GERALDO SOARES FONTES JUNIOR

EXPLORANDO O ESPAÇO DE SOLUÇÕES NO POSICIONAMENTO E ROTEAMENTO DE CÉLULAS QCA NO ESQUEMA DE CLOCK USE

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS – BRASIL
2018

Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa

T
F683e
2018

Fontes Junior, Geraldo Soares, 1988-
Explorando o espaço de soluções no posicionamento e roteamento de células QCA no esquema de clock USE / Geraldo Soares Fontes Junior. – Viçosa, MG, 2018.
xi, 73f. : il. (algumas color.) ; 29 cm.

Inclui apêndice.
Orientador: Ricardo dos Santos Ferreira.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.71-73.

1. Circuitos lógicos. 2. Nanotecnologia. 3. Nanoeletrônicos. 4. Autômato celular. 5. Pontos quânticos. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CCD 22. ed. 621.395

GERALDO SOARES FONTES JUNIOR

EXPLORANDO O ESPAÇO DE SOLUÇÕES NO POSICIONAMENTO E ROTEAMENTO DE CÉLULAS QCA NO ESQUEMA DE CLOCK USE

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 17 de janeiro de 2018.

Jose Augusto Miranda Nacif

Omar Paranaiba Vilela Neto

Ricardo dos Santos Ferreira
(Orientador)

Dedico este trabalho a minha família e a minha namorada Michele.

Agradecimentos

Aos meus pais, pelo apoio e incentivo durante essa jornada.

Ao meu orientador Ricardo dos Santos Ferreira, pela orientação, pelo apoio e pela grande ajuda e conselhos que possibilitaram a realização desse trabalho.

Ao parceiros Pedro Arthur Silva, Juliana Resende e José Augusto Nacif por toda ajuda fornecida.

A Divisão de Apoio ao Desenvolvimento Científico e Tecnológico da Universidade Federal de Viçosa.

Ao departamento de Informática da UFV e seus professores.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo financiamento fornecido para realização desse trabalho.

A Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG) e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Em especial a minha namorada Michele pelo companheirismo em todos momentos dando todo apoio, carinho e amor.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

FONTES JUNIOR, Geraldo Soares, M.Sc., Universidade Federal de Viçosa, janeiro de 2018. **Explorando o Espaço de Soluções no Posicionamento e Roteamento de Células QCA no Esquema de Clock USE.** Orientador: Ricardo dos Santos Ferreira.

A nanotecnologia de Autômatos Celulares com Pontos Quânticos (QCA) é uma promissora alternativa a atual tecnologia baseada em silício para fabricação de circuitos integrados. Diferente das tecnologias baseadas em silício, QCA não utiliza o fluxo de correntes elétricas na codificação de informações, e sim interação Coulombiana, resultando em baixo consumo energético. O projeto de circuitos QCA introduz novos desafios aos tradicionais métodos usados no mapeamento, posicionamento e roteamento de circuitos no nível de portas. O uso de um esquema de clock é recomendável para garantir a escalabilidade e regularidade nas soluções, pois todos caminhos internos em um circuito QCA devem estar balanceados. Este trabalho propõe uma abordagem inovadora para o posicionamento e roteamento de circuitos QCA de forma semi-automática, baseando-se em sobreposição de layout criados sobre o esquema de clock USE. É apresentada uma forma eficiente de particionamento criada sobre o paradigma de divisão e conquista a fim de reduzir o esforço na obtenção de soluções. As partições são criadas através de análises topológicas sobre o grafo onde é estudado o impacto de caminhos reconvergentes e interações de redes complexas. O uso de portas de maioria na construção de circuitos mostra consideráveis benefícios ao projeto de circuitos QCA, reduzindo sua complexidade e melhorando a qualidade das soluções encontradas. Os resultados obtidos nesse trabalho apresentam redução na área total das soluções de mais de 50% em comparação com abordagens anteriores. Por fim, as soluções geradas são validadas na ferramenta de simulação QCADesigner.

Abstract

FONTES JUNIOR, Geraldo Soares, M.Sc., Universidade Federal de Viçosa, January, 2018. **Exploring the Solution Space in Placement and Routing of QCA Cells in Clock Scheme USE.** Adviser: Ricardo dos Santos Ferreira.

The nanotechnology of Quantum Dot Cellular Automata (QCA) is promising alternative to current silicon-based technology used in the manufacture of integrated circuits. Unlike silicon-based technologies, QCA does not use the flow of electric currents in information coding, this is accomplished through electromagnetic interaction, allowing to reach high frequencies with low energy consumption. The QCA circuit design introduces new challenges to the traditional methods used in the mapping, placement and routing of circuits at the gate-level. The use of a clock scheme is recommended to guarantee the scalability and regularity of the solutions, since all internal paths in a QCA circuit must be balanced. This work proposes a novel approach to automatically map a gate-level circuit onto a QCA layout by using a merge overlapping approach, and a universal clock scheme to provide scalability. An efficient form of partitioning created on the division and conquest paradigm is presented in order to reduce the effort to obtain solutions. Partitions are created through topological analyzes of the graph where the impact of reconverting paths and complex network interactions is analyzed. The use of majority gates in circuit construction provides considerable benefits to the design of QCA circuits, reducing their complexity and improving the quality of the solutions. The results obtained in this work present a reduction in the total area of the solutions of more than 50% compared to previous approaches. Finally, the generated solutions are validated in the QCADesigner simulation tool.

Capítulo 1

Introdução

1.1 Motivação

A Lei de Moore [Moore, 1998] se sustentou por várias décadas mas vem se aproximando do limite físico das tecnologias baseadas em silício. Segundo a lei de Moore, o número de transistores em circuitos integrados cresce exponencialmente em função do tempo [Moore, 1998], dobrando a cada 18 meses. Ao atingir escalas cada vez menores, surgem grandes desafios como desempenho energético, escalabilidade, frequência de relógio, confiabilidade entre outros. Para suprir tais necessidades em escala nanométrica novas tecnologias vem sendo propostas.

Autômatos celulares com pontos quânticos (*Quantum-dots Cellular Automata* - QCA) são uma alternativa promissora na escala nanométrica. Diferente das tecnologias baseadas em silício como o MOSFET (*Metal Oxide Semiconductor Field Effect Transistor*), que usa o fluxo de corrente elétrica para codificar informações, o QCA usa a polarização das células para codificá-las podendo assim alcançar altas frequências de relógio com um baixo consumo de energia [Kim et al., 2006]. Embora tenha sido proposta há duas décadas [Lent & Tougaw, 1997], ainda faltam muitas ferramentas para se realizar todo o processo de projeto, incluindo a síntese lógica, o mapeamento tecnológico, o posicionamento e o roteamento (*Placement and Routing* - P&R).

Uma das principais diferenças entre a síntese de circuitos para silício, mais especificamente para CMOS, da síntese para QCA encontra-se no sincronismo na propagação dos sinais. Em QCA mesmo para circuitos combinacionais o uso de alguma estrutura de sincronização é necessário. Em circuitos QCA, todos sinais de entrada de uma porta devem chegar ao mesmo tempo e para tal, todos fios devem estar balanceados dentro de algum esquema de relógio, ou clock. Nesta dissertação

usaremos o termo clock. Diferente da síntese QCA, em silício ou CMOS não é necessário clock para circuitos combinacionais e para circuitos sequenciais, pode-se definir o clock como o maior tempo para o qual todos sinais de entrada se propaguem até as saídas do circuito. A maioria das abordagens anteriores para QCA faziam uso de síntese manual ou uso de zonas de clock irregulares o que não se adequá tão bem para um processo de mapeamento tecnológico automático. Uma proposta recentemente apresentada em [Reis et al., 2016] e [Trindade et al., 2016] fazem uso de um esquema de zonas de clock chamado USE (*Universal, scalable and efficient clock scheme*) [Campos et al., 2016] para realizar síntese automática em nível de portas. Em [Trindade et al., 2016] a síntese limita-se a circuitos pequenos, cerca de 10 portas, e com número de entradas limitada a no máximo duas por porta lógica AOI(*And, Or e Inverter*). Esta restrição limita o projeto de circuitos para QCA, onde o uso de portas de maioria com três entradas reduz significativamente o tamanho dos layouts gerados. Em [Reis et al., 2016], todos circuitos tem seu P&R feitos manualmente o que faz com que os fios fiquem muito grandes em virtude do seu balanceamento.

1.2 Objetivos e Metas

Esse trabalho tem como objetivo propor uma heurística capaz de realizar posicionamento e roteamento (P&R) de circuitos QCA de forma automática reduzindo a área total da soluções encontradas, oferecendo suporte ao uso de portas de maioria. Para validar a proposta, todas as soluções serão verificadas no simulador QCADesigner [Walus et al., 2004]. Entre as estratégias adotadas nesse trabalho, o objetivo será priorizar novas abordagens visando reduzir o espaço de busca na solução do problema de posicionamento e roteamento. Além disso, visando compreender o espaço de busca, um objetivo específico deste trabalho será buscar características topológicas presente nos grafos dos circuitos digitais e analisar o seu impacto sobre os algoritmos de P&R.

1.3 Organização da dissertação

No Capítulo 2, são apresentados os conceitos básicos sobre os Autômatos celulares com pontos quânticos (*Quantum-dots Cellular Automata* - QCA). Além disso, iremos ilustrar como construir circuitos com fios, cruzamento de fios e portas lógicas básicas. No Capítulo 2, também introduzimos os conceitos de P&R, os

esquema de sincronismo para QCA e mais especificamente, a estrutura de zonas de clock do USE [Campos et al., 2016] que é a camada alvo das técnicas apresentadas neste trabalho. Por fim é mostrado o simulador QCADesigner no qual o P&R de um circuito QCA pode ser validado.

O Capítulo 3 aborda os trabalhos relacionados que tratam o problema de geração de layouts e P&R para circuitos QCA. Muitas abordagens para geração de layout são manuais e irregulares, o que dificulta a comparação com nosso trabalho. Para as abordagens automatizadas, que tratam circuitos com mais de 100 portas lógicas, não é possível comparar pois estas não geram o layout final.

O Capítulo 4 mostra o fluxo proposto para o processo de P&R passando pela estrutura de entrada, as estrutura de dados usadas para representar um circuito QCA e o USE, as metodologias de partição e sobreposição usadas e o detalhamento dos algoritmos propostos. Nesse capítulo também é feito um estudo sobre características topológicas dos grafos de circuitos e como tais características podem ser usadas para auxiliar todo processo. Ademais, mostramos que é possível usar portas de maioria com 5 entradas para implementar um subconjunto de funções e minimizar o layout final.

O Capítulo 5 apresenta os resultados obtidos pela heurística e a validação no simulador. O Capítulo 6 apresenta a conclusão e os trabalhos futuros.

Capítulo 2

Autômatos Celulares com Pontos Quânticos (QCA)

2.1 Célula QCA

O paradigma QCA é construído sobre o conceito de célula QCA a qual contém um número definido de pontos quânticos, também referidos como dots, sendo esses basicamente regiões no espaço onde pode existir ou não uma carga elétrica. Uma célula QCA é constituída de quatro dots posicionados nas extremidades da célula, a qual possui uma barreira de potencial grande o suficiente para evitar que as cargas elétricas escapem do seu interior. Os dots são interligados entre si, como ilustrado na Figura 2.1, permitindo assim que em certas condições dois elétron possam transitar livremente entre os dots.

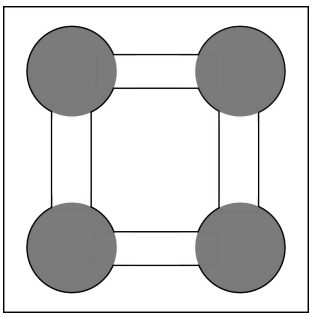


Figura 2.1: Tunelamento entre quatro pontos quânticos em uma célula QCA.

Considerando uma célula como um sistema isolado, as forças de repulsão geradas pelas interações coulombianas entre os elétrons tende a posicioná-los em extremidades opostas diagonalmente, sendo assim possível uma célula assumir dois estados bem definidos e distintos, os quais são referidos como polarização da célula (P) em

[Tougaw & Lent, 1994]. Na Figura 2.2 os dois possíveis estados de polarização, P = +1 ou P = -1, são apresentados e através deles tem-se a capacidade de se codificar a informação binária como mostrado em [Orlov et al., 1997]. Sendo possível controlar o trânsito dos elétrons entre os dots, ora permitindo ora bloqueando, ou seja garantir ou não o tunelamento destes, então pode-se manter a polarização de uma célula e por indução polarizar uma segunda célula a partir da primeira posicionando-as lado a lado. A polarização da segunda célula será a mesma da primeira célula, uma vez que a interação entre os elétrons confinados nos dots tendem a repelir os elétrons da segunda célula, transmitindo assim sua polarização.

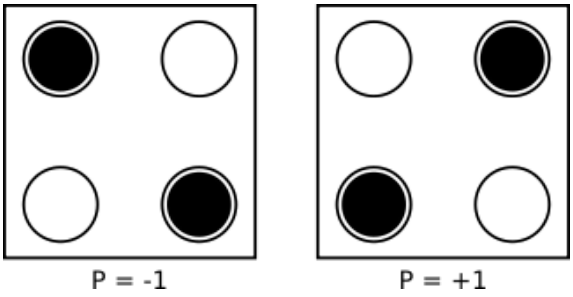


Figura 2.2: Dois possíveis estados de polarização em uma célula QCA.

Na Figura 2.3, o processo de polarização de uma célula QCA *C2* induzida pela polarização de uma segunda célula *C1* é representado mostrando seus elétrons como sendo círculos pretos posicionados nos dots de suas respectivas células. *C1* encontra-se em um estado que bloqueia o tunelamento dos elétrons sendo esses confinados aos dots *d1* e *d2*, já *C2*, representada na cor branco, permite o tunelamento dos elétrons. Ao serem posicionadas lado a lado as forças de repulsão mais significativas sobre o elétron de *C2* são mostradas na forma de vetor de força, essa é uma simplificação do sistema e não representa todas interações que realmente existem sobre o mesmo, mas ilustra o comportamento esperado que os elétrons em *C2* adotam a fim de minimizar a energia desse sistema.

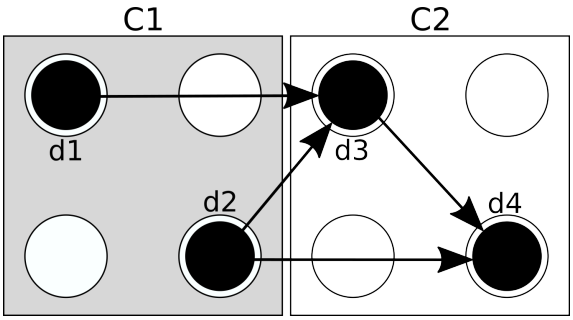


Figura 2.3: célula QCA induzindo a polarização de uma segunda célula.

2.2 Elementos Básicos

Nesta seção iremos apresentar os elementos básicos para construção de circuitos QCA: fios, bifurcação (ou fanout), cruzamento de fios e portas lógicas.

2.2.1 Fio

Em um circuito, um fio tem a função de transmitir informações podendo ser abstraído como sendo um canal de comunicação entre dois elementos. Baseando-se na ideia de polarização por indução, um fio de células QCA pode ser criado posicionando varias células lado a lado como visto na Figura 2.4, sendo o tamanho do fio limitado a fim de manter sua consistência. Ao se polarizar uma ponta do fio e impedir que sua polarização seja alterada, esta polarização será transmitida por todas células no fio até alcançar a célula na extremidade oposta fazendo assim com que todas células tenham o mesmo estado de polarização. Dessa forma um fio QCA é capaz de transmitir informação usando a força de interação Coulombiana entre suas cargas, não havendo assim necessidade de fluxo de corrente elétrica.

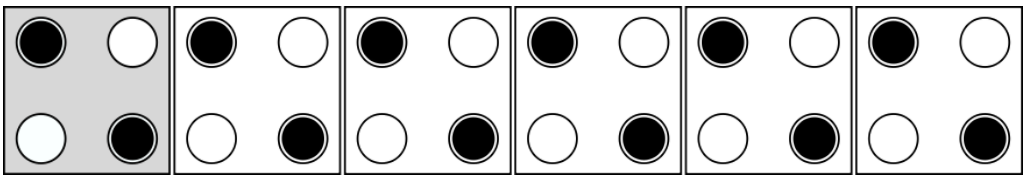


Figura 2.4: Fio construído com células QCA.

Para propagar um sinal para vários destinos é necessário criar bifurcações ou também referenciados como fanout. Mesmo existindo um fanout em um fio como representado na Figura 2.5, sua polarização se mantém estável nas duas direções de propagação. Pode parecer que a interação entre as células C3 e C4 pode levar a inversão na polarização do sinal de C4, entretanto a resultante das forças que atuam sobre seus elétrons fará a polarização manter o menor estado de energia no sistema.

2.2.2 Inversor

Ao se posicionar duas células QCA diagonalmente o estado de polarização propagado tende a ser invertido e dessa forma é possível a construção de inversores como o da Figura 2.6. Esse modelo permite inverter sinal integrando o inversor a um fio e dessa forma reduzir o número de portas lógicas de um circuito. Entretanto, neste modelo de inversor tem-se o risco maior de interferência caso haja fios posicionados próximos.

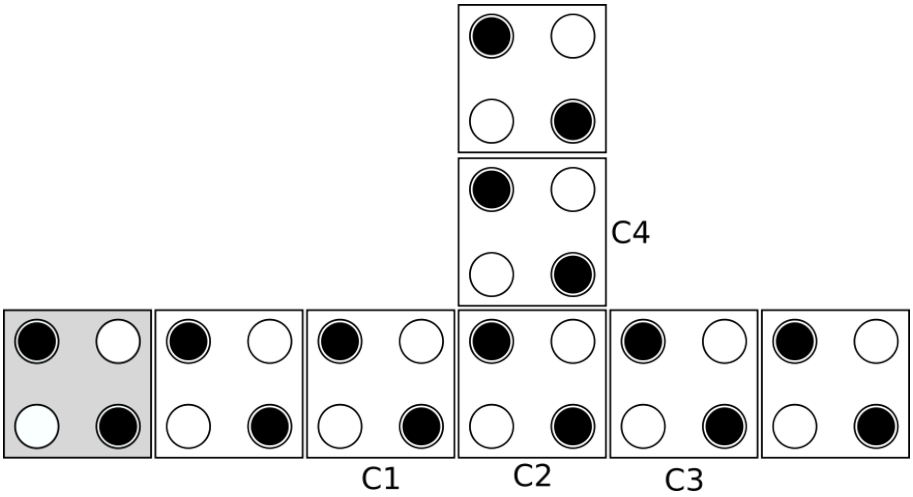


Figura 2.5: Fanout em um fio construído com células QCA.

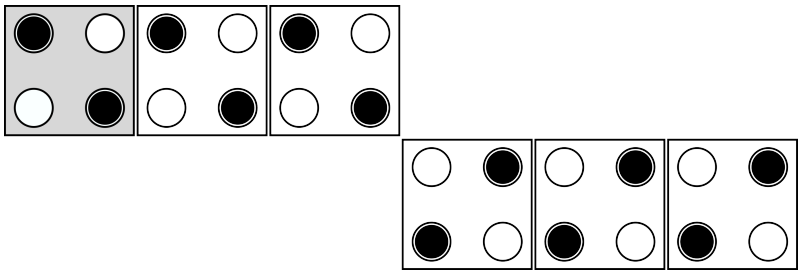


Figura 2.6: Inversão da Polarização de células posicionadas ortogonalmente.

Outra alternativa mostrada na Figura 2.7 foi sugerida em [Tougaw & Lent, 1994] e mostra um inversor sendo construído utilizando redundância no sinal de polarização a fim de garantir maior integridade, isso reduz os riscos de ruídos externos ao fio alterarem seu estado de polarização.

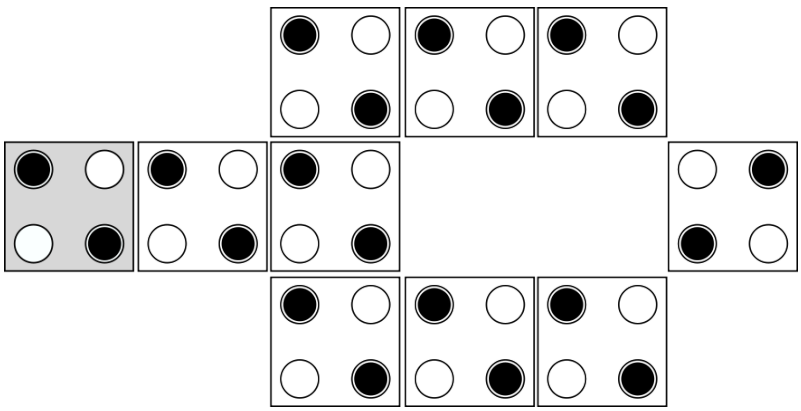
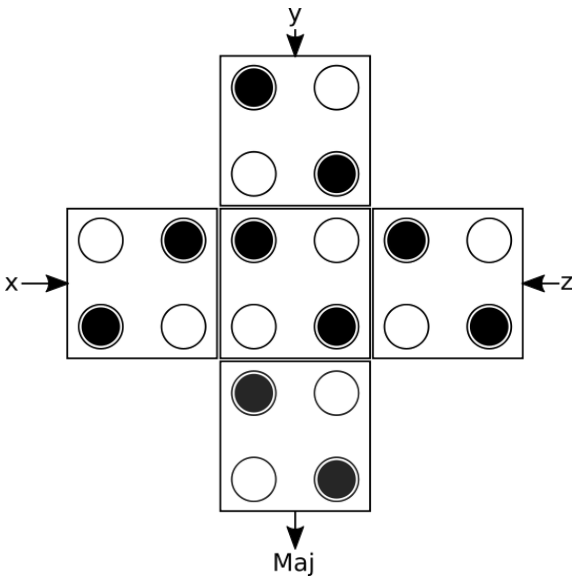


Figura 2.7: Inversor proposto em [Tougaw & Lent, 1994] construído com redundância de sinal.

2.2.3 Porta de Maioria(Majority)

Uma porta de maioria é o elemento lógico básico usado para construir portas lógicas em QCA, seu nome é devido a maneira como se comporta sendo seu sinal de saída igual ao da maioria das entradas. Um layout para construção da porta de maioria de três entradas usando células QCA foi proposto em [Tougaw & Lent, 1994] como mostrado pela Figura 2.8a, nele uma porta de maioria com entradas x, y e z e saída Maj está representada.



(a) Projeto em cruz para uma porta de maioria.

	x	y	z	Maj
E	0	0	0	0
	0	1	0	0
	1	0	0	0
	1	1	0	1
OU	0	0	1	0
	0	1	1	1
	1	0	1	1
	1	1	1	1

(b) Tabela Verdade fixando a entrada z.

Figura 2.8: Porta de maioria e tabela verdade fixando uma entrada

Como mencionado, nesta dissertação iremos utilizar o esquema de clock USE [Campos et al., 2016], sendo que o layout apresentado na figura 2.8a não é compatível com as restrições do esquema de clock adotado. Isso ocorre, pois no USE não é permitido a propagação de sinais na mesma direção provenientes de fios de três lados distintos de uma célula. Assim outro layout foi proposto em [Reis et al., 2016] para a construção da porta de maioria de três entradas. Esta implementação permite que duas entradas possam ser posicionadas de um mesmo lado da porta como mostrado na Figura 2.9. Variações na posição de entrada podem ser feitas como por exemplo alterar o sentido da entrada y para que essa seja feita pela direita da célula e não por baixo como é mostrado.

A porta maioria possui grande versatilidade, fixando-se a polarização de uma de suas células de entrada a permite assumir o comportamento de portas lógicas "E"e "OU". Fixando a polarização de uma célula para que represente o valor binário 1

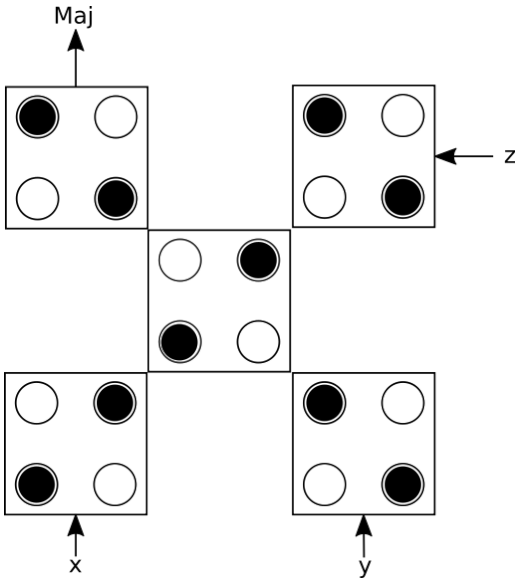


Figura 2.9: Porta de maioria usada com o USE.

a porta de maioria assume o mesmo comportamento de uma porta lógica "OU"e quando fixado em 0 obtém-se o comportamento da porta "E". A Figura 2.8a mostra a tabela verdade ao se fixar a entrada z e seus valores fixados estão evidenciados em vermelho.

2.2.4 Cruzamento de Fios

O cruzamento de fios facilita o processo de P&R de um circuito. Uma solução é adotar a estratégia de cruzamento coplanar como visto em [Patitz, 2006] [Bhanja et al., 2006], nesse tipo de cruzamento os dots de um fio são rotacionados em 45°, como mostrado na Figura 2.10, o que permite que ambos sinais dos dois fios possam ser propagados sem que haja a interferência de um sobre o outro, entretanto essa configuração acarreta na perda de desempenho em todo o circuito [Graunke et al., 2005].

Outra estratégia para tratar o cruzamento vista em [Shin et al., 2013] resolve o problema através do acréscimo de multicamadas se sobrepondo, dessa maneira um cruzamento é feito elevando-se um fio através do empilhamento de células de forma a criar um distanciamento vertical entre eles, como mostrado na Figura 2.11. Um dos possíveis problemas dessa estratégia pode ser ocasionado pela interferência que o sinal em um fio pode causar no sinal do outro. Entretanto esse problema pode ser resolvido aumentando o espaçamento entre as camadas. Nesse trabalho foi adotada esta forma para representar o cruzamento de fios por ser menos restrita, mantendo a uniformidade das células QCA.

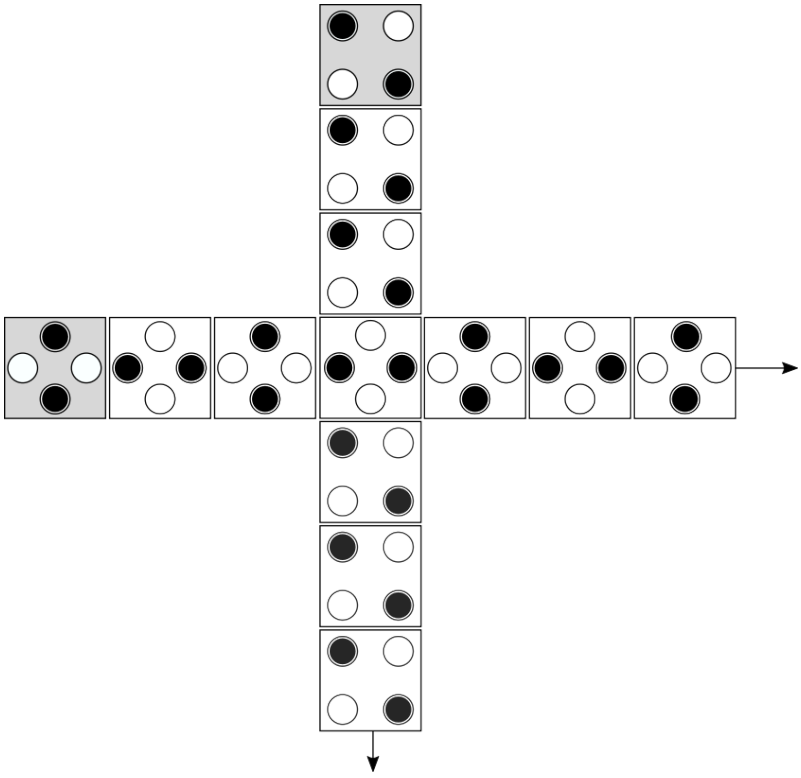


Figura 2.10: Cruzamento coplanar.

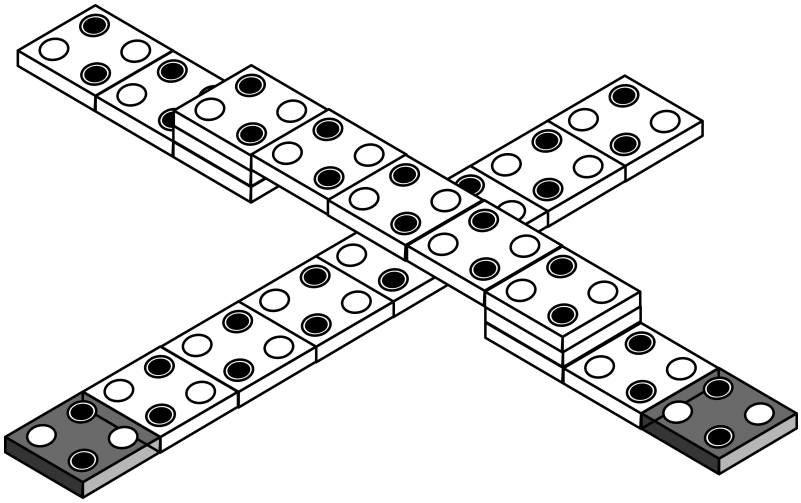


Figura 2.11: Cruzamento em camadas.

2.3 Zonas de Clock

Em circuitos QCA, o clock pode controlar a propagação dos sinais e pode ser gerado por um campo elétrico criado para controlar barreiras de tunelamento e assim permitir ou não a polarização de uma célula. O clock permite sincronizar células QCA e evitar a propagação de algum sinal em uma célula antes que todos os outros

sinais de entrada tenham de fato a alcançado.

Como definido em [Lent & Tougaw, 1997], o sinal de clock em QCA pode ser criado a partir de quatro fases referidas como *Switch*, *Hold*, *Release* e *Relax*. Na fase *Switch*, o potencial das barreiras de tunelamento são gradualmente aumentadas permitindo assim que a célula se polarize em decorrência da polarização das suas entradas. No fim dessa fase a polarização da célula se estabiliza não podendo ser mais alterada levando assim a fase de *Hold* onde as barreiras são mantidas elevadas e a polarização da célula pode ser usada como entrada para a próxima zona de clock. Na fase de *Release*, as barreiras começam a serem abaixadas e a célula inicia o processo de despolarização e por fim na fase de *Relax*, as barreiras são mantidas baixas e a célula assume o estado de despolarização.

Um grupo de células QCA em uma mesma região podem ser agrupadas sob a mesma fase de clock formando assim uma zona de clock onde independente do número de células nessa zona, o atraso será sempre o mesmo [Walus et al., 2003].

2.4 Esquema de Clock USE

Vários problemas surgem ao se definir um esquema de clock para QCA, fios muito longos, falta de uniformidade nas zonas de clock e espaço subutilizado estão entre os apontados em [Niemier, 2000]. Uma proposta feita em [Vankamamidi et al., 2008] apresenta uma alternativa para alguns desses problemas porém continua apresentando muito espaço inutilizado e não suporta circuitos com realimentação de sinal.

Proposto em [Campos et al., 2016], o esquema de clock USE (*Universal, scalable and efficient*) atende aos problemas para os quais a proposta feita em [Vankamamidi et al., 2008] não consegue tratar. Entre as principais características do USE estão a escalabilidade, a eficiência e a possibilidade de realizar P&R para qualquer tipo de circuito sendo assim universal.

O USE é composto por zonas de clock bem delimitadas, possibilitando assim a criação de soluções uniformes e regulares. Uma matriz de zonas de clock organizadas de forma a definir um fluxo de informações regulado pelas fases de clock da forma ao esquema de clock USE. Sendo uma matriz de zonas de clock bem definidas, é possível aumentar seu tamanho o quanto for necessário, garantindo assim a escalabilidade capaz de criar circuitos de qualquer tamanho.

Uma zona de clock USE é criada como um conjunto de célula QCA agrupadas sob a mesma fase de clock. Dessa maneira todas células QCA em uma fase de clock

USE possuem o mesmo valor de polarização. Nessa trabalha é adotado o tamanho de 5x5 células QCA para uma zona de clock USE. Esse tamanho pode crescer de maneira limitada, uma vez que zonas de clock USE com muitas células tendem a apresentar problemas de consistência, podendo ser ocasionados pela corrente de fuga.

Na Figura 2.12 o fluxo é mostrado. As zonas de clock são classificadas usando números de 1 a 4 referenciando as quatro fases de clock no QCA. Cada número corresponde a uma fase de clock QCA. Nesse exemplo a fase de clock *Hold* está associada as zonas de clock 1, a fase de clock *Switch* está associada as zonas de clock 2, a fase de clock *Release* está associada as zonas de clock 3 e a fase de clock *Relax* está associada as zonas de clock 4. Dessa forma, o fluxo da informação se dá em ordem crescente indo da zona de menor valor para a zona de maior valor, voltando a zona 1 quando atinge a zona de valor 4. Dessa forma pode haver fluxo da zona de clock 3 para a zona 4 e da zona 4 para a zona 1, mas não pode existir fluxo da zona 1 para a zona 4 ou da zona 4 para a zona 3. Isso decorre da definição das fases de clock QCA, as quais são base para a criação do esquema de clock USE.

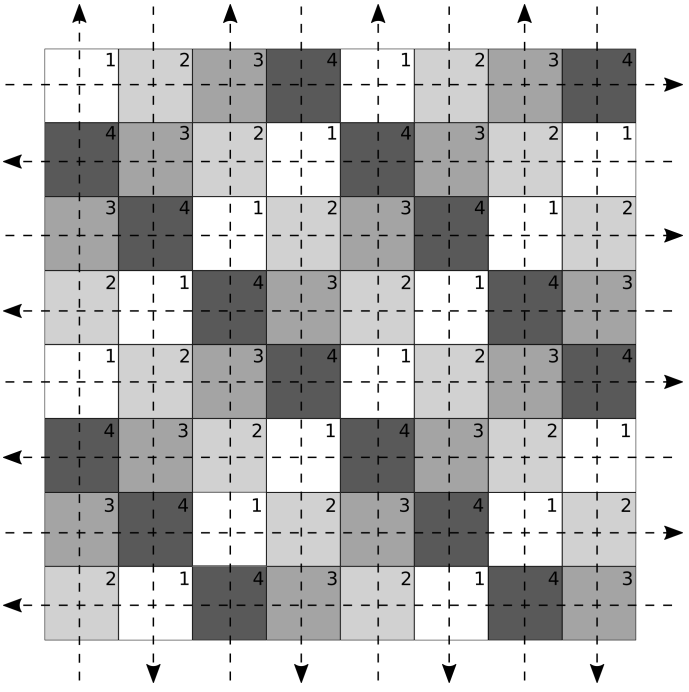


Figura 2.12: Esquema de clock USE.

Com a definição regular das zonas de clock no esquema USE, é possível criar circuitos de qualquer tamanho. A regularidade do USE permite ainda a criação de circuitos QCA com grande regularidade, uma vez que o fluxo de informações pode se dar em todas as direções. Essa característica permite com que um fluxo

de realimentação de sinal possa ser criado para qualquer circuito, independente do tamanho do caminho do fluxo.

Como mostrado em [Campos et al., 2016], o esquema de clock USE permite implementar cruzamentos de fios, tanto coplanar quanto cruzamento em camadas. Ainda em [Campos et al., 2016] é mostrado que as zonas de clock podem ser criadas facilmente por um circuito com baixa complexidade baseado na proposta apresentada em [Vankamamidi et al., 2008].

2.5 Posicionamento e Roteamento

O legado de ferramentas e processos das tecnologias atuais é uma das barreiras que impendem que novas tecnologias sejam adotadas. É preciso incorporar a atual tecnologia de fabricação de circuitos aos novos paradigmas que estão sendo propostos. Em [Henderson et al., 2004], uma abordagem na construção de circuitos QCA foi proposta baseando-se na metodologia usada atualmente na fabricação de circuitos CMOS.

O P&R de um circuito é feito ao se definir posições para cada porta do circuito e locais para os fios que as interligam de forma a sincronizar os sinais em todo circuito. Uma das principais diferenças entre as tecnologias CMOS e QCA encontra-se no modo como o sincronismo é feito. Um sinal de clock entre outras coisas visa garantir que nenhum sinal se propague de uma porta antes que essa esteja estabilizada, evitando a propagação de informações antes que todos sinais de entradas tenham de fato sido processados.

2.6 Simuladores

Devido a dificuldade de implementação física de circuitos QCA no estágio atual, torna-se indispensável o uso de simuladores para validar resultados desenvolvidos para tais circuitos. Mesmo em outras tecnologias consolidadas, os simuladores garantem uma validação primária antes que de fato se tenha uma implementação física, evitando possivelmente assim o mal uso de recursos acarretado por uma falha possível de ser identificada por simulação.

QCADesigner é um simulador de circuitos QCA de distribuição livre [Walus et al., 2004], desenvolvido na universidade de Calgary no Canadá em parceria com a universidade de Notre Dame. Além das diversas ferramentas para construção de circuitos QCA, o QCADesigner permite a implementação do layout de zonas de

clock USE [Campos et al., 2016] e apesar de sua interface de construção de soluções ser inteiramente gráfica, é possível transformar uma solução gerada para o formato de arquivo aceito por ele. Uma das dificuldades apresentadas por esse simulador está na forma como a validação de circuitos deve ser feita, a saída do QCADesigner é apresentada graficamente no formato de ondas o que dificulta a validação automática das soluções, fazendo com que essa tenha de ser feita manualmente. Nesse trabalho será adotado o QCADesigner como simulador para validar soluções.

Capítulo 3

Trabalhos Relacionados

3.1 Introdução

O Paradigma QCA se tornou um candidato a substituir a atual tecnologia usada na fabricação de circuitos CMOS. Entretanto, espera-se que haja o reaproveitamento em parte do processo já consolidado na fabricação de circuitos. Um dos principais desafios relacionados a criação de circuitos QCA está no problema de posicionamento e roteamento(P&R), o qual se difere significativamente do processo de P&R em circuitos CMOS.

3.2 Posicionamento e Roteamento em CMOS

O problema de posicionamento e roteamento em circuito CMOS e em FPGA é NP-completo [Donath, 1980]. A abordagem tradicional executa o posicionamento de forma independente e separada do roteamento. Primeiro é realizado o posicionamento com algum modelo de custo. Posteriormente, o roteamento é feito. Caso não seja possível rotear, um novo posicionamento deve ser gerado. O posicionamento pode ser classificado em três categorias: Simulated-annealing, particionamento e analítico. A maioria dos algoritmos de posicionamentos são baseados em simulated annealing(SA) [Luu et al., 2011]. Entretanto, o tempo de execução do SA pode ser elevado, logo várias otimizações no SA para P&R foram propostas. A segunda categoria particiona o circuito em circuitos menores e aplica SA nas partições [Maidee et al., 2005]. Apesar da redução no tempo de execução, esta abordagem em geral piora a qualidade da solução por ter apenas uma visão local nas partições. A terceira categoria é baseada em modelos analíticos [Lin et al., 2013], esta abordagem

também realiza partições mas com uma estimativa mais em alto nível usando modelos analíticos, para em uma segunda etapa realizar o posicionamento detalhado da partição onde em geral SA é aplicado localmente.

A abordagem SA é uma meta-heurística baseada no processo de resfriamento. No início do processo pode-se fazer movimentos maiores, e a medida que o tempo progride, o resfriamento vai restringindo a movimentos menores. No problema de posicionamento, a técnica SA consiste em trocar as posições de porta lógicas (CMOS) ou LUT (FPGA), fazendo uma permutação entre as portas ou LUTs. O custo do posicionamento é avaliado após a troca. No início do processo se aceita até trocas com pouco ganho ou mesmo com perda para tentar sair de mínimos locais. Depois, com o decorrer do tempo apenas trocas com ganho são aceitas. Para um circuito com centenas de portas, milhões de trocas são efetuadas durante a execução do posicionamento SA. Considerando a tecnologia alvo circuitos QCA, as abordagens de SA que são baseadas em uma estratégia local ficam bem restritas, pois uma troca pode afetar um caminho que afeta todo o circuito, sendo necessário reposicionar várias portas para voltar a ter um solução válida. Portanto, novas abordagens foram propostas para os circuitos QCA.

3.3 Posicionamento e Roteamento em QCA

Abordagens anteriores para o P&R de QCA podem ser classificados em nível de portas como em [Ravichandran et al., 2004] e [Bubna et al., 2008] ou em nível de layout como mostrado em [Teodósio & Sousa, 2007], [Trindade et al., 2016] e [Reis et al., 2016]. A abordagem baseada em níveis percorre o grafo do circuito em largura por níveis, onde todos nós para um dado nível l são posicionados sob uma mesma zona de clock. Todos nós no nível logo abaixo, *l+1*, devem ser posicionados mantendo a menor distância possível para o nível anterior e minimizando o cruzamento de fios.

3.3.1 Nível de Porta

A abordagem baseada em níveis de portas proposta em [Ravichandran et al., 2004] e [Bubna et al., 2008] insere nós identidade (que apenas deixam repassar o sinal de entrada) para balancear os caminhos e replica alguns nós para buscar reduzir os cruzamentos de fios entre dois níveis. A Figura 3.1a mostra um grafo com três níveis l, *l+1* e *l+2* contendo 6 fios e 1 cruzamento entre os níveis l e *l+1*, 7 fios e 2 cruzamentos entre os níveis *l+1* e *l+2*. A Figura3.1b mostra o resultado

da técnica ao ser aplicada sobre o grafo da Figura 3.1a entre os níveis *l* e *l+1*. Um nó identidade (cor preta) foi inserido para balancear o caminho entre os nós *c* e *h*. Os nós *d* e *f* mostrados em cinza, foram duplicados e permutados de lugar a fim de reduzir os cruzamentos de fios entre os níveis *l* e *l+1*. No grafo resultante da Figura3.1b, o número de fios entre *l* e *l+1* se mantêm o mesmo mas o número de cruzamentos de fios se torna zero. Entretanto essa estratégia não resolve o problema para cruzamentos de fios e interconexões mais complexas, pois simplesmente propagam os cruzamentos para o próximo nível como ilustrado na Figura3.1b, onde mesmo reduzindo em 1 o número de cruzamentos entre os níveis *l* e *l+1* o número de fios entre *l+1* e *l+2* sobe de 7 para 9 e o número de cruzamentos sobe de 2 para 7. Portanto, quando essa abordagem é executada sobre todos níveis de um grafo, o resultado final apresenta um considerável aumento na complexidade das interconexões no nível das entradas primárias, o qual não é tratado em [Ravichandran et al., 2004] e [Bubna et al., 2008]. Além disso, o layout QCA não é gerado nem validado nestes trabalhos anteriores [Ravichandran et al., 2004; Bubna et al., 2008].

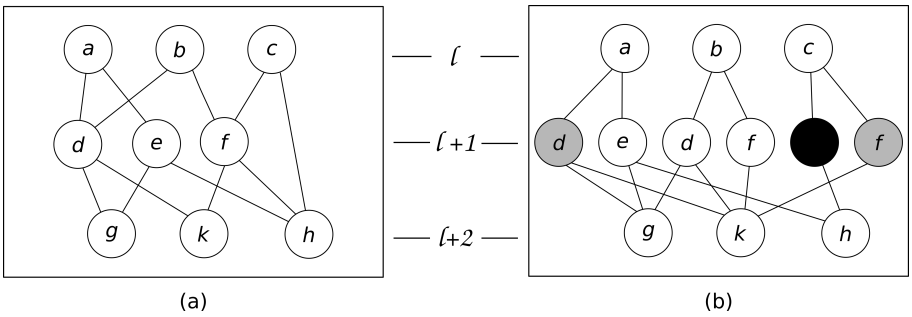


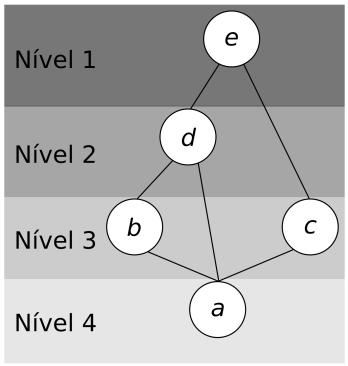
Figura 3.1: Inserção e duplicação de nós no grafo.

3.3.2 Nível de Layout

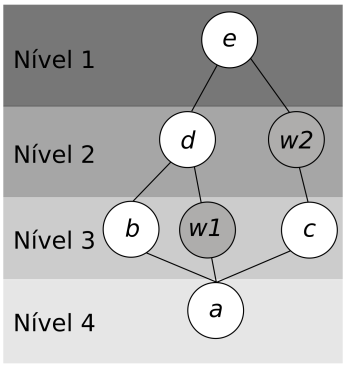
A abordagem em nível de layout deve respeitar regras como distância mínima entre fios, bifurcações, etc. Ao respeitar as regras, a complexidade do problema de P&R aumenta. Em [Reis et al., 2016], uma abordagem sobre o esquema de clock USE é apresentada, entretanto o P&R é feito de forma manual. Abordagens que geram o P&R de forma automática são apresentados em [Trindade et al., 2016] e [Teodósio & Sousa, 2007]. Em [Teodósio & Sousa, 2007] os fios de entradas primárias consomem mais de 30% da área total devida a replicação de nós. Esta abordagem não é baseado no esquema de clock USE e utiliza zonas de clock irregulares que são geradas durante o processo. Em [Trindade et al., 2016], a primeira abordagem para o P&R sem replicação de nós baseada no esquema USE foi apresentada. Nessa

abordagem, o P&R é feito por níveis e nós identidade para balanceamento de rota são inseridos no grafo, antes de fazer o P&R. O balanceamento é feito por níveis usando a ordem ASAP (As Soon As Possible). O P&R é feito nível a nível após balancear cada rota, dessa forma um nível é posicionado baseando-se apenas no nível anterior a ele e a distância entre eles vai sendo aumentada até que todos nós naquele nível possam ser posicionados, isso gera uma forma fácil de se posicionar níveis iniciais, entretanto da mesma forma que a abordagem a nível de portas tende a propagar interconexões mais complexas para o nível das entradas primárias, essa abordagem também a faz. Além disso, essa abordagem tende a propagar a complexidade para níveis onde reconvergencias são originadas. O distanciamento entre níveis não deve ser analisado unicamente baseando-se em um ótimo local e por isso ao fazer uso de uma heurística gulosa para posicionar cada nível, essa abordagem [Trindade et al., 2016] desconsidera informações topológicas que impactam na qualidade do P&R. Além disso, a abordagem apresentada em [Trindade et al., 2016], se limita a grafos pequenos com padrões de reconvergências simples.

A Figura 3.2 ilustra o processo de balanceamento dos níveis. A Figura 3.2a ilusta o grafo sem balanceamento e a Figura 3.2b o grafo balanceado após a inserção dos nós de balanceamento *w1* e *w2*. Uma vez balanceado os níveis, o posicionamento irá percorrer o grafo da saída para as entradas, começando cada nível com um valor mínimo de distância e propagando para o próximo nível tendo apenas o nível anterior como referência. Dessa forma, os caminhos reconvergentes, que tem impacto no roteamento, são totalmente ignorados.



(a) Grafo sem balanceamento.



(b) Grafo balanceado.

Figura 3.2: Grafo separado por níveis

A abordagem em nível de layout proposta em [Trindade et al., 2016], apesar de criar projetos de circuitos QCA automaticamente para pequenos grafos AIOG, não explora características topológicas do grafo importantes para o P&R. Nessa

abordagem o espaçamento entre níveis é definido localmente de forma gulosa, buscando sempre que possível diminuir o espaçamento local entre o nível atual e o nível anterior já posicionado. Dessa forma as soluções geradas tendem a apresentar pior qualidade quando comparadas com soluções criadas a partir da análise topológica do grafo. Nesse trabalho, as soluções obtidas analisando o impacto topológico dos caminhos recovergentes se mostram superiores, em função da área total, quando comparados com os resultados obtidos em [Trindade et al., 2016]. Para ilustrar melhor o impacto de usar uma política gulosa de espaçamento de zonas de clock iremos utilizar um exemplo. Na abordagem [Trindade et al., 2016], a primeira tentativa de posicionamento do nível $i + 1$ em relação ao nível i, irá buscar uma solução inicial com apenas uma zona de clock entre os níveis.

A Figura 3.3 ilustra dois resultados para o grafo do circuito somador de 1 bit com diferentes espaçamentos entre níveis. O grafo de cada solução mostra o valor de espaçamento entre cada nível, que será referenciado pelo vetor de pesos [2,1,2,1,1,2] vinculado ao grafo da Figura 3.3a. Observe que o nível zero (da saída primária do circuito) tem apenas o vértice k. No próximo nível, tem-se dois vértices i e j que para esta solução não gulosa começam com um espaçamento de 2 zonas de clock. No próximo nível, o espaçamento usa apenas 1 zona de clock e no nível subsequente usa 2 novamente. Já a solução gulosa, irá começar com o espaçamento mínimo nos primeiros níveis, como ilustra o vetor de pesos [1,1,2,1,1,2] vinculado a Figura 3.3b que seria uma solução gerada pela abordagem proposta em [Trindade et al., 2016]. Já o distanciamento entre níveis para o grafo mostrado na Figura 3.3a foi feito de forma a permitir maior flexibilidade no posicionamento de cada nó, sendo para isso estudado as interações entre cada caminho reconvergentes encontrado nesse grafo. Os resultados obtidos pelas duas abordagem são mostrados lado a lado ao seus respectivos grafos. É notável a redução na área da solução mostrada na Figura 3.3a quando comparada com a área da Figura 3.3b obtida de forma gulosa [Trindade et al., 2016]. A redução da área é de 43%, além de apresentar aproveitamento muito maior da área total gasta. Além disso, nossa abordagem explora várias soluções, pois em função do vetor de pesos, o espaço a ser explorado se modifica, possibilitando reduções.

Diferente das abordagens anteriores[Trindade et al., 2016] com o esquema de clock USE, este trabalho apresenta um exploração ampla do espaço de soluções, geração de mais de uma solução, estudo do impacto dos caminhos reconvergentes, técnicas de particionamento baseados no paradigma da divisão e conquista para reduzir a complexidade e mapear circuitos maiores de forma automática. Além disso, portas com três entradas são mapeadas, o que não ocorre na abordagem

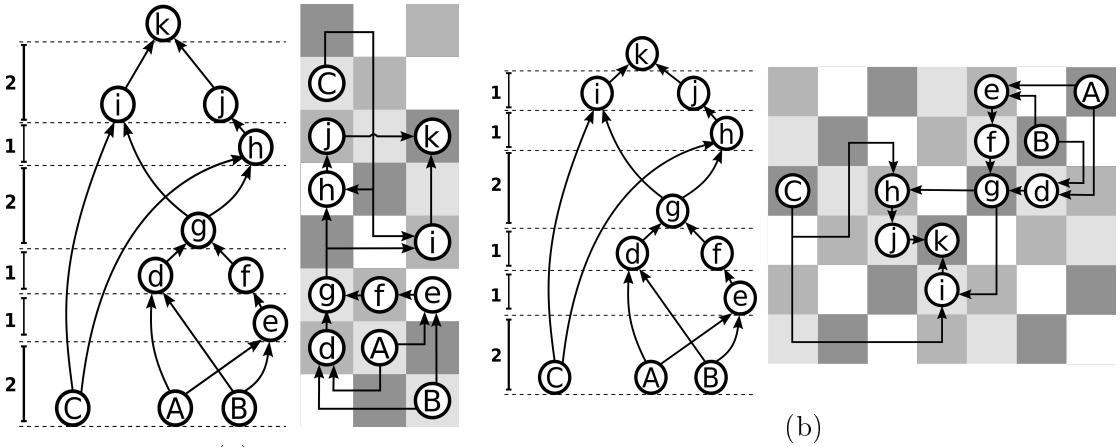


Figura 3.3: Impacto do espaçamento entre níveis.

anterior [Trindade et al., 2016] e também introduzimos a possibilidade de mapear um subconjunto das funções lógicas da porta de maioria de entradas limitada a três ou quatro entradas.

O trabalho recentemente apresentado em [Torres et al., 2018], busca fornecer uma alternativa para o P&R de circuitos QCA. Entretanto, os circuitos QCA gerados mostram falta de sincronismo em suas entradas. Além da possível inconsistência gerada pela falta de sincronismo, isso torna a comparação de resultados entre trabalhos um tarefa custosa. Isso porque, é preciso sincronizar as entradas dos circuitos QCA manualmente para que a comparação seja válida.

Capítulo 4

Algoritmo de Posicionamento e Roteamento

4.1 Formulação do Problema

Nesta dissertação iremos mapear um grafo no nível de portas lógicas em um layout QCA com o esquema de clock USE. Mais especificamente, o posicionamento e roteamento de um circuito terá como entrada um grafo no nível de portas no formato MIG(*Majority Inverter Graph*). O MIG tem três componentes: entradas primárias, saídas primárias e nós internos ou portas. As portas lógicas são do tipo portas de maioria ou inversor. Ao separar o MIG em níveis, convencionando nível zero para todas as portas de saída, define-se a distância inicial entre níveis para representar um parâmetro necessário para o P&R e a partir do MIG pode-se prosseguir no processo.

Dado o MIG de um circuito e uma matriz de células QCA no formato USE, o problema de posicionamento consiste em encontrar e vincular cada nó do grafo MIG a uma única posição da matriz garantindo que cada célula USE seja ocupada por no máximo um nó. As posições dos nós devem preservar as mesmas distâncias em zonas de clock dos valores de espaçamento representadas no MIG. Suponha um nó A no nível *n1* e um nó B no nível *n2* do MIG e suponha que exista uma aresta de A para B. Suponha que o espaçamento de n_1 e n_2 seja igual a d_{12}. As distancias entre as posições vinculadas a A e B na matriz USE deve ser de d_{12} zonas de clock USE.

Estabelecidas as posições de dois nós na matriz USE, então o processo de roteamento é realizado buscando uma sequência de posições na matriz que preservem

o fluxo de caminhos no USE. A essa sequência é dada o nome de rota. A rota terá origem na posição ocupada por A na matriz USE e tendo como destino a posição ocupada por B, além disso a rota não pode conter nenhuma posição que já esteja ocupada por portas lógicas ou portas de entrada ou saída. Fios duplos são permitidos e alguns cruzamentos, que são parâmetros que podem ser configurados com um determinado limiar. Importante ressaltar que a rota deve ter o número de posições igual a distância entre A e B, representada no MIG, subtraída de 1 por conter a posição de origem. Vale observar que podem existir mais de uma rota com o mesmo tamanho entre origem e destino ao percorrer posições diferentes.

Uma célula USE é definida como sendo a área quadrada, com altura e largura do tamanho de cinco células QCA, onde podem haver até 25 células QCA distribuídas nessa área de 5x5 desde que não se sobreponham. Ao se encontrar uma solução sua qualidade é avaliada em função da área total em células USE. Número de cruzamentos e percentual de ocupação da área total são considerados fatores secundários na classificação de uma solução.

4.2 Algoritmo

Um algoritmo exato para encontrar soluções ótimas de P&R para um dado grafo MIG deve explorar todo espaço de busca. A tarefa de explorar todo espaço de busca se limita a grafos de tamanho reduzido. No caso de P&R para QCA, o tamanho de um grafo MIG pode ser definido em função do número de vértices, de arestas e deve incluir também o espaçamento entre níveis bem como suas reconvergencias e outras informações topológicas. O espaçamento entre níveis afeta a solução e contribui para o crescimento do espaço de busca. Vale ressaltar que o espaçamento nesta dissertação segue uma busca em largura usando a política ASAP (As Soon As Possible) para simplificar, pois o grau de liberdade seria maior se fosse permitido variar os nós que tem valores diferentes de profundidade entre o ASAP e a política ALAP (As Later As Possible). Uma métrica capaz de medir o tamanho de um grafo auxilia no processo de partição, resultando assim na simplificação do P&R. O particionamento permite decompor o problema, para que o P&R de cada partição seja realizado individualmente.

O esquema de clock USE cria restrições na distribuição de zonas de clock e na forma que as entradas de uma porta recebe os sinais, pois esses devem percorrer a mesma distância em zonas de clock entre os níveis consecutivos i e i+1. As possibilidades de posicionamento com distâncias de 2 a 5 são ilustradas na Fi-

gura 4.1. Pode-se observar as distâncias para o posicionamento das entradas para uma determinada porta A. Existem apenas 4 opções de posicionamento válido para suas entradas com distância 2, marcadas em cinza na Figura 4.1a. Considerando a distância 3, existem 8 possibilidades de posicionamento. Observe que existe um padrão em diagonal que é gerado pela arquitetura de linhas e colunas do USE, onde as linhas pares tem o roteamento da direita para esquerda e as linhas ímpares da esquerda para a direita. Em termos de coluna, nas pares o roteamento é de cima para baixo e as ímpares de baixo para cima. À medida que a distância cresce existem mais possibilidades, porém aumenta o atraso do circuito. Nas possibilidades de posicionamento ilustradas com uma distância i, é possível gerar distâncias i+4, i+8, ... devido ao USE que agrupa todas essas distâncias sob a mesma classe de congruência modulo 4.

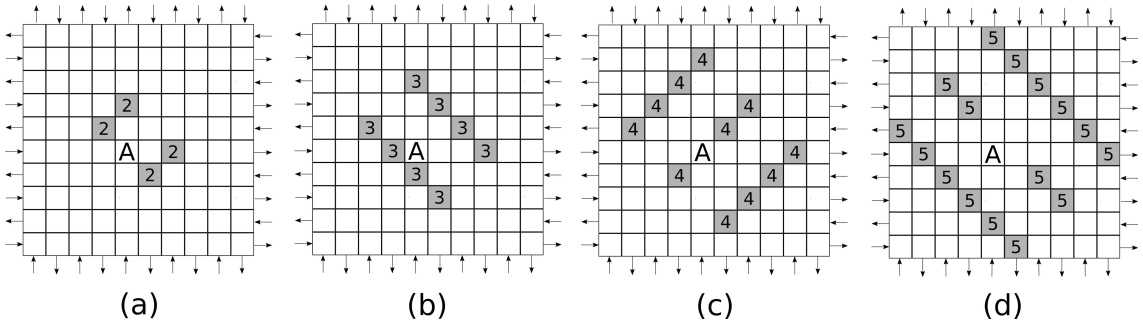


Figura 4.1: Distâncias na matriz USE: (a) Distância 2; (b) Distância 3; (c) Distância 4; (d) Distância 5.

Um nível com n nós é posicionado avaliando combinações de todas posições candidatas para cada nó do nível. Um dos critérios para o particionamento é buscar reduzir o número de nós por nível. O número de locais válidos aumenta com a distância como mostrado na Figura 4.1, tornando-se um fator significativo no cálculo do número de combinações a serem avaliadas.

A Figura 4.2 mostra duas diferentes configurações de espaçamentos para um mesmo grafo de circuito. A Figura 4.2a e 4.2c apresenta o valor de espaçamento entre cada nível assim como o comprimento em células USE que cada fio deve possuir para garantir o sincronismo, sendo esse calculado a partir dos espaçamentos entre níveis.

O espaçamento entre cada nível é definido previamente por um vetor de inteiros chamado vetor de pesos. Esse vetor possui n-1 posições sendo n o número total de níveis que o grafo possui. O valor na l-ésima posição do vetor de pesos, representa o espaçamento entre os níveis l e l+1 no grafo. Sendo assim o vetor de pesos associado ao grafo da Figura 4.2a é [2,2,2] e da Figura 4.2c é [1,2,1]. Um resultado de P&R

para o grafo da Figura 4.2a é mostrado na Figura 4.2b e para o grafo da Figura 4.2c é mostrado na Figura 4.2d. Nesse exemplo a redução no espaçamento implica na diminuição da área total da solução, entretanto, isso nem sempre ocorre. Existe situações onde uma estratégia gulosa acaba acarretando no aumento da área na solução final.

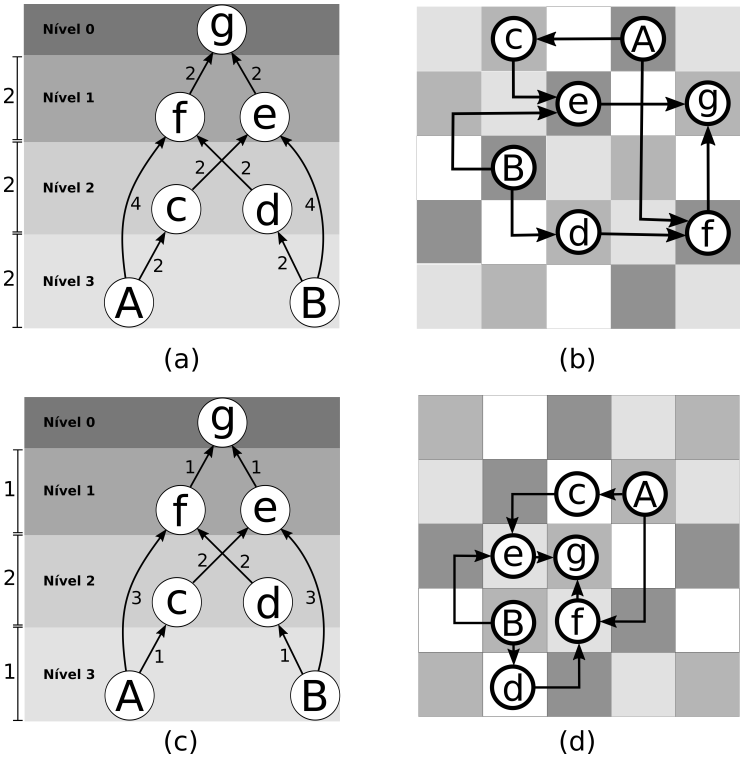


Figura 4.2: Dois exemplos com variação do espaçamento entre níveis: (a) Grafo com vetor de pesos (2,2,2); (b) Layout USE para (2,2,2); (c) Grafo com vetor de pesos (1,2,1); (d) Layout USE (1,2,1).

Cada solução de P&R encontrada mapeia inicialmente um MIG em uma matriz de células USE. Posteriormente, a solução final representada em função de células QCA é transformada a partir de uma solução USE. Uma célula USE representa uma zona de clock e é formada pelo agrupamento de células QCA sob uma mesma fase de clock. A Figura 4.3a mostra o circuito de um somador de 2 bits, os números dentro das portas representam o atraso total, medidos em células USE, dos seus sinais de entrada.

O exemplo da Figura 4.3b ilustra a transformação de um grafo no nível de portas lógicas em um layout QCA, passando pelo passo intermediário de geração do posicionamento e roteamento no USE. O exemplo ilustra a representação com portas lógicas já organizada em níveis por profundidade. Observe que para este exemplo, cada nível está rotulado com um espaçamento inicial 2 na Figura 4.3a. O

posicionamento irá se basear nesta distância para encontrar uma solução na grade do USE como ilustrado na Figura 4.3b. Posteriormente, em uma etapa final, o layout QCA é gerado. O USE possibilita esta abstração do layout final.

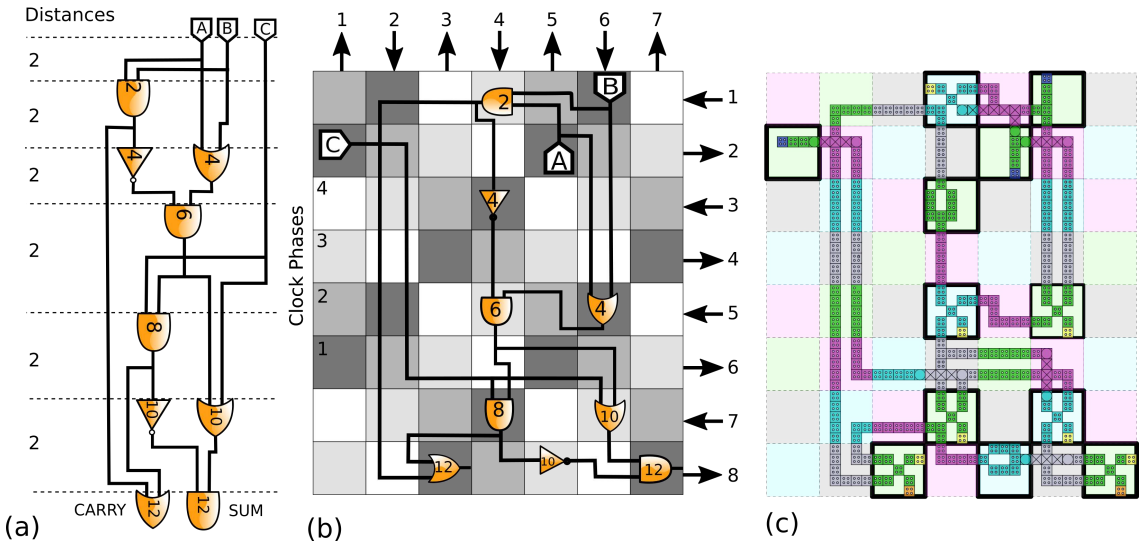


Figura 4.3: Somador de 1 bit: (a) Grafo AOIG com pesos (2,2,2,2,2,2); (b) Layout USE; (c) Layout QCA detalhado.

O fluxo completo de execução é apresentado na Figura 4.4, a partir de um arquivo descrevendo o grafo MIG. Uma partição é um subconjunto de nós conexos e cada uma é definida no arquivo de configuração. Inicialmente as duas primeiras partições têm o P&R realizado e todas suas soluções encontradas são combinadas através da sobreposição. A seguir cada partição tem seu P&R realizado e suas soluções combinadas com o resultado armazenado das sobreposições das partições anteriores. Após a ultima partição ser sobreposta, cada solução encontrada é transformada em formato de células QCA. Por fim cada solução QCA gera um layout QCA que pode ser validado no QCADesigner.

Nas próximas sessões todo o fluxo é mostrado detalhadamente. Na Sessão 4.3 é mostrado a estrutura de entrada para o algoritmo. A Sessão 4.4 mostra a estrutura de dados criada para descrever o grafo de circuito e armazenar uma solução. Na Sessão 4.5 o algoritmo de posicionamento e roteamento é mostrado de forma detalhada. A Sessão 4.6 aborda o particionamento do grafo, baseado no paradigma de divisão e conquista. A Sessão 4.7 mostra a técnica de sobreposição, a qual permite construir soluções maiores através da combinação de duas soluções menores. Na Sessão 4.8 é abordado o impacto dos caminhos reconvergentes sobre o problema de P&R. A Sessão 4.9 apresenta a poda de soluções, a qual permite reduzir significativamente o espaço de busca.

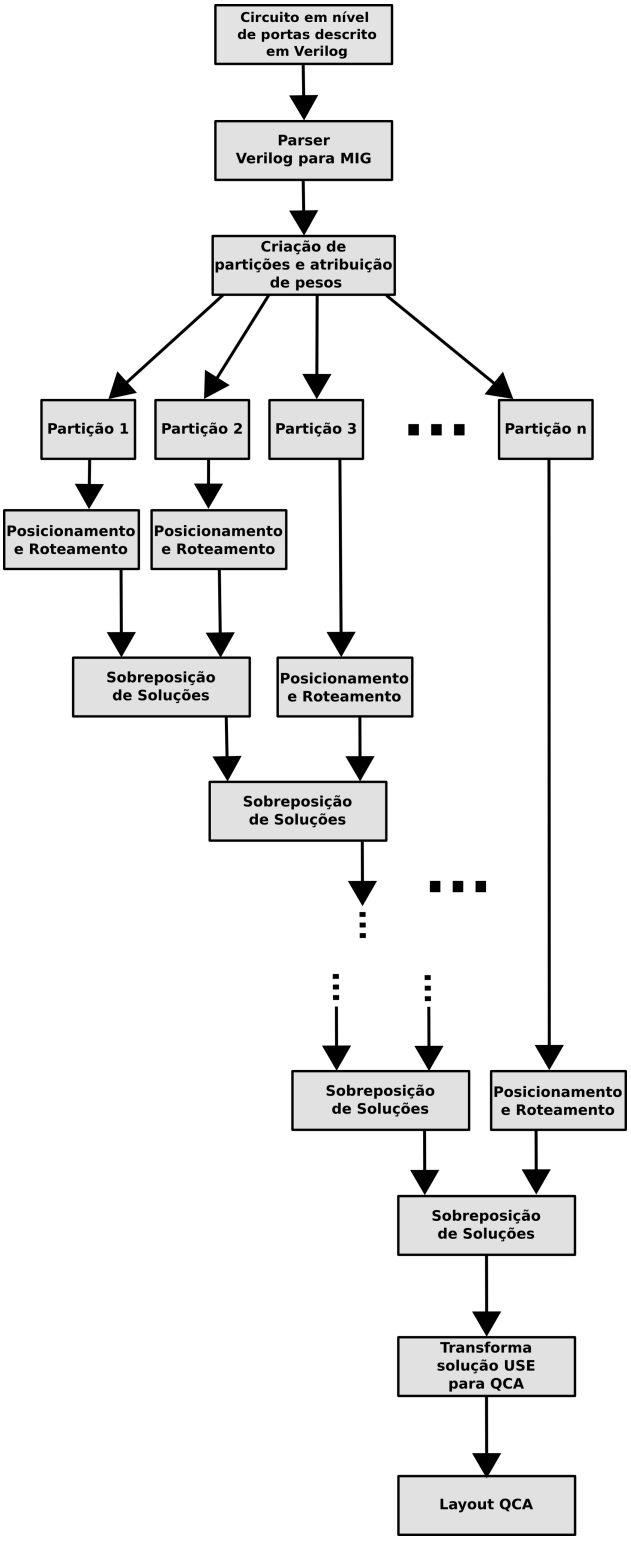
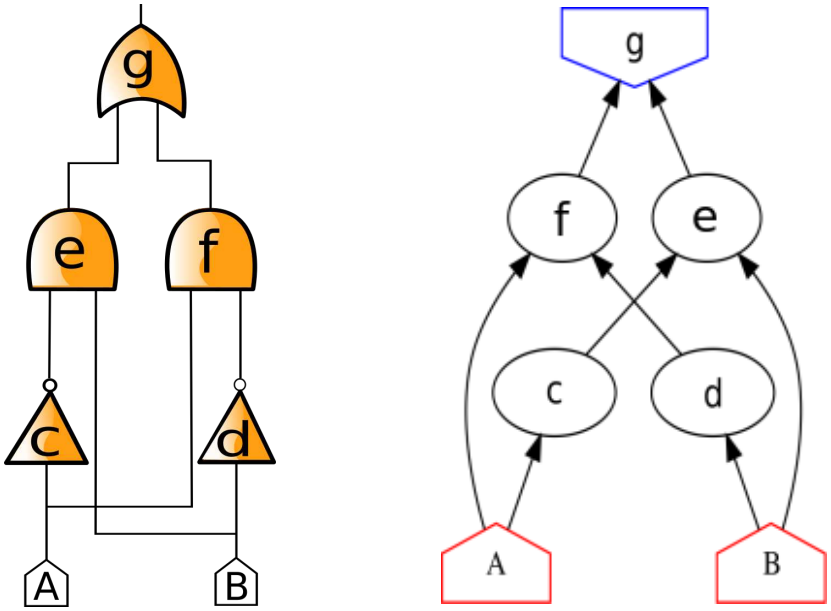


Figura 4.4: Fluxo completo de execução.

4.3 Estrutura de Entrada

O circuito usando lógica MIG é descrito em arquivo na linguagem Verilog. O arquivo de configuração é associado ao circuito para descrever as partições que serão feitas sobre ele. O arquivo de configuração também define a ordem na qual as partições devem ser solucionadas. Por fim, um limite máximo no número de soluções a serem exploradas é estipulado no arquivo de configuração.

Um circuito é representado por um grafo direcionado com portas lógicas, entradas e saídas. Na Figura 4.5a é mostrado um exemplo de um circuito da função lógica *OU exclusivo* (Xor) de 2 bits e na Figura 4.5b uma representação como grafo orientado.



(a) Circuito Lógico (b) Grafo Orientado

Figura 4.5: Circuito lógico e grafo orientado de um xor

Verilog é uma linguagem de descrição de hardware e nesse trabalho foi usado um subconjunto de Verilog para descrever um circuito lógico na forma de um grafo orientado. Apenas quatro primitivas da linguagem foram usadas: as entradas (input), as saídas (output), fios (wires) para nós internos e um comando assign para descrever a função de cada nó. Todas portas devem estar representadas explicitamente em função de suas entradas. Nesse trabalho, inversores serão tratados como portas lógicas, visto que também podem ser implementados nos fios. A seguir é mostrado o código em Verilog correspondente a Figura 4.5a:

```
module xor1bit ( A, B, g);
  input  A, B;
  output g;
  wire c, d, e, f;
  assign c = ~A;
  assign d = ~B;
  assign e = ( c & B );
  assign f = ( A & d );
  assign g = e | f;
endmodule
```

4.4 Estruturas de Dados

O grafo inicial é gerado automaticamente após um parser do arquivo Verilog e do arquivo de configuração com o espaçamento. Durante o processo, cada nó armazena uma lista de rotas. Uma rota é um conjunto de pontos formados por dois valores inteiros indicando a linha e a coluna na matriz USE que o ponto representa. Uma rota sempre deve ser formada de maneira que seu primeiro e último elemento represente a posição das portas de origem e destino, respectivamente. Os demais elementos de uma rota devem necessariamente conter posições onde não haja nenhuma porta posicionada, isso garante que nenhum fio irá ocupar a mesma posição de uma porta lógica. Através dessa estrutura de grafo é possível armazenar o grafo e uma solução de P&R para o mesmo. Quando todas rotas de todos os níveis do grafo são mapeadas na matriz USE, é gerado uma solução.

A matriz é criada, como sendo uma matriz quadrada de células USE, onde sua orientação é definida em função da linha e coluna onde ela se encontra. Para linhas de valor par convenciona-se que a célula naquele local possui entrada pela esquerda e saída pela direita, já para valor ímpar o inverso. Para colunas de valor par a entrada é por baixo e a saída é por cima, e para ímpar a entrada é por cima e a saída por baixo. A Figura 4.6 mostra um exemplo em uma matriz 4x4 onde as linhas e colunas são numeradas de zero a três e o fluxo, representado pelas setas, está em função do número da linha e coluna.

Uma célula da matriz USE contém um valor inteiro de ocupação. A ocupação de uma célula USE é definida como o número de elementos nela posicionados, categorizando-os em dois tipos: fio e porta. Fator de ocupação é um número atri-

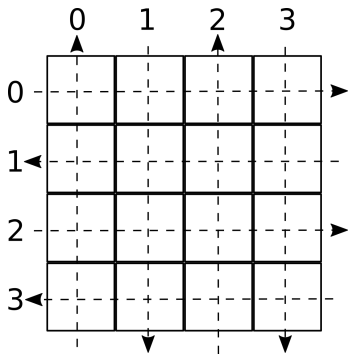


Figura 4.6: Matriz de células USE 4x4 com devida orientação de cada célula.

buído a estes elementos para categorizar. Nesta dissertação usamos como convenção um fio ter o valor igual a 1 e para uma porta o valor 5. Esse fator é usado para impedir que fios e portas sejam posicionados na mesma célula e também para limitar a quantidade máxima de fios em uma mesma célula, tendo limite máximo de fios em uma célula USE definido por uma constante de valor 2.

4.5 Algoritmo

O algoritmo faz o P&R percorrer recursivamente de todos níveis abaixo de um único nó, as soluções são analisadas até o nível para o qual seja possível posicionar e rotear. Uma vez que o último nível do grafo seja alcançado, então obtém-se uma solução e essa é armazenada em uma lista de soluções. O algoritmo trabalha de maneira a construir e destruir soluções parciais ou finais permitindo assim percorrer a árvore de busca de soluções como mostrado na Figura 4.7, realizando um back-tracking até todo espaço de busca ser explorado ou um número limite definido de soluções ser alcançado.

As soluções para um circuito variam em função da posição do nó inicial, isso é devido ao padrão criado pelo USE onde limita-se as posições de entrada de um nó. Quatro combinações devem ser analisadas durante o P&R, sendo elas: o nó inicial posicionado em uma célula de valor de linha e coluna par, linha par e coluna ímpar, linha ímpar e coluna par e linha ímpar e coluna ímpar. Cada padrão de posicionamento gera um conjunto diferente de soluções.

O tamanho da matriz é definido como sendo o dobro da distância entre o último nível e o nível zero, o que garante que o grafo possa ser completamente posicionado respeitando o espaçamento entre níveis. Criada a matriz USE, o algoritmo inicia posicionando o nó de nível zero na posição dada pelo piso do valor obtido da divisão do tamanho de linhas da matriz sobre dois. A partir desse ponto todos os nós do

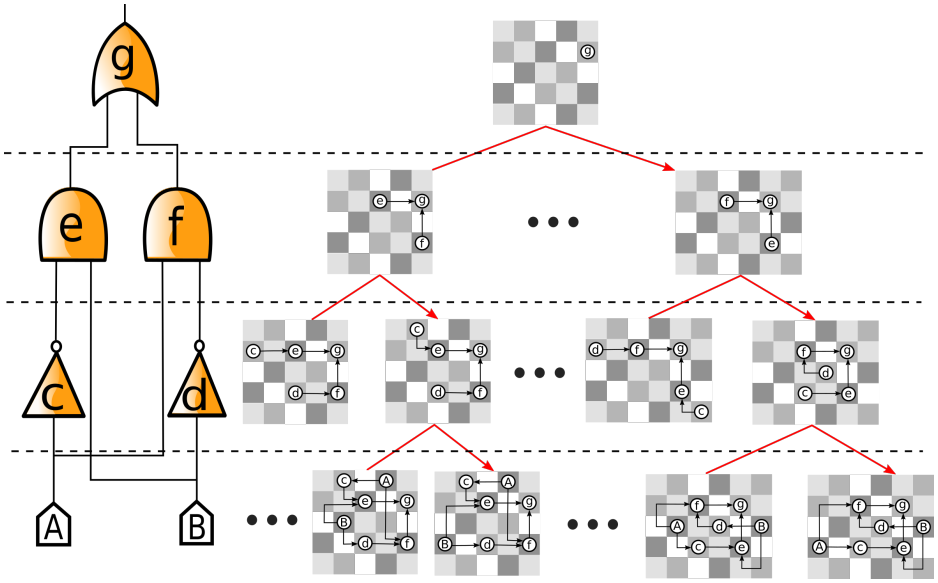


Figura 4.7: Árvore de Busca de soluções mostrando o Bracktracking.

grafo serão posicionados em função de suas saídas já posicionadas, assim sendo, para cada nó no nível 1 é criado dinamicamente uma lista com todas as possíveis posições válida para posicionamento. Uma posição é considerada válida, para um dado nó, quando está vazia e sua distância na matriz USE para cada nó já posicionado é a mesma distância estabelecida entre os níveis do grafo, sendo o nó a posicionar e os já posicionados relacionados por uma aresta. A Figura 4.8 mostra a distância a qual cada célula está em relação à posição central, com valor zero.

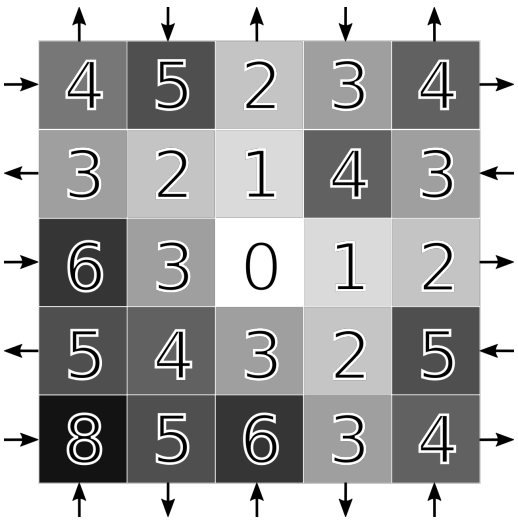


Figura 4.8: Matriz de Distâncias.

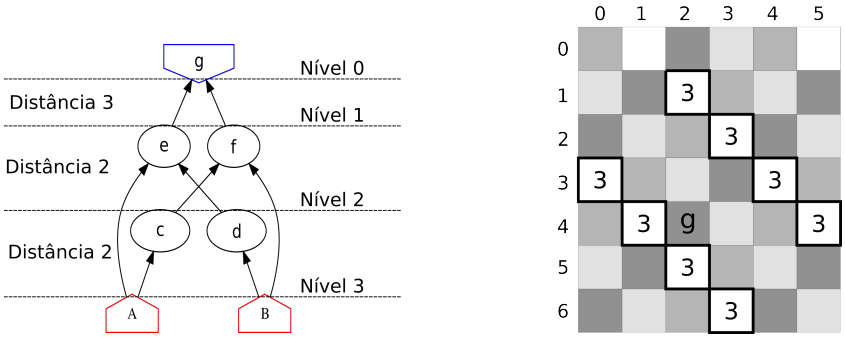
Usando como exemplo o grafo do circuito *Xor*, mostrado na Figura 4.9a, o algoritmo posiciona o vértice g na posição central representada pela distância zero,

a seguir para os nós de nível 1, *e* e *f*, é preenchida uma lista de possíveis posições válidas. Nesse exemplo é mostrado na Figura 4.9b essas possíveis posições em uma matriz USE, tendo como lista para os nós de nível 1 *e* e *f* o conjunto P={(3,0), (4,1), (5,2), (6,3), (1,2), (2,3), (3,4), (4,5)}. A partir da lista de posições válidas de cada nó no nível atual é gerado um conjunto contendo todas as possíveis combinações selecionando 1 elemento da lista de posições de cada nó. No exemplo da Figura 4.9 o tamanho do conjunto gerado obtido pela combinação de 8 elementos combinados 2 a 2 resultando em um total de 28 possíveis combinações. Generalizando para um nível com n nós e supondo que o vetor de posições válidas de cada nó tem uma mesma quantidade k de elementos, o que na prática é variável, então a quantidade de possíveis soluções de posicionamento para esse nível é dado pela combinação de n elementos k a k matematicamente representados por:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Cada elemento no conjunto de combinações cria um ramo na árvore de busca da Figura 4.7. O que é feito nessa etapa do algoritmo é analisar ramos um a um, fazendo o P&R de cada combinação e expandindo o próximo nível dinamicamente tendo como referência os anteriores já posicionados e roteados. O processo se repete para os demais níveis de forma recursiva até que o último nível esteja posicionado ou até alguma combinação não permita realizar o P&R, seja por inconsistências no posicionamento de alguma porta ou por exceder o limite de cruzamentos de fios definido. Caso o último nível seja corretamente processado então esse configuração se torna uma solução válida e é armazenada em uma lista de soluções, logo após isso o P&R é desfeito para o último nível, o que possibilita retroceder na árvore de busca e analisar a próxima combinação no conjunto como pode ser vista na Figura 4.7. Quando o P&R não é realizado com sucesso para alguma combinação então aplica-se o retrocesso para se analisar a próxima combinação. O processe se repete até que todos ramos sejam analisados ou o número de soluções armazenadas atinja um limite previamente definido no arquivo de configuração.

O algoritmo de P&R deve ser executado após o nó de nível zero ser posicionado, um laço o posiciona em quatro posições de linha e coluna diferentes, explorando assim todas as configurações de posicionamento. A função *combinacao* gera todas possíveis combinações para um nível com base no vetor de posições de cada nó nele. A função *posicionamentoRoteamentoNivel* aplica o P&R para um dado nível assim como a função *desfazPosicionamentoRoteamentoNivel* faz o contrário. A variável *V3* é um vetor que guarda soluções geradas pela função *combinacao* para cada nível,



(a) Distância entre Níveis

(b) Posições Válidas para os Nós e e f

Figura 4.9: Circuito lógico e grafo orientado de um xor

nivelMaximo é uma variável que representa o nível máximo do MIG. Dito isso o pseudocódigo do algoritmo de posicionamento e roteamento(P&R) é apresentado a seguir:

Algorithm 1 Posicionamento e Roteamento - P&R

entrada: Um inteiro *nivel*
Saída: Um vetor de MIG *solucoes*
Variável global: Interiro *nivelMaximo*
Variável local: Vetor de soluções para um nível $V3$

```
1:  function POSICIONAMENTOROTEAMENTO(nivel)
2:      if nivel <= nivelMaximo then
3:          V3 ← COMBINACOES(nivel)
4:          if V3 está vazio then
5:              POSICIONAMENTOROTEAMENTO(nivel + 1)
6:          else
7:              for i ← 0 até tamanho de V3 - 1 do
8:                  POSICIONAMENTOROTEAMENTONIVEL( V3[i] )
9:                  if nivel foi posicionado e roteado then
10:                     if nivelMaximo foi posicionado e roteado then
11:                         insere a solução encontrada no vetor solucoes
12:                     end if
13:                     POSICIONAMENTOROTEAMENTO(nivel + 1)
14:                 end if
15:                 DESFAZPOSICIONAMENTOROTEAMENTONIVEL(nivel)
16:             end for
17:         end if
18:     end if
19: end function
```

4.6 Partição

Uma partição é um subconjunto conexo de nós de um grafo e tem como objetivo representar um subgrafo de tamanho reduzido. O particionamento é uma abordagem de divisão e conquista usada para dividir e reduzir o problema principal em subproblemas pequenos o suficiente para poderem ser explorados exaustivamente pelo algoritmo de P&R.

A solução de partições é feita de maneira ordenada tendo como base uma lista de partições onde cada uma ocupa uma única posição. Algumas regras devem ser seguidas para que a próxima etapa possa ser feita. A princípio duas partições que ocupam posições consecutivas na lista devem possuir pelo menos um nó em comum, podendo haver mais de um, isso permite uma forma otimizada de combinar soluções. Se existe uma aresta $A \rightarrow B$ conectando duas partições diferentes, então um dos dois nós, A ou B, deve necessariamente ser um ponto em comum as partições.

O impacto de particionamento no tempo de execução e na qualidade da solução depende de vários fatores topológicos do grafo onde conceitos da teoria das redes complexas podem ser aplicados. A teoria de redes complexas busca representar relações e propriedades nas redes baseadas em sua topologia. Uma rede pode ser interpretada como grafo e medidas feita sobre a rede proporcionam uma forma de classificar os grafos de circuitos digitais além de auxiliar no processo de partição do grafo. Uma importante característica que pode ser analisada em uma partição é fornecida pelo coeficiente de agrupamento ou clusterização. É esperado que em uma partição, os nós estejam fortemente relacionados. A principal função da partição é tratar localmente tais relações encapsulando-as e simplificando o processo de P&R do grafo completo.

Muito utilizado na análise de redes sociais com grafos não direcionados, o coeficiente de clusterização de um nó é calculado como o número de conexões entre seus vizinhos dividido pelo máximo de conexões que podem existir entre eles. A Figura 4.10a mostra as possíveis conexões, em vermelho, entre os nós vizinhos do nó A. Nesse exemplo o máximo de conexões entre os vizinhos é 3. Na Figura 4.10b mostra no grafo onde exitem duas conexões entre os vizinhos de A. Nesse grafo o coeficiente de agrupamento para o nó A é dado pelo número real de conexões de seus vizinhos divididos pelo total de possíveis conexões entre eles, sendo seu valor igual a $2/3$.

O coeficiente de clusterização poderia ser usado para criar partições. Entretanto, os grafos de circuitos digitais são direcionados e não se adaptam diretamente a esta métrica. Portanto foi preciso criar outra métrica inspirada no coeficiente de

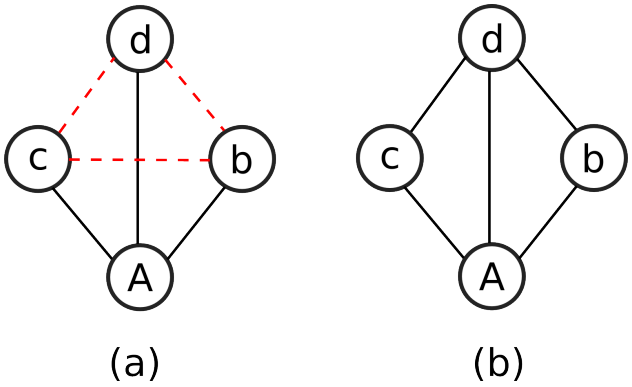


Figura 4.10: Calculo do coeficiente de agrupamento.

clusterização, denominado aqui como coeficiente de reconvergencia. Este coeficiente foi criado para medir a qualidade de uma partição tendo como base a proporção de reconvergencias que são capturadas nela. O cálculo do coeficiente de reconvergência é feito sobre seu nó de origem. Ao se definir relações indiretas entre os nós da reconvergencia, excluindo relações com o nó onde os caminhos convergem, é criado o coeficiente de reconvergencia como sendo o número de relações em uma partição divididos pelo número de relações contidas na reconvergencia. A Figura 4.11b mostra uma partição feita sobre o grafo da Figura 4.11a. Essa partição engloba toda reconvergencia com origem no nó A, sendo seu coeficiente de reconvergencia igual a 1. Na Figura 4.11c, uma outra partição sobre o grafo da Figura 4.11a é apresentada. Essa partição possui 5 das 8 relações apresentadas na reconvergência, tendo portanto coeficiente igual a 5/8.

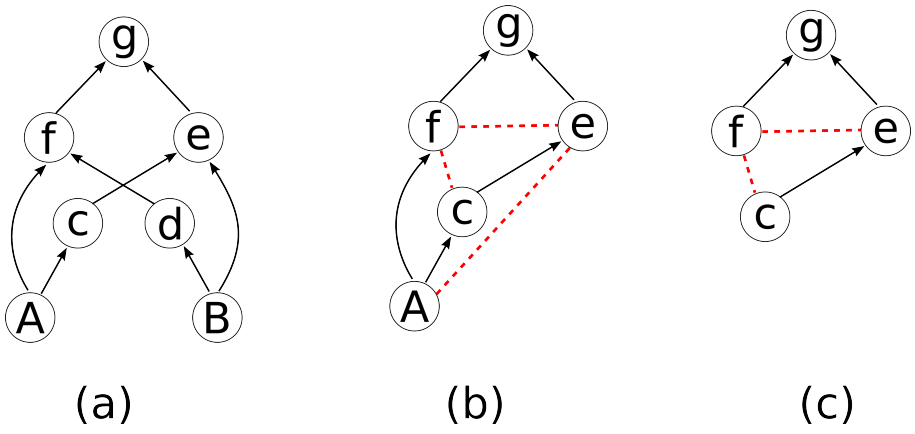


Figura 4.11: Coeficiente de reconvergência.

4.7 Sobreposição de soluções

A sobreposição permite combinar soluções incorporando elemento de ambas, fornecendo assim uma maneira de construir soluções completas através da sobreposição de soluções parciais. Para sobrepor soluções deve-se atender a duas condições. Primeiro é necessário ter pelo menos um nó na interseção de duas partições. Segundo, as partições vão sendo sobrepostas em sequência, a ordem da sequência deve garantir que a interseção é não vazia ao incluir uma nova partição.

Na Figura 4.12, duas partições são feitas sobre o grafo do circuito *Xor*, a partição 1 contém os nós delimitados pela área cinza e a partição 2 é formada pelos nós delimitados pela linha pontilhada, observa-se que a interseção das partições contém os nós *g*, *e* e *f*. Diferente das abordagens tradicionais de mapeamento ou posicionamento em circuitos digitais, as partições propostas neste trabalho não são disjuntas e tem nós em comum.

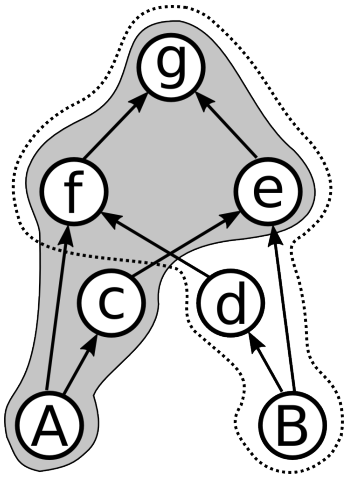


Figura 4.12: Particionamento sobre o grafo do circuito *Xor*.

A princípio, um nó na interseção das partições é escolhido e a partir dele uma solução é transladada de forma que a posição na matriz USE desse nó coincida em ambas soluções desde que o traslado da solução seja válido. O translado de uma solução deve ser feito de forma a preservar as características da solução para não gerar inconsistência com o esquema do USE. Isso é feito escolhendo-se um nó e a ele impondo a condição de manter o mesmo layout de entrada e saída, ou seja, se um nó está em uma posição de linha par e coluna impar, ele só poderá ser movido para alguma outra posição que preserve essa característica, o mesmo se aplica para as demais combinações de linha e coluna pares e impares. Outra forma de se garantir essa característica é limitar as constantes de variação vertical e horizontal apenas a valores pares. Satisfeita essa condição, duas constantes resultantes da diferença da

entre a posição inicial e a posição final indicam a variação vertical e horizontal a qual a solução deve ser transladada.

Se o translado da solução pode ser realizado o próximo passo é garantir que todos nós que pertencem a interseção das partições estejam na mesma posição. Se a posição de todos nós na interseção dos grafos coincide então cada solução é mapeada em uma matriz USE e essa é transformada em uma matriz de fator de ocupação, isso permite resolver o problema de sobreposição através de uma soma de matrizes e algumas operações lógicas básicas sobre seus elementos.

Neste trabalho, a implementação da matriz de ocupação é formada por números inteiros de 32 bits que são conceitualmente divididos em 3 faixas como mostrado na Figura 4.13. 14 bits são usados para indexar nós do grafo, existindo a possibilidade de indexar dois nós por posição na matriz de ocupação. Os quatro bits menos significativos são usados para controlar a ocupação na posição correspondente. Com 14 bits é possível indexar 16384 nós de forma única podendo assim suportar grafos com no máximo essa quantidade de nós, caso haja necessidade de trabalhar com grafos maiores essa estrutura pode ser estendida para suportar lista de inteiros podendo assim ser escalável para trabalhar sobre grafos maiores.

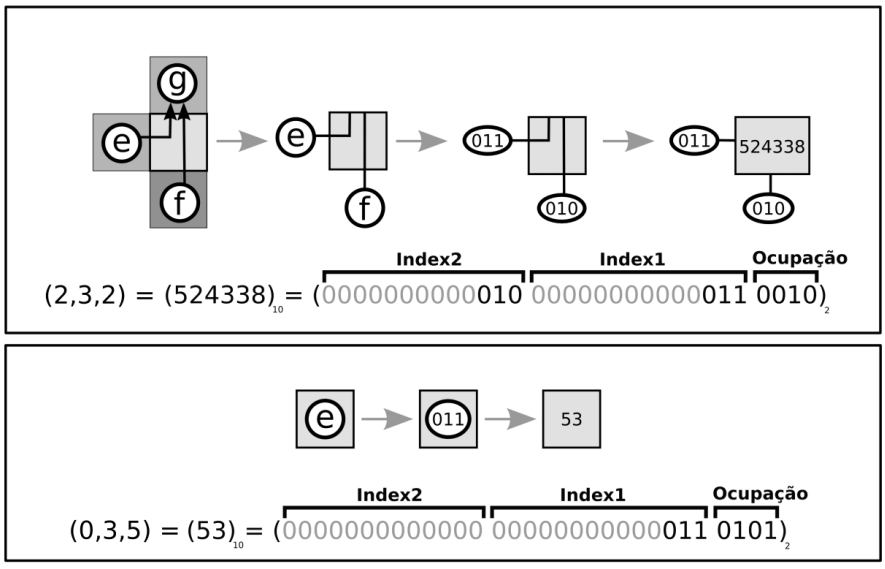
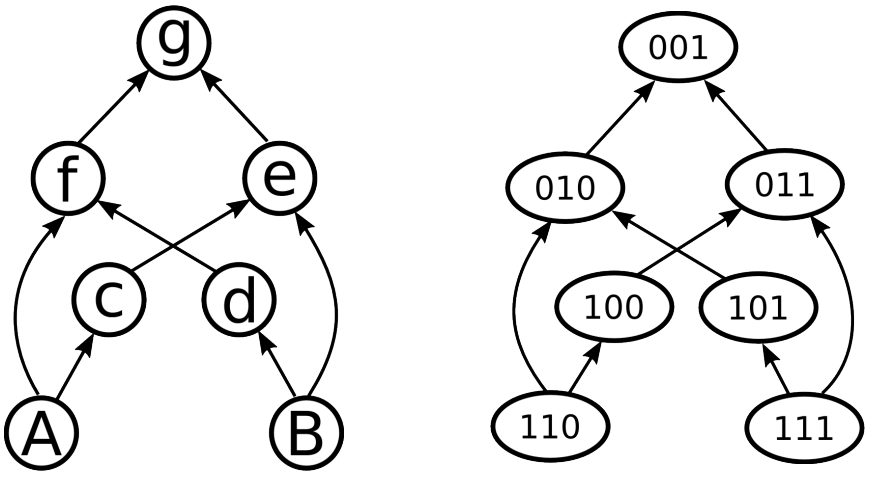


Figura 4.13: Divisão de um inteiro de 32 bits em uma matriz de ocupação.

A Figura 4.14b mostra uma indexação para o grafo na Figura 4.14a, cada nó está associado a um único índice. A indexação é feita por um hash perfeito permitindo assim recuperar o nome de um nó através do seu índice.

Na Figura 4.13 são apresentados 2 exemplos de ocupação para uma célula juntamente com sua mascara de ocupação. No primeiro exemplo uma célula é

ocupada por dois fios, sendo cada fio associado ao valor mapeado a seu nó de origem. Sua mascara de ocupação associa essa célula aos dois índices, um de cada fio, e associa a ocupação o valor 2, indicando assim a existência de dois fios nessa célula. No segundo exemplo é ilustrado a situação onde uma porta ocupa a célula. Nesse caso, um único índice é associado a célula, o índice da porta em si, e ao seu valor de ocupação é atribuído 5.



(a) Grafo *Xor*. (b) Indexação de nós.

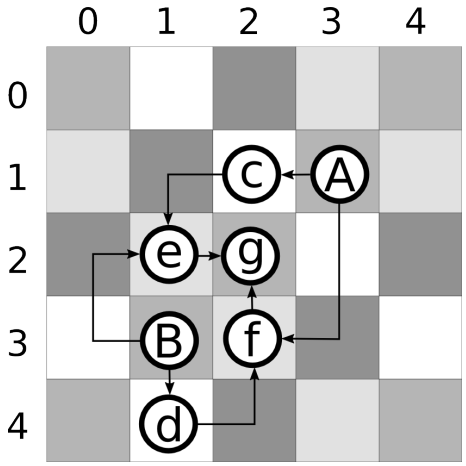
Figura 4.14: Indexação de nós no grafo do circuito *Xor*

A matriz de ocupação associada a uma solução encontrada para o circuito *Xor* é apresentada na Figura 4.15a, a solução associada a essa matriz é apresentada na Figura 4.15b. Fios têm fator de ocupação igual a 1 e o índice associado a ele é igual ao índice de sua porta de origem. Cada fio possui apenas uma porta de origem e sendo assim possui um único índice atrelado. O fato de haver 2 índices em cada posição visa permitir representar cruzamentos de fios, nesse caso os dois índices serão diferentes de zero e a ocupação será a soma das ocupações individuais. A Figura 4.13 representa a posição (1,3) na matriz de ocupação da Figura 4.15a. A ocupação nesse posição é 5 indicando ser ocupada por uma porta, index1 é igual a 110 que corresponde ao nó *A* e index2 igual a 0 por não corresponder a nenhum nó.

O processo de validar uma sobreposição pode ser feito realizando o P&R de duas soluções sobre o mesmo local, caso isso possa ser feito tem-se a garantia que a sobreposição é valida. Outra forma mais eficiente de validar alguma sobreposição é através da matriz de ocupação. Tendo duas matrizes de ocupação devidamente preenchidas, as posições correspondentes tem seus 4 bits de ocupação somados. Caso o resultado obtido seja maior que 5, tem-se a garantia que a sobreposição dessas duas soluções é inválida. Uma sobreposição é considerada válida se os valores de

	0	1	2	3	4
0	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
1	(0,0,0)	(0,4,1)	(0,4,5)	(0,6,5)	(0,0,0)
2	(0,7,1)	(0,3,5)	(0,1,5)	(0,6,1)	(0,0,0)
3	(0,7,1)	(0,7,5)	(0,2,5)	(0,6,1)	(0,0,0)
4	(0,0,0)	(0,5,5)	(0,5,1)	(0,0,0)	(0,0,0)

(a) Matriz de ocupação.



(b) Solução em matriz USE.

Figura 4.15: Matriz de ocupação associada a uma solução para o circuito *Xor*

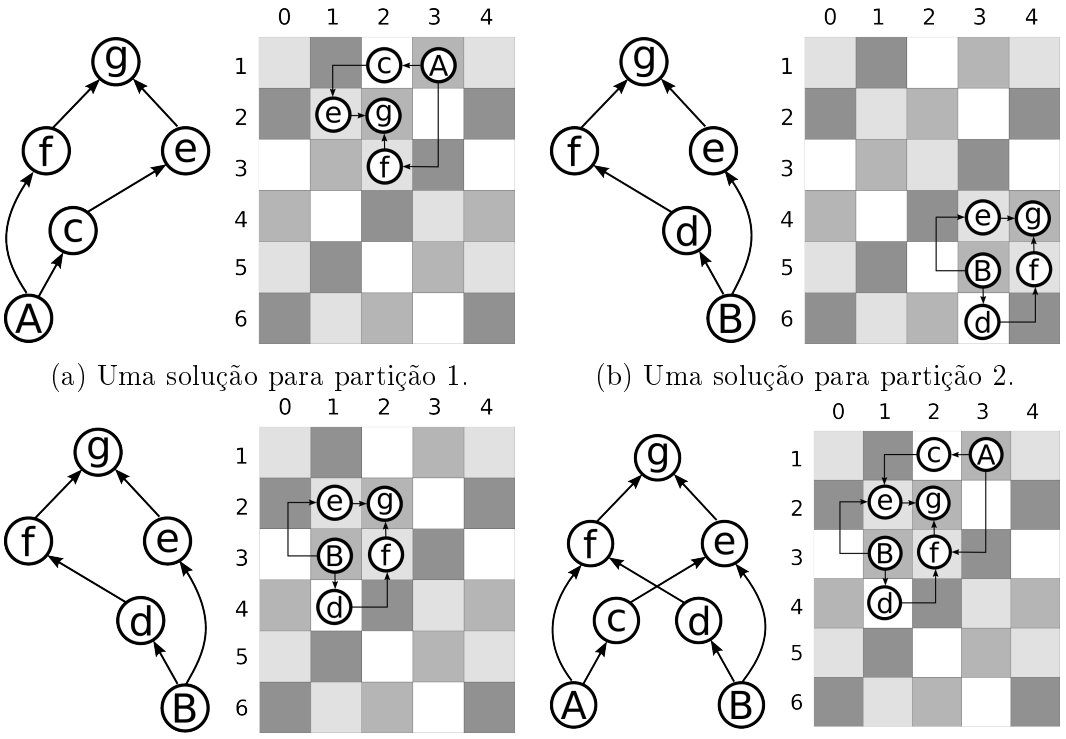
ocupação somados de cada posição nas 2 matrizes forem 1, 2 ou 5. Essa estratégia evita esforço desnecessário, realizando o P&R apenas de soluções compatíveis.

A Figura 4.16b mostra uma solução encontrada para a partição 2, nesse exemplo é calculado sobre o nó g as constantes de deslocamento que possibilitam mover esse nó da posição (4,4) para a posição (2,2) onde o nó g da partição 1 se encontra. Aplicando a variação resultante de -2 vertical e -2 horizontal sobre todos elementos do grafo se obtêm o resultado mostrado na Figura 4.16c a qual é sobreposta a uma solução encontrada para partição 1 mostrada na Figura 4.16a. Uma vez garantida a compatibilidade das soluções, todos nós na interseção ocupam o mesmo lugar após a transposição e vértices não se sobrepõe a nós, a soluções são sobrepostas gerando uma solução USE para o circuito *Xor* apresentada na Figura 4.16d.

4.8 Caminhos Reconvergentes

A complexidade do problema de P&R cresce exponencialmente e deve-se ter formas de se descartar soluções parciais antes mesmo que essas possam ser expandidas na árvore de busca. Esse processo é chamado poda, pois através dele todo um ramo da árvore é cortado antes mesmo que possa ser expandido, eliminando parte do espaço de busca que não irá gerar soluções viáveis.

Existem diversas maneiras de se aplicar uma poda sobre soluções, uma delas já é usada e encontra-se no próprio algoritmo de P&R, que ao processar nível a nível de forma ordenada não expande soluções para as quais algum nível não possa ser corretamente processado, eliminando assim quaisquer soluções que possam ser cria-



(a) Uma solução para partição 1.

(b) Uma solução para partição 2.

(c) transposição da solução da partição 2.

(d) Sobreposição de soluções das partições 1 e 2.

Figura 4.16: Circuito lógico e grafo orientado de um xor

das a partir daquele ponto. Uma solução parcial gera um número muito grande de soluções que se expandem a partir dela e dessa forma caso não seja devidamente eliminada o quanto antes, os impactos que ela poderá causar podem mascarar soluções viáveis quando um limite de tempo de busca é imposto.

Estratégias de poda eficientes são criadas sobre características que podem ser identificadas antes mesmo que a expansão seja feita, geralmente restrições físicas e topológicas do grafo. Na poda feita durante o P&R é usada uma restrição física que determina que se não existe pelo menos uma combinação de pontos na matriz USE válida para um nível então a partir desse ponto nenhuma solução pode ser criada. O que é proposto nessa seção é analisar não só os elementos físicos do P&R mas também fatores topológicos do grafo que fornecem informações adicionais sobre possíveis soluções. Um desses fatores é a reconvergencia de caminhos.

Normalmente um nó tem sua posição restrita durante o P&R pela posição de suas entradas e saídas, essa restrição é observada em forma de arestas no grafo o que pode ser interpretado como uma dependência direta. Esse mesmo nó também é limitado de maneira indireta pelas entradas/saídas dos nós que são suas entradas e saídas, pois por transitividade as dependências são passadas nó a nó. Na verdade,

a maioria dos nós de um grafo conexo possui uma dependência dos demais sendo o que muda é a intensidade. É dessa forma que explora-se o conceito de caminhos reconvergentes como sendo o par de rotas com tamanho maior que dois e com nó inicial e final em comum sendo os demais distintos em cada rota. Uma reconvergencia estabelece uma forte dependência entre nós de suas rotas pois a partir de uma abordagem em níveis, como a adotada nesse trabalho, os nós devem manter uma distância compatível e balanceada.

A abordagem de partição do grafo gera um grande problema que resulta na perda de informação contida apenas no grafo completo, onde informações topológicas podem ser perdidas. Um exemplo de informação que pode ser perdida durante esse processo é justamente a relação entre nós de caminhos reconvergentes que podem ser separados em partições distintas. Em um particionamento ideal toda reconvergencia é incorporada dentro de uma única partição, o que geralmente ocorre apenas para grafos muito pequenos. Esse problema ocasiona um prolongamento excessivo no tempo em que uma solução é mantida e expandida, pois apenas quando todos nós da reconvergencia forem incorporados na construção da solução completa é que a poda será efetivamente realizada, resultando assim na inserção de soluções parciais desnecessárias durante a busca. Nesse trabalho buscamos identificar cada caminho reconvergente antes de realizar a partição e usar esta informação para eliminar soluções nas partições.

4.9 Poda de soluções

Estabelecendo uma analogia com a condição de existência de triângulos é possível formular uma maneira de podar soluções de partições tendo os caminhos reconvergentes previamente identificados. Caso a partição seja capaz de capturar por completo uma reconvêrgencia nada pode ser feito, mas caso essa seja dividida em partições distintas então a medida que ela vai sendo mapeada deve-se estabelecer uma relação entre os nós de caminhos distintos pertencentes a mesma. Uma forma de estabelecer essa relação é fazendo uso da teoria das redes complexas.

A teoria de redes complexas busca representar as relações e identificar propriedades em uma rede, que nesse contexto corresponde a um grafo. Por exemplo, suponha o coeficiente de clusterização que tem como princípio representar agrupamentos intrínsecos na rede. Essa propriedade explora a transitividade na rede e busca estabelecer uma relação entre dois nós desde que esses possuam uma relação com um terceiro em comum. Tome como exemplo três nós A, B e C, caso haja uma aresta ligando o nó A e B e outra ligando B e C então existe uma grande possibilidade de existir uma aresta ligando A a C formando um triângulo entre os três.

Os grafos nesse trabalho são direcionados e portanto a clusterização não pode ser usada, mas baseando-se nela é possível estabelecer uma relação entre a topologia e o mapeamento físico dos caminhos reconvergentes, como já mencionado. Para isso é suposto que uma reconvergência foi dividida entre duas ou mais partições, e como o P&R é feito a partir das saída, em algum momento parte da reconvergencia já estará processada. Nessa etapa será preciso calcular a distância física entre os dois nós com maior nível já posicionados em cada uma das duas rotas. Essa distância, juntamente com a distância restrita pelos níveis ainda não posicionados, formam um pseudo-triângulo e nesse momento estima-se que a distância física e as distâncias restritas devem obedecer a regra de existência de triângulos. Uma vez essa não satisfeita então a poda pode ser feita com sucesso.

O cálculo de distância em uma matriz USE não é feito de forma euclidiana e uma das principais características que não se mantém é a comutatividade. A distância de um ponto A para um ponto $B(d_{A,B})$ não é a mesma distancia de B para $A(d_{B,A})$, sendo um resultado direto da restrição de sentido em função da posição. Por esse motivo é preciso sempre estar bem definido o sentido sobre o qual distância será calculada.

Durante o P&R todas reconvergências são monitoradas,caso a distância física na base da reconvergência já posicionada seja maior que a soma das distâncias dos

níveis abaixo dela até o nó de entrada que cria a reconvergência, então pode-se garantir que a partir dessa configuração é impossível concluir o P&R, uma vez que não existem dois caminhos com distância limitada a essa soma que convirjam em uma mesma posição. A partir dos dois nós com maior nível já posicionados, um em cada caminho da reconvergência, e o nó com maior nível na reconvergência ainda não posicionado, tem-se que para o nó ainda não posicionado poderá existir uma posição válida se e somente se a distância entre os posicionados for no máximo o dobro da distância entre algum deles para o não posicionado. Caso não satisfeita essa condição então qualquer solução que possa ser derivada dessa configuração não será válida, portanto essa deve ser descartada.

A Figura 4.17a mostra duas partições, *p1* delimitada pela linha pontilhada contendo os nós A, f e c e a partição *p2* delimitada pela área cinza contendo os nós g, e, f e c. Na Figura 4.17b é mostrada uma solução de P&R para *p2*. Para que uma solução de p1 possa ser sobreposta à solução de p2 mostrada na Figura 4.17b, os nós c é f devem estar posicionados da mesma maneira em ambas. A linha vermelha pontilhada representa a distância física na matriz USE entre c e f, sendo $(d_{c,f})$ igual a 10. Para essa configuração, uma posição válida para A deve estar a distância 2 do nó c e distância 4 do nó f. A menor distância de uma rota de c para f é 10, uma segunda rota de c para f passando por uma possível posição onde A será posicionado possuirá tamanho igual a soma das distâncias $(d_{c,A}) + (d_{A,f})$.

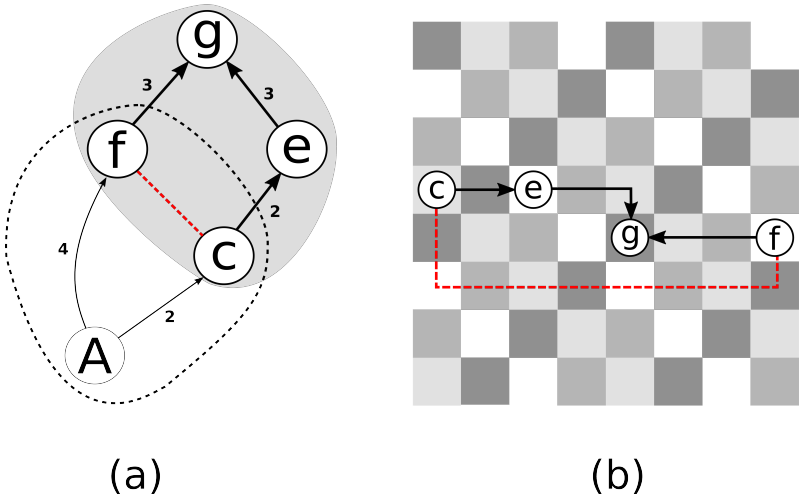


Figura 4.17: Relação entre distância USE e espaçamento entre níveis.

A distância de c para f na Figura 4.17 é miníma e portanto $(d_{c,f}) < (d_{f,A}) + (d_{A,c})$. Note que $(d_{f,A})$ não é necessariamente a mesma que $(d_{A,f})$, sendo $(d_{A,f})$ definida como 4 no grafo da Figura 4.17a. Na poda definida nesse trabalho considera-se a variação máxima na diferença causada pela inversão no sentido de uma rota que

é de 3 células USE, quando a rota ainda não existe mas já se conhece seu tamanho pelo grafo. A Figura 4.18 ilustra o impacto da inversão na direção de uma rota em seu tamanho. A Figura 4.18a mostra a mesma partição definida na Figura 4.17a. Nessa partição uma condição necessária para se posicionar A é de que $(d_{c,f}) < (d_{c,A}) + (d_{A,f})$. Logo pela Figura 4.18b conclui-se que quaisquer soluções que posicionem c e f a distância 10 não podem posicionar o nó A por não existir posição válida. Assim o ramo que seria expandido na árvore de busca a partir da solução da Figura 4.17b é podado.

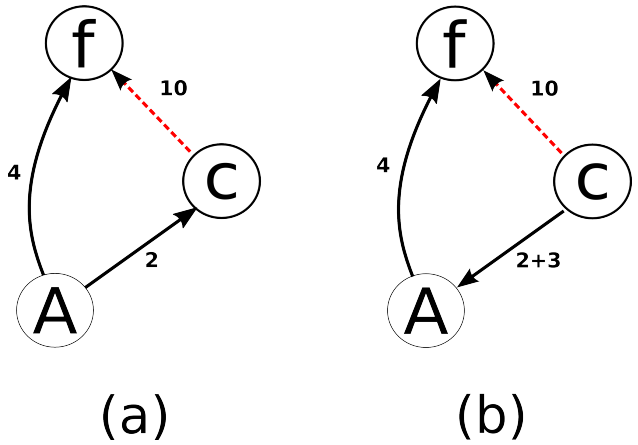


Figura 4.18: Inversão de sentido em uma rota.

Conclui-se que a separação de uma reconvergência entre 2 ou mais partições resulta no problema de propagação de soluções inválidas. A poda por reconvergência é uma forma de se evitar que tal situação ocorra. Tendo que se restringir o espaço de busca devido ao seu tamanho, é necessária uma forma de eliminar soluções inválidas que possivelmente irão saturá-lo.

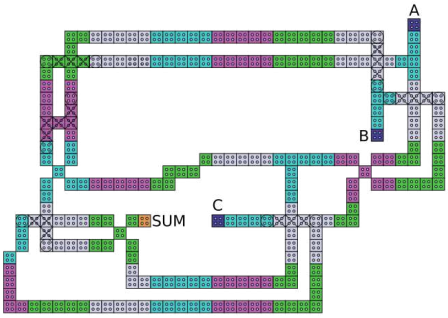
Capítulo 5

Resultados

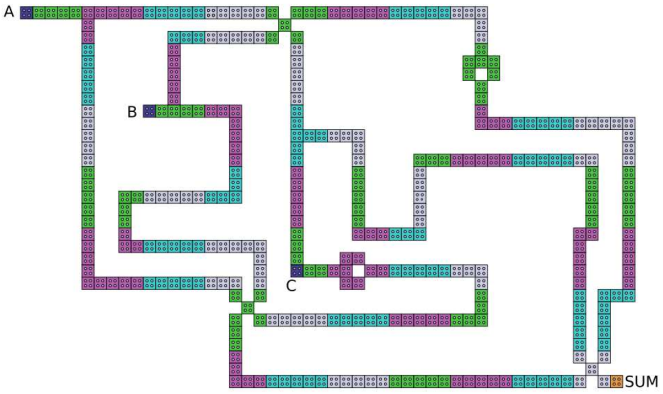
Inicialmente, na seção 5.1, iremos comparar os resultados gerados pela abordagem de posicionamento e roteamento proposta nesta dissertação com os trabalhos anteriores [Reis et al., 2016] e [Trindade et al., 2016] que também utilizam o esquema de clock USE. Os resultados desse trabalho estão quantificados em função de células USE. Tanto área livre e total quanto atraso total serão representados em função de células USE. Os resultados apresentados nesse capítulo são selecionadas entre todas soluções encontradas segundo o critério de menor área. Na seção 5.2 mostramos o impacto da presença de caminhos reconvergentes na área do layout final. Na seção 5.3, o P&R proposto neste trabalho é avaliado para vários circuitos que incluem portas de maioria com 3 entradas e circuitos com até uma centena de portas lógicas. Na seção 5.4, mostramos que é possível também fazer o P&R com portas de maioria de 5 entradas que são limitadas a funções lógicas com três entradas. Esta técnica é promissora, como ilustra o exemplo que apresenta uma redução de 47% na área final. Na seção 5.5, mostramos o impacto das técnicas de poda na redução do espaço de busca. No Apêndice A são mostrados os grafos de circuito assim como suas soluções encontradas nesse trabalho.

5.1 Comparação com trabalhos anteriores baseados no esquema USE

Ao introduzir o particionamento e sobreposição para reduzir a complexidade e possibilitar a exploração do espaço de solução, a proposta de P&R apresentada neste trabalho consegue reduzir de forma significativa a área ocupada e o atraso quando



(a) Layout do somador de 1 bit obtido com o P&R com particionamento e sobreposição.



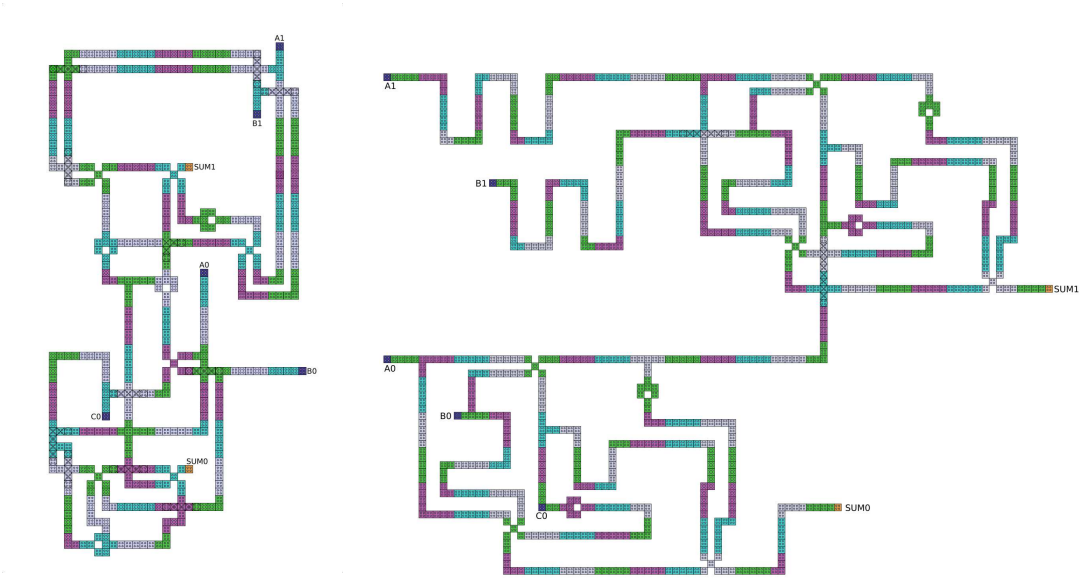
(b) Layout do somador de 1 bit gerado manualmente em [Reis et al., 2016].

Figura 5.1: Comparação entre P&R proposto com o layout manual [Reis et al., 2016] para somador de 1 bit

comparadas as soluções apresentadas em [Reis et al., 2016] e [Trindade et al., 2016]. Na Figura 5.1a é mostrado para um circuito somador de 1 bit construído com portas de maioria. Este resultado foi obtido nesse trabalho com o uso do esquema de clock USE. A Figura 5.1b mostra o resultado para o mesmo circuito criado manualmente em [Reis et al., 2016]. O resultado por sobreposição reduz em 43% a área total em comparação com o solução feita manualmente em [Reis et al., 2016].

Seguindo uma abordagem de biblioteca de células, o somador de 1 bit mostrado na Figura 5.1b foi usada na construção manual de um somador ripple carry de 2 bits usando dois somadores de 1 bit. As entradas do segundo somador são balanceadas prolongando seus fios para sincroniza-las com a saída do primeiro somador. Esse resultado é apresentado na Figura 5.2b. O mesmo circuito foi mapeado pela técnica proposta nesta dissertação usando a sobreposição/partição e o resultado é mostrado

na Figura 5.2a. Podemos observar uma redução na área total de 3 vezes quando comparado ao proposto em [Reis et al., 2016].



(a) Layout para somador de 2 bits gerado com a abordagem de sobreposição/particionamento

(b) Somador de 2 bits a partir de duas células de 1 bit [Reis et al., 2016].

Figura 5.2: Comparação de layout manual [Reis et al., 2016] em bitslice com o layout gerado com o P&R proposto

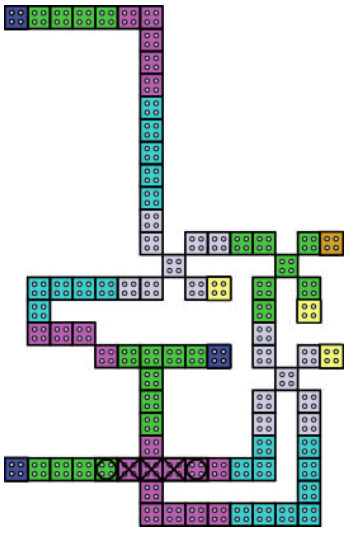
Diferente dos resultados criados manualmente em [Reis et al., 2016], os resultados apresentados em [Trindade et al., 2016] são obtidos automaticamente através de um algoritmo de P&R. Apenas alguns circuitos foram comparados pois o P&R não trata portas de maioria de três entradas ficando limitado a circuitos descritos com portas *E/Ou/Inversores*. Na tabela 5.1 são comparados esses resultados aos obtidos nesse trabalho, a área é expressa em função de células USE. Embora sejam resultados para circuitos pequenos e apenas para grafos AOIG a técnica de sobreposição se mostra mais eficiente ao reduzir a área total ocupada em até 57%. Um ponto a ser ressaltado, a técnica proposta faz uma exploração de um espaço maior de soluções em comparação com a abordagem gulosa apresenta em [Trindade et al., 2016].

Os resultados apresentados em [Torres et al., 2018], mostram falta de sincronização nos sinais de entrada dos circuitos. A fim de realizar uma comparação tendo os mesmos critérios, as entradas do circuito *mux2x1* foram balanceadas manualmente. A Figura 5.3b mostra o layout final do circuito *mux2x1* mostrados em [Torres et al., 2018]. Tendo suas entradas balanceadas manualmente, o tamanho total é de 25 célu-

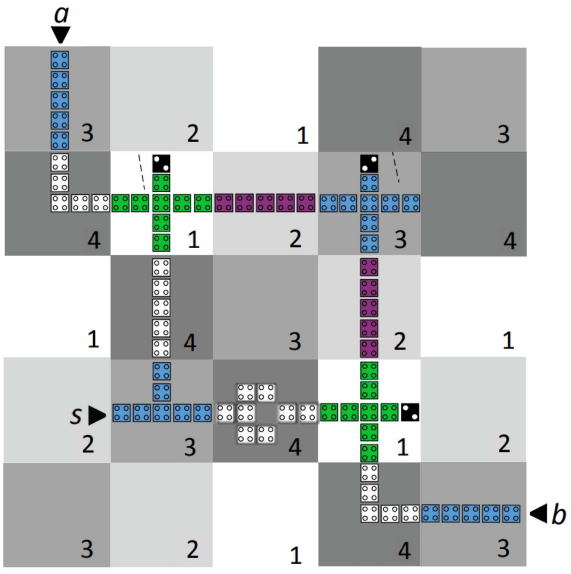
Tabela 5.1: Comparação do P&R com Partição/Sobreposição com os resultados gerados pelo P&R proposto em [Trindade et al., 2016] para circuitos $E/OU/Inversor$ sem portas Inversores.

Graph	Gates	[Trindade et al., 2016] Area(céulas USE)	Técnica proposta com sobreposição. Area(células USE)	Redução da área total.
MUX2x1	7	20	15	25%
XOR	7	28	16	43%
Full-adder 1-bit	12	28	28	0%
XNOR	9	30	24	20%
Parity Generator	13	90	48	47%
Parity Checker	19	96	60	38%

las USE. A Figura 5.3a mostra o resultado encontrado nesse trabalho para o mesmo circuito. A área total é de 15 células USE. Assim, ao se balancear as entradas no resultado mostrado em [Torres et al., 2018], esse trabalho mostra a diminuição de 40% para o circuito *mux2x1*.



(a) Layout para mux2x1 gerado com a abordagem de sobreposição/particionamento.



(b) Layout para mux2x1 apresentado em [Torres et al., 2018].

Figura 5.3

Outra forma de comparação com o trabalho apresentado em [Torres et al., 2018], consistiu em remover a restrição de sincronismo nas entradas dos circuitos QCA gerados por partição/sobreposição. Foi analisado os melhores resultados obtidos para o circuito *C17*. A Figura 5.4(a) mostra o resultado obtido nesse trabalho

e a Figura 5.4(b) o resultado obtido pela abordagem mostrada em [Torres et al., 2018]. Ambos resultados apresentam falta de sincronismo em suas entradas, a fim de garantir uma comparação justa. O resultado obtido nesse trabalho, com particionamento e sobreposição, mostra uma redução de 29% na área total ocupada em células USE.

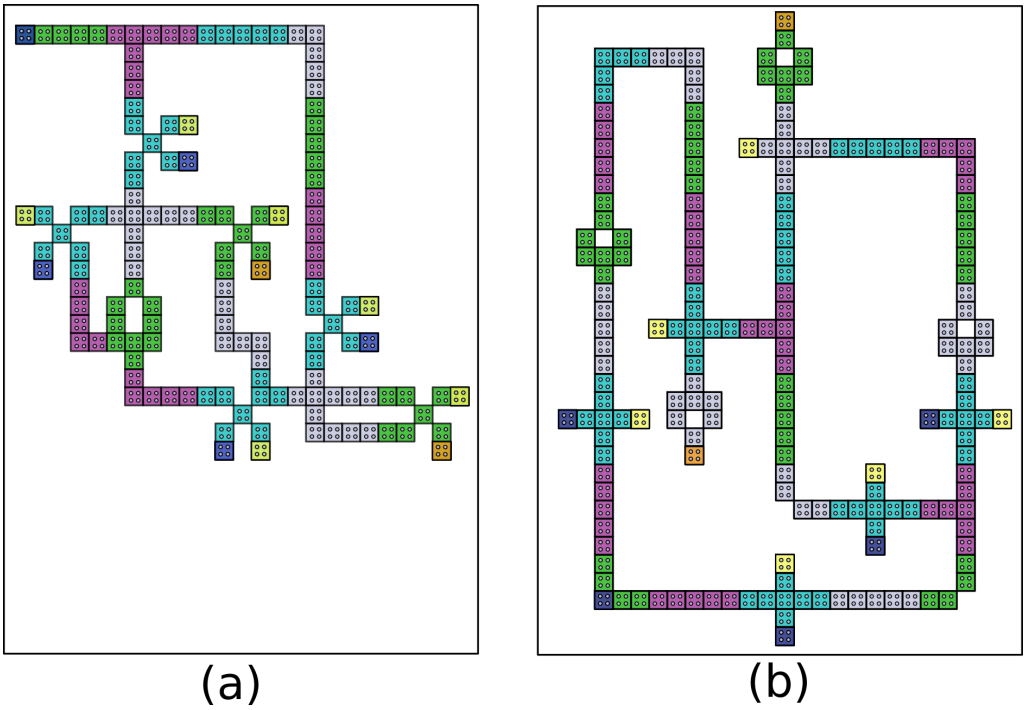


Figura 5.4: Comparação de resultados obtidos por sobreposição para o circuito *C17* com os resultados apresentados em [Torres et al., 2018] .

5.2 Impacto da Topologia na Área Final

Não só o tamanho de um circuito mas também a topologia do grafo que o descreve e seus caminhos reconvergentes tem forte influência sobre o P&R assim como na qualidade das soluções e a dificuldade em encontrá-las. Nesse trabalho isso é exemplificado no estudo de casos para um circuito Xor de 5 bits. Três variações para esse circuito são analisadas. A primeira chamada XOR_{bit} é construída manualmente usando o projeto apresentado em [Amarú et al., 2014] para o circuito Xor de 3 bits, que foi expandido usando a técnica de bitslice para fazer um XOR com 5 bits, usando dois XOR de 3 bits em cascata. A segunda implementação do circuito é o XOR_{maj} que é gerado por uma ferramenta de síntese e mapeamento com lógica limiar adaptada para capturar portas de maioria proposto em [Neutzling et al., 2017].

Por fim o XOR_{ABC} é obtido usando script de otimização aplicados a circuito Xor realizada pela ferramente de síntese ABC [Brayton & Mishchenko, 2010]. Algumas métricas para as três variações estão representadas na Tabela 5.2. Podemos observar dois fatos. Primeiro, esses resultados mostram que mesmo com número de portas 3 vezes menor a versão XOR_{bit} possui praticamente a mesma área da XOR_{ABC} e atraso próximos nas três versões. Ambas possuem um número significativo de reconvergências em relação ao número de portas. O segundo fato é o circuito Xor_{maj} que possui poucas reconvergências resultou no menor layout, com uma redução de 30% em área em comparação com as duas outras implementações.

Tabela 5.2: Variações de Layouts para Xor5

Grafo	Portas	Níveis	Caminhos reconvergentes	Área	Atraso
Xor_{bit}	11	7	10	270	21
Xor_{maj}	25	12	**8**	195	22
Xor_{ABC}	31	12	13	272	26

5.3 Avaliacao do Posicionamento e Roteamento

A tabela 5.3 mostra o resultado para um conjunto de vários circuitos, entre eles estão benckmarks retirados do MCNC(Microelectronics Center of North Carolina), evidenciando a área total e o atraso medidos em células USE juntamente com informações sobre os grafos. Através desses dados pode-se observar como os caminhos reconvergentes afetam o projeto final de um grafo. O grafo *majority* tem o número de portas próximo aos grafos *cm82a* e *b1* mas a área do seu projeto QCA é significativamente inferior, observa-se que o número de caminhos reconvergentes do grafo *majority* é também inferior aos outros dois grafos.

O circuito *parity* mostra o desempenho do algoritmo para grafos com mais de 100 portas. Esse grafo apresenta um grande número de portas e de caminhos reconvergentes. Entretanto a localidade das interconexões permite particionar todo o grafo de forma a que toda reconvergência fique encapsulada em uma partição. Ao encapsular as reconvergências, essas podem ser tratadas localmente e o espaço de solução é explorado de forma eficiente.

Tabela 5.3: Resultado a nível de porta e modelo QCA

Grafo	Portas	Níveis	Caminhos. reconvergentes	Área	Atraso
1b Maj Add	4	5	7	40	10
MUX2x1	4	4	1	15	4
1b AOI Add	7	7	7	56	12
2b Maj Add	8	6	10	98	12
b1	9	5	5	153	11
XOR	5	4	2	16	4
C17	5	5	2	48	10
XNOR	7	6	2	24	8
t	8	5	1	60	10
Full-adder 1-bit	9	7	5	56	12
Parity Generator	10	7	4	48	14
clpl	10	11	0	132	12
newtag	11	6	0	80	12
Xor_{bit}	11	7	10	270	21
majority	12	8	3	99	16
Parity Checker	15	7	6	60	14
cm82a	16	7	8	220	14
Xor_{maj}	25	12	8	195	22
Xor_{ABC}	31	12	13	272	26
parity	102	16	30	986	48

5.4 Porta de maioria de 5 entradas

A porta de maioria tem uma implementação compacta em QCA, ocupando apenas uma célula USE. Portanto, a lógica majorada fornece uma forma eficiente e compacta de se descrever um circuito lógico comparada a portas lógicas convencionais E/OU. Uma porta lógica convencional pode representar um único operador lógico binário, sendo necessárias várias portas para representar um equação lógica complexa. Entretanto, uma porta de maioria é capaz de mapear várias equações lógicas usando apenas uma célula USE. A relação entre quais equações uma porta de maioria pode mapear esta relacionada, entre outras coisas, ao número de entradas com as quais ela é construída.

O modelo de célula USE de tamanho 5x5 adotado nesse trabalho, permite o roteamento de até três fios por célula USE, possibilitando assim o uso de portas com até 6 entradas. Entretanto por questão de flexibilidade, o número de fios paralelos foi restringido ao limite de dois por célula USE, sendo possível construir portas com

até 4 entradas. Dessa forma o algoritmo é capaz de realizar o P&R para circuitos construídos com portas de maioria de 5 entradas, desde que essas se limitem ao máximo de quatro sinais de entrada. Ao se restringir o número de sinais de entrada em quatro, é preciso duplicar sinais ou fixar entradas como constantes. Isso reduz a gama de funções mapeadas por porta de maioria de 5 entradas com tal restrição. Entretanto mesmo com essa limitação, a porta de maioria de 5 entradas tende a proporcionar vários benefícios quando usada na construção de circuitos comparada ao uso de portas de maioria de 3 entradas.

O impacto causado pelo uso de portas de maioria de 3 entradas é visível em relação ao uso de portas lógicas convencionais *E*, *OU* e *inversor*. O uso da porta de maioria proporciona ganhos significativos tanto em qualidade de soluções encontradas quanto em tempo e complexidade para se encontra-las. Visto isso, foi avaliado o uso de um subconjunto restrito de portas de maioria de 5 entradas. O modelo de célula USE usado permite modelar portas de maioria com até 4 sinais de entrada, entretanto ao se replicar algum sinal de entrada dentro da porta ou se fixar uma entrada como constante, é possível trabalhar com um conjunto reduzido de portas de maioria de 5 entrada. Isso acarreta em uma diminuição no conjunto de funções lógicas que podem ser mapeadas em uma porta, mas fornece um forma inicial de se avaliar o quão efetiva pode ser a adoção de portas de maioria com mais entradas. Algumas funções lógicas com até quatro entradas(A, B, C e D) que podem ser mapeadas em uma porta de maioria 5 são mostradas na Tabela 5.4.

Tabela 5.4: Funções lógicas suportadas pela porta de maioria de 5 entradas, apresentadas em [Silva et al., 2018].

Funções lógicas	Entradas				
AB	A	B	-1	-1	1
$A+B$	A	B	-1	1	1
ABC	A	B	C	-1	-1
$A+B+C$	A	B	C	1	1
$A(B+C)$	A	A	B	C	-1
$A+BC$	A	A	B	C	1
$A(B+C+D)+BCD$	A	A	B	C	D
$A(BC+BD+CD)+BCD$	A	B	C	D	-1
$A(B+C+D)+BC+BD+CD$	A	B	C	D	1

O circuitos criados com portas de maioria de 5 entradas foram obtidas pela lógica limiar proposta em [Neutzling et al., 2017] usadas para transformar um AOIG em um MIG com suporte a portas de 5 entradas. Apenas circuitos formados por

portas de maioria que mapeiam equações lógicas compatíveis com as descritas anteriormente foram usadas. O layout de porta de maioria com 5 entradas usado foi proposto em [Silva et al., 2018], o qual pode ser visto na Figura 5.5.

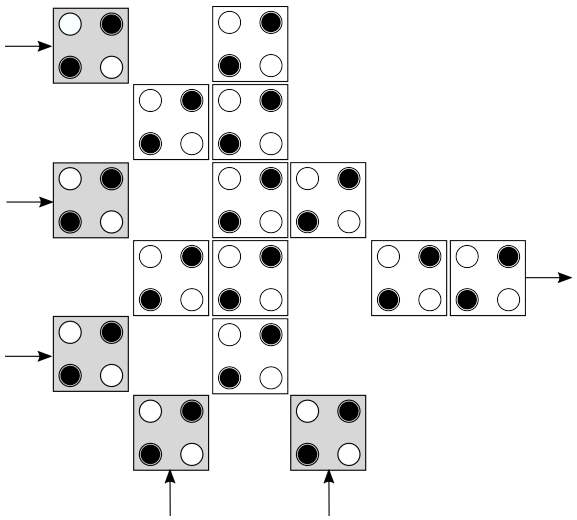
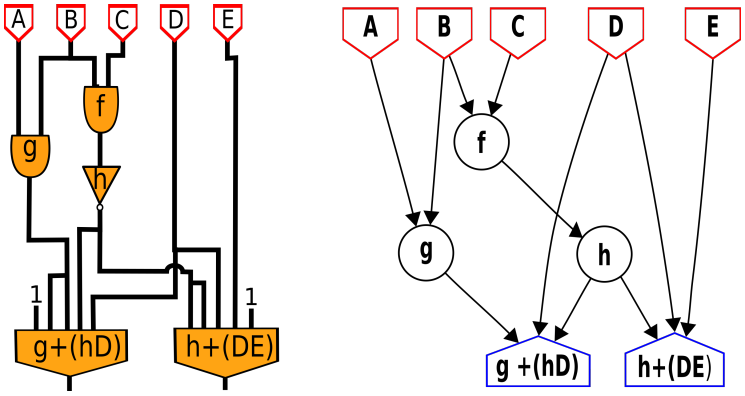


Figura 5.5: Porta de maioria de 5 entradas proposta em [Silva et al., 2018].

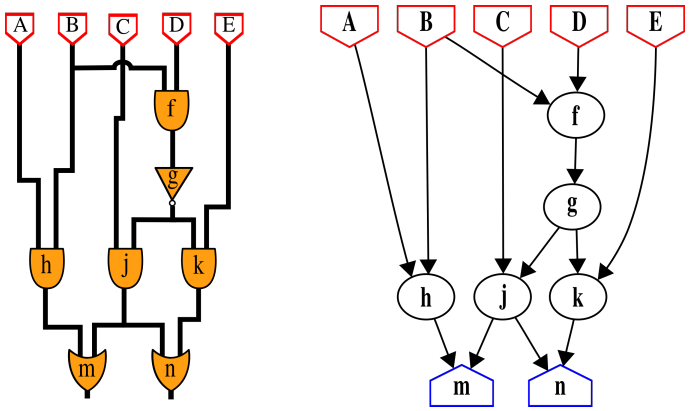
Na Figura 5.6b é mostrado o circuito *C17* criado com portas lógicas *E*, *OU* e *inversor*, juntamente com o grafo que representa o circuito. Na Figura 5.6a o circuito *C17* mostrado foi criando contendo 2 portas de maioria de 5 entradas. A equação lógica a qual elas implementam é mostrada dentro das portas de maioria. As portas de maioria de 5 entrada possuem replicação em um sinal de entrada e fixação de outro sinal de entrada no valor constante 1, como pode ser visto pelo seu circuito.

O processo de P&R é exatamente o mesmo usado para MIG de 3 entradas. A partir da solução encontrada o layout da porta de maioria é mudado, quando necessário, e as entradas são rearranjadas para se adaptarem ao novo layout da porta de maioria com 5 entradas. O resultado pode ser visto na Figura 5.7a, a qual mostra um resultado de P&R para o circuito *C17*, evidenciando em vermelho as 2 portas de maioria de 5 entradas contidas no circuito. Na Figura 5.7b pode-se ver o resultado para o mesmo circuito construído apenas com portas de maioria com 3 entradas.

Um mesmo circuito pode apresentar uma redução no número de portas, fios, níveis, reconvergências entre outras características, quando construído com portas de maioria, em comparação ao mesmo construído apenas com portas lógicas. Esse resultado é mostrado na Tabela 5.5. O mesmo ocorre quando comparadas portas de maioria com número maior e menor de entradas. Essa redução impacta o processo



(a) Circuito e grafo *C17* contendo portas de maioria de 5 entradas.

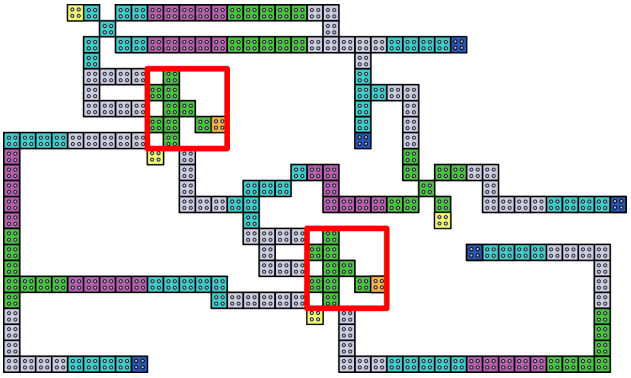


(b) Circuito e grafo *C17* contendo portas de maioria de 3 entradas, usadas para mapear portas lógicas E/OU .

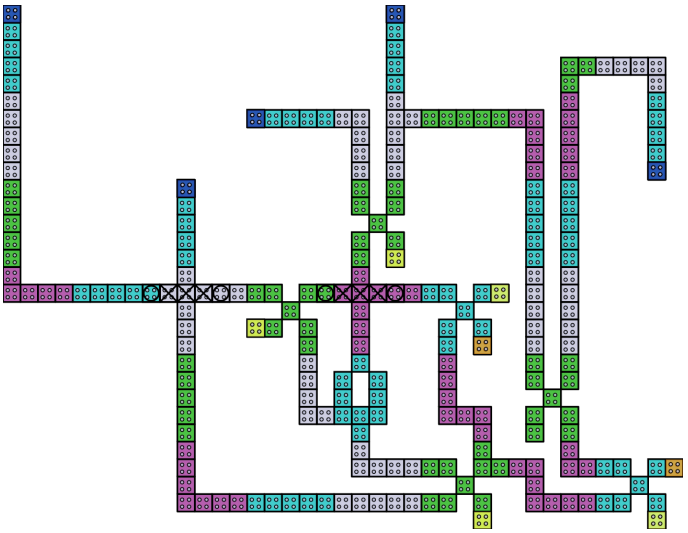
Figura 5.6: Circuito e grafo *C17*.

de P&R diretamente, influenciando na qualidade das soluções encontradas e simplificando o processo de busca. Para mostrar o impacto do uso da porta de maioria com 5 entradas na construção de um circuito, são comparados os melhores resultados obtidos pelo algoritmo de P&R para o circuito *clpl*. O mesmo circuito é construído usando-se portas de maioria com 3 e 5 entradas, sendo criados pela lógica de limiar apresentada em [Neutzling et al., 2017], o que permite obter ambos a partir de um AIOG do circuito *cplp*. Na versão construída com portas de maioria de 3 entradas, a lógica limiar não é capaz de capturar nenhuma porta de maioria para esse circuito. Portanto as portas de maioria são usadas para criar portas lógicas "E"e "OU". Na versão de 5 entradas a lógica limiar foi capaz de mapear todas portas do circuito o que resulta no circuito formado apenas por portas de maioria com 5 entradas.

Nesse circuito as portas de maioria mapeiam funções lógicas replicando um sinal de entrada e fixando uma entrada constante, logo apenas três sinais de entradas



(a) Layout QCA para circuito *C17* usando porta de maioria de 5 entradas.



(b) Layout QCA para circuito *C17* usando porta de maioria de 3 entradas.

Figura 5.7: Layouts QCA para circuito *C17*.

Tabela 5.5: Resultado a nível de porta e modelo QCA

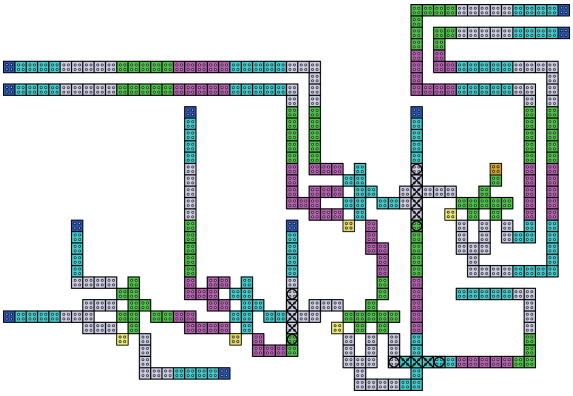
	Porta de maioria de 3 entradas (usando apenas 2 entradas)	Porta de maioria de 5 entradas (usando apenas 3 entradas)	Redução
Portas	10	5	50%
Fios	20	15	25%
Níveis no grafo	11	5	54%
Área de P&R	132	70	47%

são necessários. Dados 3 sinais de entrada A, B e C para uma porta *MAJ5*, a função lógica mapeada pela porta *MAJ5*, no circuito *clpl*, é obtida fixando uma entrada em

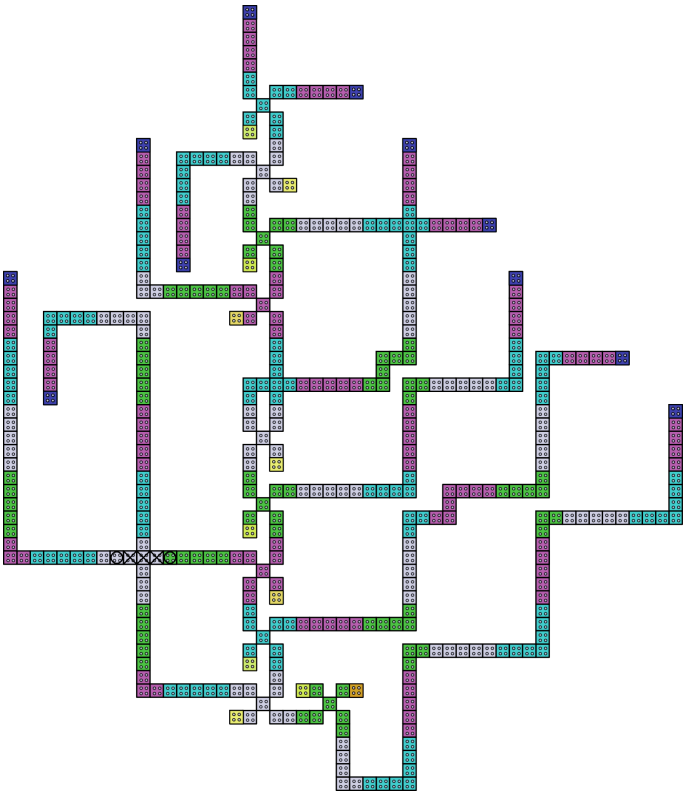
1 e replicando o sinal da porta A.

$$MAJ5 = A + BC$$

Na Figura 5.8 pode-se ver a impacto causado pelo uso da porta de maioria com 5 entradas, mostrada na Figura 5.8a, em relação a portas de maioria de 3 entradas usadas para mapear portas lógicas "E"e "OU", mostrada na Figura 5.8b. O uso de portas de maioria de 5 entradas causa uma redução na área total da solução de 47% em comparação a melhor solução encontrada para o grafo construído com portas de 3 entradas.



(a) Projeto QCA construido com portas de maioria de 5 entradas.



(b) Projeto QCA construido com portas de maioria de 3 entradas.

Figura 5.8: Comparação de layout para o circuito *clpl* usando portas de maioria de 3 e 5 entradas.

5.5 Poda de soluções

A poda de soluções é feita sobre restrições criadas sobre reconvergências durante a sobreposição. Devido ao elevado número de soluções parciais geradas, torna-se inviável analisá-las uma a uma. Por esse motivo é preciso limitar o tamanho do espaço de busca. Na árvore de busca esse processo equivale a analisar apenas uma parte dos seus ramos ignorando os demais. Logo é preciso eliminar soluções parciais as quais se tem certeza que não irão formar soluções completas, evitando que a partir delas outras soluções parciais sejam criadas. Nesse trabalho isso é feito através da poda feita sobre restrições identificadas sobre a reconvergência.

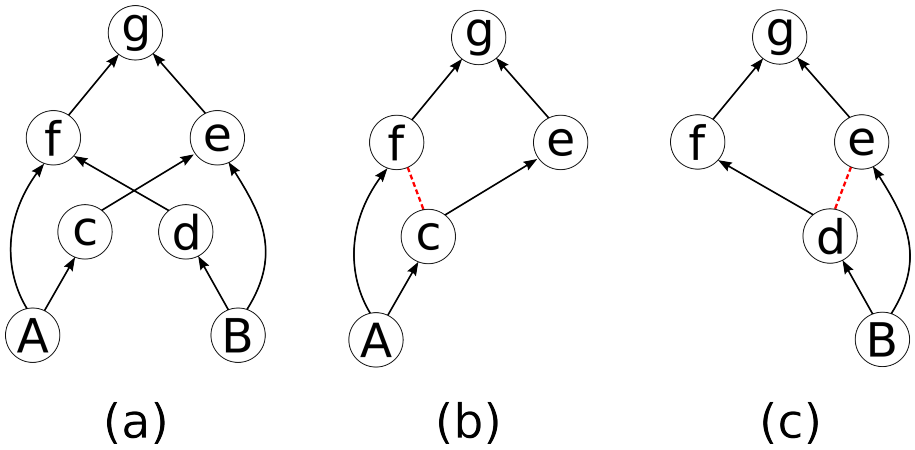


Figura 5.9: Reconvergências no grafo do circuito *Xor*.

Tendo como exemplo o grafo do circuito *Xor* mostrado na Figura 5.9a. Nesse grafo existem 2 reconvergências mostradas nas Figura 5.9b e 5.9c, onde suas restrições, usadas na poda, estão representadas pela linha pontilhada em vermelho. As restrições são criadas pela separação das reconvergências em partições distintas o que leva à perda de informações importantes para o processo de P&R. Na Figura 5.10 são mostradas as 3 partições criadas sobre o grafo do circuito *Xor*.

As podas são feitas sobre os resultados encontrados para a partição 1 da Figura 5.10. Essa partição possui duas restrições sobre a distância na matriz USE entre os nós c e $f(d_{c,f})$ e sobre os nós d e e $(d_{d,e})$. A inversão no sentido das distancias $d_{A,c}$ e $d_{B,d}$ para $d_{c,A}$ e $d_{d,B}$ respectivamente podem causar uma diferença de até 3 unidades, para compensar tal diferença é somado a contante 3 a cada restrição, as quais são mostradas a seguir:

$$(d_{c,f}) < (d_{c,A}) + 3 + (d_{A,f})$$

$$(d_{d,e}) < (d_{d,B}) + 3 + (d_{B,e})$$

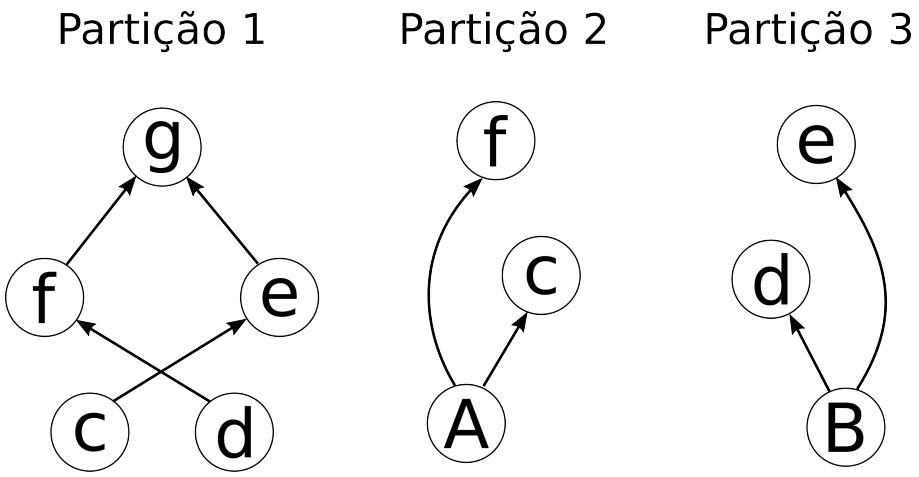


Figura 5.10: partições feitas sobre o grafo do circuito *Xor*.

Do conjunto de soluções obtidas para partição 1, se alguma não atende a pelo menos uma entre as duas restrições, então ela é podada. Desse forma a propagação criada pela sobreposição dessa solução as demais é interrompida, diminuindo assim o número de soluções para partição 1 que não irão gerar um resultado completo. Pelo fato de parte das restrições estarem em função da distância estipulada pelo vetor de pesos, o número de soluções eliminadas pode variar em função do mesmo.

Na Tabela 5.6 são apresentadas algumas variações de valores para o vetor de pesos. Para cada variação são mostradas o número de soluções analisadas com e sem a poda. Em ambos os casos o número de soluções completas encontradas para o grafo do circuito *Xor* foram as mesmas, sendo o valor mostrado na Tabela 5.6 referentes ao número de soluções parciais analisadas e sobrepostas. Conclui-se assim que foi possível eliminar até 36% das soluções sem ocasionar perda alguma no resultado final.

Tabela 5.6: Resultados da poda por reconvergência no grafo do circuito *Xor*

	Nº de soluções intermediárias		
Vetor de pesos	Sem poda	Com poda	Redução
4,3,2	61751	57953	6,15%
4,3,1	37452	23964	36%
3,3,1	9487	6485	31,6%
4,2,1	4404	3309	24,9%
4,4,2	161214	131062	18,7%

Capítulo 6

Conclusão

O paradigma QCA é um dos candidatos a substituir as tecnologias baseadas em silício, entretanto ainda faltam muitas ferramentas para construção e síntese automatizada de circuitos QCA. Esse trabalho apresenta uma nova abordagem na geração automática de layout para QCA, analisando o impacto de caminhos reconvergentes na etapa de posicionamento e roteamento, explorando um conjunto mais amplo de soluções em relação aos trabalhos anteriores além de introduzir o particionamento e sobreposição para reduzir a complexidade do processo de busca e validação de soluções. A adoção do esquema de clock USE garante a escalabilidade e padronização na criação de layout QCA. O USE já foi validado a nível de simulação, o que simplifica a validação dos resultados gerados, uma vez sendo diretamente transformados para serem simulados pela ferramenta QCADesigner.

A nova abordagem proposta nesta dissertação introduz o particionamento e sobreposição de soluções de subgrafos, onde a análise de caminhos reconvergentes permite a exploração mais eficiente do espaço de projetos. Um estudo de casos realizado mostra o impacto das reconvergencias sobre o processo de P&R e como essa informação pode ser usada para podar soluções. Os layouts apresentados nesse trabalho reduzem, em média, 2 vezes a área gasta quando comparados aos layouts construídos manualmente ou de forma automática pelas abordagens anteriores baseadas no esquema de clock USE. O fluxo de execução é quase totalmente automatizado o que reduz o esforço manual na síntese. Entretanto, o arquivo de configuração com os espaçamentos e a exploração de várias partições diferentes são pontos que podem ser melhorados. Além disso é necessário explorar as simetrias para reduzir ainda mais o espaço de busca com técnicas mais eficientes de poda.

Outra contribuição deste trabalho é o uso de um subconjunto de funções da

porta de maioria de 5 entradas limitadas a funções com 3 variáveis. Este ponto pode ser melhor explorado para reduzir a complexidade no nível de síntese, etapa anterior ao P&R. Utilizar a porta de maioria com 5 entradas é também possível com alterações no roteamento para gerar o acoplamento com as entradas das portas respeitando as restrições do USE.

Resumindo, trabalhos futuros poderão adicionar suporte a circuitos sequenciais assim como máquina finita de estados para aproveitar os caminhos bi-direcionais que o USE proporciona. Como já mencionado, estender o suporte a portas de maioria com mais entradas, visto que atualmente está restrito a portas de maioria de 3 entradas, o que possivelmente garantirá uma maior aproveitamento e melhor qualidade no projeto dos circuitos. O particionamento poderá ser feito dinamicamente tornando o algoritmo mais flexível na busca de soluções e com isso automatizando por completo todo o fluxo de trabalho.

Apêndice A

Projetos de circuitos QCA

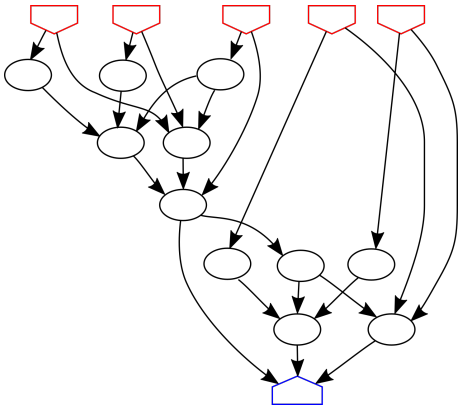


Figura A.1: Grafo do circuito XOR_{bit}.

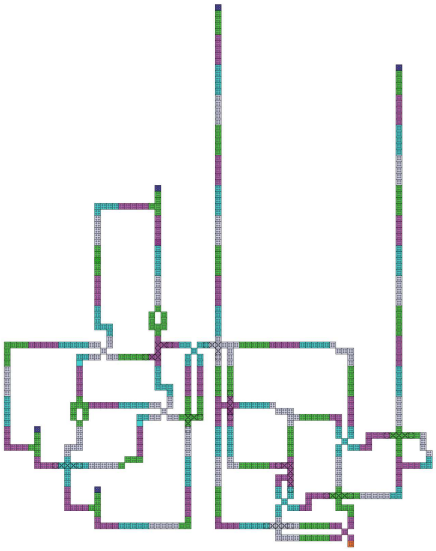


Figura A.2: Projeto QCA para o circuito XOR_{bit}.

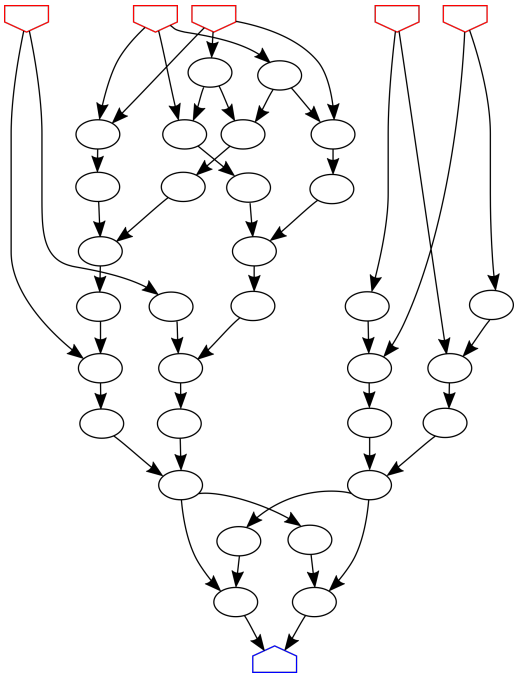


Figura A.3: Grafo do circuito XOR_{ABC}.

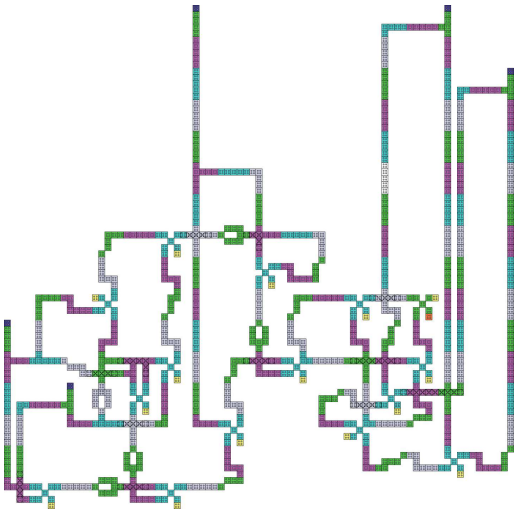


Figura A.4: Projeto QCA para o circuito XOR_{ABC}.

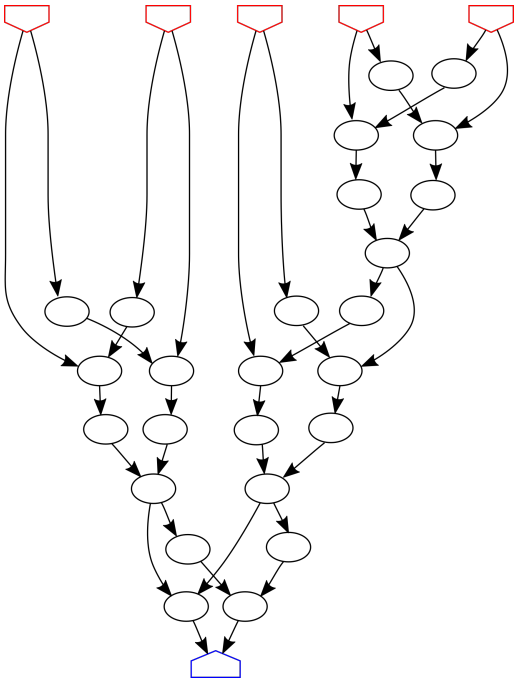


Figura A.5: Grafo do circuito XOR_{maj}.

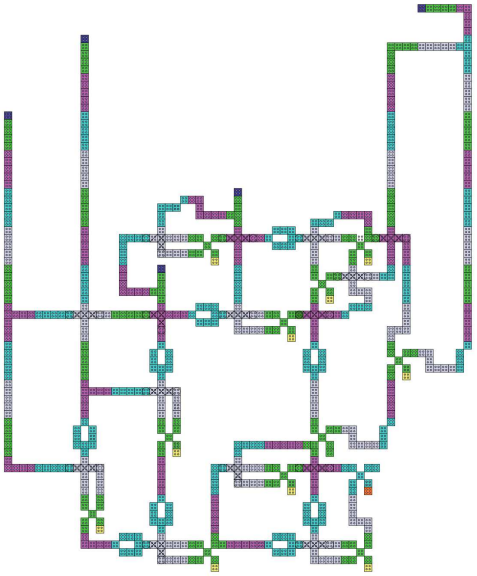


Figura A.6: Projeto QCA para o circuito XOR_{maj}.

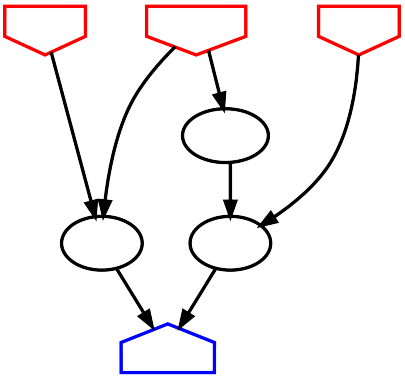


Figura A.7: Grafo do circuito *MUX2x1*.

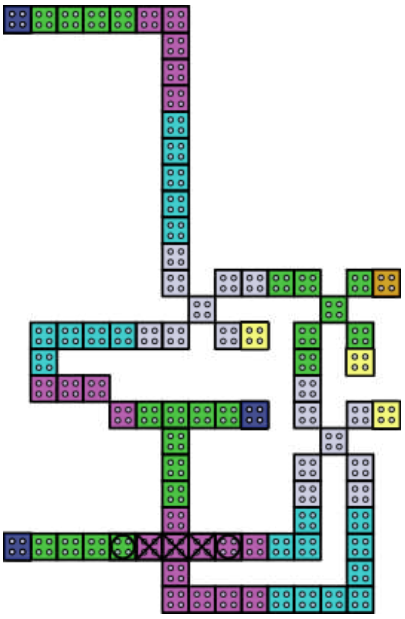


Figura A.8: Projeto QCA para o circuito *MUX2x1*.

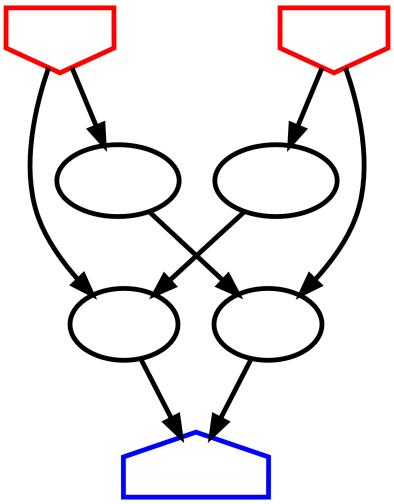


Figura A.9: Grafo do circuito *XOR*.

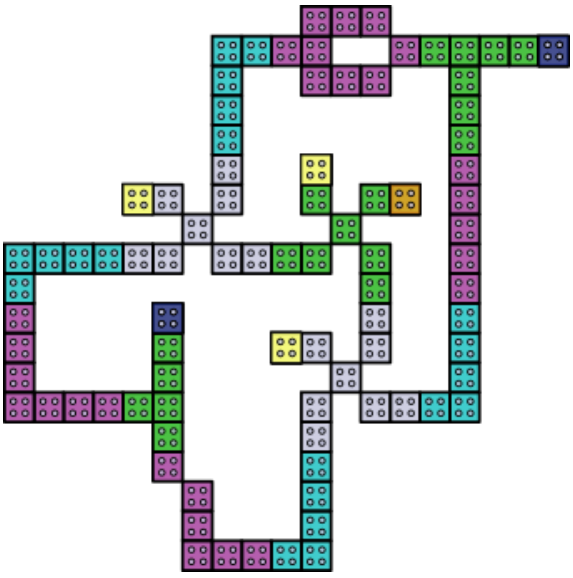


Figura A.10: Projeto QCA para o circuito *XOR*.

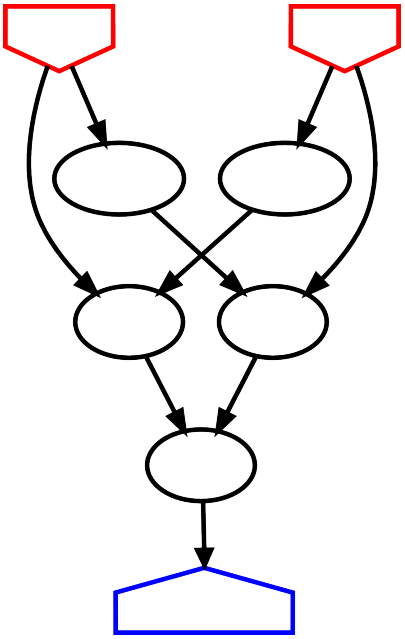


Figura A.11: Grafo do circuito *XNOR*.

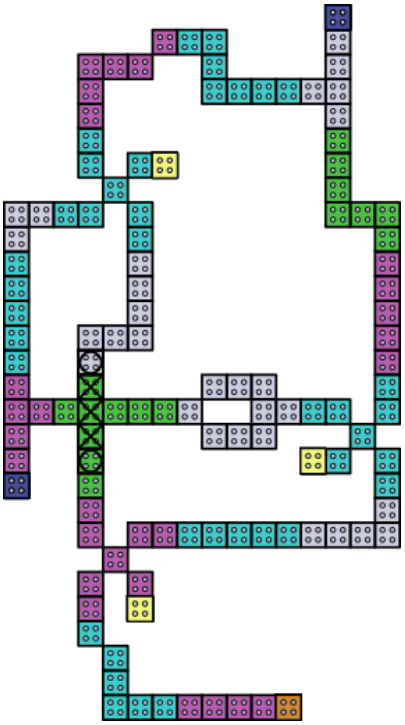


Figura A.12: Projeto QCA para o circuito *XNOR*.

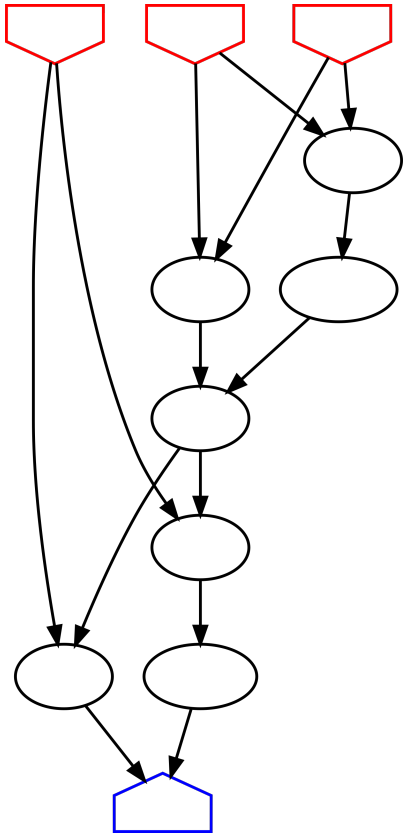


Figura A.13: Grafo do circuito *Full-adder 1-bit*.

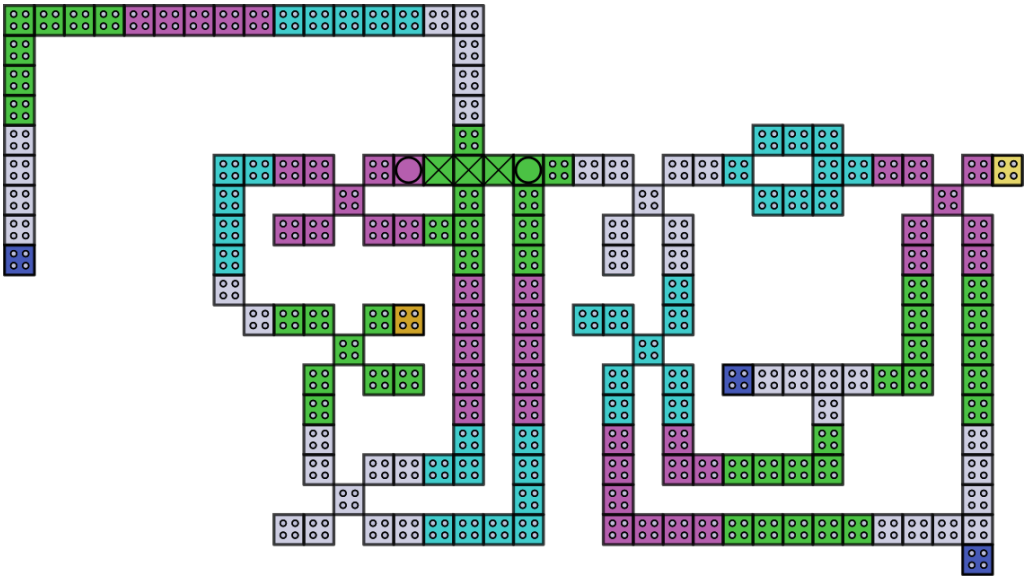


Figura A.14: Projeto QCA para o circuito *Full-adder 1-bit*.

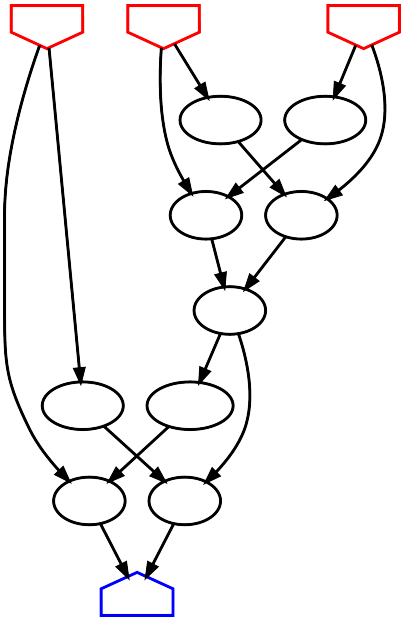


Figura A.15: Grafo do circuito *Parity Generator*.

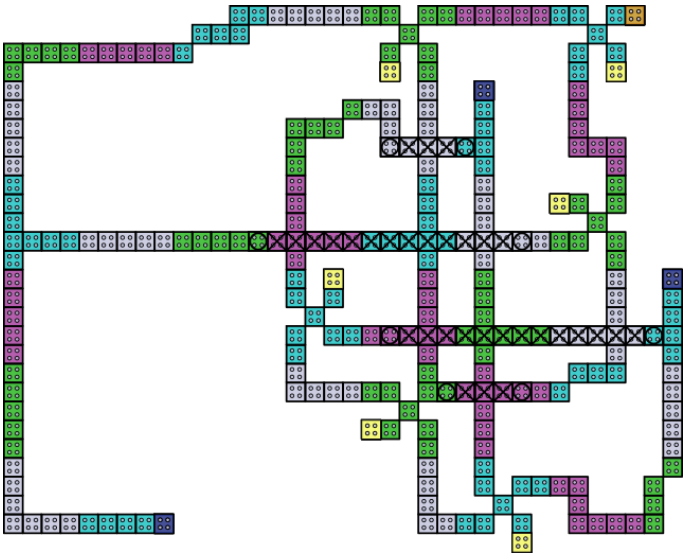


Figura A.16: Projeto QCA para o circuito *Parity Generator*.

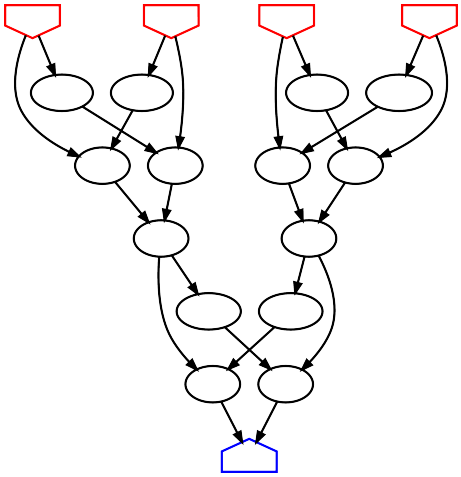


Figura A.17: Grafo do circuito *Parity Checker*.

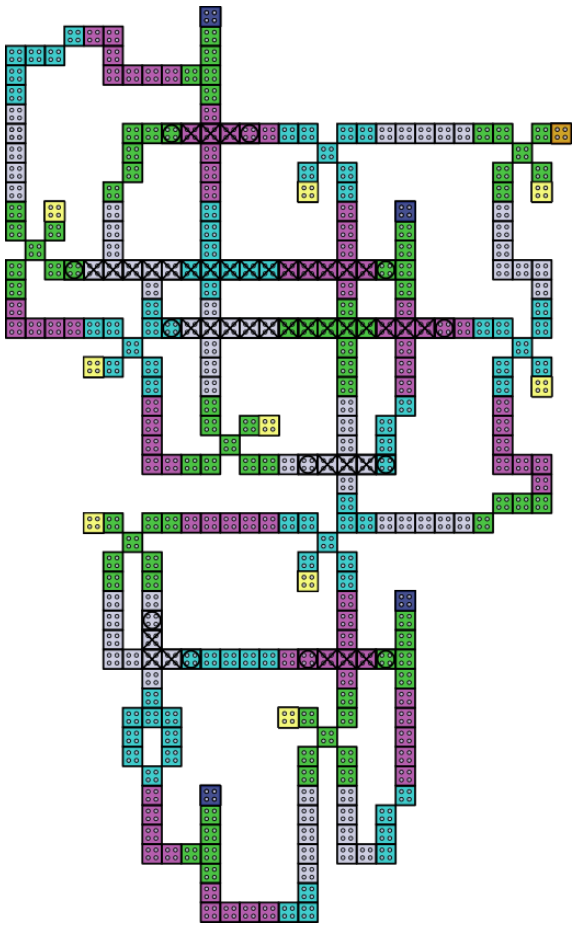


Figura A.18: Projeto QCA para o circuito *Parity Checker*.

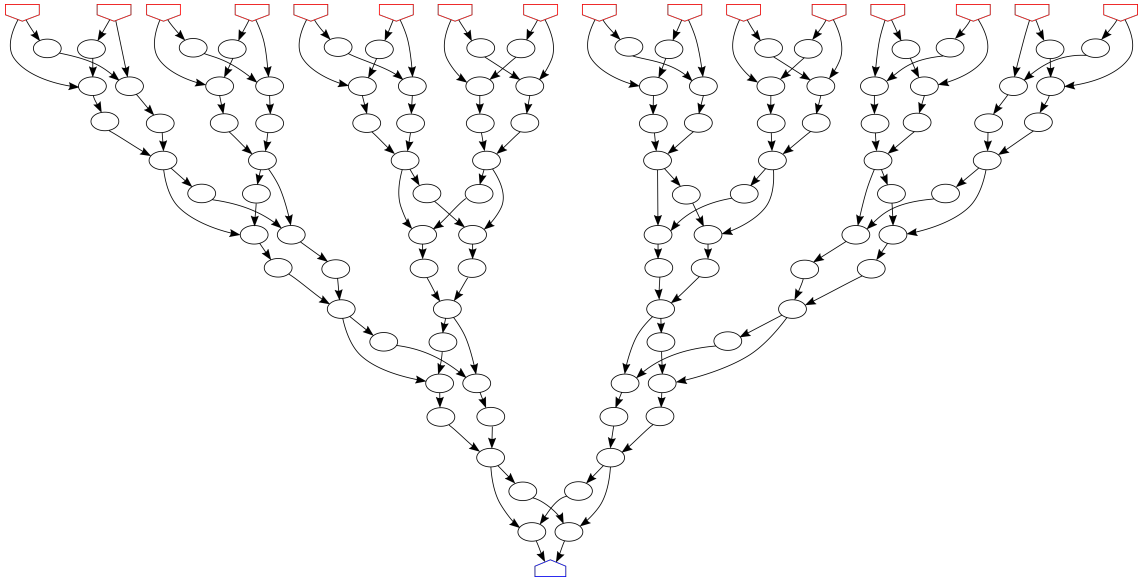


Figura A.19: Grafo do cricuito Parity.

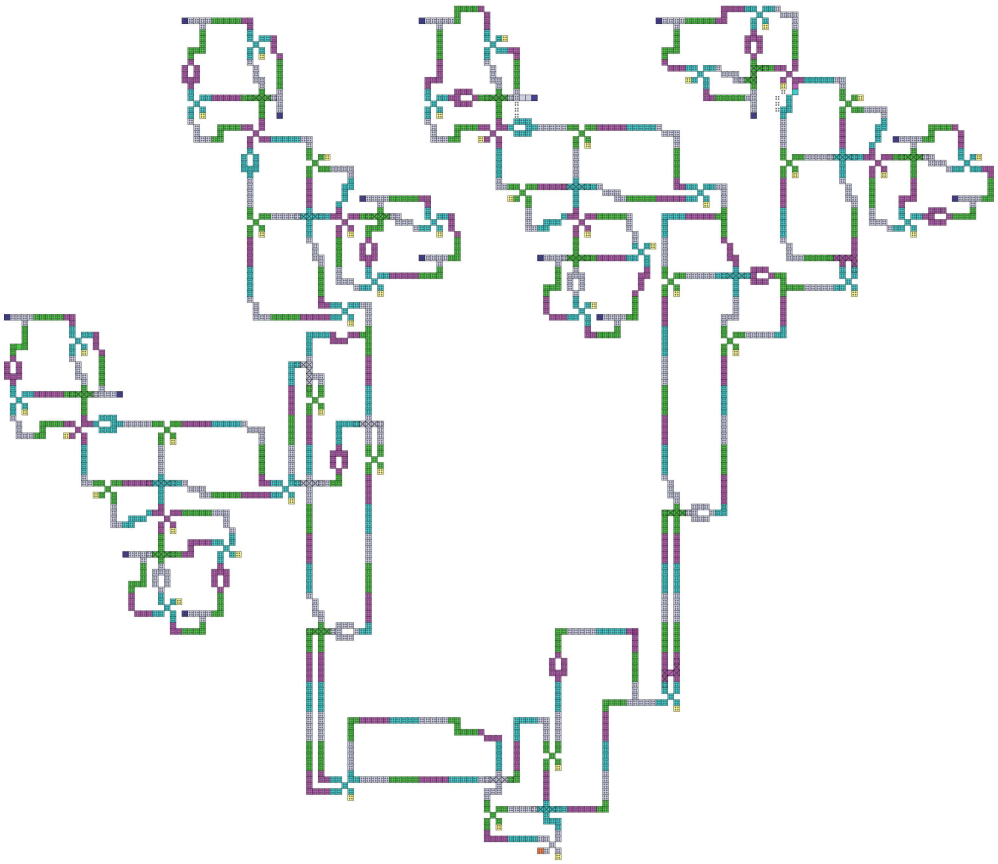


Figura A.20: Design QCA para o cricuito Parity.

Referências Bibliográficas

Amarú, L.; Gaillardon, P.-E. & De Micheli, G. (2014). Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization. In *Proceedings of the 51st Annual Design Automation Conference*, pp. 1--6. ACM.

Bhanja, S.; Ottavi, M.; Lombardi, F. & Pontarelli, S. (2006). Novel designs for thermally robust coplanar crossing in qca. In *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pp. 786--791. European Design and Automation Association.

Brayton, R. & Mishchenko, A. (2010). Abc: An academic industrial-strength verification tool. In *Computer Aided Verification*, pp. 24--40. Springer.

Bubna, M.; Roy, S.; Shenoy, N. & Mazumdar, S. (2008). A layout-aware physical design method for constructing feasible qca circuits. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pp. 243--248. ACM.

Campos, C. A. T.; Marciano, A. L.; Neto, O. P. V. & Torres, F. S. (2016). Use: a universal, scalable, and efficient clocking scheme for qca. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):513--517.

Donath, W. E. (1980). Complexity theory and design automation. In *Proceedings of the 17th Design Automation Conference*, pp. 412--419. ACM.

Graunke, C. R.; Wheeler, D. I.; Tougaw, D. & Will, J. D. (2005). Implementation of a crossbar network using quantum-dot cellular automata. *IEEE Transactions on Nanotechnology*, 4(4):435--440.

Henderson, S. C.; Johnson, E. W.; Janulis, J. R. & Tougaw, P. D. (2004). Incorporating standard cmos design process methodologies into the qca logic design process. *IEEE Transactions on nanotechnology*, 3(1):2--9.

Kim, K.; Wu, K. & Karri, R. (2006). Quantum-dot cellular automata design guideline. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 89(6):1607--1614.

Lent, C. S. & Tougaw, P. D. (1997). A device architecture for computing with quantum dots. *Proceedings of the IEEE*, 85(4):541--557.

Lin, T.-H.; Banerjee, P. & Chang, Y.-W. (2013). An efficient and effective analytical placer for fpgas. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13. ACM.

Luu, J.; Kuon, I.; Jamieson, P.; Campbell, T.; Ye, A.; Fang, W. M.; Kent, K. & Rose, J. (2011). Vpr 5.0: Fpga cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling. *ACM Trans. Reconfigurable Technol. Syst.*, 4(4).

Maidee, P.; Ababei, C. & Bazargan, K. (2005). Timing-driven partitioning-based placement for island style fpgas. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):395–406.

Moore, G. E. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82--85.

Neutzling, A.; Martins, M. G.; Callegaro, V.; Reis, A. I. & Ribas, R. P. (2017). A simple and effective heuristic method for threshold logic identification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Niemier, M. T. (2000). *Designing digital systems in quantum cellular automata*. PhD thesis, University of Notre Dame.

Orlov, A.; Amlani, I.; Bernstein, G.; Lent, C. & Snider, G. (1997). Realization of a functional cell for quantum-dot cellular automata. *Science*, 277(5328):928--930.

Patitz, Z. D. (2006). *Fault Tolerant Quantum-dot Cellular Automata Majority Gate Design*. PhD thesis, Oklahoma State University.

Ravichandran, R.; Ladiwala, N.; Nguyen, J.; Niemier, M. & Lim, S. K. (2004). Automatic cell placement for quantum-dot cellular automata. In *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, pp. 332--337. ACM.

Reis, D. A.; Campos, C. A. T.; Soares, T. R. B.; Neto, O. P. V. & Torres, F. S. (2016). A methodology for standard cell design for qca. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pp. 2114--2117. IEEE.

Shin, S.-H.; Jeon, J.-C. & Yoo, K.-Y. (2013). Wire-crossing technique on quantum-dot cellular automata. In *NGCIT2013, the 2nd International Conference on Next Generation Computer and Information Technology*, volume 27, pp. 52--57.

Silva, P. A. R. L.; Alves, J. R. S. B.; Ferreira, R. S.; Neto, O. P. V. & Nacif, J. A. M. (2018). A novel five-input multiple-function qca threshold gates. *IEEE International Symposium on Circuits and Systems, submitted.*

Teodósio, T. & Sousa, L. (2007). Qca-lg: A tool for the automatic layout generation of qca combinational circuits. In *Norchip, 2007*, pp. 1--5. IEEE.

Torres, F. S.; Wille, R.; Niemann, P. & Drechsler, R. (2018). An energy-aware model for the logic synthesis of quantum-dot cellular automata. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

Tougaw, P. D. & Lent, C. S. (1994). Logical devices implemented using quantum cellular automata. *Journal of Applied physics*, 75(3):1818--1825.

Trindade, A.; Ferreira, R.; Nacif, J. A. M.; Sales, D. & Neto, O. P. V. (2016). A placement and routing algorithm for quantum-dot cellular automata. In *Integrated Circuits and Systems Design (SBCCI), 2016 29th Symposium on*, pp. 1--6. IEEE.

Vankamamidi, V.; Ottavi, M. & Lombardi, F. (2008). Two-dimensional schemes for clocking/timing of qca circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):34--44.

Walus, K.; Dysart, T. J.; Jullien, G. A. & Budiman, R. A. (2004). Qcadesigner: A rapid design and simulation tool for quantum-dot cellular automata. *IEEE transactions on nanotechnology*, 3(1):26--31.

Walus, K.; Jullien, G. & Dimitrov, V. (2003). Computer arithmetic structures for quantum cellular automata. In *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on*, volume 2, pp. 1435--1439. IEEE.