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

JORGE ALBERTO FERREIRA

Método para Legalização de Circuitos com Células de Altura Múltipla

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Microeletrônica

Orientador: Prof. Dr. Paulo Francisco Butzen
Co-orientador: Prof. Dr. Ricardo Reis

Porto Alegre
2022

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Ferreira, Jorge Alberto

Método para Legalização de Circuitos com Células de Altura Múltipla / Jorge Alberto Ferreira. – Porto Alegre: PGMICRO da UFRGS, 2022.

84 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2022. Orientador: Paulo Francisco Butzen; Co-orientador: Ricardo Reis.

1. Microeletrônica. 2. VLSI. 3. Síntese Física. 4. Posicionamento. 5. Legalização. 6. Algoritmos de legalização. I. Butzen, Paulo Francisco. II. Reis, Ricardo. III. Título.

“The world ain’t all sunshine and rainbows.
It’s a very mean and nasty place and I don’t care how tough you are it will beat
you to your knees and keep you there permanently if you let it.
You, me, or nobody is gonna hit as hard as life.
But it ain’t about how hard ya hit.
It’s about how hard you can get hit and keep moving forward.
How much you can take and keep moving forward.
That’s how winning is done!”

— ROCKY BALBOA

AGRADECIMENTOS

Este trabalho foi desenvolvido graças à bolsa concedida pela CAPES, a quem dedico aqui os meus sinceros agradecimentos.

Gostaria de dedicar especial agradecimento a minha mãe, Irlene Ferreira, pelo incentivo ao estudo, e a minha namorada, Caroline Godois Destri, por saber compreender a minha ausência e meu desgaste durante o mestrado.

Ao meu orientador, Paulo Butzen, agradeço pela confiança, disponibilidade para orientação, críticas, sugestões, paciência e, principalmente, amizade e carinho. Muito mais que orientador de graduação e mestrado, foi e será um amigo que pretendo levar para a vida.

Ao meu co-orientador, Prof. Ricardo Reis, e ao Instituto de Informática por receberam e oportunizarem uma excelente infraestrutura com uso de ferramentas, laboratórios, eventos e pelos colegas que fiz neste período.

Aos meus avós que tenho a certeza que estão orgulhosos de mim onde eles estiverem. Aos meus queridos amigos e professores do Grupo de Sistemas Digitais e Embarcados (GSDE - FURG), onde começou o meu interesse pela microeletrônica. Em especial a professora Cristina Meinhardt, que foi minha primeira orientadora dentro de um projeto, agradeço pelas dicas, amizade, aprendizado, apoio e, principalmente, os churrascos. Agradeço também aos meus colegas da graduação da Engenharia de Computação da FURG.

A todas as pessoas que ajudaram e incentivaram na concretização do trabalho, em especial a família da minha namorada, a minha “ermã"Nicole San Martin Soares e sua família e, sem dúvidas, ao meu psicólogo, Mateus Pavei, os meus sinceros agradecimentos.

RESUMO

Desde a década de 1970, novas tecnologias de semicondutores impactam nossa sociedade. Desde então, o número de componentes num mesmo circuito é dobrado a cada dois anos, seguindo a Lei de Moore. Com esse avanço, os microprocessadores atuais possuem bilhões de transistores nos seus circuitos. Porém, esses avanços impuseram regras de projeto que trouxeram novos desafios para as etapas de otimização. Para auxiliar nesses obstáculos foi preciso utilizar *softwares* de EDA (do inglês *Eletronic Design Automation*). Hoje em dia, as ferramentas EDA são usadas em projetos de fluxo de células padrão desde seus estágios iniciais até finais. O fluxo de células padrão é composto por uma sequência de fluxos para elaborar e sintetizar o circuito. Dentre estes fluxo está o fluxo da síntese, onde está a etapa de posicionamento. Uma das etapas do posicionamento é a legalização cujo objetivo é mover as células para posições válidas e remover suas sobreposições. A legalização possui desafios como o número de células, células de altura mista, roteabilidade, comprimento de fio, regras de projeto complexas como regiões de *fence*. Os métodos de legalização são categorizados em heurísticos e analíticos para enfrentar esses desafios. Neste trabalho, é proposto um método heurístico de legalização de células de altura mista. Este trabalho foi baseado em etapas de trabalhos existentes na literatura. Além disso, nossa legalização é aplicada incrementalmente através de etapas que priorizam as células que violam suas regiões, após isso prioriza grupos de células sobrepostas. Os experimentos realizados mostram que nosso método proposto permite reduzir mais do que 15% o tempo de execução e a diferença nos resultados é mínima em comprimento de fio.

Palavras-chave: Microeletrônica. VLSI. Síntese Física. Posicionamento. Legalização. Algoritmos de legalização.

ABSTRACT

Since the 1970s, new semiconductor technologies have impacted our society. After that, the number of components in the same circuit is doubled every two years, following Moore's Law. Today's microprocessors have billions of transistors in their circuits with this advancement. However, these advances imposed design rules that brought new challenges to the optimization steps. EDA (Electronic Design Automation) software was necessary to help with these obstacles. EDA tools use in standard cell flow designs from their initial to final stages. The standard cell flow comprises a sequence of flows to elaborate and synthesize the circuit. Among these flows is the physical synthesis flow, which is the placement step. One of the placement steps is legalization which aims to move cells to proper positions and remove their overlaps. Legalization has challenges like the number of cells, mixed-cell height, routability, wirelength, and complex design rules like fence regions. Legalization methods are categorized into heuristic and analytical to address these challenges. This work proposes a heuristic method of legalization of mixed-height cells. This work is based on steps of existing works in the literature. Additionally, our legalization is applied incrementally through steps that prioritize cells that violate their regions, then prioritize groups of overlapping cells. The experiments show that our proposed method allows us to reduce the execution time by more than 15%, and the difference in the results is minimal in terms of wirelength.

Keywords: Microelectronics, VLSI, Physical Synthesis, Placement, Legalization, Legalization Algorithms.

LISTA DE ABREVIATURAS E SIGLAS

BJT	*Bipolar Junction Transistor*
CAD	*Computer Aided Design*
CPU	*Central Processing Unit*
CNN	*Convolutional Neural Network*
CTS	*Clock Tree Synthesis*
DAG	*Directed Acyclic Graph*
DRC	*Design Rule Checking*
EDA	*Eletronic Design Automation*
FET	*Field-Effect Transistor*
GND	*Ground*
GPU	*Graphics Processing Unit*
HDLs	*Hardware Description Language*
HPWL	*Half-Perimeter Wirelength*
ICCAD	*International Conference On Computer Aided Design*
IGBT	*Insulated Gate Bipolar Transistor*
ILP	*Integer Linear Programming*
IP	*Intellectual Property*
ISPD	*International Symposium on Physical Design*
LCP	*Linear Complementarity Problem*
LP	*Linear Programming*
LVS	*Layout vs. Schematic*
M1	Metal 1
M2	Metal 2
MGL	*Multi-Row Global Legalization*

MMSIM	*Modulus-Based Matrix Splitting Iteration Methods*
MOSFET	*Metal–Oxide–Semiconductor Field-Effect Transistor*
NBLG	*Negotiation-Based Legalizer*
QP	*Quadratic Programming*
RMMSIM	*Robust Modulus-Based Matrix Splitting Iteration Methods*
RMST	*Rectilinear Minimum Spanning Tree*
RSMT	*Rectilinear Steiner Minimal Tree*
RTL	*Register-Transfer Level*
VDD	*Power*
VHDL	*VHSIC Hardware Description Language*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

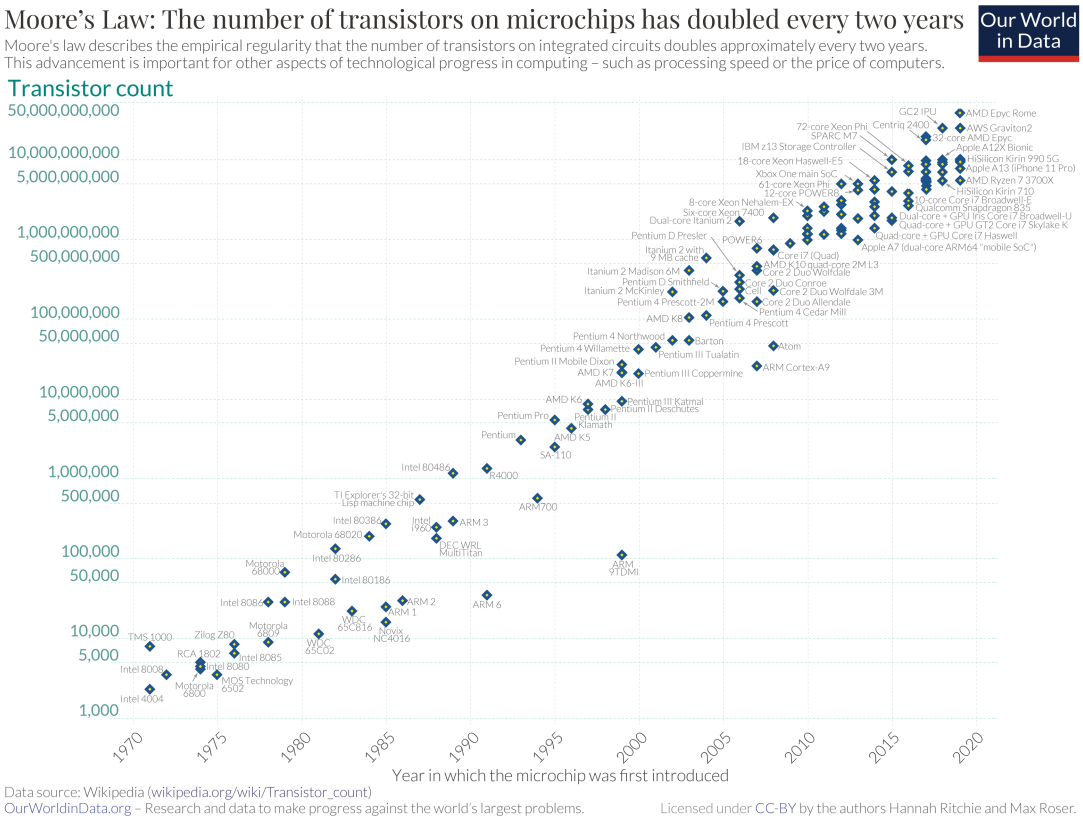
Desde os anos 50 acontece uma mudança na indústria que saiu do mecânico para o eletrônico. Essa mudança, é devido à invenção do transistor que foi inventado em 1947. Um dispositivo semicondutor que pode ser usado como uma chave, permitindo computação rápida e não mecânica. Os transistores podem ser categorizados em três categorias: BJT (*bipolar junction transistor*), FET (*field-effect transistor*) e IGBT (*Insulated Gate Bipolar Transistor*).

No início dos anos 60 foi inventado o MOSFET (*metal–oxide–semiconductor field-effect transistor*) que possibilitou a construção dos circuitos integrados. Mas foi no final dos anos 60 e início dos anos 70 que o verdadeiro impacto nos transistores ocorreu quando novas tecnologias permitiram que vários transistores e outros componentes fossem construídos em um mesmo bloco.

A partir desse ponto, o número de componentes integrados em um mesmo circuito dobrava cerca de a cada dois anos, seguindo a Lei de Moore (MOORE et al., 1975). Em suas primeiras análises, Moore observou essa tendência de crescimento nos circuitos, provando ser um dos maiores exemplos de conhecimento prévio do futuro, ou uma detecção de tendência. A Figura 1.1 apresenta a quantidade de transistores dos principais microprocessadores, com a presença de mais de 10 bilhões transistores nos microprocessadores atuais. Inicialmente, a Lei de Moore não passou apenas de uma observação, mas as indústrias de semicondutores tornara isso uma profecia autorrealizável, obtendo um desenvolvimento acelerado ao nível de *hardware*. Porém, a indústria teve que investir em pesquisa e desenvolver testes em novos circuitos fazendo com que houvesse a formulação de uma segunda "Lei de Moore". Essa segunda lei, era previsto um aumento no custo dos circuitos seguido de um aumento do desempenho. Apesar disso, pesquisadores da área dizem que a Lei de Moore pode estar chegando ao fim.

À medida que o número de componentes crescia, a complexidade do projeto de um circuito integrado também aumentava. Os primeiros projetos foram, literalmente, projetados à mão, mas isso logo se tornou um obstáculo e o primeiro *software* para auxiliar no projeto do circuito foi criado em meados da década de 70. Esses *softwares* eram chamados CAD (*Computer Aided Design*), recentemente, passaram a ser chamados EDA (*Electronic Design Automation*), sendo uma categoria de ferramentas para projetar circuitos. Atualmente, a maioria dos projetos de circuitos digitais são feitos utilizando ferramentas de EDA. As ferramentas trabalham juntas em um fluxo de projeto para fabri-

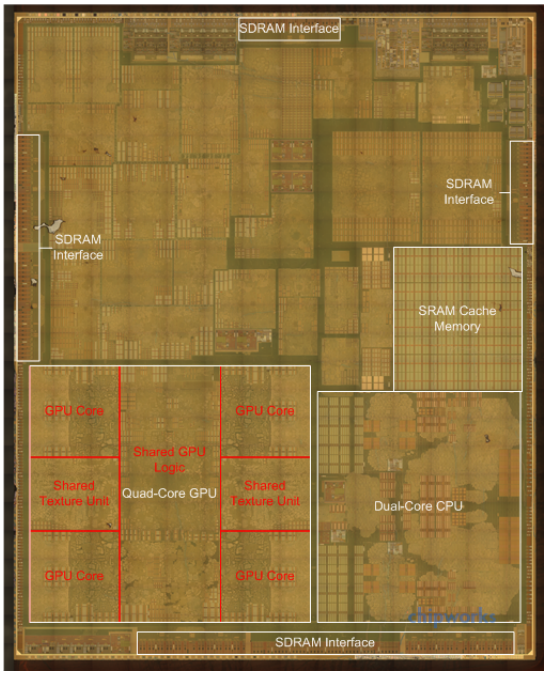
Figura 1.1: Número de transistores em circuitos integrados (1971-2020) (ROSER; RITCHIE, 2022)



car e analisar circuitos inteiros. Como os circuitos podem ter bilhões de componentes, as ferramentas de EDA são essenciais para o projeto. As ferramentas de EDA, além de projetar e analisar, possuem foco de *softwares*, em simulação, na preparação da fabricação e segurança funcional. Com aumento crescente na densidade dos transistores e a miniaturização da tecnologia impõem novos problemas que devem ser otimizados no fluxo de projeto. Circuitos modernos também possuem uma abundância de componentes e uma grande diversidade, que podem ser macro blocos, memória, aceleradores de *hardware*. Por exemplo, o circuito projetado pela Apple de 20 nm chamado A8, ilustra alguns desses problemas, que possui cerca de 2 bilhões de transistores, diferente do antecessor A7 que possui mais de um bilhão de transistores encontrados. Também o tamanho do *die* do A8 é cerca de 13% menor que o *die* do A7. No circuito A8 possui blocos que podem ser identificados como controladores de áudio, codificadores/decodificadores de vídeo ilustrado na Figura 1.2.

Para a fabricação de circuitos, existem duas metodologias que se destacam, *Full Custom* e *Standard Cell*. Na metodologia *Full Custom*, os projetos são customizados. A metodologia *standard cell*, conhecida também como células padrão, trabalha com células que já estão previamente projetadas, caracterizadas e testadas. Essa metodologia é com-

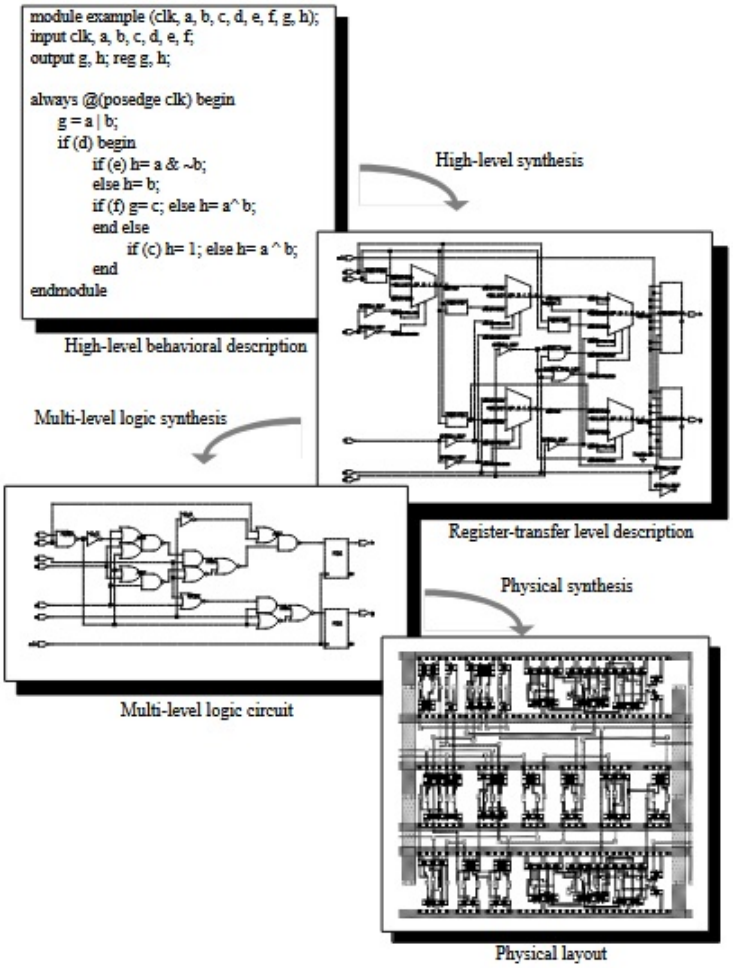
Figura 1.2: Circuito Apple A8 (HO BRANDON CHESTER, 2014)



posta por uma sequência de etapas para elaborar e sintetizar circuitos, cada etapa sendo composta por uma ou mais ferramentas executadas de forma sequencial, ou iterativa. Outras ferramentas também são utilizadas para medir o desempenho, o consumo de energia e entre outras características do circuito. Essas etapas e ferramentas compõem o fluxo de projeto de circuito. O fluxo de projeto de circuito é composto por ferramentas de síntese do circuito, simulação e verificação formal e implementação de estruturas para teste. As principais etapas de síntese neste fluxo de projeto são ilustradas na Figura 1.3. No fluxo, a etapa inicial é a especificação do sistema. Então, a arquitetura do circuito é projetada. O comportamento do circuito é descrito no nível RTL (*Register-Transfer Level*). As funções lógicas em RTL são otimizadas no estágio da síntese lógica. Nesse estágio da síntese, o circuito é otimizado considerando a biblioteca de células. Na última etapa, a síntese física, o leiaute do circuito é construído e otimizado. O *floorplanning* é estabelecido. As células são colocadas em posições otimizadas e seus pinos são roteados com segmentos de metais. A etapa da síntese física é dividida em particionamento, *floorplanning*, posicionamento, síntese da árvore de *clock* (CTS, *Clock Tree Synthesis*), roteamento e *timing closure*.

A etapa de posicionamento é dividido em posicionamento global, legalização e posicionamento detalhado, é composta por vários algoritmos de otimizações. Esses algoritmos contam com heurísticas e métodos formais para otimizar os objetivos do circuito sujeito a restrições. No posicionamento global, as posições das células são otimizadas

Figura 1.3: Principais etapas de síntese do projeto (KRAVETS, 2001)



para minimizar o *wirelength*, sujeito a utilização da densidade de área. O *timing driven* e roteamento do circuito podem ser considerados nesta etapa. Na legalização, as sobreposições das células são removidas e as células são alinhadas ao circuito. Por fim, no posicionamento detalhado, o circuito é otimizado localmente. Uma célula ou um pequeno grupo de células são movidas para posições otimizadas em iterações do algoritmo. Os objetivos dos algoritmos nessa etapa podem ser reduzir o *wirelength*, consumo de energia, tempo de propagação e melhorar a roteabilidade.

Posicionamento é um problema difícil de obter uma solução otimizada. Os algoritmos de posicionamento dependem de métodos heurísticos e formais. As tecnologias avançadas impõem novas restrições de projeto que devem ser abordadas. A formulação do problema de posicionamento e as restrições de projeto abrem oportunidades para pesquisas heurísticas e métodos formais para algoritmos de posicionamento para melhorar a solução de posicionamento.

Na etapa da legalização, como dito antes, os algoritmos movem as células para remover as sobreposições. Algoritmos de legalização podem degradar a solução do posi-

cionamento. Além disso, na legalização é preciso alinhar as células nas trilhas de alimentação corretas. Normalmente, algoritmos de legalização exigem uma *netlist* de entrada posicionada globalmente. Esses algoritmos de legalização fornecem, geralmente, um posicionamento válido. Alguns algoritmos de legalização podem otimizar o posicionamento realizando um posicionamento detalhado também. Portanto, a solução legalizada pode ser melhorada. Alguns algoritmos de legalização também podem ser executados na etapa de posicionamento detalhado.

Portanto, algoritmos de legalização podem otimizar a *netlist* em vários objetivos, com inúmeras restrições e com a necessidade de minimizar os recursos computacionais. Esses algoritmos podem melhorar a solução do posicionamento de acordo com uma métrica. Algoritmos de legalização podem se concentrar em otimizar objetivos como *displacement*, impacto no *wirelength*, densidade, roteabilidade. Algoritmos de legalização também podem ter que lidar com IP (*Intellectual Property*) e células de diferentes alturas, conhecidas como células *multi-row*.

1.1 Objetivos

O objetivo desta dissertação é explorar o problema da legalização quando aplicada em circuitos integrados, considerando de forma paralela os desafios relacionados às arquiteturas complexas e tecnologias avançadas.

Para obter o objetivo mencionado acima, podemos destacar as seguintes contribuições:

- Revisar os fluxos de células padrão, síntese física e posicionamento;
- Apresentar os conceitos de um circuito para a etapa da legalização;
- Explicar o problema da legalização e seus desafios;
- Revisar o estado da arte;
- Desenvolver um algoritmo que se aplique de forma rápida;
- Apresentar um método de legalização e seus resultados.

1.2 Organização

Neste trabalho é organizado com este capítulo introdutório e três capítulos resumidos a seguir.

No Capítulo 2, será revisado uma introdução aos circuitos digitais e ao projeto digital. Esse capítulo contém os conhecimentos básicos para a melhor compreensão do restante do trabalho. Inicialmente será apresentado de forma geral o fluxo de projeto baseado em biblioteca de células. Na sequência é discutido em detalhes os fluxos da síntese física com destaque para a etapa de posicionamento. Por fim são apresentados os conceitos fundamentais relacionados à legalização nesta dissertação.

No Capítulo 3, será apresentado em detalhes o tema abordado nesta dissertação. Esse capítulo contém os principais problemas da etapa da legalização. Inicialmente será apresentada a definição do problema. Na sequência são apresentados os desafios atuais. Por fim, é apresentado os principais algoritmos de legalização do estado da arte.

No Capítulo 4 é apresentado a principal contribuição deste trabalho. Esse capítulo contém os passos do algoritmo proposto. Esse algoritmo é uma adaptação de alguns passos dos algoritmos principais descritos no capítulo anterior.

No Capítulo 5 são discutidos os principais resultados dos experimentos realizados com o algoritmo de legalização. Este capítulo contém as informações necessárias dos resultados. Inicialmente, é apresentado as especificações da máquina que foram executados o algoritmo proposto e outros algoritmos para comparação. Na sequência, as métricas utilizadas são explicadas e a descrição das características dos *benchmarks*. Por fim, os resultados são descritos.

As conclusões e os trabalhos futuros são apresentados no Capítulo 6.

2 CONHECIMENTOS BÁSICOS

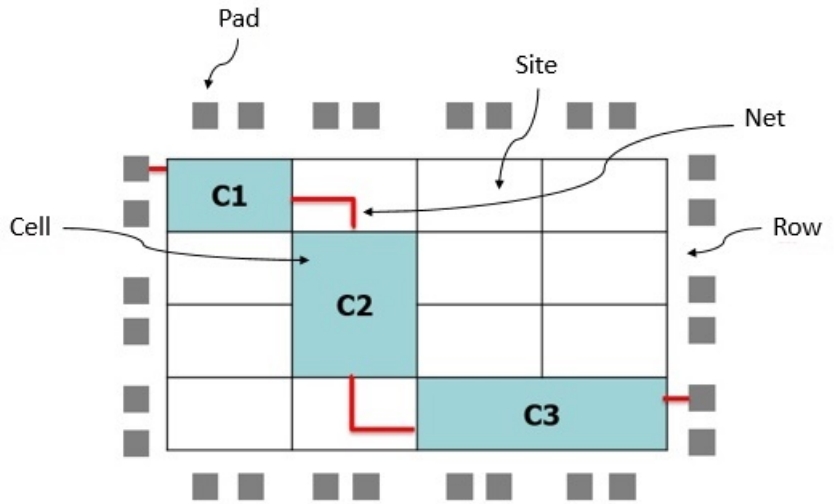
Neste capítulo serão apresentados uma base fundamental para entender este trabalho. Começamos com a apresentação das nomenclaturas necessárias para a etapa de legalização. Na mesma Seção 2.1, serão apresentados as categorias de estimativas de fio. Na Seção 2.2 será revisado o projeto de um circuito integrado utilizando o fluxo de células padrão. Nas seções seguintes, serão apresentados os fluxos das etapas da síntese física e de posicionamento, etapas que pertencem ao projeto de células padrão. Esse capítulo descreve esse conjunto de fluxos para explicar em que parte a legalização está inserida no fluxo para que sua definição seja abordada no Capítulo 3.

2.1 Nomenclatura/Definições

Antes de apresentar as principais etapas do fluxo de projeto de um circuito integrado para o melhor entendimento de onde está inserida a etapa da legalização, essa seção apresenta as nomenclaturas relacionadas com o tema da dissertação. Na sequência, serão apresentadas as principais métricas utilizadas para avaliar um posicionamento legalizado.

Algumas nomenclaturas são adotadas neste trabalho e são necessárias para melhor entender o fluxo do posicionamento e suas subetapas, posicionamento global, legalização e posicionamento detalhado. A Figura 2.1 ilustra estar nomenclaturas:

Figura 2.1: Conceitos básicos



- *Row*: a área do leiaute do circuito abrange várias linhas horizontais de altura fixa, com nenhum espaço entre elas.

- *Site*: cada *row* consiste em vários *sites* alinhados em um *grid* que podem ser ocupados pelos componentes do circuito. Um *site* livre é um *site* que não é ocupado por nenhum componente.
- Célula (*Cell*): é um componente do circuito utilizado tanto na etapa de síntese lógica quanto na etapa de síntese física. Elas representam as portas lógicas que implementarão o circuito. As células podem ter altura e largura variadas. As alturas das células são descritos em número de *rows*, enquanto a largura são descritas em número de *sites*. Exemplo disso está ilustrado na Figura 2.1, onde a célula C1 representa uma célula com uma *row* e largura de um *site*. A célula C2 representa uma célula com duas *rows* e largura de um *site*. Por fim, a célula C3 representa uma célula de uma *row* e largura de dois *sites*;
- *Pads*: são os pinos de E/S do circuito. Geralmente são encontrados nas bordas do circuito;
- *net*: são conjuntos de células/pads que estão conectadas entre si. O conjunto de todas das *nets* é denominado *netlist*. Na Figura 2.1 está exemplificada nas linhas vermelhas, onde as células C1 e C2 pertencem a mesma *net*.

Apresentados os conceitos básicos, é descrito uma das métricas utilizadas para avaliação dos resultados que é o *wirelength*. O *wirelength* de uma *net* é difícil antecipar durante a etapa de posicionamento porque isso depende da qualidade do roteamento adotado. Existem várias abordagens que estimam o *wirelength* de um determinado posicionamento, por exemplo, o HPWL (*Half-Perimeter Wirelength*), o RMST (*Rectilinear Minimum Spanning Tree*) e o RSMT (*Rectilinear Steiner Minimal Tree*), ilustrados na Figura 2.2.

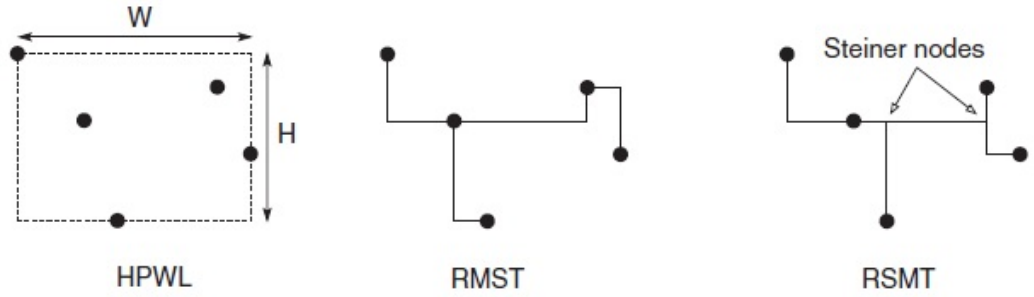
A métrica de estimativa do *wirelength* mais utilizada é o HPWL. O HPWL de uma *netlist* é determinado como a metade do perímetro do menor retângulo que contorna todos os pinos da mesma. O HPWL é popular porque pode ser calculado em tempo linear, ou seja, o tempo de execução aumenta linearmente com o tamanho da entrada (número de pinos da *net* que está sendo calculado o HPWL). Ele pode ser implementado com uma simples função que recebe de entrada as coordenadas dos pinos. O HPWL fornece o comprimento exato para *nets* com 2 e 3 pinos, porém, com 4 e 5 pinos ele fornece uma estimativa otimista de *wirelength* (WANG; CHANG; CHENG, 2009).

Outra abordagem de estimativa é baseada no RMST. O RMST é uma árvore com o mínimo comprimento total na distância Manhattan, em que é apenas a soma das diferenças entre x e y, para conectar um conjunto de nós. A estimativa do *wirelength* usando o RMST

é exato para *nets* com 2 pinos, mas pode superestimar o *wirelength* de *nets* roteadas de maneira ideal com três ou mais pinos em até 50%. Na prática, o RMST pode produzir estimativas de comprimento mais precisas que o HPWL, principalmente com *nets* com grande número de pinos, porém, seu tempo de execução é muito maior que o HPWL (WANG; CHANG; CHENG, 2009).

Uma abordagem altamente precisa é baseada no RSMT, sendo uma árvore com o mínimo comprimento total da borda na distância Manhattan para conectar um determinado conjunto de nós, possivelmente através de alguns nós extras. Se não houver congestionamento de roteamento, o RMST é a melhor aplicação para rotear uma *net*, pois, ela oferece o comprimento mínimo para o roteamento. Assim, o *wirelength* fornecido pelo RSMT tem ligação alta com o *wirelength* para o roteamento, ao não ser que o posicionamento esteja muito congestionado e seja preciso desviar o roteamento (WANG; CHANG; CHENG, 2009).

Figura 2.2: Exemplos de estimativa de *wirelength*. (WANG; CHANG; CHENG, 2009)

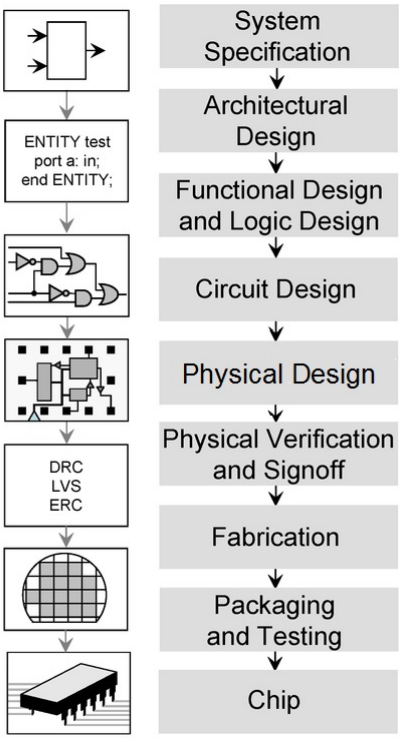


Outra métrica utilizada é o *displacement*. Essa métrica é utilizada na solução da legalização, que deve mover as células para alinhar as células nas *rows* e *sites* e para remover as sobreposições. O *displacement* é medido pela distância Manhattan ou Euclidiana entre sua posição no posicionamento global e legalizado (OIKONOMOU et al., 2017). Um posicionamento legalizado com um alto *displacement* pode identificar um posicionamento global de má qualidade e/ou um algoritmo de legalização ruim.

2.2 Fluxo de células padrão

O fluxo de projeto de células padrão é separado em etapas distintas, como ilustra a Figura 2.3. As primeiras etapas são de alto nível, enquanto as etapas posteriores estão em níveis inferiores de abstração. Essas etapas permitem que os projetistas progridam de etapa em etapa até a implementação final do circuito (KAHNG et al., 2011).

Figura 2.3: Fluxo de células padrão (KAHNG et al., 2011)



O fluxo inicia-se com a especificação do sistema, onde definem os objetivos gerais e os requisitos de alto nível do sistema. Esses objetivos e requisitos abrangem funcionalidade, desempenho, dimensões físicas e tecnologia de produção.

O projeto da arquitetura básica deve ser determinada para atendar às especificações do sistema. Algumas dessas decisões são: integração de blocos analógicos e de sinais mistos, gerenciamento de memória, esquema de endereçamento, número e categorias de núcleos, uso de blocos de propriedade intelectual, requerimentos de desempenho.

No projeto funcional e lógico, as principais unidades funcionais do sistema são identificadas. Isso também identifica os requisitos de entradas, saídas e comportamento de atrasos entre as unidades. A área, potência e outros parâmetros de cada unidade são estimados.

O projeto lógico é executado no nível RTL utilizando HDL (*Hardware Description Language*) por meia de ferramentas que definem o comportamento funcional e de tempo do circuito. Duas HDLs comuns são o *Verilog* e *VHDL* (*VHSIC Hardware Description Language*). Os módulos HDL devem ser completamente simulados e verificados. As ferramentas de síntese lógica automatizam o processo de conversão de HDL em elementos do circuito. Ou seja, dada uma descrição HDL e uma biblioteca de células, uma ferramenta de síntese lógica pode mapear a funcionalidade descrita para uma lista de *nets*,

ou *netlist*, e elementos de circuitos específicos, como as células padrão.

No projeto de circuito, a ferramenta de síntese lógica já mapeou a maioria da lógica do circuito. No entanto, vários elementos críticos devem ser projetados ao nível de transistor. Os elementos projetados no nível do circuito incluem blocos de memória, E/S, circuitos analógicos, multiplicadores. A exatidão do projeto ao nível de circuito é predominantemente verificada por ferramentas de simulação de circuito, como o SPICE. Alguns trabalhos relacionados com às três últimas etapas são: (BAVARESCO, 2008), (ALEGRETTI, 2013), (SCHNEIDER, 2007), (MATOS, 2018), (SCARTEZZINI, 2014), (MARQUES, 2008), (NEUTZLING, 2017).

Durante a síntese física, a netlist é convertida em um leiaute. O leiaute é criado convertendo todos os componentes do projeto em suas representações geométricas. Em outras palavras, todos os macros, células, portas, transistores, etc., com formas e tamanhos fixos por fabricação são atribuídos a localizações e conexões de roteamento concluídas em camadas de metal. O resultado da síntese física é um conjunto de especificações de fabricação que devem ser verificadas nas etapas anteriores. A síntese física é executada conforme as regras do projeto que representam as limitações físicas do meio de fabricação. Por exemplo, todos os fios devem ser separados por uma distância mínima prescrita e ter uma largura mínima prescrita. Como tal, o leiaute do projeto deve ser recriado para cada nova tecnologia de manufatura. A síntese física impacta diretamente no desempenho do circuito, área, confiabilidade, potência e rendimento de fabricação. Devido à sua alta complexidade, o projeto físico é dividido em várias etapas, descritas na Seção 2.3. No fim da síntese física, a otimização elétrica do leiaute é realizada. Resistências, capacitâncias e indutâncias são extraídas do leiaute completo e, em seguida, passam por ferramentas de análise temporal para verificar o comportamento funcional do circuito. Se as análises verificarem um comportamento errado, serão realizadas otimizações incrementais no projeto. Alguns trabalhos relacionados com a etapa de posicionamento são: (MEINHARDT, 2006), (POSSER, 2015), (ALMEIDA, 2014), (SANTOS, 2005).

Após a síntese física, o leiaute deve ser totalmente verificado para garantir a construção elétrica e lógica correta. Alguns problemas encontrados durante a verificação física podem ser tolerados se seu impacto no rendimento do circuito for insignificante. Em outros casos, o leiaute deve ser alterado, mas essas alterações devem ser minímas e não devem introduzir novos problemas. Algumas dessas verificações são o DRC (*Design Rule Checking*), que verifica se o leiaute atende a todas as restrições impostas pela tecnologia, e o LVS (*leiaute vs. Schematic*), que verifica a funcionalidade do projeto. Alguns traba-

lhos relacionados com esta etapa são: (ZIMPECK, 2019), (PARIS, 2017), (GEISSLER, 2014).

O leiaute final, geralmente representado no formato *GDSII*, é enviado para fabricação. Na fábrica, o projeto é padronizado em diferentes camadas usando processos fotolitográficos. As máscaras são usadas de forma que apenas certas camadas sejam expostas a uma fonte de luz. A produção de um circuito requer muitas máscaras. Modificar o projeto requer mudanças em algumas ou todas as máscaras. Os circuitos são fabricados em *wafers* de silício redondo com diâmetro variado. No final do processo de fabricação, o *wafer* fabricado é cortado em circuitos individuais. Os circuitos devem então ser testados e rotulados como funcionais ou com defeito. Os circuitos funcionais são encapsulados e testados para garantir que ele atenda todas as especificações do projeto e ele funcionar corretamente.

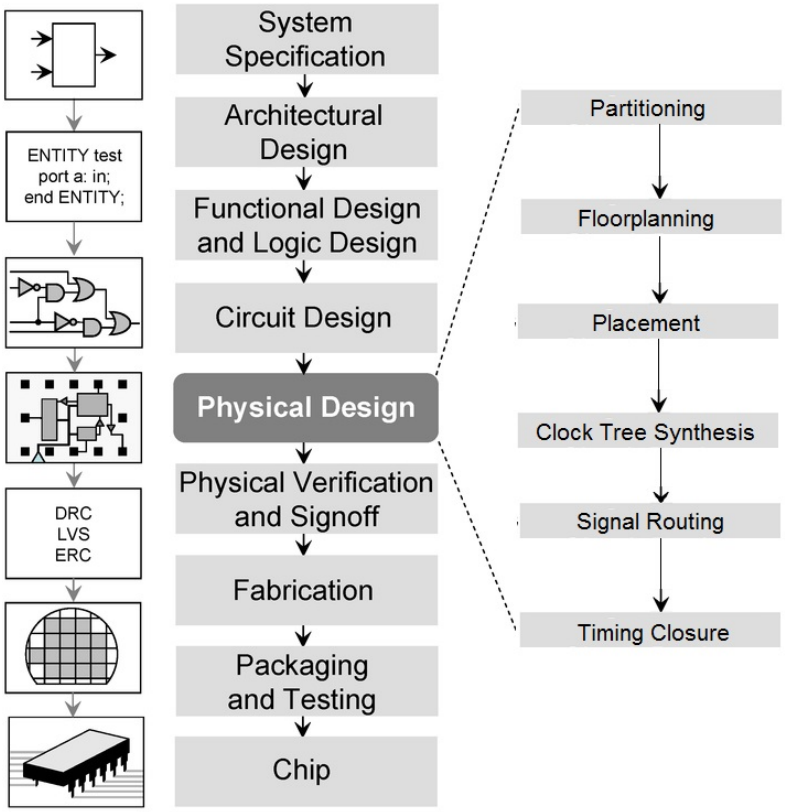
2.3 Fluxo da síntese física

Nessa seção será detalhado o fluxo da síntese física que foi uma das primeiras áreas de foco na pesquisa e desenvolvimento de EDA. A síntese física impacta diretamente no desempenho do circuito, na área, na confiabilidade, na potência e no rendimento da fabricação. Na Figura 2.4 ilustra o fluxo da síntese física que serão descritas a seguir, o fluxo é dividido em várias etapas devido à sua complexidade. O fluxo inicia-se com a etapa de particionamento e *floorplanning*, passa pela etapa de posicionamento, geração da árvore de *clock* e roteamento e, por fim, a análise temporal final.

Uma abordagem popular para diminuir a complexidade do projeto é particioná-los em módulos menores. Esses módulos podem variar entre um pequeno conjunto de componentes elétricos a um circuito totalmente funcional. O particionador divide o circuito em vários subcircuitos (partições ou blocos). Essa divisão tem como intuito minimizar o número de conexões entre as partições, sujeito a restrições de projeto, como tamanhos máximos de partições e atraso máximo. O *floorplanning* é uma etapa necessária porque fornece uma ideia inicial que avalia as decisões arquitetônicas, estima a área do circuito, estima o atraso e o congestionamento dos fios. À medida que a tecnologia avança a complexidade do projeto aumenta e o tamanho do circuito fica maior. Para lidar com a crescente complexidade do projeto, os IPs são amplamente utilizados. Essa tendência torna o *floorplanning* muito mais crítico para a qualidade de um projeto.

O posicionamento é o processo de determinar a localização e a orientação de to-

Figura 2.4: Fluxo da Síntese Física



dos os elementos do circuito, células e macro blocos, no leiaute. Essa localização possui determinadas restrições da solução, como sobreposições, e objetivos de otimização, como minimizar o *wirelength*. Essa etapa é importante no fluxo do projeto porque afeta a capacidade de roteamento, o desempenho, a distribuição de calor e o consumo de energia do circuito. A etapa de posicionamento possui um fluxo que, normalmente, é feito pelas etapas de posicionamento global, legalização e posicionamento detalhado. Esse fluxo será detalhado na Seção 2.4. Alguns trabalhos relacionados com a etapa de posicionamento são: (PINTO, 2011), (MONTEIRO, 2019), (FOGAÇA, 2016), (HENTSCHKE, 2007).

Na etapa de síntese da árvore de *clock*, a parte sequencial do circuito precisa operar corretamente, ou seja, o processamento dos dados deve ocorrer de maneira ordenada e síncrona. Uma *net* de *clock* é necessária para entregar o sinal do *clock* a todos os elementos sequenciais. À medida que a distribuição para interconexões longas se tornaram mais comuns devido ao escalonamento da tecnologia, o controle dos tempos de chegada de cada transição do *clock* em diferentes elementos sequenciais se tornaram mais difíceis. Se não for controlado adequadamente, o *clock skew* pode comprometer o desempenho e a correta funcionalidade do sistema. Alguns trabalhos relacionados com a etapa de *clock* são: (WILKE, 2008), (FLACH, 2010).

Após o posicionamento das células e a síntese da árvore de *clock*, as *nets* do circuito precisam ser roteadas. Essa etapa tem que considerar questões de fabricação, como *vias* redundantes e regras de espaçamento. O roteamento é, normalmente, dividido em duas etapas: roteamento global e roteamento detalhado (WESTE; HARRIS, 2010). A etapa de roteamento global particiona a região do roteamento em blocos e decide os caminhos para todas as *nets*. O roteamento detalhado determina os segmentos e as *vias* exatas para cada uma das *nets*. Alguns trabalhos relacionados com a etapa de roteamento são: (REIMANN, 2013), (TUMELERO, 2015), (NUNES, 2013), (JOHANN, 2001).

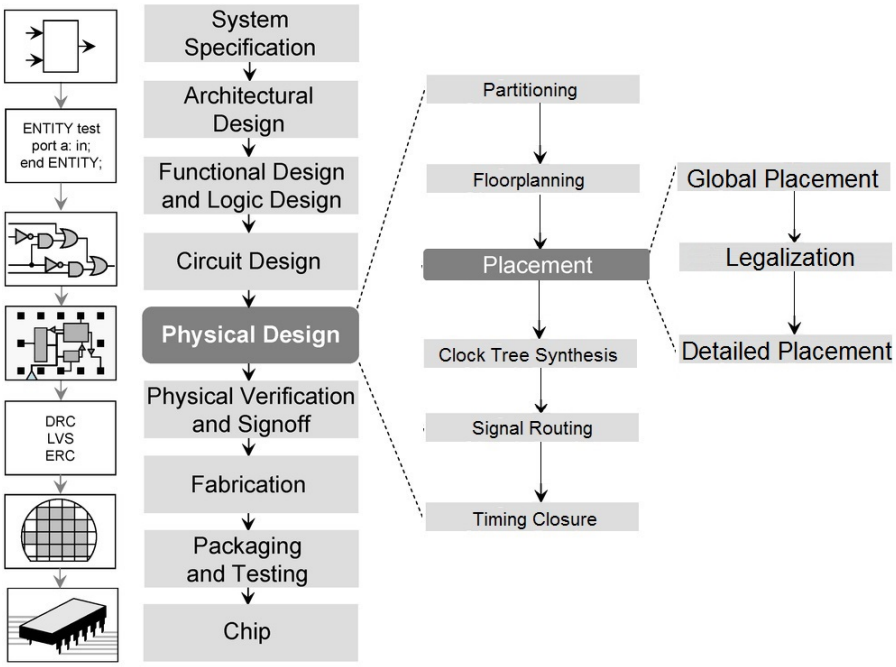
O leiaute de um circuito não deve apenas satisfazer os requisitos geométricos, por exemplo, células sobrepostas e roteamento, mas também atender as restrições de tempo do projeto, por exemplo, tempos de *setup* e *hold*. A etapa de *timing closure* é o processo de otimização que atende à esses requisitos e restrições. Esse processo integra em toda a síntese física, principalmente nas etapas de posicionamento e roteamento (KAHNG et al., 2011). Na síntese física a melhora do tempo é feita alterando a *netlist*. Essas mudanças na *netlist* pode ser redimensionamento de células, inserção de *buffers* e/ou reestruturação do circuito devido aos seus caminhos críticos.

2.4 Etapas do posicionamento

Tradicionalmente, as etapas do posicionamento acontecem após o particionamento e o *floorplanning* e antes do roteamento no fluxo de projeto. Essas etapas correspondem a um conjunto de três etapas: posicionamento global, legalização e posicionamento detalhado, como ilustrado na Figura 2.5. O fluxo do posicionamento procura determinar a localização das células. Após o posicionamento, o roteamento é executado.

O posicionamento é o processo de determinar a localização e a orientação das células na superfície do circuito. O posicionamento é uma etapa importante no fluxo de projeto VLSI, porque afeta o desempenho de um circuito. O posicionamento determina na maioria o *wirelength*, que possui relação direta com o atraso dos fios de interconexão. Como o processo tecnológico continua a reduzir, o atraso das interconexões se tornou um fator determinante para o desempenho do circuito. O posicionamento também determina a capacidade de roteamento de um projeto. Uma solução de posicionamento bem construída terá menos demanda no roteamento. Da mesma forma, o posicionamento influencia na distribuição de calor na superfície do circuito. Um circuito com temperatura desigual pode levar a problemas de confiabilidade e problemas em atingir a frequência de operação

Figura 2.5: Fluxo do posicionamento



inicialmente projetada. Por fim, o consumo de energia é afetado pelo posicionamento em função das resistências e capacitâncias dos fios que deverão interligar as células.

O posicionamento é um problema computacionalmente difícil. Como projetos com milhares de células são comuns nos circuitos atuais, é um desafio projetar algoritmos de posicionamento eficientes para produzir soluções de posicionamento de alta qualidade. Por isso, uma maneira de solucionar esse problema da complexidade é realizar o posicionamento em várias etapas. Um fluxo comum é o feito por posicionamento global, legalização e posicionamento detalhado.

O posicionamento global visa gerar uma solução de posicionamento aproximada que pode ter violações, como sobreposições entre os objetos posicionáveis (células e macro blocos), mantendo uma visão global de toda a *netlist*. A qualidade e o tempo de execução da solução do posicionamento são impactados nessa etapa. Algoritmos baseados no particionamento, a *netlist* e o leiaute são divididos em *subnetlists* e sub-regiões menores, respectivamente, de acordo com uma função custo baseado no corte. Esse processo é repetido até elas serem pequenas o suficiente para otimizar. Um exemplo dessa abordagem é o *min-cut placement*. Outra abordagem é utilizar técnicas analíticas que modelam o problema de posicionamento usando uma função objetivo (custo), que pode ser maximizada ou minimizada. Exemplos de técnicas analíticas incluem o posicionamento quadrático e *force-directed placement*. A outra abordagem é baseada nos algoritmos estocásticos, os movimentos aleatórios são usados para otimizar uma função custo. Um

exemplo dessa abordagem é o *simulated annealing*. O trabalho relacionamento com a etapa de posicionamento global é: (PLÁCIDO, 2016).

O posicionamento global atribui uma posição para cada um dos objetos. No entanto, essas posições não se alinham com as trilhas de alimentação e podem ter problemas a com orientações desses objetos, além de estarem sobrepostas. Portanto, o posicionamento global precisa ser legalizado. A legalização busca tornar a solução do posicionamento global numa solução legal, ou seja, busca remover as sobreposições alinhando nas *rows* corretas conforme as trilhas e em *sites* válidos, enquanto tenta minimizar o deslocamento dos objetos da sua posição global, assim como o impacto do *wirelength*. A legalização é necessária não apenas após o posicionamento global, mas também no posicionamento detalhado.

Dado que a solução do posicionamento legalizado está disponível, essa solução pode ser melhorada em relação a um determinado objetivo através de técnicas do posicionamento detalhado que são normalmente feitas incrementalmente, como trocas de células vizinhas para reduzir o *wirelength* ou mover os objetos para criar espaço para outro. Alguns posicionadores detalhados visam a roteabilidade, visto que as topologias de rota podem ser determinadas assim que o posicionamento legal estiver disponível. O *simulated annealing* pode ser facilmente abordado também para realizar o posicionamento detalhado, mas essa técnica geralmente é lenta. Outra técnica é considerar iterativamente janelas e usar *branch-and-bound* para reorganizar os objetos de forma otimizada em cada janela (WANG; CHANG; CHENG, 2009). O posicionamento quadrático também pode ser abordado no posicionamento detalhado como um segundo estágio, o primeiro estágio o posicionamento global. Em que, o posicionamento detalhado legaliza todas as células e produz um posicionamento de alta qualidade e sem sobreposições. Alguns trabalhos relacionados com a etapa de posicionamento detalhado são: (MONTEIRO, 2014), (FLACH, 2015).

O posicionamento é um problema fundamental nos projetos de VLSI que existem várias formulações e algoritmos diferentes na literatura (WANG; CHANG; CHENG, 2009). Nesta seção foram apresentadas abordagens das etapas de posicionamento global e posicionamento detalhado. A etapa da legalização por ser o foco de investigação dessa dissertação será explorada em detalhes no Capítulo 3.

3 LEGALIZAÇÃO

Este capítulo será focado no tema principal desta dissertação. O capítulo será dividido em seções que descrevem o problema da legalização, os desafios para legalizar um circuito com células padrão, e, uma breve, descrição sobre os algoritmos de legalização do estado da arte.

3.1 Definição do problema

Dado um posicionamento global, onde as células não precisam estar alinhadas e sobreposições são permitidas, a legalização é o processo para tornar a solução válida. Para validação, a legalização precisa deixar a solução sem sobreposições e sem nenhuma violação vinculada a tecnologia de fabricação. Como objetivo principal, essa etapa precisa encontrar um posicionamento legalizado o mais próximo possível da solução do posicionamento global, ou seja, perturbando a solução global o mínimo possível. No entanto, os movimentos nas células são necessários para alinhar as células nas *rows*, remover as sobreposições de células e outras otimizações. *Displacement* é a métrica que avalia esses movimentos medidos pela distância Manhattan entre a posição global e legalizada. Além disso, a legalização é necessária não apenas após o posicionamento global, mas também após mudanças incrementais, como no posicionamento detalhado, mas também na inserção de *buffer* durante a etapa da síntese física.

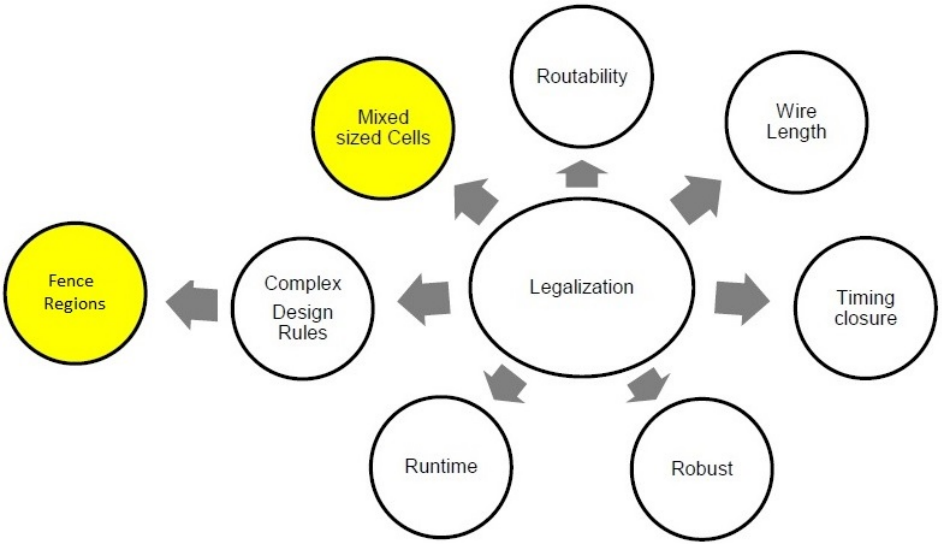
3.2 Desafios

A Figura 3.1 ilustra os desafios da legalização. A roteabilidade e o *wirelength* são otimizados durante o posicionamento global. No entanto, na legalização, a roteabilidade e o *wirelength* são degradadas, pois, a solução do posicionamento global é modificada para deixar o circuito válido. As restrições de tempo são um desafio também porque pode não haver espaço para inserções de *buffers* que atendem os requisitos de tempo. O algoritmo de legalização deve ser robusto para ser um algoritmo que idealmente explore todas as soluções possíveis. Além de ser robusto, o algoritmo precisa ter um baixo tempo de execução. Além das restrições de tempo, existem outras regras de projeto complexas para a legalização. Exemplo disso, é a forma irregular da área do circuito por alguns obstáculos,

podendo ser uma IP. Geralmente, esses desafios têm origem no posicionamento global e no *floorplaning*.

À medida que a tecnologia avança, regras e restrições de projeto mais complicadas estão sendo impostas ao problema de posicionamento. As bibliotecas de células atuais podem ser compostas de células de diferentes alturas, chamadas células *multi-row* (denominadas também como *Mixed-sized cells*). Essa característica com células *multi-row* torna a legalização mais complexa que a legalização de células de altura única (*single-row*), como é destacado a Figura 3.1. A legalização de uma célula *multi-row* pode afetar várias *rows* simultaneamente.

Figura 3.1: Desafios da legalização das células multi-row (DARAV et al., 2017b)



Em um projeto de células *single-row* as *rows* de VDD (*Power*) ou VSS (ou GND *Ground*) são intercaladas entre as rows das células, e cada célula deve estar alinhada e orientada corretamente para que seus pinos de VDD/VSS correspondam as *rows* correspondentes. A Figura 3.5 ilustra como as células são intercaladas pelo VDD e VSS. Para uma célula de altura ímpar (por exemplo, uma célula *single-row* ou de três *rows* de altura), esse alinhamento pode ser efetuado de forma direta ou invertendo na vertical a célula. Por outro lado, as células com altura par (por exemplo, célula com duas *rows* de altura) devem ser alinhadas com a trilha de VDD/VSS, para poderem ser alinhadas em todas as outras *rows* com o trilho de alimentações corretas. A Figura 3.3 ilustra essa legalização com células *multi-row* considerando o problema de alinhamento nas trilhas de VDD/VSS. Conforme mostrado na Figura, as células A e D são de altura ímpar e podem ser posicionadas em qualquer trilha, correspondendo diretamente as trilhas de alimentação VDD/VSS correto ou invertendo a orientação das células verticalmente, enquanto as células B e C são de altura par e devem corresponder a uma trilha de alimentação do

mesmo tipo. Portanto, os limites da célula B estarem alinhados a uma trilha de VDD, torna esse alinhamento incorreto. O mesmo acontece com a célula C num alinhamento de uma trilha de VSS, além disso, essa incompatibilidade na trilha de alimentação não pode ser resolvida com a inversão da orientação vertical da célula.

Figura 3.2: Exemplos de como o VDD e o VSS são intercalados numa célula (DARAV et al., 2017b)

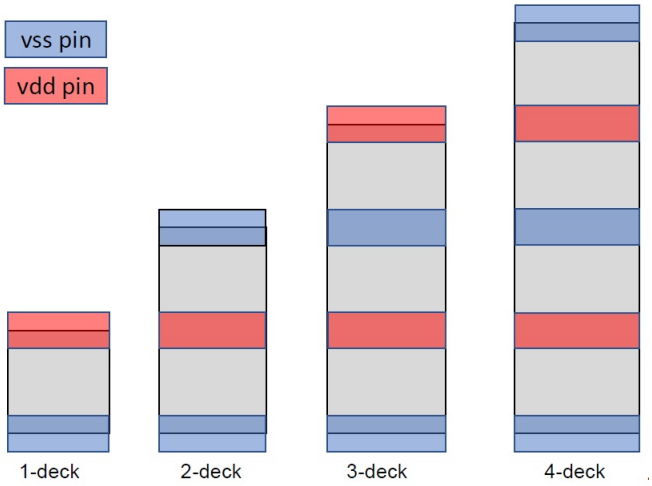
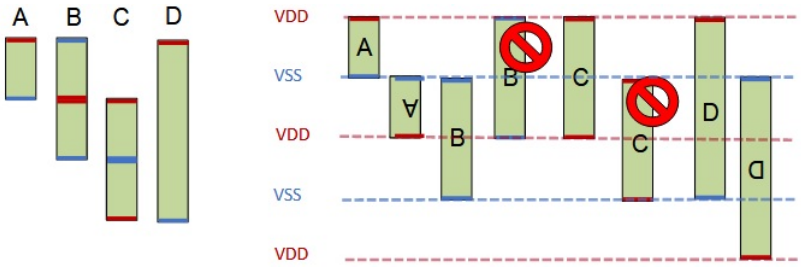


Figura 3.3: Exemplos de células *multi-rows* e seu alinhamento nas trilhas de alimentação (WANG et al., 2017)

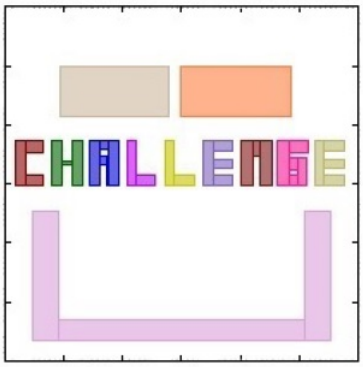


Outra complexidade nos projetos atuais são as *fence regions*. A *fence region* é uma restrição no *floorplaning* e tornam o problema da legalização mais desafiadora e impactam a qualidade de outros objetivos e restrições, como o *wirelength* e a roteabilidade.

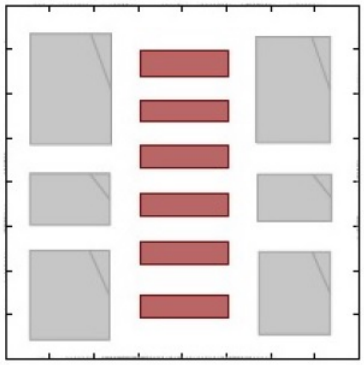
As restrições das *fence regions* são que não permitem as células que não são atribuídas a ela dentro dos seus limites. Além disso, as células atribuídas a uma *fence region* não podem ser posicionadas fora do limite definido. A Figura ilustram dois *floorplannings*. A figura à esquerda ilustra um *floorplanning* que inclui doze *fence regions* com as áreas separadas em diferentes cores. A figura à direita ilustra o *floorplanning* que possui uma *fence region* divididas em seis retângulos vermelhos.

Recentemente, em 2017, o ICCAD (*International Conference On Computer Aided Design*) (DARAV et al., 2017b) promoveu uma competição sobre legalização de células

Figura 3.4: Exemplo de restrições por *fence region* (ZHU et al., 2020)



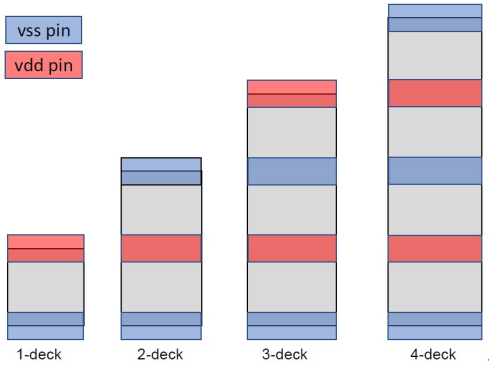
O leiaute de des_perf_b_md1 com 12 *fence regions* em cores diferentes.



O leiaute de edit_dist_a_md2 com 1 *fence region*, que consiste em 6 sub-regiões separadas.

multi-row. Nessa competição foram usadas células de até 4 *rows*, como é ilustrado na Figura 3.5. Os circuitos *benchmarks* que foram utilizados estão ilustrados na Tabela 3.1 com suas características, entre elas: o nome de cada *benchmark*, o número de *rows* nos projetos, o número de macro blocos, o número de células, de *nets*, de *fence regions*, de *pads*, respectivamente. A porcentagem das células de cada tipo também é descrita. Por fim, as densidades dos circuitos de cada projeto. Esses *benchmarks* serão os circuitos utilizados nesse trabalho.

Figura 3.5: Células *multi-row* com suas fontes de alimentação VDD e VSS (DARAV et al., 2017b)



Alguns algoritmos de legalização também possuem outra complexidade que é possuir uma legalização guiada pelas regras de roteamento. As restrições de roteamento incluem espaçamento entre as bordas das células, acesso de pino e curto no pino. Os sinais dos pinos das células não podem dar curtos ou serem inacessíveis devido às grades de VDD/VSS e aos pinos de E/S. Essa complexidade acontece devido à contínua miniaturização dos transistores (XU et al., 2017), associada com a redução no número de *tracks* na célula, fazendo com que a roteabilidade interna dentro de uma célula se torne inadequada. Nas tecnologias abaixo de 10 nm, existem células com apenas seis *tracks* de altura

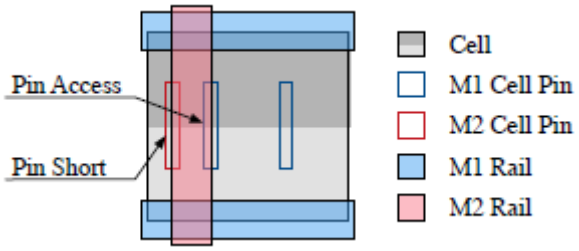
Tabela 3.1: As características dos *benchmarks* utilizados na competição de legalização de *multi-row* do ICCAD 2017 (DARAV et al., 2017b)

							% cell types				
Design	#Row	#Macros	#Cells	#Nets	#Fence Region	#I/O	1xH	2xH	3xH	4xH	Utilization
des_perf_a_md1	450	4	103589	115187	4	374	95.66	4.34	0.00	0.00	55.11
des_perf_a_md2	450	4	103589	115187	4	374	96.99	1.00	1.00	1.00	55.92
des_perf_b_md1	300	0	112679	122951	12	374	94.80	5.20	0.00	0.00	54.98
des_perf_b_md2	300	0	112679	122951	12	374	90.47	6.02	2.01	1.50	64.69
edit_dist_1_md1	361	0	130661	133223	0	2574	90.31	6.12	2.04	1.53	67.47
edit_dist_a_md2	400	6	127414	134051	1	2574	90.31	6.12	2.04	1.53	59.42
edit_dist_a_md3	400	6	119626	134051	1	2574	93.88	2.04	2.04	2.04	57.22
fft_2_md2	171	0	32281	33307	0	3010	89.62	6.56	2.18	1.64	83.12
fft_a_md2	400	6	30625	32090	0	3010	89.57	6.59	2.19	1.65	32.41
fft_a_md3	400	6	30625	32090	0	3010	93.42	2.19	2.19	2.19	31.24
pci_bridge32_a_md1	200	4	29533	34058	3	361	90.39	6.07	2.02	1.52	49.57
pci_bridge32_a_md2	200	4	29533	34058	3	361	85.51	7.08	4.04	3.37	57.72
pci_bridge32_b_md1	400	6	26134	32546	3	361	90.38	6.07	2.02	1.52	28.68
pci_bridge32_b_md2	400	6	26134	32546	3	362	97.97	1.01	1.01	1.01	19.72
pci_bridge32_b_md3	400	6	26134	32546	3	363	94.94	1.01	2.02	2.02	23.98

(MATTII et al., 2017), resultando em dificuldade no projeto de células complexas, como multiplexadores (BAEK et al., 2008) e *multi-bit flip-flop* (LIN; HSU; CHANG, 2011) com alto *drive strength*.

Um problema de curto no pino é quando ele se sobrepõe a uma trilha de VDD/VSS ou um pino de E/S e estão na mesma camada de metal. Se a camada de metal da trilha de VDD/VSS ou um pino de E/S estiver apenas em uma camada acima da camada do pino de sinal este pino será inacessível. Conforme mostrado na Figura 3.6, o pino esquerdo na camada M1 (metal 1) tem problema de acesso de pino com a trilha na camada M2 (metal 2) e o pino M2 dá curto com trilho M2.

Figura 3.6: Acesso de pinos e curto de pino . (LI et al., 2018)

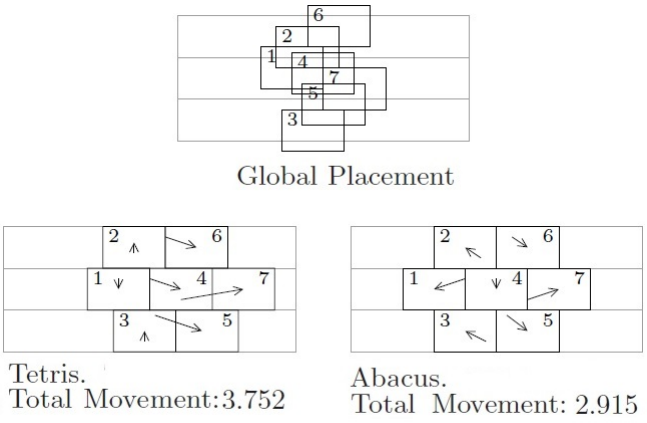


3.3 Estado da arte

Os métodos de legalização podem ser categorizados como: algoritmos heurísticos e analíticos. A maioria dos métodos de legalização se enquadram no primeiro grupo que resolve as violações (como sobreposições de células), explorando um algoritmo guloso

de mover as células em áreas violadas para locais apropriados. Os métodos heurísticos podem levar a um movimento desnecessariamente grande de células para projetos com bloqueios ou alta utilização, pois, eles tentam resolver localmente em vez de usar uma abordagem global. Isso pode alcançar uma solução em um tempo prático, mas não garante uma boa solução. Tetris (HILL, 2002) e Abacus (SPINDLER; SCHLICHTMANN; JOHANNES, 2008) demonstraram ser os métodos mais populares desta categoria. Tetris é um método rápido, na época se denominava posicionamento detalhado, que legaliza as células uma a uma sempre nos *sites* da esquerda para direita. Após uma célula ser legalizada, ela seria fixa. Abacus é semelhante, mas move as células que já foram legalizadas, colocando todas numa posição ótima com o mínimo de movimento. A Figura 3.7 ilustra as diferenças desses métodos. A figura ilustra um posicionamento global que precisa ser legalizado. Como também, a figura demonstra o problema do Tetris detalhado anteriormente. Assim como a figura ilustra o método do Abacus que move células já legalizadas, que resulta num menor movimento total das células.

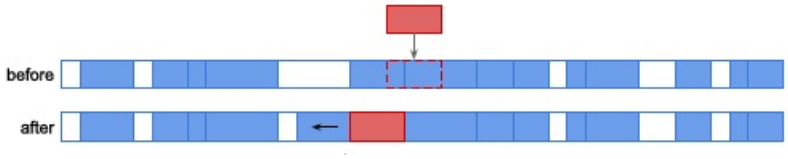
Figura 3.7: Posicionamentos legalizados de Tetris (HILL, 2002) e Abacus (SPINDLER; SCHLICHTMANN; JOHANNES, 2008)



Esses dois métodos se tornaram clássicos. Muitos outros algoritmos de legalização se basearem nas suas soluções como, por exemplo, (DARAV et al., 2015) um posicionador global e legalizador derivado do Abacus que considera as *fence regions*, e inclui um método para ajustar a densidade para ter um roteamento global de boa qualidade. (PUGET et al., 2015) legaliza as células uma a uma e escolhe o *displacement* mínimo necessário para colocar uma célula numa *row*, mas com uma função diferente do Abacus, além de suportar circuitos com bloqueios. A Figura 3.8 ilustra como é seu método de colocação das células, que mantém a ordem relativa das células. Essa manutenção da ordem é uma propriedade bem conhecida como manutenção da qualidade do posicionamento.

Outros métodos são o (WANG et al., 2017) apresenta um método de legalização

Figura 3.8: Exemplo de legalização do (PUGET et al., 2015)



que estudou Abacus — método de legalização para células *single-row*, porém, não adequado para lidar com *multi-row* — e analisou suas capacidades, insuficiências e estendeu suas vantagens para desenvolver um algoritmo que abordasse a legalização *multi-row*. A Figura 3.9 ilustra como é feita a legalização deste método com uma função de reconhecimento de espaços em branco e um esquema de otimização para lidar com esse problema. (OIKONOMOU et al., 2017) apresenta Domocus, um método de legalização que paraleliza o Abacus. A abordagem foi dividir a área do circuito e considerar essas subáreas de forma independente. Os conflitos existentes são resolvidos de maneira sequencial. A Figura 3.10 ilustra o *speedup* do Domocus sobre o Abacus. Conforme a Figura ilustra o desempenho do HPWL e *displacement* do Domocus, para 1, 2, 4, 8 e 12 *threads*, como uma melhoria percentual em relação ao Abacus. A figura ilustra os resultados médios da melhoria em todos os circuitos.

Figura 3.9: Método que legaliza as células posicionando nos espaços em branco. (WANG et al., 2017)

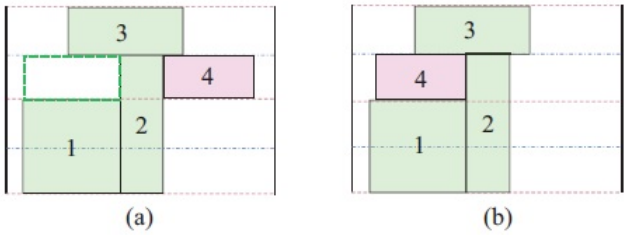
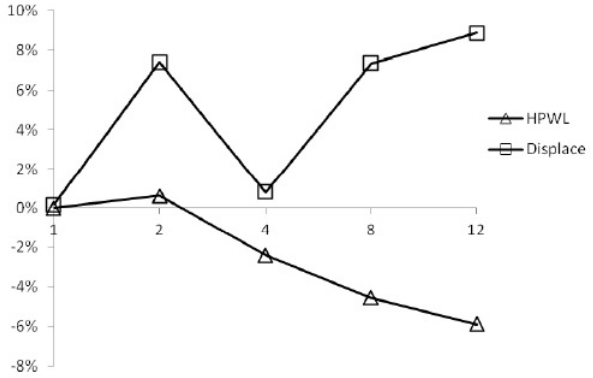


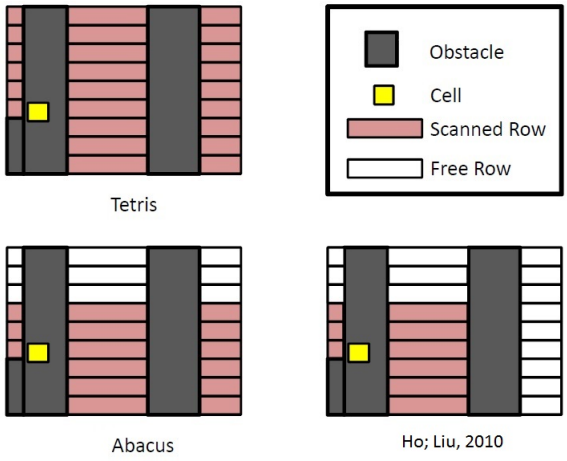
Figura 3.10: Melhoria do desempenho médio (%) para as métricas de HPWL e *displacement* (eixo x mostra o número de *threads*) (OIKONOMOU et al., 2017)



Além destes, outros métodos de legalização utilizaram métodos heurísticos como

(KAHNG; MARKOV; REDA, 2004). Esse método de legalização verifica se tem mais células que a capacidade da *row* e depois remove as sobreposições nas posições verticais conservando o HPWL horizontal. (CONG; XIE, 2008) propõe um método de legalização de *multi-row*. Além da legalização, propõe um posicionamento detalhado. Utiliza grafos para determinar a posição das células resolvidas por LP (*Linear Programming*) que primeiro legaliza os macro blocos. (HO; LIU, 2010) apresenta um método que legaliza utilizando uma árvore binária que investiga as *rows* e verifica a melhor *row* para minimizar o HPWL e o *displacement* de cada célula. A ordenação das células é pela posição x e metade da largura das células. A Figura 3.11 ilustra como a investigação de *rows* de Tetris, Abacus e (HO; LIU, 2010) é realizado. Tanto Tetris quanto Abacus realizam uma varredura de *rows* com o custo vertical mínimo. O Tetris verifica todas as *rows* como ilustra a figura. Abacus classifica as *rows* pela posição y. Um limite é calculado assumindo que as células são movidas apenas verticalmente. Abacus move a célula para a melhor *row* primeiro e, em seguida, tenta mover a célula para as *rows* dentro desse limite não excedendo o custo mínimo de uma posição legalizada já encontrada, como ilustra a figura. O legalizador (HO; LIU, 2010) constrói uma árvore binária que verifica apenas algumas *rows* candidatas, como ilustra a figura.

Figura 3.11: Método de (HO; LIU, 2010) que só verifica as *rows* quando são necessárias, enquanto Tetris verifica todas as *rows* e Abacus verifica as *rows* pela posição y



Igualmente, (CHOW; PUI; YOUNG, 2016) propõe um algoritmo guloso que busca uma solução de legalização com mínimo *displacement* para cada célula dentro de uma região local definida em torno da sua posição definida pelo posicionamento global de maneira sequencial. A Figura 3.12 ilustra como o algoritmo MLL (*Multi-row Local Legalization*) funciona para legalizar células *multi-row* em uma região local. A figura mostra o posicionamento original e a marcação tracejada em vermelho será onde será a região local.

Após isso mostra a região local extraída. A seguir, o algoritmo MLL dará uma solução legal inserindo a célula alvo (t) e legalizando na região local. Por fim, o posicionamento resultante é então mostrado na figura. Quando uma solução legal não for encontrada com o algoritmo, ele abortará e retornará sem alterar o posicionamento. Após a primeira iteração e quando ainda houver alguma célula não posicionada, o processo é repetido para cada célula não posicionada, incluindo adição número aleatório nas posições das células. Essa iteração de posicionamento é repetida até que todas as células sejam colocadas. (DO; WOO; KANG, 2019) propõe um método de legalização para células *multi-row* orientada a regiões de *fence*. O algoritmo consiste em etapas de pré-legalização, que organiza as células fora e dentro das regiões de *fence* e somente, após isso a legalização é feita. Por fim, uma etapa de aprimoramento da qualidade utiliza o *simulated annealing* para reduzir o *displacement*. A Figura 3.13 ilustra como é realizada a pré-legalização. As Células da região de *fence* que estão fora das suas regiões. O Deslocamento para a região. As Células da região de *fence* que estão dentro da sua região. As Células que precisa estar fora da região de *fence* são deslocadas.

Figura 3.12: Exemplo de extração de uma região local. (CHOW; PUI; YOUNG, 2016)

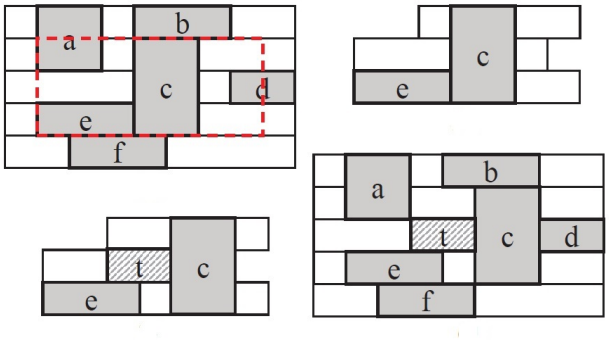
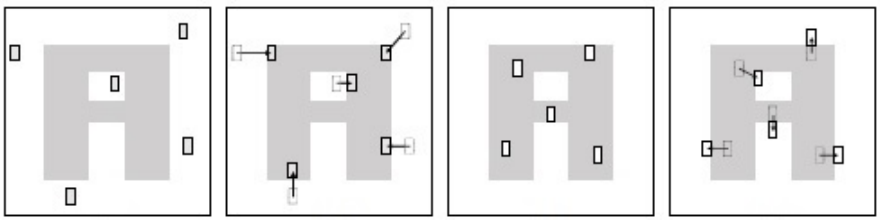


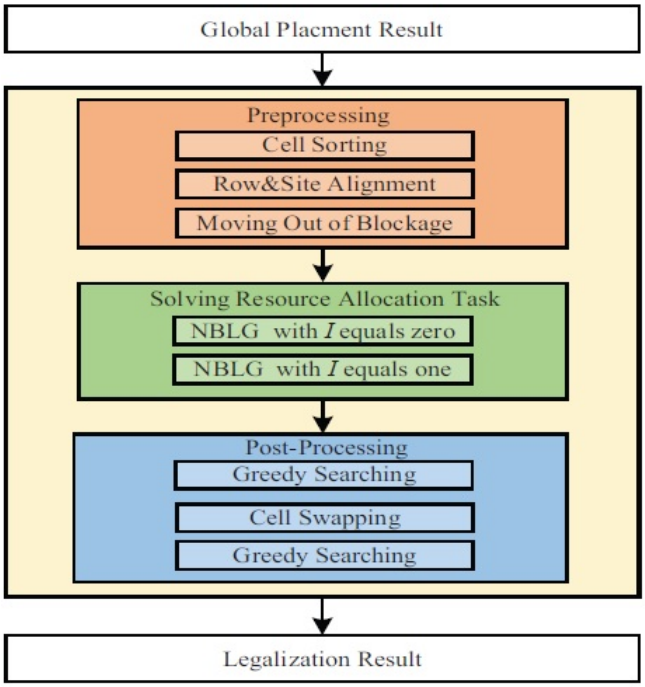
Figura 3.13: Exemplo de como é realizada a pré-legalização



Do mesmo modo, (CHEN et al., 2021) apresenta um método chamado NBLG (*negotiation-based legalizer*) que propõe reduzir o *displacement* médio e o *displacement* máximo para circuito *multi-row*, considerando as *fence regions* e as restrições de tecnologia. A Figura 3.14 ilustra o fluxo geral do algoritmo de legalização. O algoritmo consiste em três partes: pré-processamento, resolver a tarefa de alocação de recursos e

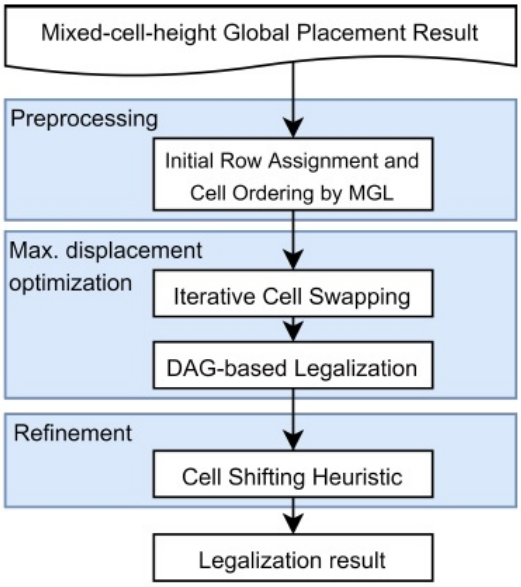
pós-otimização. Dado um resultado de posicionamento global, dimensionam o tamanho do circuito pela largura do *site*, afasta as células dos bloqueios/macro blocos e alinham as células nas *rows* e *sites* corretos. Na etapa principal da resolução do problema de alocação de recursos, o NBLG elimina todas as sobreposições entre as células iterativamente com base em cinco componentes de negociação (*Negotiation Object*, *Cost*, *Action*, *Order* e *Region*). Por fim, dois métodos de pós-otimização diminuem ainda mais o *displacement* das células. (WU; MAK; CHU, 2022) apresenta um método de legalização de células *multi-row* em tempo linear altamente eficiente que otimiza tanto o *displacement* total quanto o *displacement* máximo da célula. A Figura 3.15 ilustra o fluxo da abordagem. Neste método existe três estágios no fluxo: pré-processamento, otimização do *displacement* máximo e refinamento. Na etapa do pré-processamento, uma técnica de inserção de células e ordenamento de células é obtida pelo MGL (LI et al., 2018). Na segunda etapa, propõem um algoritmo iterativo de troca de células para alterar às atribuições nas *rows* e a ordem das células críticas para redução máxima do *displacement*. Em seguida, um algoritmo baseado em DAG (*Directed Acyclic Graph*) é aplicado para minimizar ainda mais o *displacement* máximo. Na etapa de refinamento, propõem uma heurística de *displacement* para reduzir o *displacement* total das células sem aumentar o *displacement* máximo da célula.

Figura 3.14: Fluxo do algoritmo de legalização (CHEN et al., 2021)



Por outro lado, abordagens analíticas formulam modelos matemáticos aplicando funções objetivas e buscam a solução ideal globalmente para o problema da legalização

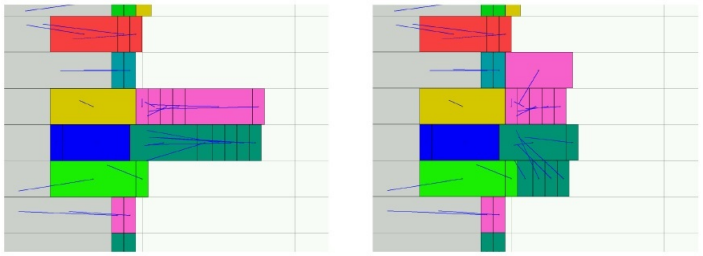
Figura 3.15: Fluxo geral do método (WU; MAK; CHU, 2022)



como *network flow*. Nos métodos de *network flow*, a legalização é modelada como um fluxo de custo mínimo (AHUJIA; MAGNANTI; ORLIN, 1993), onde as células (fluxo) são movidas entre nós de *source* e *sink* para resolver partições com excesso de fluxo e minimizar o movimento das células. Por exemplo, (BRENNER; PAULI; VYGEN, 2004) explora um método de legalização de fluxo de custo mínimo, propondo dois algoritmos, o primeiro é baseado em LP e ILP (*Integer Linear Programming*) relaxado. Nesse método eles particionam a área do circuito em colunas iguais e consideram os macro blocos sendo fixos. Além de ter a preocupação com a roteabilidade e o *timing*. (CHO et al., 2010) propõe um algoritmo de legalização baseado em histórico com uma formulação de *network flow* de custo mínimo, que encontra um posicionamento com o mínimo *displacement* de maneira iterativa. Em (BRENNER, 2013), o algoritmo particiona o circuito e para cada partição atribui células a essas partições. Para evitar partições cheias demais, utilizam a atribuição por um algoritmo que calcula o fluxo entre as posições. Por utilizar uma abordagem diferente de legalização baseada em fluxo perdem a propriedade de encontrar um fluxo ideal. A Figura 3.16 ilustra uma etapa final, que permite que as células para partições vizinhas em *rows* vizinhas se isso diminuir o movimento total. Esta etapa tem uma oportunidade de melhorar o resultado. Na Figura 3.16, é possível ver uma pequena parte do circuito, onde os bloqueios são mostrados em cinza, enquanto as células são coloridas de acordo com suas partições. As linhas azuis conectam o centro da célula com suas posições antes e depois da legalização. A Figura 3.16 ilustra o posicionamento legalizado antes da pós-otimização. Os dois grupos de células no centro cabem nas suas partições, mas eles possuem um movimento horizontal significativo. Com isso, algumas células são

movidas para cima ou para baixo em partições vizinhas livres. O resultado é ilustrado também na Figura 3.16.

Figura 3.16: Exemplo do posicionamento antes e após a pós-otimização. (BRENNER, 2013)



Igualmente, (DARAV et al., 2017a) é um método de legalização baseada em fluxo, mas com diferenças quando comparada aos anteriores. Nesta solução o movimento máximo da célula é considerado como uma restrição. O algoritmo pode legalizar projetos complexos e com alta densidade. A Figura 3.17 ilustra como o algoritmo funciona. Uma região de uma solução de posicionamento global é ilustrado com o arranjo de células. Após isso, a quantidade de espaço em branco disponível para cada partição.. As partições, coloridas em rosa, com valores negativos estão com excesso de células. Neste exemplo, a partição cheia é localizada na partição [1,2] (linha 1, coluna 2) deve ser resolvida. Para melhorar a ilustração, as células de dentro desta partição são coloridas em laranja. Em seguida, as células sujeitas a movimentos para partições vizinhas são mostradas em azul e roxo. A seguir, os caminhos candidatos são simbolizados por $(\uparrow)$ e caminhos inválidos são simbolizados por $(\perp)$. Posteriormente, é ilustrada a nova disposição das células após mover as células em todos os caminhos candidatos. Por fim, as partições com excesso de células são mostradas. Conforme mostrado na Figura a partição [1,2] não está mais com excesso de células. Além disso, nenhuma nova partição com excesso de células é criada ou não ocorre o aumento de produção de partições com excesso de células.

Além de (LI et al., 2018) um método de legalização de células *multi-row* por uma técnica de inserção de célula baseada em janelas, chamada MGL (*multi-row global legalization*), e duas otimizações baseadas no *network flow* e pós-processamento. A Figura 3.18 ilustra um experimento realizado por eles, que verifica a eficácia das duas etapas de pós-processamento. O exemplo é a otimização do deslocamento máximo em que as células vermelhas são do mesmo tipo, enquanto as linhas vermelhas conectam as células às suas posições globais iniciais. As células com um grande *displacement* são movidas para locais mais próximos como ilustram as figuras.

Diferentemente, (CHEN et al., 2017) utiliza outro método analítico, um método

Figura 3.17: Exemplo de um posicionamento inicial,posição de cada partição, caminhos candidatos e uma solução para a partição [1,2]. (DARAV et al., 2017a)

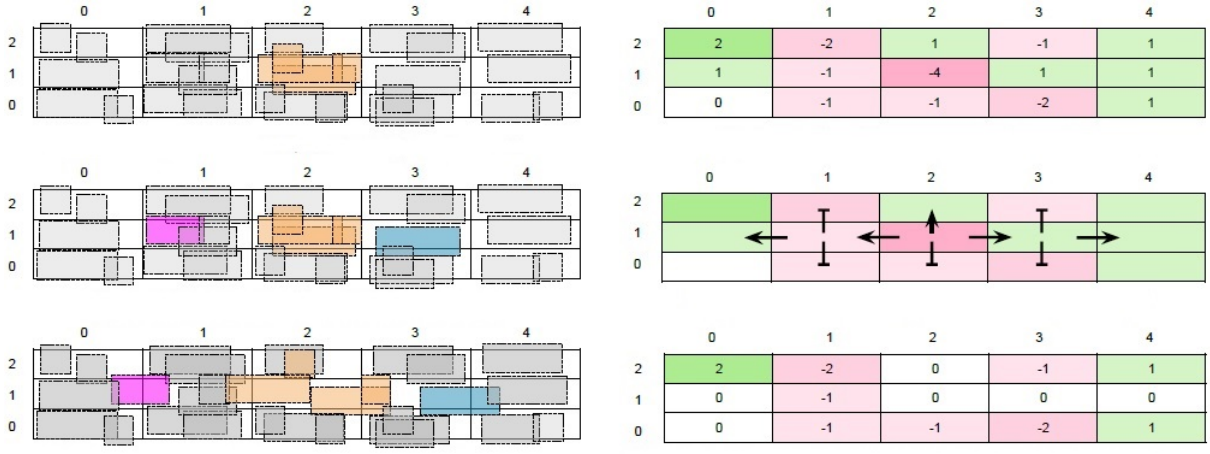
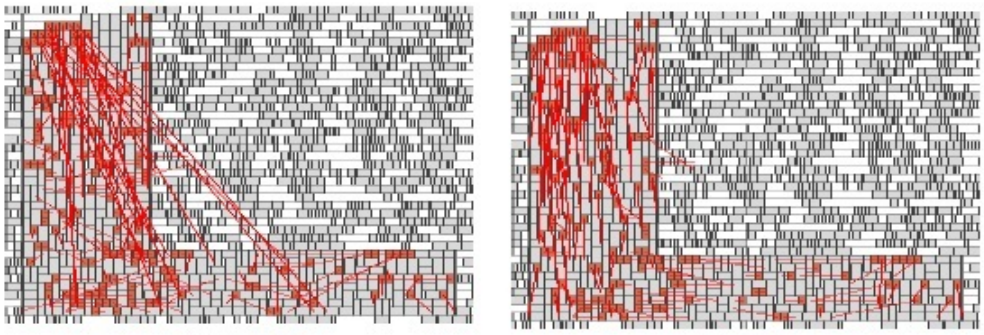


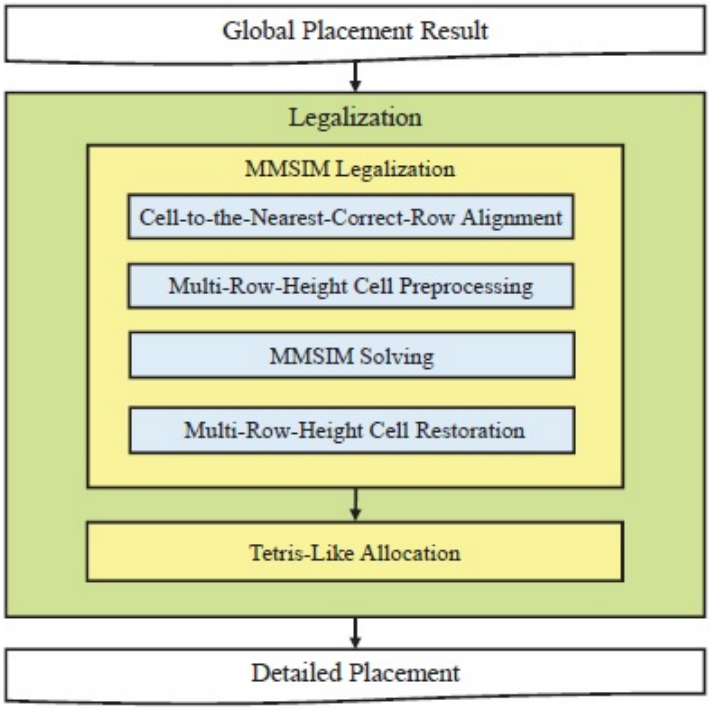
Figura 3.18: Exemplo da otimização do *displacement* máximo. (LI et al., 2018)



rápido e quase ideal para resolver o problema da legalização. Primeiramente, converte o problema da legalização em LCP (*Linear Complementarity Problem*). Após isso, eles convertem em matrizes para atender o requisito do MMSIM (*Modulus-Based Matrix Splitting Iteration Methods*) (BAI, 2010). Esse método MMSIM garante a otimização se nenhuma célula for posicionada além do limite direito do circuito. Se houver, é utilizada uma abordagem semelhante ao Tetris é utilizada para alinhar as células que estão fora do limite direito. Por utilizar um método analítico, o posicionamento resultante ainda pode ter sobreposições e precisar de outra etapa de legalização. A Figura 3.19 ilustra o fluxo geral do algoritmo de legalização de células *multi-row*. A entrada é o resultado de um posicionamento global, que calcula a melhor posição para cada célula, ignorando as sobreposições entre as células. O algoritmo de legalização primeiro alinha cada célula na *row* correta mais próxima, ignorando restrições a borda direita do circuito. Se uma célula for *multi-row*, ela será dividida em subcélulas *single-rows*. Após esse pré-processamento das células *multi-row*, o MMSIM é usado para resolver o problema LCP. Em seguida, as células *multi-row* são restauradas. Finalmente, é utilizada uma abordagem semelhante ao

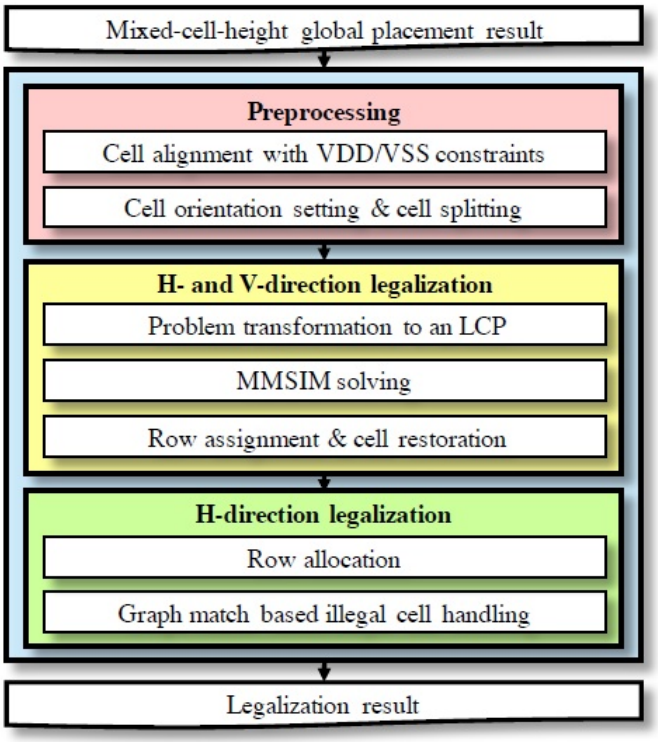
Tetris é utilizada para alinhar todas as células.

Figura 3.19: Fluxo de legalização (CHEN et al., 2017)



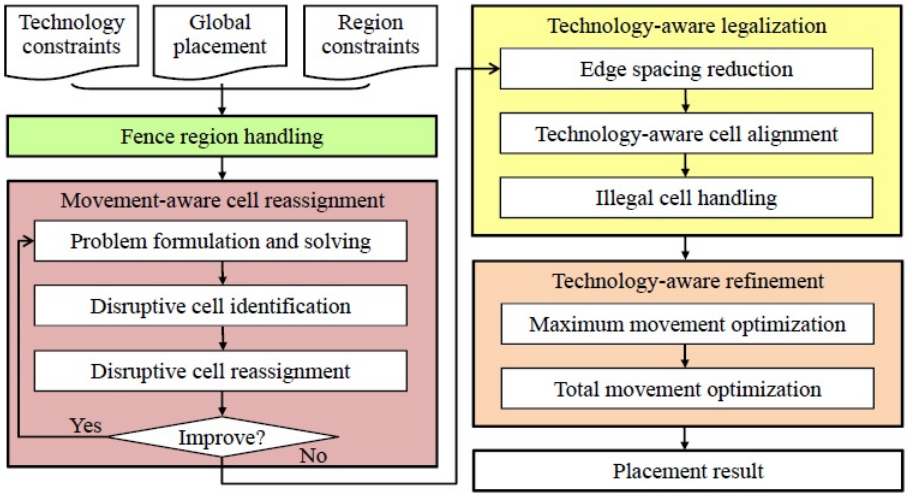
Igualmente, (LI et al., 2019) apresenta um algoritmo analítico de legalização de células *multi-row* que minimiza simultaneamente os *displacement* médios e máximos das células. A Figura 3.20 ilustra o fluxo de legalização que consiste em três etapas principais: *preprocessing*, *horizontal-and-vertical-direction legalization* e *horizontal-direction legalization*. Na etapa de *preprocessing*, todas as células são alinhadas às suas *rows* próximas corretas, atendendo às restrições do alinhamento de VDD/VSS, deslocando as células na direção vertical. Além disso, cada célula é dividida em subcélulas *single-rows*. Na etapa *horizontal-and-vertical-direction legalization*, o problema é modelado em um problema LCP, que considera simultaneamente o movimento da célula nas direções horizontal e vertical. Em seguida, aplicam o MMSIM para resolver o LCP. Após a execução do algoritmo baseado no MMSIM, todas as células *multi-row* são restauradas. Após essa etapa, ainda podem existir sobreposições. Na etapa *horizontal-direction legalization*, primeiro alinham cada célula ao seu *site* mais próximo. Depois disso, se uma célula não se sobrepõe a nenhuma outra célula, a posição dessa célula será fixa. Entretanto, uma célula é violada quando estiver sobreposta com qualquer outra célula ou fora do limite direito. Para cada célula violada, procuram todos os espaços vazios possíveis. Em seguida, um grafo bipartido para modelar as células violadas e espaços vazios é construído. Após essa operação, todas as células são posicionadas legalmente na região do circuito integrado.

Figura 3.20: Fluxo de legalização (CHEN et al., 2017)



Do mesmo modo, (ZHU et al., 2020) apresenta um algoritmo de legalização eficaz para resolver projetos com células *multi-row* e restrições de *fence region*. A Figura 3.21 ilustra o fluxo geral dos algoritmos, que consiste em quatro etapas. Primeiro apresentam uma técnica de tratamento de *fence region*, que divide o projeto em vários subcircuitos, onde cada subcircuito corresponde a uma região. Para cada célula, o algoritmo primeiro alinha as células no seu subcircuito correspondente e *rows* corretas mais próximas. A etapa de *movement-aware cell reassignment* é operada iterativamente da seguinte maneira. Em cada iteração, com base no posicionamento e ordenação das células atuais, afrouxando as restrições de limite, primeiro eles formulam o problema da legalização que consideram os movimentos totais, máximos e as restrições da tecnologia com problema de QP (*Quadratic programming*) e resolvem pelo MMSIM. Então, com base no resultado do posicionamento obtido, identificam células que possuem um movimento excessivo. Além disso, realizam um algoritmo de fluxo de custo mínimo (SMITH, 1994) para reposicionar as células às suas melhores *rows* possíveis. O processo termina se a função de *score* não for melhorada. Na etapa de *technology-aware legalization*, removem todas as sobreposições das células e alinham as células aos *sites*. Por fim, apresentam um método de troca de células para reduzir o movimento máximo e, em seguida, um gráfico bipartido é construído para reduzir ainda mais o movimento total, garantindo que o movimento máximo não aumente.

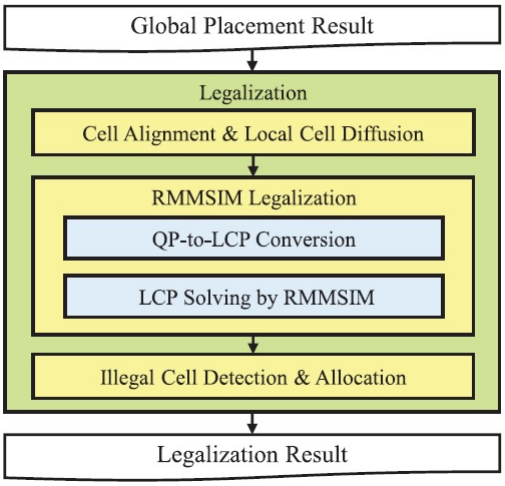
Figura 3.21: Fluxo do algoritmo (ZHU et al., 2020)



Além de (CHEN et al., 2020) apresenta um algoritmo de legalização que, novamente, com base no MMSIM. O problema de legalização de células *multi-row* é convertido em um problema LCP e desenvolveram um RMMSIM (*Robust Modulus-Based Matrix Splitting Iteration Methods*) para resolver o LCP eficientemente. A Figura 3.22 ilustra o fluxo geral do algoritmo de legalização de células *multi-row*. A entrada do fluxo é a solução do posicionamento global, que calcula a melhor posição alinhada para cada célula, ignorando as sobreposições entre as células. Se a distribuição das células em uma *row* for localmente dispersa, então as sobreposições entre essas células na *row* podem ser resolvidas de maneira desejável pela legalização do RMMSIM. No entanto, se for localmente denso, o alinhamento das células podem não ser desejáveis. Portanto, para alcançar uma solução de legalização de alta qualidade, é adotado ainda *local cell diffusion* em regiões densas para obter uma melhor atribuição e ordenação das células. Então, ignorando as restrições do limite direito do circuito, convertem o problema QP em LCP e utilizam o RMMSIM para resolver o LCP. Por fim, alinham todas as células nos *sites* e *rows* e utilizam uma abordagem de atribuição ilegal de células para corrigir as células sobrepostas ou fora do limite direito do circuito.

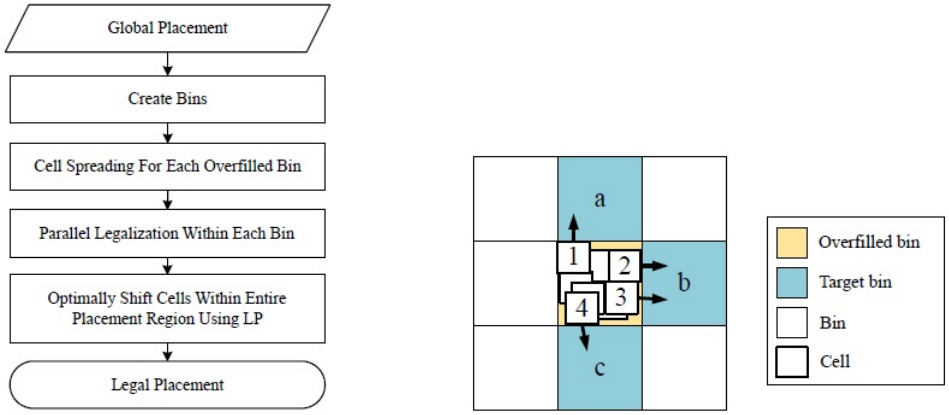
Outras abordagens utilizam a técnica de paralelismo, além do citado Domocus, (HUNG; CHOU; MAK, 2017) apresenta um método de legalização para células *multi-row* em paralelo. A Figura 3.23 ilustra o fluxo geral do método. Primeiro, eles particionam a região do posicionamento para formar partições uniformes e geram uma atribuição inicial das células às partições conforme a solução do posicionamento global. Em seguida, espalham as células de partições com alta densidade para partições vizinhas com densidade relativamente baixas. Depois disso, é realizada uma legalização de forma paralela

Figura 3.22: Fluxo de legalização (CHEN et al., 2020)



em cada partição. Por fim, uma abordagem de LP é realizada para refinar a qualidade da solução. A Figura 3.23 também ilustra como é feito o espalhamento das células de uma partição com alta densidade para partições vizinhas.

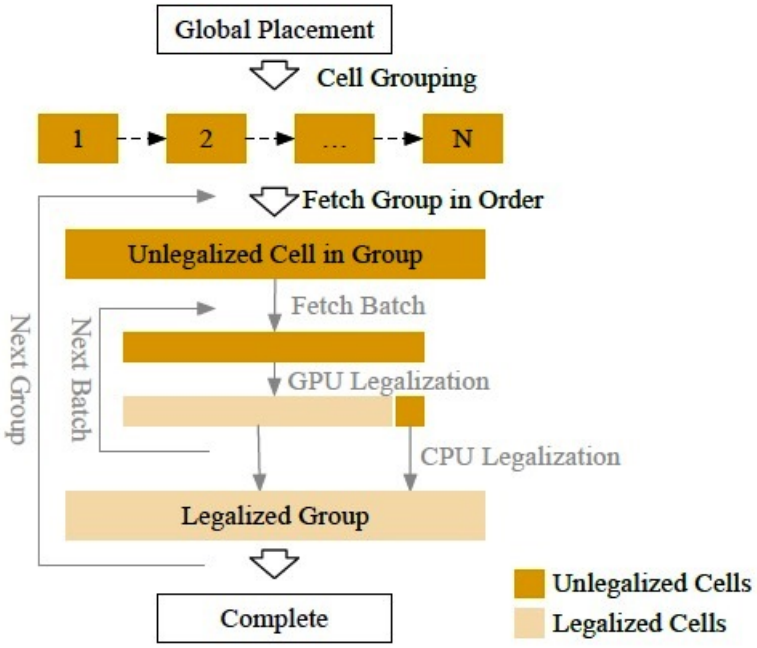
Figura 3.23: Fluxo geral da solução proposta e exemplo de espalhamento de células (HUNG; CHOU; MAK, 2017)



Igualmente, (YANG et al., 2022) apresenta um método que, pela primeira vez, fornece uma legalização ultrarrápida, propõe um algoritmo de legalização para projeto de células *multi-row*, aproveitando os recursos da CPU (unidade central de processamento, do inglês *Central Processing Unit*) e GPU (Unidade de Processamento Gráfico, do inglês *Graphics Processing Unit*). Eles exploram o paralelismo e desenvolvem técnicas de aceleração em GPU dedicadas para algoritmos de legalização. A Figura 3.24 ilustra o fluxo da legalização do método. Para o equilíbrio do tempo de execução e desempenho, eles propõem uma estratégia de agrupamento que separa as células em diferentes grupos por tamanho de células. Outra parte importante da estrutura de legalização é o agendamento de tarefas, que precisa atribuir tarefas de legalização com eficiência para a CPU e/ou GPU.

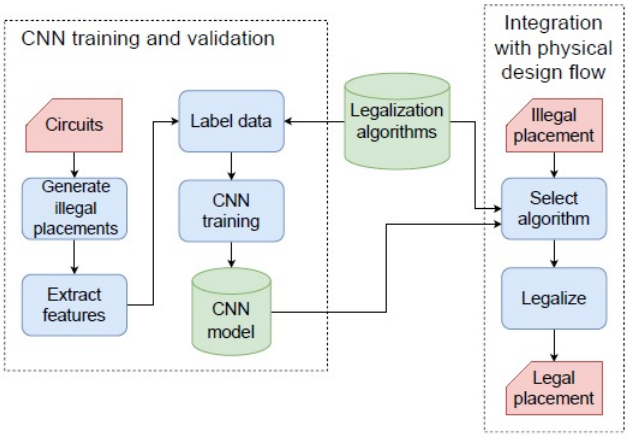
A estratégia principal é atribuir a maioria das tarefas fáceis à GPU e deixar a minoria das células complicadas para a CPU. Para legalizar, primeiro o algoritmo de legalização da CPU coloca determinadas células em uma posição legalizada com movimentos mínimos. Essa etapa se assemelha ao MGL (LI et al., 2018) com técnicas aprimoradas com melhor tempo de execução. Enquanto na legalização da GPU, a hierarquia de *threads* da GPU são acomodadas, cada bloco de *threads* é responsável pela legalização de uma célula.

Figura 3.24: Fluxo da legalização heterogênea (YANG et al., 2022)



Por fim, (NETTO et al., 2021) desenvolveram um *framework* para selecionar um algoritmo de legalização que melhor se adapta a um determinado posicionamento, aumentando assim a qualidade do resultado e evitando tempos de execução excessivamente altos. A Figura 3.25 ilustra o fluxograma do *framework* da estrutura de seleção do algoritmo proposto, composta por duas partes principais: treinamento e validação da rede neural convolucional (CNN do inglês *Convolutional Neural Network*) e integração do fluxo da síntese física. A primeira parte tem algumas etapas. Primeiro, geram posicionamentos ilegais dos circuitos de entrada para aumentar a quantidade de dados de treinamento e validação. Diante disso, o modelo da CNN é treinado. Todo esse processo de treinamento é executado fora do fluxo da síntese física e o modelo CNN é salvo. A própria etapa de seleção do algoritmo é integrada ao fluxo da síntese física. Nesta etapa, utilizam o modelo CNN para prever o melhor algoritmo de legalização a ser utilizado para cada posicionamento ilegal. Em seguido, legalizam o posicionamento utilizando o algoritmo selecionado. Os algoritmos selecionados foram os (ZHU et al., 2020), (DO; WOO; KANG, 2019) e (LI et al., 2018).

Figura 3.25: Fluxograma do *framework* de seleção de algoritmo proposto (NETTO et al., 2021)



3.4 Conclusões

Esta seção apresenta a Tabela 3.2 apresenta as características de cada método apresentando e descrito anteriormente. As características são que alguns legalizadores lidam bloqueios e esses bloqueios podem ser algum macro bloco fixo anteriormente à etapa de posicionamento, uma IP, ou é uma categoria de obstrução no *floorplanning* para inserção de *buffers* que serão posicionados durante a etapa de *timming*. Porém, alguns métodos conseguem lidar com a legalização de macro blocos, ou seja, são macro blocos que chegam a etapa de posicionamento não fixos, são como uma célula. Um exemplo disso é o método do (CONG; XIE, 2008) que realiza uma legalização chamada *two steps*, primeiro legaliza os macro blocos, seguido das células. Regiões de *Fence* uma das restrições mais desafiadoras destacada durante a competição do ISPD (*International Symposium on Physical Design*) 2015 (BUSTANY et al., 2015). Um exemplo para lidar com regiões de *fence* é como o (DO; WOO; KANG, 2019) realiza uma etapa denominada pré-legalização que organiza as células da região de *fence* que posicionamento global colocou fora da região de *fence*. Com a crescente complexidade nos projetos modernos dos circuitos, as células padrão costumam ter diferentes alturas de células com baseado na área, potência e velocidade. (CHOW; PUI; YOUNG, 2016) propõe o primeiro método de legalização de células *multi-row* com o mínimo *displacement*. Alguns métodos visam minimizar o HPWL, enquanto outros se concentram no *displacement*, sendo o médio e/ou máximo, outros lidam simultaneamente com essas medidas, mas poucos lidam com o tema da roteabilidade, uns buscam preservar a roteabilidade dada pelo posicionamento global. Apenas o (LI et al., 2018) lida com os problemas da roteabilidade, que incluem acesso de pinos, pino curto e espaçamento de células, métrica importante para diminuição das violações

reais encontradas no roteamento detalhado. Alguns métodos têm a característica de não iniciar a legalização imediatamente, fazem uma espécie de pré-processamento, a maioria dos métodos analíticos fazem um pré-processamento. Outro exemplo disso, é o (OIKONOMOU et al., 2017) baseado na divisão do circuito em subáreas para ser consideradas independentes. A característica de pós-otimização é como se fosse um refinamento da qualidade da solução. Um exemplo disso é o (DO; WOO; KANG, 2019) que otimiza os resultados utilizando *simulated annealing*. Alguns métodos utilizam a forma de trabalhar em paralelo em alguma etapa. Além do exemplo do (OIKONOMOU et al., 2017), o (HUNG; CHOU; MAK, 2017) legaliza paralelamente em cada partição criada.

Tabela 3.2: Características dos legalizadores

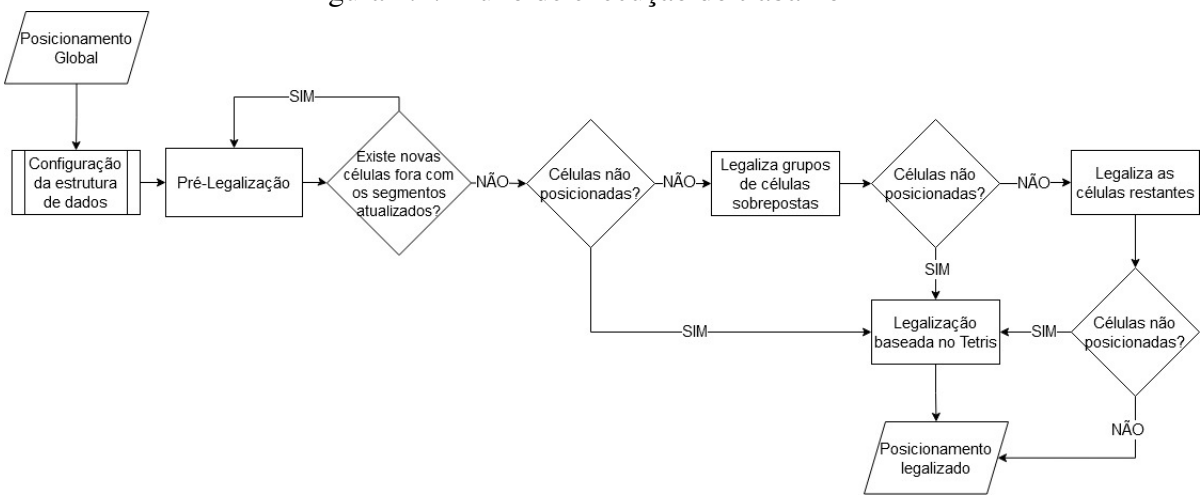
Principal autor	Bloqueios	Macro blocos	Regiões de *fence*	*Multi-row*	Roteabilidade	Pré-processamento	Pós-otimização	Paralelismo
Hill, 2002	X							
Brenner, 2004					X		X	
Kahng, 2004						X		
Spindler, 2008	X				X			
Cong, 2008		X						
Ho, 2010	X	X				X		
Cho, 2010	X					X	X	
Brenner, 2013	X					X	X	
Darav, 2015	X		X			X		X
Puget, 2015	X							
Chow, 2016	X			X				
Wang, 2017				X			X	
Darav, 2017	X				X	X		
Oikonomou, 2017						X		X
Chen, 2017				X		X		
Hung, 2017				X		X	X	X
Li, 2018	X		X	X	X		X	X
Do, 2019			X	X		X	X	
Li, 2019	X			X		X		
Zhu, 2020	X		X	X		X	X	
Chen, 2020	X			X				
Netto, 2021	X		X	X		X		
Chen, 2021	X	X	X	X		X	X	X
Yang, 2022	X		X	X	X	X		X
Wu, 2022	X		X	X		X	X	
Ferreira, 2022	X		X	X		X	X	

4 ALGORITMO PROPOSTO

Nesse capítulo, serão apresentados os detalhes da solução proposta deste trabalho. O capítulo será divido em seções que descrevem as funções que formam o fluxo do algoritmo. Algoritmo implementado no *framework* Rsyn (FLACH et al., 2017). Um *framework* em C ++ gratuito e de código aberto para pesquisa e desenvolvimento de síntese física e com uma *interface* gráfica para o usuário. Ele é projetado para ser muito modular e extensível de forma incremental. Novos componentes podem ser facilmente integrados, tornando o Rsyn cada vez mais valioso como uma estrutura para incentivar a pesquisa na síntese física. Os componentes padrão e de terceiros podem ser misturados através de código ou linguagem de *script* para criar um fluxo de projeto abrangente. O Rsyn foi criado para permitir que os pesquisadores se concentrem no que é realmente importante para suas pesquisas, gastando menos tempo no desenvolvimento da infraestrutura. Nosso método pode legalizar circuitos que possuem *fence regions*, bloqueios e células *multi-row*.

A Figura 4.1 ilustra o fluxo de execução do trabalho que tem como entrada um posicionamento global. Após a entrada, o algoritmo possui um processo pré-definido que configura dados importantes para o restante do fluxo. Após isso, a etapa de pré-legalização, legalizar as células que estão fora de suas regiões corretas. Em seguida, os grupos de células que estão sobrepostas são legalizadas. Por fim, as células restantes são legalizadas. Caso alguma célula não possa ser legalizada em algum desses processos de legalização, esse fluxo é interrompido e a região terá um processo de legalização baseada no método Tetris, por ser um método rápido e garante a legalização.

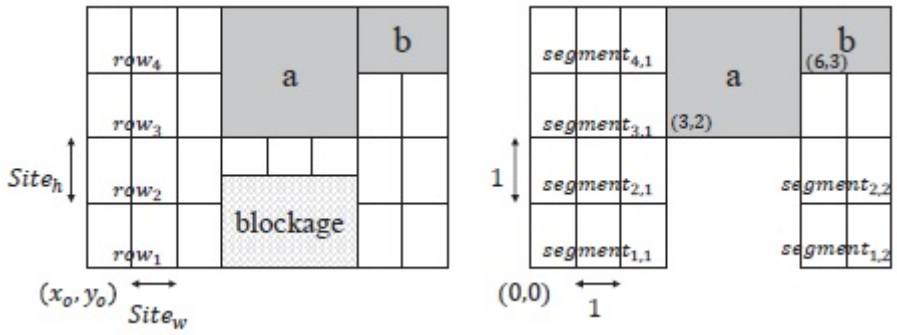
Figura 4.1: Fluxo de execução do trabalho



4.1 Configuração da estrutura de dados

Essa função se baseia na etapa de (CHOW; PUI; YOUNG, 2016), que diferencia segmentos das *rows*. Em que um segmento é definido como sequência contínua sem *sites* bloqueados, como ilustra a Figura 4.2.

Figura 4.2: Exemplo de como as rows são transformadas em segmentos (CHOW; PUI; YOUNG, 2016)



O fluxo inicia-se com a configuração da estrutura de dados descrito no Algoritmo 1. O algoritmo começa criando, na linha 3, o conjunto de *rows*, que possui informações necessárias para as orientações na hora do posicionamento. Na linha 4, a região do circuito é criada a partir da área do circuito. Nas linhas 5–7, caso o circuito possua macro blocos, os segmentos são fragmentados de acordo a área do macro bloco. Nas linhas 8–12, existe uma rotina no caso de existir *fence regions* no circuito. A primeira ação dessa rotina é criar os segmentos para a *fence region* atual. Após isso, essa *fence region* é adicionada na lista de regiões. E por fim, os segmentos da *fence region* atualiza os segmentos do circuito (R_0). A função termina com a identificação e separação das células de acordo com suas regiões, que serão legalizadas nas funções posteriores.

4.2 Pré-legalização

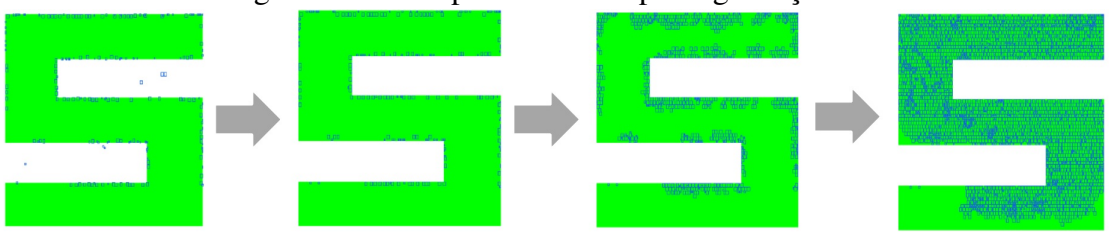
Essa primeira etapa é baseada na etapa de pré-legalização (DO; WOO; KANG, 2019). Essa etapa legaliza apenas células que violam uma *fence region*. As células que violam uma *fence region* são movidas para dentro ou para fora da *fence region*. Nossa abordagem, inicialmente, também legaliza as células que violam uma *fence region*, e também as células que violam a região do circuito. Após isso, legaliza as células que viola as suas regiões de acordo com seus segmentos, atualizados a cada posicionamento. Essa etapa dura até não restar nenhuma célula fora dos segmentos pertencentes a sua

Algoritmo 1: Pseudocódigo da Configuração da estrutura de dados

```
1  R = Lista de regiões;
2  F = lista de fence regions;
3  Cria o conjunto de rows para as restrições de alinhamento;
4  Cria R_0 com os segmentos a partir das bordas do circuito;
5  if Se o circuito possuir macro blocos then
6  |   Atualiza os segmentos de R_0 de acordo com as bordas dos macro blocos;
7  end
8  in 1/A,2/B,3/C,4/D,5/E,6/F
9  for i ← 0 to F do
10 |   Cria segmentos de F_i;
11 |   Adiciona F_i a R;
12 |   Atualiza segmentos de R_0;
13 end
14 Identifica as células de acordo com sua região;
```

região. A cada nova verificação de células fora dos segmentos é chamada *rounds*. Caso alguma célula de alguma não possa ser legalizada, seja na primeira etapa ou em alguns do *rounds*, essa região será legalizada com um método de legalização baseada no Tetris. A Figura 4.3 ilustra uma fence region com suas células violadas pelo posicionamento global, ela sendo preenchida com a primeira verificação, após ela sendo preenchida com alguns *rounds* e com o *round* final desta etapa.

Figura 4.3: Exemplo de como a pré-legalização atua



O algoritmo da pré-legalização é descrito no Algoritmo 2. O algoritmo inicia-se nas linhas 4–11 verificando uma célula de cada vez se ela está fora dos segmentos. Caso uma célula se encontra fora dos segmentos da sua região, essa célula é retirada da lista de todas as células. Nas linhas 12–29 é um *loop* que só termina caso não existam mais células fora dos segmentos. A linha 14 ordena as células fora pelo maior displacement, permitindo que as células mais afastadas tenham prioridade no posicionamento. É nas linhas 15–20 que esse posicionamento é realizado para cada célula, caso não seja possível um posicionamento a função tetrisBased é acionada com a região da célula não posicionada. A função Posiciona será explicada na Seção 4.3. Da mesma maneira, a função tetrisBased será explicada na Seção 4.6. Na linha 21, a lista de células fora é esvaziada.

Nas linhas 22–27, a mesma verificação das células fora, mas agora é com os segmentos fragmentados com o posicionamento das células que estavam fora anteriormente.

Algoritmo 2: Pseudocódigo da pré-legalização

```
1  {C} = Lista de todas as células;
2  C_F = Lista de células fora;
3  R = Lista de regiões;
4  for i ← 0 to R do
5      for j ← 0 to C do
6          Verifica se C está fora dos segmentos de sua região a partir do seu
            posicionamento global;
7          if Se C estiver fora then
8              C_F ← C;
9          end
10     end
11 end
12 for i ← 0 to R do
13     while C_F ≠ 0 do
14         Ordena C_F pelo maior displacement;
15         for j ← 0 to C_F do
16             Posiciona j;
17             if Se C_F não for posicionada then
18                 tetrisBased(R);
19             end
20         end
21         C_F = 0;
22         for j ← 0 to C do
23             Verifica se C está fora dos segmentos de sua região a partir dos
                segmentos atualizados;
24             if Se C estiver fora then
25                 C_F ← C;
26             end
27         end
28     end
29 end
```

4.3 Posiciona célula

A função cuja finalidade é achar um segmento em que a célula caiba com o menor *displacement*. Ao encontrar esse segmento, a função coloca a célula na posição encontrada e atualiza os segmentos.

O Algoritmo 3 se refere a função que posiciona as células, a função se inicia

calculando o *site* e a *row* mais próximo do posicionamento global da célula. Esse cálculo não necessariamente coloca a célula dentro da sua região correta. Após isso, nas linhas 2–3, é realizada uma busca por um ou mais segmentos nessa *row* calculada. Sempre que um segmento é escolhido pelo menor *displacement*, a posição da célula pode ser a mesma calculada ou movida para dentro do segmento. Além de ser movida para dentro, a célula pode ser deslocada para as laterais do segmento para respeitar a regra de espaçamento para evitar espaços vazios. Nas linhas 4–11, há uma condição que procura evitar que a célula seja deslocada para longe do cálculo inicial. Nas linhas 5–10 é uma rotina é feita para procurar por segmentos nas *rows* próximas de modo a tentar diminuir o *displacement* horizontal. Essa rotina é interrompida quando o *displacement* se torna menor do que o número de *rows* visitada. Na linha 12, a célula é posicionada conforme a procura dos segmentos e ela não será mais mexida. Por fim, na linha 13, os segmentos são atualizados, podendo ser encurtados no seu início ou no fim. Outras opções de atualizações seriam a eliminação desse segmento, por conta do encaixe perfeito da célula no segmento, ou fragmentação do mesmo.

Algoritmo 3: Pseudocódigo do Posicionamento Legalizado

```
1  Calcula o site e row mais próxima da posição da célula;
2  Procura por segmentos na row calculada;
3  Posição da célula é determinada de acordo o segmento escolhido;
4  if Se o displacement for maior que uma altura de uma row then
5  |  while displacement > nº de rows visitadas do
6  |  |  Procura por segmentos nas rows próximas;
7  |  |  if Se diminuir o displacement then
8  |  |  |  Posição da célula é determinada de acordo o novo segmento
   |  |  |    escolhido;
9  |  |  end
10 |  end
11 end
12 Posiciona célula;
13 Atualiza segmentos de acordo com a posição da célula;
```

4.4 Legalização de células sobrepostas

Essa função é quando não se tem mais nenhuma célula fora de seus segmentos da sua região. A partir de agora, o fluxo visa identificar células sobrepostas, formar grupos destas células, ordená-las e, por fim, legalizá-las.

O Algoritmo 4 inicia nas linhas 3–6, ordenando as células de cada região pela sua posição pelo eixo x. Esse ordenamento facilita a identificação das sobreposições das células. Após isso, na linha 5, as células são verificadas se estão sobrepostas e agrupadas. A verificação é feita com uma procura a cada duas células, sendo que, essa segunda célula sobreposta é enviada para uma função recursiva em busca de outras sobreposições. As células sobrepostas ao serem identificas, são colocadas em grupos e removidas da lista de células. Após identificar e agrupar as células, nas linhas 7–9, nosso algoritmo ordena os grupos dando prioridade aos grupos de maiores células sobrepostas. Nas linhas 10–14, todos os grupos de cada região tem suas células ordenadas pelo maior *displacement*. Nas linhas 15–22, as células são posicionadas uma a uma, na função abordada na Seção 4.3. Caso uma célula não possa ser posicionada, a função tetrisBased é acionada recebendo a região da célula não posicionada. Essa função será abordada na Seção 4.6.

Algoritmo 4: Pseudocódigo da legalização de células sobrepostas

```
1  C_d = Lista de células dentro;
2  R = Lista de regiões;
3  for i ← 0 to R do
4  |  Ordenar células pela sua posição X;
5  |  Verifica se as células estão sobrepostas e as coloca no mesmo grupo;
6  end
7  for i ← 0 to R do
8  |  Ordena grupos pelo maior tamanho de cada R;
9  end
10 for i ← 0 to R do
11 |  for j ← 0 to C_d do
12 |  |  Ordena as células de cada grupo pelo maior displacement;
13 |  end
14 end
15 for i ← 0 to R do
16 |  for j ← 0 to C_d do
17 |  |  Posiciona C_d;
18 |  |  if Se C_d não for posicionada then
19 |  |  |  tetrisBased(R);
20 |  |  end
21 |  end
22 end
```

4.5 Legalização das células restantes

Essa função é uma das funções finais do fluxo. Caso o circuito consiga legalizar as células nas funções anteriores essa função legaliza as células restantes do circuito. Essas células restantes do circuito são as células que não estavam sobrepostas inicialmente na função 4.4.

O Algoritmo 5 inicia nas linhas 3–7, calculando o *displacement* de cada célula restantes de cada região. Após isso, nas linhas 8–16, a cada região as células são ordenadas pelo maior *displacement* e posicionadas a partir desse ordenamento. Nas linhas 12–14, acontece o mesmo processo de quando uma célula não posicionada, a função tetrisBased será explicada a seguir na Seção 4.6.

Algoritmo 5: Pseudocódigo da legalização das células restantes

```
1  C_r = Lista das células restantes;
2  R = Lista de regiões;
3  for i ← 0 to R do
4  |   for j ← 0 to C_r do
5  |   |   Calcula displacement C_d;
6  |   end
7  end
8  for i ← 0 to R do
9  |   Ordena células da região R pelo maior displacement;
10 |   for j ← 0 to C_r do
11 |   |   Posiciona C_r;
12 |   |   if Se C_r não for posicionada then
13 |   |   |   tetrisBased(R);
14 |   |   end
15 |   end
16 end
```

4.6 Legalização Tetris-based

A função legalização Tetris-based é apenas acionada caso uma célula não encontre um segmento que possa ser posicionada. Ao ser acionada, essa função legaliza toda a região baseada no método de legalização o Tetris (HILL, 2002), explicado no Capítulo 3. A função legalização Tetris-based é muito parecida com a função de posicionamento da Seção 4.3.

O Algoritmo 6 inicia na linha 2, apagando a lista das células da função que não

conseguiu posicionar a célula e, se existir, as células das funções posteriores. Na linha 3, os segmentos da região R são restaurados para os segmentos iniciais da Função 4.1. Na linha 4, o ordenamento é priorizado pela altura das células, caso as alturas forem iguais, a prioridade é a menor soma do x e y da posição global das células. Nas linhas 5–16, todas as células da região R são posicionadas. Na linha 6, calcula o *site* e a *row* mais próximo do posicionamento global da célula. Na linha 7, a célula é alinhada à esquerda do segmento escolhido. Nas linhas 8–15, há uma condição que procura evitar que a célula deslocada longe do cálculo inicial. Nas linhas 9–14 é uma rotina que procura por segmentos nas *rows* próximas de modo a tentar diminuir o *displacement*. Essa rotina é interrompida quando o *displacement* se torna menor do que o número de *rows* visitada. Na linha 16, a célula é posicionada conforme a procura dos segmentos e ela não será mais mexida. Por fim, na linha 17, os segmentos são atualizados.

Algoritmo 6: Pseudocódigo do tetrisBased(R)

```
1  C_t = Lista de todas as células que pertece a R;
2  Apaga a lista de todas as células restantes da região R;
3  Restaura segmentos de R;
4  Ordena células de R;
5  for i ← 0 to C_t do
6      Calcula o site e row mais próxima da posição da C_t ;
7      Posição da célula é determinada à esquerda do segmento de acordo com a
         row calculada;
8      if Se o displacement for maior que uma altura de uma row then
9          while displacement > nº de rows visitadas do
10             Procura por segmentos nas rows próximas;
11             if Se diminuir o displacement then
12                 Posição da célula é determinada à esquerda de acordo o novo
                     segmento escolhido;
13             end
14         end
15     end
16     Posiciona célula;
17     Atualiza segmentos de acordo com a posição da célula;
18 end
```

4.7 Conclusão

Neste capítulo foi apresentado os detalhes da solução proposta. O fluxo de trabalho foi apresentado e detalhado seus passos. Foi descrito cada função do trabalho com

seus pseudocódigos. O fluxo iniciou com uma solução de posicionamento global, legalizando inicialmente as células fora. Após isso, as células sobrepostas são legalizadas e, por fim, as células restantes são legalizadas. Esse fluxo é quando todas as células são posicionadas. Quando há uma célula que não pode ser legalizada, o fluxo termina com uma legalização baseada no método do Tetris. Os resultados desse fluxo serão apresentados no Capítulo 5.

5 RESULTADOS

O objetivo deste capítulo é apresentar os resultados da solução proposta. Para avaliar nosso método, comparamos nossos resultados com os métodos RippleLG (LI et al., 2018) e OpenDP (DO; WOO; KANG, 2019). A escolha desses dois métodos se deve por serem os dois métodos que tivemos acesso. Na comparação deste trabalho, gostaríamos de dar destaque a métrica de *runtime*, para isso é preciso que todos os métodos executem na mesma máquina. Baixamos o código do OpenDP do seu repositório público, enquanto o binário do RippleLG foi fornecido pelo seu autor. Lembrando que, esses dois métodos foram explicados na Seção 3.3. Além disso, será apresentado as configurações da máquina onde nossos experimentos foram realizados. Por fim, serão apresentados as métricas obtidas da implementação e a descrição dos *benchmarks* utilizados para os resultados.

5.1 Ánálises

Nosso método de legalização foi compilado com o g++ versão 6.5.0. Os experimentos foram realizados na seguinte configuração:

- Sistema operacional Ubuntu 16.04 LTS
- Processador Intel® Core™ i7-4790K CPU @ 4.00GHz x 8
- 31.4 GB de memória
- 10 execuções para obtenção de um tempo médio

5.2 Métricas

A métrica de *displacement* foi estimada de quanto as células foram deslocadas a partir do posicionamento global até seu posicionamento legalizado. O *wirelength* é estimado antes da legalização, ou seja, do posicionamento global, após a legalização e até após o posicionamento detalhado. Sendo assim, é possível calcular seu impacto no *wirelength*. O tempo de execução foi calculado utilizando uma extensão de tempo, chamada Multitime (TRATT, 2021), que executa um comando várias vezes e imprime as médias, desvios padrões, tempo mínimo, medianas e tempo máximo. Nossas métricas são tratadas de formas individuais, diferentemente da competição do ICCAD'17. A pontuação da

competição era o produto da pontuação dessas três métricas parciais.

5.3 Descrição dos benchmarks

Os circuitos *benchmarks* fornecem um método de comparar o desempenho de vários processos em diferentes arquiteturas de circuito ou sistemas. Os circuitos *benchmarks* adotados foram disponibilizados pelo ICCAD'17 *contest in multi-deck standard cell legalization and benchmarks*. Esses *benchmarks* possuem células *multi-row*, macro bloco e *fence regions*. A Tabela 5.1 demonstra as informações e a diversidade dos 13 circuitos. São apresentados o número de células de cada circuito e categorizado por suas alturas. O número de macro bloco, o número de *fence regions* e o HPWL do posicionamento global.

Tabela 5.1: Características dos circuitos benchmarks do ICCAD'17

Benchmarks	**Nº de células**				**Macro blocos**	**Fence Regions**	**GP HPWL μm (E+06)**
	1xH	**2xH**	**3xH**	**4xH**			
des_perf_a_md1	103589	4699	0	0	4	4	2,16
des_perf_a_md2	105030	1086	1086	1086	4	4	2,18
des_perf_b_md1	106782	5862	0	0	0	12	2,11
des_perf_b_md2	101908	6781	2260	1695	0	12	2,14
edit_dist_1_md1	118005	7994	2664	1998	0	0	4,01
edit_dist_a_md2	115066	7799	2599	1949	6	1	5,10
edit_dist_a_md3	119616	2599	2599	2599	6	1	5,33
fft_2_md2	28930	2117	705	529	0	0	0,45
fft_a_md2	27431	2018	672	504	6	0	1,09
fft_a_md3	28609	672	672	672	6	0	0,95
pci_bridge32_a_md1	26680	1792	597	448	4	3	0,46
pci_bridge32_a_md2	25239	2090	1194	994	4	3	0,57
pci_bridge32_b_md1	26134	1756	585	439	6	3	0,66
pci_bridge32_b_md2	28038	292	292	292	6	3	0,58
pci_bridge32_b_md3	27452	292	585	585	6	3	0,58

Os circuitos dos *benchmarks* são diversos para exemplificar as características, o circuito que possui maior proporção de células *multi-rows* tem cerca de 15%. Por outro lado, os circuitos que possuem as menores proporções de células *multi-rows* tem apenas 3%. O circuito que possui o maior número de células tem mais de 130 mil células. Por outro lado, os circuitos que possuem o menor número de células tem menos de 29 mil células. Sobre os números de macro blocos, os circuitos que possuem o maior número de macro blocos são de 6 macro blocos. Enquanto três circuitos não possuem nenhum macro bloco. Os circuitos que possuem o maior número de *fence regions* possuem 12 *fence regions* cada. Enquanto três circuitos não possuem nenhuma *fence region*. Outro detalhe é que apenas dois circuitos não possuem nenhum macro bloco ou *fence region*.

5.4 Resultados do fluxo

Todas as tabelas abaixo, vão ilustrar as etapas do fluxo detalhado no Capítulo 4. O circuito des_perf_b_md1 foi o circuito escolhido para ilustrar os resultados deste fluxo. Esse circuito é o único que teve, em algumas regiões, células que não conseguiram ser posicionadas com a primeira forma de legalização.

A Tabela 5.2 ilustra os resultados da primeira parte do fluxo que identifica as células por região. A tabela também ilustra os resultados das células dentro e fora da sua região correta a partir do posicionamento. Cada região possui sua ocupação e as células são categorizadas em *single-row* e *multi-row*.

Tabela 5.2: Resultados do circuito des_perf_b_md1

Região	Ocupação (%)	Células dentro		Células fora	
		Single	Multi	Single	Multi
0	56	94280	0	1778	0
1	68	405	133	17	38
2	78	466	219	51	39
3	81	515	283	53	48
4	96	233	206	34	29
5	96	230	198	33	41
6	91	400	331	59	71
7	93	611	394	43	33
8	69	495	270	54	57
9	87	503	278	57	57
10	10	370	199	14	3
11	12	442	239	10	6
12	64	5436	2521	193	169

Nesta tabela podemos destacar que, apesar do circuito possuir 55% de ocupação, as regiões possuem uma variedade de ocupação que variam de 10% até mais de 95%. Outro destaque seria que o posicionador global, além da violação com as sobreposições, viola as regiões corretas das células. A proporção de células fora da sua região varia de menos de 2%, como na Região 0, sendo a região do circuito, a mais de 15%, como na região 6.

A Tabela 5.3 ilustra os resultados das células que estão fora da sua região correta. Os resultados são divididos em estimativa e o posicionamento legalizado. Os dois resultados apresentam a média e o máximo *displacement* de cada região. Os resultados de estimativa são os mínimos *displacement* de cada célula, podendo duas ou mais células poderem ocupar o mesmo lugar. Enquanto o posicionamento legalizado são as células

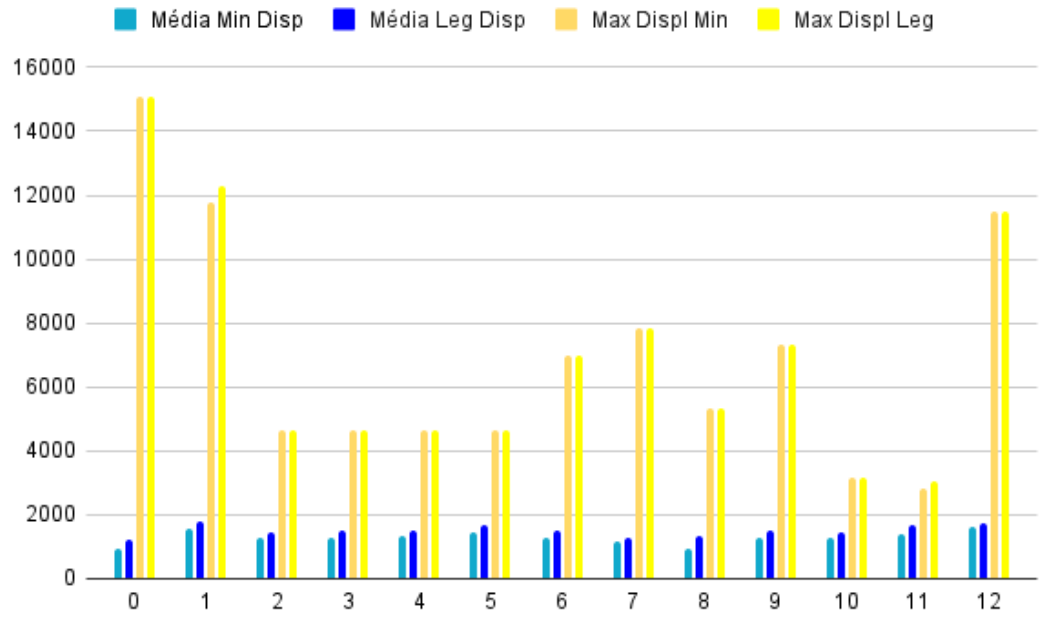
legalizadas uma-a-uma, a partir do ordenamento do maior *displacement*, e com a atualização dos segmentos a cada posicionamento.

Tabela 5.3: Resultados da estimativa e da legalização da pré-legalização a partir do posicionamento global

			Estimativa		Legalizado	
Região	Single	Multi	Média Min Disp	Max Displ Min	Média Leg Disp	Max Displ Leg
0	1778	0	940	15087	1248	15087
1	17	38	1577	11780	1778	12310
2	51	39	1256	4636	1427	4636
3	53	48	1296	4660	1495	4660
4	34	29	1359	4648	1513	4648
5	33	41	1456	4621	1660	4621
6	59	71	1285	7007	1488	7007
7	43	33	1147	7812	1258	7812
8	54	57	939	5321	1332	5321
9	57	57	1277	7301	1514	7301
10	14	3	1273	3141	1432	3141
11	10	6	1420	2840	1683	3021
12	193	169	1614	11515	1735	11515

Na Figura 5.1 é possível observar que, o máximo *displacement* de quase todas as regiões possuem o mesmo da estimativa. As exceções são as regiões 1 e 11, com um acréscimo de 4,5% e 6,37%, respectivamente. Enquanto a média dos *displacement* teve uma variação de acréscimo de 7% a 41%, com uma média de 17%. Esse acréscimo da diferença da estimativa e do posicionamento legalizado se dá pelo motivo que nem todas as células são posicionadas no seu mínimo *displacement* estimado.

Figura 5.1: Displacements médios e máximos a partir das suas estimativas e após legalizados



A Tabela 5.4 ilustra os resultados das células que ficam fora a cada atualização dos segmentos. A tabela contém os *rounds* de cada região, e também o número de células *single-row* e *multi-row*. A média e o máximo *displacement* legalizado de cada região também são ilustradas. E por fim, o número de células que não conseguiram ser posicionadas. Os *rounds* é a quantidade de rodadas que cada região identificou células fora dos segmentos e precisou legalizar a(s) célula(s).

Tabela 5.4: Resultados da pré-legalização

Região	#Rounds	Single	Multi	Média Disp Leg	Max Disp Leg	Não posicionadas
0	40	9790	0	1136	4982	0
1	10	392	130	2008	17590	0
2	9	458	212	2498	23609	0
3	9	515	280	3914	57774	0
4	6	233	192	5010	54361	14
5	6	230	186	6114	62523	12
6	7	400	330	4763	51270	1
7	8	611	371	3369	51725	23
8	11	487	267	3675	27865	0
9	7	503	278	6645	52763	0
10	2	1	0	728	728	0
11	4	8	8	1560	4579	0
12	33	4142	2071	1530	11439	0

Nesta tabela podemos destacar que a média das células posicionadas por *rounds* variam de menos de uma célula por *round* a quase 245 células por *round*, as regiões 10 e a 0, respectivamente. Outro destaque seria a proporção das células posicionadas nessa etapa. As proporções variam de 0,17% a 90,84%, as regiões 10 e 7, respectivamente. Porém, a região 7, assim como as regiões 4, 5 e 6, tiveram células não posicionadas pelo principal método de legalização do algoritmo. Essas regiões terão seus resultados ilustrados com o método baseado no Tetris, explicado na Seção 4.6.

A Tabela 5.5 ilustra as células que ficaram nos segmentos do circuito e quantidade de células que estão sobrepostas. A tabela também contém apenas as regiões que não tiveram células não posicionadas, o número das células *single-row* e *multi-row* das células que estão dentro e das células sobrepostas. Junto a isso, a média e o máximo *displacement* e, por fim, as número de células restantes.

Nesta tabela é possível destacar que, a proporção das células dentro que estão sobrepostas é acima dos 50%, com a exceção das regiões 3 e 9. Sendo que a região 3 possui 3 células *multi-row* que não estão sobrepostas. Enquanto a região 9 não possui mais nenhuma célula, pois, já foram legalizadas nas etapas anteriores. Em relação às células sobrepostas, a proporção varia de 51% a 80%, sendo as regiões 11 e 2, respectivamente.

Tabela 5.5: Resultados das células dentro e o número de células sobrepostas

	Células dentro		Células sobrepostas				
Região	Single	Multi	Single	Multi	Média Leg Disp	Max Displ Leg	Células restantes
0	84490	0	66475	0	1009	4263	18015
1	13	3	10	0	1044	1866	6
2	8	7	7	5	1251	2163	3
3	0	3	0	0	0	0	3
8	8	3	4	2	1121	1374	5
9	0	0	0	0	0	0	0
10	369	199	194	145	1055	2966	229
11	434	231	208	133	1062	2551	324
12	1294	450	982	351	1200	5592	411

Em relação ao *displacement*, a média do *displacement* legalizado são quase as mesmas entre as regiões, com a exceção também das regiões 3 e 9, a média vária entre 1009 a 1251. Quanto ao máximo *displacement* legalizado, as regiões que possuem maiores resultados são as regiões 12 e 0, com 5592 e 4263, respectivamente. Em contraste, as regiões 8 e 1, possuem os menores resultados, com 1374 e 1866, respectivamente. É possível fazer uma relação com o número de células sobrepostas e o número de células *multi-row* sobrepostas com o resultado do máximo *displacement*.

A Tabela 5.6 ilustra as regiões que possuem células restantes, separadas em *single-row* e *multi-row*, e que não possuem células não posicionadas. Os resultados também são divididos em estimativa e o posicionamento legalizado. Às duas categorias apresentam a média e máximo *displacement*.

Tabela 5.6: Resultados das células restantes

			Células restantes			
			Estimativa		Legalizado	
Região	Single	Multi	Média Leg Displ	Max Leg Displ	Média Leg Displ	Max Leg Displ
0	18015	0	777	3793	870	3793
1	3	3	717	1949	817	1949
2	1	2	457	826	457	826
3	0	3	1242	1902	1242	1902
8	4	1	641	1919	641	1919
10	175	54	753	3248	801	3248
11	226	98	772	2916	838	2916
12	312	99	938	7516	1117	7516

Nesta tabela podemos destacar que apenas 8 das 13 regiões tiveram células restantes para legalizar. A proporção de células restantes com as células dentro inicialmente variam de 0,37% a 47%, sendo as regiões 3 e 11, respectivamente. Com esses resultados é possível suspeitar que a região 3 possui áreas congestionadas, porém, conseguindo legalizar todas as células. Por outro lado, as regiões 10 e 11, possuem a proporção de células restantes com as células dentro inicialmente acima dos 40%. É possível suspeitar

que essas duas regiões possuem áreas dispersas. Outro destaque é pequena diferença, de algumas regiões, entre a estimativa e o posicionamento legalizado. A média ficou com um aumento de um pouco mais de 5%, a região 12 é a região que possui o maior aumento com cerca de 19%. Enquanto as regiões 2,3 e 8 não possuem aumento em relação à estimativa. Por fim, outra métrica estimada foi o máximo *displacement*. Em todas as regiões o máximo *displacement* se manteve sem aumento ao legalizar as células.

A Tabela 5.7 ilustra as regiões que tiveram células que não posicionadas nas etapas anteriores. A essas regiões foram aplicadas um método de legalização baseado no método de legalização Tetris (HILL, 2002). A tabela ilustra o número de células *single-rows* e *multi-rows*, a média e o máximo *displacement*.

Tabela 5.7: Resultados da função Tetris-based

			Estimativa inicial		Tetris-based	
Região	Single	Multi	Média Min Disp	Max Displ	Média Leg Disp	Max Displ Leg
4	267	235	1096,5	3367,5	13653	59993
5	263	239	1124,5	3335,5	13359	48713
6	459	402	1058,5	4547	16346	51526
7	654	427	962	4946	16433	94000

Nesta tabela podemos observar que o médio e o máximo *displacement* aumentam significativamente em comparação com as estimativas do início da etapa, seja com as células de dentro da região ou de fora a partir do posicionamento global. O aumento do médio *displacement* ficou, em média, cerca de 14x maior. A região 7 possui o maior aumento, com cerca de 17x. A região 5 possui o menor aumento, com cerca de 11x. E cerca de 15x teve de aumento, em média, o máximo *displacement*. A região 7 também ficou com o máximo *displacement*, cerca de 19x maior. E a região 6, ficou com o menor aumento, com cerca de 11x. Apesar desses aumentos, essas regiões não ficaram com nenhuma célula não posicionada, diferentemente do método anterior.

5.5 Resultados finais

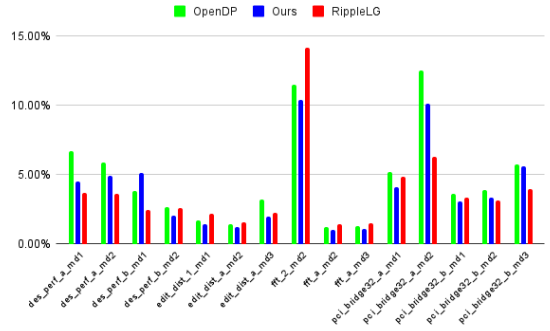
A Tabela 5.8 ilustra os resultados dos 15 circuitos comparando nosso trabalho com outros dois trabalhos, OpenDP e RippleLG. A tabela possui informações como o Δ HPWL, que seria o aumento do HPWL no circuito legalizado, o máximo e médio *displacement* e tempo de execução para legalizar o circuito.

A Figura 5.2a ilustra os gráficos dos resultados sobre ΔHPWL de cada método. Na figura é possível observar que nosso trabalho possui o melhor resultado em 9 dos 15

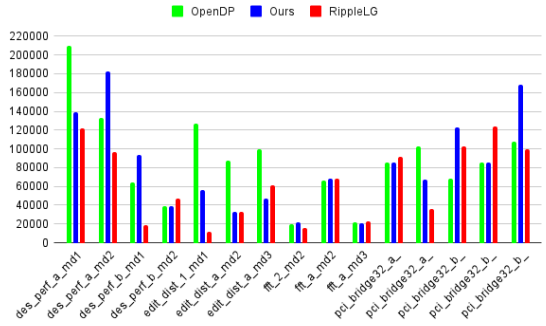
Tabela 5.8: Resultados

	Δ HPWL (%)			Max displacement			Avg Displacement			Runtime (s)		
Benchmarks	OpenDP	RippleLG	Ours	OpenDP	RippleLG	Ours	OpenDP	RippleLG	Ours	OpenDP	RippleLG	Ours
des_perf_a_md1	6.70	3.66	4.52	209368	121460	139471	1813	1406	1562	8.80	204.86	4.06
des_perf_a_md2	5.90	3.60	4.93	132798	96110	182977	1704	1389	1633	6.10	186.88	4.84
des_perf_b_md1	3.85	2.46	5.14	64247	18910	94000	1290	926	1503	4.94	156.51	4.17
des_perf_b_md2	2.64	2.59	2.00	38720	46648	38678	1279	1289	1244	4.17	193.28	3.91
edit_dist_1_md1	1.73	2.16	1.42	127218	11482	56188	1304	1442	1257	3.54	160.55	10.30
edit_dist_a_md2	1.44	1.53	1.20	87003	32800	32800	1313	1331	1264	3.75	199.72	5.74
edit_dist_a_md3	3.19	2.27	1.96	100003	61215	46600	2178	1773	1782	4.07	241.76	5.17
fft_2_md2	11.52	14.20	10.39	20170	16141	22260	1806	2088	1932	0.84	240.66	1.01
fft_a_md2	1.19	1.39	1.04	66699	68695	68695	1081	1172	1018	1.94	37.06	1.05
fft_a_md3	1.32	1.48	1.10	21476	22518	20318	1014	1088	969	1.79	21.44	0.90
pci_bridge32_a_md1	5.19	4.82	4.10	85143	91857	85143	1492	1481	1438	1.58	51.72	0.68
pci_bridge32_a_md2	12.56	6.31	10.12	102385	36131	67222	2632	1897	2512	1.89	57.50	0.78
pci_bridge32_b_md1	3.61	3.37	3.07	68004	102815	123310	1489	1516	1448	2.51	50.15	0.83
pci_bridge32_b_md2	3.88	3.11	3.36	85808	123495	85808	1362	1317	1304	3.25	49.62	0.79
pci_bridge32_b_md3	5.77	3.96	5.61	107536	99645	167936	1688	1480	1655	3.80	49.80	0.81

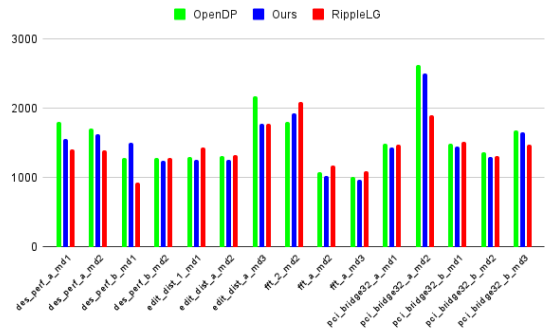
Figura 5.2: Resultados obtidos



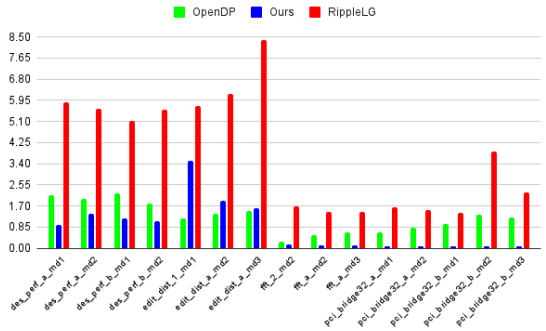
(a) Δ HPWL



(b) Max Displacement



(c) Average Displacement



(d) Runtime

circuitos. Em apenas um circuito nosso trabalho possui o pior resultado em comparação com os outros dois trabalhos. Além de possuir a pior média do Δ HPWL, o OpenDP possui o pior resultado em 9 de 15 circuitos. O RippleLG, apesar de possuir a melhor média, tem apenas o melhor resultado em 6 circuitos e também possui o pior resultado em outros 5 circuitos.

Outra métrica que podemos observar é o máximo *displacement*. A Figura 5.2b ilustra os gráficos dos resultados sobre o máximo *displacement*. Nos gráficos é possível observar que o RippleLG possui melhores resultados em 8 circuitos, enquanto nosso

trabalho possui o menor máximo *displacement* em 6 circuitos e o OpenDP têm em apenas 4 circuitos. Em alguns circuitos, nosso trabalho possui resultados iguais com os melhores máximos *displacement*. No circuito edit_dist_a_md2, nosso trabalho possui o menor máximo *displacement* com o RippleLG. Nos circuitos pci_bridge32_a_md1 e pci_bridge32_b_md2, nosso trabalho possui o melhor resultado com o OpenDP. Nosso trabalho possui o pior máximo *displacement* em 6 circuitos. Enquanto os outros dois trabalhos possuem o pior máximo *displacement* em 5 circuitos. Sendo que no circuito fft_a_md2 nosso trabalho possui o pior máximo *displacement* com o RippleLG.

Além do máximo *displacement*, o médio *displacement* é outra métrica que devemos observar. A Figura 5.2c ilustra os gráficos dos resultados sobre o *displacement* médio. Nesses gráficos é possível observar que nosso trabalho possui o menor *displacement* médio em 8 dos 15 circuitos. Enquanto o RippleLG possui o menor *displacement* médio em apenas 6 circuitos. O OpenDP possui o menor *displacent* médio no circuito fft_2_md2. Nosso trabalho só possui o pior *displacement* médio no des_perf_b_md1. Em contrapartida, RippleLG e OpenDP, possuem os piores *displacement* médios em 7 circuitos cada um.

A última métrica a se observar é o *runtime*. A Figura 5.2d ilustra os gráficos dos resultados sobre *runtime*. Nos gráficos é possível observar que nosso trabalho possui superioridade e possui o menor *runtime* em 12 dos 15 circuitos. O OpenDP possui os outros 3 melhores resultados. O RippleLG possui os piores resultados de *runtime* nos 15 circuitos.

A Tabela 5.9 ilustra os circuitos, o número total de *rounds* de cada circuito, a média de *rounds* pelo número de regiões de cada circuito e o maior *runtime* de todas as regiões de cada circuito. Essa tabela explica o porquê do nosso algoritmo não possuir melhores resultados em *runtime* em três circuitos. Os circuitos edit_dist_1_md1, edit_dist_a_md2 e edit_dist_a_md3 são os maiores circuitos em média de *rounds* isso identifica um gargalo em nosso algoritmo.

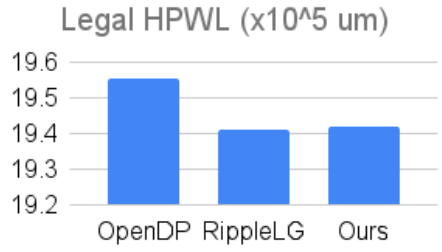
Fazendo uma análise final dos resultados na Figura 5.3, escolhemos esses 3 quesitos para mostrar que nosso método teve melhor desempenho. Onde se manteve perto da melhor média em relação ao impacto do HPWL e o *displacement* médio com um *runtime* bem menor.

Na Figura 5.3a ilustra os gráficos das médias dos resultados sobre o ΔHPWL, onde nenhum trabalhou chegou a ultrapassar em 5% na média. Nos gráficos é possível observar que nosso trabalho está em igualdade em relação ao trabalho do RippleLG,

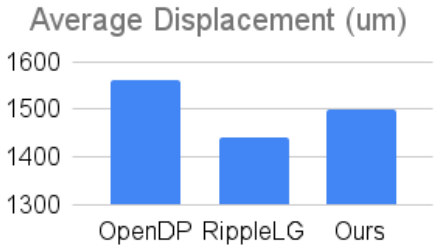
Tabela 5.9: Resultados da etapa de pré-legalização de todos os circuitos

Design	total rounds	average rounds	runtime máximo (s)
des_perf_a_md1	187	37,4	0,85
des_perf_a_md2	280	56	1,21
des_perf_b_md1	152	11,69	0,30
des_perf_b_md2	276	21,23	0,96
edit_dist_1_md1	300	300	2,9
edit_dist_a_md2	142	71	1,70
edit_dist_a_md3	129	64,5	1,41
fft_2_md2	94	94	0,14
fft_a_md2	12	12	0,02
fft_a_md3	12	12	0,02
pci_bridge32_a_md1	59	11,8	0,05
pci_bridge32_a_md2	75	15	0,05
pci_bridge32_b_md1	68	17	0,05
pci_bridge32_b_md2	77	19,25	0,03
pci_bridge32_b_md3	77	19,25	0,04

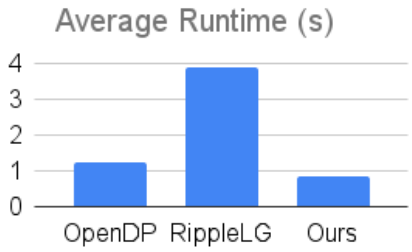
Figura 5.3: Média dos resultados obtidos



(a) Legal HPWL



(b) Average Displacement



(c) Average Runtime

enquanto o método do OpenDP está apenas um pouco acima.

O médio *displacement* é outra métrica que devemos observar. A Figura 5.3b ilustra as médias dos resultados sobre o *displacement* médio. Todos os métodos possuem médias menores que o tamanho de uma *row* (2000μ), ou seja, pouco movimento de células para legalizá-las. Nos gráficos é possível observar que o RippleLG possui uma média menor de *displacement*, mas sem superioridade.

A última métrica que vamos observar é o *runtime*. A Figura 5.3c ilustra a média dos resultados sobre o *runtime*. Onde nenhum trabalho chegou a ultrapassar em 4s na média para legalizar um circuito, mas é notável que nosso trabalho possua superioridade em obter uma média menor de *runtime*. OpenDP está um pouco acima na média, enquanto o RippleLG está consideravelmente maior.

6 CONSIDERAÇÕES FINAIS

Nesse trabalho, duas revisões foram apresentadas. A primeira foi a revisão do projeto digital no Capítulo 2. Nesse capítulo foram revisados os fluxos de células padrão, síntese física, posicionamento, seguido da apresentação dos conceitos básicos de um circuito para melhor entendimento do problema da legalização.

A segunda revisão foi abordar o problema da legalização no Capítulo 3. Na sequência, foi apresentado os novos desafios da legalização que entre eles, a preocupação com a roteabilidade, região de *fence* e células *multi-row*. Após isso, foram apresentados os métodos de legalização e categorizados em métodos heurísticos e analíticos. A abordagem heurística pode alcançar uma solução em um tempo prático, mas não garante uma solução ideal. Enquanto, os métodos analíticos convertem o problema de legalização em outro problema que pode ser resolvido usando *solvers* existentes. Diferentemente do método heurístico, a abordagem analítica garante a solução ótima se a *fence region* for simples, mas não quando for complexa.

A maioria dos algoritmos de legalização ainda se baseiam nos algoritmos clássicos, Tetris e Abacus, que tem como foco principal corrigir a sobreposição de células, em vez de lidar com outros problemas também. Esse problema foi ilustrado na Tabela 3.2 em que, apenas, alguns algoritmos lidam com a roteabilidade, *fence region* ou células *multi-row*. Lidar com o problema da roteabilidade é um importante desafio, pois o fluxo de células padrão não é linear. Ou seja, um bom posicionamento não é garantia de um roteamento satisfatório, talvez seja necessária realizar a etapa de posicionamento novamente.

Neste trabalho também tem a contribuição de um algoritmo rápido de legalização de células padrão que lida com *fence regions* e células *multi-row*. Nosso algoritmo move, primeiramente, as células que violam a sua região para sua região correta. Ao mover as células violadas, elas são alinhadas e fixas. Para evitar o aumento do máximo *displacement*, as células são ordenadas pelo seu maior *displacement* a partir de um cálculo primário. Após isso, é identificado grupos de células sobrepostas. Esses grupos são legalizados a partir do maior tamanho de cada região. Por fim, as células restantes são legalizadas. Caso em algumas dessas etapas alguma célula não possa ser legalizada, o algoritmo legaliza todas as células da região numa legalização baseada no algoritmo do Tetris.

Por fim, este trabalho deixa o problema da legalização como um problema que

ainda requer investigação. Acreditamos que seja preciso investigar em outras abordagens para legalização, como o caso de *machine learning*, como foi o caso do trabalho do (NETTO et al., 2021). Acreditamos também que o paralelismo pode ser mais explorado para acelerar o processo de posicionamento, como foi o caso do trabalho do (YANG et al., 2022) que explorou o paralelismo da GPU. Além de quem for trabalhar com legalização, precisará ter o conhecimento algoritmo e compreensão das soluções para esse problema, além do conhecimento do fluxo de células padrão. Desta forma, esse trabalho atinge o seu objetivo que é apresentar o problema, o histórico sobre algoritmos de legalização e desenvolver uma nova solução.

REFERÊNCIAS

AHUJIA, R.; MAGNANTI, T. L.; ORLIN, J. B. Network flows: Theory, algorithms and applications. **New Jersey: Rentice-Hall**, 1993.

ALEGRETTI, C. G. P. Analytical logical effort formulation for local sizing. 2013.

ALMEIDA, A. F. C. d. Investigating techniques to reduce soft error rate under single-event-induced charge sharing. 2014.

BAEK, S.-H. et al. Ultra-high density standard cell library using multi-height cell structure. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Smart Structures, Devices, and Systems IV**. [S.l.], 2008. v. 7268, p. 72680C.

BAI, Z.-Z. Modulus-based matrix splitting iteration methods for linear complementarity problems. **Numerical Linear Algebra with Applications**, Wiley Online Library, v. 17, n. 6, p. 917–933, 2010.

BAVARESCO, S. On-silicon testbench for validation of soft logic cell libraries. 2008.

BRENNER, U. Bonnplace legalization: Minimizing movement by iterative augmentation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 32, n. 8, p. 1215–1227, 2013.

BRENNER, U.; PAULI, A.; VYGEN, J. Almost optimum placement legalization by minimum cost flow and dynamic programming. In: **Proceedings of the 2004 international symposium on Physical design**. [S.l.: s.n.], 2004. p. 2–9.

BUSTANY, I. S. et al. Ispd 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In: **Proceedings of the 2015 Symposium on International Symposium on Physical Design**. [S.l.: s.n.], 2015. p. 157–164.

CHEN, J. et al. Nblg: A robust legalizer for mixed-cell-height modern design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, 2021.

CHEN, J. et al. Toward optimal legalization for mixed-cell-height circuit designs. In: **Proceedings of the 54th Annual Design Automation Conference 2017**. [S.l.: s.n.], 2017. p. 1–6.

CHEN, J. et al. A robust modulus-based matrix splitting iteration method for mixed-cell-height circuit legalization. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, ACM New York, NY, USA, v. 26, n. 2, p. 1–28, 2020.

CHO, M. et al. History-based vlsi legalization using network flow. In: **Proceedings of the 47th Design Automation Conference**. [S.l.: s.n.], 2010. p. 286–291.

CHOW, W.-K.; PUI, C.-W.; YOUNG, E. F. Legalization algorithm for multiple-row height standard cell design. In: IEEE. **2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)**. [S.l.], 2016. p. 1–6.

CONG, J.; XIE, M. A robust mixed-size legalization and detailed placement algorithm. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 27, n. 8, p. 1349–1362, 2008.

DARAV, N. K. et al. A fast, robust network flow-based standard-cell legalization method for minimizing maximum movement. In: **Proceedings of the 2017 ACM on International Symposium on Physical Design**. [S.l.: s.n.], 2017. p. 141–148.

DARAV, N. K. et al. Iccad-2017 cad contest in multi-deck standard cell legalization and benchmarks. In: IEEE. **2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.], 2017. p. 867–871.

DARAV, N. K. et al. High performance global placement and legalization accounting for fence regions. In: IEEE. **2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.], 2015. p. 514–519.

DO, S.; WOO, M.; KANG, S. Fence-region-aware mixed-height standard cell legalization. In: **Proceedings of the 2019 on Great Lakes Symposium on VLSI**. [S.l.: s.n.], 2019. p. 259–262.

FLACH, G. et al. Rsyn: An extensible physical synthesis framework. In: **Proceedings of the 2017 ACM on International Symposium on Physical Design**. New York, NY, USA: ACM, 2017. (ISPD '17), p. 33–40. ISBN 978-1-4503-4696-2. Disponível em: <http://doi.acm.org/10.1145/3036669.3038249>.

FLACH, G. A. Clock mesh optimization. 2010.

FLACH, G. A. Discrete gate sizing and timing-driven detailed placement for the design of digital circuits. 2015.

FOGAÇA, M. P. A new quadratic formulation for incremental timing-driven placement. 2016.

GEISSLER, F. d. A. Metodologia de injeção de falhas baseada em emulação de processadores. 2014.

HENTSCHKE, R. F. Algorithms for wire length improvement of vlsi circuits with concern to critical paths. 2007.

HILL, D. **Method and system for high speed detailed placement of cells within an integrated circuit design**. [S.l.]: Google Patents, 2002. US Patent 6,370,673.

HO BRANDON CHESTER, C. H. . R. S. J. **The iPhone 6 Review**. 2014. Disponível em: <https://www.anandtech.com/show/8554/the-iphone-6-review/2>.

HO, T.-Y.; LIU, S.-H. Fast legalization for standard cell placement with simultaneous wirelength and displacement minimization. In: SPRINGER. **IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip**. [S.l.], 2010. p. 291–311.

HUNG, C.-Y.; CHOU, P.-Y.; MAK, W.-K. Mixed-cell-height standard cell placement legalization. In: **Proceedings of the on Great Lakes Symposium on VLSI 2017**. [S.l.: s.n.], 2017. p. 149–154.

JOHANN, M. d. O. Novos algoritmos para roteamento de circuitos vlsi. 2001.

KAHNG, A. B. et al. **VLSI Physical Design: From Graph Partitioning to Timing Closure**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2011. ISBN 9789048195909.

KAHNG, A. B.; MARKOV, I. L.; REDA, S. On legalization of row-based placements. In: **Proceedings of the 14th ACM Great Lakes symposium on VLSI**. [S.l.: s.n.], 2004. p. 214–219.

KRAVETS, V. N. Constructive multi-level synthesis by way of functional properties. **Doctoral Dissertation**, 2001.

LI, H. et al. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In: **Proceedings of the 55th Annual Design Automation Conference**. [S.l.: s.n.], 2018. p. 1–6.

LI, X. et al. Analytical mixed-cell-height legalization considering average and maximum movement minimization. In: **Proceedings of the 2019 International Symposium on Physical Design**. [S.l.: s.n.], 2019. p. 27–34.

LIN, M. P.-H.; HSU, C.-C.; CHANG, Y.-T. Recent research in clock power saving with multi-bit flip-flops. In: IEEE. **2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)**. [S.l.], 2011. p. 1–4.

MARQUES, F. d. S. Technology mapping for virtual libraries based on cells with minimal transistor stacks. 2008.

MATOS, J. M. A. d. Graph based algorithms to efficiently map vlsi circuits with simple cells. 2018.

MATTII, L. et al. Ir-drop aware design & technology co-optimization for n5 node with different device and cell height options. In: IEEE. **2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.], 2017. p. 89–94.

MEINHARDT, C. Geração de leiautes regulares baseados em matrizes de células. 2006.

MONTEIRO, J. L. Algoritmo de posicionamento analítico-detalhado guiado a caminhos críticos. 2014.

MONTEIRO, J. L. Algorithms to improve area density utilization, routability and timing during detailed placement and legalization of vlsi circuits. 2019.

MOORE, G. E. et al. Progress in digital integrated electronics. In: MARYLAND, USA. **Electron devices meeting**. [S.l.], 1975. v. 21, p. 11–13.

NETTO, R. et al. Algorithm selection framework for legalization using deep convolutional neural networks and transfer learning. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, 2021.

NEUTZLING, A. Thereshold logic technology mapping for emerging nanotechnologies. 2017.

NUNES, L. d. M. Redução de congestionamento em roteamento global de circuitos vlsi. 2013.

OIKONOMOU, P. et al. Domocus: Lock free parallel legalization in standard cell placement. In: IEEE. **2017 6th International Conference on Modern Circuits and Systems Technologies (MOCAST)**. [S.l.], 2017. p. 1–4.

PARIS, L. A. d. Análise e mitigação dos efeitos da eletromigração em interconexões metálicas de circuitos integrados. 2017.

PINTO, F. d. A. Posicionamento visando redução do comprimento das conexões. 2011.

PLÁCIDO, H. Posicionamento global de células em circuitos vlsi. 2016.

POSSER, G. Electromigration aware cell design. 2015.

PUGET, J. C. et al. Jezz: An effective legalization algorithm for minimum displacement. In: **Proceedings of the 28th Symposium on Integrated Circuits and Systems Design**. [S.l.: s.n.], 2015. p. 1–5.

REIMANN, T. J. Roteamento global de circuitos vlsi. 2013.

ROSER, M.; RITCHIE, H. **Technological Progress @ONLINE**. 2022. Disponível em: <https://ourworldindata.org/technological-progress>.

SANTOS, C. L. d. Verificação e otimização de atraso durante a síntese física de circuitos integrados cmos. 2005.

SCARTEZZINI, G. Low-power design using networks of transistors. 2014.

SCHNEIDER, F. R. Building transistor-level networks following the lower bound on the number of stacked switches. 2007.

SMITH, D. K. Network flows: Theory, algorithms, and applications. **Journal of the Operational Research Society**, Taylor & Francis, v. 45, n. 11, p. 1340–1340, 1994.

SPINDLER, P.; SCHLICHTMANN, U.; JOHANNES, F. M. Abacus: fast legalization of standard cell circuits with minimal movement. In: **Proceedings of the 2008 international symposium on Physical design**. [S.l.: s.n.], 2008. p. 47–53.

TRATT, L. **Multime @ONLINE**. 2021. Disponível em: <https://tratt.net/laurie/src/multitime>.

TUMELERO, D. Exploração de paralelismo no roteamento global de circuitos vlsi. 2015.

WANG, C.-H. et al. An effective legalization algorithm for mixed-cell-height standard cells. In: IEEE. **2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)**. [S.l.], 2017. p. 450–455.

WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. **Electronic design automation: synthesis, verification, and test**. [S.l.]: Morgan Kaufmann, 2009.

WESTE, N.; HARRIS, D. **CMOS VLSI Design: A Circuits and Systems Perspective**. 4th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0321547748.

WILKE, G. R. Analysis and optimization of mesh-based clock distribution architectures. 2008.

WU, C.-H.; MAK, W.-K.; CHU, C. Linear-time mixed-cell-height legalization for minimizing maximum displacement. In: **Proceedings of the 2022 International Symposium on Physical Design**. [S.l.: s.n.], 2022. p. 211–218.

XU, X. et al. Standard cell library design and optimization methodology for asap7 pdk. In: IEEE. **2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)**. [S.l.], 2017. p. 999–1004.

YANG, H. et al. Mixed-cell-height legalization on cpu-gpu heterogeneous systems. 2022.

ZHU, Z. et al. Mixed-cell-height legalization considering technology and region constraints. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 39, n. 12, p. 5128–5141, 2020.

ZIMPECK, A. L. **Circuit-level approaches to mitigate the process variability and soft errors in FinFET logic cells**. Tese (Doutorado) — Institut Supérieur de l'Aéronautique et de l'Espace (ISAE); Universidade . . . , 2019.

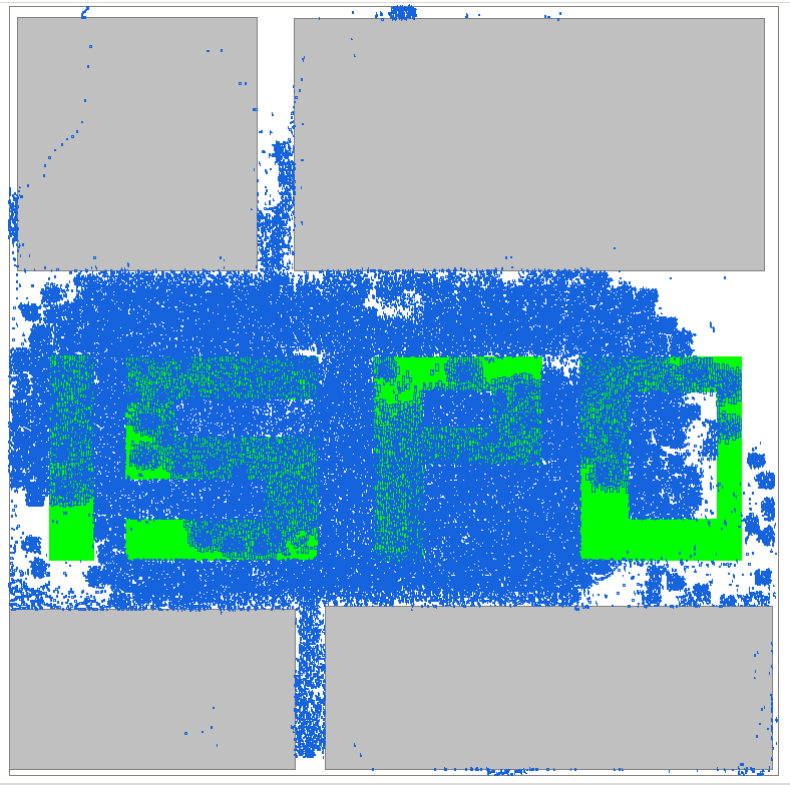
APÊNDICE A — CIRCUITOS UTILIZADOS

Neste Apêndice são ilustrados os circuitos utilizados neste trabalho a partir das suas soluções de posicionamento global.

A.1 des_perf_a

A Figura A.1 ilustra o circuito que possui dois variações, esses circuitos possuem quatro macroblocos e quatro *fence regions* cada um.

Figura A.1: des_perf_a



A.2 des_perf_b

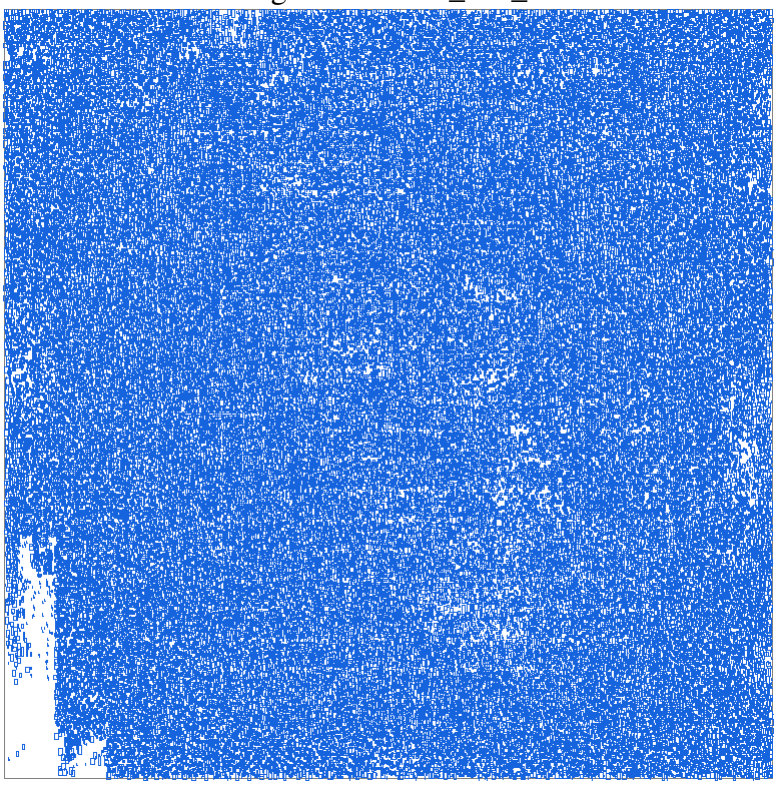
A Figura A.2 ilustra o circuito que possui dois variações, esses circuitos não possuem macroblocos, mas possuem doze *fence regions* cada um.

Figura A.2: des_perf_b

A.3 edit_dist_1_md1

A Figura A.3 ilustra o circuito que possui apenas uma variação, esse circuito não possui macrobloco e *fence region*.

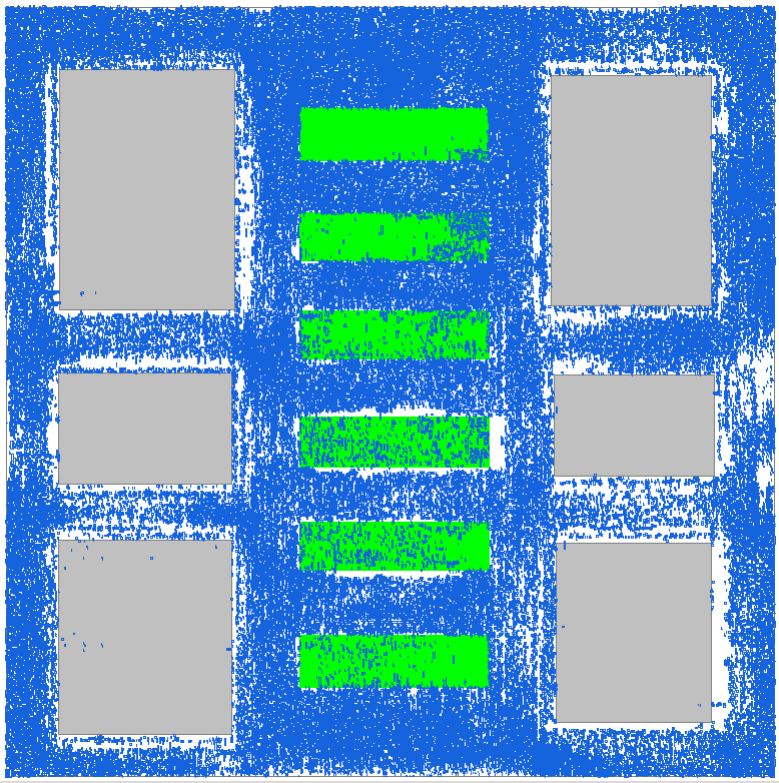
Figura A.3: edit_dist_a



A.4 edit_dist_a

A Figura A.4 ilustra o circuito que possui duas variações, esses circuito possuem seis macroblocos e apenas uma *fence region* cada um.

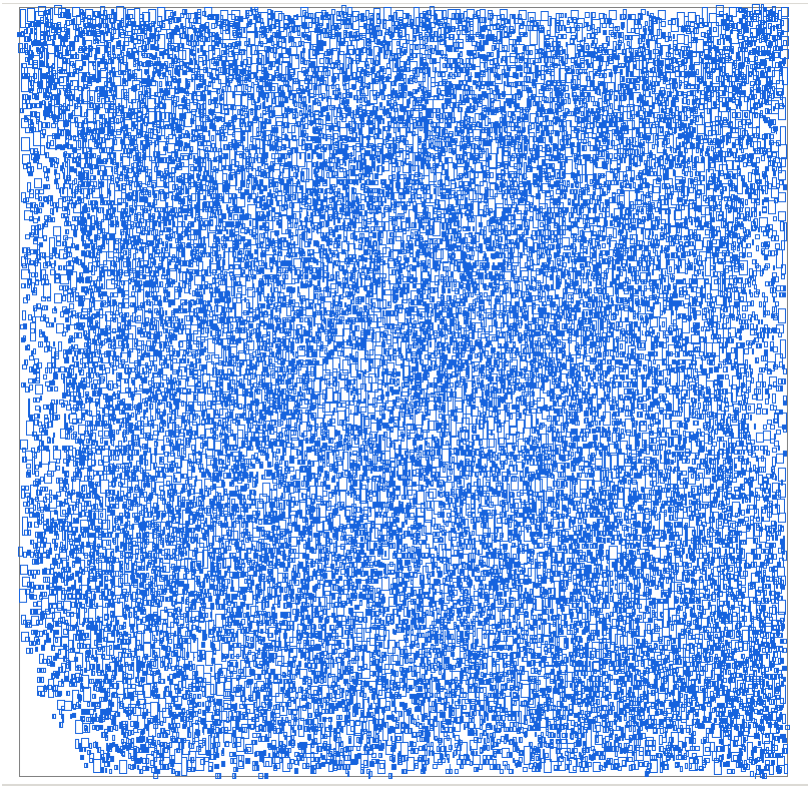
Figura A.4: edit_dist_a



A.5 fft_2

A Figura A.5 ilustra o circuito que possui apenas uma variação, esse circuito não possui macrobloco e *fence region*.

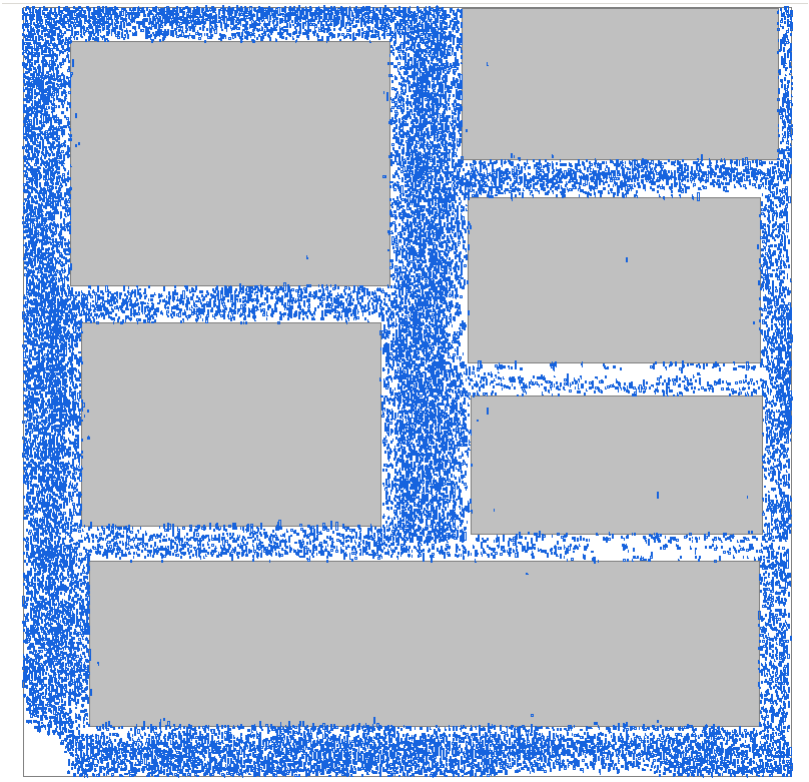
Figura A.5: fft_2



A.6 fft_a

A Figura A.6 ilustra o circuito que possui duas variaçõs, esses circuitos seis possuem macroblocos e nenhuma *fence region* cada um.

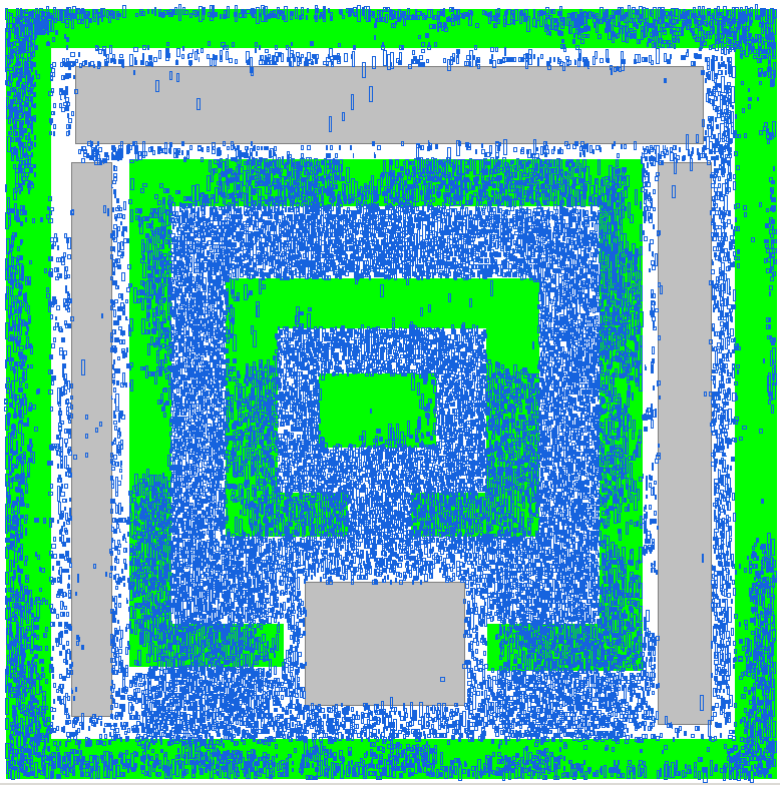
Figura A.6: fft_a



A.7 pci_bridge32_a

A Figura A.7 ilustra o circuito que possui duas variaçõs, esses circuitos possuem quatro macroblocos e três *fence regions* cada um.

Figura A.7: pci_bridge32_a



A.8 pci_bridge32_b

A Figura A.8 ilustra o circuito que possui três variaçõs, esses circuitos seis possuem macroblocos e três *fence regions* cada um.

Figura A.8: pci_bridge32_b