. O subgrafo 3 pode ser representado pela equação 2.5.

$$f = (a \cdot b) \cdot (a \cdot c) \tag{2.3}$$

$$f = a \cdot (b \cdot c) \tag{2.4}$$

$$f = b \cdot (a \cdot c) \tag{2.5}$$

Figura 2.7: Três diferentes AIGs para a função f=abc. Fonte: (MISHCHENKO; CHATTERJEE; BRAYTON, 2006)

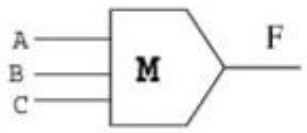


AIGs são usados em ferramentas estado da arte de síntese lógica, principalmente com métodos de reescrita de AIGs (MISHCHENKO; CHATTERJEE; BRAYTON, 2006). Um método de reescrita de AIGs substitui repetidamente pequenos pedaços equivalentes de AIG, de modo a otimizar funções de custo para o circuito global. Os três subgrafos na figura 2.7 são equivalentes e a referência original demonstra que qualquer um dos três pode ser a melhor opção dependendo do contexto do circuito global. Assim, estas pequenas substituições repetitivas fazem parte dos métodos de otimização que são estado da arte.

2.4.3 Majority-Inverter Graph

Um grafo de majoritárias e inversores (MIG, do inlês *Majority-Inverter Graph*) são redes Booleanas no formato de grafos direcionado acíclico (DAG), onde seus nodos são compostos de portas majoritárias de três entradas e seus vértices são regulares ou representam inversões Booleanas. A figura 2.8 apresenta um exemplo de um nodo MIG e sua respectiva tabela verdade, no qual pode ser produzida com as lógicas *and* e *or* e sua saída será o valor binário que se apresenta em maioria entre as entradas da porta majoritaria. Em situações específicas, os MIGs providenciam otimizações lógicas superiores quando comparados com a representação do circuito no formato de AIG (AMARú; GAILLARDON; MICHELI, 2014).

Figura 2.8: Símbolo e tabela verdade de um nodo MIG. Fonte: (SEN et al., 2012)



A	B	C	F	
0	0	0	0	
0	0	1	0	
0	1	0	0	AND Logic
0	1	1	1	A fixed to 0
1	0	0	0	
1	0	1	1	
1	1	0	1	OR logic
1	1	1	1	A fixed to 1

2.5 Suporte a Implementações físicas

Nesta seção discutimos suporte a implementações físicas. Todo circuito digital acaba sendo implementado em uma tecnologia física em maior ou menor grau de abstração. Estas implementações físicas são importantes para entender as métricas usadas para avaliar o resultado da síntese. Esta seção foca em listar algumas implementações e discutir suas métricas.

2.5.1 Redes de transistores

Nas tecnologias atuais as portas lógicas são feitas com transistores que funcionam como chaves (POSSANI et al., 2012). Existem vários trabalhos que mapeiam lógica diretamente para redes de transistores. Dentre outros, podemos citar (JUNIOR et al., 2006), (ROSA et al., 2007), (da Silva; REIS; RIBAS, 2009) e (ROSA et al., 2009). Estes trabalhos podem principalmente ter aplicações em síntese de redes de transistores para geração de bibliotecas de células.

Para trabalhos focados em redes de transistores, a métrica mais usualmente adotada é o número de transistores. Esta métrica considera normalmente o número de transistores sem levar em conta o dimensionamento.

2.5.2 Bibliotecas de células

A maior parte dos circuitos digitais feitos atualmente segue uma metodologia baseada em células. Assim, o circuito digital é mapeado para uma rede de portas lógicas que são instâncias de células definidas em uma biblioteca. Um exemplo de gerador de bibliotecas de células é o trabalho de Togni (TOGNI et al., 2002). Uma biblioteca bastante usada em trabalhos acadêmicos é a biblioteca Nangate (MARTINS et al., 2015).

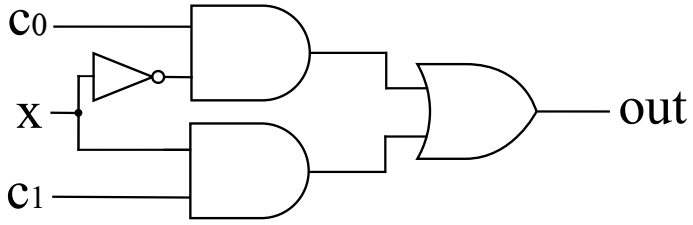
Para trabalhos que otimizam circuitos utilizando bibliotecas de células, a métrica mais adotada é o somatório da área das instâncias de células. Note que esta área depende das instâncias específicas, já que cada função lógica pode estar disponível em mais de um tamanho de célula.

2.5.3 Multiplexadores

Utilizando uma rede de multiplexadores, é possível representar uma função lógica de um circuito combinacional. Considere a figura 2.9. Um multiplexador tem uma entrada Booleana de seleção (X) e duas entradas Booleanas de dados ($c0, c1$); apresentando também uma saída Booleana (out). Um multiplexador 2x1 é um circuito que usa a entrada de seleção para escolher uma entre as entradas de dados, a fim de selecionar a lógica expressa para o circuito. A figura 2.9 ilustra um

multiplexador 2x1, cuja funcionalidade é dada pela Equação 2.6.

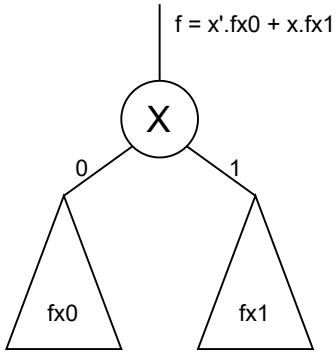
Figura 2.9: Um circuito multiplexador 2x1.



$$f(x) = (x \cdot c_1) + (\overline{x} \cdot c_0) \tag{2.6}$$

Observe que a equação do multiplexador 2x1 (equação 2.6) e a equação da decomposição de Shannon (equação 2.1) são equivalentes quando $c_1 = f_{x=1}$ e $c_0 = f_{x=0}$, significando que um nodo de BDD pode ser implementado logicamente como um circuito multiplexador 2x1. Desta forma, o nodo de um ROBDD pode ser visto como um multiplexador (ASHAR; DEVADAS; KEUTZER, 1993). A figura 2.10 representa um nodo de BDD e sua equação correspondente a decomposição de Shannon, que é equivalente à equação de um multiplexador 2x1.

Figura 2.10: Decomposição de Shannon para um nodo de BDD.



2.5.4 Look-up Tables e FPGAs

Uma *Look-up Table* (LUT) corresponde a uma tabela lógica que determina qual será a sua saída a partir de um número predeterminado de entradas. Uma LUT de n entradas também pode ser vista como uma representação de uma tabela verdade (programável ou configurável) correspondente a uma função lógica de n entradas. Desta maneira, qualquer função lógica de n entradas pode ser implementado por uma LUT de n ou mais entradas.

Um circuito *Field Programmable Gate Array* (FPGA) é composto de diversas LUTs, ou seja, ele tem capacidade de se tornar qualquer circuito, dentro do número disponível de blocos lógicos em formato de LUT. Assim, circuitos configurados em FPGAs implementam a lógica combinacional na forma de uma rede de LUTs interconectadas. Quando uma FPGA é configurada, ela simplesmente configura as tabelas verdades das LUTs, e as conexões (programáveis) entre as LUTs.

2.6 Fluxo de projeto

Existem vários fluxos de projeto possíveis para o projeto de circuitos integrados digitais. É tarefa impossível adquirir conhecimento detalhado do funcionamento de fluxos de projeto incluindo o funcionamento interno das ferramentas no espaço e no tempo de uma dissertação. Nesta seção tentamos esclarecer ao menos as etapas de síntese que estão sendo utilizadas neste trabalho, bem como discutir suas limitações.

2.6.1 Captura do projeto

A captura do projeto consiste em obter uma descrição do circuito inicial, descrito pelo usuário, na forma de uma estrutura de dados que será utilizada no processamento. Isto varia de ferramenta para ferramenta e até com os usuários e os objetivos. A seguir discutiremos estes aspectos conforme endereçados pela dissertação.

Considerando a descrição inicial feita pelo usuário, normalmente esta é uma descrição da qual se podem extrair ou uma tabela verdade ou um conjunto de equações que representam o circuito. Por exemplo, mesmo usando um formato padrão como verilog, existem construções que podem ser mais facilmente mapeadas para tabela verdade ou para equações, dependendo da construção. No caso desta dissertação, a descrição de entrada foi limitada ao formato *.pla*, que é um dos formatos popularizados pela ferramenta SIS de Berkeley. A justificativa para isto é que a dissertação nasceu em um escopo de participação do concurso de síntese lógica do IWLS onde o formato de entrada era o formato *.pla*. Este formato *.pla* é uma forma de descrever uma tabela verdade como uma soma de produtos, e o método de criação

de ROBDD a partir de uma tabela verdade ilustrado anteriormente na seção 2.4.1.2 pode ser utilizado para obter o ROBDD a partir do arquivo *.pla*.

Considerando a estrutura de dados interna, ferramentas acadêmicas estado da arte usam AIGs e MIGs. Por exemplo, no caso do ABC, que é a ferramenta acadêmica considerada estado da arte, a estrutura de dados interna é um AIG. No caso desta dissertação, a estrutura de dados utilizada internamente é um ROBDD, que não é uma estrutura de dados estado da arte por problemas de tamanho do ROBDD, que resulta em excesso de área do circuito e também em problemas de escalabilidade. De novo, esta escolha se justifica porque a participação que o grupo LogiCS fez no concurso do IWLS optou por usar BDDs por ser a forma mais fácil de construir uma ferramenta em tempo hábil, ao mesmo tempo que permitia aos alunos ganhar experiência em ROBDDs que são uma estrutura de dados importante conceitualmente.

2.6.2 Otimizações

No escopo desta dissertação, dois tipos de otimização lógica foram adotadas. O objetivo do concurso do IWLS era reduzir o número de nodos de AIG necessários para implementar funções Booleanas. A abordagem adotada pelo grupo LogiCS faz isto a partir da criação de um BDD e posterior criação de um AIG a partir do BDD. As otimizações lógicas adotadas são discutidas nas próximas sub-seções.

2.6.2.1 Construção e ordenamento do BDD

Em uma primeira aproximação, um BDD tem uma relação de 1 para 3 em termos de nodos comparado com um AIG. Esta relação acontece porque cada nodo do BDD é um multiplexador que têm três nodos de AIG (três portas lógicas de duas entradas, o inversor não conta pois no AIG as inversões estão nos arcos, não nos nodos). Assim é interessante diminuir o número de nodos do BDD para diminuir o número de nodos do AIG derivado.

A construção de um ROBDD, aplicando regras de redução, leva a um partilhamento de nodos, reduzindo portanto o número total de nodos. O Fato de o ROBDD ser ordenado faz com que o tamanho do grafo dependa da ordem das variáveis. Assim, no escopo desta dissertação são explorados vários (100 ordenamentos

distintos) ordenamentos de variáveis distintos escolhidos randomicamente para construir o ROBDD e é escolhido aquele que resulta em um ROBDD com menor número de nodos. Um trabalho futuro é utilizar um algoritmo de reordenamento.

2.6.2.2 Reduções propostas

Outra otimização possível no fluxo proposto são reduções do caso geral do multiplexador. Cada nodo do BDD pode ser transformado em um multiplexador que contém três nodos de AIG,no caso geral.

A principal contribuição desta dissertação é mostrar que oito tipos distintos de simplificações são possíveis no multiplexador de três nodos para circuitos de apenas dois, um ou até zero nodos de AIG, dependendo de condições no BDD. Estas reduções são apresentadas no capítulo 4.

2.6.3 Métricas

Como o objetivo do concurso do IWLS era reduzir o número de nodos de AIG necessários para implementar funções Booleanas, esta é a métrica adotada nesta dissertação. Esta é uma métrica bastante adotada e aceita na comunidade de síntese lógica, e existe um sentimento generalizado desta comunidade em acreditar que o número denodos de AIG enquanto métrica de otimização de rede lógica se traduz bem na área do circuito final na síntese física. Ou seja: Um AIG com número menor de nodos se traduz em um circuito menor após o mapeamento tecnológico para bibliotecas de células. Este não é o caso do número de literais como métrica.

2.7 Relação com os trabalhos do grupo LogiCS

Esta seção descreve o contexto mais amplo desta dissertação. Vamos descrever este contexto geral a partir dos trabalhos do grupo LogiCS. A dissertação foca no tópico de síntese lógica (REIS; DRECHSLER, 2018) de circuitos digitais (WAGNER; REIS; RIBAS, 2006). A síntese lógica transforma uma descrição lógica inicial do circuito em uma rede de primitivas lógicas tais como células de bibliotecas (TOGNI et al., 2002). A etapa de síntese lógica que cria uma rede de células em uma determinada tecnologia é conhecida como mapeamento tecnológico (REIS et

al., 1997; REIS, 1999; CORREIA; REIS, 2004). A síntese lógica é uma etapa importante no fluxo de projeto de circuitos digitais tais como microprocessadores (ROSA et al., 2003). As células da biblioteca são feitas de transistores, e a síntese lógica pode ser usada para sintetizar redes de transistores para as células da biblioteca (JUNIOR et al., 2006; da Silva; REIS; RIBAS, 2009; ROSA et al., 2007; BUTZEN et al., 2010a; BUTZEN et al., 2012; ROSA et al., 2009).

A síntese eficiente de redes de transistores pode ser importante para otimizar custos relacionados a área (POLI et al., 2003), variabilidade (da Silva; REIS; RIBAS, 2009; BUTZEN et al., 2010a; BUTZEN et al., 2012) e potência (BUTZEN et al., 2010b). Métodos propostos no grupo LogiCS, tais como síntese baseada em cortes KL (MACHADO et al., 2012) e síntese baseada em composição funcional (MARTINS; RIBAS; REIS, 2012) são usados em novas tecnologias (NEUTZLING et al., 2013; MARRANGHELLO et al., 2015; NEUTZLING et al., 2015; NEUTZLING et al., 2018; NEUTZLING et al., 2019), circuitos robustos (GOMES et al., 2014; GOMES et al., 2015) e circuito assíncronos (MOREIRA et al., 2014).

Este trabalho se encaixa no contexto do grupo LogiCS ao apresentar uma nova forma de gerar circuitos a partir de BDDs. Vários trabalhos do grupo fizeram isto tendo como objetivo a geração de redes de transistores. No caso deste trabalho o objetivoé gerar redes de portas lógicas simples para minimizar o número de nodos de um AIG derivado estruturalmente de um ROBDD.

2.8 Contribuições deste capítulo

Este capítulo apresentou conceitos básicos de síntese lógica necessários ao entendimento desta dissertação. Foram discutidos tipos de funções Booleanas,bem como tipos de estruturas de dados para representação de funções Booleanas.

3 TRABALHOS ANTERIORES

3.1 Sobre este capítulo

Este capítulo revisa trabalhos anteriores que estão ligados a esta dissertação. São discutidos métodos de síntese lógica baseados em diferentes estruturas de dados, incluindo AIGs, MIGs e BDDs.

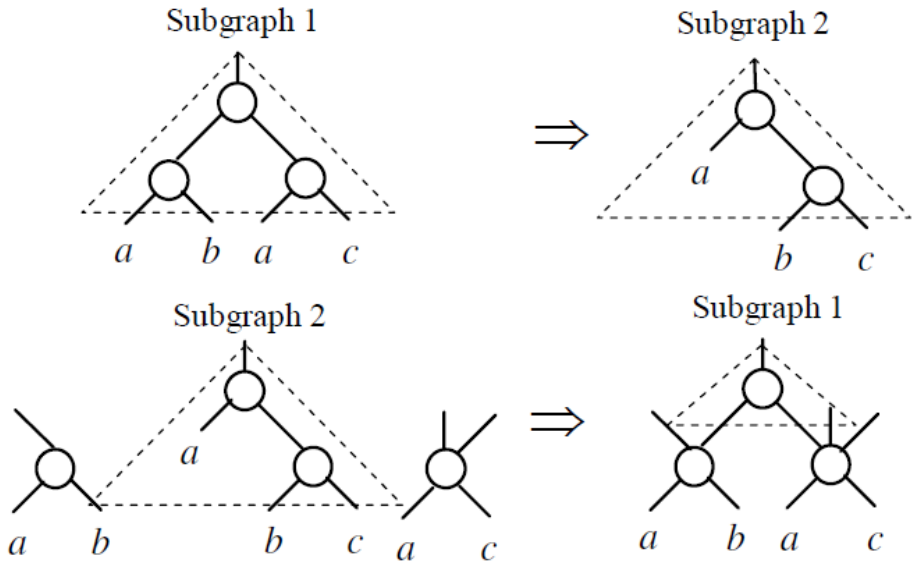
3.2 Métodos baseados em AIG

A maior parte dos métodos modernos de síntese lógica são baseados em grafos de ANDs e inversores (AIGs, de And-Inverter-Graphs em inglês). Por exemplo, a ferramenta estado da arte ABC desenvolvida em Berkeley é fortemente baseada em AIGs. AIGs são uma estrutura de dados simples onde todos os nodos são ANDs de duas entradas e, portanto, todos os nodos ocupam o mesmo espaço de memória. Esta uniformidade de tamanho faz com que AIGs possam ser representados como vetores e processados vetorialmente.

A síntese lógica em AIGs se baseia bastante na reescrita de AIGs, onde pequenos pedaços do grafo são substituídos por pedaços equivalentes (logicamente equivalentes), mas com estrutura mais favorável (MISHCHENKO; CHATTERJEE; BRAYTON, 2006). Esta substituição repetitiva faz com que otimizações locais produzam otimizações globais com qualidade significativa.

A figura 3.1 apresenta duas possibilidades de reescrita de AIGs com vantagem, mas na direção inversa (depende do contexto). A reescrita do subgrafo 1 para o subgrafo 2 (parte superior da figura) é vantajosa quando não é afetada pela existência de outros nodos similares no AIG. Neste caso se passa de três para dois nodos de AIG no total. Já a reescrita do subgrafo 2 para o subgrafo 1 (parte inferior da figura) é vantajosa quando é afetada pela existência de outros nodos similares no AIG, que podem ser reaproveitados. Neste caso se passa de um total de quatro para três nodos de AIG.

Figura 3.1: Duas possibilidades de reescrita de AIGs com vantagem, mas na direção inversa (depende do contexto). Fonte: (MISHCHENKO; CHATTERJEE; BRAYTON, 2006)



3.3 Métodos baseados em MIG

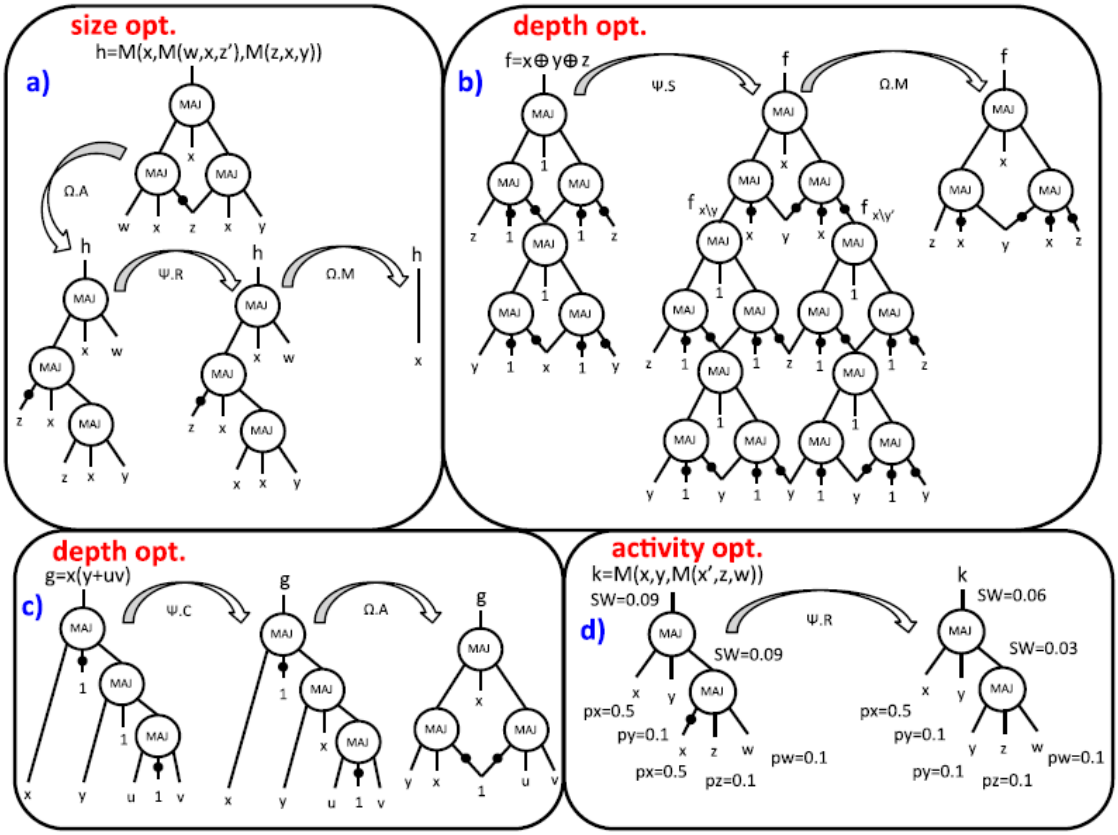
Um grafo de majoritárias e inversores (MIG, de majority-inverter graph em inglês) é um grafo acíclico dirigido que consiste em nós de maioria de três entradas e arestas regulares/complementadas. Um MIG é uma rede lógica homogênea com grau igual a 3 e cada nó representando a função majoritária. Em um MIG, as arestas são marcadas por um atributo regular ou complementado.

A estrutura de dados do tipo MIG foi proposta como uma mudança de paradigma na representação e otimização para síntese lógica, usando apenas portas majoritárias (MAJ) e inversores (INV) como operações básicas (AMARú; GAILLARDON; MICHELI, 2016). O uso de majoritárias como nodos permite a manipulação do grafo diretamente com algebra de majoritárias.

A motivação para o uso de grafos baseados em majoritárias acontece porque cadeias de carry em circuitos aritméticos são formadas por portas majoritárias. Assim, otimizações baseadas em MIG usam transformações complexas em álgebra de majoritárias para reescrever diretamente este tipo de circuitos (aritméticos) que tem uma reputação de serem circuitos difíceis para ferramentas de otimização que não se baseiam em majoritárias.

A figura 3.2 apresenta possibilidades de reescrita de MIGs, para otimização de diferentes critérios de otimização. Para maiores detalhes veja a descrição do

Figura 3.2: Possibilidades de reescrita de MIGs, para otimização de diferentes critérios de otimização. Fonte: (AMARú; GAILLARDON; MICHELI, 2016)



trabalho original de Luca Amaru (AMARú; GAILLARDON; MICHELI, 2016).

3.4 Métodos baseados em BDDs

Existem vários modos de gerar um circuito a partir de um BDD representando a lógica. Estes métodos podem se dividir entre métodos não-estruturais, métodos estruturais e métodos mistos. A seguir, exemplos de métodos conforme esta classificação serão apresentados.

3.4.1 Métodos não estruturais baseados em BDDs

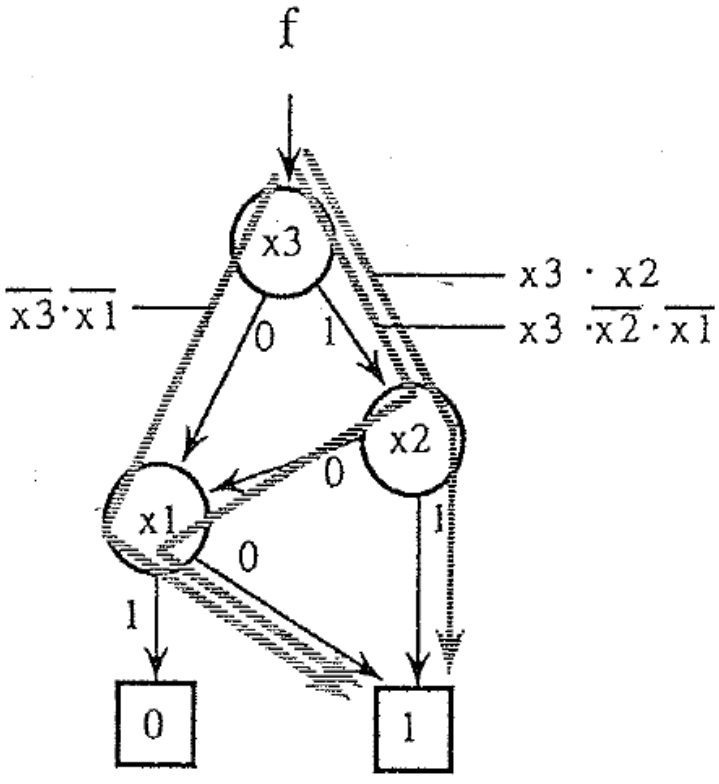
Por métodos não-estruturais entendemos aqui métodos que geram um circuito que não guarda equivalência com a estrutura do BDD.

3.4.1.1 Método de Minato Monrealle para SOPs

O método de Minato-Monrealle gera uma soma de produtos (SOP) a partir de um BDD (MINATO, 1993). A figura 3.2 mostra que para derivar uma soma de produtos (SOP) de um BDD, é possível enumerar os caminhos que levam a T1, mas esta SOP pode ser simplificada. O método proposto por Minato demonstra como fazer esta simplificação de forma rápida.

Como a soma de produtos gerada pelo método de Minato é simplificada, a SOP não guarda relação direta com a estrutura do BDD. Assim, este método pode ser considerado não estrutural já que a relação inicial com os caminhos do BDD é desfeita na simplificação da SOP.

Figura 3.3: Para derivar uma soma de produtos (SOP) de um BDD, é possível enumerar os caminhos que levam a T1, mas esta SOP pode ser simplificada. Fonte: (MINATO, 1993)



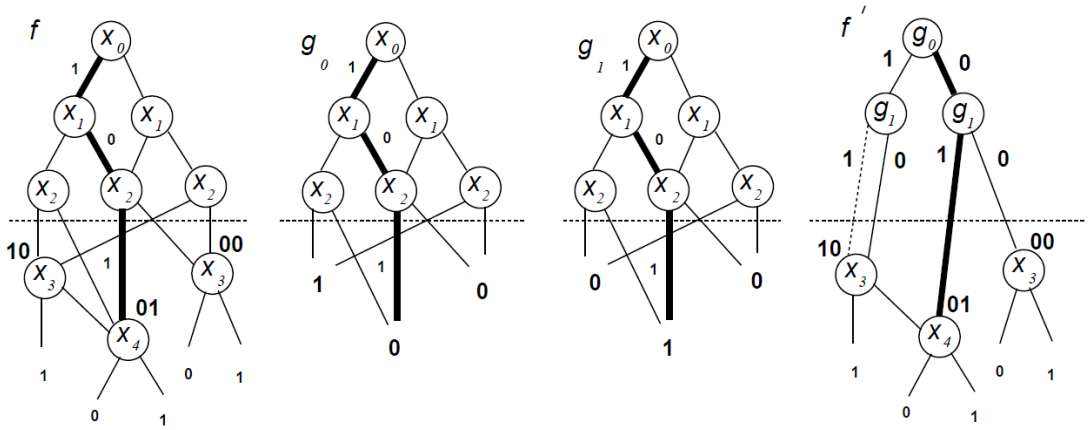
3.4.1.2 Decomposição funcional baseada em BDDs

Métodos de decomposição funcional baseados em BDDs quebram BDDs grandes em BDDs de tamanho mais aceitável (LAI; PAN; PEDRAM, 1996). Neste processo são introduzidas variáveis intermediárias que criam um conjunto de BDDs que

já não guarda relação com a estrutura dos BDDs iniciais.

No BDD mostrado as 3 variáveis superiores (X_1, X_2 e X_3) apontam para somente 3 nodos abaixo da linha de corte. Estes três nodos podem ser endereçados com códigos de dois bits (10, 01 e 00), conforme mostrado no BDD original mais a esquerda. Assim é possível reescrever o topo do BDD com duas funções g_0 e g_1. Da esquerda para a direita a figura mostra o BDD original (em função de X_1, X_2 e X_3), o BDD para a função intermediária g_0, o BDD para a função intermediária g_1, e finalmente o BDD reescrito em função de g_0 e g_1. Para maiores detalhes veja (LAI; PAN; PEDRAM, 1996).

Figura 3.4: No BDD mostrado as 3 variáveis superiores (X_1, X_2 e X_3) apontam para somente 3 nodos abaixo da linha de corte. Assim é possível reescrever o topo do BDD com duas funções g_0 e g_1. Fonte: (LAI; PAN; PEDRAM, 1996)



3.4.2 Métodos estruturais baseados em BDDs

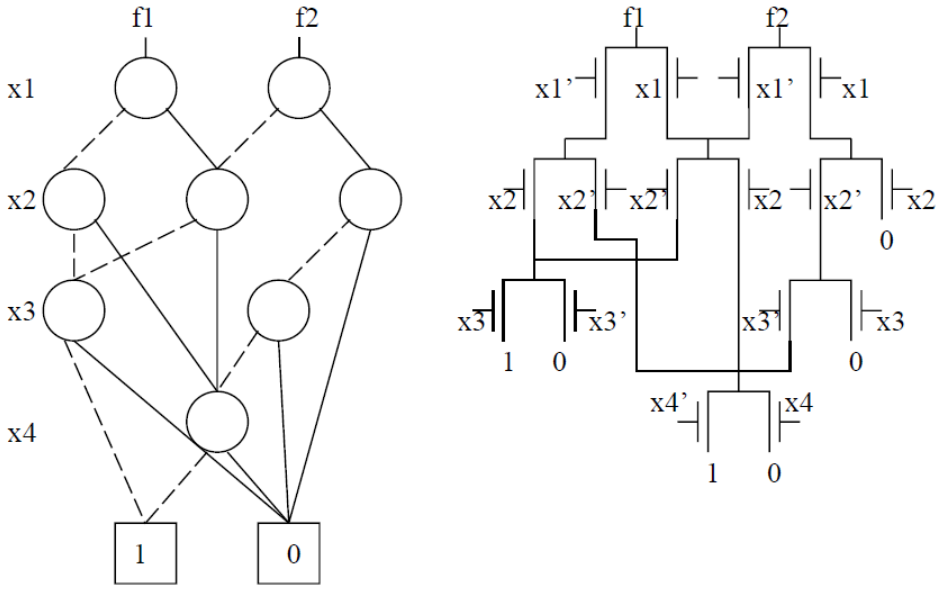
Por métodos estruturais entendemos aqui métodos que geram um circuito que guarda equivalência com a estrutura do BDD. As próximas subseções descrevem métodos que recaem nesta classificação.

3.4.2.1 Métodos baseados em Lógica de Transistor de passagem

Métodos baseados em lógica de transistor de passagem (PTL, Pass Transistor Logic, em inglês) são bastante estruturais, pois guardam forte relação com a estrutura do BDD (BERTACCO et al., 1997). Esta relação se dá porque cada arco do BDD é substituído por um transistor de passagem, de modo a formar uma rede de transistores que segue diretamente a estrutura do BDD.

A figura 3.5 mostra a relação entre um BDD e a rede PTL derivada. Note que existe um transistor de passagem para cada arco do BDD, e a estrutura é a mesma.

Figura 3.5: Relação estrutural entre um BDD e a rede PTL derivada. Fonte: (BERTACCO et al., 1997)



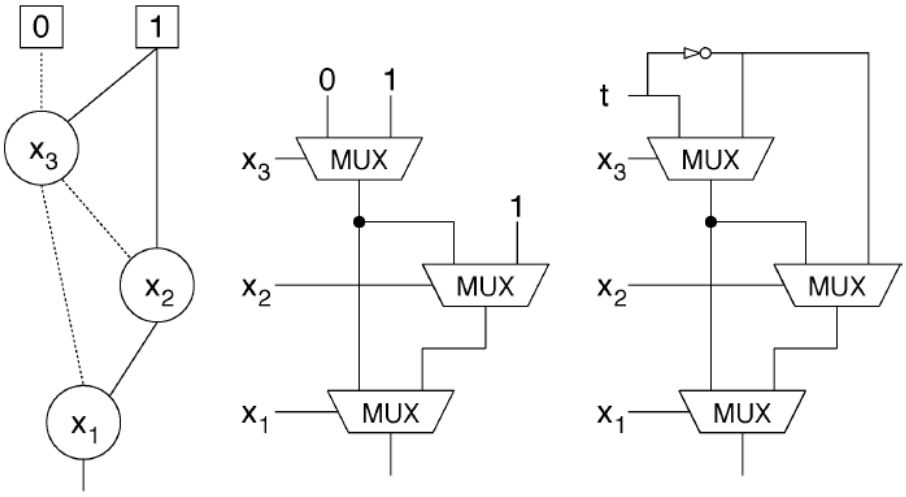
3.4.2.2 Circuitos baseados em multiplexadores

Neste tipo de método, cada nodo é substituído por um multiplexador (DRECHSLER; SHI; FEY, 2004). Portanto é um método estrutural, já que a correspondência com a estrutura estará representada por esta associação. A figura 3.6 apresenta dois circuitos baseados em multiplexadores derivados estruturalmente de um BDD. O primeiro circuito é um circuito regular, enquanto o segundo é um circuito 100% testável.

O método apresentado na dissertação se enquadra aqui e resultou em uma publicação no ISVLSI 2022 (BRANDãO et al., 2022). A contribuição proposta na dissertação, discutida no próximo capítulo, é demonstrar que simplificações são possíveis nos multiplexadores.

A abordagem proposta por Drechsler (DRECHSLER; SHI; FEY, 2004) é utilizar circuitos compostos de multiplexadores a partir de BDDs para obter circuitos 100% testáveis. Um possível trabalho futuro é verificar se as simplificações que estamos propondo podem se aplicar neste contexto, mesmo que com modificações.

Figura 3.6: Relação estrutural entre um BDD e a rede de multiplexadores derivada. Fonte: (DRECHSLER; SHI; FEY, 2004)



3.4.2.3 Métodos para gerar redes para portas lógicas

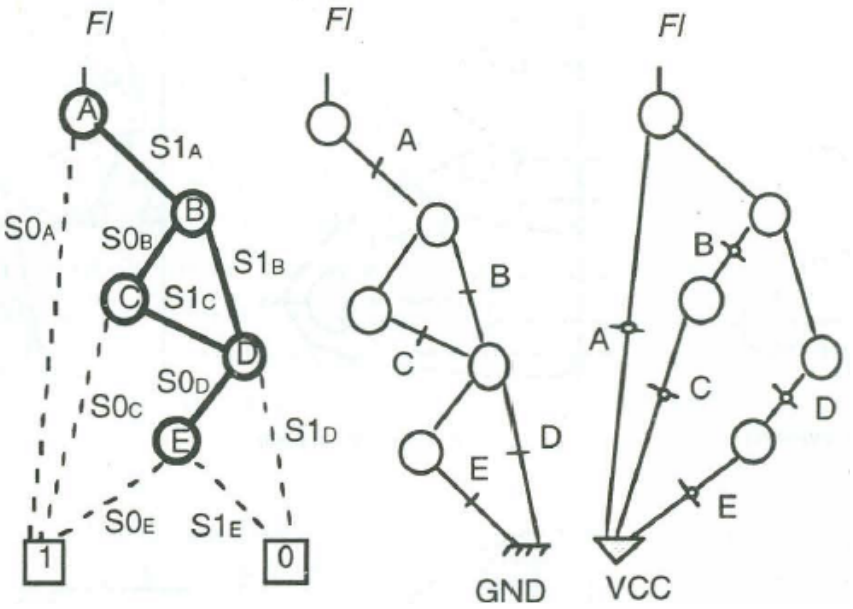
Alguns métodos geram redes para portas lógicas. Na abordagem proposta por Andre Reis (REIS et al., 1997) um BDD é composto por associações do tipo E/OU é possível extrair sua lógica por decomposição de E/OU para gerar redes série/paralelo. Além disso, o método segue o principio de realizar cortes no BDD, para limitar o número de transistores em série nas redes das portas lógicas. Neste caso o método é estrutural porque o BDD é construído para ter uma estrutura que permita a decomposição AND/OR. Porém o método não precisava ser implementado com BDDs e teve uma versão melhorada refatorada usando árvores como estrutura de dados depois de alguns anos (CORREIA; REIS, 2004).

Quanto a geração de redes de transistores para portas lógicas individuais, métodos mais avançados a partir de BDDs podem ser encontrados em (POLI et al., 2003) e (CALLEGARO et al., 2010). Estes métodos se inspiram parcialmente nos métodos de geração de redes de transistores a partir de BDDs propostos por Andre Reis (REIS et al., 1997). A figura 3.7 ilustra a relação estrutural entre um BDD e as redes de transistores derivadas para o pull-down e para o pull-up de uma célula.

3.5 Métodos mistos

Métodos mistos usam uma abordagem que é parte estrutural e parte não estrutural. Uma característica poderia ser aplicar decomposição funcional (não estru-

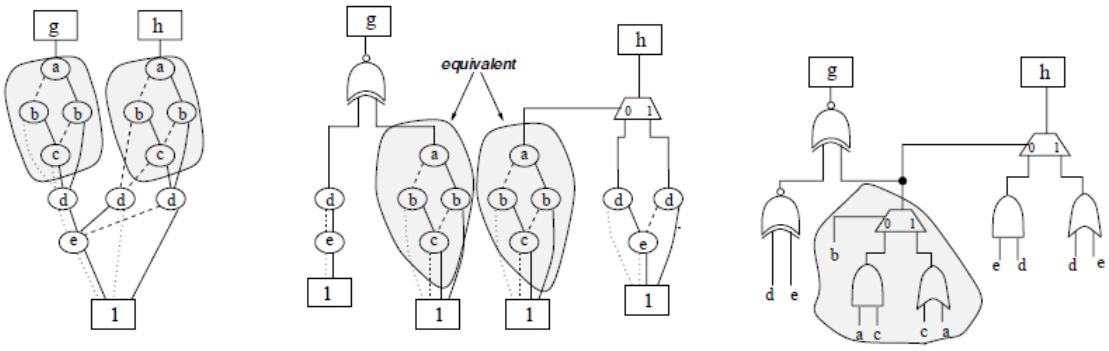
Figura 3.7: Relação estrutural entre um BDD e as redes de transistores derivadas para o pull-down e para o pull-up de uma célula. Fonte: (REIS et al., 1997)



tural) e depois gerar um circuito para cada BDD da decomposição com um método estrutural. Outra possível característica seria gerar parte do circuito com multiplexadores e parte com portas lógicas, em uma abordagem distinta mas que guarda semelhanças com a apresentada nesta dissertação.

A proposta de Yang e Ciesielski (YANG; CIESIELSKI, 2000) engloba estas duas características conforme se pode observar na figura 3.8. Para maiores detalhes recomenda-se a leitura da referência (YANG; CIESIELSKI, 2000).

Figura 3.8: BDD e circuitos derivados de forma mista entre PTL e portas lógicas (CMOS). Fonte: (YANG; CIESIELSKI, 2000)



3.6 Contribuições deste capítulo

Este capítulo discutiu e classificou as principais formas de se gerar circuitos digitais a partir de BDDs. A proposta desta dissertação foi enquadrada nesta discussão e classificação.

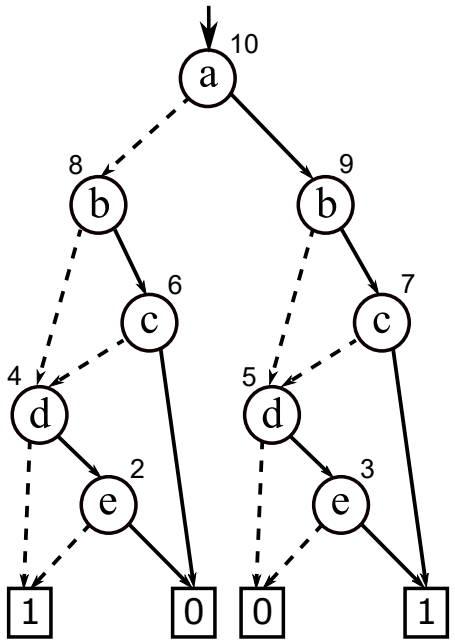
4 REDUÇÕES PROPOSTAS

4.1 Sobre este capítulo

Este capítulo apresenta as reduções propostas para gerar circuitos baseados em multiplexadores a partir de BDDs. Também é descrito brevemente como as simplificações propostas podem ser consideradas em um método. Estas são as principais contribuições da dissertação.

4.2 Um exemplo construído para apresentar todos os casos

Figura 4.1: Um exemplo de ROBDD construído para apresentar todos os casos de redução propostos neste trabalho, bem como um nodo onde as reduções não se aplicam.



Nesta seção é apresentado um exemplo de BDD que foi construído para apresentar todos os casos de redução propostos neste trabalho, bem como um nodo onde as reduções não se aplicam. Este exemplo será usado para explicar a geração de circuitos baseados em multiplexadores a partir de BDDs, com ou sem as reduções propostas. O exemplo é apresentado na Figura 4.1, e é composto por nove nodos não-terminais numerados de 2 a 10. Se forem utilizados multiplexadores, sem reduções, o circuito teria três portas (sem contar inversores) por nodo não terminal, totalizando 27 portas. No entanto, todos os nodos não terminais têm possíveis re-

duções a serem aplicadas. O único nodo para o qual uma célula genérica de 3 portas para um multiplexador 2x1 é necessário para o nodo raiz 10. Na próxima seção discutiremos como gerar circuitos para todos os nodos do BDD da Figura 4.1.

4.3 Reduções propostas neste trabalho

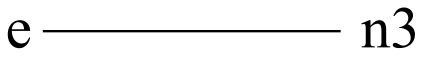
Como descrevemos na seção anterior, nem todos nodos BDD vão precisar de um multiplexador genérico como da Figura 4.18. Nas próximas subseções iremos descrever oito possíveis reduções, seguido pelo caso genérico quando nenhuma redução é possível.

4.3.1 Redução 1 - Map1

O primeiro tipo de redução pode ser aplicado quando $f_{x=1} = 1$ e $f_{x=0} = 0$. Este é o caso do nodo 3 controlado pela variável e, como $n3_{e=1} = 1$ e $n3_{e=0} = 0$. Neste caso, a equação para o caso genérico é $n3(e) = (e \cdot n3_{e=1}) + (\overline{e} \cdot n3_{e=0})$ e ela pode ser reduzida a um fio expresso pela equação 4.1, como mostra a figura 4.2.

$$n3(e) = e \tag{4.1}$$

Figura 4.2: Redução a um fio $n3(e) = e$.

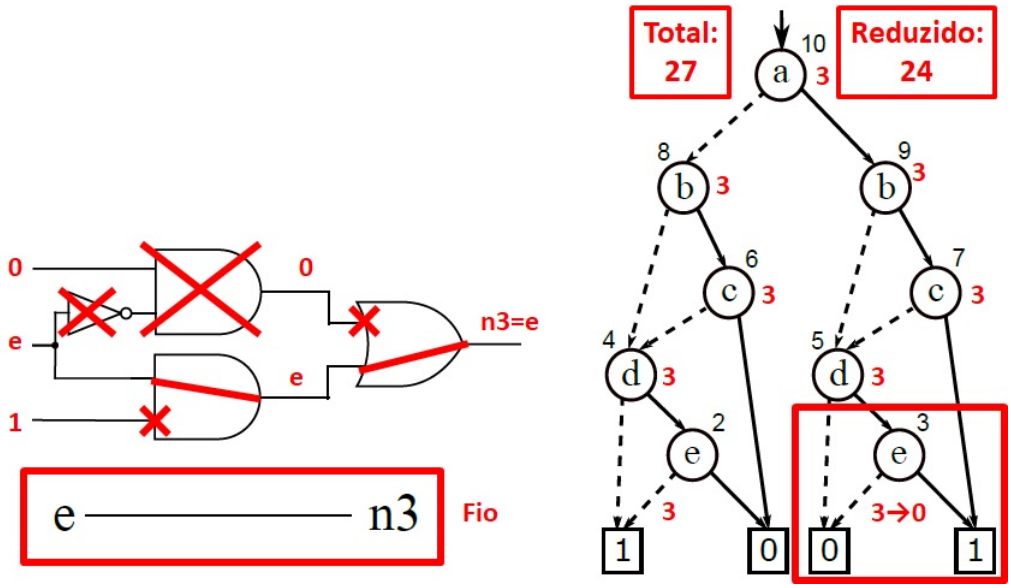


A figura 4.3 mostra a redução 1 no contexto do BDD exemplo. A redução map1 pode ser aplicada no nodo 3 do BDD exemplo. Esta redução resulta em uma economia de três nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por um fio, que corresponde a zero nodos de AIG.

4.3.2 Redução 2 - Map2

O segundo tipo de redução pode ser aplicado quando $f_{x=1} = 0$ e $f_{x=0} = 1$. Este é o caso do nodo 2, controlado pela variável e, como $n2_{e=1} = 0$ e $n2_{e=0} = 1$. Neste caso a equação para o caso genérico é $n2(e) = (e \cdot n2_{e=1}) + (\overline{e} \cdot n2_{e=0})$, que

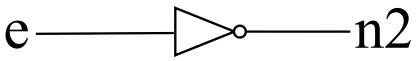
Figura 4.3: Redução map1 no contexto do BDD exemplo aplicada no nodo 3, resultando em uma economia de três nodos de AIG.



pode ser reduzida a um inversor expresso pela equação 4.2, como mostra a Figura 4.4.

$$n2(e) = \overline{e} \tag{4.2}$$

Figura 4.4: Redução a um inversor $n2(e) = \overline{e}$.

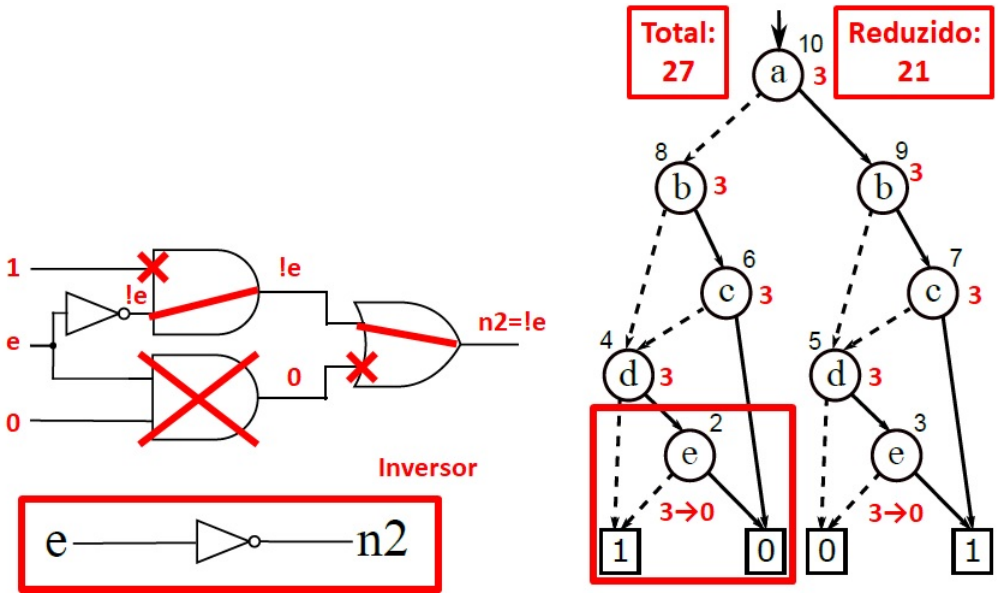


A figura 4.5 mostra a redução 2 no contexto do BDD exemplo. A redução map2 pode ser aplicada no nodo 2 do BDD exemplo. Esta redução resulta em uma economia de três nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por um inversor, que corresponde a zero nodos de AIG. Ressaltamos que em um AIG os inversores não são nodos, mas sim etiquetas em arcos.

4.3.3 Redução 3 - Map3

O terceiro tipo de redução pode ser aplicado quando $f_{x=0} = 0$. Este é o caso do nodo 5, controlado pela variável d, como $n5_{d=0} = 0$. Neste caso a equação para o caso genérico é $n5(d) = (d \cdot n5_{d=1}) + (\overline{d} \cdot n5_{d=0})$, no qual pode ser reduzida a uma *and* de 2 entradas expressa pela equação 4.3, como mostra a Figura 4.6. Note que

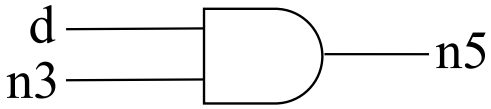
Figura 4.5: Redução map2 no contexto do BDD exemplo aplicada no nodo 2, resultando em uma economia de três nodos de AIG.



a função $n3$ é dada pela equação 4.1.

$$n5(d) = d \cdot n5_{d=1} = d \cdot n3 \tag{4.3}$$

Figura 4.6: Redução a uma *and* de 2 entradas $n5(d) = d \cdot n3$.

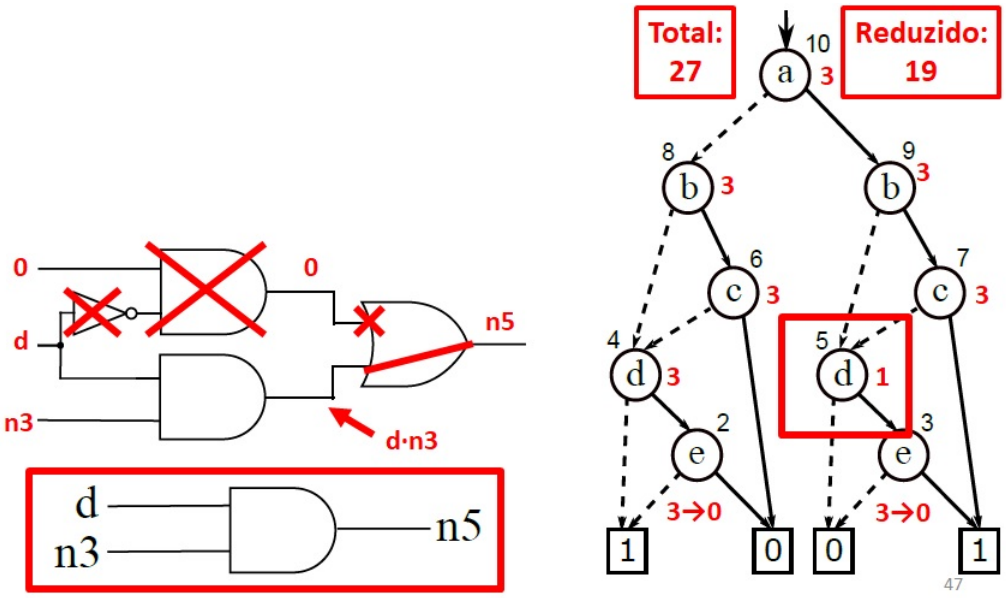


A figura 4.7 mostra a redução 3 no contexto do BDD exemplo. A redução map3 pode ser aplicada no nodo 5 do BDD exemplo. Esta redução resulta em uma economia de dois nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta AND2, que corresponde a um nodo de AIG.

4.3.4 Redução 4 - Map4

O quarto tipo de redução pode ser aplicado quando $f_{x=0} = 1$. Este é o caso do nodo 4, controlado pela variável d, como $n4_{d=0} = 1$. Neste caso, a equação para o caso genérico é $n4(d) = (d \cdot n4_{d=1}) + (\overline{d} \cdot n4_{d=0})$, no qual pode ser reduzada a uma *or* de 2 entradas expressa pela equação 4.4, como mostra a Figura 4.8. Note que a

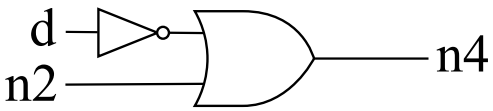
Figura 4.7: Redução map3 no contexto do BDD exemplo aplicada no nodo 5, resultando em uma economia de dois nodos de AIG.



função $n2$ é dada pela equação 4.2.

$$n4(d) = d \cdot n4_{d=1} + \overline{d} = n4_{d=1} + \overline{d} = n2 + \overline{d} \tag{4.4}$$

Figura 4.8: Redução a uma porta *or* de 2 entradas e um inversor $n4(d) = n2 + \overline{d}$.

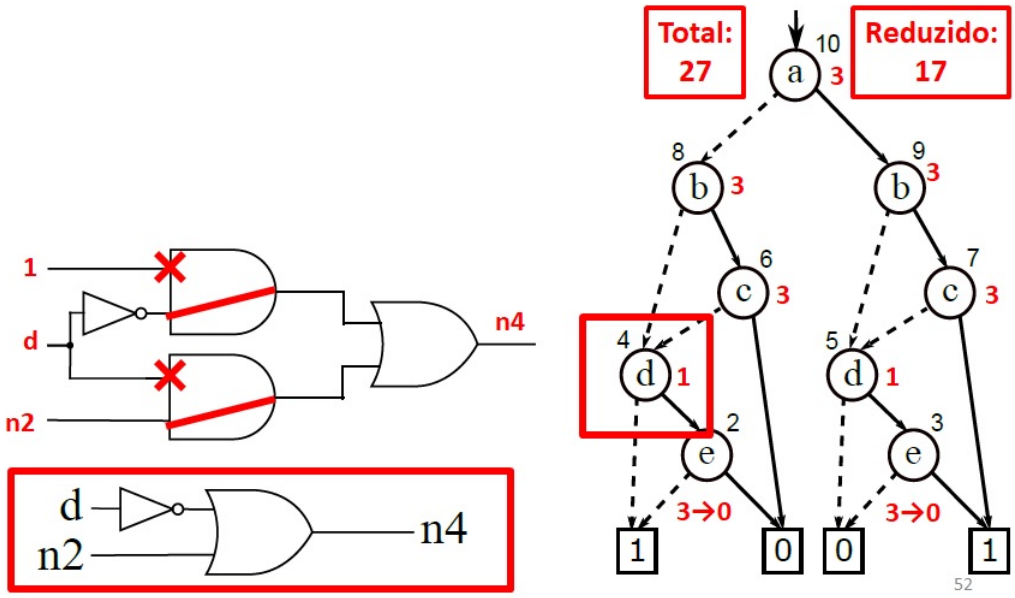


A figura 4.9 mostra a redução 4 no contexto do BDD exemplo. A redução map4 pode ser aplicada no nodo 4 do BDD exemplo. Esta redução resulta em uma economia de dois nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta OR2 e um inversor, que correspondem a um nodo de AIG. Ressaltamos que em um AIG os inversores não são nodos, mas sim etiquetas em arcos.

4.3.5 Redução 5 - Map5

O quinto tipo de redução pode ser aplicado quando $f_{x=1} = 0$. Este é o caso do nodo 6, controlado pela variável c, como $n6_{c=1} = 0$. Neste caso, a equação para o caso genérico é $n6(c) = (c \cdot n6_{c=1}) + (\overline{c} \cdot n6_{c=0})$, no qual pode ser reduzida a uma

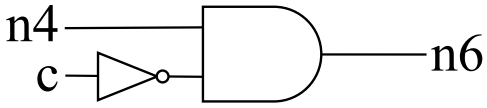
Figura 4.9: Redução map4 no contexto do BDD exemplo aplicada no nodo 4, resultando em uma economia de dois nodos de AIG.



and de duas entradas expressa pela equação 4.5, como mostra a Figura 4.10. Note que a função $n4$ é dada pela equação 4.4.

$$n6(c) = \overline{c} \cdot n6_{c=0} = \overline{c} \cdot n4 \tag{4.5}$$

Figura 4.10: Redução a uma porta *and* de 2 entradas e um inversor $n6(c) = n4 + \overline{c}$.

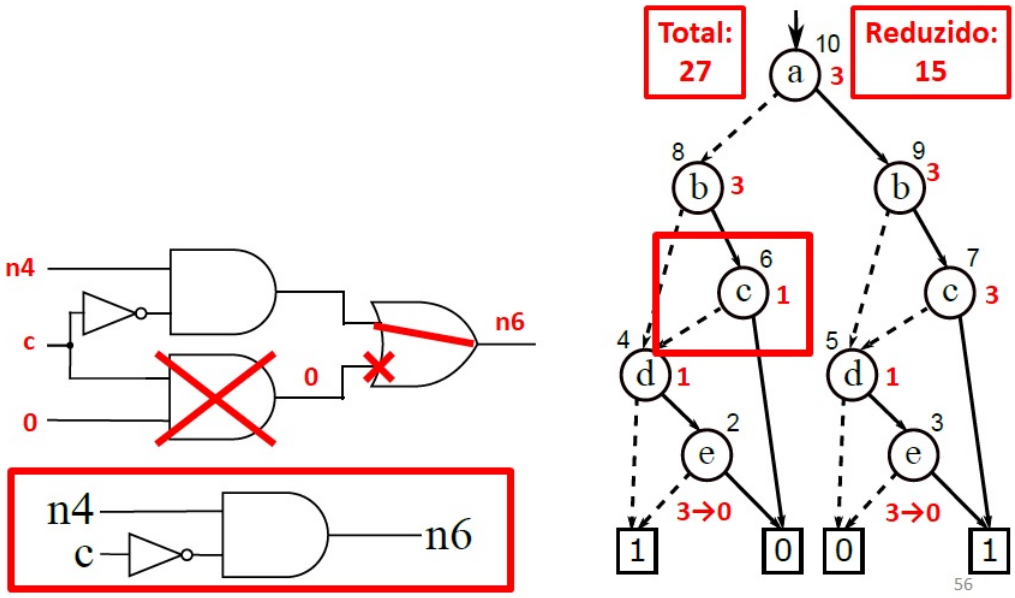


A figura 4.11 mostra a redução 5 no contexto do BDD exemplo. A redução map5 pode ser aplicada no nodo 6 do BDD exemplo. Esta redução resulta em uma economia de dois nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta AND2 e um inversor, que correspondem a um nodo de AIG. Ressaltamos que em um AIG os inversores não são nodos, mas sim etiquetas em arcos.

4.3.6 Redução 6 - Map6

O sexto tipo de redução pode ser aplicado quando $f_{x=1} = 1$. Este é o caso do nodo 7, controlado pela variável c, como $n7_{c=1} = 1$. Neste caso, a equação para

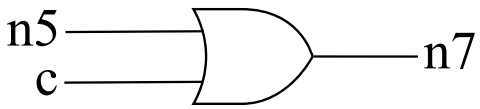
Figura 4.11: Redução map5 no contexto do BDD exemplo aplicada no nodo 6, resultando em uma economia de dois nodos de AIG.



o caso genérico é $n7(c) = (c \cdot n7_{c=1}) + (\overline{c} \cdot n7_{c=0})$, no qual pode ser reduzida a uma *or* de 2 entradas expressa pela equação 4.6, como mostra a Figura 4.12. Note que a função $n5$ é dada pela equação 4.3.

$$n7(c) = c + \overline{c} \cdot n7_{c=0} = c + n7_{c=0} = c + n5 \tag{4.6}$$

Figura 4.12: Redução a uma porta *or* de duas entradas $n7(c) = n5 + c$.

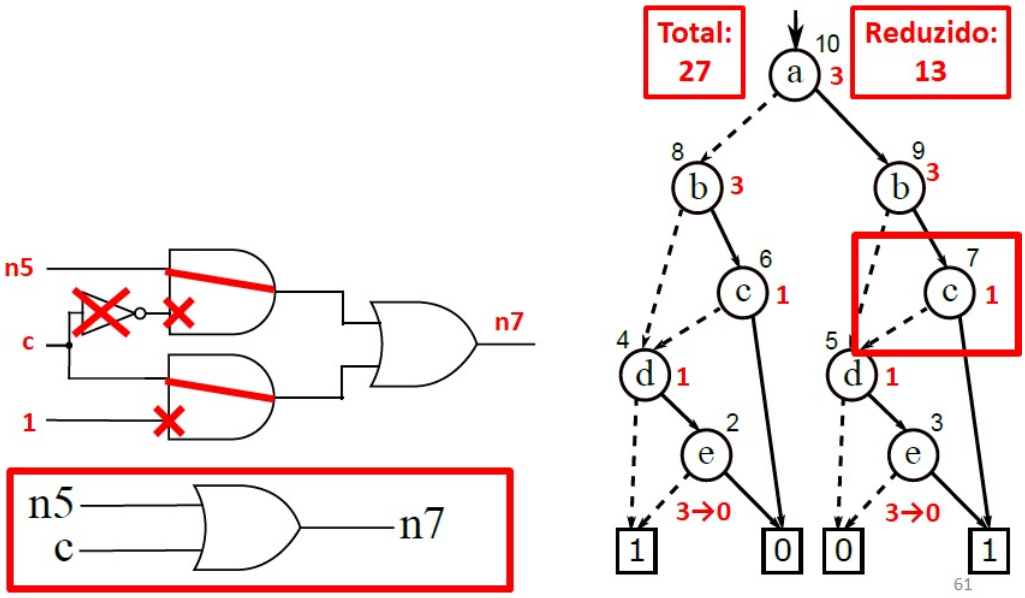


A figura 4.13 mostra a redução 6 no contexto do BDD exemplo. A redução map6 pode ser aplicada no nodo 7 do BDD exemplo. Esta redução resulta em uma economia de dois nodos de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta NOR2, que corresponde a um nodo de AIG.

4.3.7 Redução 7 - Map7

Esta simplificação é permitida quando a função é *unate* negativa em relação à variável que controla o nodo. Por exemplo, considere o nodo 8 na Fig. 4.15, controlado pela variável b. A subfunção com raiz no nodo 8 é *unate* negativa na

Figura 4.13: Redução map6 no contexto do BDD exemplo aplicada no nodo 7, resultando em uma economia de dois nodos de AIG.



variável b. A propriedade de *unateness* negativo pode ser testada fazendo um OR lógico entre as duas funções apontadas pelo nodo 8: o cofator negativo $n8_{b=0} = n4$ e o cofator positivo $n8_{b=1} = n6$. Quando o OR entre os dois cofatores é igual ao cofator negativo, a função é *unate* negativa. A tabela 4.1 mostra que $(n8_{b=0} + n8_{b=1}) = (n8_{b=0})$, de modo que $n8$ é *unate* negativa na variável b .

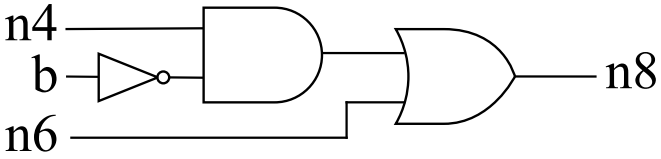
Tabela 4.1: Tabela verdade mostrando que o nodo 8 é unate negativo na variavel b, como $n8_{b=0}$ OR $n8_{b=1}$ é igual a $n8_{b=0}$.

			map7		
c	d	e	$n4 = n8_{b=0}$	$n6 = n8_{b=1}$	$n4 + n6$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	0	0	0

No nodo b na figura 4.14 será possível a aplicação desta simplificação, pois a variável b é unate positiva. Nesta derivação era necessário 12 nodos para representar a função, após simplificação reduz para 11 o números de portas lógicas desta nova

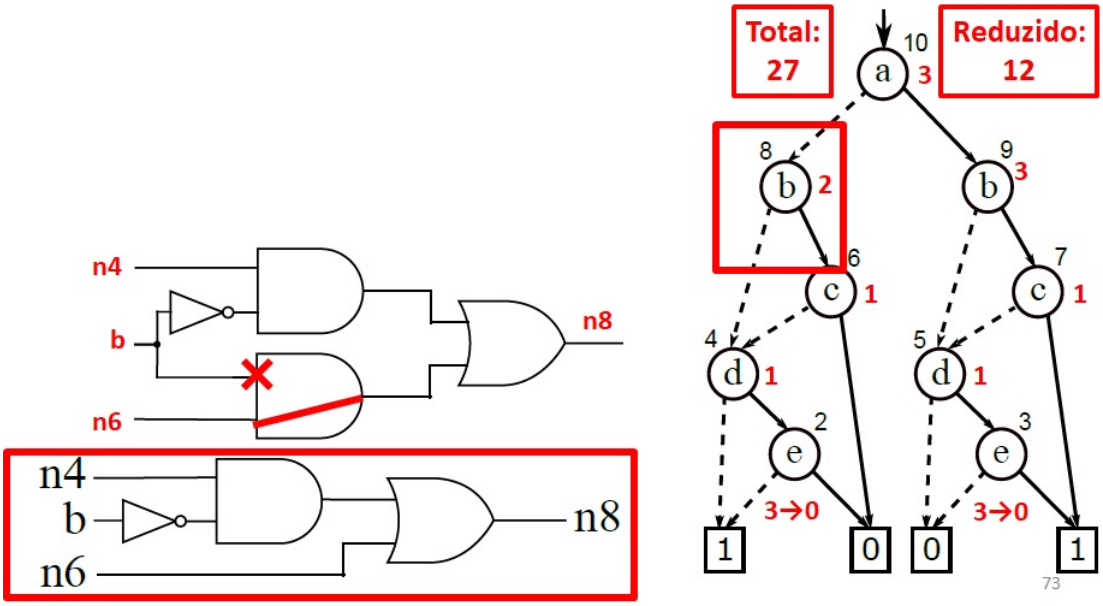
representação.

Figura 4.14: $f = \overline{b}(\overline{d} + \overline{e}) + (\overline{c}(\overline{d} + \overline{e}))$



A figura 4.15 mostra a redução 7 no contexto do BDD exemplo. A redução map7 pode ser aplicada no nodo 8 do BDD exemplo. Esta redução resulta em uma economia de um nodo de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta AND2, uma porta OR2 e um inversor, que correspondem a dois nodos de AIG. Ressaltamos que em um AIG os inversores não são nodos, mas sim etiquetas em arcos.

Figura 4.15: Redução map7 no contexto do BDD exemplo aplicada no nodo 8, resultando em uma economia de um nodo de AIG.



4.3.8 Redução 8 - Map8

Esta simplificação é permitida quando a função é *unate* positiva em relação à variável que controla o nodo. Por exemplo, considere o nodo 9 na Fig. 4.16, controlado pela variável b. A subfunção com raiz no nodo 9 é *unate* positiva na variável b. A propriedade de *unate* positiva pode ser testada fazendo um OR lógico entre as duas funções apontadas pelo nodo 9: o cofator negativo $n9_{b=0} = n5$ e o

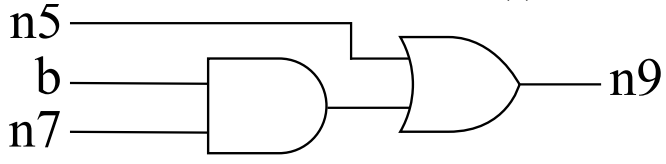
Tabela 4.2: Tabela verdade mostrando que o nodo 9 é unate positivo na variavel b, como $n9_{b=0}$ OR $n9_{b=1}$ é igual a $n9_{b=1}$.

			map8		
c	d	e	$n5 = n9_{b=0}$	$n7 = n9_{b=1}$	n5 + n7
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	1	1

cofator positivo $n9_{b=1} = n7$. Quando o OR entre os dois cofatores é igual ao cofator positivo, a função é *unate* positiva. A tabela 4.1 mostra que $(n9_{b=0} + n9_{b=1}) == (n9_{b=1})$, então $n9$ é *unate* positiva na variável b .

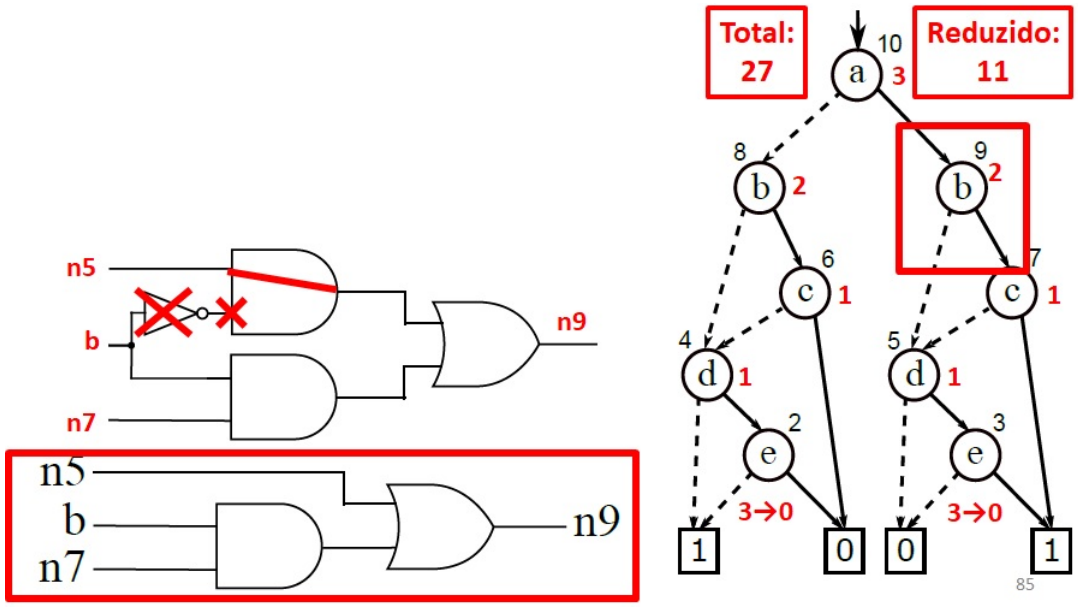
Este tipo de simplificação ocorrerá após explorarmos a função, e encontrar um nodo qualquer, e ao fazer uma OR usando $c0$ e $c1$ deste nodo, a saída produzida pela porta OR, for equivalente ao sinal do próprio $c1$, ou seja $c0 + c1 == c1$, no exemplo da figura 4.16 o nodo a passará por esse processo. Observe que neste caso de simplificação a figura 4.16, representa a função $f = |(ab + cd)$ onde o nodo a observado é unate negativo, e através desta característica será possível aplicar esta redução. Antes da redução utilizava-se 12 portas lógicas na representação da função,reduzindo para 11 este valor.

Figura 4.16: Redução unate positiva $n9(b) = n5 + b \cdot n7$



A figura 4.17 mostra a redução 8 no contexto do BDD exemplo. A redução map8 pode ser aplicada no nodo 9 do BDD exemplo. Esta redução resulta em uma economia de um nodo de AIG, já que um multiplexador, que corresponde a três nodos de AIG é substituído por uma porta AND2 e uma porta OR2, que correspondem a dois nodos de AIG.

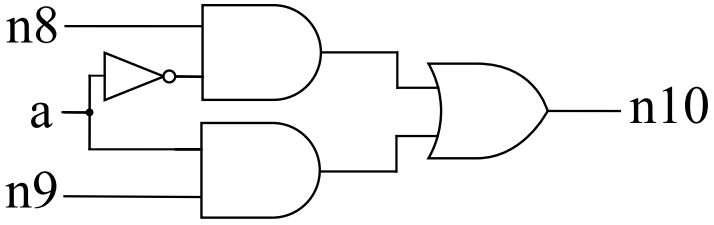
Figura 4.17: Redução map8 no contexto do BDD exemplo aplicada no nodo 9, resultando em uma economia de um nodo de AIG.



4.3.9 Caso Genérico

O caso geral acontece no nodo 10 da Figura 4.1, controlado pela variável a. Este nodo é *binate* em relação à variável a. A propriedade de binate pode ser testada se fazendo uma lógica OR entre as duas funções apontadas pelo nodo 10: o cofator negativo $n10_{a=0} = n8$ e o cofator positivo $n10_{a=1} = n9$. Quando a função OR entre os dois cofatores é diferente entre os cofatores, a função é binate. A Tabela 4.3 mostra (resumidamente em apenas duas linhas) que $n10_{a=0} + n10_{a=1}$ é diferente tanto de $n10_{a=0}$ quanto de $n10_{a=1}$, de modo que $n10$ seja binate na variável a. Nenhuma simplificação pode ser aplicada para nodos binate. O caso geral de um multiplexador 2x1 é aplicado sem reduções, conforme ilustrado na Figura 4.18.

Figura 4.18: Circuito Derivado nodo $n10(a) = a \cdot n9 + \overline{a} \cdot n8$

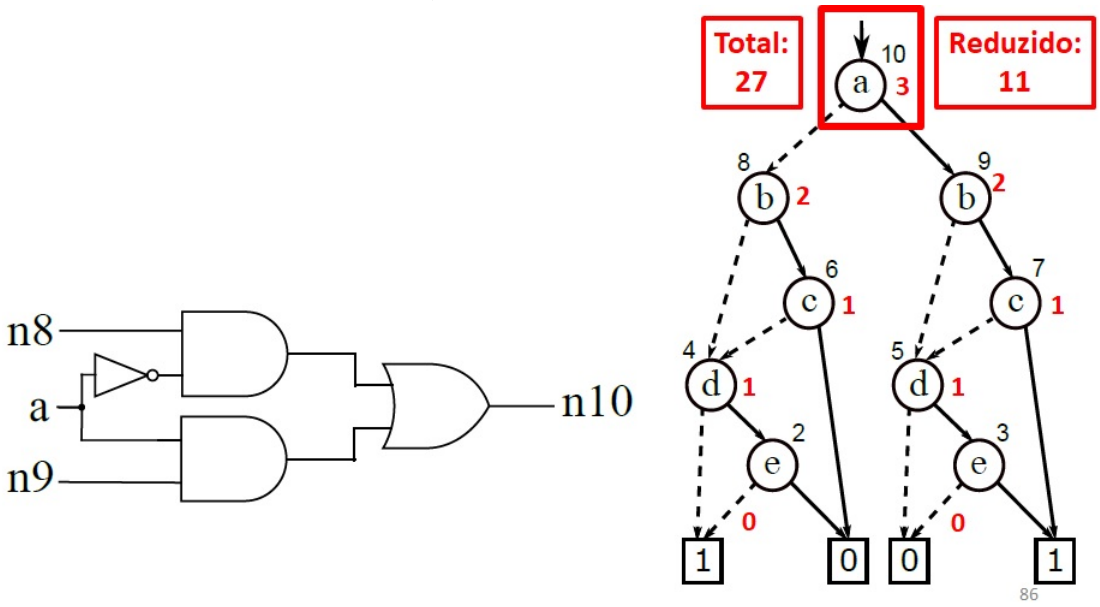


A figura 4.19 mostra o caso genérico no contexto do BDD exemplo. O caso genérico deve ser aplicado no nodo 10 do BDD exemplo. Como nenhuma redução é aplicável, se utiliza um multiplexador, que corresponde a três nodos de AIG. Neste caso não houve economia de nodos, já que as otimizações não foram aplicadas.

Tabela 4.3: Tabela verdade mostrando que $n10$ é binate.

				general case		
b	c	d	e	$n8 = n10_{a=0}$	$n9 = n10_{a=1}$	$n8 + n9$
0	0	0	0	1	0	1
1	1	1	1	0	1	1

Figura 4.19: Caso genérico no contexto do BDD exemplo aplicado no nodo 10, resultando em três nodos de AIG, sem nenhuma economia.



4.4 Método considerando as simplificações propostas

Nesta seção descrevemos brevemente como considerar as simplificações em um método de síntese. Note que isto de certa forma foi apresentado nas seções anteriores ao descrever a aplicação de cada caso no BDD exemplo. O método é bem direto, pois se aplica a cada nodo individualmente. Simplesmente se percorre o BDD e se testa as condições de cada nodo para ver se simplificações são aplicáveis. Caso simplificações não se apliquem a um determinado nodo, o caso genérico é usado, sem economia. Caso o teste das condições de cada nodo individual permitam estas simplificações, estas podem ser aplicadas. Obviamente o melhor resultado acontece quando todas as simplificações possíveis são aplicadas. Na seção de resultados, para avaliar a contribuição individual de cada simplificação, fizemos versões sub-ótimas do método onde apenas uma simplificação é permitida ou proibida.

4.5 Contribuições do capítulo

Este capitulo apresentou de forma sistêmica 8 formas de derivar circuitos digitais a partir de Diagramas de Decisão Binários (BDDs). E ao exploramos este casos, aplicando as simplificações propostas, demonstramos que a aplicação dessas simplificações resulta em AIGs menores derivados de BDDs.

5 RESULTADOS

Para verificar o efeito das reduções propostas, realizamos três experimentos. Esses experimentos foram baseados em um subconjunto dos benchmarks da EPFL (AMARÚ; GAILLARDON; MICHELI, 2015). Para cada benchmark investigado, foi gerado um BDD, com ordenação de variáveis gerada com CUDD (SOMENZI, 2018). Diferentes AIGs (de acordo com o experimento) foram gerados permitindo a aplicação de diferentes subconjuntos das reduções propostas. As três experiências são descritas a seguir.

Em nosso primeiro experimento comparamos o efeito de permitir que apenas uma das reduções possíveis fosse aplicada, comparando com o efeito de um circuito que não utiliza nenhuma das reduções propostas. Os resultados são mostrados na Tabela 5.1. A primeira coluna (*benchmark*) dá o nome do benchmark. A segunda coluna (*no map*) fornece o número de nós AIG obtidos do BDD usando a implementação do multiplexador padrão para todos os nós do BDD sem usar as reduções propostas. As colunas a seguir (rotuladas *map1* a *map8*) apresentam os resultados quando apenas uma das reduções (referenciadas pelo número) é aplicada.

Em nosso segundo experimento comparamos o efeito de excluir apenas uma das possíveis reduções a serem aplicadas, comparando com o efeito de um circuito utilizando todas as reduções propostas. Os resultados são mostrados na Tabela 5.2. A primeira coluna (*benchmark*) dá o nome do benchmark. As colunas a seguir (rotuladas *no map1* a *no map8*) apresentam os resultados quando apenas uma das reduções (referenciadas pelo número) é excluída da aplicação. A última coluna (*all maps*) fornece o número de nós AIG obtidos do BDD usando todas as reduções propostas para otimizar o número de nós AIG.

Em nosso terceiro experimento, analisamos o efeito do uso de ABC como pós-processamento para nosso método. Os AIGs resultantes (do nosso método) foram lidos no ABC e os dados (número de nós e níveis) foram extraídos usando relatórios ABC. Os resultados são mostrados na Tabela 5.3. A tabela é dividida em cinco grupos de colunas. O primeiro grupo é rotulado *benchmark description* e apresenta o nome, o número de entradas e saídas para cada benchmark. O segundo grupo de colunas (*original AIG*) exibe o número de nós AIG e níveis do benchmark AIG de origem conforme distribuído. O terceiro grupo de colunas (*initial BDD*) é uma única coluna que mostra o número de nós BDD no BDD gerado a partir do AIG

original. O quarto grupo de colunas, rotulado *AIG without proposed maps*, apresenta o número de nós AIG que é obtido diretamente do BDD sem aplicar as reduções propostas (coluna *nós*), e então o número de nós e níveis reportados após a leitura do AIG no ABC (observe que o ABC realiza otimizações no número de nós para todos os 10 benchmarks). O quinto grupo de colunas, rotulado *AIG using proposed maps* é semelhante ao quarto. No entanto, apresenta o número de nós AIG que é obtido diretamente do BDD aplicando os mapas propostos (coluna *nodes*), e então o número de nós e níveis reportados após a leitura do AIG no ABC (observe que o ABC realiza otimizações para 7 de 10 benchmarks).

Observando a tabela 5.3, é possível notar que para 6 benchmarks (*adder*, *cavlc*, *dec*, *i2c*, *mem_ctrl* e *router*) o número de nós AIG aumentou em relação ao AIG original. Isso se deve à estrutura do ROBDD, onde a restrição da estrutura de dados a uma ordenação fixa pode implicar em grandes BDDs, principalmente para grandes circuitos. Este pode ser um preço a pagar quando for desejável que o circuito tenha uma estrutura baseada em multiplexadores (KUSHNEROV; MEDINA; YAKOVLEV, 2021), devido a outras razões. Trabalhos futuros podem considerar maneiras de mitigar esse grande efeito ROBDD, por exemplo, quebrando (decompondo) o ROBDD grande em BDDs menores ou gerando o circuito a partir de BDDs não ordenados (por exemplo, BDDs livres).

Ainda de acordo com a Tabela 5.3, é possível notar que para 4 benchmarks (*bar*, *ctrl*, *int2float* e *priority*) o número de nós AIG diminui, quando comparado ao AIG original. Para os benchmarks *bar* e *int2float* o número de níveis é ligeiramente aumentado (de 12 a 14 e de 16 a 18, respectivamente). O melhor desempenho para o nosso método é verificado nos benchmarks *ctrl* e *priority*, onde tanto o número de nós quanto a profundidade são reduzidos em relação ao AIG original. No caso do benchmark *priority*, a redução do número de níveis é significativa, de 250 para 127 níveis.

Tabela 5.1: Número de nodos AIG obtidos através da aplicação de uma única redução por vez, comparado a não utilizar as reduções propostas .

Benchmark	no map	map1	map2	map3	map4	map5	map6	map7	map8
adder	3624	3603	3603	3366	3368	3368	3366	3238	3235
bar	3072	2688	3072	2816	3072	3072	2816	3072	2944
cavlc	1269	1257	1254	1215	1229	1129	1193	1141	1164
ctrl	261	255	258	205	259	225	253	238	217
dec	1530	1527	1527	1020	1528	1020	1528	1275	1275
i2c	4086	3819	3993	3706	3936	3216	3660	3528	3636
int2float	396	375	378	332	340	380	346	366	343
mem_ctrl	267621	265320	266940	260377	264873	253799	260417	238677	239983
priority	2691	2667	2691	2675	2691	2049	1539	2370	2115
router	798	792	792	702	654	650	764	614	719

Tabela 5.2: Número de nodos AIG otidos através da exclusão de uma redução por vez, comparado a aplicação de todas reduções.

benchmark	no map1	no map2	no map3	no map4	no map5	no map6	no map7	no map8	all maps
adder	2342	2342	2457	2456	2456	2457	2472	2473	2335
bar	2816	2688	2688	2688	2688	2688	2688	2688	2688
cavlc	885	886	904	896	946	915	924	925	881
ctrl	145	144	169	143	160	145	148	157	143
dec	509	509	762	508	762	508	508	508	508
i2c	2254	2196	2266	2209	2569	2289	2244	2301	2165
int2float	227	226	245	242	222	238	220	223	220
mem_ctrl	196297	195757	198385	196677	202214	198365	216416	216711	195530
priority	897	889	889	889	1210	1457	889	889	889
router	326	326	370	394	396	339	364	340	324

Tabela 5.3: Efeitos das reduções no AIG final.

Descrição do benchmark			AIG Original		BDD inicial	AIG sem os mapas propostos			AIG usando os mapas propostos		
Nome	Entradas	Saídas	nodos	níveis	nodos BDD	nodos	nodos ABC	níveis ABC	nodos	nodos ABC	níveis ABC
adder	256	129	1020	255	1208	3624	2367	510	2335	2090	384
bar	135	128	3336	12	1024	3072	2688	14	2688	2688	14
cavlc	10	11	693	16	423	1269	826	18	881	738	18
ctrl	7	26	174	10	87	261	137	10	143	126	9
dec	8	256	304	3	510	1530	508	7	508	508	7
i2c	147	142	1342	20	1362	4086	2445	41	2165	2062	35
int2float	11	7	260	16	132	396	244	20	220	200	18
mem_ctrl	1204	1231	46836	114	89207	267621	214024	280	195530	178897	221
priority	128	8	978	250	897	2691	1457	254	889	889	127
router	60	30	257	54	266	798	460	110	324	305	77

6 CONCLUSÃO E TRABALHOS FUTUROS

Catalogamos e analisamos as condições necessárias para a aplicação de um conjunto de oito reduções a serem aplicadas em circuitos baseados em multiplexadores derivados a partir de BDDs. Acreditamos que cada uma das simplificações pode aparecer separadamente, sem muita discussão e sem aplicação sistemática em BDDs na literatura anterior. Assim, uma revisão sistemática destas reduções na literatura anterior é dificultada porque tais simplificações podem aparecer sem muito alarde quase que como nota de rodapé em artigos da área. Neste sentido a sistematização proposta aqui é uma contribuição positiva. A contribuição deste trabalho é que enumeramos sistematicamente o conjunto de reduções possíveis, categorizamos e aplicamos na geração de circuitos baseados em multiplexadores a partir de BDDs, medindo os efeitos obtidos no circuito final. As reduções não estão restritas a ROBDDs, mas podem ser aplicadas a qualquer estrutura de circuito baseada em multiplexadores, incluindo árvores de decisão e BDDs não ordenados (também conhecidos como free BDDs). Esta dissertação resultou em uma publicação no ISVLSI 2022 (BRANDãO et al., 2022).

O trabalho fez parte de uma submissão (que incluía outros métodos, que não fazem parte do escopo desta dissertação) ao concurso do IWLS. O principal objetivo deste trabalho era tentar mitigar a relação um para três entre nodos de BDD e nodos de AIG. Isto foi importante no escopo do concurso do IWLS porque a função de custo a ser minimizada era o número de nodos em um AIG. Para o exemplo construído para ter um caso de cada simplificação possível, reduzimos o número de nodos de AIG de 27 para 11. Assim um BDD de 9 nodos resultou em um AIG de 11 nodos, mitigando a relação 1:3 que gerava inicialmente 27 nodos de AIG para 9 nodos de BDD, ou seja: 9 multiplexadores com três nodos cada quando nenhuma redução se aplica.

Sabe-se que BDDs podem ter um tamanho exponencial em função do número de entradas da função, se a função não for adequada a BDDs ou se o ordenamento não for bom. Porém existem funções que se adaptam bem a representação com BDDs, resultando em um pequeno número de nodos. Desta forma, em nossos resultados houve casos em que nosso método perdeu bastante em relação ao ABC, mas aconteceram também alguns poucos casos em que o método proposto deu bons resultados em relação ao ABC, possivelmente porque a função lógica tivesse uma

boa representação em BDDs, mas também porque a mitigação da relação 1:3 entre o número de nodos de BDDs e o número de nodos de AIGs foi realizada com relativo sucesso. Em termos de número total de nodos, para os benchmarks apresentados, a versão sem simplificações resulta em 285348 nodos, enquanto a versão com todas as simplificações resulta em 205683 nodos. Isto resulta em uma redução de 27.9% no numero de nodos. Outra forma de entender este resultado é que a relação de 1 para três foi mitigada para uma relação de 1 para 2.16. Assim acreditamos que as contribuições feitas são válidas e suficientes para o escopo de uma dissertação.

REFERÊNCIAS

AKERS, S. B. Binary decision diagrams. **IEEE Transactions on computers**, IEEE, n. 6, p. 509–516, 1978.

AMARÚ, L.; GAILLARDON, P.-E.; MICHELI, G. D. The EPFL combinational benchmark suite. In: PROCEEDINGS OF THE 24TH INTERNATIONAL WORKSHOP ON LOGIC & SYNTHESIS (IWLS). **Proceedings...** [S.l.: s.n.], 2015.

AMARú, L.; GAILLARDON, P.-E.; MICHELI, G. D. Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization. In: 2014 51ST ACM/EDAC/IEEE DESIGN AUTOMATION CONFERENCE (DAC). **Proceedings...** [S.l.: s.n.], 2014. p. 1–6.

AMARú, L.; GAILLARDON, P.-E.; MICHELI, G. D. Majority-inverter graph: A new paradigm for logic optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 35, n. 5, p. 806–819, 2016.

ASHAR, P.; DEVADAS, S.; KEUTZER, K. Path-delay-fault testability properties of multiplexor-based networks. **Integration**, Elsevier, v. 15, n. 1, p. 1–23, 1993.

BERTACCO, V. et al. Decision diagrams and pass transistor logic synthesis. In: INT'L WORKSHOP ON LOGIC SYNTH. **Proceedings...** [S.l.: s.n.], 1997. v. 168.

BRANDãO, E. D. et al. Possible reductions to generate circuits from bdds. In: 2022 IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI). **Proceedings...** [S.l.: s.n.], 2022. p. 406–409.

BRYANT, R. E. Graph-based algorithms for boolean function manipulation. **Computers, IEEE Transactions on**, IEEE, v. 100, n. 8, p. 677–691, 1986.

BRYANT, R. E.; HEULE, M. J. Dual proof generation for quantified boolean formulas with a bdd-based solver. In: CONFERENCE ON AUTOMATED DEDUCTION (CADE). **Proceedings...** [S.l.: s.n.], 2021.

BUTZEN, P. et al. Design of cmos logic gates with enhanced robustness against aging degradation. **Microelectronics Reliability**, v. 52, n. 9, p. 1822–1826, 2012. ISSN 0026-2714. SPECIAL ISSUE 23rd EUROPEAN SYMPOSIUM ON THE RELIABILITY OF ELECTRON DEVICES, FAILURE PHYSICS AND ANALYSIS. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271412002892>.

BUTZEN, P. F. et al. Transistor network restructuring against nbti degradation. **Microelectronics Reliability**, v. 50, n. 9, p. 1298–1303, 2010. ISSN 0026-2714. 21st European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271410004130>.

BUTZEN, P. F. et al. Standby power consumption estimation by interacting leakage current mechanisms in nanoscaled cmos digital circuits. **Microelectronics Journal**, v. 41, n. 4, p. 247–255, 2010. ISSN 0026-2692. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S002626921000042X>.

CALLEGARO, V. et al. Switchcraft: A framework for transistor network design. In: PROCEEDINGS OF THE 23RD SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2010. (SBCCI '10), p. 49–53. ISBN 9781450301527. Available from Internet: <https://doi.org/10.1145/1854153.1854167>.

CORREIA, V.; REIS, A. Advanced technology mapping for standard-cell generators. In: PROCEEDINGS OF THE 17TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2004. (SBCCI '04), p. 254–259. ISBN 1581139470. Available from Internet: <https://doi.org/10.1145/1016568.1016636>.

da Silva, D. N.; REIS, A. I.; RIBAS, R. P. Cmos logic gate performance variability related to transistor network arrangements. **Microelectronics Reliability**, v. 49, n. 9, p. 977–981, 2009. ISSN 0026-2714. 20th European Symposium on the Reliability of Electron Devices, Failure Physics and Analysis. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0026271409002546>.

DRECHSLER, R.; SHI, J.; FEY, G. Synthesis of fully testable circuits from bdds. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 23, n. 3, p. 440–443, 2004.

GOMES, I. A. C. et al. Methodology for achieving best trade-off of area and fault masking coverage in atmr. In: 2014 15TH LATIN AMERICAN TEST WORKSHOP - LATW. **Proceedings...** [S.l.: s.n.], 2014. p. 1–6.

GOMES, I. A. C. et al. Using only redundant modules with approximate logic to reduce drastically area overhead in tmr. In: 2015 16TH LATIN-AMERICAN TEST SYMPOSIUM (LATS). **Proceedings...** [S.l.: s.n.], 2015. p. 1–6.

GUNTHER, W.; DRECHSLER, R. Minimization of free bdds. In: PROCEEDINGS OF THE ASP-DAC '99 ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE 1999 (CAT. NO.99EX198). **Proceedings...** [S.l.: s.n.], 1999. p. 323–326 vol.1.

ITRS. THE INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS: EXECUTIVE REPORT. In: ITRS 2.0. **Proceedings...** [s.n.], 2015. Available from Internet: <http://www.semiconductors.org>.

JUNIOR, L. S. da R. et al. Fast disjoint transistor networks from bdds. In: PROCEEDINGS OF THE 19TH ANNUAL SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2006. (SBCCI '06), p. 137–142. ISBN 1595934790. Available from Internet: <https://doi.org/10.1145/1150343.1150381>.

KUSHNEROV, A.; MEDINA, M.; YAKOVLEV, A. Towards hazard-free multiplexer based implementation of self-timed circuits. In: IEEE. 2021 27TH IEEE INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS (ASYNC). **Proceedings...** [S.l.], 2021. p. 17–24.

LAI, Y.-T.; PAN, K.-R.; PEDRAM, M. Obdd-based function decomposition: algorithms and implementation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 15, n. 8, p. 977–990, 1996.

LEE, C.-Y. Representation of switching circuits by binary-decision programs. **The Bell System Technical Journal**, Nokia Bell Labs, v. 38, n. 4, p. 985–999, 1959.

MACHADO, L. et al. Kl-cut based digital circuit remapping. In: NORCHIP 2012. **Proceedings...** [S.l.: s.n.], 2012. p. 1–4.

MARRANGHELLO, F. S. et al. Factored forms for memristive material implication stateful logic. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, v. 5, n. 2, p. 267–278, 2015.

MARTINS, M. et al. Open cell library in 15nm freepdk technology. In: PROCEEDINGS OF THE 2015 SYMPOSIUM ON INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2015. (ISPD '15), p. 171–178. ISBN 9781450333993. Available from Internet: <https://doi.org/10.1145/2717764.2717783>.

MARTINS, M. G. A.; RIBAS, R. P.; REIS, A. I. Functional composition: A new paradigm for performing logic synthesis. In: THIRTEENTH INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN (ISQED). **Proceedings...** [S.l.: s.n.], 2012. p. 236–242.

MINATO, S.-i. Fast generation of prime-irredundant covers from binary decision diagrams. **IEICE transactions on fundamentals of electronics, communications and computer sciences**, The Institute of Electronics, Information and Communication Engineers, v. 76, n. 6, p. 967–973, 1993.

MINATO, S.-i.; ISHIURA, N.; YAJIMA, S. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In: IEEE. 27TH ACM/IEEE DESIGN AUTOMATION CONFERENCE. **Proceedings...** [S.l.], 1990. p. 52–57.

MISHCHENKO, A.; CHATTERJEE, S.; BRAYTON, R. Dag-aware aig rewriting a fresh look at combinational logic synthesis. In: PROCEEDINGS OF THE 43RD ANNUAL DESIGN AUTOMATION CONFERENCE. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2006. (DAC '06), p. 532–535. ISBN 1595933816. Available from Internet: <https://doi.org/10.1145/1146909.1147048>.

MOEINZADEH, H. et al. Evolutionary-reduced ordered binary decision diagram. In: 2009 THIRD ASIA INTERNATIONAL CONFERENCE ON MODELLING SIMULATION. **Proceedings...** [S.l.: s.n.], 2009. p. 142–145.

MOORE, G. Cramming More Components onto Integrated Circuits. **Electronics**, 1965.

MOREIRA, M. et al. Semi-custom ncl design with commercial eda frameworks: Is it possible? In: 2014 20TH IEEE INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS. **Proceedings...** [S.l.: s.n.], 2014. p. 53–60.

NEUTZLING, A. et al. A simple and effective heuristic method for threshold logic identification. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 37, n. 5, p. 1023–1036, 2018.

NEUTZLING, A. et al. Synthesis of threshold logic gates to nanoelectronics. In: 2013 26TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI). **Proceedings...** [S.l.: s.n.], 2013. p. 1–6.

NEUTZLING, A. et al. Effective logic synthesis for threshold logic circuit design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 38, n. 5, p. 926–937, 2019.

NEUTZLING, A. et al. Threshold logic synthesis based on cut pruning. In: 2015 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD). **Proceedings...** [S.l.: s.n.], 2015. p. 494–499.

POLI, R. et al. Unified theory to build cell-level transistor networks from bdds [logic synthesis]. In: 16TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2003. SBCCI 2003. PROCEEDINGS.. **Proceedings...** [S.l.: s.n.], 2003. p. 199–204.

POSSANI, V. N. et al. Optimizing transistor networks using a graph-based technique. **Analog Integrated Circuits and Signal Processing**, Springer, v. 73, p. 841–850, 2012.

REIS, A. Covering strategies for library free technology mapping. In: PROCEEDINGS. XII SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (CAT. NO.PR00387). **Proceedings...** [S.l.: s.n.], 1999. p. 180–183.

REIS, A. I.; DRECHSLER, R. **Advanced logic synthesis**. Springer, 2018. ISBN 978-3-319-88407-3. Available from Internet: <https://doi.org/10.1007/978-3-319-67295-3>.

REIS, A. I. et al. Library free technology mapping. In: _____. VLSI: INTEGRATED SYSTEMS ON SILICON: IFIP TC10 WG10.5 INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION 26–30 AUGUST 1997, GRAMADO, RS, BRAZIL. **Proceedings...** Boston, MA: Springer US, 1997. p. 303–314. ISBN 978-0-387-35311-1. Available from Internet: <https://doi.org/10.1007/978-0-387-35311-1_25>.

ROSA, L. S. et al. Scheduling policy costs on a java microcontroller. In: MEERSMAN, R.; TARI, Z. (Ed.). ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS 2003: OTM 2003 WORKSHOPS. **Proceedings...** Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 520–533. ISBN 978-3-540-39962-9.

ROSA, L. S. da et al. A comparative study of cmos gates with minimum transistor stacks. In: PROCEEDINGS OF THE 20TH ANNUAL CONFERENCE ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN. **Proceedings...** New York, NY, USA: Association for Computing Machinery, 2007. (SBCCI '07), p. 93–98. ISBN 9781595938169. Available from Internet: <https://doi.org/10.1145/1284480.1284511>.

ROSA, L. S. da et al. Switch level optimization of digital cmos gate networks. In: 2009 10TH INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN. **Proceedings...** [S.l.: s.n.], 2009. p. 324–329.

SEN, B. et al. An efficient multiplexer in quantum-dot cellular automata. In: RAHAMAN, H.; CHATTOPADHYAY, S.; CHATTOPADHYAY, S. (Ed.). PROGRESS IN VLSI DESIGN AND TEST. **Proceedings...** Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 350–351. ISBN 978-3-642-31494-0.

SHANNON, C. E. The synthesis of two-terminal switching circuits. **The Bell System Technical Journal**, Nokia Bell Labs, v. 28, n. 1, p. 59–98, 1949.

SOMENZI, F. CUDD: Colorado University decision diagram package release 2.7.0. **University of Colorado at Boulder**, 2018.

TOGNI, J. et al. Automatic generation of digital cell libraries. In: PROCEEDINGS. 15TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN. **Proceedings...** [S.l.: s.n.], 2002. p. 265–270.

WAGNER, F. R.; REIS, A. I.; RIBAS, R. P. **Fundamentos de circuitos digitais**. [S.l.]: Sagra Luzzatto, Porto Alegre, 2006.

WILLE, R.; HILLMICH, S.; BURGHOLZER, L. **Tools for Quantum Computing Based on Decision Diagrams**. 2021.

YANG, C.; CIESIELSKI, M. Synthesis for mixed cmos/ptl logic. In: PROCEEDINGS DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION 2000 (CAT. NO. PR00537). **Proceedings...** [S.l.: s.n.], 2000. p. 750–.