UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

João Carlos Prats Ramos

Impacto da Otimização Lógica Baseada em Programação Genética Cartesiana na Síntese de Circuitos Aproximados

Florianópolis
2024

João Carlos Prats Ramos

Impacto da Otimização Lógica Baseada em Programação Genética Cartesiana na Síntese de Circuitos Aproximados

Trabalho de conclusão de curso submetido ao Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia Eletrônica.
Orientadora: Prof. Cristina Meinhardt, Dra.
Coorientador: Prof. Jônata Tyska Carvalho, Dr.

Florianópolis
2024

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Ramos, João Carlos Prats
Impacto da otimização lógica baseada em programação genética cartesiana na síntese de circuitos aproximados / João Carlos Prats Ramos ; orientadora, Cristina Meinhardt, coorientador, Jônata Tyska Carvalho, 2024.
55 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2024.

Inclui referências.

1. Engenharia Eletrônica. 2. Síntese Lógica. 3. Síntese Física. 4. OpenROAD. 5. Programação Genética Cartesiana. I. Meinhardt, Cristina. II. Carvalho, Jônata Tyska. III. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. IV. Título.

João Carlos Prats Ramos

Impacto da Otimização Lógica Baseada em Programação Genética Cartesiana na Síntese de Circuitos Aproximados

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Engenharia Eletrônica e aprovado em sua forma final pelo Curso Engenharia Eletrônica.

Florianópolis, 22 de maio de 2024.

Documento assinado digitalmente
Daniela Ota Hisayasu Suzuki
Data: 06/06/2024 20:56:45-0300
CPF: ***.397.389-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Daniela Ota Hisayasu Suzuki, Dra.
Coordenadora do Curso

Banca Examinadora:

Documento assinado digitalmente
Cristina Meinhardt
Data: 03/06/2024 15:27:11-0300
CPF: ***.850.380-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Cristina Meinhardt, Dra.
Orientadora
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Fabian Leonardo Cabrera Riano
Data: 04/06/2024 10:45:12-0300
CPF: ***.071.638-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Fabian Leonardo Cabrera Riaño, Dr.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Eduardo Luiz Ortiz Batista
Data: 06/06/2024 08:07:58-0300
CPF: ***.521.889-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Eduardo Luiz Ortiz Batista, Dr.
Avaliador
Universidade Federal de Santa Catarina

Florianópolis, 2024.

Este trabalho é dedicado à minha família e colegas.

AGRADECIMENTOS

Agradeço à minha mãe por sempre guiar a minha vida da melhor forma possível e por me incentivar e apoiar desde pequeno a estudar e dar o meu melhor.

À minha companheira Ana, por me apoiar tanto durante este período da graduação e ser minha parceira para a vida.

Sou muito grato à minha orientadora Cristina e ao coorientador Jônata por todos os ensinamentos durante este trabalho e por terem almejado e executado junto comigo este TCC.

E, por último, aos meus amigos, tanto os que já conhecia antes quanto aos que conheci durante a graduação; com eles ao lado, muitas coisas se tornaram mais fáceis e felizes.

RESUMO

Circuitos integrados permitiram a evolução tecnológica chegando aos dias atuais, onde a sociedade depende de dispositivos, aparelhos e veículos compostos por uma série de circuitos integrados. A variedade de funcionalidades a serem projetadas em circuitos integrados é enorme, tornando o projeto de circuitos integrados desafiador. Um dos principais modos de projeto de circuitos integrados é utilizando um fluxo de projeto baseado em biblioteca de células. Dada as complexidades de projeto, este fluxo é dividido em vários passos dentro de duas etapas principais: a síntese lógica e a síntese física. Esta definição tradicional permite que ferramentas específicas para cada etapa sejam desenvolvidas, explorando diferentes algoritmos e estruturas de dados. Dentro da síntese lógica, a otimização lógica é uma etapa inicial e crucial. Os efeitos dessa etapa impactam diretamente as métricas finais de área, potência e atraso observados após todas as etapas seguintes da síntese. Os avanços significativos em Aprendizado de Máquina (ML) motivam o seu uso na síntese lógica de circuitos combinacionais. Técnicas de ML são particularmente adequadas para síntese e otimização lógica aproximada em cenários de projeto de circuitos integrados para aplicações tolerância a erros, principalmente visando eficiência energética. Recentemente, a utilização de Programação Genética Cartesiana (CGP) foi explorada em uma ferramenta para síntese lógica tradicional e voltada para otimização energética, síntese de funções incompletas e aprendizado de lógica. Esta ferramenta foi avaliada e comparada com outros trabalhos relacionados quanto as métricas tradicionais de síntese lógica de profundidade lógica e número de nodos, e também de acurácia nos casos de aproximação e aprendizado. Entretanto, o impacto destas ferramentas nas próximas etapas de síntese não foi avaliado. Neste sentido, este trabalho investiga o impacto nas métricas de área, *delay* e potência da exploração do uso de técnicas de aprendizado de máquina no aprendizado lógico e na otimização lógica para aprimorar a eficiência energética na síntese de circuitos aproximados.Esta iniciativa é realizado com ferramentas de código aberto, promovendo a acessibilidade e a colaboração na área de *Design* Eletrônico Automatizado (EDA). Neste trabalho são consideradas três abordagens de otimização lógica aproximada estado-da-arte com base em *mixed-ML*, Árvores de Decisão e Programação Genética Cartesiana. Para mostrar a eficácia de cada uma dessas abordagens, este trabalho apresenta uma análise comparativa dos resultados de síntese física para um conjunto de circuitos aproximados ao adotar um fluxo de síntese aberto e uma biblioteca de células de tecnologia de 45 nm. Das técnicas avaliadas, a otimização lógica baseada em CGP mostra uma redução de mais de 50% em média de área, potência e atraso em comparação com a abordagem de *mixed-ML*. No entanto, essa melhoria foi acompanhada por uma perda média de acurácia de cerca de 5%. Esses resultados destacam o potencial substancial da abordagem CGP na otimização de circuitos aproximados.

Palavras-chave: Síntese Lógica. Síntese Física. OpenROAD. Programação Genética Cartesiana.

ABSTRACT

Integrated circuits have enabled technological evolution, leading to the current era where society depends on devices, appliances, and vehicles composed of a myriad of integrated circuits. The variety of functionalities to be designed into integrated circuits is enormous, making integrated circuit design challenging. One of the primary methods of integrated circuit design is using a library-based design flow. Given the complexities of design, this flow is divided into several steps within two main stages: logic synthesis and physical synthesis. This traditional definition allows specific tools for each stage to be developed, exploring different algorithms and data structures. Within logic synthesis, logic optimization is an initial and crucial step. The effects of this step directly impact the final metrics of area, power, and delay observed after all subsequent synthesis steps. Significant advances in Machine Learning (ML) motivate its use in logic synthesis of combinational circuits. ML techniques are particularly suitable for approximate logic synthesis and optimization in integrated circuit design scenarios for error-tolerant applications, primarily targeting energy efficiency. Recently, the use of Cartesian Genetic Programming (CGP) has been explored in a tool for traditional logic synthesis and energy optimization, incomplete function synthesis, and logic learning. This tool was evaluated and compared with other related work regarding traditional logic synthesis metrics such as logic depth and node count, as well as accuracy in approximation and learning cases. However, the impact of these tools on the next synthesis steps was not evaluated. In this regard, this work investigates the impact on area, delay, and power metrics of exploring the use of machine learning techniques in logic learning and optimization to enhance energy efficiency in approximate circuit synthesis.This initiative is carried out with open-source tools, promoting accessibility and collaboration in the field of Electronic Design Automation (EDA). In this work, three state-of-the-art approximate logic optimization approaches based on mixed-ML, Decision Trees (DT), and Cartesian Genetic Programming are considered. To demonstrate the effectiveness of each of these approaches, this work presents a comparative analysis of physical synthesis results for a set of approximate circuits when adopting an open synthesis flow and a 45nm technology cell library. Of the evaluated techniques, CGP-based logic optimization shows an average reduction of over 50% in area, power, and delay compared to the mixed-ML approach. However, this improvement was accompanied by an average accuracy loss of about 5%. These results highlight the substantial potential of the CGP approach in optimizing approximate circuits.

Keywords: Logic Synthesis. Physical Synthesis. OpenROAD . Cartesian Genetic Programming.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

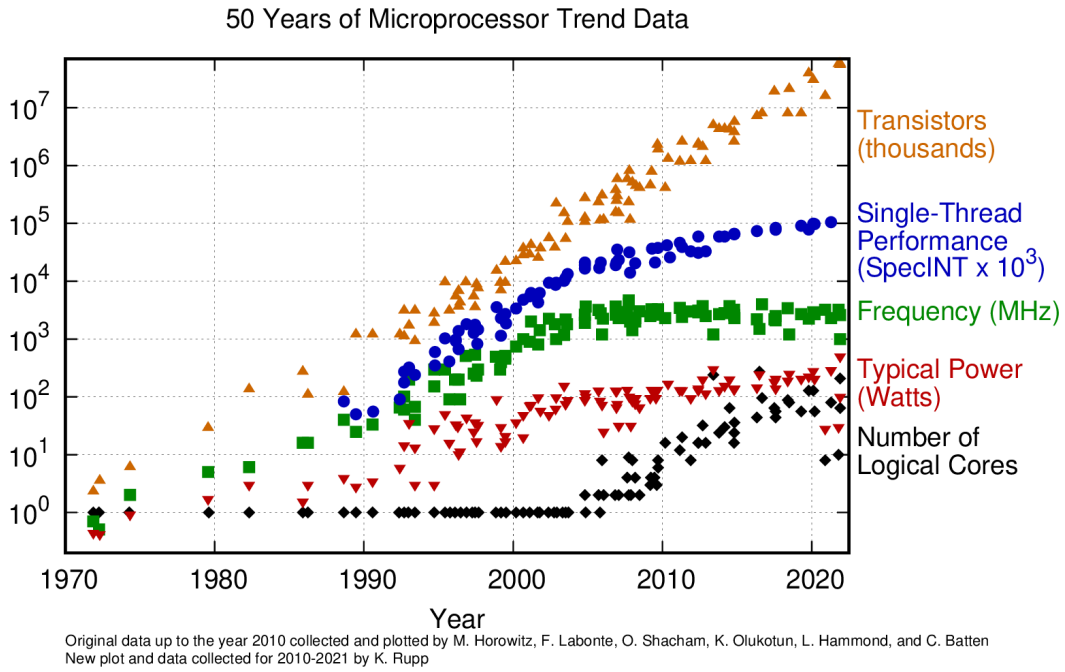
ABC	*A System for Sequential Synthesis and Verification*
AIG	*And-Inverter-Graph*
ASIC	*Application Specific Integrated Circuit*
CAD	*Computer-Aided Design*
CGP	*Cartesian Genetic Programming* - Programação Genética Cartesiana
CI	Circuito Integrado
CIFAR-10	*Canadian Institute for Advanced Research*
DARPA IDEA	*Defense Advanced Research Projects Agency Intelligent Design of Electronic Assets*
DT	*Decision Trees* - Árvores de Decisão
EDA	*Electronic Design Automation*
EQN	*Equation*
ERI	*Electronics Resurgence Initiative*
ES	Estratégias de Evolução
FPGA	*Field-Programmable Gate Array*
HDL	*Hardware Description Language* - Linguagem de Descrição de Hardware
IEEE	*Institute of Electrical and Electronics Engineers*
IWLS	*International Workshop on Logic and Synthesis*
LUT	*Look-Up Tables networks*
MIS	*Multilevel Logic Interactive Synthesis System*
ML	*Machine Learning* - Aprendizado de Máquina
MNIST	*Modified National Institute of Standards and Technology*
MVSIS	*Multi-Valued Logic Synthesis*
OpenROAD	*Foundations and Realization of Open, Accessible Design*
PLA	*Programmable Logic Array*
PoS	produto-de-somas
RTL	*Register Transfer Level*
SBCCI	*Symposium on Integrated Circuits and Systems Design*
SIS	*Synthesis Interactive System*
SoP	somas-de-produtos
UFSC	Universidade Federal de Santa Catarina
VHDL	*VHSIC Hardware Description Language*
VLSI	*Very Large Scale Integration*
XOR	OU-Exclusivo

SUMÁRIO

1 INTRODUÇÃO

O mundo contemporâneo tem a tecnologia como parte fundamental de seu funcionamento. O número de transistores por componente cresceu rapidamente ao longo das décadas, aumentando o desempenho das tecnologias que os utilizam. Essa expansão permite a maior complexidade das funções realizadas em *hardware* (MOORE *et al.*, 1965). A Figura 1 apresenta o crescimento exponencial do número de transistores de 1970 a 2020 (RUPP, 2022). O gráfico também detalha a evolução da performance de *single-threads*, estratégia adotada para aumentar o número de instruções realizadas pelo processador, mesmo que a frequência permaneça estabilizada nas últimas décadas. Por um lado, isso é positivo, pois possibilita a construção de sistemas avançados e de alto desempenho e a criação de tecnologias modernas. Por outro lado, torna o processo de produção mais desafiador e demorado, exigindo aprimoramentos na área do desenvolvimento de ferramentas de apoio ao projeto de circuitos integrados, conhecidas como *Electronic Design Automation* (EDA) para acompanhar esses avanços (BEEREL; PEDRAM, 2018).

Figura 1 – Evolução das características dos circuitos ao longo dos anos



Fonte: (RUPP, 2022).

Desta forma, o projeto de circuitos usualmente envolve gerar uma descrição fabricável do circuito partindo de uma descrição de alto nível, utilizando linguagens de descrição de *hardware*. Um dos principais modos de projetar e fabricar um circuito integrado é seguindo fluxo *Standard Cell*. Esse processo é realizado com ferramentas que realizam duas grandes etapas: (1) síntese lógica, que traduzem a descrição de alto nível para uma

versão intermediária mais próxima dos dispositivos utilizados na fabricação (e.g., *Standard Cells*), chamada *netlist*; e (2) síntese física, que realiza posicionamento das células, geração de caminhos de roteamento e demais etapas necessárias para o circuito ser fabricável (CHATTERJEE *et al.*, 2006; REIS; MATOS, 2018).

A otimização lógica, um dos passos da síntese lógica, é uma das etapas com potencial de proporcionar benefícios na produção e qualidade dos circuitos, gerando interesse da indústria pela sua melhoria. Esses procedimentos geram ganhos no desempenho com a redução do número de termos da função e profundidade lógica, impactando área, atraso de propagação, e consumo energético. Na otimização lógica, observamos algumas iniciativas para explorar *Machine Learning* - Aprendizado de Máquina (ML). Diferentemente da otimização lógica tradicional, onde a especificação da função do circuito é fornecida por equações ou Tabelas-Verdade que definem precisamente todo o comportamento do circuito, na síntese lógica aproximada temos a possibilidade de gerar circuitos para funções lógicas cujas saídas esperadas não são completamente corretas ou conhecidas.

A síntese de circuitos aproximados é apropriada para cenários resilientes a erros, onde a computação aproximada é uma alternativa para tornar o fluxo de projeto mais rápido e obter soluções com eficiência energética, entretanto, podendo permitir imprecisões observadas na saída esperada. Essas aplicações possuem requisitos de acurácia menos restritos para as funções implementadas. Nesses projetos, pequenas distorções na saída podem ser aceitáveis, possibilitando a otimização dos projetos em outras restrições como área, consumo de energia e frequência de operação. Exemplos bem conhecidos desta estratégia incluem técnicas de compressão de vídeo, que aceitam uma pequena taxa de erro na reprodução de *pixels* sem ser perceptível ao sistema visual humano. Isso permite trocar a qualidade do vídeo por desempenho (JIANG *et al.*, 2020).

Algumas soluções abordam a geração de um circuito lógico razoavelmente pequeno que tenta generalizar um espaço de entrada inteiro com base em amostras fornecidas seletivamente, visando a melhor acurácia. Neste caso, quando as saídas esperadas da função a ser otimizada não são conhecidas, parte-se para uma abordagem de Aprendizado Lógico. No problema de aprendizado lógico, temos um conjunto de *don't knows* definidos como as saídas cujo valor é desconhecido. É diferente dos *don't cares* tradicionais, ou seja, os valores de saída que podem assumir qualquer valor no problema. As soluções de aprendizado lógico têm que inferir os valores prováveis para o conjunto de *don't knows*, com base em poucas entradas fornecidas pela Tabela-Verdade completa de um circuito específico.

O desenvolvimento de soluções baseadas em *machine learning* para o aprendizado lógico ou síntese de funções incompletas foi proposto no *International Workshop on Logic and Synthesis* (IWLS) Contest em 2020 (IWLS'20 ORGANIZING COMMITTEE, 2020). A principal finalidade do concurso foi obter a melhor acurácia com um limite de 5000 nodos lógicos. O número de entradas das funções lógicas disponibilizadas neste *benchmark* varia

de 10 a 768. Um conjunto de soluções foram propostas nesta competição, sendo reportada uma visão geral em (RAI; NETO *et al.*, 2021). Destas soluções, destacaram-se o vencedor que produziu a solução com melhor acurácia geral para os *benchmarks* considerados (MIYASAKA *et al.*, 2021), um time que gerou os menores circuitos em número de nodos (TENACE; CALIMERA, 2018b), e uma solução gerada por participantes da Universidade Federal de Santa Catarina (UFSC) que explorou o uso de *Cartesian Genetic Programming* - Programação Genética Cartesiana (CGP) (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022), pela diversidade na técnica de *machine learning* explorada. Mais detalhes destas técnicas serão apresentados nos Capítulos de 2 e 3. Estas três técnicas se destacaram na competição do IWLS quanto as métricas tradicionais de síntese lógica. Entretanto, até onde sabemos neste momento, estes trabalhos não foram avaliados sobre como estas técnicas influenciam os resultados de área, potência e atraso (frequência) finais da síntese dos circuitos integrados, após interagirem com as demais etapas do fluxo de síntese.

Buscando compreender tais impactos, este trabalho compara essas três abordagens de síntese lógica para o mesmo conjunto de *benchmarks* seguindo um mesmo fluxo de síntese física. Desta forma, até onde foi levantando no referencial teórico, este é o primeiro trabalho a apresentar resultados de síntese física de circuitos gerados para este conjunto de *benchmarks* de funções aproximadas, discutindo a relação entre o número de nós e a profundidade lógica observada na fase de síntese lógica e os impactos nas métricas de atraso, potência e área obtidas na síntese física.

1.1 OBJETIVOS

O objetivo geral deste trabalho é analisar três abordagens diferentes de otimização lógica quanto aos impactos na síntese de circuitos integrados seguindo um fluxo *Standard Cell*. As abordagens de otimização lógica exploradas são: uma baseada em um conjunto de técnicas de *machine learning*, chamada de *Mixed*-ML (MIYASAKA *et al.*, 2021), uma em *Decision Trees* - Árvores de Decisão (DT). (TENACE; CALIMERA, 2018b), e uma baseada em CGP (BERNDT; CAMPOS *et al.*, 2021). A avaliação dos impactos na síntese física são observados utilizando um fluxo comum de síntese, adotando o fluxo de síntese aberto *Foundations and Realization of Open, Accessible Design* (OpenROAD). Uma das principais contribuições deste trabalho é avaliar os ganhos de um fluxo de otimização lógica baseado em programação genética cartesiana para síntese de circuitos aproximados. Além disso, define-se os seguintes pontos como objetivos específicos:

- Configurar e adaptar o ambiente OpenROAD para a integração e execução do fluxo proposto;
- Avaliar benefícios e desvantagens dos fluxos de otimização lógica em termos de área, *delay* e potência.

Além disso, este trabalho tem como objetivo explorar soluções que utilizem ferramentas *open-source* para a síntese dos circuitos integrados e no desenvolvimento e avaliação das atividades realizadas.

1.2 ORGANIZAÇÃO DO TEXTO

O próximo Capítulo apresenta os principais conceitos para o entendimento do trabalho desenvolvido.O Capítulo 3 apresenta as principais motivações da realização deste trabalho . O Capítulo 4 analisa trabalhos que tratem de assuntos relacionados ao projeto proposto. Já o Capítulo 5 discute a proposta e o desenvolvimento da integração e avaliação de circuitos aproximados por meio da aplicação da técnica de Programação Genética Cartesiana, apresentando os resultados comparados com outras técnicas avaliadas no Capítulo 6. As conclusões gerais sobre o que se observou durante o desenvolvimento e desempenho de cada fluxo é visto no Capítulo 7.

2 CONCEITOS FUNDAMENTAIS

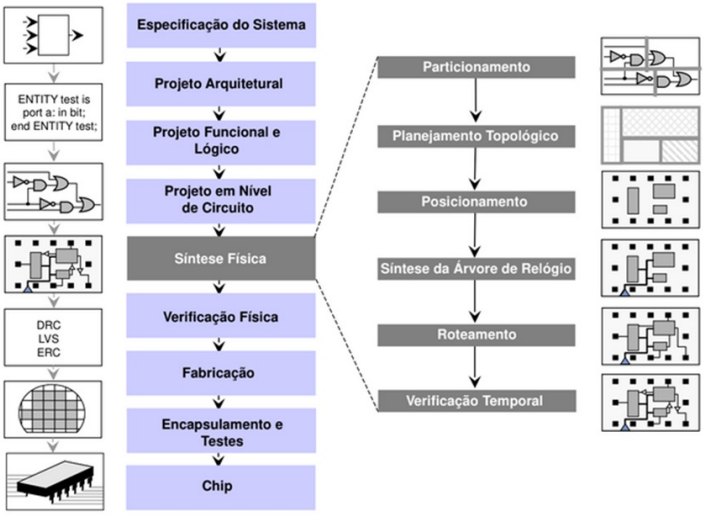
O estudo em questão adota o fluxo baseado em CGP proposto por (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022) que é apresentado no Capitulo 3. Para compreender e desenvolver o fluxo proposto neste estudo, é essencial descrever as estruturas e ferramentas utilizadas, a fim de compreender as razões por trás de suas escolhas. Este capítulo aborda os principais conceitos de Síntese Lógica e explora algumas ferramentas consideradas durante a execução deste trabalho.

2.1 FLUXO DE SÍNTESE BASEADO EM BIBLIOTECA DE CÉLULAS

O projeto destes circuitos necessita de um fluxo de ferramentas de EDA, geralmente dividido em etapas para atingir a fabricação de circuitos integrados. O fluxo de projeto de circuitos integrados é um conjunto de procedimentos que permite aos projetistas percorrerem um caminho desde a especificação do sistema até sua implementação no silício (WESTE; HARRIS, 2010).

A Figura 2 apresenta um fluxo de projeto para a metodologia baseada em biblioteca de células divido em nove estágios: Especificação do Sistema, Projeto Arquitetural, Projeto Funcional e Lógico, Projeto em Nível de Circuito, Síntese Física, Verificação Física, Fabricação, Encapsulamento e Teste, e finalmente, a entrega do chip projetado. A primeira etapa é a modelagem em alto nível, responsável pela descrição comportamental do sistema utilizando uma linguagem *Hardware Description Language* - Linguagem de Descrição de Hardware (HDL) como: *SystemC*, *VHSIC Hardware Description Language* (VHDL), Verilog ou até mesmo C/C++. O objetivo é descrever o comportamento do sistema de forma que ele possa ser simulado e validado.

Figura 2 – Fluxo de Síntese



Fonte: Adaptado de:(KAHNG *et al.*, 2011)

Entretanto, essa descrição é de alto nível dificultando a síntese de um hardware diretamente a partir dela. Portanto, a descrição comportamental é traduzida para uma descrição *Register Transfer Level* (RTL). Nesta descrição, o comportamento do circuito é representado através do fluxo de sinais entre registradores e as operações lógicas realizadas nestes sinais.

No projeto funcional e lógico, a descrição RTL é inicialmente transformada em um circuito com portas lógicas genéricas básicas. Assim é possível realizar otimizações independentes de tecnologia através da reestruturação das portas lógicas no circuito. Após isso, o circuito passa pelo mapeamento tecnológico. Esta etapa é responsável por substituir as portas lógicas genéricas por portas lógicas presentes na biblioteca de células adotada no projeto. Depois de mapeado, o circuito passa pela otimização dependente da tecnologia. Ela reestrutura o circuito baseada em informações elétricas das portas lógicas, como atrasos, área e consumo de potência. A síntese física é responsável pela última fase antes de enviar o projeto para manufatura. Ela é responsável por converter um circuito no domínio estrutural para o domínio físico.

Dentro do fluxo de síntese física, a primeira etapa é chamada de planejamento topológico. Com o crescimento da complexidade dos circuitos integrados, o uso de abordagens hierárquicas passou a ser amplamente adotado (WANG; CHANG; CHENG, 2009). Estas abordagens dividem os componentes do circuito em grandes módulos de acordo com a tarefa que exercem. O objetivo do planejamento topológico é realizar uma divisão inicial no chip, reservando uma região para cada um desses blocos. As duas etapas seguintes da síntese física, posicionamento e roteamento, acontecem individualmente em cada um desses módulos. O posicionamento é a etapa responsável por encontrar uma posição válida para cada uma das células na área disponível para o projeto em um chip, obedecendo ainda outras restrições de projeto. Esta etapa tem uma grande importância para o projeto, pois afeta diretamente o desempenho, consumo de energia e atrasos do circuito. Posicionadores geralmente recebem como entrada uma relação de todas as células e suas conexões, conhecida como *netlist*. Além disso, ferramentas de posicionamento também recebem uma descrição da biblioteca de células, que representa o conjunto de células disponíveis para a composição dos circuitos integrados. Em projetos *Very Large Scale Integration* (VLSI) é cada vez mais comum incluir grandes módulos proprietários no desenvolvimento do projeto, tais como memórias ou dispositivos analógicos. Neste caso, os algoritmos de posicionamento devem ser capazes de lidar com estes módulos, conhecidos como obstáculos ou macro blocos, aumentando a complexidade da tarefa de posicionamento. Outra etapa importante é o roteamento, responsável por determinar as rotas exatas pelas quais os fios que realizam a troca de sinais entre as células (interconexões) serão realizadas.

Circuitos modernos podem conter milhões de conexões e as ferramentas de roteamento devem ser capazes de planejar todas elas. Para realizar esta tarefa, o roteamento é dividido em dois passos: o roteamento global e o roteamento detalhado. O roteamento

global divide a área do chip em canais adjacentes e cria as rotas entre estes canais. Já o posicionamento detalhado é responsável por determinar o caminho exato que os fios percorrerão. Em circuitos síncronos, os dados são processados obedecendo um sinal de *clock*. Para garantir o funcionamento da lógica do circuito é necessário garantir que este sinal chegue em cada célula sequencial no tempo necessário. Pensando nisso, a etapa de geração da árvore de*clock* é responsável por distribuir este sinal garantindo que o atraso relativo entre cada célula seja o mínimo possível.

Em um fluxo sequencial de síntese, depois que o circuito é posicionado, roteado e a árvore de *clock* é gerada, ele passa por ferramentas de teste e verificação. Uma das ferramentas de verificação é responsável pela extração de efeitos parasitas associados às conexões. Esses efeitos são devido às indutâncias, capacitâncias e resistências do circuito. Uma vez que essas informações são obtidas é possível simular a execução do hardware. Neste ponto é possível verificar com maior acurácia se o circuito corresponde às especificações de projeto, verificando as características de atrasos, potência e área. Esta etapa é conhecida por ser um gargalo no projeto pois muitas vezes é necessário realizar novamente o posicionamento e o roteamento repetidamente até que se atinja às especificações de projeto desejadas. Outras ferramentas de verificação e validação são responsáveis por verificar ruídos, queda na tensão de alimentação e limites de eletromigração. Se o projeto atender às especificações, está pronto para fabricação.

Este trabalho avalia o impacto de um conjunto de ferramentas de otimização lógica na síntese física. A próxima seção detalha mais sobre a síntese lógica e síntese lógica aproximada. Na Seção 3.3, apresenta-se o fluxo de síntese adotado neste trabalho.

2.2 SÍNTESE LÓGICA

O desenvolvimento de circuitos integrados envolve várias fases de abstração e otimização, começando com o conceito inicial do projetista até a conclusão do circuito. A etapa de Síntese Lógica é essencial nesse processo, pois sua função é transformar uma definição de alto nível da tecnologia em uma versão mais para uma de mais baixo nível, fornecendo os elementos necessários para os próximos passos.

A representação de *hardware* em alto nível é realizada através de uma *HDL*. Inicialmente, por questões de praticidade, essa descrição é feita de forma comportamental, indicando o funcionamento da lógica por meio de equações Booleanas ou algoritmos em linguagem de alto nível, sem a necessidade de muitos detalhes. Embora esse formato seja mais acessível para os humanos, pois se assemelha mais à nossa linguagem, ele não é suficiente para dar continuidade ao fluxo de síntese, pois carece de informações importantes, como a lista de conexões (*netlist*) necessária para as etapas subsequentes, como o posicionamento e roteamento do circuito.

Para converter a descrição comportamental em estrutural, é necessário realizar uma Síntese de Alto Nível, que transforma a descrição comportamental da tecnologia em

uma descrição de *hardware*, como, por exemplo, RTL. Posteriormente, a Síntese Lógica é aplicada, dividindo-se em três fases: a transformação do RTL em estruturas intermediárias, o mapeamento para a tecnologia do circuito final(*Field-Programmable Gate Array* (FPGA) ou *Application Specific Integrated Circuit* (ASIC)) e otimizações específicas para a tecnologia mapeada. Além disso, técnicas de otimização lógica também são aplicadas nas estruturas intermediárias, juntamente com aprimoramentos posteriores ao mapeamento tecnológico.

Na primeira fase, são aplicadas transformações que independem da tecnologia do circuito final, baseando-se apenas no comportamento funcional desejado. Isso resulta na geração de uma estrutura intermediária, que pode assumir diversas formas, como uma expressão Booleana, uma rede Booleana ou um grafo *And-Inverter-Graph* (AIG).

Uma maneira viável de realizar a síntese independente de tecnologia é por meio de métodos de síntese de funções Booleanas. Esses métodos têm como objetivo representar um comportamento lógico, frequentemente expresso por meio de uma tabela verdade, em uma expressão Booleana composta por literais e operadores Booleanos.Existem diversos métodos de síntese de funções Booleanas e cada método pode apresentar diferentes abordagens. Uma função Booleana pode ser sintetizada em uma estrutura de dois níveis, como somas-de-produtos (SoP) ou produto-de-somas (PoS), ou em uma expressão fatorada de estrutura multi-nível (MICHELI, 1994). Existem abordagens onde a representação de funções pode ser restringido ao uso de operações de *AND* e *OR*, enquanto outras abordagens utilizam também a operação de OU-Exclusivo (XOR).

Em seguida, o mapeamento tecnológico mapeia partes da estrutura intermediária em células com informações relativas a tecnologia do circuito final. Por fim, a otimização dependente de tecnologia realiza otimizações nas células mapeadas, como redimensionamento de células e duplicação de lógica.

2.2.1 Síntese Lógica Aproximada

A estratégia de Computação Aproximada pode ser incorporada em diversas fases do projeto de circuitos, incluindo a etapa de Síntese Lógica. Nessa fase, a automação dos processos é mais viável, tornando possível a aplicação de aproximações sem exigir um alto nível de especialização por parte do projetista. Isso permite a implementação da estratégia em qualquer tipo de circuito, mesmo sem um conhecimento prévio abrangente do mesmo (SCARABOTTOLO *et al.*, 2020).

Essas técnicas têm como objetivo modificar a função que descreve o circuito, reduzindo seu tamanho final enquanto se alcança um nível aceitável de erro em comparação com a função original. Uma abordagem para alcançar essa redução é alterar intencionalmente algumas saídas da Tabela Verdade. Outra estratégia envolve a manipulação da função Booleana que representa o circuito. Independentemente do método utilizado, o objetivo é obter uma função menor do que a original, sem comprometer significativamente

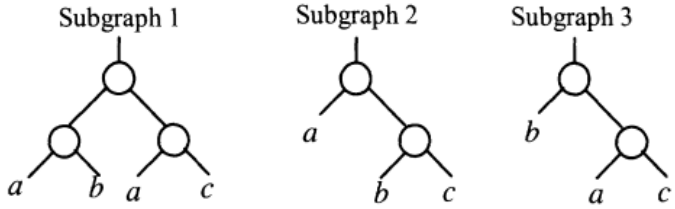
a acurácia da saída

2.2.2 Estrutura de dados AIG

A estrutura de dados utilizada em qualquer algoritmo ou programa computacional é de extrema importância quanto ao desempenho e qualidade do resultado final gerado. Uma conhecida estrutura na área de tecnologia é a de grafos. Um grafo é uma rede de pontos, chamados nodos, conectados por linhas ou, quando o sentido da ação é importante, setas (BERGE, 2001). Eles podem ser utilizados para representar uma grande variedade de situações cotidianas e são aplicados na representação de mapas, estruturas sociais, diagramas, etc.

A estrutura de dados do estado-da-arte para otimizações independentes de tecnologia durante a síntese lógica é o AIG (RIENER *et al.*, 2019). AIG é um grafo dirigido e acíclico cujos nodos podem ter zero ou duas entradas, sendo que no primeiro caso se tem uma entrada primária e no segundo uma porta *AND*. Ainda, cada aresta pode ser complementada ou não, representando as portas inversoras. Para a identificação das saídas, os nodos podem ser marcados como saída primária (MISHCHENKO, A.; CHATTERJEE, S.; BRAYTON, R., 2006). A Figura 3 ilustra a otimização de um subgrupo do AIG, mostrando inicialmente um grafo para uma função descrita com três operações, sendo uma entre ae b, outra entre a e c e a terceira entre os produtos iniciais. A mesma função pode ser representa na forma dos outros dois subgrafos, com menos operações, entretanto variando a ordem das variáveis. Isso representa uma redução no número de nodos necessários (operações) e em entradas variáveis para representar uma mesma operação.

Figura 3 – Otimização de um subgrupo do AIG



Fonte: (MISHCHENKO, Alan; CHATTERJEE, Satrajit; BRAYTON, Robert, 2006)

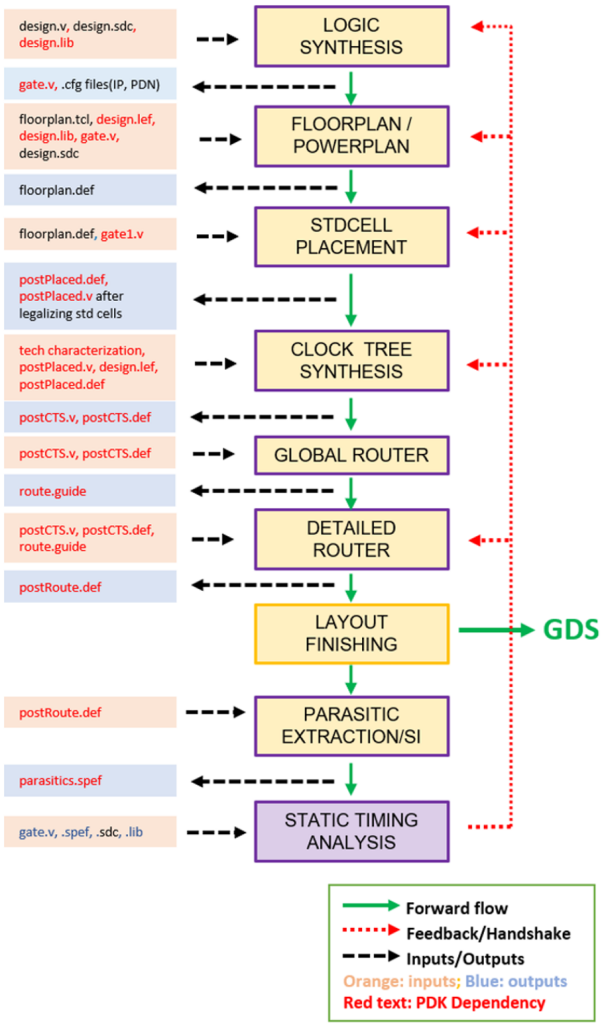
2.3 FLUXO DE SÍNTESE OPENROAD

O desenvolvimento de circuitos integrados Circuito Integrado (CI)s envolve várias etapas, cada uma requerendo algoritmos robustos e complexos para otimizar o design. A complexidade dessa atividade gera um valor significativo e desperta grande interesse em ferramentas capazes de executar tais etapas com eficiência. No entanto, muitas dessas ferramentas são de natureza privada, o que representa um obstáculo para o avanço do desenvolvimento de CIs em áreas de pesquisa com recursos financeiros limitados. Ademais,

os softwares desenvolvidos por empresas privadas restringem o acesso aos algoritmos apenas aos seus funcionários, o que impede o compartilhamento do conhecimento agregado ao projeto. Este cenário destaca a importância de estratégias colaborativas e acesso equitativo a ferramentas e recursos para promover avanços significativos no campo dos circuitos integrados.

Uma dessas estratégias colaborativas foi iniciada em julho de 2018 durante o evento *Electronics Resurgence Initiative* (ERI) *Summit*, realizado em São Francisco, Califórnia. Este projeto foi concebido dentro do programa *Defense Advanced Research Projects Agency Intelligent Design of Electronic Assets* (DARPA IDEA) (LIM, s.d.). Chamado de OpenROAD, foi desenvolvido pelo Departamento de Defesa dos Estados Unidos com o objetivo de criar um compilador de hardware de propósito geral totalmente automatizado (OPENROAD, 2022b). Essa iniciativa visa capacitar até mesmo pesquisadores com conhecimento limitado em EDA a desenvolverem tecnologias avançadas

Figura 4 – Funcionalidades do fluxo OpenROAD



Fonte: (AJAYI; BLAAUW, 2019)

Na representação do fluxo do OpenROAD, apresentada na Figura 4, uma sequência

de etapas é desenvolvida. Cada uma dessas etapas envolve o uso de diferentes ferramentas, que se destina a ser totalmente de código aberto, todas as tecnologias intermediárias empregadas ao longo do processo também seguem essa premissa. A Tabela 2 fornece uma visão detalhada de algumas dessas ferramentas, juntamente com as respectivas etapas em que são aplicadas.

O fluxo apresentado na Tabela tem como ponto de partida descrições de circuitos na linguagem Verilog. Inicialmente, realiza-se a etapa de Síntese Lógica com o auxílio do Yosys e *A System for Sequential Synthesis and Verification* (ABC). Finalizada a Síntese Lógica passa-se por etapas como *Floorplan* que inicializa o layout do chip. O ioPlacer é responsável pelo posicionamento dos pinos, enquanto o ICeWall lida com as conexões de nível de chip. TritonMacroPlacer e RePlAce trabalham no posicionamento de macro e global, respectivamente. OpenSTA entra para análise de *timing*, enquanto FastRoute e *Antenna Checker* são utilizados para o roteamento global. TritonRoute é responsável pelo roteamento detalhado, e o preenchimento de metal é realizado pelo *Metal Fill*. Outras ferramentas e processos intermediários integram o fluxo OpenROAD completo e podem ser verificados na documentação do projeto.

Uma visão abrangente das ferramentas utilizadas pode ser observada na Tabela 2. Yosys e ABC são fundamentais para a síntese lógica. Estas ferramentas serão melhor detalhadas a seguir. A ferramenta *Floorplan* inicializa o planejamento topológico do *layout* do chip. O ioPlacer é responsável pelo posicionamento dos pinos, enquanto o ICeWall lida com as conexões de nível de chip. TritonMacroPlacer e RePlAce trabalham no posicionamento de macro e global, respectivamente. OpenSTA entra em cena para análise de *timing*, enquanto FastRoute e *Antenna Checker* são utilizados para o roteamento global. TritonRoute é responsável pelo roteamento detalhado, e o preenchimento de metal é realizado pelo *Metal Fill*. Essas ferramentas, combinadas, oferecem uma plataforma robusta para o desenvolvimento eficiente de circuitos integrados.

Tabela 2 – Algumas Ferramentas Usadas no OpenROAD

Ferramenta	**Função**
Yosys e ABC	Síntese Lógica
Floorplan	Inicializa o Floorplan
ioPlacer	Posicionamento de Pinos
ICeWall	Conexões Chip-Level
TritonMacroPlacer	Posicionamento Macro
RePlAce	Posicionamento Global
OpenSTA	Análise de *Timing*
FastRoute e Antenna Cheker	Roteamento Global
TritonRoute	Roteamento Detalhado
Metal Fill	Preenchimento de Metal

Fonte: (`https://theopenroadproject.org/`, 2024)

As ferramentas integradas neste trabalho são amplamente reconhecidas na área de

síntese lógica e física, com diversos estudos publicados que demonstram seu desempenho. Todas essas ferramentas são de código aberto, enquadrando-se nos requisitos exigidos para o que está sendo aqui proposto. Dentre estas ferramentas, destacam-se a ferramenta ABC e Yosys, detalhadas a seguir.

2.3.1 ABC e Yosys

Desde 1987, a Universidade da Califórnia tem sido ativa no desenvolvimento de ferramentas de síntese lógica, sendo responsável pela criação de programas como: *Multilevel Logic Interactive Synthesis System* (MIS) (BRAYTON, Robert K *et al.*, 1987), *Synthesis Interactive System* (SIS) (SENTOVICH *et al.*, 1992) e *Multi-Valued Logic Synthesis* (MVSIS) (CHAI *et al.*, 2003), até chegar no ABC em 2005. O ABC é um sistema *Computer-Aided Design* (CAD) que aplica transformações lógicas escaláveis baseadas em AIGs utilizando-se de algoritmos de otimização para este formato, e demonstrou desempenho superior ou igual ao de seus predecessores (BRAYTON; MISHCHENKO, 2010), exigindo menos memória e tempo de execução. Essas vantagens o permitem trabalhar com circuitos maiores do que os anteriores conseguiam.

Devido à sua ampla gama de funcionalidades na síntese lógica e à sua comprovada capacidade de lidar com circuitos de entrada maiores ao longo do tempo, o ABC tem sido integrado em outras ferramentas de síntese lógica, como o Yosys (WOLF; GLASER; KEPLER, 2013). Essa integração é impulsionada pelo reconhecimento do ABC como uma ferramenta complementar valiosa para o nosso fluxo de trabalho.Uma outra característica importante é a capacidade de realizar conversões entre os formatos frequentemente utilizados nessa etapa, como *Programmable Logic Array* (PLA), *Equation* (EQN), AIG,Verilog. Neste trabalho, a ferramenta tem o objetivo de aplicar conversões de formatos no nosso caso transformando um arquivos AIG para Verilog com intuito de integrar com outra ferramenta de código aberto crucial para o funcionamento do fluxo planejado: o OpenROAD.

O Yosys (WOLF; GLASER; KEPLER, 2013) é um *framework* de código livre utilizado na síntese RTL em Verilog. Ele desempenha um papel fundamental na síntese lógica do OpenROAD, oferecendo um suporte abrangente ao Verilog-2005 e fornecendo um conjunto básico de algoritmos de síntese adequados para diversas áreas de aplicação. Sua integração no fluxo do OpenROAD permite a realização eficiente da síntese lógica e a geração de resultados de alta qualidade.

3 MOTIVAÇÃO - FLUXO DE OTIMIZAÇÃO LÓGICA BASEADO EM CGP

A motivação deste trabalho surge das avaliações preliminares da ferramenta desenvolvida por um grupo de pesquisadores da UFSC para fornecer uma proposta de otimização lógica baseada em programação genética cartesiana (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022; BERNDT; CAMPOS *et al.*, 2021). A ferramenta surgiu no contexto da competição do IWLS 2020 (IWLS'20 ORGANIZING COMMITTEE, 2020). A ferramenta proposta possui como objetivo principal otimizar a solução visando a melhor acurácia possível, e o menor número de nodos. Este fluxo de otimização lógica proposto é a base de avaliação deste trabalho.

Apesar da ferramenta ter reportado bons resultados na otimização conjunta dos fatores (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022), fica a pergunta de pesquisa: Como esta abordagem de otimização lógica pode afetar, positiva ou negativamente, os resultados de síntese física de um circuito.

Esta é a questão central deste trabalho. Para facilitar a compreensão desta ferramenta desenvolvida e que serve de ponto base desta pesquisa, este Capítulo apresenta os principais conceitos de programação genética cartesiana e as principais características da ferramenta de otimização lógica baseada em CGP.

3.1 PROGRAMAÇÃO GENÉTICA CARTESIANA

A Programação Genética Cartesiana é uma técnica de computação evolucionária (MILLER, Julian F., 2011). Originalmente, utiliza grafos dirigidos acíclicos representados como uma grade de duas dimensões de nodos computacionais para representar programas (MILLER, Julian F., 2011). Tipicamente, a cada geração, uma população de programas é avaliada para que se estabeleça a acurácia de cada um de seus indivíduos e, em seguida, uma nova população é formada pelo programa com maior acurácia (ou seja, que produz o maior número de saídas esperadas) e novos indivíduos gerados a partir de mutações (pequenas mudanças) de tal programa (MILLER, Julian F., 2011). O termo "cartesiano"em CGP é usado porque as soluções candidatas são compostas por uma rede bidimensional de nós. O CGP é um algoritmo de busca estocástica que utiliza uma abordagem evolutiva para explorar o espaço de parâmetros da solução, geralmente começando a partir de uma população aleatória de soluções candidatas.

É uma abordagem baseada em população que frequentemente utiliza as Estratégias de Evolução (1+λ)- Estratégias de Evolução (ES) (MILLER, Julian Francis, 2019) para buscar o espaço de parâmetros.

O CGP é utilizado em inúmeras aplicações, como a otimização do processo de aprendizado de redes neurais convolucionais (SUGANUMA *et al.*, 2020). Também foi explorado como um mecanismo na detecção, identificação e correção de falhas em nível de software (KHALILIAN; BARAANI-DASTJERDI; ZAMANI, 2021). Recentemente, o

CGP tem sido empregado na otimização de problemas de circuitos integrados, onde cada indivíduo é um circuito, representado por uma matriz inteira bidimensional descrevendo as funções e as conexões entre os nós (HODAN; MRAZEK; VASICEK, 2021; KOCNOVA; VASICEK, 2020; KOCNOVÁ; VASICEK, 2021; BERNDT, A.; ABREU, B. A. de *et al.*, 2021).

A Figura 5 ilustra o processo de busca CGP, apresentando um CGP hipotético com o formato XAIG (HÁLEČEK; FIŠER; SCHMIDT, 2018). Essa estrutura compreende todos os recursos da estrutura AIG tradicional, com a adição de que os nós podem ser compostos por lógica XOR, não apenas ANDs. Os valores de aptidão são apresentados como valores de acurácia. No exemplo, a população CGP contém três indivíduos, e ela é ajustada por três gerações em busca de uma solução. Em nosso exemplo, a geração 0 é criada aleatoriamente, o que significa que os nós têm funções lógicas aleatórias, e suas conexões invertidas ou diretas são configuradas aleatoriamente.

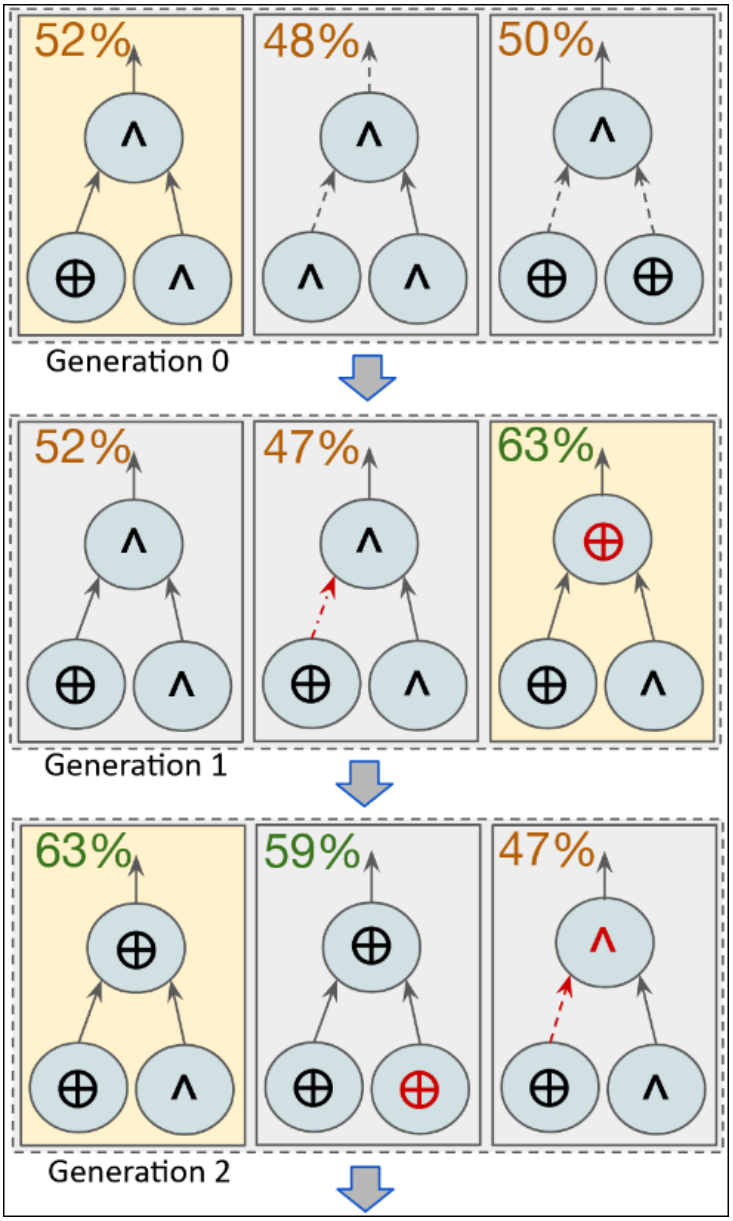
A Figura 5 também apresenta a acurácia da aptidão de cada AIG considerando a inicialização aleatória. O indivíduo com o maior valor de acurácia de aptidão (destacado em amarelo) é escolhido como o *pai* para toda a próxima geração. Todos os três indivíduos na geração 1 são descendentes do mesmo ancestral, que tem uma acurácia de 52%, embora dois indivíduos tenham valores de acurácia diferentes do seu ancestral. Isso ocorre porque eles sofreram mutações (marcadas em vermelho) durante seu processo de procriação. O indivíduo do meio na geração 1 teve uma conexão mutada para uma inversão, e o indivíduo da direita teve a função do seu último nó mutada de AND para XOR. Essas duas mutações refletiram em uma acurácia menor e maior, respectivamente, para cada indivíduo. Observe que o indivíduo da direita na geração 1 é escolhido como o pai para a próxima geração 2. As mutações na geração 2 não foram capazes de alcançar nenhuma melhoria na acurácia da aptidão. Portanto, se a execução fosse até a geração 3, seu pai seria o mesmo que na geração 2. O exemplo mostrado na Figura 5 segue uma configuração de aprendizado da regra $(1+2)-ES$, indicando que temos um ancestral e três indivíduos para cada geração. O sinal $'+'$ indica elitismo na procriação da geração, o que significa que o indivíduo mais apto sempre será copiado para a próxima geração (HANSEN; ARNOLD; AUGER, 2015).

Conforme demonstrado por (MILANO; PAGLIUCA; NOLFI, 2019), a busca do CGP tem uma convergência melhor se soluções fenotipicamente maiores forem consideradas candidatos preferenciais ao analisar os *scores* de aptidão dos indivíduos. Os fluxos propostos utilizam essa técnica selecionando AIGs com tamanhos funcionais maiores quando ocorre um empate nos valores de aptidão.

3.2 FLUXO DE OTIMIZAÇÃO LÓGICA BASEADA EM CGP

Na otimização lógica, os nodos representam portas lógicas. Para isso, os circuitos podem ser representados por AIGs. Em (BERNDT, A.; ABREU, B. A. de *et al.*, 2022), um fluxo de otimização lógica baseado em CGP é proposto, buscando melhorar a acurácia e o

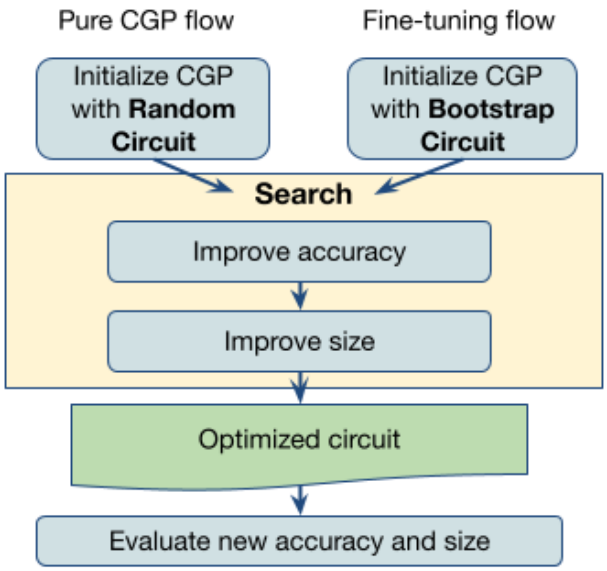
Figura 5 – Desenvolvimento evolutivo CGP



Fonte: (BERNDT, A.; ABREU, B. A. de *et al.*, 2022).

tamanho dos circuitos aproximados gerados. A minimização lógica com CGP mostrou-se efetiva quando considerados simultaneamente a acurácia e o número de nodos obtidos. O fluxo de síntese baseado em CGP é apresentado na Figura 7. O objetivo deste fluxo é otimizar tabelas de verdade completas e incompletas, além de circuitos previamente otimizados representados no formato AIG. Ele oferece soluções de minimização lógica a partir de tabelas de verdade ou ajuste fino para blocos de circuitos, considerando cenários imprecisos, como na computação aproximada. O fluxo também trabalha com funções desconhecidas em blocos de propriedade intelectual ou funções de entrada maiores, onde a utilização de uma tabela de verdade completa pode ser inviável para métodos de minimização tradicionais.

Figura 6 – Fluxo CGP (BERNDT, A.; ABREU, B. A. de *et al.*, 2022)



Fonte: (MISHCHENKO, Alan; CHATTERJEE, Satrajit; BRAYTON, Robert, 2006)

O fluxo proposto pode ser iniciado de duas maneiras distintas: Com a inicialização aleatória, denominada *Pure CGP*, que realiza uma otimização baseada apenas em CGP a partir de uma tabela de verdade completa ou incompleta; ou com a inicialização *bootstrapped*, na qual o fluxo baseado em CGP inicia-se com um circuito previamente otimizado no formato AIG, permitindo um ajuste fino.

A primeira fase do fluxo de otimização lógica baseado em CGP é a de inicialização aleatória, que consiste na geração de uma população inicial aleatória de indivíduos, ou seja, de circuitos no formato AIG.

A região marcada em amarelo na Figura 7 engloba as etapas relacionadas à inicialização e otimização do CGP, realizando as evoluções e modificações necessárias em cada geração genética do algoritmo, até melhorar a acurácia. Após esta primeira etapa, o circuito é otimizado novamente com o algoritmo genético, entretanto, agora considerando como métrica de otimização o número de nodos do CGP. Tais circuitos passam por uma fase de evolução, repetida ao longo de diversas gerações. Nesta, a acurácia dos indivíduos é avaliada a partir de *mini-batches*, formados por uma parcela aleatoriamente extraída das linhas da tabela-verdade utilizada como conjunto de treinamento. A quantidade de linhas que compõem cada um dos *mini-batches* é dada pelo hiperparâmetro *batch size*, enquanto que a quantidade de gerações que um *mini-batch* deve permanecer inalterado é definida por *change each*.Escolhe-se então o circuito com maior acurácia para ser o pai da próxima geração de circuitos e os mesmos são gerados a partir de mutações de tal pai. Após o processo de evolução ser concluído, o circuito criado e otimizado pelo fluxo tem sua acurácia avaliada e um arquivo que contém a descrição do mesmo é gerado.

A saída do fluxo é um circuito otimizado que será avaliado quanto à métricas de síntese lógica, tais como: número de nodos, profundidade lógica e acurácia.

3.3 CONSIDERAÇÕES FINAIS

Embora esta ferramenta tenha sido desenvolvida no grupo de pesquisa da UFSC, é importante analisar outras soluções que abordam o mesmo problema de otimização lógica de circuitos aproximados. Estes trabalhos serão discutidos no Capítulo 4 e avaliados seguindo a mesma metodologia.

Ao final deste trabalho, será fornecida uma visão geral do comportamento final da síntese de circuitos aproximados. Além disso, é fundamental considerar como essas ferramentas impactam a síntese como um todo avaliando suas métricas. A intenção é promover um fluxo totalmente *open source*, visando a acessibilidade e a colaboração na comunidade de EDA.

4 TRABALHOS RELACIONADOS

Este trabalho tem como foco avaliar o impacto da otimização lógica aplicada no desenvolvimento de CIs em casos em cenários de computação aproximada, e neste tópico serão apresentados alguns estudos recentes, com objetivo de explorar o estado da arte.

Na otimização lógica, alguns algoritmos clássicos desempenham esse papel: o Mapa de Karnaugh (KARNAUGH, 1953), técnica visual para simplificar expressões booleanas, útil para circuitos com poucas variáveis; o Quine-McCluskey (QUINE, 1955), algoritmo para simplificar funções booleanas identificando implicações entre termos e agrupando-os com base no menor número de diferenças; e o Espresso (RUDELL; SANGIOVANNI-VINCENTELLI, 1987; BRAYTON, Robert K. *et al.*, 1984), implementação eficiente do Quine-McCluskey, utilizado em *design* de circuitos para otimizar expressões booleanas e reduzir o número de portas lógicas necessárias.

No entanto, esses algoritmos podem apresentar um elevado custo computacional, especialmente para o processamento de circuitos com muitas entradas. Dessa forma, surgem estudos como alternativas para o aprimoramento da síntese lógica, juntamente com a aplicação de Aprendizado de Máquina. Recentemente, técnicas de Aprendizado de Máquina passaram a ser utilizadas na área de EDA, ainda possuindo bastante espaço para crescimento (BEEREL; PEDRAM, 2018; PANDEY, 2018).

Em conjunto com o crescente avanço em pesquisas na área da computação aproximada, uma vez que essa abordagem possibilita reduzir a área e o consumo de energia de circuitos desenvolvidos. Além disso a computação aproximada apresenta oportunidades de aplicação em diversas áreas da computação, as quais incluem a utilização em algoritmos de *Deep Learning* e desenvolvimento de CIs (VENKATARAMANI *et al.*, 2015). Um desses estudo é o trabalho de Masadeh, Hasan e Tahar (2018), onde os autores exploraram uma síntese lógica, propondo a implementação de Computação Aproximada em Cis, utilizando circuitos somadores, multiplicadores e divisores, uma vez que, por terem funcionalidades bem conhecidas, permitem o desenvolvimento de lógicas de aproximação feitas manualmente (MASADEH; HASAN; TAHAR, 2018).

Observamos algumas iniciativas para explorar ML, a maioria inicialmente motivada pelo concurso de síntese lógica proposto na conferência IWLS em 2020 (IWLS'20 ORGANIZING COMMITTEE, 2020). Algumas técnicas foram propostas no IWLS para o aprendizado lógico e a síntese de circuitos aproximados, explorando abordagens de ML como: DT (TENACE; CALIMERA, 2018a; ABREU *et al.*, 2021; ZENG; DAVOODI; TOPALOGLU, 2021), *Random Forests* (ZENG; DAVOODI; TOPALOGLU, 2021), *Look-Up Tables networks* (LUT) (CHATTERJEE, 2018; MIYASAKA *et al.*, 2021), Redes Neurais (MIYASAKA *et al.*, 2021) e Programação Genética Cartesiana - CGP (BERNDT, A.; ABREU, B. A. de *et al.*, 2021) (BERNDT, A.; ABREU, B. A. de *et al.*, 2022). Uma visão geral das soluções propostas pelos demais competidores foi reportada em (RAI; NETO

et al., 2021). É interessante reportar que mais de um time explorou árvores de decisão, mas sem resultados tão expressivos em alguma métrica para merecerem maior destaque que a solução de (TENACE; CALIMERA, 2018b). Desta forma, podemos resumir as diferenças entre os três métodos de síntese de circuitos aproximados pela técnica de *machine learning* adotada e pelo objetivo de otimização principal, conforme mostra a Tabela 3.

Tabela 3 – Trabalhos Relacionados

Referencia	Métodos de *Machine Learning*	Principal objetivo
(BERNDT, A. A. S.; ABREU, B. *et al.*, 2022)	CGP	otimização conjunta de número de nodos e de acurácia
(MIYASAKA *et al.*, 2021)	LUT e redes neurais	otimização de acurácia
(TENACE; CALIMERA, 2018b)	DT	otimização de número de nodos

Fonte: (O próprio autor, 2024)

Em (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022), foi proposto o método de otimização baseado em população com Programação Genética Cartesiana apresentado no Capítulo 3. A ferramenta de otimização lógica baseada em CGP explora o algoritmo genético para aprendizado do comportamento do circuito descrito via uma tabela verdade (BERNDT, A. A. S.; ABREU, B. *et al.*, 2022). Esta ferramenta demonstra potencial na redução do número de nodos necessários, o que pode resultar em menos área, menos consumo de energia e menores atrasos críticos. Entre os competidores, a abordagem de Programação Genética Cartesiana também ganhou visibilidade devido à sua diversidade em relação às técnicas de ML exploradas.

O vencedor do concurso IWLS 2020 explorou uma abordagem *Mixed* de ML, combinando LUT e redes neurais, fornecendo os resultados com melhor acurácia (MIYASAKA *et al.*, 2021). Esta solução, composta de várias técnicas de ML, avaliava os resultados de cada técnica, escolhendo a alternativa com maior acurácia dentre as soluções para um único circuito. Esta exploração paralela de várias técnicas pode ser custosa em tempo de execução. Os dados de tempo de execução de treinamento não foram considerados na avaliação do concurso do IWLS 2020 e não foram reportados pelos autores.

Por outro lado, uma abordagem baseada em árvores de decisão DT recebeu menção honrosa por fornecer soluções com menos nós (TENACE; CALIMERA, 2018b), ou seja, entregando o circuito com a menor área, entretanto, com níveis baixos de acurácia em geral para os circuitos avaliados.

Pelos destaque destes três métodos de síntese para circuitos aproximados na competição do IWLS 2020, estes trabalhos foram escolhidos para serem analisados conjuntamente com a otimização baseada em CGP desenvolvida no grupo. Desta forma, foram conseguidos os resultados de síntese lógica diretamente com os três times, no formato AIG. A

análise do impacto na síntese física seguirá um fluxo comum partindo das descrições AIG fornecidas.

5 DESENVOLVIMENTO

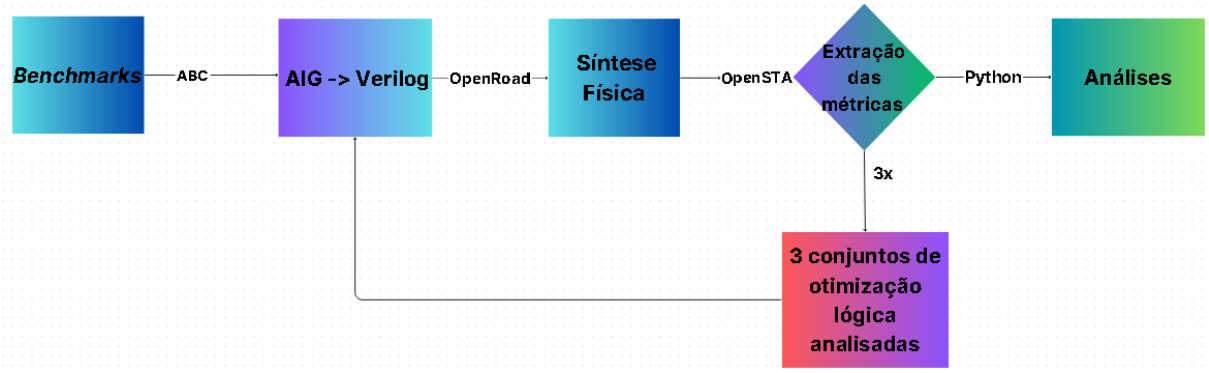
Com o objetivo de avaliar diferentes abordagens de otimização lógica para circuitos aproximados e seus impactos nas sínteses após a síntese lógica, este trabalho adota um fluxo de projeto para permitir uma avaliação igualitária das técnicas de otimização lógica quanto as métricas de *delay*, potência e área. A Figura 7 ilustra a visão geral do processo de desenvolvimento deste trabalho e as ferramentas utilizadas em cada fase.

Nos últimos anos, o compartilhamento de conhecimento entre desenvolvedores cresceu significativamente com as ferramentas de código aberto. A área científica pode tirar proveito dessas ferramentas e ajudar na criação de novas, ampliando o horizonte da tecnologia. Por isso, decidiu-se pela utilização apenas de *softwares* de código aberto no desenvolvimento deste projeto.

O desenvolvimento deste trabalho foi dividido em cinco etapas distintas: 1) escolha dos conjuntos de dados (*benchmarks*), 2) conversão dos arquivos, 3) integração com o OpenROAD, 4) extração de métricas, e, por fim, 5) uma análise de resultados. A seguir, serão detalhados cada um destes passos.

Este estudo selecionou para a avaliação dos resultados de síntese física as três técnicas que obtiveram bons resultados na competição IWLS 2020 em relação às métricas tradicionais de síntese lógica: a técnica de *Mixed-ML* (MIYASAKA *et al.*, 2021), que alcançou os melhores resultados de acurácia; a técnica DT (TENACE; CALIMERA, 2018b), que proporcionou soluções de circuito com a menor área; e a abordagem de Programação Genética Cartesiana (BERNDT, A.; ABREU, B. A. de *et al.*, 2022), que foi o ponto inicial deste trabalho. É relevante destacar que esse fluxo proposto é capaz de utilizar qualquer circuito descrito no formato AIG como ponto de partida, possibilitando a exploração de AIGs otimizados por diferentes técnicas e a aplicação do fluxo CGP como uma etapa adicional de otimização de ajuste fino.

Figura 7 – Fluxo de Desenvolvimento do trabalho



Fonte: (O próprio autor, 2024)

5.1 PASSO 1:*BENCHMARKS*

Os experimentos foram realizados com o conjunto de *benchmarks* fornecido pelo concurso IWLS 2020 (IWLS 2020 PROGRAMMING CONTEST, 2020). A Tabela 4 apresenta o conjunto de *benchmarks*, composto por 100 funções incompletas. As funções representam diferentes tipos de lógica, como circuitos aritméticos, de lógica aleatória, e também de aplicações dentro do domínio de *machine learning*, com soluções para problemas clássicos como o *Modified National Institute of Standards and Technology* (MNIST) de reconhecimento de números, e o *Canadian Institute for Advanced Research* (CIFAR-10) de classificação de imagens. Estas funções diferem significativamente no número de entradas comparado as funções aritméticas.

Cada função é composta por três conjuntos de dados: treino, validação e teste, fornecidos como tabelas de verdade no formato PLA. As técnicas de otimização utilizaram os conjuntos de dados de treino e validação durante o treino e validação, e o conjunto de dados de teste para avaliar os experimentos comparativos. Todos os valores de acurácia apresentados aqui são baseados neste último conjunto,e foram extraídos utilizando a ferramenta ABC.

Tabela 4 – Informações sobre o conjunto de circuitos

Função	Tipo de lógica	# Entradas Mín	# Entradas Max	Domínio lógico
00-09	Somadores	32	512	Aritmético
10-19	Subtratores, Restos	32	512	
20-29	Multiplexadores	16	256	
30-39	Comparadores	20	200	
40-49	Raiz quadrada	10	256	
50-59	PicoJava	16	394	Aleatória
60-74	MCNC *benchmarks*	16	52	
75-79	Funções simétricas	16	16	
80-89	MNIST	196	196	Aprendizado de Máquina
90-99	CIFAR-10	768	768	

Fonte: Adaptado de: (RAI; AL., 2021) e (ZENG; DAVOODI; TOPALOGLU, 2021).

5.2 PASSO 2: CONVERSÃO DE FORMATO

O formato de descrição de tabelas verdades PLA é um formato textual, onde são definidas as saídas em função das entradas, praticamente uma reprodução textual da tabela verdade. Este formato é bastante utilizado na especificação de tabelas verdades dada a simplicidade e facilidade na visualização da informação. Entretanto, para o processamento dos algoritmos de síntese, o formato AIG é o mais recomendado, por reduzir a quantidade de memória necessária para representar a estrutura de dados e por facilitar o desenvolvimento de algoritmos de busca nos grafos gerados, facilitando as tarefas de

otimização. As alternativas de síntese lógica utilizaram as descrições PLA como entrada de seus algoritmos, mas a solução final deveria ser fornecida no formato AIG.

As ferramentas ABC e Yosys desempenham papéis cruciais neste etapa, possibilitando a transformação de descrições de circuitos lógicos otimizados e prontos para a próxima etapa do processo de desenvolvimento de *hardware*, chamada de síntese física. Foi realizada uma conversão dos arquivos AIGs para o formato *Verilog*, utilizando a ferramenta de código aberto Yosys.Durante esse processo, foi elaborado um *script* em *Bash* para automatizar a conversão para os 100 soluções de cada técnica. Com esse resultado em mãos, tornou-se possível mapeá-lo para a plataforma de síntese desejada.

A saída deste passo pode ser utilizada para integração com diferentes plataformas de síntese, tais como FPGAS. Neste trabalho, optou-se pela síntese de circuito integrado baseada em biblioteca de células. Desta forma, após essa etapa, o estudo concentrou-se em integrar com a ferramenta OpenROAD, que possibilita este tipo de síntese e é uma ferramenta aberta e gratuita.

5.3 PASSO 3: SÍNTESE DO HARDWARE

Com o objetivo de avaliar a eficiência de cada estratégia de otimização lógica proposta, foi realizada a integração na versão mais atual do OpenROAD. Os circuitos foram implementados utilizando a biblioteca de células de 45 nm da *Nangate* disponibilizada pelo OpenROAD (OPENROAD, 2022a), seguindo o fluxo apresentado na Figura 8. Vale lembrar que esse fluxo incluiu uma nova síntese lógica realizada pelas ferramentas ABC e Yosys, sendo esta etapa realizada para todas as técnicas de otimização lógica.

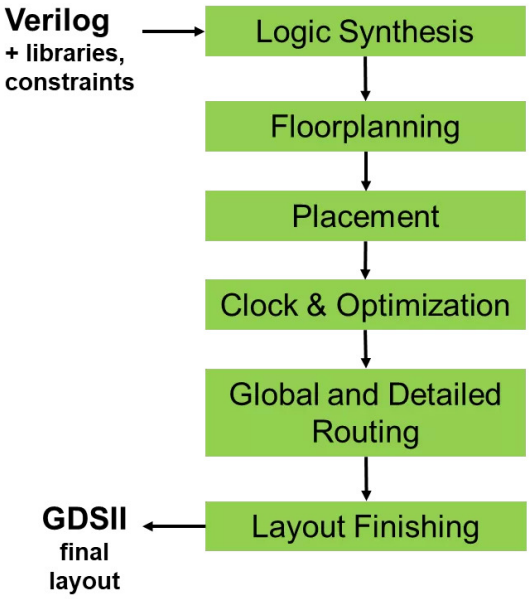
A síntese física é composta pelas etapas de *Floorplaning*, posicionamento, geração da rede de *clock* e otimização, até a conclusão do roteamento sem violações de regras de projeto e finalização do *layout* no formato a ser enviado para a fabricação. A síntese neste experimento foi configurada sem restrição de *timing*, para permitir a comparação do atraso. As demais configurações da síntese foram mantidas na configuração padrão do OpenROAD, sem ajustes específicos para cada circuito ou síntese, na tentativa de fornecer maior similaridade nas condições de síntese para todas as análises.

Para essa fase, foram elaborados *scripts* em *Bash* e *Python* com o objetivo de configurar o ambiente do OpenROAD. Esses *scripts* desempenharam um papel crucial no processo, automatizando a geração dos arquivos de síntese física para cada equipe. Isso possibilitou a extração das métricas desejadas e foi fundamental para o desenvolvimento do trabalho

5.4 PASSO 4: EXTRAÇÃO DE MÉTRICAS

Após a finalização da síntese física, foram selecionadas quatro métricas para comparar os diferentes fluxos: o tamanho do circuito gerado, a potência total, o atraso crítico

Figura 8 – Fluxo de síntese utilizado pelo OpenROAD



Fonte: (OPENROAD, 2022a)

do circuito (*delay*) e a acurácia. Para extrair essas métricas, utilizamos a ferramenta *OpenSTA* para fornecer os dados de*timing* e potência após as etapas de síntese física. Vale destacar que a análise foi realizada sobre potência total, mas uma parcela significativa dessa potência foi a potência estática, pois o chaveamento das entradas não foi considerado. Dentro do ambiente do OpenROAD, extraímos a métrica de área de cada circuito. Por fim, utilizando a ferramenta ABC, como mencionado anteriormente, extraímos a métrica de acurácia como definida para a competição do IWLS. Esta métrica calcula o percentual de erros e acertos na saída esperada de linhas da tabela verdade. Neste contexto, esta acurácia está mais próxima do conceito de erro relativo acumulado. Isso significa que a métrica de acurácia do ABC reflete a relação de erros relativos nas saídas dos circuitos.

Para automatizar o processamento dos circuitos também foi desenvolvido um *script* contendo sequências de comandos que visaram realizar o processo de extração das métricas dos diferentes arquivos de relatórios gerados no fluxo OpenROAD para os múltiplos circuitos.

5.5 PASSO 5: ANÁLISES

Após a extração dos dados, empregamos o ambiente *Jupyter* em linguagem *Python* para realizar a manipulação e análise dos dados obtidos, isto proporcionou explorar os resultados, permitindo uma variedade de operações desde a limpeza inicial dos dados até análises mais complexas e visualizações desejadas. Utilizamos as bibliotecas de análise de dados padrão em *Python*, como *Pandas*, *NumPy* e *Matplotlib*, para processar os dados extraídos, calcular estatísticas relevantes.

Nessa etapa, definimos quatro conjuntos de análise com objetivos específicos.

O Conjunto A engloba todos os circuitos avaliados com os três métodos de otimização lógica, permitindo uma análise global do desempenho do fluxo CGP em comparação com os métodos de referência.

O Conjunto B buscamos examinar o desempenho do fluxo CGP e dos métodos de referência em situações de alto desempenho.

No Conjunto C, consideramos circuitos com resultados de acurácia semelhantes. Essa análise permite investigar como o fluxo CGP se comporta em cenários onde a acurácia é muito próxima e não se faz tão relevante.

Por fim, o Conjunto D consiste em circuitos que apresentam diferenças significativas na acurácia entre os métodos de otimização lógica. Aqui, exploramos como o fluxo CGP se destaca ou enfrenta desafios dentro do *benchmark*.

Esses conjuntos de análise foram definidos para proporcionar uma compreensão abrangente do desempenho do fluxo CGP em relação aos métodos comparados, considerando diferentes cenários e critérios de avaliação que serão explorados no próximo capítulo.

6 RESULTADOS

A avaliação do fluxo CGP envolve a análise de várias métricas para determina sua eficácia. As métricas escolhidas para esta avaliação foram potência total, *delay* crítico, área e acurácia. O objetivo principal é comparar os resultados do fluxo proposto baseado em CGP com os resultados das técnicas que obtiveram maior acurácia e menor área no IWLS 2020. Desta forma, esta abordagem valida as vantagens e desvantagens comparativas entre os métodos. Os três métodos de otimização baseados em ML foram avaliados. Os resultados foram divididos em quatro conjuntos para uma análise mais aprofundada. A definição dos conjuntos é a seguinte:

- Conjunto A abrange todos os circuitos avaliados com os três métodos de otimização lógica.
- Conjunto B inclui apenas circuitos que alcançaram uma acurácia superior a 90% em todos os métodos testados.
- Conjunto C representa circuitos com resultados de acurácia semelhantes, com uma variação máxima de 5% entre os métodos de otimização lógica.
- Conjunto D consiste em circuitos que exibem diferenças significativas na acurácia entre os métodos de otimização lógica, sendo o complemento do Conjunto C.

Essa organização dos conjuntos permite uma análise detalhada das estratégias de otimização e seus impactos nos resultados. Primeiramente vamos abordar os resultados do conjunto A, apresentados na Tabela 5, dando uma visão geral dos resultados ao considerar todos os *benchmarks*.

Tabela 5 – Tabela de Resultados Conjunto A

Métricas	Mixed-ML	DT	CGP
Delay (ns)	0,54	0,42	**0,27**
Potência (μW)	49,22	**19,01**	19,05
Área (μm^2)	2014,15	843,91	**837,69**
Acurácia (%)	**89,67**	85,87	84,35

Fonte: (O próprio autor, 2024)

Observando a Tabela 5 , o *Mixed*-ML entrega consideravelmente a otimização lógica com a maior acurácia média (89,67%). Quando comparada a técnica DT, houve uma redução de 3,8%. A técnica de CGP, por sua vez, registrou a menor acurácia geral,

com uma redução média de 6 % comparado ao *Mixed*-ML. Entretanto, tal resultado pode ainda permanece competitivo para aplicações tolerantes a erro.

Apesar da redução de 6% na acurácia, observa-se que a estratégia CGP se destaca nas métricas físicas avaliadas. Os resultados indicam que o CGP alcançou o menor *delay* médio entre as três abordagens de otimização lógica com 0,27 ns, cerca de 35% de redução em comparação com a segunda técnica com o melhor resultado de *delay*, a estratégia baseada em DT. Observa-se que tanto as técnicas baseadas em DT quanto em CGP exibiram resultados de potência semelhantes, ambos em torno de 19 µW, enquanto o resultado de *Mixed*-ML é aproximadamente 61% maior que os outros.

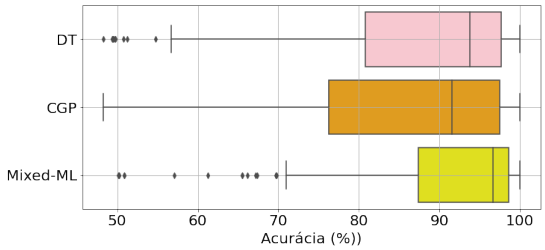
Além disso, os resultados demonstram que os resultados da técnica baseada em CGP também resultaram na menor área (837,69 μm) para os circuitos, com uma redução média de 0,7 % em comparação com seu concorrente mais próximo, DT. Esta é uma observação relevante uma vez que a otimização lógica baseada em DT fornece a solução de otimização lógica com menor número de nodos considerando o AIG gerado. Isso reflete a necessidade de inserção de células extras durante a síntese destes circuitos, sendo na maioria dos casos, relacionados a inserção de *buffers.*

A média é sensível a valores extremos, pois é afetada por todos os valores no conjunto de dados. Já a mediana, é mais robusta a esses valores atípicos, pois é menos influenciada por eles, além disso, é uma medida de posição central mais resistente a variações. Desta forma, apresentamos os *boxplots* dos resultados para o Conjunto A na Figura 9. O *boxplot* nos fornece uma análise visual da posição do conjunto de dados, dispersão, simetria, caudas e valores atípicos. Em relação à posição dos dados, observamos a linha central do retângulo (a mediana). A dispersão dos dados é representada pelo tamanho do retângulo. A simetria é observada pela posição da linha mediana em relação ao centro do retângulo. Vale ressaltar que a mediana é a medida de tendência central mais adequada quando os dados têm uma distribuição assimétrica, pois a média aritmética é influenciada pelos valores extremos. Por fim, as linhas que se estendem do retângulo até os valores atípicos podem fornecer o comprimento das caudas da distribuição.

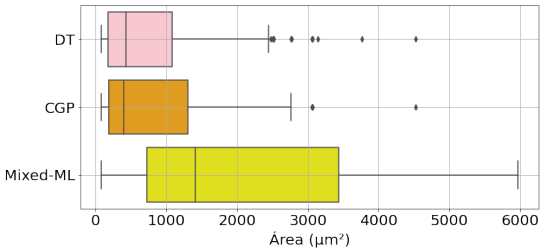
A avaliação mais detalhada do conjunto A, apresentada na Figura 9, nos mostra quão difícil é para todas as abordagens aprender algumas funções, se refletindo em grandes desvios nos resultados de acurácia. Aproximadamente 14,3 % do *benchmark* consiste em um subconjunto de funções para as quais nenhuma das abordagens pode alcançar mais de 60 % de acurácia, mostrando o quão desafiador é o problema de aprendizado lógico para algumas funções. Como mencionado por (MIYASAKA *et al.*, 2021), somadores, multiplicadores e raízes quadradas geralmente são a lógica mais difícil de aprender, compostos por árvores de somadores e multiplicadores.

Estes desvios na acurácia observados nos *boxplots*, motivaram a avaliação em grupos dos circuitos, inicialmente restringindo a segunda avaliação para circuitos com acurácia elevada. No Conjunto B, onde a análise se concentrou em circuitos com exatidão superior

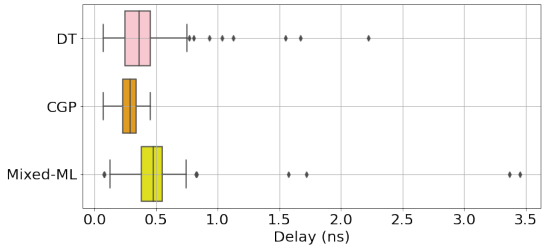
Figura 9 – Resultados de síntese física para a tecnologia de 45 nm do Conjunto A comparados com CGP, DT e *Mixed*-ML



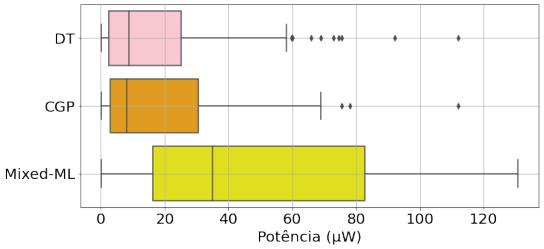
(a) *Boxplot* Acurácia - Conjunto A



(b) *Boxplot* Área - Conjunto A



(c) *Boxplot Delay* - Conjunto A

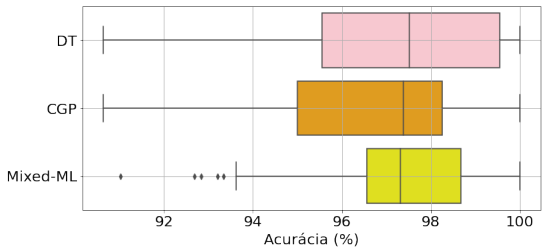


(d) *Boxplot* Potência - Conjunto A

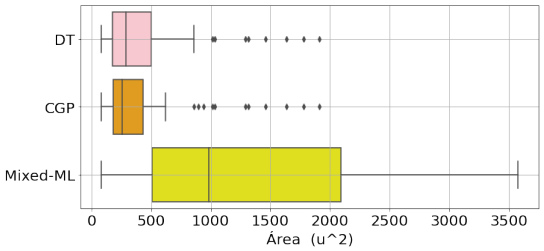
Fonte: (O próprio autor, 2024)

a 90%, a otimização lógica baseada em CGP foi bem em relação as métricas propostas. Na Figura 10, podemos observar a dispersão dos dados. Embora tenha registrado uma acurácia ligeiramente inferior em comparação com outras abordagens, o CGP alcança bons resultados com uma acurácia média de 96%.

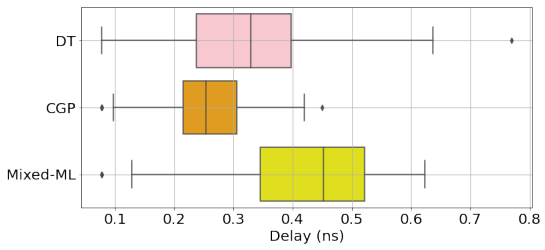
Figura 10 – Resultados de síntese física para a tecnologia de 45 nm do Conjunto B comparados com CGP, DT e *Mixed*-ML



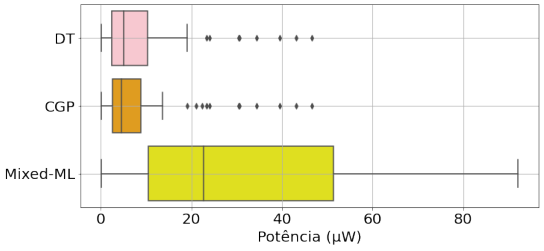
(a) *Boxplot* Acurácia - Conjunto B



(b) *Boxplot* Área - Conjunto B



(c) *Boxplot Delay* - Conjunto B



(d) *Boxplot* Potência - Conjunto B

Fonte: (O próprio autor, 2024)

Mais uma vez, no grupo B, a abordagem CGP mostrou compensações para essa diferença ao observar as métricas físicas após a síntese: observou-se uma diminuição média de atraso de 22,50% em relação à DT e 39,14% a técnica *Mixed*-ML, uma redução de área de 2,89% em comparação com a DT e *Mixed*-Ml de 64,00% , e também obteve reduções de potência de 2,24% em comparado com a DT e 68,63% em relação à *Mixed*-ML, todos os valores médios estão apresentados na Tabela 6, esses resultados destacam o desempenho compensatório do CGP em relação à acurácia.

Tabela 6 – Tabela de Resultados Conjunto B

Métricas	Mixed-ML	DT	CGP
Delay (ns)	0,41	0,33	**0,25**
Potência (μW)	31,26	10,03	**9,80**
Área (μm^2)	1295,58	480,34	**466,44**
Acurácia (%)	**97,20**	97,12	96,54

Fonte: (O próprio autor, 2024)

Observando os bons resultados da técnica baseada em CGP neste conjunto, mudamos o cenário para circuitos com acurácia semelhantes nas três abordagens de otimização lógica. Para o Conjunto C, os resultados estão ilustrados na Tabela 7, a otimização lógica baseada em CGP também se destacou na avaliação física, com uma redução média de *delay* de 34,80%, potência de 7,87% e área de 7,72% em comparação com a abordagem DT, novamente a segunda colocada. Em relação à abordagem *Mixed*-ML, os ganhos obtidos com a otimização baseada em CGP são de 40,72%, 66,25% e 62,85% para *delay*, potência e área, respectivamente.

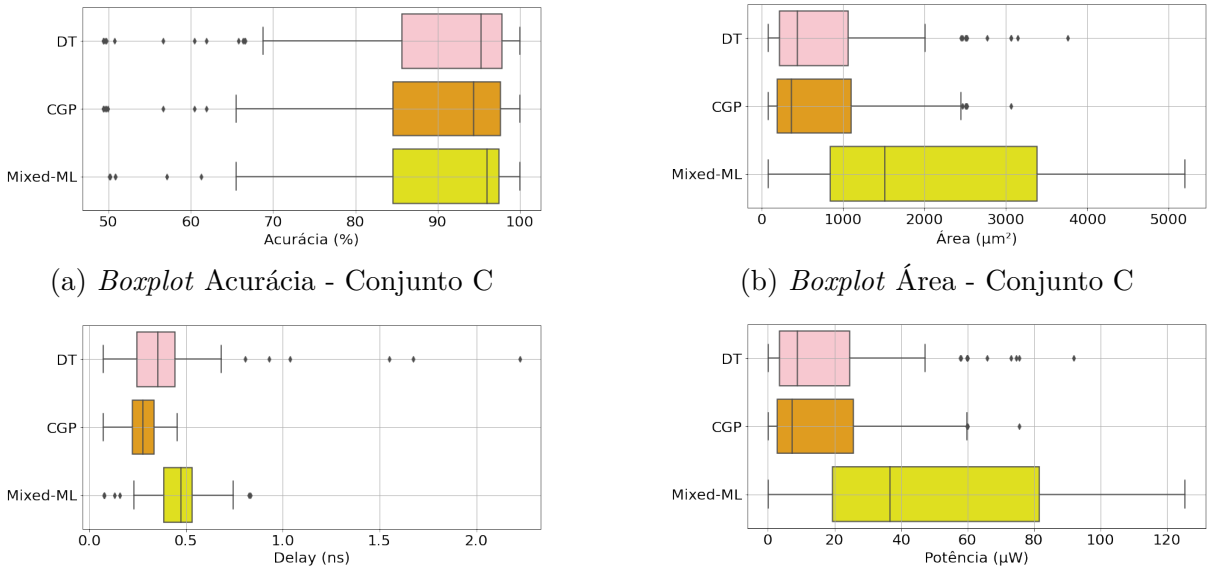
Tabela 7 – Tabela de Resultados Conjunto C

Métricas	Mixed-ML	DT	CGP
Delay (ns)	0,45	0,41	**0,27**
Potência (μW)	50,63	18,55	**17,09**
Área (μm^2)	2057,49	828,34	**764,33**
Acurácia (%)	**87,97**	87,92	87,46

Fonte: (O próprio autor, 2024)

A Figura 11 ilustra os resultados da terceira análise, considerando os circuitos com acurácia similar. Para estes casos, os circuitos considerados mantém acurácias semelhantes por definição do conjunto. Mesmo assim, pode-se observar ligeiras diferenças na mediana e desvio padrão destes circuitos em cada métrica, sendo sempre a mediana mais alta obtida para a abordagem *Mixed*-ML.

Figura 11 – Resultados de síntese física para a tecnologia de 45 nm do Conjunto C comparados com CGP, DT e *Mixed*-ML



(a) *Boxplot* Acurácia - Conjunto C

(b) *Boxplot* Área - Conjunto C

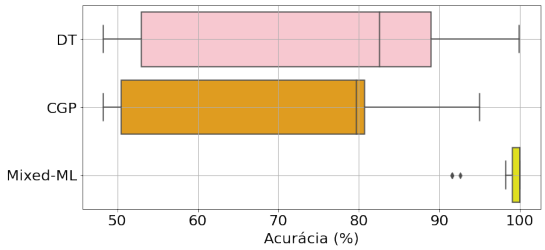
(c) *Boxplot Delay* - Conjunto C

(d) *Boxplot* Potência - Conjunto C

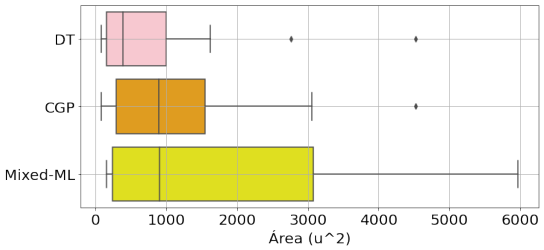
Fonte: (O próprio autor, 2024)

Dada a relevância desses resultados, foi decidido investigar se as escolhas feitas excluíam os melhores circuitos das demais técnicas, definindo assim o quarto conjunto de análise. Os *boxplots* para o conjunto D são apresentados na Figura 12. Este último conjunto considera os circuitos onde as abordagens de otimização lógica mostraram a maior divergência na acurácia, ou seja, o complemento do conjunto C. Da mesma forma que nas análises anteriores, a abordagem baseada em CGP revelou um *delay* médio menor, de 0,28 ns como observado na Tabela 8 . Isso representa uma vantagem de 37,35% em relação ao DT e de 71,92% em relação ao *Mixed*-ML. Enquanto isso, em termos de potência e área, a abordagem baseada em DT apresentou as melhores médias, com reduções de 35,89% e 96,86%, e 31,06% e 94,47%, respectivamente, em comparação com CGP e *Mixed*-ML.

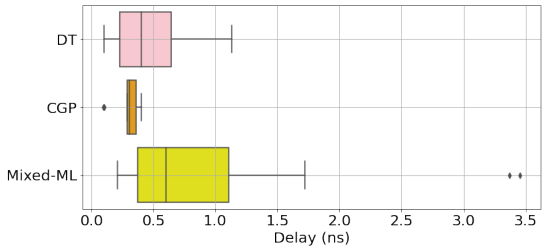
Figura 12 – Resultados de síntese física para a tecnologia de 45 nm do Conjunto D comparados com CGP, DT e *Mixed*-ML



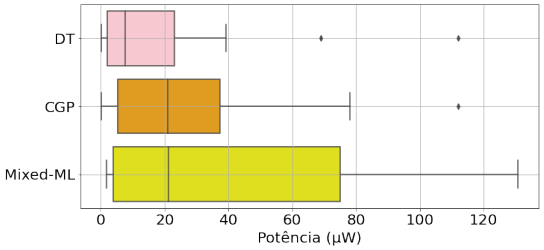
(a) *Boxplot* Acurácia - Conjunto D



(b) *Boxplot* Área - Conjunto D



(c) *Boxplot Delay* - Conjunto D



(d) *Boxplot* Potência - Conjunto D

Fonte: (O próprio autor, 2024)

Tabela 8 – Tabela de Resultados Conjunto D

Métricas	Mixed-ML	DT	CGP
Delay (ns)	1,01	0,45	**0,28**
Potência (μW)	42,04	**21,35**	29,02
Área (μm^2)	1794,60	**922,80**	1209,40
Acurácia (%)	**98,26**	75,49	68,58

Fonte: (O próprio autor, 2024)

É importante notar que mais uma vez, a abordagem *Mixed*-ML demonstrou a maior acurácia geral (98,26%). Neste conjunto, a diferença de acurácia entre *Mixed*-ML e as outras abordagens foi ainda mais pronunciada, com CGP alcançando apenas uma média de 68,58% de acurácia, enquanto a acurácia média do DT é de cerca de 75,49%. Isso destaca a superioridade da abordagem *Mixed*-ML em termos de acurácia mesmo considerando os circuitos de difícil aprendizado, como observado nos *boxplots* deste conjunto, mas isso ocorre às custas de área, potência e *delay*.

Após analisar os quatro conjuntos de resultados, é possível destacar os pontos positivos e negativos de diferentes estratégias de otimização lógica. O *Mixed*-ML se destaca como a solução com a maior acurácia. Em geral, a otimização lógica baseada em CGP

apresenta uma excelente redução no atraso, área e potência. No entanto, para um menor conjunto de circuitos, a otimização lógica baseada em DT apresenta melhores resultados de área e potência. Assim, a escolha entre abordagens dependerá das prioridades específicas do projeto, considerando os compromissos entre atraso, potência, área ocupada e acurácia. No entanto, apesar da acurácia ligeiramente menor em comparação com o *Mixed*-ML, a otimização lógica baseada em CGP permanece competitiva para a maioria dos circuitos, considerando que os resultados de potência, área e atraso são uma abordagem recomendável para aplicações de eficiência energética e tolerantes a erros.

7 CONCLUSÕES

Este estudo apresenta uma análise comparativa de diferentes estratégias de otimização lógica para circuitos integrados aproximados. Três abordagens de otimização lógica baseadas em ML foram sintetizadas para a mesma biblioteca*standard cell*, seguindo o mesmo fluxo de síntese com as mesmas restrições e condições de avaliação. Isso nos permite comparar a acurácia, potência, área e *delay*.

Ao dividir os resultados em quatro conjuntos distintos (Conjuntos A, B, C e D), foi possível explorar cenários específicos e compreender o desempenho de cada abordagem em diferentes contextos. Nestes cenários, o fluxo baseado em CGP destacou-se e emergiu como uma técnica promissora no campo da otimização lógica visando eficiência-energética e aplicações tolerantes a erros. Os resultados destacaram a capacidade da otimização lógica baseada em CGP de aprimorar o *delay*, eficiência energética e tamanho de circuitos lógicos aproximados. Observamos uma redução de mais de 50% nessas métricas em comparação com a abordagem *Mixed* de ML no cenário com todos os circuitos. Embora a abordagem de otimização lógica baseada em DT tenha demonstrado resultados competitivos em área e potência para alguns circuitos. Quando nos concentramos no *delay*, o CGP se destaca com uma redução de mais de 22% em comparação com a abordagem DT e uma redução ainda maior, superando 39%, em comparação com a abordagem *Mixed* de ML.

No entanto, é essencial notar que o sucesso do fluxo CGP está intrinsecamente relacionado ao contexto de cada projeto e aos objetivos específicos. Esta abordagem prioriza a otimização, mas pode resultar em uma redução média de acurácia de aproximadamente 6%. Portanto, a escolha dessa técnica deve ser cuidadosamente considerada ao contexto da aplicação e dos objetivos do projeto. Em geral, esses resultados fornecem informações para o desenvolvimento de estratégias eficazes para a otimização de circuitos integrados, destacando o potencial das abordagens em cada cenário.

De forma geral, esses resultados fornecem dados para o desenvolvimento de estratégias eficazes de otimização de circuitos integrados, destacando o potencial de abordagem em cada cenário. Vale também destacar também que utilização de *software* de código aberto, como o OpenROAD, contribuiu para reduzir as barreiras de custo no campo de *Electronic Design Automation* , tornando este estudo acessível para a comunidade científica. Este trabalho oferece um ponto de partida para investigações futuras na área de otimização de circuitos integrados.

Dentre os trabalhos futuros, o fluxo desenvolvido neste trabalho permitirá avaliar também os circuitos quando requisitos de frequência máxima de operação forem configurados na síntese física, assim como resultados de síntese para circuitos exatos ou resultados de outras ferramentas de otimização lógica.

7.1 PUBLICAÇÕES

Durante o desenvolvimento deste trabalho, foram aplicadas diferentes estratégias com o intuito de conhecer e avaliar a técnica de otimização lógica baseada em CGP. Ao longo do processo, os resultados obtidos foram divulgados através de publicações e apresentações realizadas em congressos e *workshops*.

Em 2023, o *poster* "Area and Power Optimization in Approximate Circuit Synthesis based on Cartesian Genetic Programming"foi apresentado no 13º *Institute of Electrical and Electronics Engineers* (IEEE) CASS Rio Grande do Sul Workshop, tendo como enfoque a avaliação das métricas de área e potência em circuitos aproximados. Esses resultados também foram apresentados no 30º Iberchip Workshop, em fevereiro de 2024 em Punta del Leste, com o título "Otimização de Área e Potência na Síntese de Circuitos Aproximados baseada em Programação Genética Cartesiana".

Além disso, os resultados deste trabalho foram submetidos para o 37º *Symposium on Integrated Circuits and Systems Design* (SBCCI) em 2024, no artigo intitulado "Impact on Delay, Power and Area of Machine Learning-based Approximate Logic Synthesis"que está em etapa de avaliação.

REFERÊNCIAS

ABREU, Brunno A. de *et al.* Fast Logic Optimization Using Decision Trees. *In:* 2021 IEEE International Symposium on Circuits and Systems (ISCAS). [*S.l.*: *s.n.*], 2021. P. 1–5. DOI: 10.1109/ISCAS51556.2021.9401664.

AJAYI, Tutu; BLAAUW, David. Openroad: Toward a self-driving, open-source digital layout implementation tool chain. *In:* PROCEEDINGS of Government Microcircuit Applications and Critical Technology Conference. [*S.l.*: *s.n.*], 2019.

BEEREL, P. A.; PEDRAM, M. Opportunities for Machine Learning in Electronic Design Automation. *In:* 2018 IEEE International Symposium on Circuits and Systems (ISCAS). [*S.l.*: *s.n.*], 2018. P. 1–5.

BERGE, Claude. **The theory of graphs**. [*S.l.*]: Courier Corporation, 2001.

BERNDT, Augusto; ABREU, Brunno A. de *et al.* A CGP-based Logic Flow: Optimizing Accuracy and Size. **Journal of Integrated Circuits and Systems (JICS)**, 2022. DOI: 10.29292/jics.v17i1.546.

_____. Accuracy and Size Trade-off of a Cartesian Genetic Programming Flow for Logic Optimization. *In:* PROCEEDINGS of the 34th Symposium on Integrated Circuits and Systems Design. Brazil: [*s.n.*], 2021. (SBCCI '21). DOI: 10.1109/SBCCI53441.2021.9529968.

BERNDT, Augusto; CAMPOS, Isac S. *et al.* Accuracy and Size Trade-off of a Cartesian Genetic Programming Flow for Logic Optimization. *In:* 2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI). [*S.l.*: *s.n.*], 2021. P. 1–6. DOI: 10.1109/SBCCI53441.2021.9529968.

BERNDT, Augusto André Souza; ABREU, Brunno *et al.* A CGP-based Logic Flow: Optimizing Accuracy and Size of Approximate Circuits. **Journal of Integrated Circuits and Systems**, v. 17, n. 1, p. 1–12, 2022.

BRAYTON, Robert; MISHCHENKO, Alan. ABC: An academic industrial-strength verification tool. *In:* SPRINGER. INTERNATIONAL Conference on Computer Aided Verification. [*S.l.*: *s.n.*], 2010. P. 24–40.

BRAYTON, Robert K *et al.* MIS: A multiple-level logic optimization system. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE, v. 6, n. 6, p. 1062–1081, 1987.

BRAYTON, Robert K. *et al.* Logic Minimization Algorithms for VLSI Synthesis . **The Kluwer International Series in Engineering and Computer Science**, v. 2, p. 1–194, 1984. Disponível em: https://doi.org/10.1007/978-1-4613-2821-6.

CHAI, Donald *et al.* MVSIS 2.0 user's manual. **Department of Electrical Engineering and Computer Sciences**, 2003.

CHATTERJEE, S. *et al.* Reducing Structural Bias in Technology Mapping. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 25, n. 12, p. 2894–2903, dez. 2006. ISSN 1937-4151. DOI: 10.1109/TCAD.2006.882484.

CHATTERJEE, Satrajit. Learning and Memorization. *In:*_____. **Proceedings of the 35th International Conference on Machine Learning**. [*S.l.*]: PMLR, out. 2018. (Proceedings of Machine Learning Research), p. 755–763. Disponível em: https://proceedings.mlr.press/v80/chatterjee18a.html.

HÁLEČEK, Ivo; FIŠER, Petr; SCHMIDT, Jan. Towards AND/XOR balanced synthesis: Logic circuits rewriting with XOR. **Microelectronics Reliability**, v. 81, p. 274–286, 2018. ISSN 0026-2714. DOI: https://doi.org/10.1016/j.microrel.2017.12.031. Disponível em: https://www.sciencedirect.com/science/article/pii/S0026271417305899.

HANSEN, Nikolaus; ARNOLD, Dirk V; AUGER, Anne. Evolution strategies. *In:* SPRINGER handbook of computational intelligence. [*S.l.*]: Springer, 2015. P. 871–898.

HODAN, David; MRAZEK, Vojtech; VASICEK, Zdenek. Semantically-oriented mutation operator in cartesian genetic programming for evolutionary circuit design. **Genetic Programming and Evolvable Machines**, Springer, v. 22, n. 4, p. 539–572, 2021. Disponível em: https://doi.org/10.1007/s10710-021-09416-6.

IWLS 2020 PROGRAMMING CONTEST. **IWLS 2020 Benchmarks**. [*S.l.*: *s.n.*], 2020. https://github.com/iwls2020-lsml-contest/iwls2020-lsml-contest. Acesso em: 5 jan. 2021.

IWLS'20 ORGANIZING COMMITTEE. **29th International Workshop on Logic & Synthesis**. [*S.l.*: *s.n.*], 2020. http://www.iwls.org/iwls2020/. Acesso em: 10 jun. 2020.

JIANG, Honglan *et al.* Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. **Proceedings of the IEEE**, v. 108, n. 12, p. 2108–2135, 2020. DOI: 10.1109/JPROC.2020.3006451.

KAHNG, Andrew B *et al.* **VLSI physical design: from graph partitioning to timing closure**. [*S.l.*]: Springer Science & Business Media, 2011.

KARNAUGH, M. The map method for synthesis of combinational logic circuits. **Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics**, v. 72, n. 5, p. 593–599, 1953. DOI: 10.1109/TCE.1953.6371932. Disponível em: https://doi.org/10.1109/TCE.1953.6371932.

KHALILIAN, Alireza; BARAANI-DASTJERDI, Ahmad; ZAMANI, Bahman. CGenProg: Adaptation of cartesian genetic programming with migration and opposite guesses for automatic repair of software regression faults. **Expert Systems with Applications**, v. 169, p. 114503, 2021. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114503. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420311477.

KOCNOVA, Jitka; VASICEK, Zdenek. EA-based resynthesis: an efficient tool for optimization of digital circuits. **Genetic Programming and Evolvable Machines**, Springer, p. 1–33, 2020.

KOCNOVÁ, Jitka; VASICEK, Zdenek. Resynthesis of logic circuits using machine learning and reconvergent paths. *In:* 2021 24th Euromicro Conference on Digital System Design (DSD). [*S.l.*: *s.n.*], 2021. P. 69–76. DOI: 10.1109/DSD53832.2021.00020. Disponível em: https://doi.org/10.1109/DSD53832.2021.00020.

LIM, Sung Kyu. **Intelligent Design of Electronic Assets (IDEA)**. [*S.l.*]: DARPA RSS. Disponível em: https://www.darpa.mil/program/intelligent-design-of-electronic-assets.

MASADEH, Mahmoud; HASAN, Osman; TAHAR, Sofiene. Comparative Study of Approximate Multipliers. *In:* PROCEEDINGS of the 2018 on Great Lakes Symposium on VLSI. Chicago, IL, USA: Association for Computing Machinery, 2018. (GLSVLSI '18), p. 415–418. DOI: 10.1145/3194554.3194626. Disponível em: https://doi.org/10.1145/3194554.3194626.

MILANO, Nicola; PAGLIUCA, Paolo; NOLFI, Stefano. Robustness, evolvability and phenotypic complexity: insights from evolving digital circuits. **Evolutionary Intelligence**, Springer, v. 12, n. 1, p. 83–95, 2019.

MILLER, Julian F. Cartesian Genetic Programming. *In:* CARTESIAN Genetic Programming. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. P. 17–34. ISBN 978-3-642-17310-3. DOI: 10.1007/978-3-642-17310-3_2. Disponível em: https://doi.org/10.1007/978-3-642-17310-3_2.

MILLER, Julian Francis. Cartesian genetic programming: its status and future. **Genetic Programming and Evolvable Machines**, Springer, p. 1–40, 2019.

MISHCHENKO, A.; CHATTERJEE, S.; BRAYTON, R. DAG-aware AIG rewriting: a fresh look at combinational logic synthesis. *In:* 2006 43rd ACM/IEEE Design Automation Conference. [*S.l.*: *s.n.*], 2006. P. 532–535. DOI: 10.1145/1146909.1147048.

MISHCHENKO, Alan; CHATTERJEE, Satrajit; BRAYTON, Robert. DAG-aware AIG rewriting: A fresh look at combinational logic synthesis. *In:* IEEE. 2006 43rd ACM/IEEE Design Automation Conference. [*S.l.*: *s.n.*], 2006. P. 532–535.

MIYASAKA, Yukio *et al.* Logic Synthesis for Generalization and Learning Addition. *In:* 2021 Design, Automation Test in Europe Conference Exhibition (DATE). [*S.l.*: *s.n.*], 2021. P. 1032–1037. DOI: 10.23919/DATE51398.2021.9474169. Disponível em: https://doi.org/10.23919/DATE51398.2021.9474169.

MOORE, Gordon E *et al.* **Cramming more components onto integrated circuits**. [*S.l.*]: McGraw-Hill New York, 1965.

OPENROAD. **The OpenROAD Project GitHub**. [*S.l.*: *s.n.*], 2022. https://github.com/The-OpenROAD-Project/OpenROAD.

_____. **The OpenROAD Project Website**. [*S.l.*: *s.n.*], 2022. urlhttps://theopenroadproject.org/.

PANDEY, M. Machine learning and systems for building the next generation of EDA tools. *In:* 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). [*S.l.*: *s.n.*], 2018. P. 411–415.

QUINE, W. V. A Way to Simplify Truth Functions. **The American Mathematical Monthly**, Taylor & Francis, v. 62, n. 9, p. 627–631, 1955. Disponível em: https://doi.org/10.1080/00029890.1955.11988710.

RAI, Shubham; AL., et. Logic Synthesis Meets Machine Learning: Trading Exactness for Generalization. *In:* IEEE. 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). [*S.l.*: *s.n.*], 2021. DOI: 10.23919/DATE51398.2021.9473972. Disponível em: https://doi.org/10.23919/DATE51398.2021.9473972.

RAI, Shubham; NETO, Walter Lau *et al.* Logic Synthesis Meets Machine Learning: Trading Exactness for Generalization. *In:* 2021 Design, Automation Test in Europe Conference Exhibition (DATE). [*S.l.*: *s.n.*], 2021. P. 1026–1031. DOI: 10.23919/DATE51398.2021.9473972.

REIS, André Inácio; MATOS, Jody MA. Physical Awareness Starting at Technology-Independent Logic Synthesis. *In:* ADVANCED Logic Synthesis. [*S.l.*]: Springer, 2018. P. 69–101.

RIENER, Heinz *et al.* On-the-fly and DAG-aware: Rewriting Boolean Networks with Exact Synthesis. *In:* 2019 Design, Automation Test in Europe Conference Exhibition (DATE). [*S.l.*: *s.n.*], 2019. P. 1649–1654. DOI: 10.23919/DATE.2019.8715185.

RUDELL, R. L.; SANGIOVANNI-VINCENTELLI, A. Multiple-Valued Minimization for PLA Optimization. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 6, n. 5, p. 727–750, 1987. DOI: 10.1109/TCAD.1987.1270318. Disponível em: https://doi.org/10.1109/TCAD.1987.1270318.

RUPP, Karl. **Microprocessor Trend Data**. [*S.l.*: *s.n.*], 2022. urlhttps://github.com/karlrupp/microprocessor-trend-data.

SCARABOTTOLO, Ilaria *et al.* Approximate Logic Synthesis: A Survey. **Proceedings of the IEEE**, v. 108, n. 12, p. 2195–2213, 2020. DOI: 10.1109/JPROC.2020.3014430.

SENTOVICH, Ellen M *et al.* SIS: A system for sequential circuit synthesis. Citeseer, 1992.

SUGANUMA, Masanori *et al.* Evolution of Deep Convolutional Neural Networks Using Cartesian Genetic Programming. **Evolutionary Computation**, v. 28, n. 1, p. 141–163, mar. 2020. DOI: 10.1162/evco_a_00253. eprint: https://direct.mit.edu/evco/article-pdf/28/1/141/1858844/evco_a_00253.pdf. Disponível em: https://doi.org/10.1162/evco_a_00253.

TENACE, Valerio; CALIMERA, Andrea. Inferential Logic: a Machine Learning Inspired Paradigm for Combinational Circuits. *In:* 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC). [*S.l.*: *s.n.*], 2018. P. 149–154. DOI: 10.1109/VLSI-SoC.2018.8644808. Disponível em: https://doi.org/10.1109/VLSI-SoC.2018.8644808.

_____. Quasi-exact logic functions through classification trees. **Integration**, v. 63, p. 248–255, 2018. ISSN 0167-9260. DOI: https://doi.org/10.1016/j.vlsi.2018.06.007. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167926017307903.

VENKATARAMANI, Swagath *et al.* Approximate computing and the quest for computing efficiency. *In:* 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). [*S.l.*: *s.n.*], 2015. P. 1–6. DOI: 10.1145/2744769.2744904.

WOLF, Clifford; GLASER, Johann; KEPLER, Johannes. Yosys-a free Verilog synthesis suite. *In:* PROCEEDINGS of the 21st Austrian Workshop on Microelectronics (Austrochip). [*S.l.*: *s.n.*], 2013.

ZENG, Wei; DAVOODI, Azadeh; TOPALOGLU, Rasit Onur. Lorax: Machine Learning-Based Oracle Reconstruction With Minimal I/O Patterns. *In:* 2021 IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI). [*S.l.*: *s.n.*], 2021. P. 126–131. DOI: 10.1109/ISVLSI51109.2021.00033. Disponível em: https://doi.org/10.1109/ISVLSI51109.2021.00033.

Otimização de Área e Potência na Síntese de Circuitos Aproximados baseada em Programação Genética Cartesiana

João Carlos Prats Ramos, Naiara Sachetti, Augusto Berndt, Jonata T. Carvalho, Cristina Meinhardt
Departamento de Informática e Estatística, PPGCC, Universidade Federal de Santa Catarina - UFSC, Brazil
{joao.carlos.prats, naiara.sachetti}@grad.ufsc.br,augusto.berndt@posgrad.ufsc.br, {jonata.tyska, cristina.meinhardt}@ufsc.br

***Abstract*—A otimização lógica é um passo crucial no processo de síntese de circuitos. Este trabalho explora o uso de aprendizado de máquina, em particular, Programação Genética Cartesiana (CGP), para melhorar a eficiência energética na síntese de circuitos aproximados. Além disso, este trabalho apresenta uma análise comparativa dos resultados de síntese física para um conjunto de *benchmarks* de circuitos aproximados. O trabalho é realizado com ferramentas de código aberto, promovendo a acessibilidade e a colaboração na área de Design Eletrônico Automatizado (EDA). Comparado com o trabalho relacionado que apresentou a melhor solução de acurácia para este conjunto de *benchmark*, os resultados demonstram que a otimização lógica baseada em CGP reduziu área, potência e atraso crítico em mais de 50% em média. Contudo, essa melhora veio acompanhada de uma perda média de acurácia de cerca de 5% . Esses resultados demonstram o substancial potencial da abordagem CGP na otimização de circuitos aproximados, enfatizando a importância de avaliar seu uso na otimização lógica.**

***Index Terms*—Síntese lógica, Síntese Física, Programação Genética Cartesiana**

I. Introdução

O mundo contemporâneo tem a tecnologia como parte fundamental do seu funcionamento. O número de transistores por componentes cresceu de maneira acelerada com o passar das décadas, possibilitando um aumento de desempenho das tecnologias que os utilizam. Essa ampliação no númedaro deda transistoredas nos circuitos pedarmiteda o aumedanto da compledaxidadeda das funçõedas redaalizadas edam hardwareda [1]. Por um lado isso é positivo, pois possibilita a construção de sistemas mais avançados e a criação de tecnologias modernas. Por outro lado, torna seu processo de produção mais difícil e demorado, exigindo aprimoramentos na etapa de desenvolvimento que acompanhem a velocidade desses avanços [2].

O projeto de circuitos usualmente envolve gerar uma descrição fabricável do circuito partindo de uma descrição de alto nível, utilizando linguagens de descrição de *hardware*. Esse processo é realizado com ferramentas que realizam duas grandes etapas: (1) síntese lógica, que traduzem a descrição de alto nível para uma versão intermediária mais próxima dos dispositivos utilizados na fabricação (e.g., *Standard Cells*), chamada *netlist*; e (2) síntese física, que realiza posicionamento das células, geração de caminhos de roteamento e demais etapas necessárias para o circuito ser fabricável.

A otimização lógica, um dos passos da síntese lógica, é uma das etapas com potencial de proporcionar benefícios na produção e qualidade dos circuitos, gerando interesse da indústria pela sua melhoria. Esses procedimentos geram ganhos no desempenho com a redução do número de termos da função e profundidade lógica, impactando área, atraso de propagação, e consumo energético. Porém, esses algoritmos apresentam um elevado custo computacional, especialmente para o processamento de circuitos que excedam 10 *bits* de entrada.

Uma alternativa para o aprimoramento da síntese lógica é a aplicação de Aprendizado de Máquina. Recentemente, técnicas de Aprendizado de Máquina passaram a serem utilizadas na área de EDA (*Electronic Design Automation*), possuindo ainda bastante espaço para crescimento [2], [3]. Os métodos de Aprendizado de Máquina permitirem treinar modelos simplificados, sem um detalhamento completo, ao custo de uma taxa de acerto (acurácia) reduzida. Isso pode ser explorado para aplicação em outra área também em crescimento e bastante promissora: a computação aproximada. Algumas aplicações possuem certa tolerância a perda de qualidade sem comprometer seu funcionamento. Exemplos bastante conhecidos são as técnicas de compressão de vídeo, que aceita uma pequena taxa de erros na reprodução dos pixeis sem que isso seja perceptível pelo sistema visual humano, viabilizando a diminuição da qualidade em troca de um melhor desempenho. A computação aproximada é um paradigma emergente na área de EDA que tem sido discutido nos últimos anos [4] e pode ser explorado na etapa de síntese lógica [5], principalmente pensando no projeto visando eficiência energética.

Nos últimos anos, o compartilhamento de conhecimento entre desenvolvedores cresceu significativamente com as ferramentas de código aberto. A área científica pode tirar proveito dessas ferramentas e ajudar na criação de novas, ampliando o horizonte da tecnologia. Por isso, decidiu-se pela utilização apenas de *softwares* de código aberto no desenvolvimento deste projeto. Uma iniciativa interessante de código aberto para a síntese de circuitos é o *OpenRoad* [6], cujo principal objetivo é justamente diminuir as barreiras de custo, que são um dos principais obstáculos para iniciantes na área de EDA.

Este trabalho utiliza o fluxo de síntese aberto Openroad para avaliar os ganhos em área, atraso e potência de um

fluxo de otimização lógica baseado em programação genética cartesiana para síntese de circuitos aproximados. A adoção de CGP para otimização lógica demonstrou bons resultados na relação entre acurácia e número de nodos [7] quando comparado com soluções estado da arte [8] propostas para o mesmo conjunto de *benchmarks* [9]. Entretanto, até onde temos conhecimento neste momento, este é o primeiro trabalho que apresenta resultados de síntese física de circuitos gerados para este conjunto de *benchmarks* de funções aproximadas, discutindo a relação entre o número de nodos e profundidade lógica observados na etapa de síntese lógica e os impactos nas métricas de atraso, potência e área obtidos na síntese física. Além disso, o trabalho apresenta uma comparação com trabalhos relacionados de síntese lógica para o mesmo conjunto de benchmarks, projetados seguindo o mesmo fluxo de síntese física.

II. Fluxo de Otimização Lógica baseada em CGP

A Programação Genética Cartesiana é uma técnica de computação evolucionária [10]. Originalmente, utiliza grafos dirigidos acíclicos representados como uma grade de duas dimensões de nodos computacionais para representar programas [10]. Tipicamente, a cada geração, uma população de programas é avaliada para que se estabeleça a acurácia de cada um de seus indivíduos e, em seguida, uma nova população é formada pelo programa com maior acurácia (ou seja, que produz o maior número de saídas esperadas) e novos indivíduos gerados a partir de mutações (pequenas mudanças) de tal programa [10].

Na otimização lógica, os nodos representam portas lógicas. Para isso, os circuitos podem ser representados por AIGs (do inglês, *AND-Inverter Graphs*, Grafos de ANDs e Inversores), a estrutura de dados do estado-da-arte para otimizações independentes de tecnologia durante a síntese lógica [11]. AIG é um grafo dirigido e acíclico cujos nodos podem ter zero ou duas entradas, sendo que no primeiro caso se tem uma entrada primária e no segundo uma porta *AND*. Ainda, cada aresta pode ser complementada ou não, representando as portas inversoras. Para a identificação das saídas, os nodos podem ser marcados como saída primária [12].

Em [7], um fluxo de otimização lógica baseado em CGP é proposto, buscando melhorar a acurácia e o tamanho dos circuitos aproximados gerados. A minimização lógica com CGP mostrou-se efetiva quando considerados simultaneamente a acurácia e o número de nodos obtidos. O fluxo de síntese baseado em CGP é apresentado na Figura 1. O objetivo deste fluxo é otimizar tabelas de verdade completas e incompletas, além de circuitos previamente otimizados representados no formato *AND-inversor* (AIG). Ele oferece soluções de minimização lógica a partir de tabelas de verdade ou ajuste fino para blocos de circuitos, considerando cenários imprecisos, como na computação aproximada. O fluxo também trabalha com funções desconhecidas em blocos de propriedade intelectual ou funções de entrada maiores, onde a utilização de uma tabela de verdade completa pode ser inviável para métodos de minimização tradicionais.

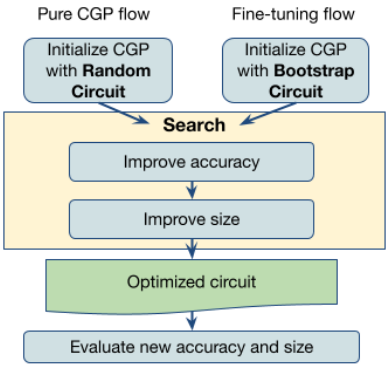


Fig. 1: Fluxo CGP [7]

O fluxo proposto pode ser iniciado de duas maneiras distintas: Com a inicialização aleatória, denominada *Pure CGP*, que realiza uma otimização baseada apenas em CGP a partir de uma tabela de verdade completa ou incompleta; ou com a inicialização *bootstrapped*, na qual o fluxo baseado em CGP inicia-se com um circuito previamente otimizado no formato AIG, permitindo um ajuste fino.

A primeira fase do fluxo de otimização lógica baseado em CGP é a de inicialização aleatória, que consiste na geração de uma população inicial aleatória de indivíduos, ou seja, de circuitos no formato AIG.

A região marcada em amarelo na Figura 1 engloba as etapas relacionadas à inicialização e otimização do CGP, realizando as evoluções e modificações necessárias em cada geração genética do algoritmo, até melhorar a acurácia. Após esta primeira etapa, o circuito é otimizado novamente com o algoritmo genético, entretanto, agora considerando como métrica de otimização o número de nodos do CGP. Tais circuitos passam por uma fase de evolução, repetida ao longo de diversas gerações. Nesta, a acurácia dos indivíduos é avaliada a partir de *mini-batches*, formados por uma parcela aleatoriamente extraída das linhas da tabela-verdade utilizada como conjunto de treinamento. A quantidade de linhas que compõem cada um dos *mini-batches* é dada pelo hiperparâmetro *batch size*, enquanto que a quantidade de gerações que um *mini-batch* deve permanecer inalterado é definida por *change each*. Escolhe-se então o circuito com maior acurácia para ser o pai da próxima geração de circuitos e os mesmos são gerados a partir de mutações de tal pai. Após o processo de evolução ser concluído, o circuito criado e otimizado pelo fluxo tem sua acurácia avaliada e um arquivo que contém a descrição do mesmo é gerado.

A saída do fluxo é um circuito otimizado que será avaliado quanto à métricas de síntese lógica, tais como: número de nodos, profundidade lógica e acurácia.

III. Metodologia

Este trabalho utilizou o fluxo baseado em CGP proposto por [7] e ilustrado na Figura 1 para a otimização lógica de circuitos aproximados. É relevante destacar que esse fluxo é capaz de utilizar qualquer circuito descrito no formato AIG como ponto

de partida, possibilitando a exploração de AIGs otimizados por diferentes técnicas e a aplicação do fluxo CGP como uma etapa adicional de otimização de ajuste fino. Os circuitos foram implementados utilizando a biblioteca de células de 45 nm da Nangate disponibilizada pelo *OpenROAD* [6], seguindo o fluxo apresentado na Figura 2. Dentre as ferramentas que compõem o fluxo de síntese do OpenRoad, podemos destacar as ferramentas ABC, Yosys e *OpenSTA*.

O ABC [13] é um software livre utilizado como o primeiro passo no fluxo de síntese do *OpenROAD*. Ele foi desenvolvido para realizar a síntese e verificação de circuitos lógicos sequenciais binários no contexto do desenvolvimento de hardware. O ABC combina técnicas de otimização lógica incluindo grafos AIG, mapeamento em tecnologia de grafos acíclicos dirigidos (DAG) para tabelas e *standard cells*, e algoritmos para síntese e verificação sequencial. Sua utilização no fluxo do *OpenROAD* contribui para a geração eficiente e precisa dos circuitos. Dentro do fluxo de síntese deste trabalho, funções foram empregadas para avaliar a acurácia de cada circuito com base na sua representação em AIG e para promover alterações de formato, fornecendo descrições RTL na linguagem Verilog para as descrições AIG avaliadas.

O Yosys [14] é um *framework* de código livre utilizado na síntese RTL em *Verilog*. Ele desempenha um papel fundamental na síntese lógica do *OpenROAD*, oferecendo um suporte abrangente ao *Verilog-2005* e fornecendo um conjunto básico de algoritmos de síntese adequados para diversas áreas de aplicação. Sua integração no fluxo do *OpenROAD* permite a realização eficiente da síntese lógica e a geração de resultados de alta qualidade. As ferramentas ABC e Yosys desempenham papéis cruciais no fluxo de síntese do *OpenROAD*, possibilitando a transformação de descrições de circuitos lógicos otimizados e prontos para a próxima etapa do processo de desenvolvimento de *hardware*, chamada de síntese física.

A síntese física é composta pelas etapas de *Floorplaning*, posicionamento, geração da rede de *clock* e otimização, roteamento e finalização do *layout* no formato a ser enviado para a fabricação. Após as etapas de síntese física, utilizamos a ferramenta *OpenSTA* para análise de *Timing* e potência. Os resultados obtidos após a finalização do *layout* foram comparados quanto ao tamanho do circuito gerado, a potência total, ao atraso crítico do circuito (*delay*) e acurácia.

A avaliação dos resultados considerou o conjunto de *benchmarks* da competição IWLS 2020 [15], composto de 100 funções lógicas incompletas. Além disso, os resultados da síntese baseada em CGP foram comparados com os resultados das melhores soluções da competição IWLS em acurácia [8]. Para automatizar o processamento dos circuitos, foi desenvolvido um *script* contendo sequências de comandos que visaram realizar o processo de extração das métricas dos diferentes arquivos de relatórios gerados no fluxo OpenRoad para os múltiplos circuitos.

IV. Resultados

A avaliação do fluxo CGP envolve a análise de várias métricas para determina sua eficácia. Como já mencionado, as

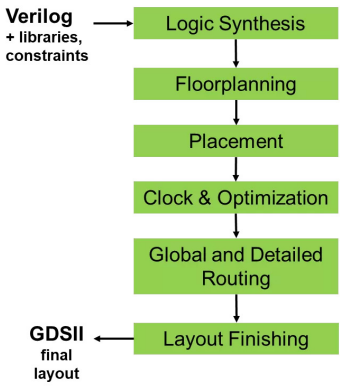


Fig. 2: Fluxo de síntese utilizado pelo *OpenROAD* [6]

métricas escolhidas para esta avaliação foram potência total, atraso crítico, área e acurácia, comparar o fluxo proposto baseado em CGP com a equipe com maior acurácia no IWLS 2020 é uma abordagem válida para avaliar seu desempenho.

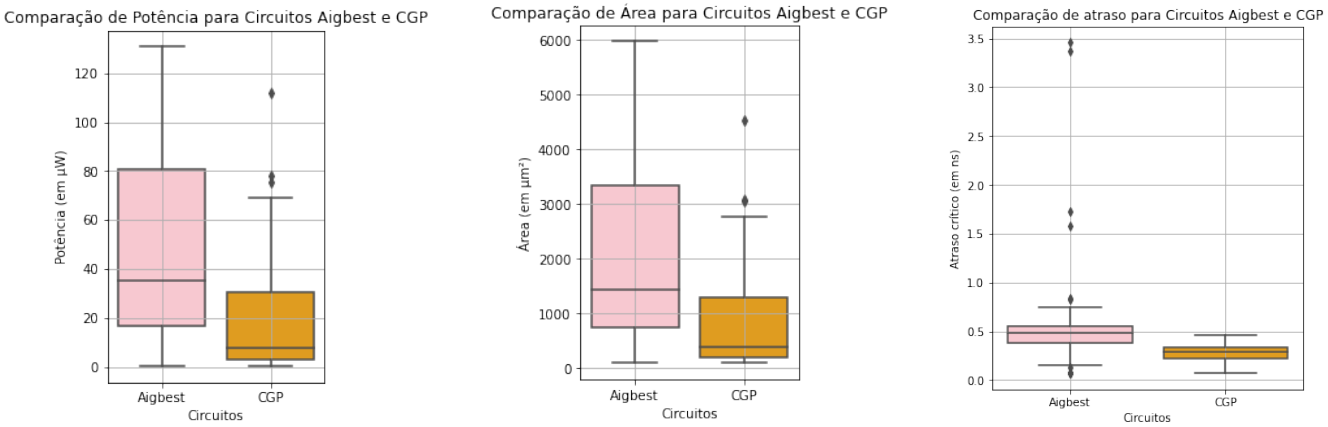
A apresentação dos boxplots das Figuras 3a, 3b e 3c demostram uma tendência de queda nas grandezas avaliadas na síntese baseada em CGP, esta redução em potência, atraso e área são vantajosas para projetistas de circuitos integrados. A representação visual nos permite, não apenas identificar as medianas, mas também compreender a dispersão dos dados, destacando valores discrepantes em cada abordagem avaliada.

Ao comparar as médias dos resultados de área, potência e *delay*, torna-se evidente que a abordagem CGP apresenta uma tendência de redução em comparação com a equipe de referência. A abordagem baseada em CGP revela reduções significativas, com uma redução de 58.89% na ocupação de área, uma diminuição de 61.79% no consumo de potência e uma queda de 50.60% no *delay*. Estes resultados indicam uma eficiência na utilização de recursos, economia de energia substancial e sugere um processamento mais rápido. Além disso, os resultados obtidos com CGP exibem uma menor variabilidade, sugerindo uma maior consistência em comparação com a equipe [8].

No entanto, como é mostrado na Figura 4 é importante destacar que, em média, a solução baseada em CGP apresenta uma diminuição de 5,80% na acurácia em relação à equipe [8]. Esse dado é fundamental a ser considerado na tomada de decisão, uma vez que a eficiência energética, economia de recursos e tempos de resposta devem ser ponderadas em relação à precisão do resultado final.

V. Conclusão

O fluxo baseado em CGP surge como uma perspectiva promissora no campo da otimização lógica. Neste estudo, apresentamos resultados que destacam a capacidade do fluxo CGP proposto para aprimorar o atraso crítico, eficiência energética e o tamanho de circuitos lógicos aproximados. Observamos uma redução de mais de 50% nessas métricas, demonstrando o potencial dessa abordagem.



(a) Resultados Potência (b) Resultados Área. (c) Resultados Atraso Crítico.

Fig. 3: Resultados de síntese física para a tecnologia de 45 nm comparando com o Aigbest [8] e os deste trabalho, com fluxo de otimização lógica baseado em CGP

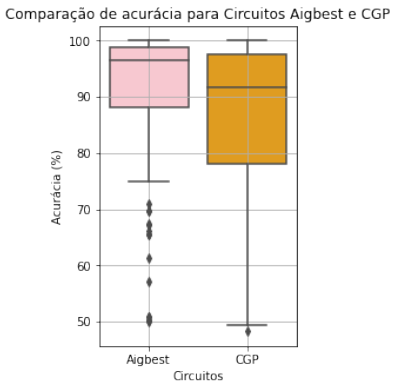


Fig. 4: Resultados Acurácia.

Entretanto, é fundamental observar que o sucesso do fluxo CGP está intrinsecamente relacionado ao contexto e aos objetivos específicos de cada projeto. É uma abordagem que prioriza a otimização, mas pode implicar em uma redução de cerca de 5% na acurácia, em média. Portanto, a escolha por essa técnica deve ser feita considerando cuidadosamente o contexto de aplicação e os objetivos do projeto.

O projeto apresentado oferece amplo potencial para exploração e aprimoramento contínuo. Uma abordagem válida para avaliar o desempenho do fluxo proposto baseado em CGP é compará-lo com a equipe que obteve a melhor solução em termos de área, considerando as métricas abordadas neste estudo. Isso proporcionará uma avaliação mais abrangente e ajudará a determinar o impacto dessa abordagem em relação a métricas cruciais.

VI .AGRADECIMENTOS

Este trabalho recebe recursos parciais dos orgãos de fomento CNPq, CAPES e PIBIC UFSC.

REFERENCES

[1] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[2] P. A. Beerel and M. Pedram. Opportunities for machine learning in electronic design automation. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

[3] M. Pandey. Machine learning and systems for building the next generation of eda tools. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 411–415, 2018.

[4] Ilaria Scarabottolo, Giovanni Ansaloni, George A. Constantinides, Laura Pozzi, and Sherief Reda. Approximate logic synthesis: A survey. *Proceedings of the IEEE*, 108(12):2195–2213, 2020.

[5] Swagath Venkataramani, Vivek J Kozhikkottu, Amit Sabne, Kaushik Roy, and Anand Raghunathan. Logic synthesis of approximate circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2503–2515, 2019.

[6] OpenROAD. The OpenROAD Project GitHub. https://github.com/The-OpenROAD-Project/OpenROAD, 2022.

[7] Augusto Berndt, Brunno A. de Abreu, Isac S. Campos, Bryan Lima, Mateus Grellert, Jonata T. Carvalho, and Cristina Meinhardt. A cgp-based logic flow: Optimizing accuracy and size. *Journal of Integrated Circuits and Systems (JICS)*, 2022.

[8] Yukio Miyasaka, Xinpei Zhang, Mingfei Yu, Qingyang Yi, and Masahiro Fujita. Logic Synthesis for Generalization and Learning Addition. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1032–1037, 2021.

[9] Shubham Rai and et al. Logic synthesis meets machine learning: Trading exactness for generalization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021.

[10] Julian F. Miller. *Cartesian Genetic Programming*, pages 17–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[11] Heinz Riener, Winston Haaswijk, Alan Mishchenko, Giovanni De Micheli, and Mathias Soeken. On-the-fly and dag-aware: Rewriting boolean networks with exact synthesis. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1649–1654, 2019.

[12] A. Mishchenko, S. Chatterjee, and R. Brayton. Dag-aware aig rewriting: a fresh look at combinational logic synthesis. In *2006 43rd ACM/IEEE Design Automation Conference*, pages 532–535, 2006.

[13] Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *International Conference on Computer Aided Verification*, pages 24–40. Springer, 2010.

[14] Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, 2013.

[15] IWLS 2020 programming contest. Iwls 2020 benchmarks. https://github.com/iwls2020-lsml-contest/iwls2020-lsml-contest, 2020.