Universidade Federal de Juiz de Fora

Programa de Pós-Graduação em Modelagem Computacional

Leandro Silva Lima

Uma Abordagem Multiobjetivo Usando Programação Genética Cartesiana Para Projeto de Circuitos Digitais Aproximados

Juiz de Fora

2019

Leandro Silva Lima

Uma Abordagem Multiobjetivo Usando Programação Genética Cartesiana Para Projeto de Circuitos Digitais Aproximados

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, na área de concentração em Modelagem Computacional , como requisito parcial para obtenção do título de Mestre em Modelagem Computacional.

Orientador: D.Sc. Heder Soares Bernardino

Coorientador: D.Sc. Helio José Corrêa Barbosa

Juiz de Fora

2019

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Lima, Leandro Silva.
Uma Abordagem Multiobjetivo Usando Programação Genética Cartesiana Para Projeto de Circuitos Digitais Aproximados / Leandro Silva Lima. – 2019.
90 f. : il.

Orientador: D.Sc. Heder Soares Bernardino
Coorientador: D.Sc. Helio José Corrêa Barbosa
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, . Programa de Pós-Graduação em Modelagem Computacional, 2019.

1. Programação Genética Cartesiana. 2.*Hardware* Evolutivo. 3. Circuitos Aproximados. I. Bernardino, Heder Soares, orient. II. Barbosa, Helio José Corrêa, coorient. III. Título.

Leandro Silva Lima

Uma Abordagem Multiobjetivo Usando Programação Genética Cartesiana Para Projeto de Circuitos Digitais Aproximados

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal de Juiz de Fora, na área de concentração em Modelagem Computacional , como requisito parcial para obtenção do título de Mestre em Modelagem Computacional.

Aprovada em: 22 de Fevereiro de 2019

BANCA EXAMINADORA

Professor D.Sc. Heder Soares Bernardino - Orientador
Universidade Federal de Juiz de Fora

Professor D.Sc. Helio José Corrêa Barbosa - Coorientador
Universidade Federal de Juiz de Fora

Professor D.Sc. Eduardo Pestana de Aguiar
Universidade Federal de Juiz de Fora

Professor D.Sc. Francisco Augusto Lima Manfrini
Instituto Federal do Sudeste de Minas Gerais - Campos de Juiz de Fora

AGRADECIMENTOS

A minha mãe, sem seu apoio nada teria acontecido.

Aos meus orientadores Prof. Heder e Prof. Helio, pela orientação, confiança, competência e paciência.

Aos meu amigos de turma, tivemos momentos bons e que jamais serão esquecidos.

Aos professores do Programa de Pós-Graduação em Modelagem Computacional pelos seus ensinamentos e profissionalismo.

Aos funcionários do Programa de Pós-Graduação em Modelagem Computacional que sempre estiveram disponíveis em caso de necessidade.

Agradeço a Universidade Federal de Juiz de Fora.

RESUMO

Os motivos que justificam estudos envolvendo *Hardware* Evolutivo (EHW), uma área voltada ao projeto de dispositivos eletrônicos que utiliza algoritmos evolutivos na concepção de determinada arquitetura como circuitos digitais e circuitos analógicos, são: (i) desenvolver modelos que não são alcançáveis quando técnicas tradicionais são empregadas; e (ii) projetar boas soluções para problemas e aplicações em que as especificações são incompletas ou não existem soluções ótimas. Neste cenário, a Computação Aproximada aparece como um novo paradigma em resposta à necessidade de melhorar o desempenho ou a eficiência energética de dispositivos eletrônicos. Na Computação Aproximada, a acurácia dos modelos pode ser relaxada, gerando projetos para usuários que estão dispostos a aceitar certas imprecisões, como em aplicações multimídia, circuitos aritméticos imprecisos, processos de compressão de imagens, aplicações de filtros digitais de resposta finita ao impulso (FIR) e resposta infinita ao impulso (IIR). Circuitos digitais obtidos via Computação Aproximada são classificados como circuitos digitais aproximados. Os requisitos de funcionalidade dos circuitos digitais aproximados são relaxados visando alcançar: (i) economia de energia, (ii) melhor velocidade de resposta, (iii) menor complexidade do circuito e (iv) menor área ocupada pelos componentes do circuito. Deseja-se investigar aqui o processo de construção de circuitos digitais combinacionais aproximados via Computação Evolucionista multiobjetivo. Para tal uma técnica de Programação Genética Cartesiana, baseada no conceito de dominância de Pareto e com tamanho populacional adaptativo para lidar com múltiplos objetivos é proposta neste trabalho visando projetar circuitos digitais aproximados analisando o compromisso entre atraso de propagação (*delay*), potência consumida e erro. Tal técnica é intitulada CGPMO+TPA. Circuitos digitais combinacionais como somadores, multiplicadores e Unidades Lógicas Aritméticas com até 16 entradas e 370 portas lógicas são considerados nos experimentos computacionais. O método CGPMO+TPA foi comparado com abordagens presentes na literatura e apresentou resultados satisfatórios.

Palavras-chave: Programação Genética Cartesiana. Computação Evolucionista. *Hardware* Evolutivo. Circuitos Aproximados.

ABSTRACT

The reasons that justify studies involving *Hardware* Evolutionary (EHW), an area dedicated to the design of electronic devices that uses evolutionary algorithms in the design of a certain architecture like digital circuits and analog circuits, are: (i) developing models that are not attainable when traditional techniques are employed; and (ii) designing good solutions to problems and applications where specifications are incomplete or there are no optimal solutions. In this scenario, Approximate Computing appears as a new paradigm in response to the need to improve the performance or energy efficiency of electronic devices. In Approximate Computing, the accuracy of the models can be relaxed, generating designs for users who are willing to accept certain inaccuracies, such as applications imprecise arithmetic circuits, image compression processes, digital finite impulse response (FIR) and infinite impulse response (IIR) applications. Digital circuits obtained through Approximate Computing are classified as circuits. The functional requirements of the approximate digital circuits are relaxed in order to achieve: (i) energy savings, (ii) better response speed, (iii) less complexity of the circuit and (iv) smaller area occupied by the circuit components. We wish to investigate here the process of construction of approximate combinational digital circuits via Multiobjective Evolutionary Computation. For such a technique of Genetic Programming Cartesiana, based on the concept of Pareto dominance and with population size adaptive to deal with multiple objectives is proposed in this work to design approximate digital circuits by analyzing the compromise between propagation delay, power consumed and error. Such a technique is entitled CGPMO + TPA. Combinational digital circuits such as adders, multipliers, and Arithmetic Logic Units with up to 16 inputs and 370 logic gates are considered in computational experiments. The CGPMO + TPA method was compared with approaches in the literature and presented satisfactory results.

Keywords: Cartesian Genetic Programming. Evolutionary Computation. Evolutionary Hardware. Approximate Circuits.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação de um sinal analógico. 19
Figura 2 – Representação de um sinal digital. 20
Figura 3 – Intervalo de tensão para binários 0 e 1. Extraído de (TOCCI; WIDMER; MOSS, 2011) 21
Figura 4 – Implementação hierárquica. Disponível em (MANFRINI, 2017). 22
Figura 5 – Exemplo de uma tabela verdade, que descreve a relação entrada/saída de um circuito lógico. Disponível em (TOCCI; WIDMER; MOSS, 2011). 23
Figura 6 – Associação de operadores booleanos. Extraído de (MANFRINI, 2017). 24
Figura 7 – Metodologia tradicional × Metodologia Evolutiva. Extraído de (MANFRINI, 2017)). 25
Figura 8 – Circuito exemplo para cálculo do fator transição de nós. 30
Figura 9 – Circuito hipotético apresentado aqui para ilustrar o cálculo do seu *delay*. 31
Figura 10 – Caminho 1. 32
Figura 11 – Caminho 2. 32
Figura 12 – Ilustração da representação das soluções candidatas na CGP. Adaptado de (HRBACEK; MRAZEK; VASICEK, 2016). 34
Figura 13 – Codificação de um circuito digital candidato por meio da abordagem CGP. O vetor V = [{**1, 3**,$\underline{0}$}, {**0, 2**,$\underline{0}$}, {**1, 2**,$\underline{0}$}, {**0, 1**,$\underline{0}$}, {**7, 6**,$\underline{1}$}, {**8, 8**,$\underline{1}$}, {*5, 8, 4, 0*}] é o cromossomo correspondente à solução candidata. Os alelos em negrito representam os índices de entradas dos nós e os alelos cujos valores estão sublinhados representam as respectivas operações lógicas. Já os genes cujos alelos estão em itálico, representam os índices de conexão das n_o saídas primárias. Extraído de (VASICEK; SEKANINA, 2015b). 35
Figura 14 – Genótipo e fenótipo antes da mutação. 37
Figura 15 – Genótipo e fenótipo depois da mutação. 37
Figura 16 – Mapeamento entre os espaços de decisão e objetivos. Extraído de (RUSSO, 2017). 42
Figura 17 – Convergência e diversidade ilustradas num problema com dois objetivos, f_1 e f_2. Adaptado de (RUSSO, 2017). 44
Figura 18 – Ordenação de Soluções em Frentes de Pareto. F_1 é a primeira frente de não dominância, F_1 é a segunda frente de não dominância e F_3 é a terceira frente de não dominância. Adaptado de (RUSSO, 2017). 45
Figura 19 – Exemplo de cálculo do hipervolume para um conjunto de 5 soluções não dominadas em um problema de otimização com dois objetivos conflitantes. A é o ponto de referência. Figura baseada em (DEB, 2001; ZITZLER; DEB; THIELE, 2000). 50
Figura 20 – Variação do tamanho da população ao longo das gerações. 52

Figura 21 – Projeções 2D da dissipação de potência, *delay* e erro das AULs evoluídas quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. . 63
Figura 22 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 65
Figura 23 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 67
Figura 24 – Projeções 2D da dissipação de potência, *delay* e erro das ALUs evoluídas quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. . 70
Figura 25 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 71
Figura 26 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 73

Figura 27 – Projeções 2D da dissipação de potência, *delay* e erro das AULs evoluídas quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 76
Figura 28 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 78
Figura 29 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II. 80

LISTA DE TABELAS

Tabela 1 – Probabilidades de sinal e fator transição de nós. 31
Tabela 2 – Valores hipotéticos de atrasos t_d para cada uma das portas lógicas. . . 31
Tabela 3 – Tabela de funcionalidades em que $A + B$ corresponde a "A ou B" lógico. $A.B$ corresponde a "A e B" lógico. A soma é indicada por "mais", ao passo que a subtração é indicada por "menos". 56
Tabela 4 – Configuração do modelo de representação CGP utilizado por Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010). 57
Tabela 5 – Elementos de processamento lógico utilizado por Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010). 58
Tabela 6 – Número de execuções independentes, que resultaram em circuitos totalmente funcionais, dos vários algoritmos para o somador de 2 bits (S2) e multiplicador de 2 bits (M2). hn^{10} diz respeito à periodização do método hES como o NSGA-II, neste caso é realizado um passo/geração da abordagem hES e 10 passos/gerações do NSGA-II. 59
Tabela 7 – Configuração do modelo de representação CGP utilizados pelos métodos CGPMO+TPA e CGP+NSGA-II. 60
Tabela 8 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW. 62
Tabela 9 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW. 64
Tabela 10 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*.Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW. 66
Tabela 11 – Pontos de referência utilizados no cálculo de hipervolume. 67
Tabela 12 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos. 68
Tabela 13 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%. 68

Tabela 14 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW. 69
Tabela 15 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW. 71
Tabela 16 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*. Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW. 72
Tabela 17 – Pontos de referência utilizados no cálculo de hipervolume. 73
Tabela 18 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos. 74
Tabela 19 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%. 74
Tabela 20 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW. 77
Tabela 21 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW. 78
Tabela 22 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*. Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW. 79
Tabela 23 – Pontos de referência utilizados no cálculo de hipervolume. 80

Tabela 24 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos. 81
Tabela 25 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%. 81

SUMÁRIO

1	**INTRODUÇÃO**	**14**
1.1	OBJETIVOS	17
1.2	ORGANIZAÇÃO DO TEXTO	18
2	**CIRCUITOS LÓGICOS COMBINACIONAIS**	**19**
2.1	SISTEMAS ANALÓGICOS	19
2.2	SISTEMAS DIGITAIS	19
2.2.1	**Vantagens dos Sistemas Digitais**	20
2.2.2	**Circuitos Digitais**	20
2.2.3	**Desenvolvimento de Sistemas Digitais**	21
2.2.4	**Circuitos Lógicos**	21
2.3	*HARDWARE* EVOLUTIVO	24
2.3.1	**Funções Objetivo de Projeto de Circuitos Digitais**	27
2.3.1.1	*Estimativa de Área Ocupada*	27
2.3.2	*Estimativa de Erro*	28
2.3.2.1	*Estimativa de Dissipação de Potência*	29
2.3.2.2	*Estimativa de Atraso de Propagação (Delay)*	31
3	**PROGRAMAÇÃO GENÉTICA CARTESIANA**	**33**
3.1	REPRESENTAÇÃO DE SOLUÇÕES	33
3.2	MUTAÇÃO NA CGP	36
3.3	RECOMBINAÇÃO NA CGP	36
3.4	ALGORITMO EVOLUTIVO	37
4	**CIRCUITOS APROXIMADOS E OTIMIZAÇÃO MULTIOBJETIVO**	**40**
4.1	FORMULAÇÃO DE OTIMIZAÇÃO MULTIOBJETIVO	41
4.2	*NON-DOMINATED SORTING GENETIC ALGORITHM II*	44
4.2.1	**Ordenação por não dominância**	46
4.2.2	***Crowding-distance***	48
4.2.3	***Crowded-Comparison Operator***	49
4.3	HIPERVOLUME	49
5	**MÉTODO PROPOSTO**	**51**
5.1	DESCRIÇÃO GERAL DO MÉTODO CGPMO+TPA	52
5.2	TRATAMENTO DAS RESTRIÇÕES	53

6 EXPERIMENTOS COMPUTACIONAIS 55
6.1 DESCRIÇÃO DOS PROBLEMAS 55
6.2 COMPARAÇÃO ENTRE hES e CGPMO+TPA 57
6.3 PROJETO DE CIRCUITOS DIGITAIS COMBINACIONAIS APROXIMADOS 59
6.3.1 **Inicialização da população de forma aleatória** 61
6.3.1.1 *Projeto de circuitos aproximados quando* 30×10^6 *avaliações de circuito foram permitidas* 61
6.3.1.2 *Projeto de circuitos aproximados quando* 60×10^6 *avaliações de circuito foram permitidas* 68
6.3.2 **Inicialização da população com arquiteturas convencionais** . . . 75

7 CONCLUSÕES E TRABALHOS FUTUROS 82

REFERÊNCIAS 84

1 INTRODUÇÃO

Em meados dos anos 1900, surgiu no Bell Laboratories uma das mais importantes invenções do milênio, o transistor (SEDRA; SMITH, 2004). Tal componente foi criado por William Shockley, John Bardeen e Walter Brattian. O crescimento do volume de produção, complexidade e avanço da indústria do transistor permitiu desenvolver sistemas eletrônicos com grande capacidade computacional em dimensões ínfimas, alavancando o potencial da eletrônica e da computação. Neste cenário, a disseminação dos dispositivos eletrônicos foi tão significativa que estes dispositivos estão totalmente integrados ao nosso cotidiano, visto que computadores pessoais, *smartfones*, *smart* TVs, eletrodomésticos, dispositivos de comunicação e muitos outros equipamentos eletrônicos só foram possíveis graças ao advento do transistor. Além de impactar o nosso dia a dia, o surgimento do transistor impulsionou a economia dos Estados Unidos da América, em razão da ascensão do Vale do Silício, uma região do Estado da Califórnia que abriga diversas companhias de tecnologia como a Apple Inc, Intel, Nvidia, National Semiconductor, Maxim Integrated Products e LSI Logic (RODRIGUES, 2010).

Em relação à síntese de dispositivos eletrônicos, é exigido do projetista grande conhecimento de uma vasta coleção de regras específicas. Neste sentido, a utilização da Computação Evolutiva como ferramenta para projetar circuitos eletrônicos pode gerar projetos competitivos e algumas vezes superiores aos projetos elaborados por especialistas (VASICEK; SEKANINA, 2011). Entretanto, o projeto de circuitos eletrônicos via Computação Evolucionista apresenta algumas dificuldades tais como:

- Representação de soluções, já que geralmente são necessários cromossomos longos para codificar soluções complexas. Cromossomos longos, no entanto, implicam em grandes espaços de busca que são tipicamente difíceis de pesquisar, gerando prejuízos para o projeto evolutivo (STEPNEY; ADAMATZKY, 2018).
- Tempo de cálculo da função de avaliação, uma vez que o tempo de processamento cresce exponencialmente com o número de entradas do circuito (MILLER, 2011) e o tempo de projeto é determinante na aplicabilidade de um método de concepção (VASICEK; SEKANINA, 2015a).

Para alcançar resultados que tragam inovação para o projeto de circuitos eletrônicos via Computação Evolucionista é importante o desenvolvimento de estratégias visando consolidar e disseminar as oportunidades que o projeto evolutivo oferece. Neste sentido, a Programação Genética Cartesiana (CGP) é definida como uma técnica de Programação Genética em que as implementações são modeladas como grafos acíclicos direcionados que têm os nós organizados em uma matriz (MILLER et al., 1997). A CGP, inicialmente desenvolvida para representar circuitos digitais, pode, também, representar diversas estruturas

como filtros digitais (MILLER, 1999b), redes neurais (KHAN; MILLER; HALLIDAY, 2011) e estruturas de processamento de imagem (JULIO et al., 2015; HUANG; KUMAR; ABBAS, 2018; KULKARNI; GUPTA; ERCEGOVAC, 2011).

Destaca-se que uma grande parcela do genótipo de indivíduos da CGP não é mapeada para o fenótipo. Contudo, durante o projeto evolutivo, esses genes (chamados de inativos por não afetarem o fenótipo) podem se tornar ativos e, assim, passam a influenciar o processo de busca. Com isso, novas (e potencialmente melhores) soluções podem ser exploradas. Isso torna a CGP uma técnica robusta e poderosa para escapar de ótimos locais (TURNER; MILLER, 2015; MANFRINI, 2017).

Por meio da CGP é possível codificar e representar de forma adequada problemas que apresentam diversas entradas e saídas, além disso, a CGP não permite um crescimento desordenado de certa implementação cujas soluções se tornam cada vez maiores sem que ocorra melhora na qualidade das soluções (GOLDMAN; PUNCH, 2015; MANFRINI, 2017). Consequentemente, a CGP tornou-se o método mais utilizado no projeto evolutivo de circuitos digitais combinacionais que buscam atender a todas as restrições impostas pela tabela verdade (VASICEK; SEKANINA, 2015b) e em projetos de otimização evolutiva que têm como principal característica a inserção de um ou mais indivíduos totalmente funcionais na população inicial (VASICEK, 2015; VASICEK; SEKANINA, 2015a; VASICEK; SEKANINA, 2014b; VASICEK; SEKANINA, 2012). Além disso, a CGP é amplamente adotada no projeto de circuitos digitais aproximados (BABU; BALAJI, 2016).

Circuitos digitais aproximados eram inicialmente construídos removendo partes dos circuitos digitais totalmente funcionais que não contribuíam significativamente para o resultado entregado pela arquitetura factível (VASICEK; SEKANINA, 2015b). A medida que a complexidade dos sistemas digitais cresceu, a abordagem manual usada para a construção de circuitos digitais aproximados tornou-se obsoleta e ineficiente, consequentemente métodos automatizados de construção de circuitos aproximados passaram a ser empregados. Tais métodos levam a uma solução aproximada restringida por um critério predefinido (SEKANINA; VASICEK, 2013). A tendência é criar metodologias sistemáticas para projetar circuitos digitais aproximados em vez de adotar o modo manual (VASICEK; SEKANINA, 2014a; VASICEK; SEKANINA, 2015b). As metodologias automatizadas *Systematic methodology for Automatic Logic Synthesis of Approximate Circuits* (SALSA) (VENKATARAMANI et al., 2012) e *Substitute-And-Simplify* (SASIMI) (VENKATARAMANI; ROY; RAGHUNATHAN, 2013) são frequentemente adotadas para o projeto de circuitos digitais aproximados.

A metodologia SALSA é iniciada com uma descrição exata do circuito e usa uma função de qualidade, em termos de erro, *delay*, área ocupada ou potência dissipada, para verificar se uma restrição predefinida é atendida ou não. Essa metodologia tem sido usada como ferramenta no projeto de somadores, multiplicadores, filtros digitais de

resposta finita ao impulso (FIR) e resposta infinita ao impulso (IIR), além de Blocos de Transformada Discreta de Cosseno (DCT) (BABU; BALAJI, 2016). Já a metodologia SASIMI tenta identificar pares de sinais com alta probabilidade de apresentarem valores iguais. Quando esses sinais são encontrados ocorre a substituição de um sinal pelo outro e uma parte redundante do circuito (que exibe os mesmos valores de sinal) pode ser removida, resultando em menor dissipação de potência, *delay* e área ocupada. Além disso, a abordagem SASIMI pode ser combinada com outras abordagens de otimização, como *downsizing* de portas (criação de portas menores que o normal para reduzir a dissipação de energia, em troca de atrasos maiores) em caminhos críticos e também a técnica de *voltage over-scaling*, resultando em economia de energia e redução da área ocupada pelos componentes digitais (VASICEK; SEKANINA, 2015b).

Um algoritmo evolucionista deve lidar com múltiplos objetivos no projeto evolutivo de circuitos digitais aproximados. Neste cenário, o que geralmente se encontra na literatura são abordagens evolutivas sem restrições e que adotam os três objetivos (a serem minimizados) que seguem: (i) uma estimativa de erro para avaliar a funcionalidade de um circuito digital, (ii) dissipação de potência, e (iii) *delay*. Observa-se que quanto melhor é o funcionamento do circuito em termos de erro, maior será o consumo de energia, *delay* e área ocupada pelos componentes digitais (HRBACEK; MRAZEK; VASICEK, 2016).

Sekanina et al. (SEKANINA; VASICEK, 2013) propôs um método baseado na CGP que aborda o problema da síntese de circuitos digitais aproximados por meio da construção de um conjunto de circuitos que tentam aproximar a funcionalidade alvo utilizando recursos restritos (quantidade de portas lógicas disponíveis). Estes circuitos foram analisados em termos de violação de funcionalidade (erro absoluto médio). O circuito que apresentou o melhor equilíbrio entre os recursos consumidos e o erro medido foi considerado como solução final. Compensações entre funcionalidade e área (número de portas lógicas utilizadas) foram consideradas no processo evolutivo. Os autores não consideraram a dissipação de potência como objetivo durante o processo evolutivo, pois argumentaram que seria extremamente demorado. No entanto, a dissipação de potência dos circuitos obtidos no final do processo evolutivo foi calculada pela ferramenta SIS (SENTOVICH et al., 1992).

O método proposto por Sekanina et al. (SEKANINA; VASICEK, 2013) começa com a tarefa de projetar um circuito alvo A visando descobrir o número mínimo de portas (n) necessárias para sua implementação. Então, aproximações de A são criadas. O circuito aproximado A_1 é projetado por meio da CGP podendo consumir até $n-1$ portas. Neste contexto, como o objetivo da CGP é minimizar a função de erro, a evolução é interrompida quando a condição de parada é satisfeita (determinado erro absoluto médio). A primeira aproximação do circuito alvo (circuito A_1) exibe erro e_1. Outras versões aproximadas de A ($A_2, \ldots, A_k$) que têm erro $e_2, \ldots, e_k$ são projetadas por meio da CGP quando $n-2, \ldots, n-k$ portas são fornecidas. No final, o projetista pode escolher a

solução que exibe o melhor compromisso entre a funcionalidade e o número de portas.

Vasicek et al.(VASICEK; SEKANINA, 2015b) desenvolveu um método sistemático baseado na CGP para projetar circuitos digitais aproximados que explora a abordagem de projeto orientada por área. Essa abordagem permite que o usuário controle os recursos (elementos de processamento lógico) utilizados durante o processo evolutivo. A abordagem orientada por área é útil em situações em que existe restrição de área que um circuito digital pode ocupar, como quando um filtro digital precisa ser aproximado, pois a arquitetura convencional não pode ser implementada no espaço disponível em um chip. O objetivo do método é projetar circuitos digitais que exibem o menor erro possível quando os recursos são limitados, neste contexto em razão da estratégia proposta ser baseada na CGP foi possível obter mais compensações entre velocidade de computação, área ocupada em um chip e dissipação de potência do que os métodos baseados em modificações manuais. Novas implementações de circuitos digitais aproximados como multiplicadores, somadores e circuitos computacionais medianos foram obtidas, reforçando a ideia de que métodos baseados na CGP são adequados para projeto de circuitos digitais aproximados.

Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) investigou o impacto gerado ao se periodizar uma estratégia de busca local baseada na CGP com algoritmos de busca multiobjetivo na evolução de somadores e multiplicadores de 2 bits. Neste cenário, compensações entre área ocupada e estimativa de atraso de propagação foram analisadas quando uma métrica de erro foi adotada como restrição. Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016) propôs uma abordagem evolutiva multiobjetivo para projetar circuitos digitais aproximados que utiliza a Programação Genética Cartesiana para representar circuitos candidatos e o algoritmo *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) (DEB et al., 2002) para lidar com os múltiplos objetivos. Neste contexto, versões aproximadas de multiplicadores e somadores de 8 bits foram projetadas com economias significativas de dissipação de potência e *delay*. A contribuição deste trabalho é o desenvolvimento de uma técnica de Programação Genética Cartesiana multiobjetivo voltada para o projeto de circuitos digitais aproximados.

1.1 OBJETIVOS

O objetivo deste trabalho é propor uma estratégia de construção de circuitos digitais combinacionais aproximados via Computação Evolucionista multiobjetivo. Para isso busca-se:

- Desenvolver uma técnica de Programação Genética Cartesiana multiobjetivo para o projeto de circuitos digitais combinacionais aproximados;
- Propor uma estratégia de adaptação do tamanho da população, que leva em consideração conceitos de dominância de Pareto e *crowding distance* (DEB et al., 2002);

item Avaliar a técnica desenvolvida em três cenários: (i) construção de circuitos digitais sem conhecimento prévio e considerando restrições de projeto, (ii) iniciando a construção de circuitos digitais aproximados sem conhecimento prévio, e (iii) considerando uma arquitetura convencional como solução inicial;

- Projetar circuitos digitais aproximados levando-se em conta compensações entre atraso (*delay*), potência dissipada e funcionalmente em termos de erro;
- Projetar circuitos digitais aproximados com até 16 entradas e 370 portas lógicas, que podem ser considerados grandes, visto que a literatura envolvendo o projeto de circuitos a nível de portas lógicas via Programação Genética Cartesiana geralmente trata, por exemplo, de multiplicadores e somadores de 2, 3 e 4 bits, isto é, circuitos com no máximo 8 entradas e 72 portas lógicas (MILLER, 2011);
- Validar o algoritmo desenvolvido, comparando-o com resultados apresentados pela literatura.

1.2 ORGANIZAÇÃO DO TEXTO

O restante deste trabalho está dividido da seguinte forma. O Capítulo 2 apresenta conceitos relacionados à circuitos lógicos combinacionais. O Capítulo 3 apresenta conceitos relacionados à Programação Genética Cartesiana. O Capítulo 4 introduz fundamentos de circuitos digitais aproximados e otimização multiobjetivo. O Capítulo 5 descreve o método proposto. O Capítulo 6 apresenta os experimentos computacionais que validaram a metodologia proposta. O Capítulo 7 apresenta as conclusões e possíveis trabalhos futuros.

2 CIRCUITOS LÓGICOS COMBINACIONAIS

Este capítulo trata de assuntos relacionados aos circuitos lógicos combinacionais. A Seção 2.1 apresenta uma visão a respeito de sistemas analógicos, ao passo que a Seção 2.2 discute a respeito de sistemas digitais, suas vantagens quando comparados à sistemas analógicos e circuitos digitais em geral. A Seção 2.3 trata do projeto evolutivo de circuitos digitais, apresentando vantagens e desafios do projeto via Computação Evolucionista, além das funções objetivo mais utilizadas no contexto do projeto evolutivo de circuitos digitais.

2.1 SISTEMAS ANALÓGICOS

Em sistemas analógicos, as quantidades são representadas por medidas de grandezas físicas como graus de rotação de um disco e a intensidade da corrente e tensão elétrica (TOCCI; WIDMER; MOSS, 2011). Cita-se como exemplo de sistemas analógicos microfones de áudio, velocímetro de automóveis e termômetros. Em relação ao microfone, a tensão de saída gerada é proporcional à amplitude das ondas sonoras que o atingem. Como consequência, as variações nos valores de tensão oscilam de acordo com as variações das ondas sonoras na entrada.

Grandezas representadas de forma analógica, como na Figura 1, são sinais contínuos, ou seja, variam ao longo de um intervalo contínuo de valores. A velocidade de um automóvel pode assumir qualquer valor pertencente ao intervalo de $0km/h$ até $180km/h$, dependendo do modelo; a temperatura registrada por um termômetro pode variar entre todos os valores de um hipotético intervalo de valores de temperatura e a tensão de saída de um microfone pode estar em qualquer ponto do intervalo de 0 a $10mV$ (TOCCI; WIDMER; MOSS, 2011).

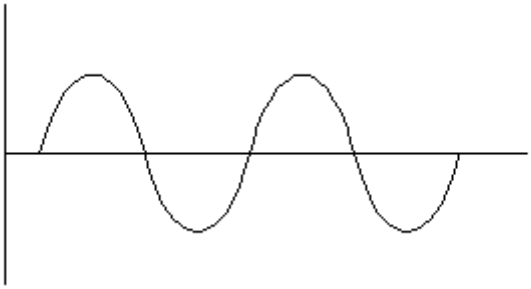

Figura 1 – Representação de um sinal analógico.

2.2 SISTEMAS DIGITAIS

Nos sistemas digitais, a representação se dá por símbolos denominados dígitos (TOCCI; WIDMER; MOSS, 2011). Cita-se como exemplo desse tipo de sistema o relógio digital, que mostra as horas e minutos do dia em forma de dígitos decimais.

O tempo varia de modo contínuo, mas em um relógio digital o tempo varia em saltos ou degraus de segundos e/ou minutos. Logo nos sistemas digitais, como na Figura 2, a representação das grandezas varia em passos discretos.

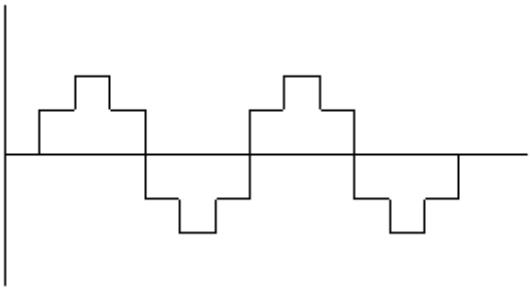

Figura 2 – Representação de um sinal digital.

2.2.1 **Vantagens dos Sistemas Digitais**

A maioria dos projetos e aplicações da eletrônica, assim como em diversas outras áreas como controle e automação de processos, utiliza técnicas e dispositivos digitais para realizar operações que anteriormente eram realizadas por meio de metodologias baseadas na eletrônica analógica. Dentre os benefícios que justificam a utilização de sistemas digitais, estão (TOCCI; WIDMER; MOSS, 2011):

- Sistemas digitais são fáceis de projetar, pois os circuitos utilizados são de chaveamento. Logo os valores exatos de tensão ou corrente não são importantes, mas sim o intervalo (Alto ou Baixo) no qual eles se localizam;
- Maior exatidão e precisão, já que os sistemas digitais podem manipular uma quantidade considerável de dígitos de precisão;
- Maior tolerância a ruídos, visto que variações de tensão são toleradas dentro de uma determinada faixa;
- Devido à existência de linguagens de programação de *hardware* é possível programar a operação dos sistemas digitais; e
- Ocupam pouco espaço e, consequentemente, um número maior de circuitos digitais podem ser colocados em um circuito integrado.

2.2.2 **Circuitos Digitais**

Os circuitos digitais são dispositivos eletrônicos projetados para produzir tensões de saída que estejam dentro dos intervalos de tensão determinados para os binários 0 e 1, como os intervalos definidos pela Figura 3. Ademais, a resposta dos circuitos digitais é

previsível, ou seja, um circuito digital responderá da mesma maneria a todos os valores de tensão pertencentes ao intervalo permitido para o valor binário 0; de maneira semelhante, os circuitos não notam diferença entre valores de tensão que estejam dentro do intervalo permitido para o valor binário 1 (TOCCI; WIDMER; MOSS, 2011).

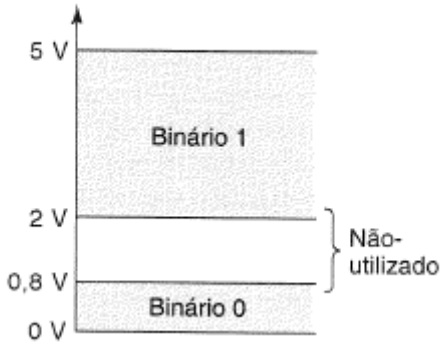

Figura 3 – Intervalo de tensão para binários 0 e 1. Extraído de (TOCCI; WIDMER; MOSS, 2011)

2.2.3 **Desenvolvimento de Sistemas Digitais**

Um sistema digital é composto por grandes quantidades de transistores. Portanto, níveis de abstrações são necessários para tornar a concepção de sistemas digitais menos complexa. Tais abstrações formam módulos com níveis hierárquicos, partido de portas lógicas e indo até processadores complexos (ERCEGOVAC; MORENO; LANG, 1998; MANFRINI, 2017). A Figura 4 apresenta os tipos de hierarquia adotadas na construção de sistemas digitais (MANFRINI, 2017): implementação *Top-down* e *Bottom-up*. Em ambas, o nível de abstração mais elevado é o próprio sistema digital (e.g. circuito integrado) e o menos elevado é o arranjo dos transistores de cada porta lógica.

Na metodologia *Top-down*, o sistema é decomposto em diversos subsistemas de menor complexidade até que se alcance um determinado nível, onde módulos previamente disponíveis permitem a sístase do projeto. Já na abordagem *Bottom-up*, subsistemas de menor complexidade são conectados visando a construção de sistemas mais elaborados. Este processo ocorre até que a especificação funcional seja atingida. Ressalta-se que tando a abordagem *Top-down* quanto a *Bottom-up* são dependentes do conhecimento e experiência do projetista. Além disso, não há garantia da otimalidade do projeto (ERCEGOVAC; MORENO; LANG, 1998; MANFRINI, 2017).

2.2.4 **Circuitos Lógicos**

Os circuitos lógicos, do mais simples ao mais complexo, são construídos por meio da conexão dos mesmos elementos, as portas lógicas. As portas lógicas são blocos construtivos que admitem uma ou mais entradas para produzir uma única saída, neste contexto as

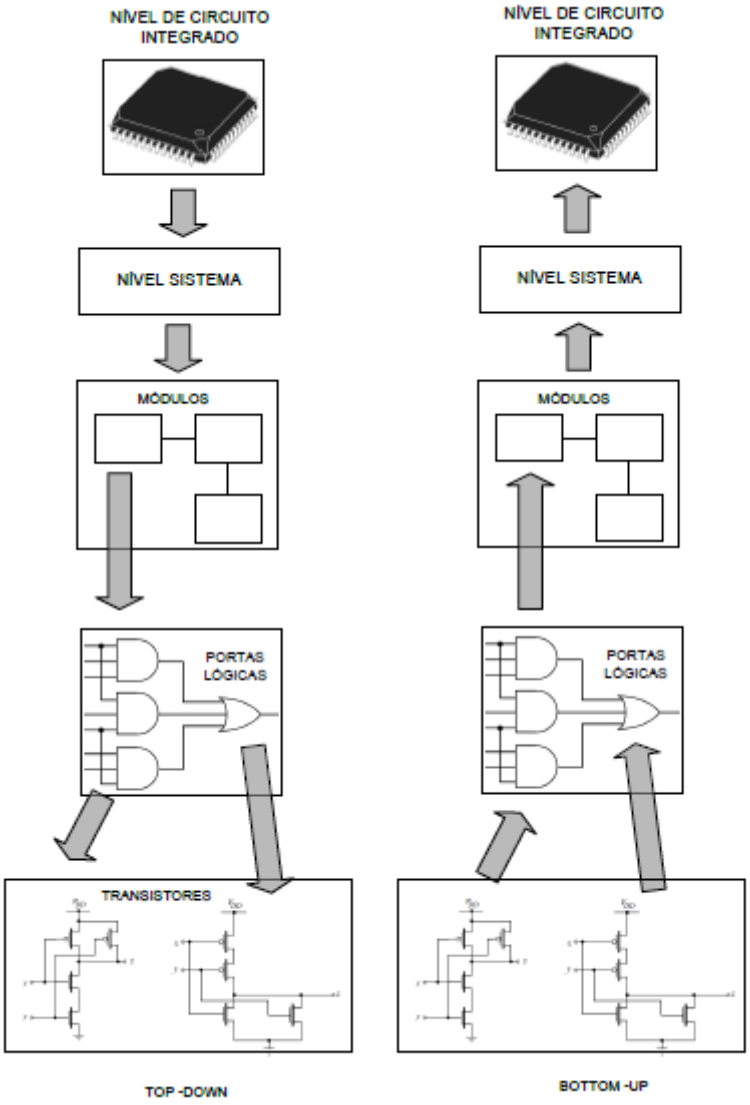

Figura 4 – Implementação hierárquica. Disponível em (MANFRINI, 2017).

entradas e a saída podem assumir o valor 0 ou 1. Neste cenário, os diferentes tipos de circuitos digitais desempenham tarefas pautadas em um determinado conjunto de regras lógicas. Por consequência, os circuitos digitais são chamados de circuitos lógicos e podem ser classificados como (MANFRINI, 2017; TOCCI; WIDMER; MOSS, 2011):

- Circuitos Lógicos Combinacionais, são tais que a saída atual, $y(t)$, depende tão somente da entrada atual, $x(t)$, por isso:

$$y(t) = F(x(t)). \tag{2.1}$$

- Circuitos Lógicos Sequenciais, são tais que a saída atual, $y(t)$, depende da entrada atual, $x(t)$, e de entradas anteriores, $x(0, t-1)$, logo:

$$y(t) = F(x(t), x(0, t-1)). \tag{2.2}$$

A descrição da funcionalidade de um circuito digital ocorre por meio da análise da tabela verdade, que descreve todas as combinações possíveis de valores para as variáveis de entrada e as respectivas saídas lógicas geradas por cada combinação de entrada. A Figura 5 ilustra uma tabela verdade com duas variáveis de entrada, denotadas por A *e* B, e a saída X. Neste cenário, cada linha da tabela verdade é considerada como restrição que deve ser atendida no projeto de um circuito lógico exato.

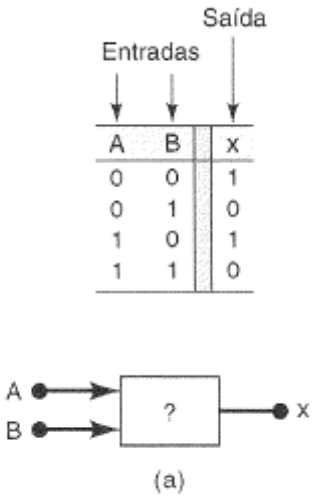

Figura 5 – Exemplo de uma tabela verdade, que descreve a relação entrada/saída de um circuito lógico. Disponível em (TOCCI; WIDMER; MOSS, 2011).

Uma forma de analisar e expressar as informações presentes na tabela verdade ocorre por meio de equações booleanas (BOOLE, 1854). As variáveis de uma equação booleana são discretas, de modo que podem assumir apenas os valores 0 e 1. No que diz respeito aos circuitos lógicos, as variáveis booleanas representam estados de tensão, sendo 5V para "Alto" e 0V para "Baixo". Considerando A e B como entradas de um circuito lógico e F a função lógica desempenhada, cita-se como principais operações booleana os operadores (MANFRINI, 2017; TOCCI; WIDMER; MOSS, 2011):

- **NOT**: O valor de saída é o valor inverso da variável de entrada A, ou seja, $F(0) = 1$ e $F(1) = 0$. Algebricamente: $F = \bar{A}$.
- **AND**: O valor da saída de $F(X)$ é igual a 1 se e somente se todas as entradas forem iguais a 1, caso contrário a saída de $F(X)$ vale 0. Algebricamente: $F = A.B$.
- **NAND**: É a operação inversa da *AND*. Algebricamente: $F = \overline{(A.B)}$.
- **OR**: O valor da saída de $F(X)$ é igual a 0 se e somente se todas as entradas forem iguais a 0, caso contrário a saída de $F(X)$ vale 1. Algebricamente: $F = A + B$.

- **NOR**: É a operação inversa da *OR*. Algebricamente: $F = \overline{A + B}$.
- **XOR**: O valor da saída de $F(X)$ é igual a 0 se e somente se os valores lógicos das entradas A e B forem iguais, caso contrário a saída de $F(X)$ vale 1. Algebricamente: $F = A \oplus B$.
- **XNOR**: O valor da saída de $F(X)$ é igual a 1 se e somente se os valores lógicos das entradas A e B forem iguais, caso contrário a saída de $F(X)$ vale 0. Algebricamente: $F = A \odot B$.

A cada operador booleano está associado uma porta lógica e uma tabela verdade. A Figura 6 mostra as associações das funções NOT, AND, NAND, OR, NOR, XOR e XNOR.

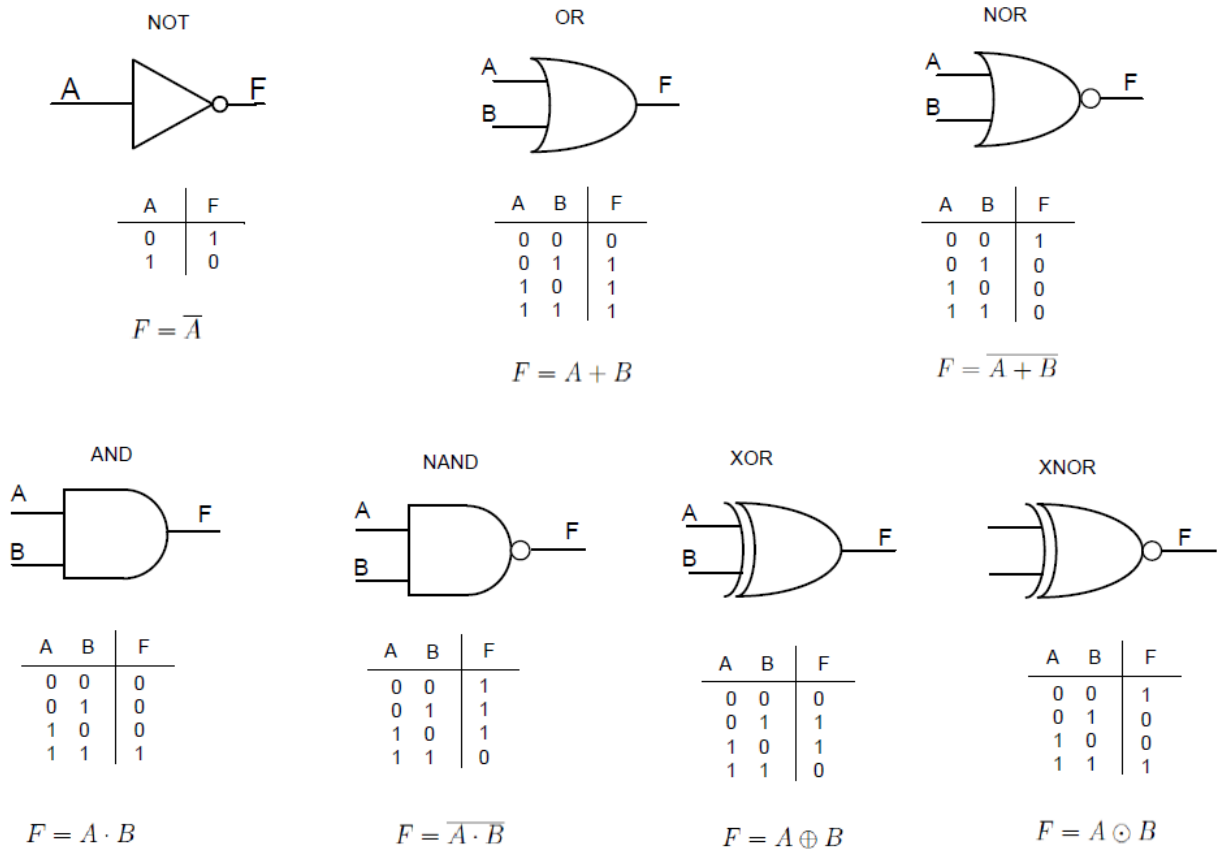

Figura 6 – Associação de operadores booleanos. Extraído de (MANFRINI, 2017).

2.3 *HARDWARE* EVOLUTIVO

O projeto de circuitos eletrônicos segue regras bem definidas, pode ser bastante complexo e exige conhecimento de metodologias especificas. Como consequência, a utilização da Computação Evolucionista como ferramenta para projetar circuitos eletrônicos

pode levar à construção de arquiteturas de circuitos diferentes das arquiteturas tradicionais (KOZA, 1994; MANFRINI, 2017), uma vez que o processo de busca explora soluções que um especialista poderia desconsiderar. Outro ponto positivo da utilização da Computação Evolucionista no projeto de circuitos é a maior autonomia do processo, pois a construção dos circuitos se torna menos dependente do conhecimento do especialista.

O contraste entre a metodologia do especialista e a evolutiva é evidenciado pela Figura 7. A metodologia do especialista é pautada em um algoritmo de projeto determinístico baseado em aplicações de regras bem definidas e que dispõe de um conjunto de componentes restrito, nota-se que, a não ser por eventual inspiração, os resultados provenientes da metodologia do especialista estão limitados a uma sub-região do espaço geral de projeto (MANFRINI, 2017). Por outro lado, na metodologia evolutiva as modificações ocorrem de forma *Bottom-up*, isto é, subsistemas de menor complexidade são conectados visando a construção de sistemas mais elaborados. Neste cenário, possíveis soluções são sintetizadas e testadas a partir da iteração entre o espaço de todos os projetos possíveis, algoritmo evolutivo, montagem e teste, logo uma diversidade maior de soluções são exploradas (MANFRINI, 2017).

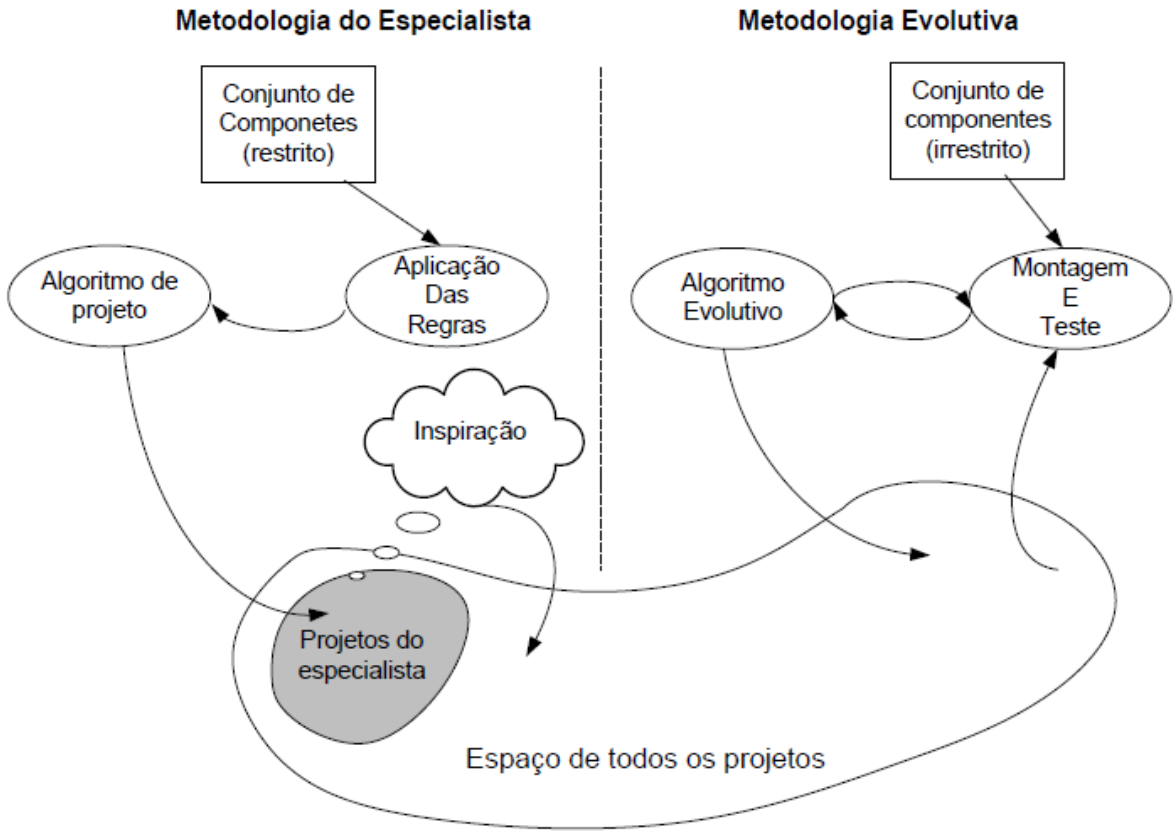

Figura 7 – Metodologia tradicional × Metodologia Evolutiva. Extraído de (MANFRINI, 2017)).

Em um cenário de otimização caixa preta, a estrutura interna de um sistema é desconhecida e as informações sobre a função objetivo são restritas às relações dos valores

de entrada e saída do sistema (LOSHCHILOV, 2017; NOCEDAL; WRIGHT, 2006). O projeto de circuitos eletrônicos por meio da Computação Evolucionista pode ser visto como uma abordagem caixa preta no seguinte sentido: em pose dos dados de entrada e saídas desejadas de um circuito, um algorítimo evolucionista se encarrega, por meio de um processo de busca, de encontrar uma configuração que seja capaz de responder aos requisitos do projeto (MILLER, 2011; MANFRINI, 2017). Neste cenário, as informações desconhecidas, dentro da caixa preta, são codificadas em cromossomos e são, então, submetidas aos mecanismos de busca dos algoritmos evolutivos (MILLER et al., 1997).

A inclusão da Computação Evolucionista às metodologias de construção de circuitos eletrônicos deu vida a uma nova linha de pesquisa, denominada *Hardware* Evolutivo (HE) (VASICEK; SEKANINA, 2011). Nesta abordagem, um algoritmo evolucionista é adotado na concepção de uma arquitetura de *hardware*. Como características pode-se citar (STEPNEY; ADAMATZKY, 2018):

- A solução final deve ser um *hardware*, ou pelo menos ter o potencial para ser implementado em *hardware*;
- A evolução pode ser intrínseca: o processo evolucionário é feito via *hardware*, ou seja, todos os circuitos candidatos são avaliados diretamente em um chip. Para que este processo seja realizado, uma plataforma reconfigurável é necessária. Neste contexto, os chips *Field Programmable Gate Array* (FPGA) são frequentemente utilizados;
- A evolução pode ser extrínseca: os circuitos candidatos são avaliados usando um simulador de circuito. Ao final do processo, obtém-se o circuito que pode ser implementados em *hardware*;
- A implementação final pode apresentar características diferentes das características apresentadas pelo processo tradicional.

O objetivo do HE é desenvolver circuitos complexos, encontrar maneiras de explorar os vastos recursos computacionais disponíveis e oferecer soluções inovadoras na construção de circuitos digitais. Bons resultados, usando *hardware* evolutivo, foram alcançados na síntese de circuitos digitais combinacionais (SHE; LAI, 2009; IRFAN et al., 2010; MILLER, 2011; MANFRINI, 2017), síntese de filtros digitais (MILLER, 1999b), filtros adaptativos (LOVAY; PERETTI; ROMERO, 2015), circuitos amplificadores (KOZA et al., 2005), engenharia biomédica (KAJITANI et al., 1999) e em aplicações tolerantes à falha (SALVADOR et al., 2011; KEYMEULEN et al., 2000; CANHAM; TYRRELL, 2002).

No entanto, o projeto de circuitos via Computação Evolucionista encontra dificuldades na representação de soluções e tempo de cálculo da função de avaliação. A primeira dificuldade está relacionada à necessidade da utilização de cromossomos longos

para representar soluções complexas o que implica em grandes espaços de busca que são tipicamente difíceis de pesquisar, gerando dificuldades ao processo de busca (STEPNEY; ADAMATZKY, 2018). Já a segunda dificuldade está relacionada à quantidade de entradas de um determinado circuito, neste cenário o tempo necessário para o cálculo da função de avaliação cresce exponencialmente com o número de entradas (MILLER, 2011) o que pode levar a uma inaplicabilidade do projeto evolutivo, uma vez que o tempo de projeto é determinante na adoção de um método de concepção (VASICEK; SEKANINA, 2015a).

Encontrar alternativas que levem a um refinamento das técnicas bioinspiradas que podem ser utilizadas na obtenção de arquiteturas complexas e compreender de forma mais profunda o mecanismo evolutivo, ou seja, como os operadores genéticos interagem com a solução (GOLDMAN; PUNCH, 2013) é o desafio enfrentado pelo *Hardware* Evolutivo no que diz respeito à escalabilidade (MILLER, 2011; HRBACEK; MRAZEK; VASICEK, 2016; STEPNEY; ADAMATZKY, 2018).

2.3.1 **Funções Objetivo de Projeto de Circuitos Digitais**

Em ciências exatas, o termo otimização ou programação matemática refere-se ao estudo de problemas que buscam minimizar ou maximizar uma função por meio da escolha sistemática dos valores de variáveis (sejam elas, reais, inteiras ou mistas) dentro de um conjunto viável. A tal função se da o nome de função objetivo (PHILLIPS; SOLBERG, 1987). Esta subseção trata das funções objetivo comumente utilizadas na abordagem evolutiva de concepção de circuitos digitais.

2.3.1.1 *Estimativa de Área Ocupada*

Analisar a área ocupada pelos componentes de um circuito digital é relevante, pois busca-se construir sistemas digitais que ocupem cada vez menos espaço visando reduções no custo de fabricação, uma vez que quanto maior é o circuito maior é a área da placa de circuito impresso necessária para sua implementação. Portanto, deve-se agregar um maior número de portas lógicas em uma única pastilha de silício (TOCCI; WIDMER; MOSS, 2011). A estimativa de área ocupada pelos componentes de um circuito digital é característica importante e não depende da aplicação específica.

Pode-se estimar a área ocupada por um circuito digital contando os elementos de processamento (e.g. transistores, portas elementares, componentes de nível funcional) usados na construção de determinada arquitetura (KAUFMANN; PLATZNER, 2007; SEKANINA; VASICEK, 2013). Supondo que o número de elementos de processamento presentes em um circuito digital c, denotado como $blocos\ usados(c)$, esteja no intervalo $0 \leq K \leq m$ que representa todos os elementos de processamento disponíveis. Com base

nesse número, a área pode ser definida como:

$$area(c) = 1 - \frac{blocos\ usados(c)}{m}. \tag{2.3}$$

Um circuito com área mínima obtém um valor de área de 1, enquanto que um circuito que utiliza todos os blocos lógicos disponíveis tem um valor de área de 0.

2.3.2 *Estimativa de Erro*

A descrição da funcionalidade de um circuito digital ocorre por meio da análise da tabela verdade, que descreve todas as combinações da variável de entrada e as respectivas saídas lógicas geradas por cada entrada. Neste contexto, as funções de estimativa erro adotadas no projeto evolutivo de circuitos digitais são baseadas em métricas de distância aritmética (Equações 2.5, 2.6, 2.7, 2.8) e na distância Hamming entre duas palavras binárias (Equação 2.4).

Considerando um vetor de entradas i, $PB^{(i)}_{factivel}$ como a palavra binária gerada na saída de um circuito digital totalmente funcional (que atende por completo a tabela verdade) de w-bits a partir da entrada i e $PB^{(i)}_{ordinario}$ como a palavra binária gerada na saída de um circuito ordinário de w-bits também a partir da entrada i, a distância de Hamming (Equação 2.4) é definida como o número de posições nas quais os bits das duas palavras binárias diferem entre si. Neste contexto, $\oplus$ representa o operador lógico *XOR* que resulta em um valor lógico verdadeiro se e somente se as entradas apresentarem dígitos diferentes.

$$d_h = \sum_{\forall i}(PB^{(i)}_{factivel} \oplus PB^{(i)}_{ordinario}). \tag{2.4}$$

Considerando $O^{(i)}_{factivel}$ como o valor de saída aritmética de um circuito digital totalmente funcional (que atende por completo a tabela verdade) de w-bits e $O^{(i)}_{ordinario}$ como o valor de saída aritmética de um circuito ordinário de w-bits gerado quando um vetor de entradas i é aplicado aos dois circuitos, temos que as funções de erro baseadas na distância aritmética mais comumente encontradas da literatura são (VASICEK; SEKANINA, 2014a):

- *Error magnitude* (e_{wst}). Esse tipo de erro é denotado como erro de pior caso e pode ser definido como:

$$e_{wst} = \max_{\forall i} |O^{(i)}_{factivel} - O^{(i)}_{ordinario}|. \tag{2.5}$$

- *Relative error* (e_{rel}). Esse tipo de erro é calculado como na Equação 2.6 e é definido como o máximo valor da razão entre a diferença absoluta dos valores de saída gerados por um circuito ordinário e o circuito factível pelo valor de saída do circuito totalmente funcional. É necessário ter atenção nos casos em que $O^{(i)}_{factivel} = 0$.

$$e_{rel} = \max_{\forall i} \frac{|O^{(i)}_{factivel} - O^{(i)}_{ordinario}|}{O^{(i)}_{factivel}}. \tag{2.6}$$

- *Average error magnitude* (e_{avg}). Esse erro é calculado como na Equação 2.7 e é definido como a soma das diferenças absolutas entre a saída do circuito digital factível e o ordinário, ou seja, erro total divido pelo número de entradas:

$$e_{avg} = \frac{e_{tot}}{2^{2w}} = \frac{\sum_{\forall i} |O^{(i)}_{factivel} - O^{(i)}_{ordinario}|}{2^{2w}}. \tag{2.7}$$

- *Error probability (error rate)* (e_{prob}). Esse tipo de erro é definido como a porcentagem de vetores de entrada para os quais a saída ordinária difere da factível:

$$e_{prob} = \frac{\sum_{\forall i}, O^{(i)}_{factivel} \neq O^{(i)}_{ordinario}}{2^{2w}}. \tag{2.8}$$

2.3.2.1 *Estimativa de Dissipação de Potência*

Os componentes construtivos de um circuito digital, assim como qualquer outro condutor de corrente elétrica, geram calor quando a corrente elétrica percorre seus terminais, logo o efeito Joule está presente em todos as aplicações digitais (HAYT; BUCK, 2003). Circuitos digitais são implementados em placas de circuito impresso que possuem limitações térmicas de operação, com isso manter a dissipação de potência em níveis mínimos é importante para o funcionamento correto e seguro dos circuitos digitais (SEDRA; SMITH, 2004). Neste contexto, as principais fontes de dissipação de potência em um circuito digital são: (CHANDRAKASAN; BRODERSEN, 1995):

1. Componente de curto-circuito ($P_{short-circuit}$).
2. Componente de corrente de fuga ($P_{leakage}$).
3. Componente de comutação ($P_{switching}$).

Por meio destes componentes, a estimativa de dissipação de potência em circuitos digitais pode ser calculada como:

$$P = I_{sc}.V_{dd} + I_{leakage}.V_{dd} + a_{0\to1}.C_L.f_{clk}.V^2_{dd}, \tag{2.9}$$

em que I_{sc} é a corrente de curto-circuito, V_{dd} é a tensão de alimentação, $I_{leakage}$ representa as correntes de fuga, C_L é a capacitância de carga, f_{clk} é a frequência do *clock* e $a_{0\to1}$ é o fator de transição de nós. Os três termos da Equação 2.9 correspondem, respectivamente, à dissipação de potência devido à corrente de curto-circuito ($P_{short-circuit}$), ao componente de corrente de fuga ($P_{leakage}$) e o componente de comutação ($P_{switching}$). Neste cenário, destaca-se que a componente de comutação ($P_{switching}$) é a componente que contribui de forma mais acentuada no cálculo da potência dissipada por um circuito digital. Isso ocorre devido à presença do fator de frequência de *clock*. Valores elevados de *clock* fazem com que a componente de comutação cresça de forma significativa, por isso existe a necessidade de

se limitar a frequência de *clock* dos sistemas digitais (CHANDRAKASAN; BRODERSEN, 1995).

Em aplicações digitais apenas dois níveis lógicos são permitidos: o nível lógico baixo, zero lógico ou 0; e o nível lógico alto, um lógico ou 1. O fator transição de nós (a probabilidade de transição de 0 para 1 de uma porta lógica) é dado pela probabilidade de que a saída de uma determinada porta seja igual a zero lógico (p_0) multiplicada pela probabilidade de que o próximo estado seja igual ao um lógico (p_1). O fator de transição de nós, definido na Equação 2.10, quantifica o número médio de vezes que uma porta lógica faz uma transição de estado que consome potência dentro de um período de *clock*.

$$a_{0\to1} = p_0.p_1 = (1 - p_1).p_1. \tag{2.10}$$

Para ilustrar o cálculo do fator transição de nós, é considerado o circuito de exemplo presente na Figura 8.

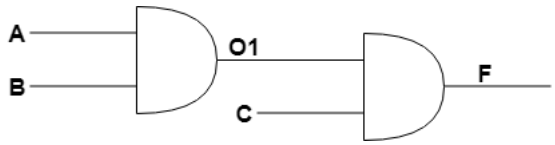

Figura 8 – Circuito exemplo para cálculo do fator transição de nós.

Neste cenário, assume-se que todas as entradas primárias (A, B, C) apresentam comportamento estático, são não correlacionadas e aleatórias, ou seja, a probabilidade de que as entradas primárias sejam iguais ao um lógico é igual à probabilidade de que as entradas primárias sejam iguais ao zero lógico ($p_1^{A,B,C} = p_0^{A,B,C} = 0,5$).

A probabilidade de que a saída da primeira porta AND, $O1$, presente no circuito da Figura 8 esteja no estado 1 é dada por:

$$p_1 = p_1^A.p_1^B. \tag{2.11}$$

Portanto, o fator transição de nós da saída $O1$ pode ser calculado como:

$$a_{0\to1} = p_0^{O1}.p_1^{O1} = (1 - p_1^{O1}).p_1^{O1} = (1 - p_1^A.p_1^B).p_1^A.p_1^B. \tag{2.12}$$

De forma similar, a probabilidade de que a saída da segunda porta AND, F, presente no circuito da Figura 8 esteja no estado de nível lógico alto é dada por:

$$p_1 = p_1^{O1}.p_1^C. \tag{2.13}$$

Logo, o fator transição de nós da saída F pode ser calculado como:

$$a_{0\to1} = p_0^F.p_1^F = (1 - p_1^F).p_1^F = (1 - p_1^{O1}p_1^C).p_1^{O1}.p_1^C. \tag{2.14}$$

Diante do exposto, a Tabela 1 apresenta os valores de probabilidades de sinal e o fator transição de nós de cada porta presente no circuito ilustrado pela Figura 8.

Tabela 1 – Probabilidades de sinal e fator transição de nós.

	O1	**F**
p_1	1/4	1/8
p_0	3/4	7/8
$a_{0\to1}$	**3/16**	**7/64**

2.3.2.2 *Estimativa de Atraso de Propagação (Delay)*

O *delay* de um circuito digital é o tempo gasto para as alterações na entrada causarem algum efeito na saída (HRBACEK; MRAZEK; VASICEK, 2016). Neste sentido, o *delay* de um circuito digital é a soma de todos os *delays* de todas as portas presentes no caminho mais longo ou caminho crítico (caminho com maior tempo de atraso entre dois pontos do circuito). Neste cenário, o caminho crítico impõe um limite de velocidade máxima de resposta (TOCCI; WIDMER; MOSS, 2011). O *delay* de um circuito digital c é definido aqui como D_c e pode ser calculado como:

$$D_c = \max_{\forall p \in caminho} \sum_{c_i \in p} t_d(c_i), \tag{2.15}$$

em que t_d é o atraso de uma célula c_i. O valor de t_d é normalmente fornecido pelos fabricantes.

A título de exemplificação, um circuito é ilustrado pela Figura 9. Além disso, a Tabela 2 apresenta valores hipotéticos de atrasos t_d para cada porta lógica que compõe o circuito da Figura 9.

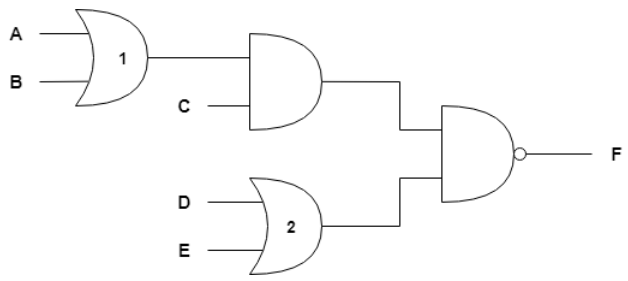

Figura 9 – Circuito hipotético apresentado aqui para ilustrar o cálculo do seu *delay*.

Tabela 2 – Valores hipotéticos de atrasos t_d para cada uma das portas lógicas.

Porta	t_d **(ns)**
OR	3
AND	5
NAND	8

O circuito ordinário ilustrado pela Figura 9 apresenta dois caminhos que ligam a saída F às entradas primárias. São eles o caminho 1 e o caminho 2.

O caminho 1 é composto pelas portas $OR_1 \rightarrow AND \rightarrow NAND$. O *delay* do caminho 1 corresponde à soma dos atrasos t_d das portas lógicas que o compõe, logo $D_{caminho1} = 16ns$. A Figura 10 ilustra o caminho 1.

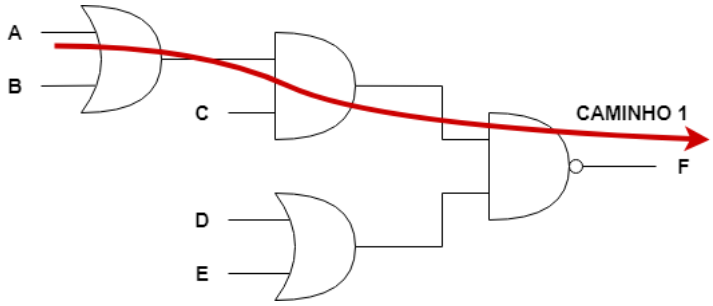

Figura 10 – Caminho 1.

Já o caminho 2 é composto pelas portas $OR_2 \rightarrow NAND$. O *delay* do caminho 2 corresponde à soma dos atrasos t_d das portas lógicas que o compõe, logo $D_{caminho2} = 11ns$. A Figura 11 ilustra o caminho 2.

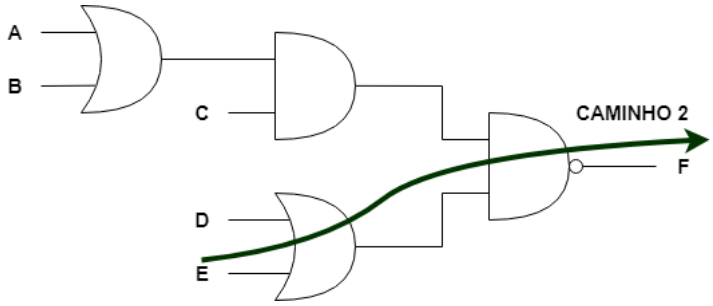

Figura 11 – Caminho 2.

Como $D_{caminho1} > D_{caminho2}$, tem-se que o caminho 1 é o caminho crítico. Consequentemente, o *delay* do circuito da Figura 9 é igual à $D_c = 16ns$.

3 PROGRAMAÇÃO GENÉTICA CARTESIANA

Os Algoritmos Genéticos (AGs) são algoritmos de otimização que utilizam processos estocásticos inspirados em mecanismos biológicos envolvidos no processo de evolução das espécies (MILLER, 2011). Os AGs são estabelecidos como ferramentas poderosas de para lidar com problemas de otimização do mundo real (KAZARLIS; KALOMIROS; KALAITZIS, 2016; MANFRINI, 2017). Neste contexto, a Programação Genética (PG) é uma técnica concebida para evoluir programas computacionais codificados por meio de árvores, sendo capaz de otimizar estruturas funcionais que realizam por exemplo operações lógicas e condicionais (KOZA, 1992).

A partir das possibilidades que a Programação Genética oferece, Miller et al. (MILLER et al., 1997) propôs um método variante da PG voltado para a evolução de circuitos digitais, batizado de Programação Genética Cartesiana[1] (CGP). Ao invés de utilizar árvores para representar as soluções como ocorre na PG, a CGP utiliza grafos acíclicos direcionados para modelar os circuitos candidatos. Os circuitos são representados como uma matriz $n_r \times n_c$ de nós interconectados, neste cenário cada nó representa um elemento de processamento lógico específico. Ressalta-se que são permitidas apenas conexões diretas com uma certa profundidade l visando evitar a realimentação dos elemento de processamento lógico (MILLER et al., 1997).

3.1 REPRESENTAÇÃO DE SOLUÇÕES

Na abordagem CGP, um circuito é representado como uma matriz $n_r \times n_c$ de nós interconectados, em que n_r é o número de linhas e n_c é o número de colunas da matriz. A representação de uma solução candidata é ilustrada na Figura 12. Os genes que constituem os cromossomos são números inteiros. Ressalta-se que o genótipo tem tamanho fixo ($n_r \times n_c$), contudo, o tamanho do fenótipo varia dentro do intervalo de zero até ($n_r \times n_c$) nós. Um fenótipo composto por zero nós ocorre quando todas as saídas de dados estão conectadas diretamente às entradas de dados.

Os elementos de processamento, representados por F podem ser portas elementares (e.g. AND, OR, NAND, NOR, XOR) ou componentes de nível funcional (e.g. somadores, comparadores, deslocadores e multiplicadores). As portas elementares são utilizadas na evolução de circuitos a nível de porta lógica como neste trabalho, já os componentes de nível funcional são adotados na evolução de circuitos a nível estrutural como filtros digitais (MILLER, 2011).

As entradas dos nós podem ser conectadas a uma das n_i entradas primárias ou a um nó em uma das l colunas anteriores em que, l é definido como *levels-back*, que controla

[1] *Cartesian Genetic Programming*

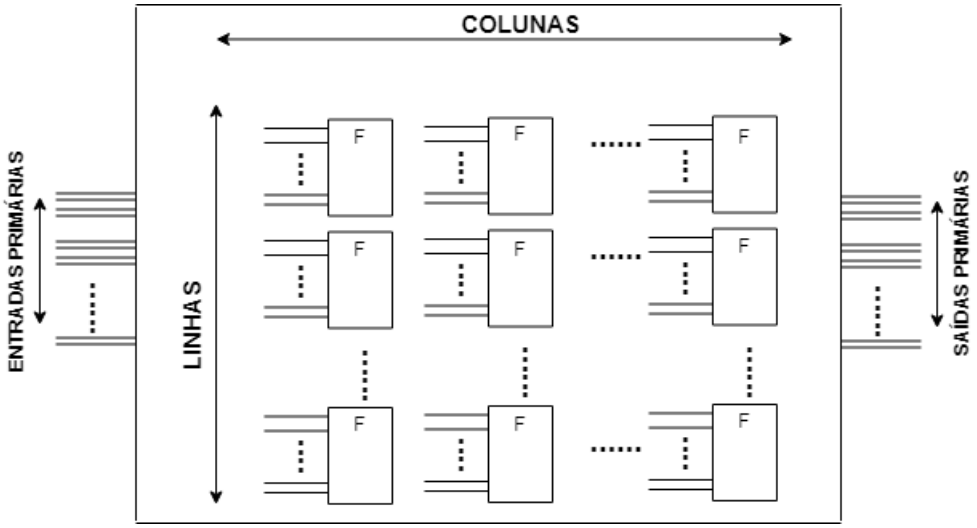

Figura 12 – Ilustração da representação das soluções candidatas na CGP. Adaptado de (HRBACEK; MRAZEK; VASICEK, 2016).

a conectividade do grafo. O *levels-back* indica a quantidade de nós à esquerda nos quais um determinado nó pode ser conectado. Por exemplo, se $l = 1$ um determinado nó só pode ter como entrada um dos nós pertencentes à coluna imediatamente a sua esquerda ou às n_i entradas primárias. Se $l = 2$ as entradas de um determinado nó só podem ser nós pertencentes às duas colunas imediatamente a sua esquerda ou às n_i entradas primárias. Um das características dos circuitos digitais combinacionais é que não ocorre realimentação dos componentes digitais (TOCCI; WIDMER; MOSS, 2011). Em abordagens CGP, não é permitido que nós pertencentes à mesma coluna sejam conectados entre si. Isso é feito para que não ocorra *feedback* de nós (MILLER, 2011; MANFRINI, 2017).

Cada nó tem um número fixo de entradas N_{ni} e saídas N_{no}, e pode executar uma das funções de processamento do conjunto Γ ($F \in \Gamma$). Cada uma das n_o saídas primárias podem ser conectadas a uma entrada primária ou a saída de um nó. Neste sentido, a CGP pode lidar com problemas com várias saídas, tal característica é uma das vantagens da representação cartesiana quando comparada com a Programação Genética tradicional (MILLER, 2011). Os genes que constituem um cromossomo de um circuito são descritos por uma sequência de inteiros que representa os índices das entradas dos nós, i_1 e i_2, e a respectiva operação lógica F a ser executada quando a saída do nó for calculada (i_1, i_2, F). Os últimos n_o inteiros da codificação CGP especificam os nós ou entradas primárias as quais as saídas primárias do circuito estão conectadas.

A Figura 13 ilustra a codificação de um circuito candidato por meio da abordagem CGP. Neste caso, o conjunto de entradas primárias é $\{a_0, a_1, b_0, b_1\}$, por outro lado o conjunto de saídas primárias é, $\{p_0, p_1, p_2, p_3\}$, logo $n_i = n_o = 4$. O número de entradas de cada nó é $N_{ni} = 2$, já o de saídas de cada nó é $N_{no} = 1$. O número de colunas $n_c = 6$ e o número de linhas $n_r = 1$, tem-se ainda que $l = 6$ e o conjunto $\Gamma = \{AND^0, OR^1\}$.

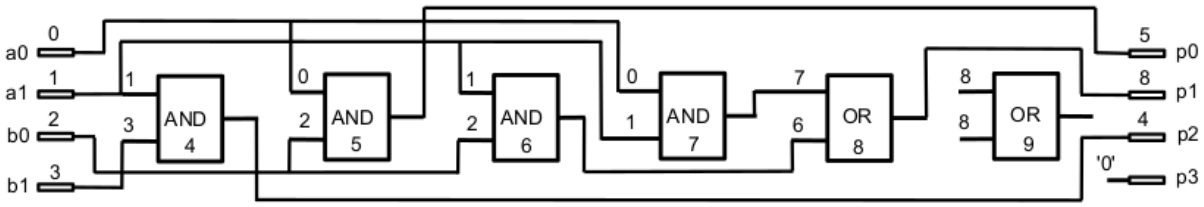

Figura 13 – Codificação de um circuito digital candidato por meio da abordagem CGP. O vetor V = [{**1, 3**, $\underline{0}$}, {**0, 2**, $\underline{0}$}, {**1, 2**, $\underline{0}$}, {**0, 1**, $\underline{0}$}, {**7, 6**, $\underline{1}$}, {**8, 8**, $\underline{1}$}, {*5, 8, 4, 0*}] é o cromossomo correspondente à solução candidata. Os alelos em negrito representam os índices de entradas dos nós e os alelos cujos valores estão sublinhados representam as respectivas operações lógicas. Já os genes cujos alelos estão em itálico, representam os índices de conexão das n_o saídas primárias. Extraído de (VASICEK; SEKANINA, 2015b).

Destaca-se que o sexto nó ({**8, 8**, $\underline{1}$}) não causa nenhum efeito no circuito final, pois sua saída não está conectada à nenhuma saída primária ou a qualquer outro nó que influencie os valores de qualquer uma das quatro saídas primárias, logo o sexto nó não faz parte do fenótipo. Este tipo de nó é classificado como nó inativo ou neutro. A CGP explora intensamente a presença de nós que não fazem parte do fenótipo, pois estes nós possibilitam uma grande diversidade no fenótipo (MANFRINI, 2017).

De acordo com a literatura, o tamanho do genótipo ($n_r \times n_c$) deve ser consideravelmente maior do que a quantidade de nós necessária para se resolver o problema, visto que, em alguns casos, 95% dos nós de uma solução candidata são inativos (MILLER, 2011; VASICEK; SEKANINA, 2015a). Para aplicações em que não há afirmações suficientes sobre a quantidade de nós necessários para resolver determinado problema, Miller et al. (MILLER, 2011) sugere adotar um valor entre 100 e 1000.

Durante o processo evolutivo, princípios de seleção e reprodução de soluções são aplicados aos indivíduos de uma população. Por meio da seleção, mecanismo que direciona a busca para a solução ótima, é determinado o número de indivíduos que serão progenitores e irão gerar um certo número de descendentes para a próxima geração. No decorrer do processo de seleção é levado em conta a aptidão dos indivíduos, logo os indivíduos que têm melhor aptidão possuem maior probabilidade de se reproduzir (EIBEN; SMITH, 2003). Neste contexto, é necessário um conjunto de operadores para que se consiga gerar novas populações que contenham indivíduos com melhor aptidão do que os indivíduos das populações passadas. No âmbito da CGP, o operador de recombinação geralmente não é utilizado, pois segundo Miller et al. (MILLER, 2011) a recombinação foi prejudicial para a evolução. Logo, o operador de mutação é comumente adotado como o único operador genético em abordagens CGP. (MILLER, 2011; KE; CAO; NIE, 2010; MANFRINI, 2017).

Devido a aleatoriedade presente no processo de busca da CGP a aptidão do melhor indivíduo pode diminuir de uma geração para outra. Para evitar que os indivíduos de maior aptidão desapareçam a abordagem elitista, que não permite que o melhor indivíduo piore de uma geração para a outra, pode ser adotada. Neste contexto, os indivíduos de maior aptidão são inseridos de forma automática na próxima geração (GASPAR-CUNHA; TAKAHASHI; ANTUNES, 2012; MANFRINI, 2017). Destaca-se que a abordagem proposta neste trabalho utiliza a abordagem elitista.

3.2 MUTAÇÃO NA CGP

A mutação pontual é o operador de mutação mais utilizado na CGP (MILLER, 2011). Neste contexto, um alelo de um gene é selecionado de forma aleatória e o valor deste alelo é trocado por outro valor válido escolhido de forma aleatória. A quantidade de genes que sofrem mutação em uma única operação de mutação pontual é definida pelo usuário. Este valor pode ser ou normalmente é definido como uma porcentagem do número total de genes presentes em um genótipo, tal porcentagem é denominada taxa de mutação pontual (μ_r) (MILLER, 2011). A taxa de mutação pontual (μ_r) é geralmente definida para modificar até 5% dos genes (HRBACEK; MRAZEK; VASICEK, 2016).

Se um gene que representa a função desempenhada por um nó é selecionado durante o processo de mutação, um valor válido para substituir o valor atual do gene é outra função presente no conjunto Γ de funções. Se um gene que representa as entradas de um nó é selecionado para mutação, um valor válido para substituir o valor atual do gene é a saída de qualquer nó anterior, respeitando o *levels-back*, ou um das entradas primárias. Por fim, se um gene que representa uma das n_o saídas primárias for escolhido para sofrer mutação, um valor válido para substituição é qualquer nó presente no genótipo ou de uma das n_i entradas primarias. Um exemplo apresentado por Miller et al. (MILLER, 2011) está na Figura 14, que representa o genótipo e fenótipo de um circuito digital codificado através da CGP, e na Figura 15, para ilustrar o resultado da aplicação do operador de mutação pontual no gene o_A. Neste exemplo, a mutação altera o alelo do gene o_A de 6 para 7. Como consequência da mutação os nós 3 e 7 que eram inativos (traços cinzentos) se tornam ativos e, em contrapartida, os nós 2, 5 e 6 tornam-se inativos. Fica claro que uma mudança pequena no genótipo pode ocasionar mudanças drásticas no fenótipo.

3.3 RECOMBINAÇÃO NA CGP

A utilização do operador de recombinação em abordagens CGP não é comum (MILLER, 1999a), visto que o operador de recombinação não contribuiu de forma significativa para melhorar a qualidade das soluções (MILLER, 2011). Visando investigar os efeitos do operador de recombinação na CGP, Clegg et al. (CLEGG; WALKER; MILLER, 2007) utilizou um operador de recombinação comumente adotado em abordagens de PG, além

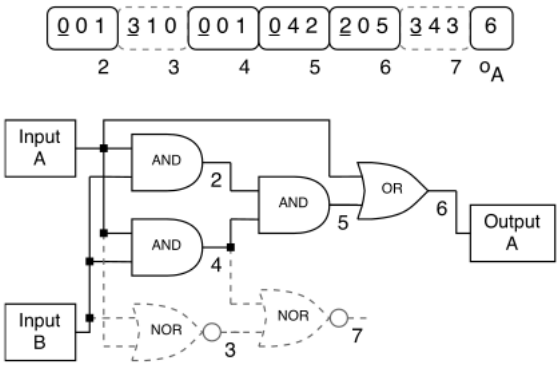

Figura 14 – Genótipo e fenótipo antes da mutação.

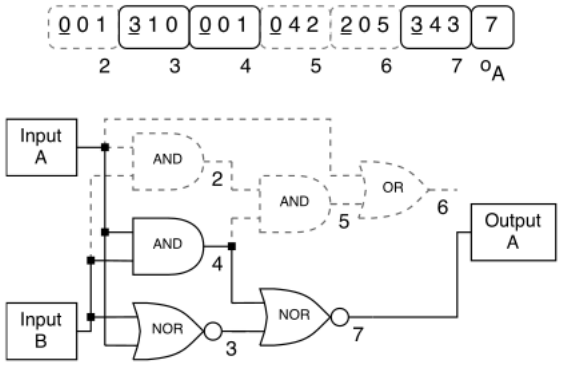

Figura 15 – Genótipo e fenótipo depois da mutação.

disso, foi adicionada uma camada extra de codificação do genótipo na qual todos os genes de um determinado cromossomo são codificados como um número dentro do intervalo $[0, 1]$. Os resultados obtidos foram promissores quando aplicados a problemas de regressão simbólica. Contudo, é necessário uma maior investigação para determinar e avaliar as vantagens da estratégia adotada (CLEGG; WALKER; MILLER, 2007).

Um novo operador de recombinação para CGP, denominado X-CGP, foi proposto por (SILVA; BERNARDINO, 2018) para projetar circuitos lógicos combinacionais. Concluiu-se que o método proposto obteve melhores resultados quando comparado com métodos que utilizam apenas um cromossomo para representar soluções, além disso, o método proposto apresentou melhor desempenho do que as técnicas multi-cromossomo em determinados problemas. Apesar dos trabalhos apresentados, segundo Miller (MILLER, 2011), a recombinação não é indicada na evolução de circuitos digitais, portanto a técnica proposta neste trabalho não utiliza o operador de recombinação.

3.4 ALGORITMO EVOLUTIVO

A estratégia evolutiva (*Evolutionary Strategy*) é o algoritmo de busca mais amplamente adotado em abordagens CGP. Em especial, a variante $(\mu + \lambda) - ES$ é a mais

comum. Neste contexto, μ representa o número de progenitores e λ representa o número de descendentes (RECHENBERG, 1973). Na estrategia evolutiva $(\mu + \lambda) - ES$, uma quantidade μ de progenitores gera, por meio de operadores genéticos, uma quantidade λ de descendentes. Os μ indivíduos que compõe a população de pais e os λ indivíduos que compõe a população de descendentes são avaliados e ordenados em relação à sua aptidão. O processo de seleção encarrega-se de selecionar os μ melhores indivíduos da população para que estes tornem-se os progenitores da próxima geração.

No contexto da CGP, normalmente adota-se $\mu = 1$ e $\lambda = 4$ (MILLER, 2011). Ressalta-se que em situações em que a aptidão do melhor descendente é igual à aptidão do progenitor, o melhor descente torna-se o progenitor da próxima geração. Essa estratégia de substituição é utilizada visando explorar a presença de nós inativos no genótipo. Tal estrategia é benéfica para o processo de busca, pois evita mínimos locais e aumenta a diversidade populacional (MILLER, 2011; STEPNEY; ADAMATZKY, 2018).

Uma implementação CGP baseada na estrategia evolutiva $(\mu + \lambda) - ES$, em que $\mu = 1$ e $\lambda = 4$ é apresentada no Algoritmo 1. Neste contexto, inicialmente os $(\mu + \lambda)$ indivíduos são gerados de forma aleatória e são, então, avaliados. O indivíduo que apresentar a melhor aptidão é selecionado como progenitor. Por meio do operador de mutação pontual, o progenitor gera quatro descendentes. Após a criação da nova população seleciona-se um novo progenitor a partir da seguinte análise:

- Se um dos descentes é melhor ou semelhante ao progenitor, então o melhor descendente substitui o progenitor;
- Se nenhum dos descentes é melhor ou semelhante ao progenitor, então o progenitor é mantido.

O algoritmo segue até que um critério de parada seja atendido.

Algoritmo 1: CGP $(1 + 4)$. Adaptado de (MILLER, 2011).

```
1  for i ← 1 to 5 do
2  |  Gera indivíduo i;
3  end
4  Progenitor ← melhor indivíduo;
5  while critério de parada não for atendido do
6  |  for i ← 1 to 4 do
7  |  |  Filho i ← Mutação(Progenitor);
8  |  end
9  |  Seleciona o progenitor usando as regras abaixo:
10 |  if Algum descendente é melhor ou igual ao progenitor then
11 |  |  Melhor descendente substitui o progenitor;
12 |  else
13 |  |  Progenitor é mantido;
14 |  end
15 end
```

4 CIRCUITOS APROXIMADOS E OTIMIZAÇÃO MULTIOBJETIVO

A Computação Aproximada (CA) aparece como um novo paradigma de projeto em resposta à necessidade incessante de que os sistemas digitais apresentem alto desempenho, sejam energeticamente eficientes e ocupem pouco espaço sem que o custo de produção e a complexidade dos sistemas aumentem (HAN; ORSHANSKY, 2013). A CA explora a capacidade de certos sistemas tolerarem perdas em relação ao comportamento funcional perfeito (i.e. precisão) em prol da diminuição da complexidade, redução de custos, eficiência energética e tempo de resposta (HAN; ORSHANSKY, 2013; MANFRINI, 2017). Em relação à sistemas digitais, aproximações podem ser introduzidas a partir do nível de portas lógicas até aproximações introduzidas à nível de linguagem de descrição de hardware (VASICEK; SEKANINA, 2015b). Neste contexto, circuitos digitais concebidos via Computação Aproximada são classificados como circuitos digitais aproximados (VASICEK; SEKANINA, 2015b).

Fundamentos da CA são encontrados no projeto de circuitos aritméticos aproximados (VASICEK; SEKANINA, 2014a; VASICEK; SEKANINA, 2015b; BABU; BALAJI, 2016; HRBACEK; MRAZEK; VASICEK, 2016), no projeto aproximado de sistemas que envolvam processamento de multimídia como aplicações de compressão de imagens (KULKARNI; GUPTA; ERCEGOVAC, 2011) e aplicações em áudio e vídeo (JULIO et al., 2015; HUANG; KUMAR; ABBAS, 2018), no projeto aproximado de blocos de processamento de alto nível como transformada discreta de cosseno, finita e infinita (VENKATARAMANI et al., 2012), filtros digitais de resposta finita ao impulso (FIR) e resposta infinita ao impulso (IIR) (SOARES; BAMPI; COSTA, 2015; KANG; KIM; KANG, 2016) e também em projetos aproximados de circuitos integrados que combinam centenas de milhares de elementos lógicos em um único chip (i.e. *Very Large Scale Integration*) (ALIOTO, 2017; BALASUBRAMANIAN; DANG; MASKELL, 2017).

Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016) propôs uma abordagem evolutiva multiobjetivo para projetar circuitos digitais aproximados que utiliza a Programação Genética Cartesiana para representar circuitos candidatos, mutação pontual para gerar descendentes e o algoritmo *Non-dominated Sorting Genetic Algorithm II* para explorar o espaço de busca analisando o *trade-off* entre precisão, *delay* e dissipação de potência. A população inicial foi composta por um conjunto de circuitos totalmente convencionais em vez de um único circuito convencional ou inicialização da população de forma aleatória. O método foi capaz de projetar centenas de versões aproximadas de somadores e multiplicadores de 8 bits que alcançaram economias de dissipação de potência e *delay*.

Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) propôs um algoritmo de busca local denominado *hybrid Evolutionary Strategy* (hES) baseado na estratégia

evolutiva $(\mu + \lambda) - ES$ e nos conceitos de dominância de Pareto para investigar os efeitos da periodização de algoritmos de busca multiobjetivo. O algoritmo hES utiliza os métodos *Fast non dominated sort* e *Crowding-distance* propostos por Deb et al. (DEB et al., 2002) para classificar as soluções em frente de não dominância e preservar a diversidade das soluções, respectivamente. A estratégia hES e sua periodização com o NSGA-II (Algoritmo 2) foi comparada com o *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (ZITZLER; LAUMANNS; THIELE, 2001) e com o próprio NSGA-II no projeto de somadores e multiplicadores de 2 bits. Neste cenário, a CGP foi adotada como modelo de representação dos circuitos e a qualidade funcional das soluções foi tratada como restrição, enquanto área do circuito e *delay* foram selecionadas como objetivos. Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) conclui, de acordo com os resultado obtidos em seu estudo, que para a evolução de circuitos digitais que utilizam a CGP como ferramenta de representação o algoritmo hES e suas periodizações são significativamente melhores que os algoritmos de referência (NSGA-II e SPEA2).

No contexto da CA e projeto de circuitos digitais, busca-se soluções que (i) apresentem pequeno atraso de propagação, (ii) são energeticamente eficientes e que (iii) apresentem boa qualidade funcional sem aumento de complexidade. Como pode ser necessário aumentar a complexidade do circuito para reduzir erros o *delay*, a eficiência energética e os custos provavelmente seriam degradados, logo essas grandezas são conflitantes (HRBACEK; MRAZEK; VASICEK, 2016; BABU; BALAJI, 2016; VASICEK; SEKANINA, 2014a; KULKARNI; GUPTA; ERCEGOVAC, 2011; SEKANINA; VASICEK, 2013; KAUFMANN; KNIEPER; PLATZNER, 2010). Consequentemente uma variedade de compensações entre erro, *delay* e a eficiência energética pode ser encontrada por meio da otimização multiobjetivo possibilitando, assim, a concepção de uma vasta quantidade circuitos digitais aproximados.

4.1 FORMULAÇÃO DE OTIMIZAÇÃO MULTIOBJETIVO

De forma geral, um problema de otimização pode ser definido como (NAKAYAMA; YUN; YOON, 2009):

$$Min/Max \quad f_t(\mathbf{x}), \qquad t = 1, 2, \ldots, T; \tag{4.1}$$

$$sujeito \quad a \quad g_j(\mathbf{x}) \geq 0, \qquad j = 1, 2, \ldots, J; \tag{4.2}$$

$$h_k(\mathbf{x}) = 0, \qquad k = 1, 2, \ldots, K; \tag{4.3}$$

$$x_i^I \leq x_i \leq x_i^S, \qquad i = 1, 2, \ldots, N. \tag{4.4}$$

O vetor $\mathbf{x}$ representa as variáveis de decisão, $f_t : \mathbb{R}^N \to \mathbb{R}$ são as funções objetivo, $g_j : \mathbb{R}^N \to \mathbb{R}$ e $h_k : \mathbb{R}^N \to \mathbb{R}$ são respectivamente as restrições de desigualdade e igualdade que devem ser respeitadas. As soluções são representadas por vetores $\mathbf{x} = (x_1, x_2, \ldots, x_N)$. Cada componente do vetor $\mathbf{x}$ é denominado variável de decisão. As restrições em (4.4) representam os intervalos válidos para os valores das variáveis de decisão, de tal maneira que cada variável de decisão pode assumir valores entre o limite inferior x_i^I e superior x_i^S. Os limites inferior (x_i^I) e superior (x_i^S) definem o espaço das variáveis de decisão $D \subseteq \mathbb{R}^N$.

Quando os conjuntos de restrições não são considerados ou não existem, então o problema de otimização é classificado como problema de otimização sem restrições. Contudo, grande parte dos problemas de otimização apresentam restrições, principalmente quando se considera a modelagem de problemas reais. Em problemas de otimização com restrições, as soluções que não violam as restrições são classificadas como soluções viáveis ou factíveis, já as soluções que violam ao menos uma das restrições são classificadas como soluções inviáveis ou infactíveis. O conjunto formado por todas as soluções viáveis define o espaço das soluções factíveis $S \subseteq D$. Cada uma das soluções $x \in D$ possui um ponto z pertencente ao espaço dos objetivos $\mathbb{O}$. O mapeamento de x em z ocorre da seguinte forma, $z = (f_1(x), f_2(x), \ldots, f_M(x))$. A Figura 16 ilustra o processo de mapeamento entre os espaços de decisão e o de objetivos para um problema com três variáveis de decisão e duas funções objetivo.

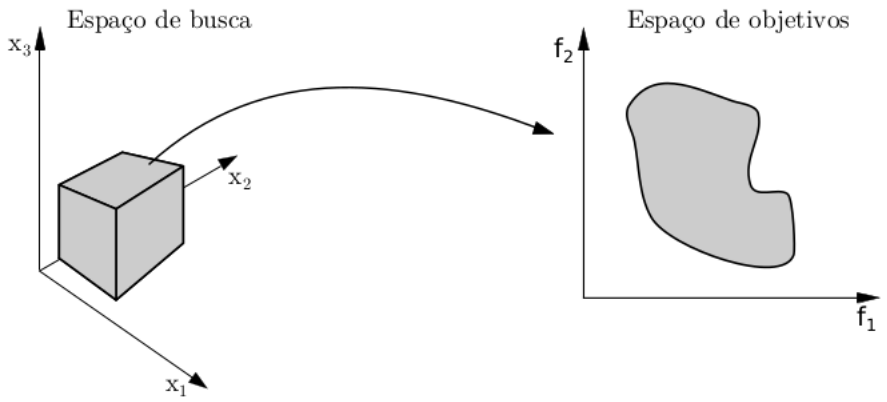

Figura 16 – Mapeamento entre os espaços de decisão e objetivos. Extraído de (RUSSO, 2017).

Ao resolver um problema mono-objetivo, busca-se uma solução que apresente o melhor valor possível (ótimo) para um função objetivo respeitando-se as restrições. Porém, em cenários em que múltiplos objetivos devem ser analisados, o conceito de otimalidade de uma solução não é trivial, visto que os objetivos são conflitantes (DEB, 2001). Portanto é necessário um critério, diferente do critério da otimalidade de uma solução, que possibilite comparar soluções de um problema multiobjetivo. Neste sentido, a dominância de Pareto é bastante difundida como método de comparação de soluções (DEB, 2001). Considerando

problemas de minimização constituídos por T objetivos, uma solução $\mathbf{X_1}$ domina uma solução $\mathbf{X_2}$ se (DEB, 2001):

- $\mathbf{X_1}$ não é pior que $\mathbf{X_2}$ em nenhum dos objetivos, ou seja, $f_t(\mathbf{X_1}) \leq f_t(\mathbf{X_2})$, tal que $t = 1, 2, \ldots, T$;
- $\mathbf{X_1}$ é estritamente melhor que $\mathbf{X_2}$ em pelo menos um dos objetivos, ou seja, $f_T(\mathbf{X_1}) < f_T(\mathbf{X_2})$, tal que $t = 1, 2, \ldots, T$.

Portanto podem ocorrer três situações:

1. $\mathbf{X_1} \prec \mathbf{X_2}$, isto é, $\mathbf{X_1}$ domina $\mathbf{X_2}$;
2. $\mathbf{X_2} \prec \mathbf{X_1}$, isto é, $\mathbf{X_2}$ domina $\mathbf{X_1}$;
3. $\mathbf{X_1}$ e $\mathbf{X_2}$ são soluções não-dominadas entre si.

Neste cenário, as soluções factíveis e não dominadas são chamadas de Soluções Pareto-ótimas ou Conjunto Pareto-Ótimo, já Frente de Pareto-Ótima é o nome dado à imagem do Conjunto Pareto-Ótimo no espaço de objetivos do problema (COELLO; LAMONT; VELDHUIZEN, 2007).

De acordo com Deb (DEB, 2001) a convergência do processo de busca e a diversidade das soluções são metas de um processo de otimização multiobjetivo. Em relação à convergência, busca-se encontrar um conjunto de soluções não-dominadas tão próximas quanto possível da Frente de Pareto-Ótima. No que diz respeito à diversidade das soluções, busca-se encontrar um conjunto de soluções não-dominadas que seja o mais diverso possível. A Figura 17 ilustra como as questões de diversidade e convergência se comportam no Espaço dos Objetivos, neste cenário f_1 e f_2 são funções objetivo conflitantes.

Os métodos clássicos de otimização multiobjetivo transformam os múltiplos objetivos em um único objetivo e, então, o problema é resolvido por intermédio de uma técnica de otimização mono-objetivo (DEB, 2001). No método da soma ponderada (ZADEH, 1963), a função objetivo é definida como a soma ponderada dos múltiplos objetivos, ou seja, cada objetivo é multiplicado por um peso definido pelo usuário. Outra estratégia para lidar com problemas de otimização multiobjetivo consiste na escolha de um dos objetivos como função objetivo, neste caso os demais objetivos são definidos como restrições do problema de otimização (HAIMES; LASDON; WISMER, 1971).

Segundo Deb (DEB, 2001), os métodos clássicos apresentam pouca diversidade de soluções, já que o problema é tratado como um processo de otimização mono-objetivo. Logo, para se obter uma gama de soluções diversas é necessário que os métodos sejam executados diversas vezes. Ademais, os métodos clássicos necessitam de informações adicionais como a definição do objetivo que será priorizado durante a busca.

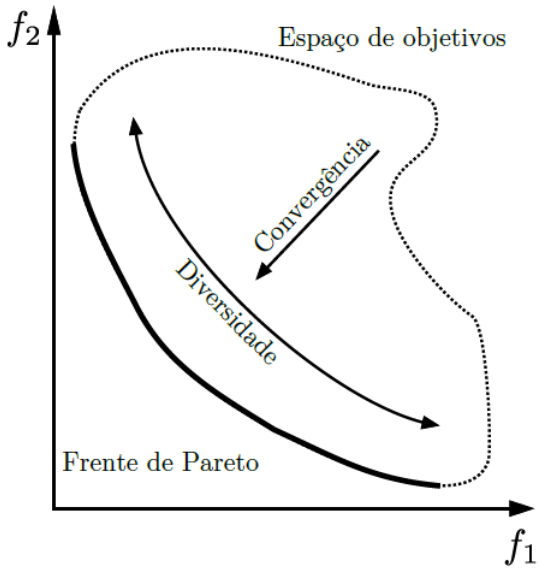

Figura 17 – Convergência e diversidade ilustradas num problema com dois objetivos, f_1 e f_2. Adaptado de (RUSSO, 2017).

Objetivando vencer as limitações impostas pelos métodos clássicos, os algorítimos evolucionistas, que utilizam um conjunto de soluções candidatas, passaram a ser adotados na resolução de problemas de otimização multiobjetivo (COELLO; LAMONT; VELDHUIZEN, 2007), uma vez que podem ser modificados para garantir diversidade de soluções e ao mesmo tempo convergir para a Frente de Pareto (DEB, 2001). Neste cenário, os algorítimos evolucionistas utilizados em processos de otimização multiobjetivo são classificados como Algoritmos Evolucionistas Multiobjetivos (MOEAs).

A maioria dos MOEAs encontrados na literatura classifica as soluções candidatas de um problema de otimização multiobjetivo por meio da Ordenação de Soluções em Frentes de Pareto (DEB, 2001). Neste contexto, os indivíduos são distribuídos em frentes de não dominância $(F_1, F_2 \ldots F_n)$. As soluções que não são dominadas compõem a primeira frente de não dominância, F_1. Já as soluções que são dominadas apenas pelas soluções presentes na primeira frente de não dominância, F_1, formam a segunda frente de não dominância, F_2. A terceira frente de não dominância, F_3, é constituída pelas soluções que são dominadas apenas pelos indivíduos da primeira e segunda frente de não dominância, e assim sucessivamente até que todos os indivíduos da população tenham sido classificados em uma frente de não dominância. A Figura 18 exemplifica a Ordenação de Soluções em Frentes de Pareto para um problema com dois objetivos. Neste cenário, f_1 e f_2 são funções objetivo conflitantes.

4.2 *NON-DOMINATED SORTING GENETIC ALGORITHM II*

Proposto por Deb (DEB et al., 2002), o algoritmo evolucionista multiobjetivo *Non-dominated Sorting Genetic Algorithm II* é um dos MOEAs mais utilizado em problemas

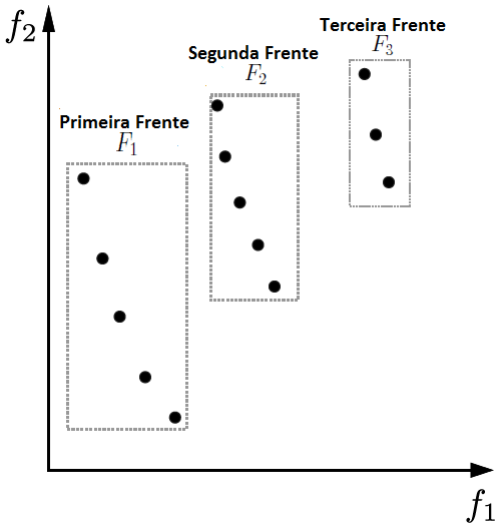

Figura 18 – Ordenação de Soluções em Frentes de Pareto. F_1 é a primeira frente de não dominância, F_1 é a segunda frente de não dominância e F_3 é a terceira frente de não dominância. Adaptado de (RUSSO, 2017).

de otimização multiobjetivo, pois introduziu um algoritmo de baixo custo computacional quando comparado com a primeira versão do método (Srinivas; Deb, 1994) para ordenar as soluções em frentes de não dominância (Algoritmo 3) e uma estratégia para promoção de diversidade populacional (Algoritmo 4). Estudos voltados para o projeto de circuitos digitais aproximados (KAUFMANN; KNIEPER; PLATZNER, 2010; HRBACEK; MRAZEK; VASICEK, 2016; PETRLIK; SEKANINA, 2013; HILDER; WALKER; TYRRELL, 2010) adotam o método de ordenação de soluções e o método de diversidade populacional propostos pelo NSGA-II. O pseudo-código do NSGA-II é mostrado no Algoritmo 2. Inicialmente, uma população de pais P_0, de tamanho N, é criada de forma aleatória. Em seguida, calcula-se a aptidão e ordena-se os indivíduos de P_0 de acordo com o valor de aptidão atribuído à cada indivíduo. Essa aptidão é calculada de acordo com o nível de não dominância e distância de cada solução em relação às outras soluções do mesmo nível de não dominância. Operadores de seleção, recombinação e mutação são adotados para gerar uma população de descendentes Q_0 de tamanho N (DEB et al., 2002). Em seguinda, enquanto um critério de parada não é atendido, combina-se as populações de pais (P_t) e descendentes Q_t, ambas de tamanho N, para gerar a população combinada $R_t = P_t \cup Q_t$ de tamanho $2N$. Neste contexto, os indivíduos de R_t são ordenados em frentes de não dominância por meio do método *Fast non Dominated sort* descrito na Seção 4.2.1, ou seja, as soluções não dominadas constituem a primeira frente de não dominância (F_1), ao passo que a segunda frente de não dominância (F_2) é formada pelos indivíduos que são dominados apenas pelas soluções presentes na primeira frente de não dominância, e assim sucessivamente.

Gera-se uma nova população de pais $P_{t+1} = \emptyset$ e defini-se um contador $i = 1$ que

refere-se a i-ésima frente de não dominância. Enquanto $|P_{t+1}| + |F_i| \leq N$ calcula-se o valor de *Crowding distance* de cada solução presente na frente F_i por meio do método *Crowding distance* descrito na Seção 4.2.2, renova-se $P_{t+1} = P_{t+1} \cup F_i$ e incrementa-se $i = i + 1$.

Para garantir que exatamente $N/2$ indivíduos façam parte da população P_{t+1}, as soluções da última frente (F_i) inseridas em P_{t+1} são ordenadas usando o operador *Crowded Comparison Operator* ($\prec_n$) que é definido na Seção 4.2.3. Neste cenário, escolhe-se os primeiros $N - |P_{t+1}|$ indivíduos da frente F_i para fazer parte da população de pais P_{t+1}. A população P_{t+1} é utilizada para gerar a população de descendentes Q_{t+1} por meio de operadores genéticos (e.g. mutação, recombinação).

Algoritmo 2: NSGA-II. Adaptado de (DEB et al., 2002).

```
1  t = 0;
2  Inicializa-População(P_0);
3  Calcula-aptidão((P_0);
4  Q_o = Cria-descendentes(P_0);
5  while critério de parada não for atendido do
6  |  R_t = P_t ∪ Q_t;
7  |  F = Fast non dominated sort(R_t);
8  |  P_{t+1} = ∅;
9  |  i = 1;
10 |  while |P_{t+1}| + |F_i| ≤ N do
11 |  |  Crowding-distance(F_i);
12 |  |  P_{t+1} = P_{t+1} ∪ F_i;
13 |  |  i = i + 1;
14 |  end
15 |  Crowded comparison operator sort(F_i, ≺_n);
16 |  K = N − |P_{t+1}|;
17 |  P_{t+1} = P_{t+1} ∪ Escolha Individuos Ultima Frente(F_i, K);
18 |  Q_{t+1} = cria-descendentes(P_{t+1});
19 |  t = t + 1;
20 end
```

4.2.1 Ordenação por não dominância

Para cada indivíduo p presente em P, dois parâmetros são calculados:

- n_p que representa o numero de indivíduos que dominam p.
- O conjunto S_p é formado por todos os indivíduos que são dominados por p.

Todos os indivíduos de P são comparados uns com os outros a fim de classificá-los de acordo com graus de dominância n_p. Portanto, se um indivíduo p pertencente a P tem $n_p = y$ significa dizer que a solução p é dominada por y soluções. Todas as soluções que

compõem a primeira frente de não dominância têm seu contador de dominância n_p igual a zero.

Cada solução p tal que $n_p = 0$, tem cada membro, q, de seu respectivo conjunto, S_p, analisado. O contador de dominância das soluções $q \in S_p$ é reduzido um por um. Ao realizar o processo de decremento do contador de dominância, se para qualquer q pertencente a S_p a contagem de dominação se tornar zero este elemento é adicionado a um outro conjunto de soluções. Tal conjunto de soluções forma a segunda frente de não dominância. Para cada membro da segunda frente de não dominância o processo é repetido até que a terceira frente seja identificada. O processo continua até que todas as frentes sejam construídas. O Algoritmo 3 ilustra o procedimento de ordenação em frentes de não dominância.

Algoritmo 3: *Fast non dominated sort.* Adaptado de (DEB et al., 2002).

input : População P
output : População ordenada em frentes de não dominância

```
for Cada p ∈ P do
    S_o = ∅;
    n_p = 0;
    for Cada q ∈ P do
        if p ≺ q then
            S_p = S_p ∪ {q};
        else if q ≺ p then
            n_p = n_p + 1;
    end
    if n_p = 0 then
        p_rank = 1;
        F_1 = F_1 ∪ {p};
end
i = 1;
while F_i ≠ ∅ do
    Q = ∅;
    for Cada p ∈ F_i do
        for Cada q ∈ S_p do
            n_q = n_q − 1;
            if n_q = 0 then
                q_rank = i + 1;
                Q = Q ∪ {q ;
            end
        end
    end
    i = i + 1;
    F_i = Q;
end
```

4.2.2 *Crowding-distance*

O *Crowding-distance* é um valor utilizado para estimar a densidade e garantir a diversidade das soluções quando o algoritmo NSGA-II é adotado. Neste contexto, visa-se garantir o espalhamento das soluções ao longo da Frente de Pareto, consequentemente a concentração de soluções sobre uma única região é reduzida.

O algoritmo para determinar o *Crowding-distance* calcula a distância entre uma solução específica e duas soluções vizinhas em relação a cada objetivo. O *Crowding-distance* das soluções que apresentam o maior valor de aptidão para cada um das funções objetivos, ou seja, aquelas soluções que ocupam os extremos da Frente de Pareto é igual a infinito, indicando, portanto, que estas soluções devem ser priorizadas em detrimento das soluções intermediárias.

Um algoritmo para calcular o *crowding-distance* pode iniciar recebendo a frente F_i de não dominância e o número T de objetivos analisados como parâmetros. A variável L guarda o número de elementos presentes em F_i. Para cada solução $i \in F_i$ atribui-se zero como valor de *Crowding-distance*. Ordena-se os os indivíduos da frente F_i de acordo cada um dos T objetivos, atribui-se *Crowding-distance* infinito às soluções presentes nos extremos da frente F_i, garantindo, portanto, que os pontos extremos sejam sempre selecionados. Em seguida, incrementa-se o *Crowding-distance* das demais soluções utilizando os valores normalizados dos objetivos das soluções adjacentes, garantido que a influência de objetivos com diferentes ordens de magnitude seja a mesma. O Algoritmo 4 ilustra o procedimento *Crowding-distance*.

Algoritmo 4: *Crowding-distance*. Adaptado de (DEB et al., 2002)

input : Frente de Pareto F_i, T: número de objetivos
output : *Crowding-distance* de cada solução

$L = |F_i|$;
for $i \leftarrow 1$ **to** L **do**
  $F_i^{Crowding\text{-}distance} = 0$;
end
for *Cada objetivo* T **do**
  $F_i = Ordenar(F_i, T)$;
  $F_1^{Crowding\text{-}distance} = F_L^{Crowding\text{-}distance} = \infty$;
  for $i \leftarrow 2$ **to** $(L-1)$ **do**
    $F_i^{Crowding\text{-}distance} = F_i^{Crowding\text{-}distance} + (F_{[i+1]}^T - F_{[i-1]}^T)/(f_T^{max} - f_T^{min})$
  end
end

4.2.3 *Crowded-Comparison Operator*

Após a classificação das soluções em frentes de não dominância e atribuição de valores de *Crowding-distance*, a seleção dos indivíduos ocorre por meio do *Crowded Comparison Operator* ($\prec_n$). Cada indivíduo na população tem dois atributos:

1. Grau de não dominância (i_{rank}) (e.i. indivíduos da frente F_i têm grau de não dominância $i_{rank} = i$).
2. *Crowding-distance* (i_{dist}).

Defini-se que $i \prec_n j$ se:

- $i_{rank} < j_{rank}$, ou
- se $i_{rank} = j_{rank}$ e $i_{dist} > j_{dist}$.

Neste contexto, entre duas soluções que apresentam diferentes graus de não dominância, prefere-se a solução com grau de não dominância menor. Já em situações em que o grau de não dominância das soluções é o mesmo, ou seja, pertencem à mesma frente, escolhe-se a solução que está localizada em uma região menos populosa, ou seja, a que possui maior valor de *Crowding-distance*.

4.3 HIPERVOLUME

Uma das métricas utilizadas para avaliar o desempenho de um MOEA é o indicador de hipervolume (ZITZLER; DEB; THIELE, 2000). O hipervolume faz uso das soluções não dominadas encontradas pelo algoritmo de busca e um ponto de referência A (podendo adotar como referência os piores valores das T funções objetivo) para calcular o volume da região coberta pelas imagens das soluções não dominadas e o ponto de referência.

Quanto maior o valor do hipervolume, melhor o desempenho do algoritmo de busca, já que valores elevados de hipervolume indicam grande espalhamento das soluções e melhor convergência do algoritmo (WHILE; BRADSTREET; BARONE, 2012). A Figura 19 apresenta um conjunto de soluções não dominadas de um problema com dois objetivos. Neste cenário ilustrativo, a região sombreada representa a região sobre a qual o hipervolume será calculado.

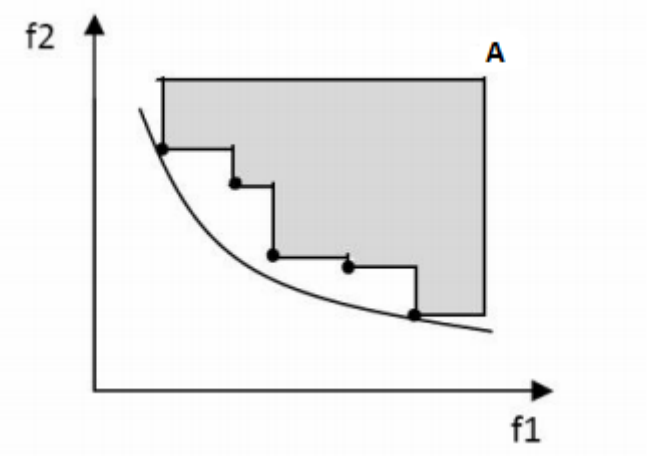

Figura 19 – Exemplo de cálculo do hipervolume para um conjunto de 5 soluções não dominadas em um problema de otimização com dois objetivos conflitantes. A é o ponto de referência. Figura baseada em (DEB, 2001; ZITZLER; DEB; THIELE, 2000).

5 MÉTODO PROPOSTO

Pretende-se, por meio deste trabalho, projetar circuitos digitais aproximados combinacionais por intermédio da Programação Genética Cartesiana considerando múltiplos objetivos. Neste contexto, a maioria dos algoritmos genéticos utilizados para lidar com problemas de otimização multiobjetivo, como o NSGA-II (DEB et al., 2002) e SPEA2 (ZITZLER; LAUMANNS; THIELE, 2001), exigem que se defina e se mantenha uma população com tamanho fixo no decorrer do processo evolutivo. Neste cenário, quanto maior é o tamanho da população uma maior quantidade de compensações entre os diversos objetivos analisados pode ser encontrada, visto que populações compostas por muitos indivíduos permitem uma cobertura maior do espaço de busca. Com isso, a probabilidade de convergência precipitada para ótimos locais é reduzida. Porém, o custo computacional para cada iteração do método associado à necessidade de se manter essa população pode ser significativo. Em contrapartida, uma população pequena permite que se cubra apenas uma pequena parte do espaço de busca, o que pode acarretar na não exploração de regiões promissoras do espaço de busca. Como consequência, a probabilidade de que o algoritmo fique preso em ótimos locais aumenta (Roeva; Fidanova; Paprzycki, 2013).

Neste sentido, dentro do contexto do projeto evolutivo de circuitos digitais combinacionais, pode-se pensar em estratégias para aproveitar de maneira mais adequada, ao longo das diversas gerações, o orçamento computacional associado ao tamanho populacional. Uma alternativa é tornar o tamanho da população adaptativo.

Para isso, a quantidade de indivíduos presentes em uma determinada população pode variar entre uma quantidade máxima e uma quantidade mínima de indivíduos ao longo das gerações. A Figura 20 ilustra o comportamento do tamanho da população em um cenário em que o tamanho populacional é adaptativo, onde Tam_Max representa o maior número de indivíduos permitido, Tam_Min representa o limite inferior do número de indivíduos, e Tam_Fix representa uma população de tamanho fixo.

Dada a possibilidade de se distribuir o orçamento computacional relacionado ao tamanho populacional ao longo das gerações, a ampla utilização da CGP no contexto do projeto evolutivo de circuitos digitais e as diversas restrições impostas pela tabela verdade, propõe-se aqui um método de busca para projetar circuitos digitais combinacionais aproximados denominado Programação Genética Cartesiana Multiobjetivo Com Tamanho Populacional Adaptativo (CGPMO+TPA) que:

- deve lidar com T funções objetivo que devem ser simultaneamente minimizadas,
- utiliza uma estratégia de adaptação do tamanho populacional,
- utiliza o operador de mutação pontual (descrito na Seção 3.2) para gerar descendentes, e

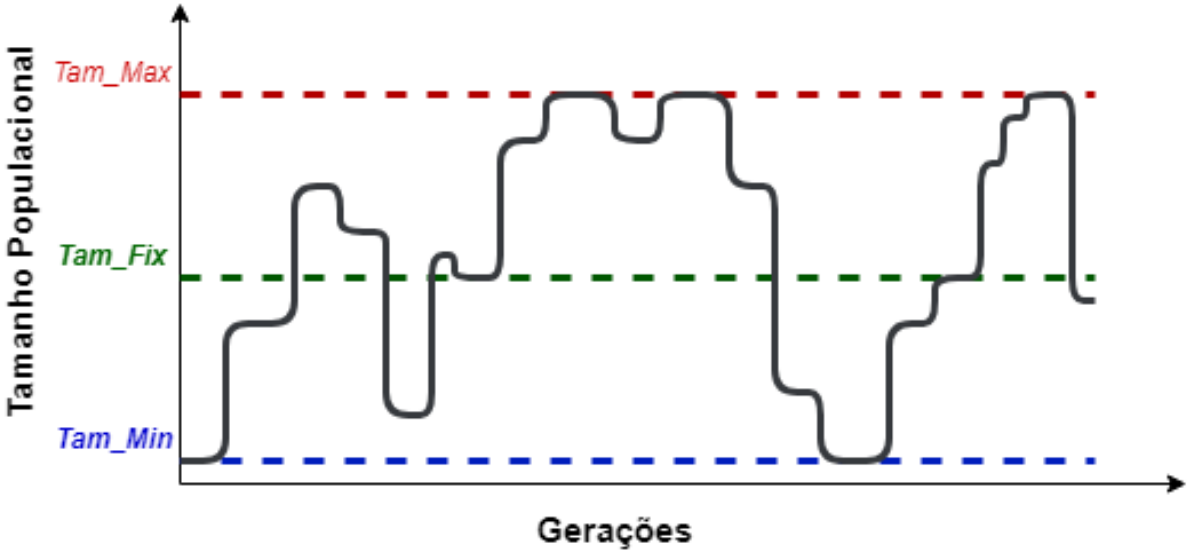

Figura 20 – Variação do tamanho da população ao longo das gerações.

- é baseando na CGP (descrita na Seção 3) e nos conceitos da dominância de Pareto (descritos na Seção 4.1).

A avaliação da funcionalidade de um circuito digital combinacional acontece mediante a analise da tabela verdade. Neste cenário, todas as linhas da tabela verdade devem ser:

- totalmente atendidas, no caso do projeto de circuitos digitais combinacionais totalmente funcionais, ou
- parcialmente atendidas, no caso do projeto de circuitos digitais combinacionais aproximados.

Inicialmente será apresentada a estratégia para lidar com circuitos aproximados. Em seguida, o método de tratamento das restrições será descrito.

5.1 DESCRIÇÃO GERAL DO MÉTODO CGPMO+TPA

Inicializa-se a população de pais (P_t) com um único indivíduo gerado de forma aleatória. No primeiro momento a população de descendentes (Q_t) não contem nenhum indivíduo. Enquanto os critérios de parada não são alcançados, os passos que seguem são executados. Primeiro, as populações de pais e descendentes são combinadas em uma única população $Pop_t = P_t \cup Q_t$ de tamanho adaptativo, isto é, o tamanho da população Pop_t não é fixo, podendo variar de uma geração para outra. Em seguida, os indivíduos não

dominados, F_1, presentes em Pop_t, são selecionadas para compor a população de pais P_t. Logo, $P_t = F_1$.

O número de indivíduos que compõe a população de pais P_t é igual a $\beta = |F_1|$, em que $|\cdot|$ é o operador de cardinalidade de conjunto.

Neste ponto, se o tamanho da população (β) de pais (P_t) é maior do que um certo limite Tam_Max, a abordagem *Crowding-distance* (Algoritmo 4) adotada no NSGA-II é utilizada para evitar que o tamanho da população aumente consideravelmente. Em seguida, as soluções são ordenadas de acordo com o *Crowded-Comparison Operator* (seção 4.2.3) e, consequentemente, as Tam_Max primeiras soluções passam a integrar a população de pais P_t, enquanto os $|P_t| - Tam\_Max$ últimos indivíduos de P_t são descartados. Finalmente, atualiza-se o valor de $\beta = Tam\_Max$. Esta estratégia garante o espalhamento das soluções ao longo da Frente de Pareto, uma vez que os indivíduos com maior valor de *Crowding-distance* são selecionados.

Finalmente, aplicando-se o operador de mutação pontual, descrito na sessão 3.2, cada um dos β indivíduos presentes na população de pais P_t, gera outros β descendentes. O Algoritmo 5 ilustra o procedimento de busca.

Algoritmo 5: CGPMO+TPA

```
1  t = 0;
2  P_t ← Inicializa população(P_t);
3  Q_t ← ∅;
4  while Critérios de parada não são alcançados do
5  |  Pop_t ← P_t ∪ Q_t;
6  |  F_1 ← Indivíduos não dominados(Pop_t);
7  |  β ← |F_1|;
8  |  P_t ← F_1;
9  |  if β > Tam_Max then
10 |  |  Crowding distance(P_t);
11 |  |  Crowded comparison operator sort(P_t, ≺_n);
12 |  |  P_t ← P_t[1 : Tam_Max];
13 |  |  β ← Tam_Max;
14 |  end
15 |  for i ← 1 até β do
16 |  |  Q_i ← Mutação(P_i);
17 |  end
18 |  t = t + 1;
19 end
```

5.2 TRATAMENTO DAS RESTRIÇÕES

O número de linhas da tabela verdade, isto é, a quantidade de restrições que devem ser atendidas, está diretamente relacionada ao número de entradas binárias de um

determinado circuito digital combinacional. Por exemplo, se um circuitos digital possui três entradas binárias, o número de restrições (linhas da tabela verdade) é definido como $2^3 = 8$. Por outro lado, tendo em vista um circuito digital combinacional com quatro entradas binárias, o número de linhas da tabela verdade é definido como $2^4 = 16$. De forma geral o número de linhas de uma tabela verdade é definido como:

$$N_{linhas} = 2^w, \tag{5.1}$$

em que w representa a quantidade de entradas binárias de um determinado circuito. Portanto, é necessário que se avalie uma grande quantidade de restrições impostas pela tabela verdade ao se projetar circuitos digitais combinacionais que apresentem uma grande quantidade de entradas binárias.

Para lidar com problemas que apresentam restrições no cenário de otimização multiobjetivo, o método proposto utiliza a abordagem *Constrained NSGA-II* proposta por Deb et al. (DEB et al., 2002). Na presença de restrições, ocorre uma modificação na definição de *dominância* entre duas soluções. Neste contexto, diz-se que uma solução $\mathbf{X_1}$ domina uma solução $\mathbf{X_2}$, se alguma das seguintes condições for verdadeira:

1. A solução $\mathbf{X_1}$ é factível e a solução $\mathbf{X_2}$ não é.
2. As soluções $\mathbf{X_1}$ e $\mathbf{X_2}$ são ambas infactíveis, mas a solução $\mathbf{X_1}$ viola menos as restrições.
3. As soluções $\mathbf{X_1}$ e $\mathbf{X_2}$ são factíveis e a solução $\mathbf{X_1}$ domina a solução $\mathbf{X_2}$.

6 EXPERIMENTOS COMPUTACIONAIS

Os experimentos computacionais que visam avaliar comparativamente o desempenho do método CGPMO+TPA são descritos neste capítulo. Os circuitos adotados foram somador e multiplicador de 2 e 8 bits, além da Unidade Lógica Aritmética. O método CGPMO+TPA foi comparado com uma abordagem baseada na CGP e NSGA-II proposta por Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016) e a estratégia hES proposta por Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010). Os experimentos foram divididos em três cenários:

1. Otimização multiobjetivo com restrições sem considerar informações do especialista do domínio.

2. Projeto aproximado de circuitos digitais combinacionais via otimização multiobjetivo sem considerar informações do especialista do domínio.

3. Projeto aproximado de circuitos digitais combinacionais via otimização multiobjetivo considerando informações do especialista do domínio.

6.1 DESCRIÇÃO DOS PROBLEMAS

Os circuitos alvo adotados são somadores de 2 (S2) e 8 bits (S8) e multiplicadores de 2 (M2) e 8 bits (M8), arquiteturas constituintes de estruturas de filtragem digital como filtros digitais de resposta finita ao impulso (FIR) e filtros digitais de resposta infinita ao impulso (IIR) (OPPENHEIM; WILLSKY; NAWAB, 2010). Estes filtros são primordiais em aplicações que envolvem processamento digital sinais como em processamento de imagens, processamento de vídeo e em sistemas de comunicação sem fio (TSAO; CHOI, 2012). Além disso, somadores e multiplicadores digitais são constituintes também de circuitos dedicados a realizar cálculo de funções hiperbólicas e trigonométricas, como o *Coordinate Rotation Digital Computer* (CORDIC) que é adotado em aplicações de estimativas de fase e frequência de sinais elétricos, sincronização em receptores digitais e em processadores gráficos 3D (AGGARWAL; MEHER; KHARE, 2016). Existem diversas arquiteturas de somadores como *Ripple Carry Adder, Carry Select Adder, Carry Lookahead Adder, multiple Tree Adder, Higher Valency Tree Adder* e diversos circuitos multiplicadores como *Ripple Carry Array Multiplier, multiple Carry Save Array, Wallace Tree* (WESTE; HARRIS, 2010). O *Ripple Carry Adder* e o *Ripple Carry Array Multiplier* foram adotados como arquiteturas de referência para o somador e multiplicador de 2 e 8 bits, respectivamente. Esta escolha se deve à utilização de tais arquiteturas em trabalhos presentes na literatura relacionados ao projeto de circuitos digitais aproximados (VASICEK; SEKANINA, 2014a; VASICEK; SEKANINA, 2015b; HRBACEK; MRAZEK; VASICEK, 2016).

Além dos somadores e multiplicadores, a Unidade Lógica Aritmética (ALU) também é usada aqui, uma vez que tal arquitetura é vital para o processamento digital de sinais e protocolos de comunicação e rede (RAHAL et al., 2018). Além disso, a ALU está presente em todos os dispositivos de computação como microprocessadores, computadores e sistemas embarcados (YADAV; KUMARI, 2017). Portanto, a Unidade Lógica Aritmética tem papel de destaque em todo campo da eletrônica digital.

A arquitetura SN54/74LS181, uma Unidade Lógica de Aritmética, que tem 8 bits de saída e 14 bits de entrada, dos quais 4 bits são reservados para cada uma das variáveis e 6 bits são utilizados para selecionar uma das 16 operações lógicas e outras variedades de operações aritméticas[1] foi adotada neste trabalho como arquitetura referência. A Tabela 6.1 mostra as funcionalidades da ALU adotada.

Tabela 3 – Tabela de funcionalidades em que $A + B$ corresponde a "A ou B" lógico. $A.B$ corresponde a "A e B" lógico. A soma é indicada por "mais", ao passo que a subtração é indicada por "menos".

Funções Lógicas	**Funções Aritméticas**	**Funções Aritméticas**
A	A *menos* 1	$A.B$ *mais* A
$\overline{A.B}$	$A.B$ *menos* 1	$A.\overline{B}$ *mais* A
$\overline{A} + B$	$A.\overline{B}$ *menos*1	$A.B$ *mais* $(A + \overline{B})$ *mais* 1
Um lógico	*menos* 1	A *menos* B
$\overline{A \oplus B}$	A *mais* $(A + \overline{B})$	$(A + \overline{B})$ *mais* 1
$\overline{B}$	$A.B$ *mais* $(A + \overline{B})$	A *mais* $(A + B)$ *mais* 1
$\overline{A + B}$	A *menos* B *menos* 1	A *mais* B *mais* 1
$A + \overline{B}$	Zero	$(A.\overline{B})$ *mais* $(A + B)$ *mais* 1
$\overline{A}.B$	A *mais* $(A + B)$	$(A + B)$ *mais* 1
$A \oplus B$	A *mais* B	A *mais* A *mais* 1
B	$A.\overline{B}$ *mais* $(A + B)$	$(A.B)$ *mais* A *mais* 1
$A + B$	A *mais*$(A + \overline{B})$ *mais* 1	$(A.\overline{B})$ *mais* A *mais* 1
Zero lógico	A *mais* A	A *mais* 1
$A.\overline{B}$	–	–
$A.B$	–	–
$\overline{A}$	–	–

Dados fornecidos por um fabricante de dispositivos digitais são usados para definir os parâmetros do circuito nas Equações 2.9 e 2.15, ou seja, I_{SC}, $I_{leakage}$, V_{dd}, C_L, f_{clk} e t_d. A fabricante Nexperia[2] foi selecionada, pois suas portas lógicas têm valores baixos de t_d (atraso de propagação de uma porta lógica). As portas são: 74LVC1G08 (AND), 74LVC1G08 (NAND), 74LVC1G32 (OR), 74LVC1G86 (XOR), 74LVC1G02 (NOR), HEF4077B (XNOR) e 74AHC1G04 (Inversor).

[1] *Data sheet* disponível em `http://pdf1.alldatasheet.com/datasheet-pdf/view/5671/MOTOROLA/SN54LS181.html`

[2] `https://www.nexperia.com/products/logic/i-o-expansion-logic/`

6.2 COMPARAÇÃO ENTRE hES e CGPMO+TPA

A fim de comparar o desempenho do método CGPMO+TPA com a estrategia hES (KAUFMANN; KNIEPER; PLATZNER, 2010), descrita no Capítulo 4, e sua periodização, foram realizados experimentos computacionais condizidos do mesmo modo que Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010). Neste contexto, somadores e multiplicadores de 2 bits, detalhados na Seção 6.1, foram os problemas analisados e vinte execuções independentes do método CGPMO+TPA foram executadas para cada circuito. As funções utilizados como objetivo durante o processo de evolução foram a estimativa de atraso de propagação,

$$D_c = \max_{\forall p \in caminho} \sum_{c_i \in p} t_d(c_i), \tag{6.1}$$

e estimativa de área ocupada,

$$area(c) = 1 - \frac{blocos\ usados(c)}{m}. \tag{6.2}$$

A distância de Hamming,

$$d_h = \sum_{\forall i} (PB^{(i)}_{factivel} \oplus PB^{(i)}_{ordinario}), \tag{6.3}$$

é adotada como uma estimativa de erro dos circuitos candidatos e é considerada como restrição neste experimento.

A Tabela 4 mostra a configuração do modelo de representação CGP para o somador e multiplicador de 2 bits, em que N_{ni} é o número de entradas e N_{no} é o número de saídas de cada nó.

Ressalta-se que a técnica CGPMO+TPA adota apenas a mutação pontual como forma de gerar descendentes, neste contexto a taxa de mutação pontual μ_r foi definida como 5%.

Tabela 4 – Configuração do modelo de representação CGP utilizado por Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010).

Probabilidade de Mutação	0,1
Probabilidade de Recombinação	0,5
n_r	1
n_c	200
l	200
N_{ni}	2
N_{no}	1

O conjunto Γ de elementos de processamento lógico é apresentado na Tabela 5.

Tabela 5 – Elementos de processamento lógico utilizado por Kaufamnn et al. (KAUFMANN; KNIEPER; PLATZNER, 2010).

Índice	**Função Lógica**	**Índice**	**Função Lógica**
0	0	10	$a \oplus b$
1	1	11	$a \oplus \overline{b}$
2	a	12	$a + b$
3	b	13	$a + \overline{b}$
4	$\overline{a}$	14	$\overline{a} + b$
5	$\overline{b}$	15	$\overline{a} + \overline{b}$
6	$a.b$	16	$a.\overline{c} + b.c$
7	$a.\overline{b}$	17	$a.\overline{c} + \overline{b}.c$
8	$\overline{a}.b$	18	$\overline{a}.\overline{c} + b.c$
9	$\overline{a}.\overline{b}$	19	$\overline{a}.\overline{c} + \overline{b}.c$

O número de entradas (n_i) e saídas primárias (n_o) é definido como três e quatro para o somador e multiplicador, respectivamente. O processo é interrompido após 400.000 avaliações do somador, e após 1.600.000 avaliações do multiplicador.

Para analisar o desempenho do método hES, Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) utilizou o indicador de qualidade de MOEAs *Additive Epsilon Indicator* (AEI) proposto por Zitzler et al. (ZITZLER et al., 2003). Para avaliar estatisticamente as sequências de valores gerados pelo indicador de qualidade, Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) adotou o teste não paramétrico de Kruskal-Wallis (CONOVER, 1999). Neste cenário, os valores de AEI encontrados por Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) não foram disponibilizados, portanto essa dissertação não apresenta uma comparação estatística entre o método CGPMO+TPA, no contexto da evolução de somadores e multiplicadores de 2 bits, a abordagem hES e sua periodização. Contudo, Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) investigou a possibilidade de projetar somadores e multiplicadores de 2 bits com erro nulo que apresentassem diversas compensações entre área e *delay*.

A Tabela 6 mostra um comparativo entre os dois melhores resultados obtidos por Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) e a abordagem CGPMO+TPA, neste cenário é analisado o número de execuções independentes que resultaram em soluções funcionalmente corretas, em relação ao erro. Kaufmann et al. (KAUFMANN; KNIEPER; PLATZNER, 2010) não disponibilizou em seu trabalho o número de circuitos corretos encontrados pelas estrategias de busca por ele analisadas.

Tabela 6 – Número de execuções independentes, que resultaram em circuitos totalmente funcionais, dos vários algoritmos para o somador de 2 bits (S2) e multiplicador de 2 bits (M2). hn^{10} diz respeito à periodização do método hES como o NSGA-II, neste caso é realizado um passo/geração da abordagem hES e 10 passos/gerações do NSGA-II.

Método	**S2**	**M2**
hES	12	15
hn^{10}	11	14
$CGPMO + TPA$	**20**	**20**
#circuitos corretos obtidos pela CGPMO+TPA	26	24

Os resultados mostrados pela Tabela 6 indicam que estratégias de busca local periodizadas com otimizadores multiobjetivo são benéficas para aplicações dentro do domínio CGP no contexto da evolução de circuitos digitais. A abordagem CGPMO+TPA mostrou-se eficiente no que diz respeito a encontrar circuitos circuitos totalmente funcionais, uma vez que em todas as 20 execuções independentes foram encontrados circuitos funcionalmente corretos. Os resultados apresentados pela Tabela 6 apontam que a abordagem CGPMO+TPA consegue performar em ambientes de evolução de circuitos digitais, pode-se, então, pensar em adota-la em projetos de circuitos digitais combinacionais aproximados.

6.3 PROJETO DE CIRCUITOS DIGITAIS COMBINACIONAIS APROXIMADOS

A abordagem proposta por Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016) utiliza o NSGA-II para lidar com múltiplos objetivos e a CGP para representação de soluções candidatas, descrita no Capítulo 4 deste trabalho, e rotulada aqui como CGP+NSGA-II foi implementada com o objetivo de ser comparada com o método CGPMO+TPA. O presente conjunto de experimentos utilizou três funções objetivo regularmente empregadas para projetar circuitos digitais aproximados no âmbito do *Hardware* Evolutivo. São elas: distância de Hamming, adotada aqui como uma estimativa de erro dos circuitos candidatos,

$$d_h = \sum_{\forall i} (PB^{(i)}_{factivel} \oplus PB^{(i)}_{ordinario}), \tag{6.4}$$

estimativa de atraso de propagação,

$$D_c = \max_{\forall p \in caminho} \sum_{c_i \in p} t_d(c_i). \tag{6.5}$$

e estimativa de dissipação de potência,

$$P = I_{sc}.V_{dd} + I_{leakage}.V_{dd} + a_{0\to1}.C_L.f_{clk}.V^2_{dd}. \tag{6.6}$$

Os experimentos computacionais foram conduzidos utilizando circuitos somadores e multiplicadores de 8 bits, além da Unidade Lógica Aritmética como problemas a serem resolvidos. Uma descrição mais detalhada desses circuitos foi apresentada na

Seção 6.1. Além disso, as duas técnicas baseadas na CGP utilizaram o mesmo conjunto Γ = {AND, NAND, OR, XOR, NOR, XNOR, INV} de elementos de processamento lógico e as mesmas configurações de parâmetros de representação da CGP, apresentadas pela Tabela 7. Além disso, o tamanho da população adotado para a CGP+NSGA-II é igual a 50.

Os métodos foram analisados considerando duas estratégias para inicialização da população: (i) circuitos iniciais gerados de forma aleatória, e (ii) inserindo uma arquitetura convencional (factível) codificada por meio da CGP. Neste contexto, em relação à técnica CGP+NSGA-II, um dos 50 indivíduos que compõe a população inicial é funcionalmente correto em termos de erro, enquanto que os outros 49 são gerados de forma aleatória. No que diz respeito à abordagem CGPMO+TPA, o único indivíduo da população inicial é funcionalmente correto em termos de erro. Isso foi feito para verificar o desempenho dos métodos na tarefa de projetar circuitos digitais aproximados do zero a nível de portas lógicas e na tarefa de otimizar arquiteturas convencionais, gerando circuitos digitais aproximados como consequência. Quando a população é inicializada de forma aleatória 30×10^6 e 60×10^6 avaliações de circuito foram permitidas. Ao passo que 30×10^6 avaliações de circuito foram permitidas quando a população é inicializada contendo um circuito convencional. Foram realizadas 30 execuções independentes de cada técnica para cada problema.

Tabela 7 – Configuração do modelo de representação CGP utilizados pelos métodos CGPMO+TPA e CGP+NSGA-II.

	Somador 8 bits	**Multiplicador 8** bits	**ALU**
n_r	1	1	1
n_c	200	1000	300
l	200	1000	300
n_i	16	16	14
n_o	9	16	8
N_{ni}	2	2	2
N_{no}	1	1	1
μ_r	5%	5%	5%

As soluções não dominadas encontradas pelos métodos CGPMO+TPA e CGP+NSGA-II ao fim do processo de evolução de cada problema tratado (S8, M8 e ALU) são apresentadas em projeções 2D da dissipação de potência, *delay* e erro. Além disso, assim como Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016), características operacionais de 10 soluções obtidas que apresentam os menores erros são apresentadas em tabelas que evidenciam a diferença entre as soluções não dominadas obtidas pelos métodos analisado.

A métrica adotada para avaliar comparativamente as abordagens CGPMO+TPA e CGP+NSGA-II foi o hipervolume calculado sobre as soluções não dominadas obtidas pelas duas técnicas. O cálculo dos valores de hipervolume foi realizado utilizando a ferramenta

desenvolvida por Fonseca, López-Ibáñez, Paquete e Guerreiro disponibilizada publicamente[3]. O teste não paramétrico de Kruskal-Wallis (CONOVER, 1999) foi executado sobre todos os valores de hipervolume obtidos em cada uma das 30 execuções independentes de cada abordagem relativas a cada circuito alvo para avaliar estatisticamente as abordagens CGPMO+TPA e CGP+NSGA-II. Este teste compara grupos de dados para determinar se as amostras provêm da mesma população, ou seja, se existe diferença estatística entre os dados. O teste foi executado utilizando o Matlab[4] adotando-se um nível de significância $\alpha = 5\%$. Neste caso, se o teste retornar p-valor $< \alpha$ rejeita-se a hipótese de que as medianas dos hipervolumes obtidos pelos métodos são iguais a um nível de significância igual a 5%.

6.3.1 **Inicialização da população de forma aleatória**

Aqui são analisados os resultados obtidos pelas técnicas de busca CGPMO+TPA e CGP+NSGA-II quando a inicialização da população ocorreu de forma integralmente aleatória e 30×10^6 e 60×10^6 avaliações de circuito foram permitidas. Primeiramente, na Seção 6.3.1.1, são discutidos os resultados relativos aos experimentos realizado quando 30×10^6 avaliações de circuito foram permitidas, na Seção 6.3.1.2 são discutidos os resultados obtidos quando 60×10^6 avaliações de circuito foram permitidas.

6.3.1.1 *Projeto de circuitos aproximados quando* 30×10^6 *avaliações de circuito foram permitidas*

A Figura 21 ilustra as projeções 2D das características operacionais (erro, *delay* e potência) das versões não dominadas da ALU obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, ao passo que a Tabela 8 apresenta características operacionais de 10 ALUs que apresentam os menores erros quando 30×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes e inicialização aleatória da população.

As projeções 2D presentes na Figura 21 indicam que as soluções encontradas pelas abordagem CGP+NSGA-II estão mais espalhadas no espaço dos objetivos do que as soluções encontradas pelo método CGPMO+TPA. Neste contexto, o circuito que apresenta o maior valor de *delay* foi encontrado pelo método CGPMO+TPA, ao passo que o circuito que apresenta o maior valor de erro e o circuito que apresenta o maior valor de dissipação de potência foram ambos obtidos pelo método CGP+NSGA-II.

Dentre os circuitos mostrados na Tabela 8, nenhuma das abordagem analisadas foi capaz de projetar circuitos com a mesma funcionalidade, em termos de erro e *delay*, que a arquitetura de referência, porém reduções em dissipação de potência foram atingidas.

[3] `http://lopez-ibanez.eu/hypervolume`

[4] `https://www.mathworks.com/help/stats/kruskalwallis.html`

Ainda em relação aos circuitos mostrados na Tabela 8, as ALUs obtidas pela técnica CGPMO+TPA apresentam erro alto (30% no mínimo) e têm *delay* no mínimo três vezes maior do que a arquitetura de referência, em contrapartida o máximo de potência dissipada corresponde à 51% da dissipação do circuito de referência. Destaca-se que as 10 ALUs obtidas pelo método CGP+NSGA-II que apresentam os menores erros, assim como as obtidas pela técnica CGPMO+TPA, apresentam erro alto ($36,5\%$ no mínimo) e têm *delay* no mínimo $3,5$ vezes maior do que a arquitetura de referência, em compensação o máximo de potência dissipada corresponde a 52% da dissipação do circuito de referência.

Dentre os circuitos apresentados na Tabela 8, aqueles obtidos por meio da abordagem CGPMO+TPA erram menos do que os circuitos encontrados pela abordagem CGP+NSGA-II. Neste cenário, a diferença entre o valor de erro do circuito obtido pela abordagem CGPMO+TPA, dentre os circuitos mostrados pela Tabela 8, que mais erra ($30,79\%$) e o obtido pelo método CGP+NSGA-II que menos erra ($36,64\%$) é de aproximadamente 6%. As 10 ALUs obtidas pela técnica CGPMO+TPA que menos erram dissipam menos potência e têm *delay*, em média, 23% menor do que as 10 ALUs obtidas pela CGP+NSGA-II que menos erram.

Tabela 8 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
29,72	560,81	51,24	36,64	701,80	51,65
29,81	517,57	41,41	37,35	685,14	48,87
29,86	866,22	40,74	37,57	685,14	37,30
29,91	371,62	27,81	37,67	685,14	36,70
30,21	371,17	25,00	37,92	684,23	33,31
30,38	327,93	25,76	38,14	383,33	50,62
30,48	326,13	24,93	38,30	669,37	27,60
30,52	326,13	21,49	38,32	373,87	32,45
30,60	309,91	25,21	38,57	358,11	24,22
30,79	317,12	21,42	38,96	358,11	23,19

A Figura 22 ilustra projeções 2D das funcionalidades (erro, *delay* e potência) das versões não dominadas do somador de 8 bits obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, ao passo que a Tabela 9 apresenta características operacionais dos 10 somadores de 8 bits que apresentam os menores erros quando 30×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes e inicialização aleatória da população.

As projeções 2D ilustradas pela Figura 22 indicam que as soluções encontradas

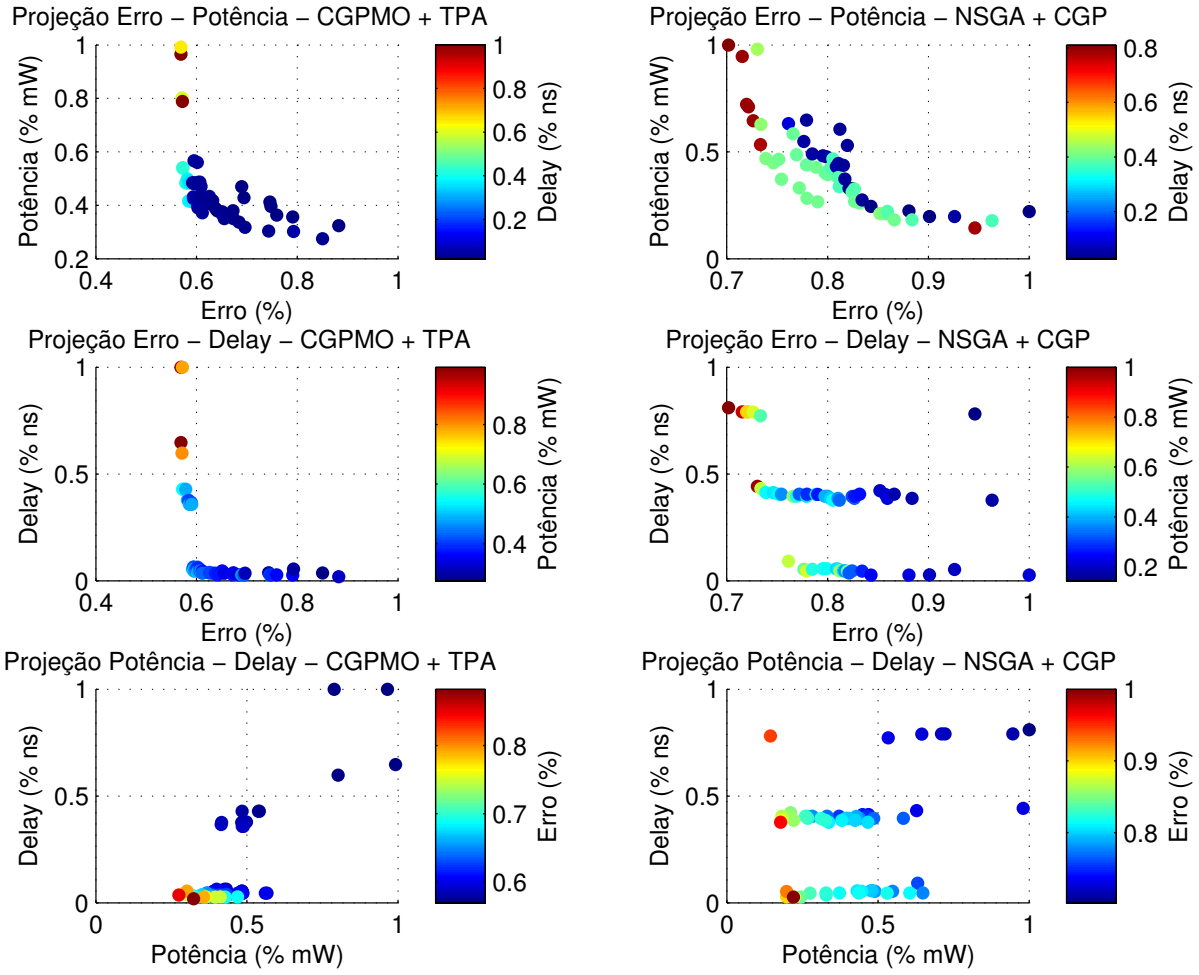

Figura 21 – Projeções 2D da dissipação de potência, *delay* e erro das AULs evoluídas quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

pelas duas abordagens ocupam regiões similares dentro do espaço dos objetivos. Além disso, o somador de 8 bits que apresenta o maior valor de *delay* e o somador de 8 bits que apresenta a maior dissipação de potência foram obtidos pelo método CGP+NSGA-II, ao passo que o circuito com maior valor de erro foi obtido pelo método CGPMO+TPA. Os circuitos apresentados pela Tabela 9 mostram que as técnicas analisadas não obtiveram somadores de 8 bits com erro nulo. As soluções encontradas apresentam erro alto, 14% no mínimo. Além disso, nenhum circuito mostrado pela Tabela 9 apresenta *delay* inferior ao *delay* do circuito de referência. Neste contexto, os valores de *delay* dos 10 somadores de 8 bits que menos erram encontrados pelas duas técnicas são superiores ao *delay* do *Ripple Carry Adder*.

Em relação à dissipação de potência, todos os 10 somadores de 8 bits que menos erram encontrados pelo método CGPMO+TPA e todos os 10 somadores de 8 bits que

menos erram obtidos pelo método CGP+NSGA-II dissipam menos potência do que a arquitetura de referência. Dentre os circuitos dos quais as funcionalidades são apresentas pela Tabela 9, o que dissipa mais potência (79.63% da potência dissipada pela arquitetura de referência) encontrado pela abordagem CGPMO+TPA é superior, em termos de eficiência energética, a todos os 10 circuitos que apresentam os menores valores de erro encontrados pela abordagem CGP+NSGA-II.

Os somadores de 8 bits obtidos pelas técnicas CGPMO+TPA e CGP+NSGA-II que têm suas características operacionais mostradas na Tabela 9 apresentam nível de erro parecido. Neste cenário, o somador de 8 bits obtido pela abordagem CGPMO+TPA mais exato tem erro igual a $15,49\%$, ao passo que o somador de 8 bits obtido pela abordagem CGPMO+TPA mais exato tem erro igual a $13,89\%$. Portanto, a diferença entre os valores de erro é de aproximadamente $1,6\%$. Dentre os 10 circuitos obtidos pela abordagem CGPMO+TPA que menos erram, o menos exato tem erro igual a $17,95\%$. Dentre os 10 circuitos obtidos pela abordagem CGPMO+TPA que menos erram, o menos exato tem erro igual a $16,88\%$. Neste cenário, a diferença entre os valores de erro é de aproximadamente 1%.

Tabela 9 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
15,49	570,82	79,63	13,89	592,22	95,11
15,54	570,82	73,98	14,18	578,21	90,20
16,58	570,82	73,00	14,70	605,45	74,73
16,63	564,20	64,86	14,87	578,21	87,25
17,08	309,73	69,81	15,10	571,60	91,39
17,32	564,20	64,27	15,44	591,05	70,79
17,35	571,21	59,36	15,92	570,82	81,49
17,60	296,50	68,99	16,49	310,12	97,90
17,70	564,20	60,14	16,66	591,44	67,47
17,95	296,50	63,68	16,88	302,72	77,83

A Figura 23 ilustra projeções 2D das funcionalidades (erro, *delay* e potência) das versões não dominadas do multiplicador de 8 bits obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, já a Tabela 10 apresenta características operacionais dos 10 multiplicadores de 8 bits com os menores erros quando 30×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes e inicialização aleatória da população.

As projeções 2D ilustradas pela Figura 23 indicam que as soluções encontradas pelas abordagem CGP+NSGA-II estão mais espalhadas no espaço dos objetivos do que as soluções encontradas pelo método CGPMO+TPA. Além disso, os dois métodos

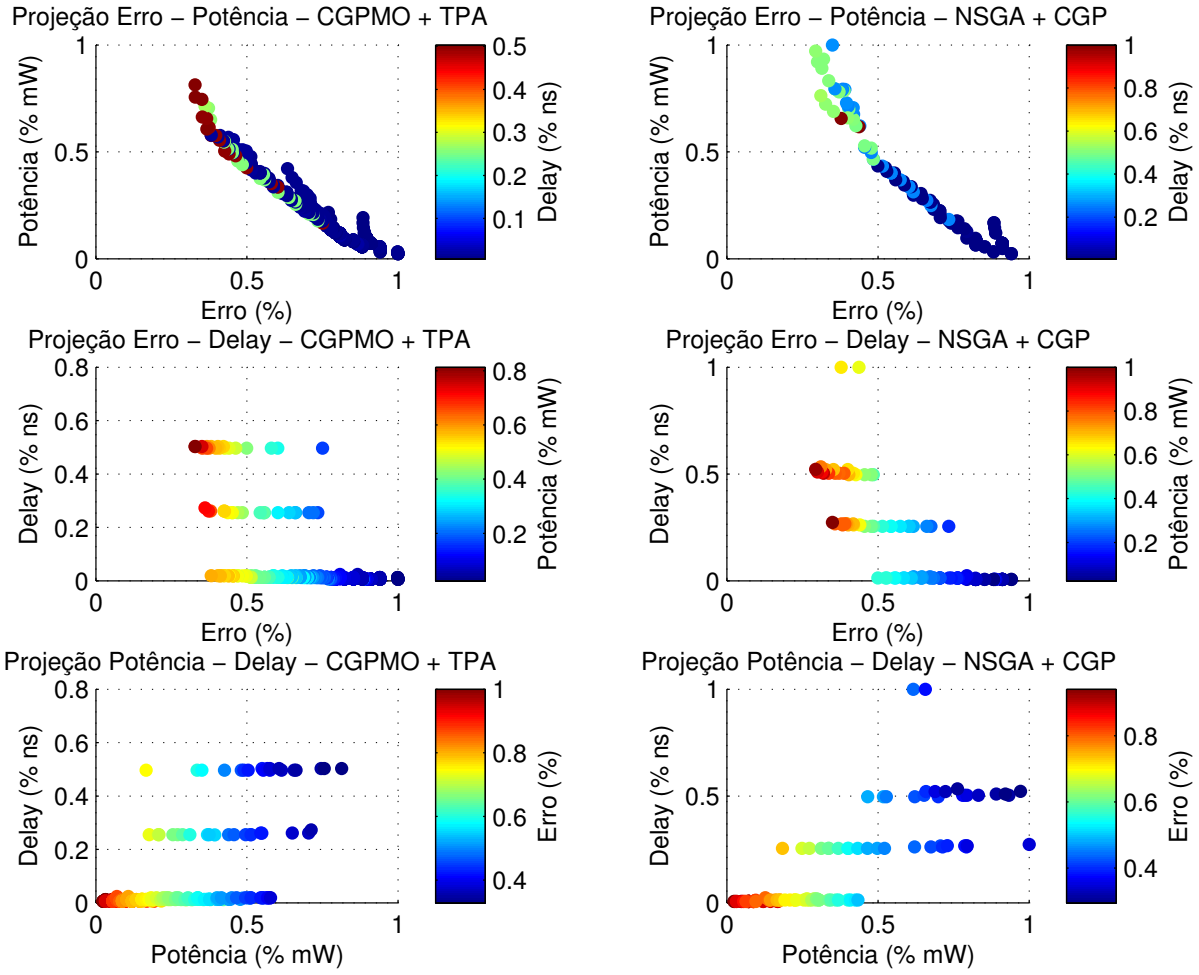

Figura 22 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

encontraram circuitos que têm o mesmo valor máximo de *delay*, ao passo que o circuito com maior valor de erro e o circuito com maior dissipação de potência foram ambos obtidos pelo método CGP+NSGA-II.

Assim como no caso da evolução da ALU e somador de 8 bits as abordagem analisadas não apresentaram como resultado multiplicadores de 8 bits com erro nulo. Dentre os circuitos mostrados pela Tabela 10, os multiplicadores de 8 bits obtidos pela técnica CGPMO+TPA têm nível de erro alto ($38,87\%$ mínimo) e *delay* no mínimo duas vezes superior ao *delay* do *Ripple Carry Array Multiplier*. Contudo, a máxima dissipação de potência, dentre todos os circuitos dos quais as características operacionais são apresentadas na Tabela 10, corresponde a 33% da dissipação do circuito de referência. Destaca-se que os multiplicadores de 8 bits obtidos pelo método CGP+NSGA-II, que têm suas características operacionais apresentadas pela Tabela 10, têm erro alto ($42,14\%$ no mínimo) e apresentam

delay no mínimo 3 vezes maior do que a arquitetura de referência, porém a máxima dissipação de potência corresponde à 46% da dissipação de potência do *Ripple Carry Array Multiplier*.

Os 10 circuitos encontrados pela abordagem CGPMO+TPA que apresentam menor erro, erram menos do que os 10 circuitos encontrados pela abordagem CGP+NSGA-II que menos erram. A diferença entre os valores de erro do multiplicador de 8 bits que mais erra, dentre os mostrados pela Tabela 10, obtido pela abordagem CGPMO+TPA e o multiplicador de 8 bits que menos erra obtido pela técnica CGP+NSGA-II é de aproximadamente 3%.

Dentre os circuitos dos quais as funcionalidades operacionais são apresentadas na Tabela 10, os multiplicadores de 8 bits encentrados pela técnica CGPMO+TPA dissipam menos potência e têm *delay*, em média, $8,5\%$ menor do que os 10 multiplicadores de 8 bits obtidos pela abordagem CGP+NSGA-II que menos erram. Ressalta-se que os 10 circuitos que menos erram obtidos por ambas as abordagens dissipam menos potência do que o *Ripple Carry Array Multiplier*.

Tabela 10 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*.Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
38,87	504,04	52,12	42,14	414,27	64,27
38,92	504,04	32,81	42,36	317,50	56,22
38,97	311,57	40,60	42,39	316,42	62,08
38,98	308,88	38,06	42,79	411,44	47,87
38,99	311,57	33,66	42,87	309,56	81,27
39,02	308,88	33,67	42,91	409,15	49,12
39,23	304,71	37,07	42,96	306,46	68,29
39,25	219,11	47,72	42,99	411,31	39,85
39,27	306,59	36,50	43,09	409,15	46,55
39,28	219,11	46,86	43,17	299,87	69,49

Para avaliar as abordagens CGPMO+TPA e CGP+NSGA-II, foi calculado o hipervolume das soluções não dominadas obtidas pelas duas técnicas. Os pontos de referência adotados foram determinados pelos maiores valores de cada função objetivo considerando as abordagens CGPMO+TPA e CGP+NSGA-II. A Tabela 11 mostra os pontos de referência adotados.

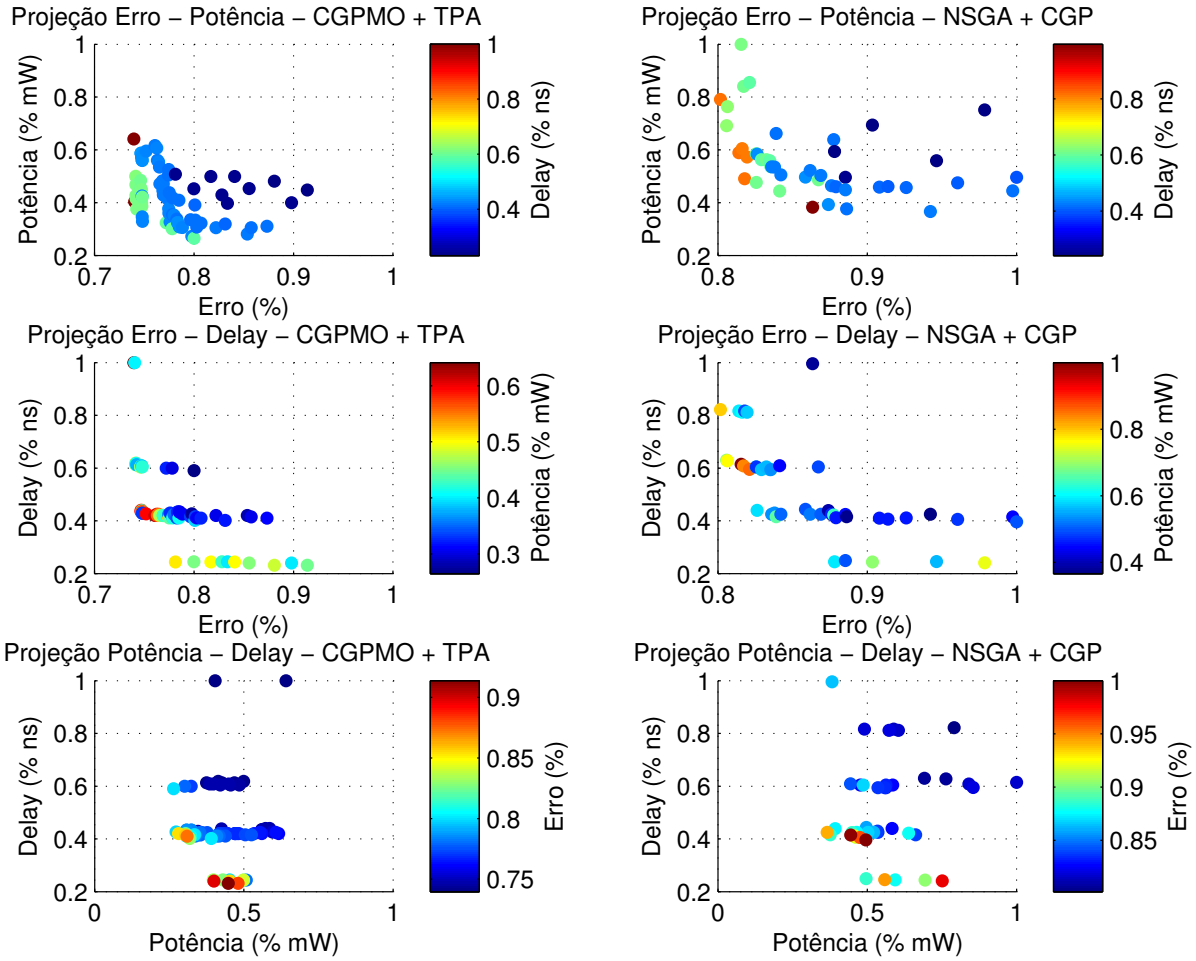

Figura 23 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Tabela 11 – Pontos de referência utilizados no cálculo de hipervolume.

Circuito	**Erro (%)**	***Delay* (ns)**	**Potência (mW)**
ALU	0,5221	192,3000	0,5861
S8	0,4722	291,7000	0,4865
M8	0,5257	374,5000	2,6115

A Tabela 12 mostra os valores de hipervolume relativos aos resultados encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II no que diz respeito à evolução da ALU, somador e multiplicador de 8 bits considerando as 30 execuções independentes, inicialização aleatória da população e quando 30×10^6 avaliações de circuito foram permitidas. Neste cenário, os valores de hipervolume foram calculados sobre os valores normalizados das diferentes funções objetivo em relação aos valores apresentados pela Tabela 11. Os

melhores resultados estão destacados em negrito na Tabela 12, neste contexto a técnica CGPMO+TP foi a que apresentou os maiores valores médios de hipervolume para todos os circuitos analisados. O hipervolume médio obtido pela abordagem CGPMO+TPA é aproximadamente 10%, 14%, 15% maior do que o hipervolume médio obtido pela abordagem CGP+NSGA-II no projeto de ALUs, somadores e multiplicadores de 8 bits, respectivamente.

Tabela 12 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos.

Circuito	Método	Melhor	Mediana	Média	STD	Pior
ALU	CGPMO+TPA	0,3532	0,3049	**0,2965**	0,0438	0,2276
	CGP+NSGA-II	0,3366	0,2720	0,2700	0,0449	0,1464
S8	CGPMO+TPA	0,4059	0,3439	**0,3475**	0,0314	0,2935
	CGP+NSGA-II	0,3770	0,3041	0,3049	0,0456	0,2388
M8	CGPMO+TPA	0,3192	0,1524	**0,1615**	0,0873	0,0218
	CGP+NSGA-II	0,1873	0,1475	0,1406	0,0384	0,0418

Por meio dos resultados do teste de Kruskal-Wallis, apresentados na Tabela 13, observa-se que os p-valores retornados, exceto quando o problema analisado é o multiplicador de 8 bits, são menores que $0,05$. Portanto, para somadores de 8 bits e ALUs, com 5% de chance de erro, conclui-se que as amostras de hipervolueme analisadas não são oriundas da mesma população reforçando a hipótese de que exite diferença significativa entre os dados obtidos pelas abordagem CGPMO+TPA e CGP+NSGA-II. Desta forma, pode-se dizer que os resultados alcançados pela técnica CGPMO+TPA são estatisticamente superiores aos obtidos pela CGP+NSGA-II na evolução de somadores de 8 bits e ALUs.

Tabela 13 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%.

Circuito	Par de Algoritmos	p-valor
ALU	CGPMO+TPA x CGP+NSGA-II	$3,756 \times 10^{-7}$
S8	CGPMO+TPA x CGP+NSGA-II	9.7733×10^{-4}
M8	CGPMO+TPA x CGP+NSGA-II	0.0690

6.3.1.2 *Projeto de circuitos aproximados quando* 60×10^6 *avaliações de circuito foram permitidas*

Os experimentos realizados quando 30×10^6 avaliações de circuito foram permitidas geraram circuitos aproximados que exibem erro e *delay* elevados. Neste sentido, ao se relaxar a exigência por funcionalidade perfeita os ganhos significativos são encontrados apenas quando se olha para a dissipação de potência dos circuitos obtidos, uma vez que estes circuitos superam as arquiteturas convencionais no que diz respeito à eficiência

energética. Visando obter ALUs, somadores e multiplicadores de 8 bits que apresentem um *trade-off* entre erro, potência e *delay* mais significativo que o encontrado quando 30×10^6 avaliações de circuito foram permitidas, os experimentos foram realizados com 60×10^6 avaliações de circuito.

A Figura 24 ilustra as projeções 2D das funcionalidades (erro, *delay* e potência) das versões não dominadas da ALU obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, já a Tabela 14 apresenta as características operacionais de 10 ALUs com os menores erros quando 60×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes e inicialização aleatória da população.

As projeções 2D ilustradas pela Figura 24 indicam que as soluções encontradas pelas abordagem CGP+NSGA-II estão mais espalhadas no espaço dos objetivos do que as soluções encontradas pelo método CGPMO+TPA. Além disso, o circuito que apresenta o maior valor de erro foi obtido pelo método CGPMO+TPA, enquanto o circuito que apresenta a maior dissipação de potência e o circuito que apresenta o maior *delay* foram encontrados pelo método CGP+NSGA-II. Por meio da análise das funcionalidades das 10 ALUs que menos erram obtidas por cada um dos métodos analisados aqui, mostradas pela Tabela 14, nota-se que quando 60×10^6 avaliações de circuitos foram permitidas, as abordagens CGPMO+TPA e CGP+NSGA-II obtiveram circuitos aproximados que não apresentam características operacionais significativamente diferentes das característica apresentadas pelas soluções obtidas quando 30×10^6 avaliações de circuitos foram permitidas.

Tabela 14 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
29,45	555,41	35,10	36,64	701,80	51,65
29,50	541,89	48,15	37,35	685,14	48,87
29,52	513,06	27,80	37,67	685,14	36,70
29,91	513,06	25,52	37,92	684,23	33,31
29,93	498,65	36,30	38,02	668,92	43,52
30,08	498,65	35,05	38,14	383,33	50,62
30,11	498,65	34,92	38,19	668,92	40,68
30,21	513,06	25,00	38,30	669,37	27,60
30,23	395,95	23,20	38,32	373,87	32,45
30,23	377,93	24,25	38,47	372,97	33,39

A Figura 25 ilustra as projeções 2D das funcionalidades (erro, *delay* e potência) das versões não dominadas do somador de 8 bits obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, já a Tabela 14 apresenta as características operacionais de 10 somadores

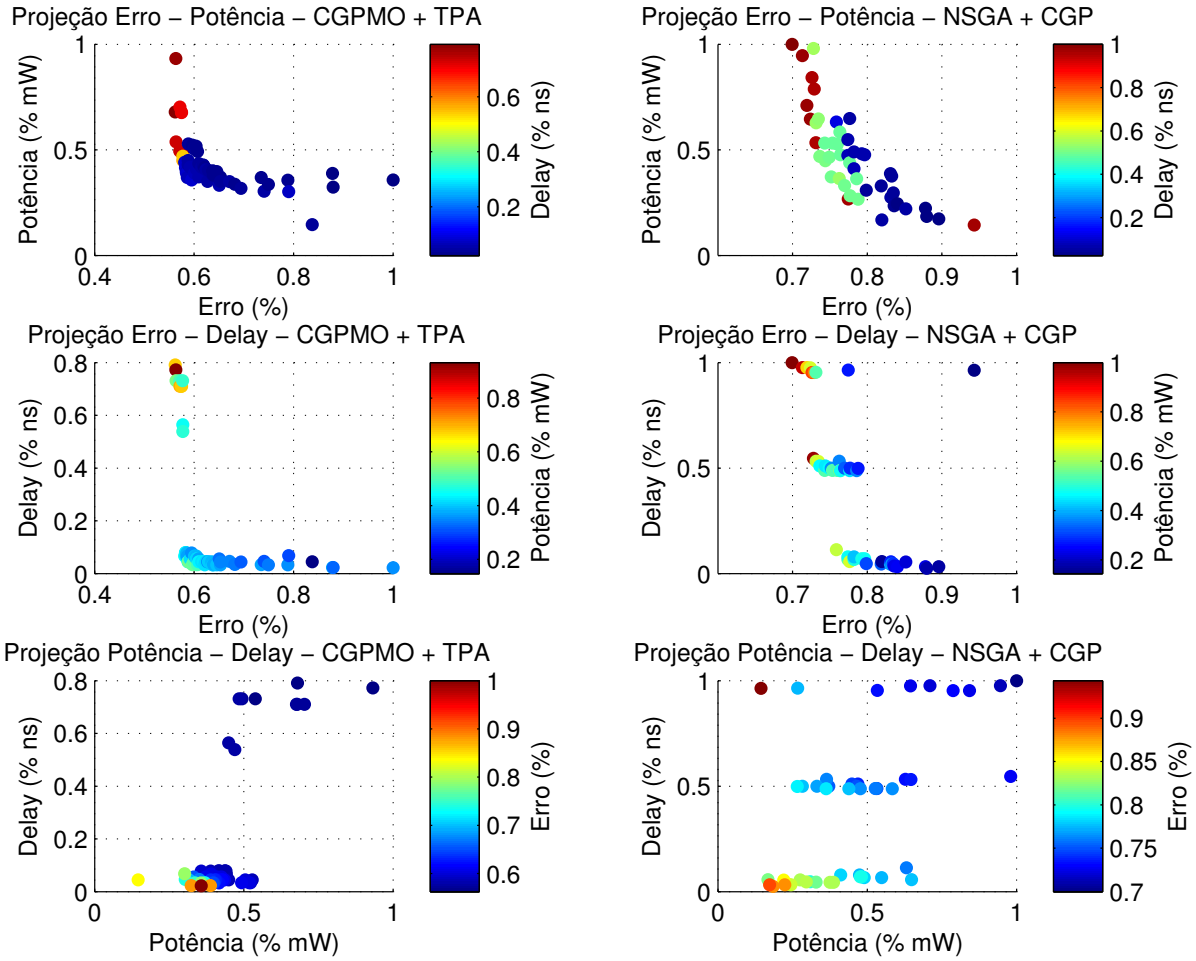

Figura 24 – Projeções 2D da dissipação de potência, *delay* e erro das ALUs evoluídas quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

de 8 bits com os menores erros quando 60×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes e inicialização aleatória da população.

As projeções 2D ilustradas pela Figura 25 indicam que as soluções encontradas pelas duas abordagens ocupam regiões parecidas dentro do espaço dos objetivos. Além disso, o circuito que apresenta o maior valor de *delay* e o circuito que apresenta a maior dissipação de potência foram obtidos pelo método CGP+NSGA-II, ao passo que o circuito com maior valor de erro foi obtido pelo método CGPMO+TPA. As funcionalidades operacionais, mostradas na Tabela 15, dos 10 somadores de 8 bits que menos erram obtidos por cada um dos métodos analisados aqui, indicam que mesmo ao se aumentar o limite de avaliações de circuitos permitidas, as características operacionais dos circuitos aproximados obtidos (erro, *delay* e dissipação de potência) não mudaram de forma significativa.

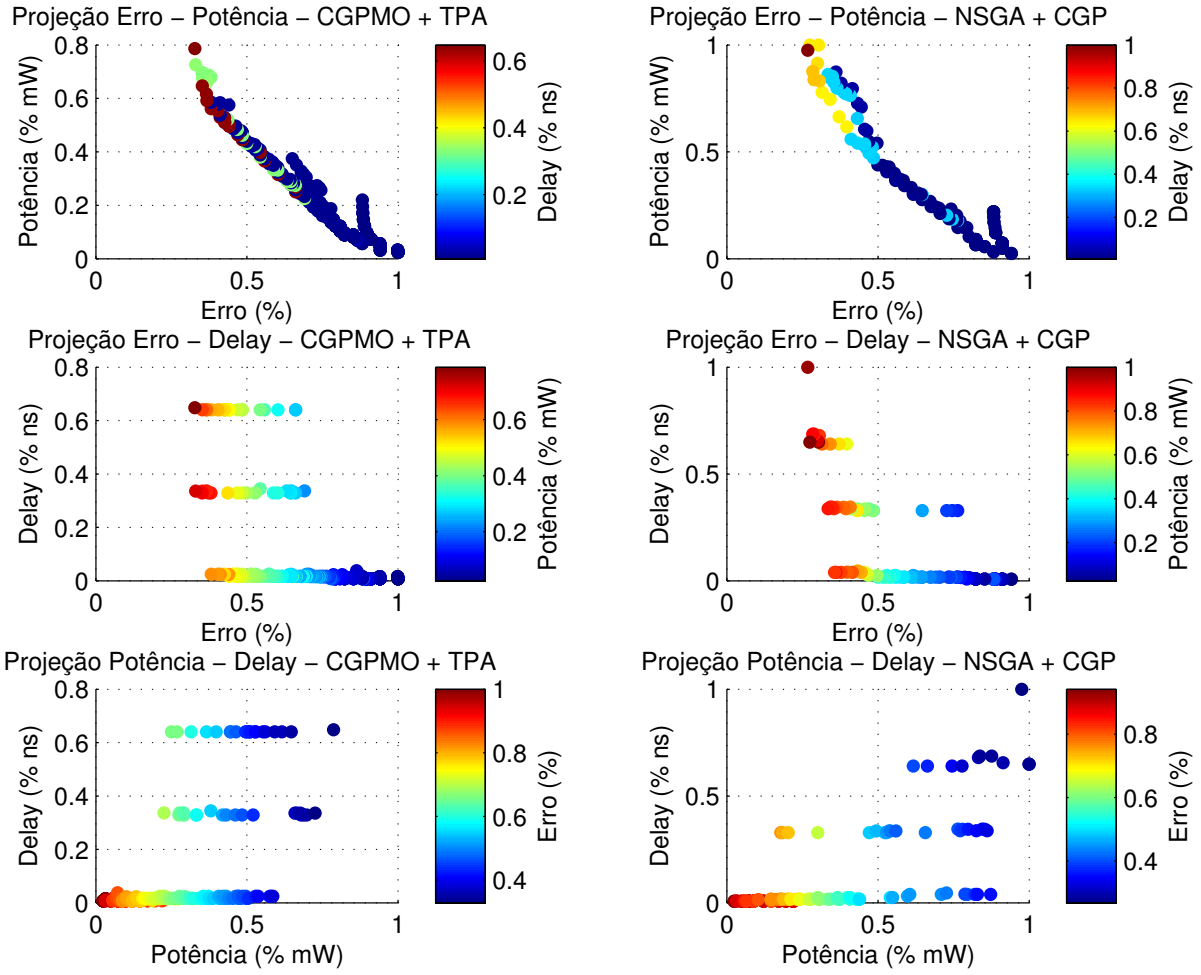

Figura 25 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Tabela 15 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
15,47	570,82	76,02	12,66	880,54	94,25
15,59	295,72	70,11	12,98	571,60	96,63
16,60	289,49	67,40	13,43	605,06	84,67
16,63	564,20	62,50	13,61	605,06	80,90
16,67	296,11	65,40	14,18	577,82	88,29
17,29	564,20	59,55	14,34	571,60	96,61
17,30	296,11	64,27	14,46	598,44	80,40
17,34	296,11	63,68	14,90	564,20	75,18
17,35	564,20	57,19	15,83	296,89	83,24
17,64	289,49	66,42	16,16	564,20	71,98

A Figura 26 ilustra as projeções 2D das funcionalidades (erro, *delay* e potência) das versões não dominadas do multiplicador de 8 bits obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II, já a Tabela 16 apresenta as características operacionais de 10 multiplicadores de 8 bits com os menores erros quando 60×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes quando a população é inicializada de forma aleatória.

As projeções 2D ilustradas pela Figura 26 indicam que as soluções encontradas pelas abordagem CGP+NSGA-II estão mais espalhadas no espaço dos objetivos do que as soluções encontradas pelo método CGPMO+TPA. Além disso, o circuito que apresenta o maior valor de erro, o circuito que apresenta a maior dissipação de potência e o circuito que apresenta o maior *delay* foram encontrados pelo método CGP+NSGA-II. As funcionalidades operacionais, mostradas na Tabela 16, dos 10 multiplicadores de 8 bits que menos erram obtidos por cada um dos métodos analisados aqui, indicam que, assim como na evolução da ALU e somador de 8 bits, o aumento do limite de avaliações de circuitos permitidas não levou à concepção de multiplicadores aproximados com características operacionais significativamente diferentes das característica apresentadas pelas soluções obtidas quando 30×10^6 avaliações de circuitos foram permitidas.

Tabela 16 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*. Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
38,34	408,75	64,83	42,14	414,27	64,27
38,35	408,75	62,31	42,20	413,59	57,24
38,36	408,75	61,13	42,36	317,50	56,22
38,38	408,75	56,30	42,39	316,42	62,08
38,39	408,75	53,79	42,87	309,56	81,27
38,40	408,75	53,65	42,91	409,15	49,12
38,41	408,75	50,84	42,92	314,00	70,19
38,42	404,17	53,14	42,96	306,46	68,29
38,43	404,17	52,80	42,99	309,42	52,11
38,44	408,75	47,94	43,09	409,15	46,55

Assim como na análise das soluções obtidas quando 30×10^6 avaliações de circuitos foram permitidas, foi calculado o hipervolume das soluções não dominadas obtidas pelas abordagens CGPMO+TPA e CGP+NSGA-II. Os pontos de referência adotados foram determinados pelos maiores valores de cada função objetivo considerando as abordagens CGPMO+TPA e CGP+NSGA-II. A Tabela 17 mostra os pontos de referência adotados.

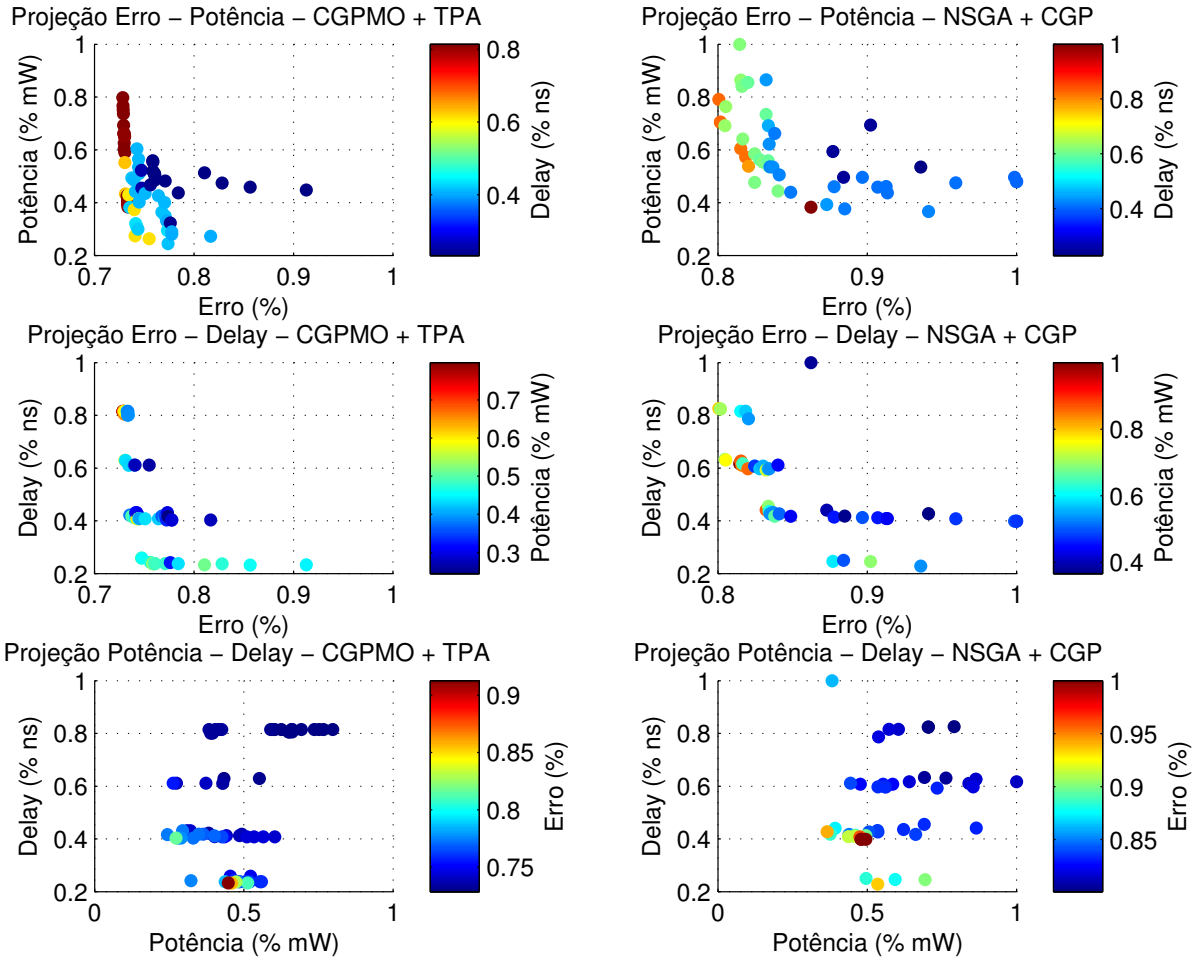

Figura 26 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada de forma aleatória e 60×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Tabela 17 – Pontos de referência utilizados no cálculo de hipervolume.

Circuito	**Erro (%)**	***Delay*** **(ns)**	**Potência (mW)**
ALU	0,5238	155,8000	0,5861
S8	0,4722	226,3000	0,4801
M8	0,5264	372,8000	2,6115

A Tabela 18 mostra os valores de hipervolume relativos aos resultados encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II no que diz respeito à evolução da ALU, somador e multiplicador de 8 bits considerando as 30 execuções independentes, inicialização aleatória da população e quando 60×10^6 avaliações de circuito foram permitidas. Neste cenário, os valores de hipervolume foram calculados sobre os valores normalizados das diferentes funções objetivo em relação aos valores apresentados pela Tabela 17. Os

melhores resultados estão destacados em negrito na Tabela 18, neste cenário a técnica CGPMO+TP foi a que apresentou os maiores valores médios de hipervolume para todos os circuitos analisados. O hipervolume médio obtido pela abordagem CGPMO+TPA é aproximadamente 24%, 10%, 13% maior do que o hipervolume médio obtido pela abordagem CGP+NSGA-II no projeto de ALUs, somadores e multiplicadores de 8 bits , respectivamente.

Destaca-se que, em relação à abordagem CGPMO+TPA, quando 60×10^6 avaliações de circuito foram permitidas o valor médio de hipervolume, em relação à evolução da ALU e multiplicador de 8 bits, foi maior do que o valor médio de hipervolume encontrado quando 30×10^6 avaliações de circuito foram permitidas, no que diz respeito ao somador de 8 bits o contrário aconteceu. Para todos os circuitos analisados, exceto pelo somador de 8 bits, os valores médios de hipervolume obtidos pela abordagem CGP+NSGA-II foram maiores quando 60×10^6 avaliações de circuito foram permitidas.

Tabela 18 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos.

Circuito	**Método**	**Melhor**	**Mediana**	**Média**	**STD**	**Pior**
ALU	CGPMO+TPA	0,3599	0,3176	**0,3208**	0,0227	0,2914
	CGP+NSGA-II	0,3346	0,2611	0,2594	0,0442	0,1790
S8	CGPMO+TPA	0,4069	0,3546	**0,3372**	0,0500	0,2120
	CGP+NSGA-II	0,3979	0,3070	0,3065	0,0439	0,2365
M8	CGPMO+TPA	0,3199	0,1674	**0,1683**	0,0783	0,0405
	CGP+NSGA-II	0,1832	0,1483	0,1492	0,0192	0,1130

Por meio dos resultados do teste de Kruskal-Wallis, apresentados na Tabela 19, observa-se que os p-valores retornados, exceto quando o problema analisado é o multiplicador de 8 bits, são menores que $0,05$. Portanto, para somadores de 8 bits e ALUs, com 5% de chance de erro, conclui-se que as amostras de hipervolueme analisadas não são oriundas da mesma população reforçando a hipótese de que exite diferença significativa entre os dados obtidos pelas abordagem CGPMO+TPA e CGP+NSGA-II. Desta forma, pode-se dizer que os resultados alcançados pela técnica CGPMO+TPA são estatisticamente superiores aos obtidos pela CGP+NSGA-II na evolução de somadores de 8 bits e ALUs.

Tabela 19 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%.

Circuito	**Par de Algoritmos**	**p-valor**
ALU	CGPMO+TPA x CGP+NSGA-II	2.4698×10^{-7}
S8	CGPMO+TPA x CGP+NSGA-II	$6,5416 \times 10^{-8}$
M8	CGPMO+TPA x CGP+NSGA-II	0.0701

6.3.2 **Inicialização da população com arquiteturas convencionais**

Os experimentos realizados quando a população foi inicializada de forma aleatória, ou seja, quando tencionou-se construir circuitos digitais aproximados do zero, de forma geral, geraram ALUs, somadores e multiplicadores de 8 bits aproximados com valores de erro elevados. Portanto, em vez de projetar circuitos aproximados a nível de porta lógica, as abordagens CGPMO+TPA e CGP+NSGA-II foram utilizadas como ferramenta para otimizar arquiteturas convencionais observando melhorar o *trade-off* entre erro, potência e *delay*. Além disso, uma vez que a técnica proposta trabalha com um conjunto de soluções candidatas, também será possível disponibilizar um conjunto de circuitos aproximados ao especialista do domínio da aplicação. Dessa forma, ele poderá escolher o que melhor se adéqua ao seu caso.

A Seção 6.3.1 aponta que mesmo ao se aumentar o limite de avaliações de circuitos permitidas para 60×10^6 os métodos analisados não geraram circuitos com características operacionais significativamente diferentes das característica apresentadas pelas soluções obtidas quando 30×10^6 avaliações de circuitos foram permitidas. Portando, aqui são analisados os resultados obtidos pelas técnicas CGPMO+TPA e CGP+NSGA-II quando 30×10^6 avaliações de circuito foram permitidas e uma arquitetura convencional foi inserida na população inicial.

A Figura 27 mostra diversas projeções 2D das características operacionais (erro, *delay* e potência) das versões não dominadas da Unidade Lógica Aritmética obtidas pelas abordagens CGPMO+TPA e CGP+NSGA-II quando 30×10^6 avaliações de circuitos foram permitidas considerando todas as execuções independentes quando a população é inicializada contendo um circuito convencional. Parâmetros de 10 ALUs que apresentam menores erros são mostrados na Tabela 20. As projeções 2D ilustradas pela Figura 27 indicam que as soluções encontradas pelas duas abordagens ocupam regiões parecidas dentro do espaço dos objetivos. O circuito que apresenta a maior dissipação de potência foi obtido pelo método CGP+NSGA-II, além disso, os dois métodos encontraram circuitos que têm o mesmo valor máximo de *delay* e erro.

As duas técnicas baseadas na CGP foram capazes de projetar ALUs aproximadas (portanto, que não atendem integralmente a tabela verdade). Os circuitos que têm suas características ilustradas na Tabela 20 dissipam menos potência do que a arquitetura convencional (chegando a dissipar no máximo $65,11\%$ da potência dissipada pela arquitetura de referência), contudo os ganhos em *delay* não são significativos.

Os métodos CGPMO+TPA e CGP+NSGA-II, a partir da inserção de uma arquitetura convencional na população inicial, encontraram ALUs que não apresentam erro. Considerando apenas circuitos factíveis, a abordagem CGPMO+TPA e a técnica CGP+NSGA-II conseguiram obter projetos com melhores dissipações de potência do que

o circuito base. Destaca-se que a abordagem CGPMO+TPA sintetizou uma ALU que apresenta *delay* e dissipação de potência inferior ao *delay* e dissipação de potência da arquitetura convencional.

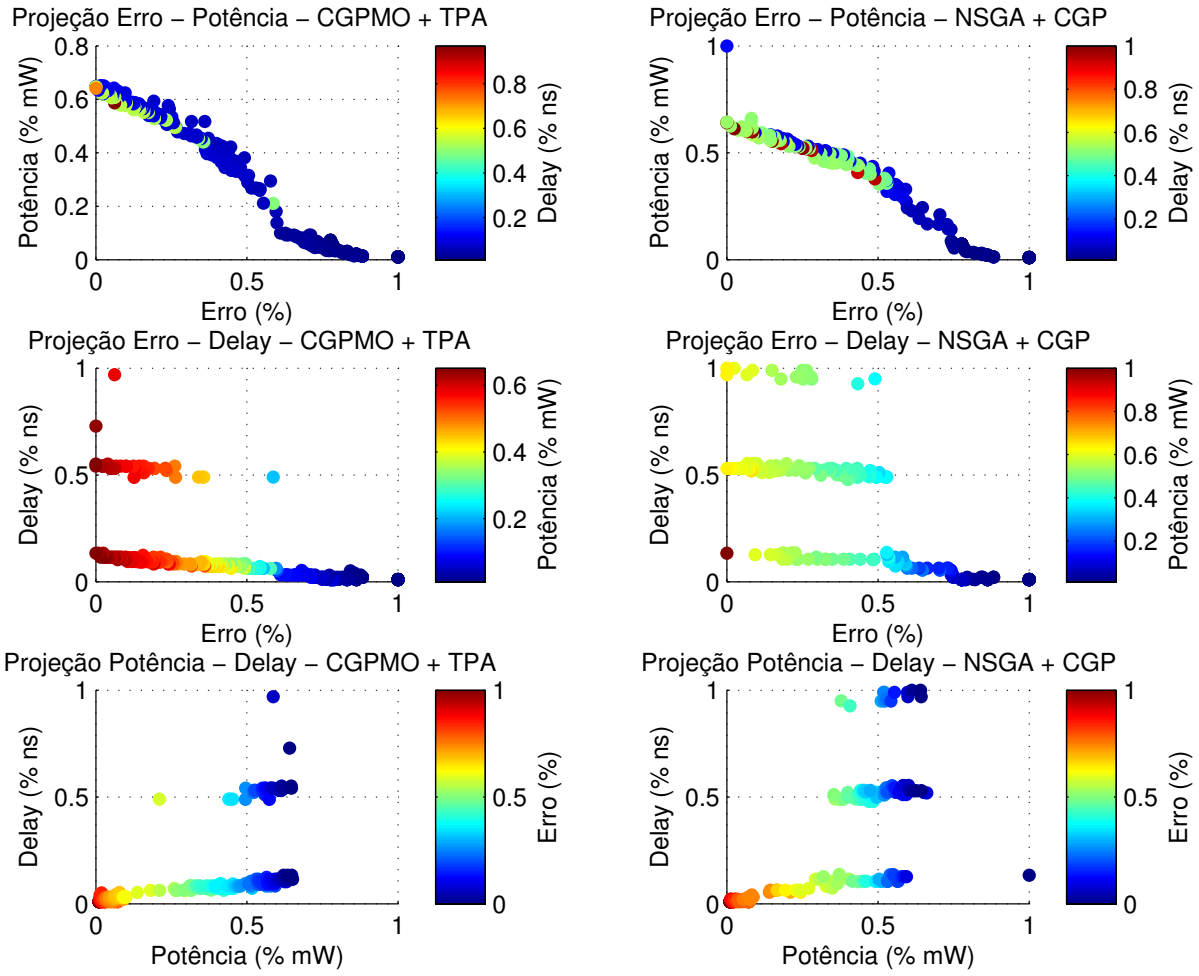

Figura 27 – Projeções 2D da dissipação de potência, *delay* e erro das AULs evoluídas quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Tabela 20 – Parâmetros das 10 ALUs que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do SN54/74LS181. Neste caso erro = 0, *delay* = 22.22 ns e potência = 1.14 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
0,00	99,55	64,80	0,00	395,50	64,34
0,00	404,05	64,74	0,00	723,87	64,30
0,00	411,71	64,71	0,00	746,85	64,11
0,00	544,14	64,11	0,52	395,50	62,55
0,33	99,55	63,46	1,04	746,85	61,24
0,68	86,04	65,11	1,70	396,40	60,59
0,93	404,05	62,39	2,12	395,50	62,15
1,04	404,05	61,85	2,61	403,15	60,20
1,09	86,04	65,04	2,73	396,40	60,42
1,27	86,04	64,66	2,86	723,87	59,93

A Figura 28 mostra diversas projeções 2D das características operacionais (erro, *delay* e potência) das versões não dominadas do somador de 8 obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II quando 30×10^6 avaliações de circuitos foram permitidas, enquanto a Tabela 21 apresenta características funcionais de 10 somadores de 8 bits que possuem o menor erro considerando todas as execuções independentes quando a população é inicializada contendo um circuito convencional. As projeções 2D ilustradas pela Figura 28 indicam que as soluções encontradas pelas duas abordagens ocupam regiões similares dentro do espaço dos objetivos. Os dois métodos obtiveram somadores de 8 bits que têm o mesmo valor máximo de *delay* e erro dissipação de potência.

Os resultados mostram que a abordagem CGPMO+TPA, assim como a abordagem CGP+NSGA-II, obtiveram somadores de 8 bits que não apresentam erro. Considerando apenas circuitos factíveis, a técnica CGPMO+TPA e abordagem CGP+NSGA-II não conseguiram obter projetos que conseguem ser mais rápidos (ter menor *delay*) e ao mesmo tempo dissipar menos potência do que o *Ripple Carry Adder*. Ainda dentro do cenário de soluções factíveis, para as duas técnicas aqui analisadas, ao se relaxar a exigência por menores *delays* obtêm-se circuitos que dissipam menos potência. As duas técnicas baseadas na CGP foram capazes de projetar somadores de 8 bits aproximados (portanto, que não atendem integralmente a tabela verdade) que apresentam diversas compensações entre erro, *delay* e potência dissipada.

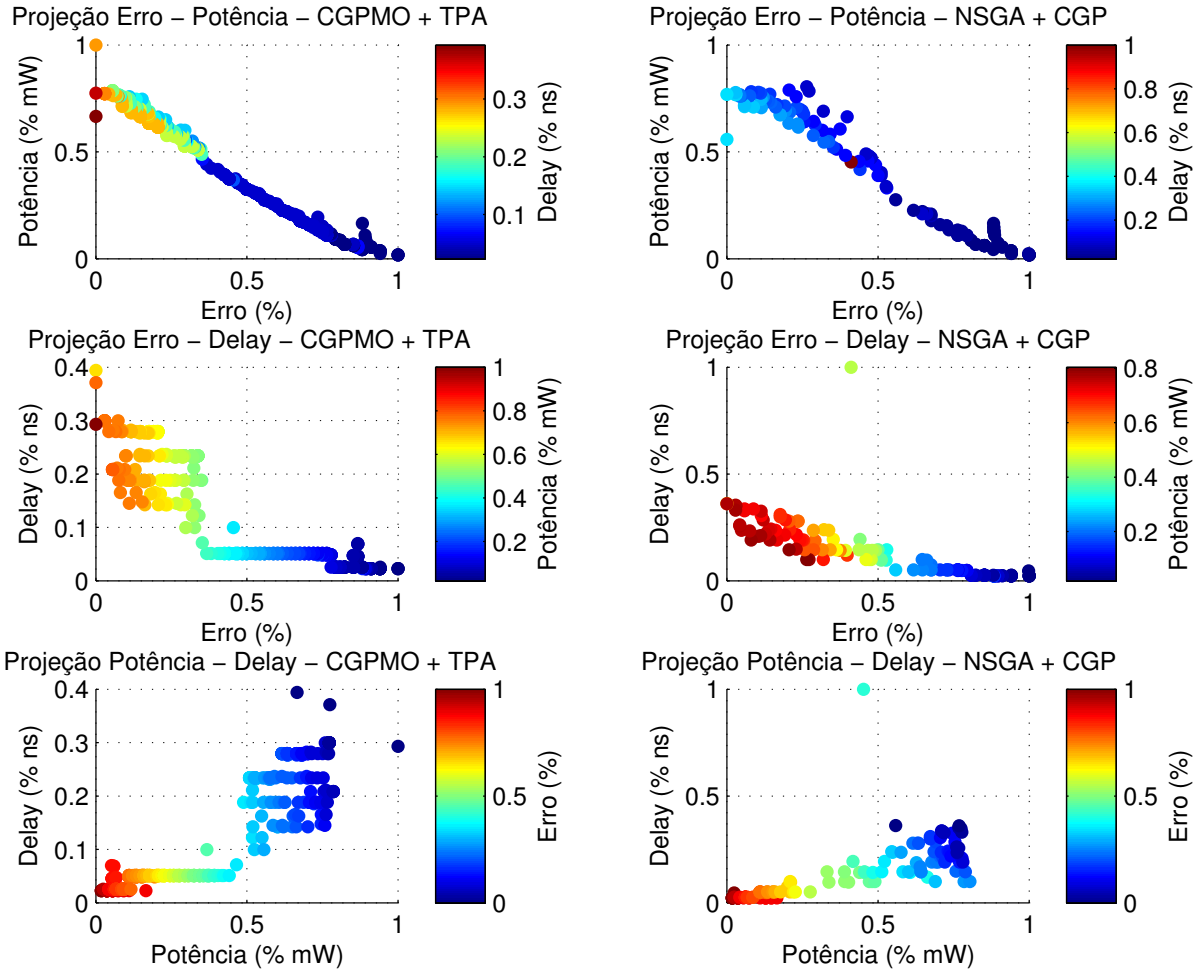

Figura 28 – Projeções 2D da dissipação de potência, *delay* e erro dos somadores de 8 bits evoluídos quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Tabela 21 – Parâmetros dos 10 somadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Adder*. Neste caso erro = 0, *delay* = 25.70 ns e potência = 0.50 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
0,00	84,82	128,18	0,00	104,28	98,54
0,00	107,39	99,28	0,00	104,67	71,58
0,00	114,01	85,39	1,39	95,72	99,61
1,37	86,77	99,05	1,40	101,56	98,43
2,02	80,93	98,75	2,02	75,88	97,45
2,60	60,31	100,74	2,28	68,48	97,71
2,71	81,71	97,76	2,78	95,72	91,94
3,13	80,93	98,30	3,58	67,70	97,71
3,32	61,09	97,51	3,82	55,25	100,08
3,47	86,77	97,33	4,17	97,28	91,00

A Figura 29 mostra diversas projeções 2D das características operacionais (erro, *delay* e potência) das versões não dominadas do multiplicador de 8 bits obtidas pelas estratégias CGPMO+TPA e CGP+NSGA-II quando 30×10^6 avaliações de circuitos foram permitidas, enquanto a Tabela 22 apresenta características funcionais de 10 multiplicadores de 8 bits com menores erros obtidos considerando todas as execuções independentes quando a população é inicializada contendo um circuito convencional. As projeções 2D ilustradas pela Figura 29 indicam que as soluções encontradas pelas duas abordagens ocupam regiões similares dentro do espaço dos objetivos, mas a densidade de soluções obtidas pela abordagem CGPMO+TPA e maior do que a densidade de soluções obtidas pelo método CGP+NSGA-II. O circuito que apresenta o maior erro foi obtido pelo método CGPMO+TPA, além disso, os dois métodos encontraram circuitos que têm o mesmo valor máximo de *delay* e dissipação de potência.

Considerando apenas circuitos factíveis, a técnica CGPMO+TPA e a abordagem CGP+NSGA-II apresentaram soluções que têm *delay* e dissipação de potência praticamente iguais ao *delay* e dissipação de potência do *Ripple Carry Array Multiplier*. Em relação aos multiplicadores de 8 bits que têm suas funcionalidades apresentadas na Tabela 22, ressalta-se que mesmo ao se relaxar a exigência por funcionalidade perfeita, as reduções em dissipação de potência e redução do *delay* foram pouco significativas. Ainda em relação os 10 multiplicadores que menos erram obtidos por cada um dos métodos aqui analisados, a redução máxima de *delay* foi de 4% tanto para os multiplicadores de 8 bits obtidos via CGPMO+TPA quanto para aqueles obtidos via CGP+NSGA-II, além disso, as reduções de dissipação de potência foram de no máximo 2% para multiplicadores de 8 bits obtidos pelas duas abordagem aqui analisadas.

Tabela 22 – Parâmetros dos 10 multiplicadores de 8 bits que apresentam os menores valores de erro quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os valores de referência são erro, *delay* e potência do *Ripple Carry Array Multiplier*. Neste caso erro = 0, *delay* = 74.30 ns e potência = 3.21 mW.

CGPMO+TPA			**CGP+NSGA-II**		
Erro (%)	*Delay* (% ns)	Potência (% mW)	Erro (%)	*Delay* (% ns)	Potência (% mW)
0,00	100,40	99,95	0,00	100,13	99,98
0,38	100,40	99,93	0,82	98,11	99,77
0,41	98,11	99,88	2,37	98,11	99,49
0,61	98,11	99,77	5,32	110,76	98,79
1,09	98,11	99,65	5,39	98,11	99,37
1,15	100,13	99,52	5,40	98,11	99,22
1,30	95,55	99,64	5,57	100,40	98,53
1,57	95,28	99,21	6,14	97,57	98,37
2,03	93,53	98,39	6,41	93,53	98,28
3,02	99,86	97,74	6,77	93,53	98,24

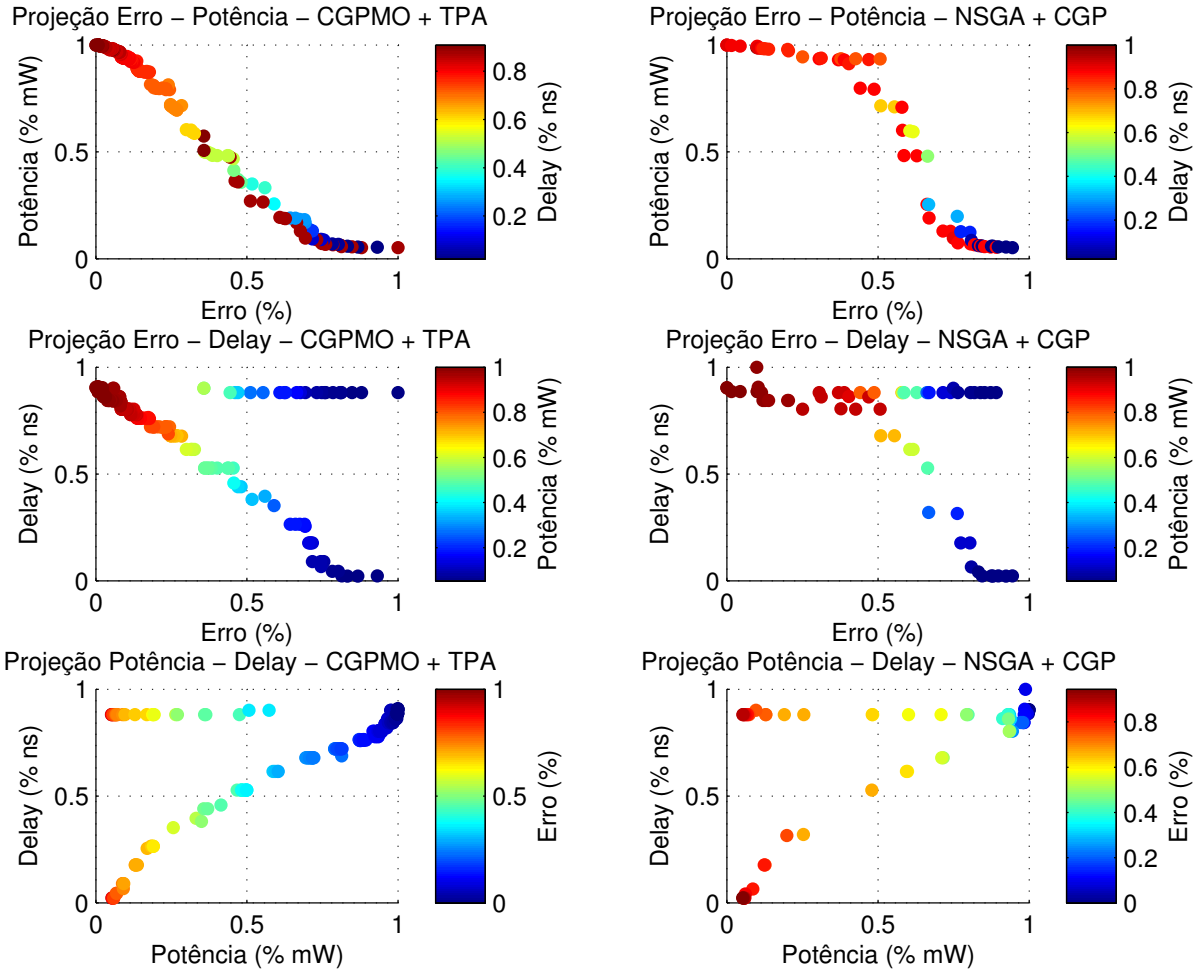

Figura 29 – Projeções 2D da dissipação de potência, *delay* e erro dos multiplicadores de 8 bits evoluídos quando a população é inicializada com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas. Os eixos coordenados estão normalizados de acordo com os maiores valores de dissipação de potência, *delay* e erro encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II.

Assim como na análise das soluções obtidas quando a população foi inicializada de maneira integralmente aleatória, foi calculado o hipervolume das soluções não dominadas obtidas pelas abordagens CGPMO+TPA e CGP+NSGA-II. Os pontos de referência adotados foram determinados pelos maiores valores de cada função objetivo considerando as abordagens CGPMO+TPA e CGP+NSGA-II. A Tabela 23 mostra os pontos de referência adotados.

Tabela 23 – Pontos de referência utilizados no cálculo de hipervolume.

Circuito	**Erro (%)**	***Delay* (ns)**	**Potência (mW)**
ALU	0,4284	165,8000	1,1348
S8	0,4722	74,4000	0,6369
M8	0,5353	82,3000	3,2135

A Tabela 24 mostra os valores de hipervolume relativos aos resultados encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II no que diz respeito à evolução da ALU, somador e multiplicador de 8 bits considerando as 30 execuções independentes, inicialização da população com uma arquitetura convencional e 30×10^6 avaliações de circuito foram permitidas Neste cenário, os valores de hipervolume foram calculados sobre os valores normalizados das diferentes funções objetivo em relação aos valores apresentados pela Tabela 23. Os melhores resultados estão destacados em negrito na Tabela 24, neste contexto a técnica CGPMO+TP foi a que apresentou os maiores valores médios de hipervolume para todos os circuitos analisados. O hipervolume médio obtido pela abordagem CGPMO+TPA é aproximadamente 46%, 19%, 40% maior do que o hipervolume médio obtido pela abordagem CGP+NSGA-II no projeto de ALUs, somadores e multiplicadores de 8 bits, respectivamente.

Tabela 24 – Valores de hipervolume encontrados pelas abordagens CGPMO+TPA e CGP+NSGA-II quando erro, *delay* e dissipação de potência são objetivos.

Circuito	**Método**	**Melhor**	**Mediana**	**Média**	**STD**	**Pior**
ALU	CGPMO+TPA	0,5354	0,4904	**0,4909**	0,0347	0,4340
	CGP+NSGA-II	0,4088	0,3397	0,3357	0,0450	0,2687
S8	CGPMO+TPA	0,5025	0,4849	**0,4687**	0,0345	0,4192
	CGP+NSGA-II	0,4422	0,4016	0,3944	0,0392	0,3326
M8	CGPMO+TPA	0,3730	0,3101	**0,2996**	0,0503	0,2028
	CGP+NSGA-II	0,2601	0,2204	0,2135	0,0368	0,1478

Por meio dos resultados do teste de Kruskal-Wallis, apresentados na Tabela 25, observa-se que os p-valores retornados são menores que $0,05$. Portanto com 5% de chance de erro, conclui-se que as amostras de hipervolume analisadas não são oriundas da mesma população, reforçando a hipótese de que exite diferença significativa entre os dados obtidos pelas abordagem CGPMO+TPA e CGP+NSGA-II. Desta forma, pode-se dizer que os resultados alcançados pela técnica CGPMO+TPA são estatisticamente superiores aos obtidos pela CGP+NSGA-II.

Tabela 25 – Teste de Kruskal-Wallis aplicado sobre os resultados de hipervolume gerados pelas abordagens CGPMO+TPA e CGP+NSGA-II como resultado da evolução da ALU, somador e multiplicador de 8 bits com nível de significância de 5%.

Circuito	**Par de Algoritmos**	**p-valor**
ALU	CGPMO+TPA x CGP+NSGA-II	2.8701×10^{-9}
S8	CGPMO+TPA x CGP+NSGA-II	6.7183×10^{-8}
M8	CGPMO+TPA x CGP+NSGA-II	3.1732×10^{-5}

7 CONCLUSÕES E TRABALHOS FUTUROS

Uma técnica de Programação Genética Cartesiana multiobjetivo, intitulada Programação Genética Cartesiana Multiobjetivo com Tamanho Populacional Adaptativo (CGPMO+TPA), foi proposta aqui para projetar circuitos digitais combinacionais aproximados. O método CGPMO+TPA, diferentemente de grande parte dos MOEAs, utiliza uma estratégia de adaptação do tamanho populacional visando diminuir o custo computacional relacionado à necessidade de se manter uma população fixa durante todo o processo evolutivo. Neste cenário, a abordagem CGPMO+TPA é baseada nos conceitos de dominância de Pareto e utiliza o *crowding distance* para evitar que o tamanho populacional aumente de forma considerável. Além disso, a codificação dos circuitos candidatos foi baseada na CGP, visto que tal algoritmo evolutivo é amplamente adotado no cenário de projeto evolutivo de circuitos digitais.

A abordagem CGPMO+TPA foi comparada com estratégias evolutivas presentes na literatura como NSGA-II, SPEA2 e hES no problema de projeto de somadores e multiplicadores de 2 bits. Neste cenário, a abordagem CGPMO+TPA foi capaz de projetar circuitos digitais corretos em um número maior de execuções independentes do que as outras abordagens analisadas. Em relação ao projeto de circuitos digitais combinacionais aproximados dois cenários foram analisados: um em que a construção dos circuitos é totalmente aleatória e outro em que considera-se uma arquitetura convencional como solução inicial. O método CGPMO+TPA foi comparado com a estratégia proposta por Hrbacek et al. (HRBACEK; MRAZEK; VASICEK, 2016), neste texto intitulada CGP+NSGA-II, que utiliza a CGP para codificar os circuitos candidatos e o NSGA-II para lidar com os múltiplos objetivos.

Os resultados obtidos apontam que as abordagens CGPMO+TPA e CGP+NSGA-II não são indicadas para a construção de arquiteturas complexas como ALUs, somadores e multiplicadores de 8 bits sem que seja inserido conhecimento do especialista do domínio do problema na evolução. Todos os circuitos encontrados sem considerar esse conhecimento, especialmente os multiplicadores, apresentaram níveis de erro elevados. Neste contexto, os projetos apresentaram melhoras em relação à dissipação de potência. Por outro lado, quando uma arquitetura convencional é adotada como solução inicial, as duas abordagens sintetizaram ALUs, somadores e multiplicadores de 8 bits que não apresentam erros em relação à tabela verdade. No que diz respeito ao projeto de ALUs aproximadas, destaca-se que ao se relaxar a exigência por funcionalidade perfeita economias expressivas de dissipação de potência são alcançadas. Em relação ao projeto de somadores de 8 bits aproximados, uma compensação mais expressiva entre erro e *delay* é observada. O projeto de multiplicadores de 8 bits aponta que mesmo ao se relaxar a exigência por funcionalidade perfeita, as reduções em dissipação de potência e redução do *delay* foram pouco significativas.

Ainda em relação ao projeto de circuitos aproximados, verificou-se que o método CGPMO+TPA obteve valores médios de hipervolume superiores aos obtidos pela estratégia CGP+NSGA-II para todos os circuitos alvo considerados aqui. O teste de Kruskal-Wallis, exceto para o projeto de multiplicadores de 8 bits sem que ocorra introdução de conhecimento do especialista, refutou a hipótese de que as soluções encontradas vêm da mesma população. Por consequência, a técnica CGPMO+TPA pode ser uma ferramente útil para projetar circuitos digitais combinacionais aproximados quando inicializa-se a população com uma arquitetura convencional.

Como trabalhos futuros pretende-se refinar o método CGPMO+TPA para lidar com otimização multiobjetivo com restrições em relação ao *delay*, pois hoje em dia as pessoas desejam que seus dispositivos sejam ainda mais rápidos. Além disso, tenciona-se considerar a estimativa de custo de fabricação de circuitos durante o processo evolutivo, já que essa métrica é desconsiderada pela literatura. As abordagens baseadas na CGP, geralmente, não utilizam o operador de recombinação durante o processo evolutivo, pois segundo Miller (MILLER, 2011) tal operador não contribui de forma significativa, neste cenário pode-se pensar em desenvolver um operador de recombinação que atue nos nós inativos. Neste sentido, a Engenharia Elétrica e a Modelagem Computacional podem potencializar as possibilidades de concepção de arquiteturas inovadores e totalmente funcionais (em relação ao erro), mais rápidas, energeticamente eficientes e que apresentem vantagens financeiras quando compradas com arquiteturas tradicionais sem que ocorra o aumento da complexidade dos circuitos.

REFERÊNCIAS

AGGARWAL, S.; MEHER, P. K.; KHARE, K. Concept, design, and implementation of reconfigurable cordic. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 24, n. 4, p. 1588–1592, April 2016. ISSN 1063-8210.

ALIOTO, M. Energy-quality scalable adaptive vlsi circuits and systems beyond approximate computing. In: *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*. [S.l.: s.n.], 2017. p. 127–132. ISSN 1558-1101.

BABU, K. S.; BALAJI, N. Approximation of digital circuits using cartesian genetic programming. In: *2016 International Conference on Communication and Electronics Systems (ICCES)*. [S.l.: s.n.], 2016. p. 1–6.

BALASUBRAMANIAN, P.; DANG, C.; MASKELL, D. L. Approximate quasi-delay-insensitive asynchronous adders: Design and analysis. In: *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. [S.l.: s.n.], 2017. p. 1196–1199. ISSN 1558-3899.

BOOLE, G. *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. London: Walton and Maberly, 1854.

CANHAM, R. O.; TYRRELL, A. M. Evolved fault tolerance in evolvable hardware. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*. [S.l.: s.n.], 2002. v. 2, p. 1267–1271 vol.2.

CHANDRAKASAN, A. P.; BRODERSEN, R. W. Minimizing power consumption in digital cmos circuits. *Proc. of the IEEE*, v. 83, n. 4, p. 498–523, 1995. ISSN 0018-9219.

CLEGG, J.; WALKER, J. A.; MILLER, J. F. A new crossover technique for cartesian genetic programming. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007. (GECCO '07), p. 1580–1587. ISBN 978-1-59593-697-4. Disponível em: <http://doi.acm.org/10.1145/1276958.1277276>.

COELLO, C. C.; LAMONT, G.; VELDHUIZEN, D. van. *Evolutionary Algorithms for Solving Multi-Objective Problems*. 2nd. ed. Berlin, Heidelberg: Springer, 2007. (Genetic and Evolutionary Computation). Disponível em: <http://books.google.com-/books?id=rXIuAMw3lGAC>.

CONOVER, W. J. *Practical nonparametric statistics*. 3. ed. New York, NY [u.a.]: Wiley, 1999. (Wiley series in probability and statistics). ISBN 0471160687.

DEB, K. *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047187339X.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, April 2002. ISSN 1089-778X.

EIBEN, A. E.; SMITH, J. E. *Introduction to Evolutionary Computing*. [S.l.]: SpringerVerlag, 2003. ISBN 3540401849.

ERCEGOVAC, M. D.; MORENO, J. H.; LANG, T. *Introduction to Digital Systems*. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc., 1998. ISBN 0471527998.

GASPAR-CUNHA, A.; TAKAHASHI, R.; ANTUNES, C. *Manual de computação evolutiva e metaheurística*. [S.l.: s.n.], 2012. (Ensino). ISBN 9789892601502.

GOLDMAN, B. W.; PUNCH, W. F. Reducing wasted evaluations in cartesian genetic programming. In: *Proceedings of the 16th European Conference on Genetic Programming*. Berlin, Heidelberg: Springer-Verlag, 2013. (EuroGP'13), p. 61–72. ISBN 978-3-642-37206-3. Disponível em: <http://dx.doi.org/10.1007/978-3-642-37207-0_6>.

GOLDMAN, B. W.; PUNCH, W. F. Analysis of cartesian genetic programming's evolutionary mechanisms. *IEEE Transactions on Evolutionary Computation*, v. 19, n. 3, p. 359–373, June 2015. ISSN 1089-778X.

HAIMES, Y. V.; LASDON, L. S.; WISMER, D. A. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-1, n. 3, p. 296–297, July 1971. ISSN 0018-9472.

HAN, J.; ORSHANSKY, M. Approximate computing: An emerging paradigm for energy-efficient design. In: *2013 18th IEEE European Test Symposium (ETS)*. [S.l.: s.n.], 2013. p. 1–6. ISSN 1530-1877.

HAYT, W.; BUCK, J. *Eletromagnetismo*. [S.l.]: LTC, 2003. ISBN 9788521613657.

HILDER, J.; WALKER, J. A.; TYRRELL, A. Use of a multi-objective fitness function to improve cartesian genetic programming circuits. In: *2010 NASA/ESA Conference on Adaptive Hardware and Systems*. [S.l.: s.n.], 2010. p. 179–185.

HRBACEK, R.; MRAZEK, V.; VASICEK, Z. Automatic design of approximate circuits by means of multi-objective evolutionary algorithms. In: *2016 International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*. [S.l.: s.n.], 2016. p. 1–6.

HUANG, J.; KUMAR, T. N.; ABBAS, H. A promising power-saving technique: Approximate computing. In: *2018 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*. [S.l.: s.n.], 2018. p. 285–290.

IRFAN, M. et al. Combinational digital circuit synthesis using cartesian genetic programming from a nand gate template. In: *2010 6th International Conference on Emerging Technologies (ICET)*. [S.l.: s.n.], 2010. p. 343–347.

JULIO, R. O. et al. Energy-efficient gaussian filter for image processing using approximate adder circuits. In: *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. [S.l.: s.n.], 2015. p. 450–453.

KAJITANI, I. et al. An evolvable hardware chip and its application as a multi-function prosthetic hand controller. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999. (AAAI '99/IAAI '99), p. 182–187. ISBN 0-262-51106-1. Disponível em: <http://dl.acm.org/citation.cfm?id=315149.315275>.

KANG, Y.; KIM, J.; KANG, S. Novel approximate synthesis flow for energy-efficient fir filter. In: *2016 IEEE 34th International Conference on Computer Design (ICCD)*. [S.l.: s.n.], 2016. p. 96–102.

KAUFMANN, P.; KNIEPER, T.; PLATZNER, M. A novel hybrid evolutionary strategy and its periodization with multi-objective genetic optimizers. In: *IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2010. p. 1–8. ISSN 1089-778X.

KAUFMANN, P.; PLATZNER, M. Moves: A modular framework for hardware evolution. In: *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*. [S.l.: s.n.], 2007. p. 447–454.

KAZARLIS, S.; KALOMIROS, J.; KALAITZIS, V. A cartesian genetic programming approach for evolving optimal digital circuits. *Journal of Engineering Science and Technology Review*, v. 9, p. 88–92, 10 2016.

KE, P.; CAO, Z.; NIE, X. A generic architecture of complete intrinsic evolvable digital circuits. In: *2010 Third International Symposium on Knowledge Acquisition and Modeling*. [S.l.: s.n.], 2010. p. 379–382.

KEYMEULEN, D. et al. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Transactions on Reliability*, v. 49, n. 3, p. 305–316, Sept 2000. ISSN 0018-9529.

KHAN, G. M.; MILLER, J. F.; HALLIDAY, D. M. Evolution of cartesian genetic programs for development of learning neural architecture. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 19, n. 3, p. 469–523, set. 2011. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162/EVCO_a_00043>.

KOZA, J. R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0-262-11170-5.

KOZA, J. R. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, v. 4, n. 2, p. 87–112, Jun 1994. ISSN 1573-1375. Disponível em: <https://doi.org/10.1007/BF00175355>.

KOZA, J. R. et al. Toward automated design of industrial-strength analog circuits by means of genetic programming. In: _____. *Genetic Programming Theory and Practice II*. Boston, MA: Springer US, 2005. p. 121–142. ISBN 978-0-387-23254-6. Disponível em: <https://doi.org/10.1007/0-387-23254-0_8>.

KULKARNI, P.; GUPTA, P.; ERCEGOVAC, M. Trading accuracy for power with an underdesigned multiplier architecture. In: *2011 24th Internatioal Conference on VLSI Design*. [S.l.: s.n.], 2011. p. 346–351. ISSN 1063-9667.

LOSHCHILOV, I. Lm-cma: An alternative to l-bfgs for large-scale black box optimization. *Evolutionary Computation*, v. 25, n. 1, p. 143–171, March 2017. ISSN 1063-6560.

LOVAY, M. A.; PERETTI, G. M.; ROMERO, E. A. Implementation of an adaptive filter using an evolvable hardware strategy. *IEEE Latin America Transactions*, v. 13, n. 4, p. 927–934, April 2015. ISSN 1548-0992.

MANFRINI, F. A. L. *Estratégias de Busca no Projeto Evolucionista de Circuitos Combinacionais*. Tese (Doutorado) — Programa de Pós Graduação em Modelagem Computacional, Universidade Federal de Juiz de Fora, Juiz de Fora, Brasil, 2017.

MILLER, J. F. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In: BANZHAF, W. et al. (Ed.). *Proceedings of the Genetic and Evolutionary Computation Conference*. Orlando, Florida, USA: Morgan Kaufmann, 1999. v. 2, p. 1135–1142. ISBN 1-55860-611-4. Disponível em: <http://citeseer.ist.psu.edu/153431.html>.

MILLER, J. F. Evolution of digital filters using a gate array model. In: POLI, R. et al. (Ed.). *Evolutionary Image Analysis, Signal Processing and Telecommunications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. p. 17–30. ISBN 978-3-540-48917-7.

MILLER, J. F. *Cartesian Genetic Programming*. [S.l.]: Springer-Verlag Berlin Heidelberg, 2011. ISBN 978-3-642-17309-7.

MILLER, J. F. et al. *Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study*. 1997.

NAKAYAMA, H.; YUN, Y.; YOON, M. *Sequential Approximate Multiobjective Optimization Using Computational Intelligence*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 3540889094, 9783540889090.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.

OPPENHEIM, A.; WILLSKY, A.; NAWAB, S. *Sinais e sistemas*. Prentice-Hall, 2010. ISBN 9788576055044. Disponível em: <https://books.google.com.br-/books?id=ZOg9bwAACAAJ>.

PETRLIK, J.; SEKANINA, L. Multiobjective evolution of approximate multiple constant multipliers. In: *2013 IEEE 16th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*. [S.l.: s.n.], 2013. p. 116–119.

PHILLIPS, D. T.; SOLBERG, J. J. (Ed.). *Operations Research: Principles and Practice, 2Nd Ed.* New York, NY, USA: John Wiley & Sons, Inc., 1987. ISBN 0471086088.

RAHAL, J. A. et al. Low power gdi alu design with mixed logic adder functionality. In: *2018 International Conference on IC Design Technology (ICICDT)*. [S.l.: s.n.], 2018. p. 9–12.

RECHENBERG, I. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973. (Problemata, 15).

RODRIGUES, G. S. de M. *A Efetividade dos Instrumentos de Financiamento Público da FINEP no Desenvolvimento do APL Eletroeletrônico de Santo Rita do Sapucaí*. Dissertação (Mestrado) — Escola Brasileira de Administração Pública e de Empresas, Fundação Getúlio Vargas, Rio de Janeiro, Brasil, 2010.

Roeva, O.; Fidanova, S.; Paprzycki, M. Influence of the population size on the genetic algorithm performance in case of cultivation process modelling. In: *2013 Federated Conference on Computer Science and Information Systems*. [S.l.: s.n.], 2013. p. 371–376.

RUSSO, I. L. de S. *Otimização Multiobjetivo e Programação Genética para Descoberta de Conhecimento em Engenharia*. Dissertação (Mestrado) — Programa de Pós Graduação em Modelagem Computacional, Universidade Federal de Juiz de Fora, Juiz de Fora, Brasil, 2017.

SALVADOR, R. et al. Fault tolerance analysis and self-healing strategy of autonomous, evolvable hardware systems. In: *2011 International Conference on Reconfigurable Computing and FPGAs*. [S.l.: s.n.], 2011. p. 164–169. ISSN 2325-6532.

SEDRA, A. S.; SMITH, K. C. *Microelectronic Circuits*. fifth. [S.l.]: Oxford University Press, 2004.

SEKANINA, L.; VASICEK, Z. Approximate circuit design by means of evolvable hardware. In: *Proc. of the IEEE Intl. Conference on Evolvable Systems (ICES)*. [S.l.: s.n.], 2013. p. 21–28.

SENTOVICH, E. et al. *SIS: A System for Sequential Circuit Synthesis*. [S.l.], 1992.

SHE, X.; LAI, J. Self-adaptive evolution of complex logic circuits. In: *2009 IEEE Workshop on Evolvable and Adaptive Hardware*. [S.l.: s.n.], 2009. p. 47–53.

SILVA, J. E. da; BERNARDINO, H. Cartesian genetic programming with crossover for designing combinational logic circuits. In: *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*. [S.l.: s.n.], 2018. p. 145–150.

SOARES, L. B.; BAMPI, S.; COSTA, E. Approximate adder synthesis for area- and energy-efficient fir filters in cmos vlsi. In: *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*. [S.l.: s.n.], 2015. p. 1–4.

Srinivas, N.; Deb, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, v. 2, n. 3, p. 221–248, Sep. 1994. ISSN 1063-6560.

STEPNEY, S.; ADAMATZKY, A. *Inspired by Nature*: Essays presented to julian f. miller on the occasion of his 60th birthday. [S.l.]: Springer-Verlag Berlin Heidelberg, 2018. ISBN 978-3-319-67996-9.

TOCCI, R.; WIDMER, N.; MOSS, G. *Sistemas Digitais - Principios E Aplicaçoes*. PEARSON BRASIL, 2011. ISBN 9788576059226. Disponível em: <https://books.google-.com.br/books?id=KI8DaAEACAAJ>.

TSAO, Y.; CHOI, K. Area-efficient vlsi implementation for parallel linear-phase fir digital filters of odd length based on fast fir algorithm. *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 59, n. 6, p. 371–375, June 2012. ISSN 1549-7747.

TURNER, A. J.; MILLER, J. F. Neutral genetic drift: An investigation using cartesian genetic programming. *Genetic Programming and Evolvable Machines*, Kluwer Academic Publishers, Hingham, MA, USA, v. 16, n. 4, p. 531–558, dez. 2015. ISSN 1389-2576. Disponível em: <http://dx.doi.org/10.1007/s10710-015-9244-6>.

VASICEK, Z. Cartesian gp in optimization of combinational circuits with hundreds of inputs and thousands of gates. In: . [S.l.: s.n.], 2015. v. 9025, p. 139–150.

VASICEK, Z.; SEKANINA, L. Formal verification of candidate solutions for post-synthesis evolutionary optimization in evolvable hardware. v. 12, p. 305–327, 09 2011.

VASICEK, Z.; SEKANINA, L. On area minimization of complex combinational circuits using cartesian genetic programming. In: *2012 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2012. p. 1–8. ISSN 1089-778X.

VASICEK, Z.; SEKANINA, L. Evolutionary design of approximate multipliers under different error metrics. In: *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*. [S.l.: s.n.], 2014. p. 135–140.

VASICEK, Z.; SEKANINA, L. How to evolve complex combinational circuits from scratch? In: *2014 IEEE International Conference on Evolvable Systems*. [S.l.: s.n.], 2014. p. 133–140.

VASICEK, Z.; SEKANINA, L. Circuit approximation using single- and multi-objective cartesian gp. In: MACHADO, P. et al. (Ed.). *Genetic Programming*. Cham: Springer International Publishing, 2015. p. 217–229. ISBN 978-3-319-16501-1.

VASICEK, Z.; SEKANINA, L. Evolutionary approach to approximate digital circuits design. *IEEE Transactions on Evolutionary Computation*, v. 19, n. 3, p. 432–444, June 2015. ISSN 1089-778X.

VENKATARAMANI, S.; ROY, K.; RAGHUNATHAN, A. Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits. In: *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. [S.l.: s.n.], 2013. p. 1367–1372. ISSN 1530-1591.

VENKATARAMANI, S. et al. Salsa: Systematic logic synthesis of approximate circuits. In: *DAC Design Automation Conference 2012*. [S.l.: s.n.], 2012. p. 796–801. ISSN 0738-100X.

WESTE, N.; HARRIS, D. *CMOS VLSI Design: A Circuits and Systems Perspective*. 4th. ed. USA: Addison-Wesley Publishing Company, 2010. ISBN 0321547748, 9780321547743.

WHILE, L.; BRADSTREET, L.; BARONE, L. A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, v. 16, n. 1, p. 86–95, Feb 2012. ISSN 1089-778X.

YADAV, N.; KUMARI, P. Design of alu using dual mode logic with optimized power and speed. In: *2017 International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT)*. [S.l.: s.n.], 2017. p. 41–45.

ZADEH, L. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, v. 8, n. 1, p. 59–60, January 1963. ISSN 0018-9286.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 8, n. 2, p. 173–195, jun. 2000. ISSN 1063-6560. Disponível em: <http://dx.doi.org/10.1162-/106365600568202>.

ZITZLER, E.; LAUMANNS, M.; THIELE, L. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: GIANNAKOGLOU, K. C. et al. (Ed.). *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Athens, Greece: International Center for Numerical Methods in Engineering, 2001. p. 95–100.

ZITZLER, E. et al. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 2, p. 117–132, April 2003. ISSN 1089-778X.