

UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

**Explorando Técnicas de Computação Aproximada para o
Projeto de Filtro Gaussiano Energeticamente Eficiente**

Talita Borges

Orientador: Prof.^a. Dra. Cristina Meinhardt
Co-orientador: Prof. Dr. Leonardo Bandeira

Rio Grande, 2021

Ficha Catalográfica

B732e Borges, Talita.

Explorando técnicas de computação aproximada para o projeto de Filtro Gaussiano energeticamente eficiente / Talita Borges. – 2021. 87 f.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-Graduação em Computação, Rio Grande/RS, 2021.

Orientadora: Dra. Cristina Meinhardt.

Coorientador: Dr. Leonardo Bandeira.

1. Computação Aproximada 2. Arquitetura Configurável
3. Eficiência Energética 4. Filtro Gaussiano I. Meinhardt, Cristina
II. Bandeira, Leonardo III. Título.

CDU 004

Catálogo na Fonte: Bibliotecário José Paulo dos Santos CRB 10/2344

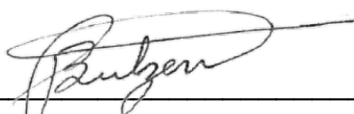
ATA DE SESSÃO DE DEFESA DE DISSERTAÇÃO DE MESTRADO

Ata nº 11/2021

Na data de 1º de junho de 2021, às 13 horas e 30 minutos, ocorreu a Sessão de Defesa de Dissertação de Mestrado de Talita Alves Borges, que apresentou a dissertação intitulada “Explorando Técnicas de Computação Aproximada para o Projeto de Filtro Gaussiano Energeticamente Eficiente”, realizada sob a orientação da Profa. Dra. Cristina Meinhardt e coorientação do Prof. Dr. Leonardo Bandeira Soares (IFRS). A banca examinadora foi constituída pelos Profs. Dr. Paulo Francisco Butzen (PPGComp), Dr. Daniel Munari Vilchez Palomino (UFPel) e Dr. Claudio Machado Diniz (UFRGS), sob a presidência da orientadora. Após a apresentação do trabalho, a banca arguiu a candidata e, a seguir, deliberou pela

- (x) aprovação da Dissertação.
- () aprovação da Dissertação, sugerindo modificações no texto.
- () reprovação da Dissertação.

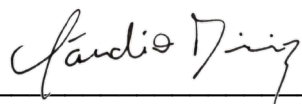
Rio Grande, 1º de junho de 2021.



Prof. Dr. Paulo Francisco Butzen (participação remota)



Prof. Dr. Daniel Munari Vilchez Palomino (participação remota)



Prof. Dr. Claudio Machado Diniz (participação remota)

Prof. Dr. Leonardo Bandeira Soares (participação remota)
CoorientadorProfa. Dra. Cristina Meinhardt (participação remota)
Orientadora

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me proporcionado ser aprovada neste mestrado em Engenharia de Computação, concedendo-me saúde, inteligência e forças para nunca desistir dessa jornada.

Aos meus pais, Mara Borges e Manoel Borges por sempre me incentivarem em minha jornada de estudos. Por terem ido morar comigo em Rio Grande para que eu não me sentisse sozinha e conseguisse focar totalmente em meus estudos. Vocês são demais e obrigada por tudo até aqui, amo vocês demais.

À minha vó Elci Alves, que hoje em dia é um anjo que cuida e zela pela nossa família, que no início deste mestrado me apoiou, mesmo sentindo saudades pela distância. Também agradeço à minha vó Iria Borba, que mesmo estando no céu, um dia me incentivou a seguir meus estudos e, se cheguei até aqui foi com o incentivo dela.

Ao meu noivo Michel Caneda, que acompanha minha trajetória desde a graduação, sempre me encorajando a seguir meus sonhos. A distância e a saudade não foram fáceis pra nós. Mas foram muitas viagens Rio Grande/Bagé e vice-versa, muitos passeios no shopping e na praia, momentos incríveis que Deus nos proporcionou viver neste período.

À minha orientadora Cristina Meinhardt, pela orientação prestada durante o mestrado, com muita responsabilidade e dedicação a todo momento. Sou muito grata por ter aceitado ser minha orientadora e ter apresentado para elaboração deste trabalho, o professor Leonardo Soares, pelos incentivos à pesquisa, certamente aprendi muito contigo à organizar meu tempo, criar rotinas de estudos, olhar para meus artigos com carinho.

A meu co-orientador Leonardo Soares, por toda paciência e dedicação, estando sempre disposto a me orientar neste trabalho. Foste um exemplo de inspiração para mim no mestrado. É uma honra poder seguir minha jornada acadêmica, adquirindo conhecimento e aprendizado contigo no doutorado na UFRGS.

Aos meus colegas de mestrado da Furg, pela amizade e parceria de estudos e trabalhos, especialmente à Stefane Menezes, Lucas Dantas, Clayton Farias, Ingrid Oliveira. Às minhas amigas da Furg, no qual pudemos conviver por 1 ano indo juntas para universidade: Luana Freire e Yanna Chang.

RESUMO

BORGES, Talita. **Explorando Técnicas de Computação Aproximada para o Projeto de Filtro Gaussiano Energeticamente Eficiente**. 2021. 87 f. . Universidade Federal do Rio Grande - FURG, Rio Grande.

Atualmente, existem muitas aplicações que demandam um projeto de circuitos integrados que otimize a eficiência energética. Nesse âmbito, arquiteturas configuráveis tornaram-se alternativas para atender, dinamicamente, aos requisitos de qualidade e energia para o processamento de imagens e vídeo. O conceito de computação aproximada emergiu como técnica promissora para obter melhorias substanciais em desempenho e eficiência energética para uma variedade de dispositivos eletrônicos. Este trabalho propõe uma exploração a nível de circuito como técnica de computação aproximada aplicada a um acelerador de filtro gaussiano 2D, com a finalidade de investigar o *trade-off* entre qualidade e eficiência energética. Os principais objetivos são: i) realizar uma exploração do espaço de projeto através da redução na largura de *bits* do filtro Gaussiano 2D, ii) analisar o impacto da computação aproximada em termos de qualidade para o processamento digital de imagens, iii) desenvolver arquiteturas aproximadas para o filtro gaussiano 2D com aproximação definida em tempo de projeto, iv) projetar aceleradores de *hardware* do filtro gaussiano 2D configurável em tempo de execução e, v) observar o comportamento em um conjunto de imagens ao se remover o sinal de controle de *reset*, a fim de analisar a degradação da qualidade, magnitude de erro, assim como a possibilidade de redução do número de ciclos durante uma nova configuração do filtro. Todos os circuitos projetados foram sintetizados para implementação de Circuitos Integrados de Aplicação Específica (ASIC), para a tecnologia ST de 65 nm. Para a arquitetura definida em tempo de projeto, a redução de energia por operação observada variou de 13,18% até 67,88%. Na arquitetura configurável em tempo de execução, os resultados de energia apresentam reduções que variam de 9,87% até 59% em relação, aos resultados da arquitetura definida em tempo de projeto. Os componentes adicionais de *hardware* referentes à técnica de *clock gating* investigada para o controle de configurabilidade implicaram em um *overhead* de 15% de aumento da área do circuito. A análise dinâmica da qualidade do filtro gaussiano 2D, considerando um conjunto de imagens com adição de ruído, atingiu um PSNR variando de 27,5 dB a 51,2 dB, no qual níveis mais baixos e mais altos estão relacionados às configurações aproximadas de 3 e 7 *bits*, respectivamente. A remoção do sinal de controle de *reset* apresenta vantagens nos níveis de PSNR, além de permitir reduções no projeto da lógica de controle para reconfigurações.

Palavras-chave: Computação Aproximada, Arquitetura Configurável, Eficiência Energética, Filtro Gaussiano.

ABSTRACT

BORGES, Talita. **Exploring Approximate Computing through the Design of an Energy-Efficient 2D Gaussian Filter**. 2021. 87 f. . Universidade Federal do Rio Grande - FURG, Rio Grande.

Currently, many applications require an integrated circuit design that optimizes energy efficiency. In this context, configurable architectures have become alternatives to dynamically meet the quality and energy requirements for image and video processing. The concept of approximate computing has emerged as a promising technique for achieving substantial improvements in performance and energy efficiency for a variety of electronic devices. This work proposes a fine-grained exploration at the level of circuit as an approximate computing technique applied to a 2D filter accelerator, aiming to investigate the trade-off between quality and energy efficiency, where it becomes possible to configure them dynamically. The main objectives are: i) to explore the design by reducing the bit-width of the bits of the 2D gaussian filter, ii) to analyze the impact of approximate computation in terms of quality for digital image processing, iii) to develop approximate architectures for the 2D gaussian filter with defined approximation at design time, iv) to design a hardware accelerator of the 2D gaussian filter configurable at run-time, and v) observe the behavior in a set of images when removing the control signal from reset, in order to analyze the quality degradation, error magnitude, as well as the reduction of the number of cycles during a new filter configuration. All projected circuits were synthesized for the implementation of Application Specific Integrated Circuits (ASIC), for ST technology 65 nm. For the architecture defined in design-time, the energy reduction per operation observed varied from 13.18% to 67.88%. In the configurable architecture at runtime, the energy results show reductions ranging from 9.87% to 59% in relation to the results of the architecture defined in design-time. The additional components of hardware concerning the clock gating technique for configurability control imply an overhead of 15% increase in the circuit area. The dynamic analysis of the quality of the 2D Gaussian filter, considering a set of images with added noise, reached a PSNR ranging from 27.5 dB to 51.2 dB, in which lower and higher levels are related to the approximate settings of 3 and 7 bits, respectively. Removing the control signal from reset has advantages in PSNR levels, in addition to allowing reductions in the design of the control logic for reconfigurations.

Keywords: Approximate Computing, Configurable Architecture, Energy Efficiency, Gaussian Filter.

LISTA DE FIGURAS

Figura 1	Espectro Eletromagnético.	19
Figura 2	Imagem digital bidimensional.	20
Figura 3	Etapas do Processamento Digital de uma Imagem.	21
Figura 4	Arquitetura do filtro gaussiano 2D 5X5 de 8 <i>bits</i>	36
Figura 5	Versões da arquitetura do filtro gaussiano 2D 5X5 de 8 <i>bits</i>	38
Figura 6	Arquitetura do acelerador do filtro gaussiano 2D com precisão configurável em tempo de execução	39
Figura 7	Fluxo de etapas para avaliação de qualidade.	41
Figura 8	Truncagem dos LSBs	42
Figura 9	Aplicação do filtro gaussiano em a) sem ruído e b) adição de ruído gaussiano.	43
Figura 10	a) Imagem Original do pássaro de 8 bits, b) Imagem ilustrativa das faixas e c) Resultado da concatenação	46
Figura 11	Resultados de Energia-nodo de tecnologia @65nm.	48
Figura 12	Tendência de redução da área	49
Figura 13	Resultados de energia da tecnologia de @65 nm da arquitetura com precisão configurável em tempo de execução	51
Figura 14	Resultados de Potência versus PSNR	52
Figura 15	Análise objetiva para a imagem de entrada sem ruído: (a) Filtro gaussiano 2D preciso (b) Perfil aproximado de 3 <i>bits</i> de redução (c) Mapa de calor com as diferenças entre os filtros gaussianos 2D preciso e de 3 <i>bits</i>	54
Figura 16	Análise objetiva para a imagem de entrada corrompida com ruído gaussiano com $\mu = 0$ e $\sigma^2 = 0,01$: (a) Filtro gaussiano 2D preciso (b) Perfil aproximado de 3 <i>bits</i> de redução (c) Mapa de calor com as diferenças entre os filtros gaussianos 2D preciso e de 3 <i>bits</i>	55
Figura 17	<i>Reset-On</i> versus <i>ResetOff</i>	57
Figura 18	Análise das combinações do sinal de <i>reset</i>	58
Figura 19	Percepção visual. a) 8 <i>bits</i> ; b) 7 <i>bits</i> ; c) 6 <i>bits</i> ; d) 5 <i>bits</i> ; e) 4 <i>bits</i> f) 3 <i>bits</i>	59
Figura 20	Percepção visual - redução de 5 <i>bits</i> sem adição de ruído	60
Figura 21	Percepção visual - redução de 5 <i>bits</i> com adição de ruído gaussiano	60
Figura 22	Análise níveis de aproximação em uma mesma imagem. a) pássaro com 8 <i>bits</i> b) pássaro com 3 <i>bits</i>	61
Figura 23	Análise níveis de aproximação em uma mesma imagem com ruído gaussiano. a) pássaro com 8 <i>bits</i> b) pássaro com 3 <i>bits</i>	62

Figura 24	Imagem 01 do Benchmark BSD.	72
Figura 25	Imagem 02 do Benchmark BSD.	73
Figura 26	Imagem 03 do Benchmark BSD.	74
Figura 27	Imagem 04 do Benchmark BSD.	74
Figura 28	Imagem 05 do Benchmark BSD.	75
Figura 29	Imagem 06 do Benchmark BSD.	75
Figura 30	Imagem 07 do Benchmark BSD.	76
Figura 31	Imagem 08 do Benchmark BSD.	77
Figura 32	Imagem 09 do Benchmark BSD.	78
Figura 33	Imagem 10 do <i>Benchmark</i> BSD.	78
Figura 34	Imagem 11 do <i>Benchmark</i> BSD.	79
Figura 35	Exemplo de código do filtro gaussiano 2D-redução de 3 bits	81
Figura 36	Continuação do código do filtro gaussiano 2D-redução de 3 bits . . .	82
Figura 37	Continuação do código do filtro gaussiano 2D-redução de 3 bits . . .	83
Figura 38	Transformação arquivo JPG/BMP para .out.	84
Figura 39	Transformação arquivo txt para JPG ou BMP.	85
Figura 40	Recorte das fitas das imagens filtradas	86
Figura 41	Colagem das versões de imagens filtradas	86
Figura 42	Código da implementação do Filtro Gaussiano.	87

LISTA DE TABELAS

Tabela 1	Resumo dos Trabalhos Relacionados	34
Tabela 2	Máscaras utilizadas	45
Tabela 3	Análise da área - tecnologia de @ 65 nm	49
Tabela 4	Análise da potência estática e dinâmica na tecnologia de @ 65nm . .	50
Tabela 5	Comparação entre as área da Arq. em tempo de projeto x Arq. em tempo de execução - tecnologia de @65nm	53
Tabela 6	Média de valores do PSNR x N de <i>bits</i> - Imagens sem ruído	54
Tabela 7	Média de valores do PSNR x N de <i>bits</i> - Imagens com adição de ruído	56
Tabela 8	Comparação geral com os trabalhos relacionados	63

LISTA DE ABREVIATURAS E SIGLAS

ACA	<i>Accuracy Configurable Adder</i>
ACCURUS	<i>Accuracy Configurable Adder Circuit,</i>
ALU	<i>Arithmetic Logic Unit</i>
ARC	<i>Application Resilience Characterization</i>
ASIC	<i>Application-specific Integrated Circuit</i>
BSD	<i>Berkeley Segmentation Dataset</i>
CADE	<i>Configurable Approximate Divider</i>
CI	<i>Integrated Circuit</i>
CLA	<i>Carry Look Adder</i>
CMY	<i>Cyan, Magenta, Yellow</i>
CMYK	<i>Cyan, Magenta, Yellow, Black</i>
CMOS	<i>Complementary Metal-Oxide-Semiconductor</i>
DBI	<i>Dynamic Binary Instrumentation</i>
DES	<i>Dynamic Effort Scaling</i>
DSP	<i>Digital Signal Processor</i>
EITAI	<i>Error Tolerant Adder II</i>
FPGA	<i>Field Gate Programmable Array</i>
EGs	<i>Equivalent Gates</i>
GDA	<i>Gracefully Degrading Adder</i>
HEVC	<i>High-Efficiency Video Coding</i>
HSI	<i>Hue, saturation, intensity</i>
IoT	<i>Internet of Things</i>
JPEG	<i>Joint Photographic Experts Group</i>
LSB	<i>Least Significant Bit</i>
MSB	<i>Most Significant Bit</i>
MSE	<i>Error quadratic medium</i>

PDP	<i>Power Delay Product</i>
PSNR	<i>Peak Signal to Noise Ratio</i>
QCS	<i>Quality Configurable System</i>
RAM	<i>Random Access Memory</i>
RCA	<i>Ripple Carry Adder</i>
RGB	<i>Red, Green, Blue</i>
RTL	<i>Register Transfer Level</i>
SDL	<i>Significance Driven Logic Compression</i>
SoC	<i>System-on-a-chip</i>
TFC	<i>Toggle Count Format</i>
VCD	<i>Value Change Dump</i>
VHDL	<i>Very High-Speed Integrated Circuits Hardware Description Language</i>
VLSA	<i>Variable Latency Speculative Adder</i>
VSH	<i>Human Visual System</i>
VLSI	<i>Very Large Scale Integration</i>

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Objetivo Geral	16
1.2	Objetivos Específicos	17
1.3	Organização do Texto	17
2	CONCEITOS FUNDAMENTAIS	18
2.1	Processamento de Imagens	18
2.2	Computação Aproximada	25
3	TRABALHOS RELACIONADOS	28
4	ARQUITETURAS PROPOSTAS PARA O FILTRO GAUSSIANO 2D 5X5	35
4.1	Projeto de arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto	37
4.2	Projeto do Filtro Gaussiano 2D com precisão configurável em Tempo de Execução	38
4.3	Modelagem do <i>hardware</i> em <i>software</i>	39
5	RESULTADOS	47
5.1	Resultados de síntese das arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto	48
5.2	Resultados de síntese das arquiteturas do filtro gaussiano 2D com precisão configurável em tempo de execução	50
5.3	Resultados da análise de qualidade da modelagem do <i>hardware</i> em <i>software</i>	53
5.4	Comparação com o Estado da Arte	62
6	CONCLUSÃO E TRABALHOS FUTUROS	64
6.1	Publicações	65
	REFERÊNCIAS	67
7	APÊNDICE A - CONJUNTO DE IMAGENS COM LARGURA DE 8 BITS DO BENCHMARK BSD	72
8	APÊNDICE B - CÓDIGOS UTILIZADOS PARA A IMPLEMENTAÇÃO EM <i>PYTHON</i> DA MODELAGEM DE AVALIAÇÃO DE INATIVAÇÃO DO RESET ENTRE MUDANÇAS DE CONFIGURAÇÕES	80

9	APÊNDICE C - CÓDIGOS <i>PYTHON</i> UTILIZADOS PARA A IMPLEMENTAÇÃO DA MODELAGEM DE ANÁLISE DE DIFERENTES CONFIGURAÇÕES EM UMA MESMA IMAGEM E INSERÇÃO DE RUÍDO GAUSSIANO	86
----------	---	-----------

1 INTRODUÇÃO

As tecnologias digitais estão cada vez mais presentes no dia-a-dia das pessoas, oferecendo uma variedade de dispositivos portáteis, tais como *smartphones*, *tablets* e *laptops*. Cada um destes dispositivos, por sua vez, apresentam aplicações com diferentes funções, tais como redes sociais e plataformas de *streaming* de conteúdo multimídia.

A maior parte do conteúdo multimídia que trafega pela internet está na forma de vídeos e imagens, que são transferidos através de dispositivos digitais. Pesquisas recentes sobre redes de dados móveis informam, que até o ano de 2023 espera-se quase 30 bilhões de dispositivos conectados na internet (CISCO, 2020).

No ano de 2019, um estudo apontou que 80% dos usuários que acessam a internet usam alguma rede social através de um *smartphone* e, com isso, acessam e transferem pela internet imagens e vídeos. Como consequência, isso acarretou em um aumento de 1000% na quantidade de dados consumidos em dispositivos portáteis nos últimos anos (NOGUEIRA, L., 2019).

O uso massivo de diversos aplicativos acaba fazendo com que esses dispositivos digitais tenham um alto custo em esforço computacional e, também exijam grande quantidade de memória para armazenamento das informações dos usuários. Além disso, o projeto de dispositivos portáteis precisa considerar questões de eficiência energética para melhor aproveitamento das baterias. Desse modo, emergem algumas categorias de aplicações frequentemente utilizadas em dispositivos portáteis tais como: i) multimídia (*i.e.*, relativo a fotos e vídeos) e ii) de comunicações (*i.e.*, relativo a transferência de fotos, vídeos e arquivos em geral).

Ademais, o emergente cenário de IoT (*Internet of Things*) (ATZORI; IERA; MORABITO, 2010) (KAMILARIS; PITSILLIDES, 2016), faz com que os dispositivos precisem interagir com os usuários, havendo então, uma grande quantidade de dados sendo coletada a partir da interação entre humanos e dispositivos. A redução da dimensão do transistor CMOS e da tensão permitiu maior fator de integração de funcionalidades em uma mesma área e a cada nova tecnologia são impostos maiores desafios aos projetistas evidenciando a necessidade por soluções energeticamente eficientes (STROLLO; ESPOSITO, 2018).

Diante desse cenário, uma estratégia apropriada para o gerenciamento da dissipação

de potência dinâmica é a técnica de *clock gating*. Essa técnica permite desativar o sinal do *clock* nos registradores em ciclos onde ele não é necessário, reduzindo então, a dissipação de potência (WU; PEDRAM; WU, 2000). Soares (SOARES et al., 2019) indica que o cenário de alto esforço computacional é desafiador para reduzir a dissipação de potência e fornecer alto desempenho computacional ao se projetar um *System-on-a-chip* (SoC), geralmente presente em dispositivos móveis.

O crescente processamento de dados originados de aplicações via serviços online, juntamente com tráfego de informações, como por exemplo, imagens e vídeos, apresenta o desafio de acelerar os cálculos complexos para essas aplicações. Com isso, busca-se melhorar o tempo de resposta e/ou reduzir a dissipação de potência, tornando a computação mais eficiente, assim como a durabilidade do tempo da bateria desses sistemas portáteis. Além disso, a computação exata ou de alta precisão nem sempre é necessária para uma gama de aplicações que toleram pequenos erros, não tendo, portanto, um efeito significativo ao se utilizar esses resultados aproximados (JIANG et al., 2020).

Nesse escopo, emerge o paradigma da computação aproximada, tendo a finalidade de aumentar o desempenho do sistema e/ou de reduzir os recursos de área e dissipação de potência em circuitos digitais. Para este tipo de aplicação é possível aceitar reduções na qualidade dos resultados e imprecisões (REDA; SHAFIQUE, 2018). Dessa forma, busca-se projetar um *hardware* aproximado capaz de aumentar a eficiência energética, ainda que com perdas na exatidão e qualidade da aplicação (SOARES et al., 2019).

Nesse contexto, os projetistas de circuitos integrados enfrentam algumas restrições tecnológicas, como por exemplo: o fim da escala de Dennard impactando severamente no tempo de vida útil de cada tecnologia (STROLLO; ESPOSITO, 2018). Ademais, pode-se citar as densidades de potência *on-chip* que estão subindo a níveis não mais sustentáveis. Como resultado disso, muitos sistemas computacionais *on-chip* não conseguem operar em altos níveis de desempenho (XU; MYTKOWICZ; KIM, 2015).

Faz-se necessário desenvolver novas abordagens de projeto para se processar uma quantidade cada vez maior de dados, aumentando-se o desempenho e melhorando a eficiência energética dos dispositivos digitais (STROLLO; ESPOSITO, 2018). O *trade-off* entre energia e precisão torna-se relevante para uma gama de aplicações tolerantes a erros, como aprendizado de máquina e processamento multimídia.

Venkataramani (VENKATARAMANI et al., 2015) apresenta algumas propriedades que definem uma aplicação resiliente às aproximações: i) existir um resultado preciso, mas também ter-se, uma variedade de resultados imprecisos aceitáveis e ii) lidar com dados ruidosos, ou seja, aplicações multimídia (codificação de vídeo, processamento de imagens, etc) podem ser classificadas como tolerantes às aproximações. Isso porque estão intrinsecamente interligados ao sistema visual humano que não possui a capacidade de reconhecer minimamente arredondamentos numéricos decorrentes da computação aproximada.

Eventualmente, o uso da computação aproximada em imagens pode causar perda de qualidade (ALMOHAMMAD; GHINEA, 2010), acarretando no estabelecimento de um *trade-off* entre a precisão e a eficiência energética. Com base nisso há tendências de projeto que habilitam a configuração da precisão em tempo de execução, dependendo de requisitos do sistema. Arquiteturas configuráveis são uma boa alternativa para atender requisitos que envolvam qualidade e energia das arquiteturas de processamento de imagem e vídeo.

O projeto das arquiteturas configuráveis permite a distinta configuração dinâmica em aplicações. Entre as técnicas adotadas para realizar o gerenciamento de energia nas arquiteturas configuráveis, as mais frequentemente adotadas na literatura são: 1) múltiplas tensões de alimentação, 2) escala dinâmica de tensão e frequência; 3) utilização de aceleradores de *hardware* com aproximações configuráveis (AKBARI et al., 2018).

Muitas das aplicações multimídia usam o filtro gaussiano 2D (GONZALEZ; WOODS, 2009). Esse filtro desfoca ou borra uma imagem, removendo, assim, os artefatos indesejáveis para aplicações de processamento de imagens. Algumas pesquisas apresentam arquiteturas definidas em tempo de projeto, como a de Ye (YE et al., 2013), Soares (SOARES et al., 2019), Qiqieh (QIQIEH et al., 2018) e Qiqieh (QIQIEH et al., 2017) denotam abordagens aproximadas do filtro gaussiano 2D, realizando uma exploração do *trade-off* entre qualidade e precisão, no escopo de computação aproximada.

Arquiteturas configuráveis em tempo de execução são exploradas e propostas para diferentes objetivos na literatura, por exemplo, citam-se: Soares (SOARES; COSTA; BAMPI, 2019), Venkataramani (VENKATARAMANI et al., 2015), Kahng (KAHNG; KANG, 2012), Benara (BENARA; PURINI, 2016) e Imani (IMANI et al., 2019). Um exemplo de aplicação de arquiteturas configuráveis é a adaptação da qualidade das aplicações de um *smartphone* quando a bateria atinge um nível inferior a determinado limite (BENARA; PURINI, 2016).

Adicionalmente, a aceleração de filtros gaussianos e, também, propostas de aproximação são levados em consideração em vários trabalhos estudados. Muitas dessas pesquisas ilustram circuitos configuráveis em tempo de execução visando obter eficiência energética, sem que haja prejuízo na qualidade da saída de suas aplicações. Sendo assim, este assunto ainda encontra-se em escopo de pesquisa aberto e ainda existem contribuições a serem exploradas nesse âmbito.

1.1 Objetivo Geral

O objetivo desta dissertação é explorar técnicas de computação aproximada para o projeto de aceleradores de *hardware* do filtro gaussiano 2D, com a finalidade de explorar o *trade-off* entre qualidade e eficiência energética dos dispositivos digitais.

1.2 Objetivos Específicos

Além do objetivo geral, este trabalho também tem como objetivos específicos:

- Pesquisar técnicas de computação aproximada no contexto deste trabalho;
- Analisar o impacto da computação aproximada em termos de qualidade para o processamento digital de imagens;
- Realizar uma exploração do espaço de projeto através da redução da largura de *bits* dos filtros gaussianos 2D;
- Projetar aceleradores de *hardware*, ou seja, um filtro gaussiano 2D com precisão configurável em tempo de execução.
- Analisar a qualidade da filtragem sob diferentes configurações aproximadas, considerando a inserção de ruído gaussiano no *benchmark*.

1.3 Organização do Texto

O presente trabalho está organizado da seguinte forma: o Capítulo 2 apresenta os conceitos fundamentais de processamento de imagens e computação aproximada. Os trabalhos relacionados que fundamentam a base teórica deste estudo são descritos no Capítulo 3. A seguir, o desenvolvimento das arquiteturas para o filtro gaussiano 2D 5x5 é mostrado no Capítulo 4. Os resultados de *hardware* e *software* são apresentados e discutidos no Capítulo 5. Finalmente, as conclusões, trabalhos futuros e publicações são reportados no Capítulo 6.

2 CONCEITOS FUNDAMENTAIS

Este trabalho abrange conhecimentos das áreas de processamento digital de imagens e de computação aproximada. Para ampliar o entendimento deste trabalho e suas aplicações, estes conceitos introdutórios serão apresentados como base para compreensão de decisões de projeto adotadas no desenvolvimento das arquiteturas do filtro gaussiano 2D.

2.1 Processamento de Imagens

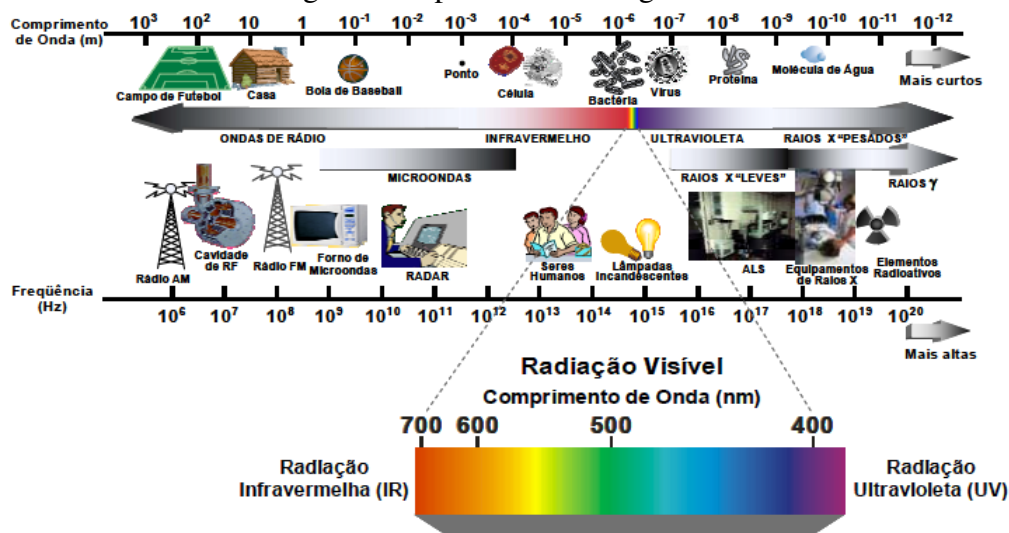
O processamento digital de imagens pode ser aplicado em diversas áreas, e isso requer uma forma de organização como, por exemplo, categorizar as imagens de acordo com sua fonte (*i.e.*, visual, foto, raios X, etc) (GONZALEZ; WOODS, 2009). A principal fonte de energia para imagens é o espectro eletromagnético. Adicionalmente, existem outras fontes como a acústica, ultrassônica e a eletrônica, bem como imagens sintéticas geradas num computador para modelagem e suavização.

A luz forma-se através de uma radiação eletromagnética e apresenta um comportamento ondulatório devido à sua frequência f e comprimento de onda γ . O sistema visual humano (SVH) se estende aproximadamente na faixa do espectro eletromagnético de 400 a 770nm, e, a essa faixa espectral, dá-se o nome de luz visível. Os valores de comprimento de onda fora dessa faixa não são perceptíveis ao olho humano (QUEIROZ; GOMES, 2006).

O espectro de cores pode ser dividido em seis amplas regiões, sendo elas: violeta, azul, verde, amarelo, laranja e vermelho. (GONZALEZ; WOODS, 2009). A Figura 1 apresenta o espectro eletromagnético que representa a distribuição da intensidade da radiação eletromagnética em função do comprimento de onda e a frequência, destacando-se a faixa que o olho humano consegue visualizar, conforme já descrito em Queiroz (QUEIROZ; GOMES, 2006). Através da análise do espectro eletromagnético, pode-se afirmar que os seres humanos possuem habilidades perceptivas e visuais limitadas (GUPTA et al., 2012). Em contrapartida, os equipamentos de geração de imagens cobrem quase todo espectro eletromagnético variando de comprimento de onda γ até ondas de rádio, podendo ope-

rar em imagens geradas por ultrassom, microscopia eletrônica e imagens geradas por um computador (GONZALEZ; WOODS, 2009).

Figura 1: Espectro Eletromagnético.



Fonte: (QUEIROZ; GOMES, 2006)

Durante anos, o SVH vem sendo um objeto de pesquisa voltado para a análise de imagens. Observou-se um conjunto de propriedades detectadas pela visão humana como cor, textura, orientação de contornos e padrões de movimento. Isso está diretamente relacionado às características que permitem aos humanos caracterizarem uma imagem (ALEXANDRE; TAVARES, 2007).

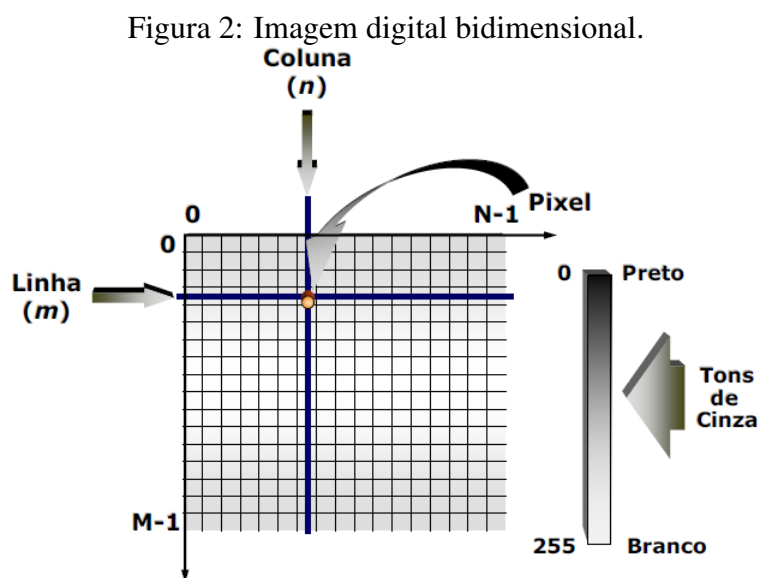
Alguns objetos emitem luz visível, logo são mais perceptíveis em função da soma de suas cores espectrais emitidas. Esse processo é denominado de processo aditivo. O processo aditivo é uma combinação variável em proporção de componentes monocromáticas das faixas espectrais às sensações de cor verde, vermelho e azul, estas que são responsáveis pela formação de todas as demais cores captadas pelo olho humano (QUEIROZ; GOMES, 2006).

A visão colorida do ser humano dá-se pela existência dos cones, que são pequenos sensores presentes nos olhos. Afirmar-se através de experimentos, que existem de 6 a 7 milhões de cones no olho humano. Eles podem ser divididos em três categorias principais de detecção: vermelho, verde e azul. Com isso, observou-se que 65% de todos os cones são sensíveis à luz vermelha, 33% são sensíveis à luz verde e apenas 2% são sensíveis à luz azul (GONZALEZ; WOODS, 2009).

Uma imagem monocromática pode ser definida por uma função de duas dimensões $f(x,y)$, onde x e y são coordenadas espaciais e f refere-se à amplitude em qualquer par de coordenadas (x,y) , no qual denomina-se intensidade ou nível de cinza da imagem naquele ponto (GONZALEZ; WOODS, 2009). Quando x , y e os valores de intensidade f forem finitos e discretos, tem-se o que se denomina de "imagem digital". A imagem digital

é composta por elementos denominados de *pixels*, no qual um *pixel* é um componente básico de uma imagem (GONZALEZ; WOODS, 2009).

A Figura 2 apresenta a notação matricial para a localização de um *pixel* em um arranjo bidimensional. Observa-se que m representa a linha da matriz em que o *pixel* se encontra, enquanto que n denota a coluna. Logo, para uma imagem digital de M linhas e N colunas, o índice m irá variar de 0 a $M-1$, enquanto que n de 0 a $N-1$ (QUEIROZ; GOMES, 2006).



Fonte: (GONZALEZ; WOODS, 2009)

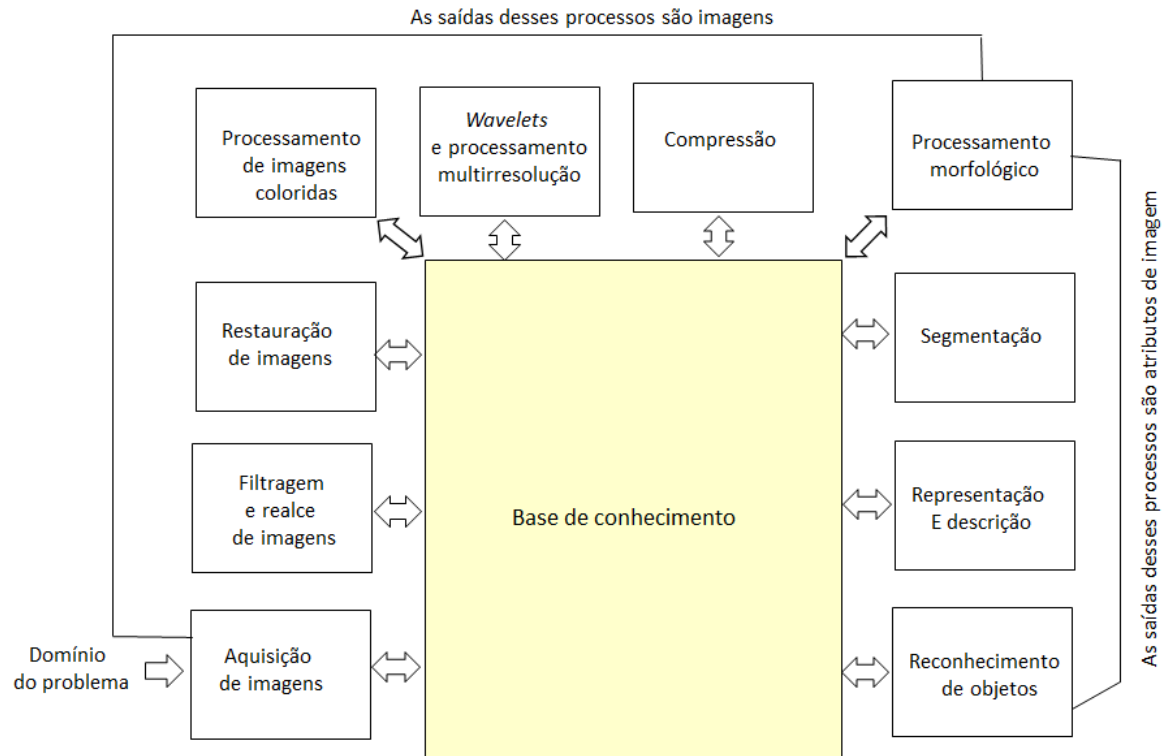
Gonzalez (GONZALEZ; WOODS, 2009) comenta sobre um paradigma útil que considera três tipos de processos computadorizados: baixo, médio e alto. O primeiro envolve operações primitivas, ou seja, pré-processamento de uma imagem para reduzir ruído, aprimoramento de contraste e nitidez de uma imagem. Em suma, esse processo recebe como entrada uma imagem e produz, como saída, imagens.

O médio trata de tarefas como segmentação, onde a imagem é particionada em regiões, descrição de objetos para redimensioná-los a um formato adequado para ser processado por um computador e reconhecimento de objetos individuais. Resumindo, trata-se de um processo que recebe imagens como entrada, porém, sua saída é formada por atributos extraídos dessas imagens, como por exemplo, bordas, contornos, etc. O último processo, "reconhece" um conjunto de objetos, desempenhando funções cognitivas associadas à visão (GONZALEZ; WOODS, 2009).

Nesse contexto, o processamento digital de imagens está intimamente ligado ao desenvolvimento de computadores digitais, pois as imagens digitais necessitam ser armazenadas, processadas, exibidas e transmitidas. Seguidamente, emergem os conceitos sobre os processos computadorizados relacionados ao processamento digital de imagens. A Figura 3 apresenta um diagrama proposto por Gonzalez (GONZALEZ; WOODS, 2009) que

transmite a ideia de todas as possíveis metodologias que podem ser aplicadas a imagens com diferentes objetivos.

Figura 3: Etapas do Processamento Digital de uma Imagem.



Fonte: (GONZALEZ; WOODS, 2009)

A **aquisição de imagens** converte uma imagem em uma representação numérica adequada ao processamento digital subsequente. Essa etapa é composta por dois elementos principais: i) um dispositivo físico sensível a uma faixa de energia no espectro eletromagnético, produzindo em sua saída um sinal elétrico proporcional ao nível de energia detectado; e, ii) a conversão do sinal elétrico analógico em informações digitais, sendo, portanto, o digitalizador (MARQUES FILHO; NETO, 1999).

O **realce de imagens** é o processo de manipulação de uma imagem para que o resultado se torne mais adequado que o original para uma determinada aplicação. A fase de **restauração de imagens** também tenta melhorar visualmente uma imagem, porém de forma objetiva, ou seja, essa etapa se baseia em modelos matemáticos ou probabilísticos de degradação de imagens (GONZALEZ; WOODS, 2009).

O **processamento de imagens** é composto por técnicas que processam *pixels*. Neste sentido, uma das técnicas de processamento de imagens para compressão destina-se à **Wavelets** que constituem os fundamentos para a representação de imagens em vários níveis de resolução. A **compressão** trabalha com técnicas de redução de dados necessárias para armazenar e/ou transmitir uma imagens com maior eficiência no tráfego. Um exemplo de técnica de compressão é o padrão de compressão de imagens JPEG (*Joint Photographic Experts Group*) (GONZALEZ; WOODS, 2009).

O **processamento morfológico** lida com ferramentas para extração de atributos de imagens. Os procedimentos de **segmentação** dividem uma imagem em partes ou objetos constituintes sendo, portanto, uma das tarefas mais difíceis do processamento de imagens. Adicionalmente, a representação e a descrição partem do resultado de um estágio de segmentação, onde normalmente os dados primários encontram-se em forma de *pixels* (GONZALEZ; WOODS, 2009).

Os **Filtros de imagens** são ferramentas capazes de transformar cada *pixel* de uma imagem, de acordo com alguma informação desta imagem, tais como seus níveis de cinza e suas frequências. Eles são utilizados para corrigir imperfeições em imagens, como ruídos, ou também podem ser usados para realçar uma imagem. A partir disso, existem dois grupos principais de técnicas de filtragem: filtragem no domínio espacial e filtragem no domínio da frequência (MARQUES FILHO; NETO, 1999).

A Filtragem no domínio da frequência torna certas operações mais simples porque manipula separadamente as altas e baixas frequências. Ao se considerar bordas e outras mudanças abruptas de intensidade dos componentes de alta frequência é possível obter um aguçamento de imagens no domínio da frequência através da filtragem passa-alta. Esse tipo de filtragem atenua os componentes de baixa frequência, não afetando informações de alta frequência na transformada de Fourier (GONZALEZ; WOODS, 2009).

A Filtragem espacial atua diretamente nos *pixels* de uma imagem. As técnicas no domínio espacial são computacionalmente mais eficientes requerendo, portanto, menos recursos de processamento para serem efetuadas. A Equação 1 apresenta os processos no domínio espacial, expresso por (GONZALEZ; WOODS, 2009):

$$g(x, y) = T[f(x, y)] \quad (1)$$

onde $f(x, y)$ é a imagem de entrada, $g(x, y)$ é a imagem de saída, e T é um operador em f definido em uma vizinhança do ponto (x, y) . Com isso, o operador pode ser aplicado a uma única imagem ou um conjunto de imagens. O processo da Equação 1 parte da movimentação da origem da vizinhança para a próxima posição e o procedimento repete-se até gerar o próximo valor da imagem de saída g (GONZALEZ; WOODS, 2009).

Geralmente, o processo inicia no canto superior esquerdo da imagem de entrada e avança *pixel* por *pixel* em uma varredura horizontal, uma linha por vez. O procedimento ignora os vizinhos externos nos cálculos especificados por T ou preenche a imagem com uma borda de zeros ou outros valores de intensidade predefinidos. Todo esse procedimento realiza então, a filtragem espacial e, a operação predefinida que acontece na vizinhança é denominada de filtro espacial, no qual o tipo de operação realizada na vizinhança determina o processo de filtragem (GONZALEZ; WOODS, 2009).

A suavização em imagens é importante, pois realiza a função de redução de ruídos presentes em uma imagem através do borramento da mesma. Sua finalidade é diminuir o

ruído, mantendo as bordas da imagem, encaixando-se na etapa de pré-processamento de uma imagem (MARQUES FILHO; NETO, 1999). A saída gerada por um filtro espacial linear de suavização é a média dos *pixels* contidos na vizinhança da máscara de filtragem. Esses filtros são chamados de filtros de média ou também de filtros passa-baixa. Pode-se implementar a filtragem passa-baixa no domínio do espacial utilizando uma máscara com todos os coeficientes positivos. Ademais, quanto maior foi o valor da máscara utilizada, maior será o borramento aplicado à imagem.

O **Filtro Gaussiano** (GONZALEZ; WOODS, 2009) é utilizado para borrar ou desfocar uma imagem na qual ele é aplicado, tendo o objetivo de reduzir os ruídos presentes em uma imagem. O ruído descreve um efeito que produz degradação aleatória e moderada da qualidade de uma imagem, dando um toque arenoso à imagem (AMBROSE; HARRIS, 2009). Dessa forma, a imagem apresenta o aspecto de suavização, ou seja, como se estivesse sendo vista através de uma lente fora do foco (GONZALEZ; WOODS, 2009)(JESUS; COSTA JR, 2015).

Algumas aplicações de visão computacional presentes em áreas como: agricultura, transporte, entre outras (SOARES; COSTA; BAMPI, 2019) utilizam o filtro gaussiano, como um atenuador de ruídos indesejáveis na detecção de bordas em imagens. Este filtro é utilizado para borrar ou desfocar a imagem na qual ele foi aplicado, reduzindo então, os ruídos presentes numa imagem.

A suavização gaussiana é aplicada no estágio de pré-processamento de uma imagem para enaltecer sua estrutura em diferentes escalas (GONZALEZ; WOODS, 2009) (JESUS; COSTA JR, 2015). Uma aplicação que utiliza o filtro gaussiano como etapa de pré-processamento é a detecção de bordas *Canny* (CANNY, 1986). Os filtros gaussianos apresentam algumas características úteis para o processamento de imagens, tais como (PEDRINI; SCHWARTZ, 2008):

- Em matrizes bidimensionais, o grau de suavização é similar em ambas as direções, devido a função gaussiana possuir simetria de rotação;
- A suavização da imagem é realizada através da substituição de cada *pixel* por uma média ponderada dos *pixels* vizinhos, sendo que o peso de cada vizinho decresce com a distância do *pixel* central;
- O grau de suavização está relacionado com a variância σ^2 , pois quanto maior seu valor, maior será a largura do filtro gaussiano, e, conseqüentemente, será maior o grau de suavização.

O Filtro Gaussiano possui coeficientes e tamanho de máscara idêntico ao utilizado na implementação de *hardware*. A suavização gaussiana utiliza máscara de coeficientes provenientes de uma função gaussiana bidimensional, conforme apresenta a Eq. 2, onde x é a distância da origem no eixo horizontal, y é a distância da origem no eixo vertical, e σ é

o desvio padrão da distribuição gaussiana dos *pixels* que compõem a imagem (PEDRINI; SCHWARTZ, 2008). Pode-se representar a função G de forma discreta, ou seja, através da matriz de coeficientes, como:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (2)$$

$$G = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{ou} \quad G = 1/273 \begin{bmatrix} 1 & 4 & 7 \\ 4 & 16 & 26 \\ 7 & 26 & 41 \end{bmatrix}$$

A análise da qualidade em filtros gaussianos 2D é realizada através da métrica objetiva *Peak Signal to Noise Ratio* (PSNR) (HORE; ZIOU, 2010). Essa métrica utiliza critérios numéricos para avaliar os níveis de degradação em imagens. Dada uma imagem de 8 *bits* f e uma imagem teste g , ambas de tamanho $M \times N$, o PSNR entre f e g é definido pela Eq. 3, onde *Mean Squared Error* (MSE) é definido na Eq. 4.

$$PSNR(f, g) = 10 \log_{10} (255^2 / MSE(f, g)) \quad (3)$$

$$MSE(f, g) = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (f_{ij} - g_{ij})^2 \quad (4)$$

O valor PSNR aproxima-se do infinito à medida que o MSE se aproxima de zero e isso mostra que um valor PSNR mais alto fornece uma maior qualidade. Por outro lado, um valor menor de PSNR implica em substancial diferença numérica entre as imagens testadas. Hore (HORE; ZIOU, 2010) aponta que o MSE e, consequentemente o PSNR, possuem o melhor desempenho na avaliação da qualidade objetiva de imagens ruidosas.

Diante desse cenário, foram elaborados testes para análise considerando imagens sem ruído e com adição de ruído. O PSNR pode ser usado para entender melhor sua diferença entre similaridade em degradações comuns, como por exemplo, o ruído gaussiano e compressão de imagens.

Assim sendo, o processamento digital de imagens abrange o processo de filtragem gaussiana. A utilização do filtro gaussiano em aplicações de visão computacional permite o uso de aproximações ou erros, devido às limitações do sistema visual humano, anteriormente comentadas. Logo, o uso de técnicas de computação aproximada acopladas ao filtro gaussiano em imagens podem ser exploradas em projetos que melhorem o *trade-off* entre eficiência energética e qualidade, permitindo aproximações que não interferem de maneira substancial na aplicação. Logo, torna-se imprescindível estudar esses conceitos para se obter uma base desses conhecimentos.

2.2 Computação Aproximada

O conceito de computação aproximada pode ser aplicado para: acelerar cálculos em diversas aplicações, minimizar área do *chip*, diminuir a dissipação de potência, além de otimizar outros parâmetros do sistema (REDA; SHAFIQUE, 2018). A computação aproximada pode ser aplicada em dispositivos digitais cujas aplicações computacionais não necessitem calcular uma resposta numérica precisa, mas sim, aceitar a produção de resultados que são "bons o suficiente".

A computação aproximada relaxa a equivalência numérica em aplicações tolerantes a erros para equilibrar o *trade-off* entre consumo de energia e qualidade (XU; MYT-KOWICZ; KIM, 2015). Dessa forma, a qualidade desses resultados torna-se eficaz para alcançar um nível de experiência aceitável pelo usuário (MITTAL, 2016). Nesse sentido, a computação aproximada utiliza de técnicas para investigar a resiliência intrínseca dos aplicativos a fim de obter eficiência energética em dispositivos digitais (VENKATARAMANI et al., 2015).

De acordo com Reda e Shafique (REDA; SHAFIQUE, 2018), a computação aproximada é relevante para aplicações que utilizam análise de dados, computação científica, aprendizado de máquina e processamento multimídia. Tais aplicações são substancialmente encontradas em sistemas computacionais da atualidade. São exemplos de aplicações tolerantes a erros, problemas de:

- busca (procura de respostas imprecisas em consultas imprecisas);
- entrega proativa de informações (propagandas, ofertas, notícias, alertas);
- mídia (áudio, vídeo ou imagens);
- processamento de dados (decisões rápidas baseadas em dados possivelmente incompletos).

A partir dessa gama de aplicações complexas existentes hoje em dia, há uma demanda de um enorme esforço computacional para atender a funcionalidade desses aplicativos, tornando ascendente o consumo de energia dos aparelhos onde estes programas são executados (REDA; SHAFIQUE, 2018). Nesse aspecto, a computação aproximada surge como um tópico a ser pesquisado, pois ela explora vários níveis de abstração, tanto em *hardware* como em *software*, com o objetivo de reduzir o consumo energético de sistemas digitais (BARUA; MONDAL, 2019). A aplicação de técnicas de computação aproximada apresenta dois aspectos importantes a serem pensados e analisados:

1. Determinar o quanto de aproximação pode ou não ser segura para um sistema ou aplicativo. Com isso, pode-se, por exemplo, restringir um sistema a permitir um erro dentro de um limite definido utilizando métricas de qualidade para diversas

aplicações, como processamento de imagens, mineração de dados, aprendizado de máquina e etc (BARUA; MONDAL, 2019).

2. O conceito de comunicação aproximada, o qual abrange como três principais métodos de aproximação e melhoria na precisão de comunicação de dados: compressão, sincronização e previsão. O principal desafio de pesquisa é a aplicabilidade de uma comunicação aproximada em aplicativos da vida real, considerando-se qualidade e desempenho (BARUA; MONDAL, 2019).

Para utilizar-se das técnicas de aproximação em *software*, os programadores declaram quais partes do programa podem ser computadas de maneira aproximada. Em *hardware*, opta-se por métodos de aproximação específica no circuito a serem aplicados. A classe dos circuitos mais utilizados para tal propósito são os circuitos aritméticos, isto é, somadores e multiplicadores (SEKANINA; VASICEK; MRAZEK, 2017).

A maioria das camadas de abstração da computação, ou seja, linguagens de programação, compiladores, sistemas operacionais, arquitetura e circuitos, são projetadas para lidar com correções. Entretanto, os algoritmos e aplicativos são traduzidos em *softwares* ou implementações de *hardware*, no qual há partes do código e/ou circuito que toleram resultados aproximados ou "suficientemente bons", emergindo o conceito de correção estrita (VENKATARAMANI et al., 2015).

A computação aproximada aplicada em *hardware* apresenta como principal abordagem a redução da precisão na computação em favor da eficiência energética. Dessa forma, desenvolvem-se circuitos mais simples com o objetivo de reduzir o tempo do caminho crítico e/ou fazer com que esses circuitos consumam menos energia (SOARES et al., 2019). Quando projeta-se um *hardware* aproximado, deve-se estar ciente que ele incluirá circuitos modificados que irão tender a apresentar resultados aproximados e/ou imprecisos (BARUA; MONDAL, 2019). Uma abordagem de projeto sistemática para a computação aproximada em *hardware* apresenta dois princípios:

1. Esforço escalável: identificar mecanismos em diferentes níveis de abstração de um projeto (circuito, arquitetura e algoritmo) para gerenciamento de aproximações na implementação. Isso favorece maior eficiência energética mantendo-se um nível aceitável ou quase idêntico de qualidade do resultado antes apresentado (CHIPPA et al., 2013).
2. Resiliência intrínseca: analisar a capacidade de até onde pode-se produzir resultados de qualidade aceitáveis (CHIPPA et al., 2013). Através disso, surge uma exploração da computação aproximada no projeto de sistemas digitais, relacionados a soluções de aceleradores de *hardware* para o processamento de imagens. (CHIPPA et al., 2013).

Algumas técnicas de computação aproximada foram adotadas na literatura, para realizar economia de energia em projetos que utilizam arquiteturas configuráveis, para uma variedade de aplicações (SEKANINA, 2016). Arquiteturas configuráveis permitem uma aproximação dinâmica, ou seja, uma configuração dinâmica no nível do circuito. Com isso, o resultado da qualidade atende aos requisitos das variáveis, um exemplo disso, é a compressão de imagens (SEKANINA, 2016). A partir disso, as técnicas de computação aproximada mais frequentes são:

1. Uso de menor número de iterações; (RODRIGUES et al., 2019);
2. Redução da largura de *bits* (RUBIO-GONZÁLEZ et al., 2013).

Nesse âmbito, técnicas de computação aproximada visam obtenção de desempenho, eficiência energética, além de tratar diversos níveis de qualidade de aplicações tolerantes a erros. Aplicações voltadas ao processamento digital de imagens aceitam aproximações ou erros, devido às limitações do sistema visual humano, anteriormente comentadas. Logo, torna-se imprescindível estudar esses conceitos para se ter uma base desses conhecimentos.

3 TRABALHOS RELACIONADOS

Estudou-se alguns trabalhos na literatura relacionados a arquiteturas aproximadas de filtro gaussiano 2D com e sem configurabilidade em termos de precisão, com o claro objetivo de estabelecer o estado da arte neste assunto. Esta seção resume trabalhos relacionados, enfatizando outras abordagens e técnicas configuráveis propostas para reduzir o consumo de energia em circuitos de processamento de vídeo e imagem.

Os somadores são considerados a base de qualquer aplicativo de computação. Nos últimos anos, surgiram diferentes propostas de somadores aproximados para impulsionar a eficiência energética do projeto digital CMOS. Na computação aproximada de nível aritmético, há três classes de somadores aproximados: i) somadores orientados à potência elétrica, ii) somadores orientados ao desempenho e iii) somadores de precisão configurável.

Os trabalhos de (KAHNG; KANG, 2012), (BENARA; PURINI, 2016) e (JULIO et al., 2015) são baseados em blocos de precisão configurável e realizam detecção e correção de erros. Dessa forma, pertencem à classe iii de somadores aproximados. Geralmente, essa classe de somadores é denotada por somadores especulativos e possuem latência variável por possuírem uma estrutura de *pipeline* para a possível correção de erros.

O somador configurável chamado de *Accuracy Configurable Adder* (ACA) (KAHNG; KANG, 2012) explora os conceitos de computação aproximada, no qual a precisão dos resultados é configurada durante o tempo de execução. Portanto, o somador ACA pode operar de maneira adaptativa, ou seja, do modo aproximado (impreciso) ao preciso. Os resultados apontam uma melhoria no desempenho computacional e uma redução de 51,6% na energia quando comparados ao somador preciso convencional. Para a análise de perda de qualidade, a métrica PSNR foi explorada. Foi obtido um PSNR de 24,5 dB, o que representou um melhor valor de PSNR entre os outros somadores aproximados analisados. Com base nisso, os autores destacaram que o ACA pode ser usado em aplicações cuja precisão pode ser configurada, uma vez que seu estudo obteve uma troca entre desempenho/potência e qualidade.

Outro trabalho que explorou circuitos aritméticos aproximados foi o de Benara (BENARA; PURINI, 2016) que implementou um somador aproximado denominado *Accu-*

racy Configurable Adder Circuit (ACCURUS), muito semelhante ao ACA (KAHNG; KANG, 2012), mas que apresenta uma melhora no modo de correção dos erros, ou seja, na precisão dos primeiros estágios do *pipeline*, a correção acontece a partir do *bit* mais significativo (MSB) da mantissa. Utilizou-se a aplicação do filtro gaussiano apenas na imagem Lena, na qual atingiu-se um valor PSNR de 40,90 dB quando comparado com o somador ACA que obteve 25,69 dB. Em síntese, o ACCURUS apresenta resultados semelhantes em energia e área ao somador ACA, destacando-se apenas por apresentar melhor precisão dos resultados.

Para avaliar os níveis de computação aproximada ou perda de precisão no fluxo de dados aritméticos que o filtro gaussiano 2D pode tolerar, em (JULIO et al., 2015) é considerado um somador aproximado proposto para implementações de filtros gaussianos 3x3 e 5x5. O trabalho trata de diferentes níveis de aproximação da topologia *Ripple Carry Adder* (RCA), que fazem parte da árvore de somadores de filtro gaussiano implementada em *hardware* e comparada com a melhor implementação precisa do mesmo filtro. Os resultados mostraram uma redução média de energia de até 40% e 25% para os filtros gaussianos aproximados 3x3 e 5x5, respectivamente, sem comprometer a qualidade das imagens filtradas obtendo relação sinal-ruído de pico PSNR de 36,6 dB.

Visando obter eficiência energética, o trabalho de De Oliveira (OLIVEIRA et al., 2016) estuda um conjunto de somadores aproximados a fim de realizar a implementação de filtros energeticamente eficientes. A partir disso, são explorados os somadores *Copy adder* (GUPTA et al., 2012) e ETAI (ZHU et al., 2010) dentro de uma arquitetura formada por operações de somas e deslocamentos para o processo de filtragem gaussiana e gradiente de imagens. A abordagem adota uma heurística de busca, baseada na magnitude de saída esperada, para combinar diferentes parâmetros de aproximações dentro das arquiteturas. Cada configuração de somadores aproximados com diferentes parâmetros de aproximação são simulados para avaliar a resposta de qualidade desses filtros. Ademais, restrições de qualidade são impostas para selecionar um conjunto de configurações sintetizadas e analisadas em relação a eficiência energética. Esse estudo apresenta, como resultado, máxima redução de energia de cerca de 50%, considerando a tecnologia de 45 nm.

Zhu (ZHU et al., 2010) propõe um Somador Tolerante a Erros II (ETAII). Ao contrário do Somador ACA, esse somador aproximado é composto por blocos independentes não sobrepostos. O ETA II é dividido em m blocos de k bits. O método proposto para atenuar o erro no resultado da soma é o uso de bloco de especulação de *carry* com k bits, baseado no *Carry Look Adder Adder* (CLA). A geração da soma é realizada pelo bloco RCA de k bits no qual a entrada de *carry* é alimentada pelo bloco de especulação anterior. Logo, o tempo do caminho crítico desse somador aproximado é composto de CLA de k bits mais o RCA de k bits. Em outras palavras, quanto maior for o atraso do circuito, maior será a precisão. Em contrapartida aos demais somadores da literatura, Zhu (ZHU

et al., 2010) não apresenta nenhuma análise de aplicação. O circuito é capaz de detectar rigorosas restrições de precisão para obter melhorias no consumo de energia, de aproximadamente 10% e desempenho, sendo que o ETAlI proposto é capaz de obter mais de 60% de melhoria em termos de *Power-Delay Product* (PDP).

A computação aproximada representa uma mudança no paradigma de projetos *Very Large Scale Integration* (VLSI) de baixa potência para multimídias resilientes a erros. Nesse sentido, a multiplicação também é uma operação aritmética fundamental para algoritmos de processamento de dados em processadores de sinais digitais (PSD). Com isso, (RAMYA; MOORTHY, 2018), (QIQIEH et al., 2018) e (QIQIEH et al., 2017) apresentam trabalhos referentes a multiplicadores aproximados que visam eficiência energética e, nessa linha de operações que envolvem alto processamento de dados, Imani (IMANI et al., 2019) propõe um divisor aproximado.

Nesse escopo, Ramya (RAMYA; MOORTHY, 2018) propõe uma arquitetura multiplicadora de multi-precisão configurável em tempo de execução. A arquitetura proposta pode ser configurada para se adaptar a diferentes níveis de precisão para arquiteturas configuráveis de granularidade grossa. No modo aproximado de precisão em *bits*, o multiplicador possui redução de 35% de potência dissipada e, para o projeto de 16 *bits*, oferece 27% de economia energética comparado ao multiplicador exato. A qualidade de um conjunto de imagens de 16 *bits* é avaliada pelas métricas qualitativas PSNR (50 dB) e SSIM (0,9).

O trabalho de Qiqieh (QIQIEH et al., 2017) apresenta um projeto de um multiplicador aproximado de eficiência energética que utiliza de compressão lógica baseada em significância *Significance Driven Logic Compression* (SDLC). A síntese mostrou uma redução no tempo de execução, consumo de energia e área. Para a análise da aplicação, os autores utilizaram uma arquitetura aproximada de filtro gaussiano, com um multiplicador com três níveis diferentes de aproximação e apresentou PSNR de 50,2 dB para a configuração de 2x3 *bits*, além de fornecer uma economia de energia de até 68,3%.

Uma nova versão do multiplicador SDLC para 16 *bits* obteve valores de PSNR entre 30,2 dB a 70,2 dB, ao tratar imagens configuradas de 2 a 8 *bits* (QIQIEH et al., 2018). Diante disso, observa-se valores maiores de PSNR no multiplicador SDLC de 16 *bits*, resultando, portanto, numa melhor qualidade de saída, além de um ganho de 80,1% de economia energética o que leva a uma qualidade aceitável na saída da imagem, quando configurado de 2 a 6 *bits*. Os autores propõem como pesquisas futuras, a implementação desse multiplicador aproximado configurável em tempo de execução.

Uma arquitetura de divisor de ponto flutuante configurável em tempo de execução chamada *Configurable Approximate Divider* (CADE) é descrita em Imani (IMANI et al., 2019). O CADE possui uma alta capacidade de correção de erros durante sua execução, fornecendo uma taxa de erros até 25% menor, além de conter mecanismos para permitir a configuração da abordagem em vários níveis durante o tempo de execução das Unidades

de Processamento Gráfico. A compressão JPEG apresentou aumento de PSNR de 35,1dB (sem ajuste) para 50,0dB quando a solução configurável é usada. Para a aplicação do filtro Blur, o PSNR médio aumenta de 35,2dB (sem ajuste) para 47,08dB. Portanto, a abordagem de ajuste oferece uma oportunidade de alterar o nível de aproximação no tempo de execução, com base nos aplicativos em execução na GPU. Por fim, o CADE provou ser 4,1 vezes mais eficiente em termos energéticos, 1,5 vezes mais rápido, além de ter uma área 1,7 vezes maior em comparação com outros divisores da literatura.

A maioria das aplicações multimídia atuais exige dispositivos com alta eficiência energética devido à necessidade de computação intensa. O Filtro gaussiano 2D é um filtro que demanda alto esforço computacional. Nesse âmbito, alguns trabalhos foram propostos para examinar técnicas de computação aproximada para o filtro gaussiano 2D, como: Ye (YE et al., 2013), Soares (SOARES et al., 2019), Kaushik (KAUSHIK; KUMAR, 2015), Soares (SOARES; COSTA; BAMPI, 2019) e Pham (PHAM-QUOC; THINH, 2017).

Kaushik (KAUSHIK; KUMAR, 2015) propõe um filtro de suavização gaussiana2D aproximado para eficiência energética. A abordagem proposta é baseada em "*Mirror Short Pixel Approximation*" e máscara gaussiana de arredondamento de coeficientes. A abordagem proposta foi modelada em linguagem de alto nível para verificar eficácia na imagem de referência, o que resulta em menor degradação de qualidade desta imagem. O projeto proposto foi realizado no FPGA Virtex 6. Os resultados da simulação mostram redução de área de 85%, de potência de 8% e, uma melhora no desempenho computacional de 58%.

Uma arquitetura aproximada para o filtro de suavização Gaussiano 2D baseado na aproximação de *pixel* é descrita em Ye (YE et al., 2013). Ye realiza uma exploração para um Sistema Configurável de Qualidade (QCS), buscando nortear uma compensação entre o *trade-off* de qualidade da aplicação e esforço computacional de acordo com as necessidades do sistema operacional. Por outro lado, esse trabalho não explora o aspecto configurável com precisão no projeto, possuindo estrutura semelhante ao do somador ETA II. Os resultados apresentam 64% de dissipação de potência em comparação com o filtro gaussiano 2D original, além de ter apresentado PSNR de 14,32dB a 26,57dB para a qualidade das imagens.

Os desafios de integração entre as camadas de abstração são explorados em Soares (SOARES et al., 2019). Este trabalho apresenta diferentes perfis de qualidade, potência e desempenho computacional para filtros gaussiano e gradiente 2D para a realização do algoritmo de detecção de borda de *Canny*. Os resultados de uma tecnologia baseada em célula padrão de 45nm mostram reduções de energia variando de 7,7% a 73,2% para vários níveis de qualidade. O benchmark utilizado é o *Berkeley Segmentation Dataset* (BSD) (MARTIN et al., 2001), em que os resultados indicaram que a solução proposta contribuiu para os estudos de qualidade, consumo de energia e desempenho computacio-

nal. Por outro lado, este trabalho explorou técnicas de aproximação em tempo de projeto para aceleradores de *hardware*.

Em outro momento, Soares (SOARES; COSTA; BAMPI, 2019) propõe uma arquitetura de um filtro gaussiano 2D 5x5 configurável com precisão de granularidade grossa, gerenciando o tamanho da máscara do filtro durante o tempo de execução. A aplicação em análise é o detector de borda *Canny*. Os resultados de energia mostram reduções de até 64%. Como o filtro gaussiano 2D foi considerado para aplicação de detecção de borda, a métrica de qualidade de conformidade de desempenho é adotada em vez do uso de PSNR. Por fim, a exploração de granularidade grossa limita o ajuste refinado da qualidade da aplicação.

Um filtro gaussiano de ponto flutuante adaptável em tempo de execução foi desenvolvido em FPGA por Pham (PHAM-QUOC; THINH, 2017). Ao contrário dessa pesquisa e da maioria encontrada na literatura, Pham (PHAM-QUOC; THINH, 2017) desenvolve a arquitetura em várias famílias de *Field Gate Programmable Array* (FPGA). Como resultado, o núcleo de filtragem, assim chamado pelos autores, pode funcionar a uma frequência máxima de até 206 MHz quando implementado no dispositivo *Xilinx* na família FPGA *Virtex 7*, consumindo poucos recursos disponíveis do FPGA (apenas 3%). A análise de desempenho mostra que o núcleo de filtragem pode executar uma filtragem gaussiana 2D em 48 imagens full-HD (1920 x 1080) por segundo quando implementadas no mesmo dispositivo, permitindo assim, que o núcleo seja integrado em sistemas de tempo real.

A pesquisa de Khandelwal (KHANDELWAL et al., 2020) propõe um acelerador de *hardware* que explora o paralelismo do algoritmo da filtragem laplaciana local no FPGA *Virtex-7*. Para isso, os autores adaptaram o filtro gaussiano de máscara 3x3, o que permitiu um projeto de alto rendimento através da convolução, no qual a taxa de transferência dos dados ocorre tanto em *pipeline* quanto de forma paralela. O cálculo de convolução não requer blocos DSP no FPGA e a função de remapeamento é realizada através de uma tabela de pesquisa, o que economiza recursos de *hardware* e latência. A precisão da implementação foi verificada através da métrica PSNR, em que a precisão foi visualmente indistinguível obtendo um PSNR variando de 30 a 50 dB. Ademais, o acelerador de *hardware* pode processar um 1 megapixel de uma imagem em 534 milissegundos (ms) versus 4 segundos de tempo gasto pela implementação da CPU de 8 núcleos. Com isso, o filtro mostra-se 7,5 vezes mais rápido do que a versão original da CPU.

Otsuka (OTSUKA et al., 2020) implementou um acelerador de *hardware* do filtro gaussiano baseado na transformada discreta do cosseno (DCT), um algoritmo que oferece melhor desempenho em termos de velocidade e precisão. Os autores concentraram sua atenção na instrução *fused multiply-add* (FMA) que realiza operações de multiplicação e adição simultaneamente. A análise da precisão de 32 e 64 *bits* avaliou o desempenho do filtro gaussiano em tempo constante. Para isso, usou-se a métrica PSNR e a imagem de

saída de precisão de 64 bits no *software* OpenCV como a imagem de resposta. A imagem de teste foi a Lenna (512x512) com valores em tons de cinza. A CPU testada foi Intel Core i9-9900K 3,60 GHz 8 cores/16 *threads*. Os resultados mostraram que o acelerador é cerca de 30% mais rápido que os métodos convencionais.

Chippa (CHIPPA et al., 2013) apresentou uma abordagem integrada baseada em três componentes para realizar a aproximação de *hardware*. Os componentes são: i) resiliência automática: para permitir que o projetista avalie quantitativamente o nível de resiliência intrínseca de cada aplicativo em avaliação; ii) a implementação de um *hardware* com mecanismos para lidar com o compromisso entre precisão computacional e energia; e iii) resiliência dinâmica, responsável por controlar o *feedback* para modular os mecanismos de escala em tempo de execução, para a resposta de várias aplicações. Dessa maneira, os autores propõem uma arquitetura denominada de Caracterização de Resiliência de Aplicativo (ARC). Ela quantifica/avalia a resiliência intrínseca dos aplicativos. O trabalho alcança reduções de energia que variam de 1,3 a 2 vezes. A economia de energia aumenta para 2 a 20 vezes além de haver uma perda de 5% na qualidade da aplicação em análise.

Com base nos trabalhos relacionados apresentados, pode-se observar que as arquiteturas com precisão configurável em tempo de execução são uma abordagem importante para melhorar o *trade-off* entre eficiência energética e qualidade das aplicações durante o tempo de execução.

Através da Tabela 1 pode-se observar que a maioria dos trabalhos encontrados na literatura sobre o filtro gaussiano 2D envolvendo computação aproximada enfatizam requisitos como: eficiência energética, buscando manter uma ótima qualidade na saída da aplicação e, também, redução de área do circuito. Porém, o trabalho de Pham (PHAM-QUOC; THINH, 2017) expõe um filtro gaussiano adaptável em tempo de execução, utilizando computação reconfigurável, tornando-se atraente devido à sua capacidade de acelerar e focar no desempenho dos sistemas computacionais.

Outro aspecto interessante refere-se à diversas arquiteturas configuráveis em tempo de execução para diferentes aplicações. Muitos artigos trabalham com operadores aritméticos, como: somadores, multiplicadores e divisores configuráveis em termos de aproximação com uma variedade de finalidades, além de alguns trabalhos tentarem maximizar o desempenho e eficiência energética do filtro gaussiano 2D. A Tabela 1 apresenta um resumo com as principais características dos trabalhos relacionados.

A partir da revisão literária observou-se que a exploração a nível de circuito, como técnica de computação aproximada, em aceleradores de *hardware* com configurabilidade dinâmica, não foi atacada. Portanto esta pesquisa irá estudar/implementar técnicas de computação aproximada para o projeto de um acelerador de *hardware* do filtro gaussiano 2D, a fim de explorar o *trade-off* entre qualidade e eficiência energética. A arquitetura projetada terá configurabilidade definida em tempo de projeto e em tempo de execução.

Tabela 1.: Resumo dos Trabalhos Relacionados

Trabalho	Objetivo	Configurabilidade	Redução de Energia	Tecnologia	Benchmark	Qualidade
(SOARES et al., 2019)	Somadores aproximados para processamento de vídeo e imagens	Em tempo de projeto	7,7% a 73,2%	ASIC 45 nm	BSD	30 dB a 50 dB
(OLIVEIRA et al., 2016)	Somadores aproximados para filtros energicamente eficientes	Em tempo de projeto	50%	ASIC 45 nm	7 blocos da Imagem Lena	28 dB a 36 dB
(ZHU et al., 2010)	Somador tolerante a erros	Em tempo de projeto	10%	nível elétrico	-	-
(QIEH et al., 2018)	Multiplicador aproximado (8 bits) de eficiência energética usando aproximações por SDLC	Em tempo de projeto	68,3%	ASIC 90 nm	Imagem Lena de 8 bits (200200)	30,2 dB a 70,2 dB
(QIEH et al., 2017)	Multiplicador aproximado (16 bits) de eficiência energética usando aproximações por SDLC	Em tempo de projeto	74%	ASIC 90 nm	Imagem Lena de 8 E 16 bits (500500)	30 dB a 50,2 dB
(KANG; KIM; KANG, 2016)	Filtro FIR energeticamente eficiente	Em tempo de projeto	50,7%	ASIC 65nm	Imagem Lena	14,7 dB a 46,3 dB
(JULIO et al., 2015)	Filtro Gaussiano usando circuitos aproximado	Em tempo de projeto	25% e 40%	ASIC 45nm	<i>Benchmark</i> Lena	30,6dB a 49,7 dB
(KAHNG; KANG, 2012)	Somador aproximado configurável com precisão	Em tempo de execução	12,4% a 51,6%	ASIC TSMC 65GP	Imagem Lena de 16 bits	24,5 dB
(SOARES; COSTA; BAMPI, 2019)	Filtro gaussiano 2D configurável	Em tempo de execução	29,1% a 64%	ASIC 45nm	16 imagens do BSD	20,6 dB a 25,7 dB
(KAUSHIK; KUMAR, 2015)	Filtro de suavização Gaussiana 2D	Em tempo de execução	Não avaliado	FPGA 45nm	Imagem Lena e Boat	20 dB a 29,36 dB
(BENARA; PURINI, 2016)	Correção de erros através do bit mais significativo	Em tempo de execução	50%	180 nm	Imagem Lena	40,9 dB
(PHAM-QUOC; THINH, 2017)	Núcleo de filtragem gaussiano	Em tempo de execução	-	FPGA	-	-
(RAMYA; MOORTHI, 2018)	Arquitetura multiplicadora de multi-precisão configurável, de granularidade grossa	Em tempo de execução	27% a 35%	ASIC 90nm	Imagem Lena de 16 bits	50 dB
(IMANI et al., 2019)	Divisor aproximado configurável em executar operações de ponto flutuante	Em tempo de execução	4,1 vezes	ASIC 45nm e nível elétrico	<i>Dataset Caltech</i> 101	35,2 dB a 47,08 dB
(KHANDELWAL et al., 2020)	Algoritmo da filtragem laplaciana local adaptada para um filtro gaussiano	Em tempo de execução	-	-	Imagem colorida	30 dB a 50 dB
(OTSUKA et al., 2020)	Filtro gaussiano baseado na transformada DCT	Em tempo de execução	-	-	Imagem Lena	85 dB a 300 dB
(CHIPPA et al., 2013)	Caracterização de Resiliência de Aplicativo	Em tempo de execução	2-20 vezes	ASIC 65nm	Conjunto de 12 <i>benchmarks</i> da mineração de dados	-
(YE et al., 2013)	Somador aproximado reconfigurável	Em tempo de execução	90%	-	Imagem gerada pelo somador	14,32 dB a 26,6 dB

Fonte: A autora

4 ARQUITETURAS PROPOSTAS PARA O FILTRO GAUSSIANO 2D 5X5

A arquitetura estática do Filtro Gaussiano 2D 5x5 escolhida como referência neste trabalho foi adotada anteriormente por Soares (SOARES et al., 2019), e parte do princípio básico da abordagem da implementação de multiplicação por constantes, onde a operação de multiplicação é implementada por operações de somas e deslocamentos, ao invés de um multiplicador convencional.

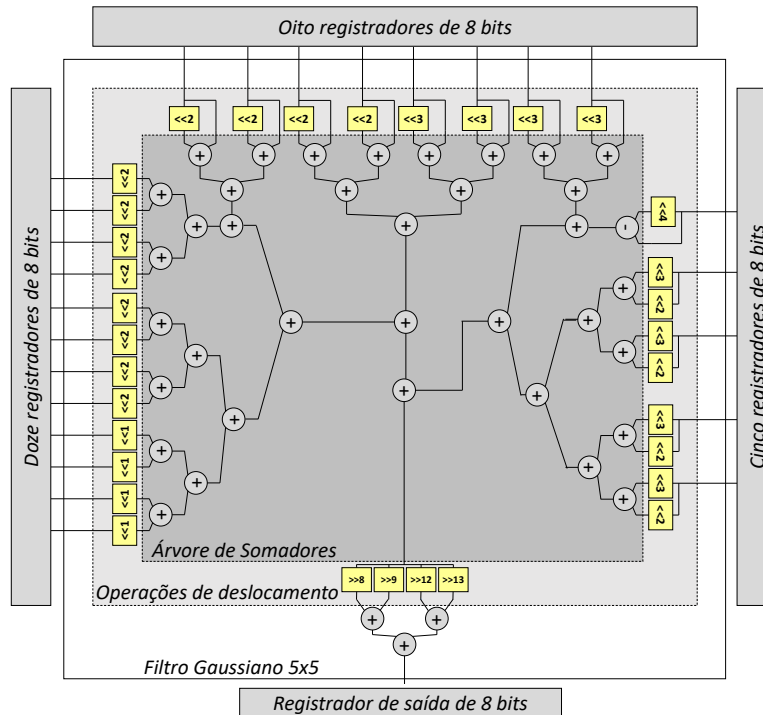
O funcionamento interno das operações ocorre da seguinte forma: dado que x é a amostra de entrada, a multiplicação apresentada na Eq. 5 pode ser executada pela Eq. 6, onde $\ll 1$ denota uma operação de deslocamento à esquerda. Dessa forma, apresenta menor esforço de *hardware*, em comparação ao se implementar topologias convencionais por um multiplicador. O deslocamento exige uma estrutura relativamente simples, com um atraso praticamente mínimo.

$$y = 3x \quad (5)$$

$$y = (x \ll 1) + x \quad (6)$$

A Figura 4 apresenta a versão da arquitetura de 8 *bits* do filtro gaussiano 2D. É possível observar que a implementação é totalmente paralela, sendo composta por operadores de somas e deslocamentos. O bloco da arquitetura contém como entrada 25 registradores de 8 *bit*, 39 somadores, 1 subtrator e 2 sinais de controle, sendo eles: o *clock* e o *reset*. O banco de registradores foi adicionado na entrada do filtro gaussiano 2D e o resultado da filtragem da arquitetura é armazenado em um registrador de saída de 8 *bits*.

Figura 4: Arquitetura do filtro gaussiano 2D 5X5 de 8 bits



Fonte: A autora

Este trabalho explora o *trade-off* entre qualidade e eficiência energética dos dispositivos digitais, a fim de investigar o impacto da computação aproximada em arquiteturas digitais. Nesta seção serão apresentadas detalhadamente as etapas e o desenvolvimento dos experimentos para a avaliação das arquiteturas do filtro gaussiano 2D em estudo. Utilizou-se uma exploração de granularidade fina, ou seja, a nível de circuitos, como técnica de computação aproximada aplicada a arquitetura referencial deste trabalho. Nessa técnica, a aproximação é explorada reduzindo-se a largura dos *bits* do filtro gaussiano 2D. A metodologia deste projeto divide-se em três partes principais:

1. Projeto de arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto;
2. Projeto do filtro gaussiano 2D com precisão configurável em tempo de execução;
3. Modelagem do *hardware* em *software*.

Inicialmente foi desenvolvida uma proposta de aproximação definida em tempo de projeto das arquiteturas do filtro gaussiano 2D. Para o desenvolvimento dessas arquiteturas utilizou-se a técnica de computação aproximada de redução da largura de *bits*. Foram desenvolvidas seis versões da do filtro gaussiano 2D, para diferentes larguras de *bits*, ou seja, 3 a 7 *bits*.

Posteriormente implementou-se uma versão do filtro gaussiano 2D com precisão configurável em tempo de execução. Nesta versão, a configuração do número de *bits* a ser usada em determinado momento, seja para a finalidade de se obter maior qualidade ou eficiência energética, é determinada em tempo de execução pelo próprio *hardware* ou usuário.

Por último, foram realizadas modelagens dessas seis versões das arquiteturas do filtro gaussiano 2D, a fim de se realizar simulações via *software*. Para observar a qualidade dos filtros aproximados, adotou-se um conjunto de imagens pertencentes ao *benchmark Berkeley Segmentation Dataset BSD* para testes na área de segmentação de imagens. As imagens originais do *benchmark BSD* utilizadas para análise deste trabalho estão apresentadas no apêndice A e também para uma percepção visual detalhada em nosso repositório: <https://github.com/TalitaBorg/Benchmarks.git>.

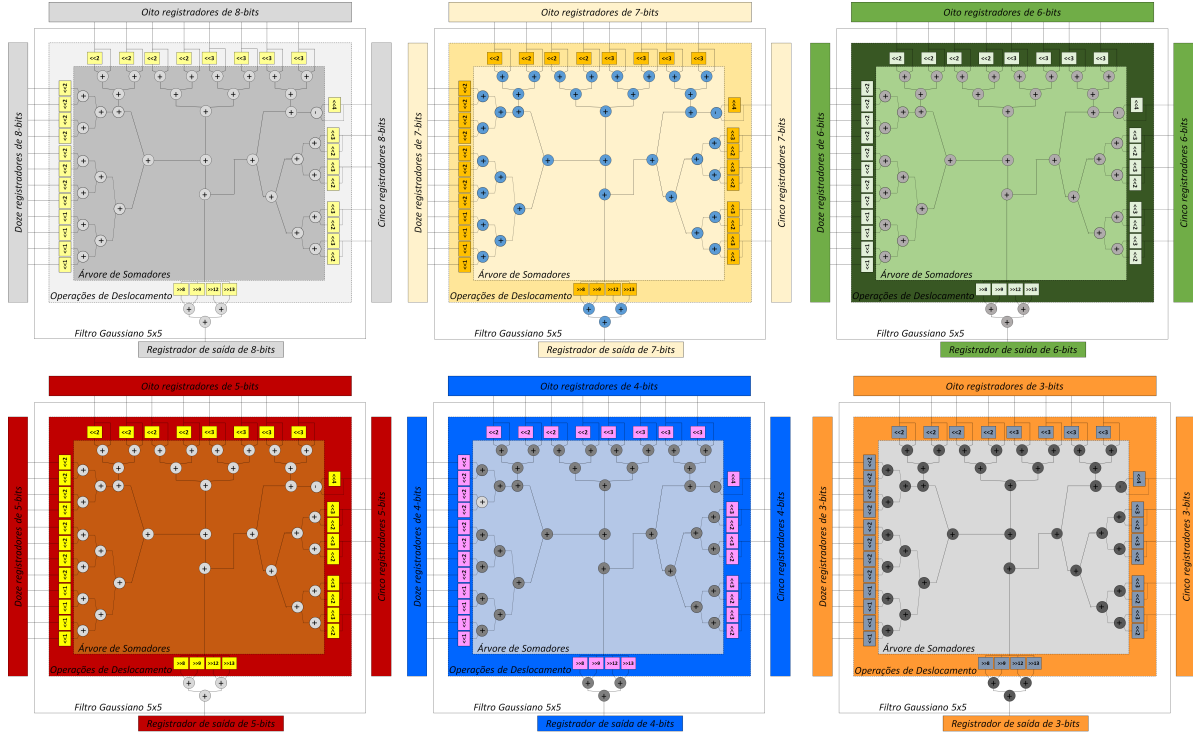
A esse conjunto de imagens, também aplicou-se ruído gaussiano para analisar a qualidade dessas imagens após passarem pelo filtro gaussiano 2D projetado. Implementou-se via *software* octave um código para realizar essa adição de ruído gaussiano, no qual todas as imagens de largura de 8 *bits* foram corrompidas por ruído gaussiano aditivo com média $\mu = 0$ e variância $\sigma^2 = 0,01$. Estes valores foram utilizados tendo como base o trabalho de Lee (LEE; TANG; PARK, 2016).

A adição de ruído gaussiano é realizada pela adoção da função embutida *imnoise* do pacote de processamento de imagem do *software* Octave. Esta função primeiro normaliza as amostras das imagens originais para inserir ainda mais ruído gaussiano, seguindo os parâmetros de média e variância dados. Depois disso, as amostras são redimensionadas no intervalo de 0 a 255 valores de 8 *bits*.

4.1 Projeto de arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto

O projeto das arquiteturas do filtro gaussiano 2D inicia com a descrição de *hardware* *Very High-Speed Integrated Circuits Hardware Description Language* (VHDL) da versão base da arquitetura que contém largura de 8 *bits*. Em seguida, foram descritas cinco versões distintas das arquiteturas do filtro gaussiano 2D, no qual cada uma das versões possui largura de *bits* diferente, ou seja, de 3 a 7 *bits*. A arquitetura possui a capacidade de filtrar 1 *pixel* por ciclo. A Figura 5 apresenta uma representação das seis versões da arquitetura do filtro gaussiano 2D com aproximação definida em tempo de projeto. Cada uma das arquiteturas recebe como entrada um conjunto de 10 imagens do *benchmark BSD*, todas em escala de cinza e também suas versões com redução na largura de *bits* (*i.e.*, de 3 a 7 *bits* de largura).

Figura 5: Versões da arquitetura do filtro gaussiano 2D 5X5 de 8 bits



Fonte: A autora

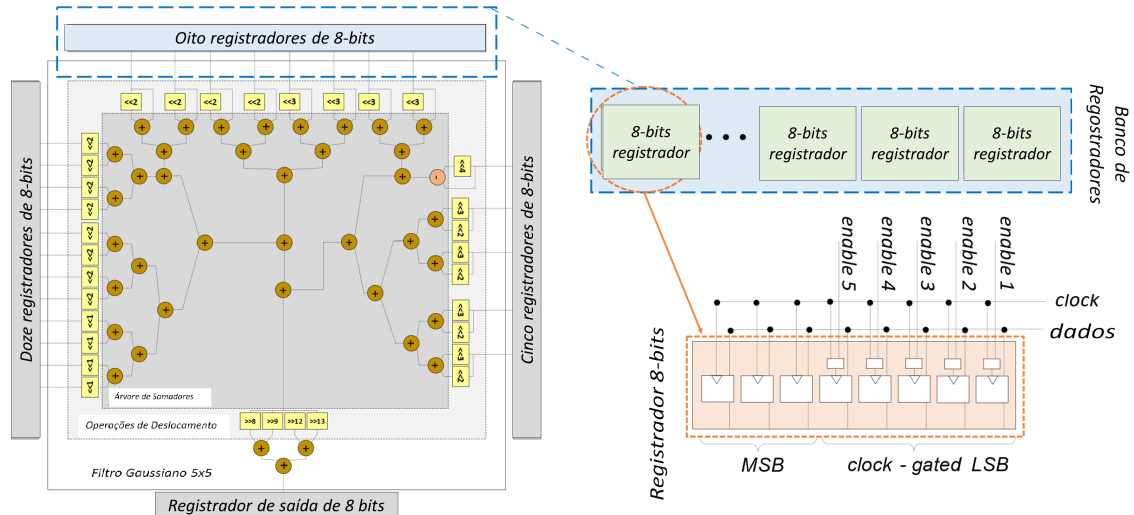
4.2 Projeto do Filtro Gaussiano 2D com precisão configurável em Tempo de Execução

A partir do desenvolvimento das arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto, foi proposta uma implementação que permitisse a configuração de qualidade e energia em tempo de execução, alterando dinamicamente o número de *bits*. O projeto desse filtro gaussiano 2D configurável em tempo de execução foi implementado na linguagem de descrição de *hardware* VHDL e prevê apenas uma implementação, no qual o gerenciamento das configurações de aproximação de 3 a 8 *bits* ocorra através do *hardware* ou do usuário, com o intuito de atender as restrições de energia ou qualidade da aplicação.

A ideia é que se possa controlar dinamicamente a largura de *bits* do filtro, a fim de permitir uma exploração do *trade-off* entre potência e qualidade da aplicação em certos pontos de operação durante o tempo de execução. A arquitetura possui um conjunto de 25 registradores de oito *bits*, em que a técnica de *clock gating* é aplicada nos cinco *bits* menos significativos (LSBs).

A Figura 6 apresenta uma ilustração da proposta configurável deste trabalho, bem como a aplicação da técnica de *clock gating* nos LSBs. Os três *bits* mais significativos (MSB) encontram-se sempre ativos, impondo assim, a limitação da largura de *bits* mínima para a arquitetura com precisão configurável em tempo de execução.

Figura 6: Arquitetura do acelerador do filtro gaussiano 2D com precisão configurável em tempo de execução



Fonte: A autora

O *Clock gating* atua na rede de distribuição do *clock* do circuito, desabilitando o sinal de *clock* nos ciclos onde ele não é necessário, reduzindo assim, a dissipação de potência em um dado momento. Portanto, muitas configurações em nível de circuitos podem ser gerenciadas durante o tempo de execução. Cada registrador do filtro é projetado com largura de 8 *bits*.

Essa arquitetura pode ser adaptada dinamicamente por meio dos sinais de *enable*. Dessa forma, em nível de sistema, é possível controlar o número de *bits* que estão em estado ativo permitindo uma configuração em tempo de execução, com base nos alvos de potência dinâmica e precisão. Ressalta-se que a lógica de controle, ou seja, a máquina de estados que ativa os sinais: *enable*, *reset* e também a técnica de *clock gating* foi simulada em *testbenchs*, ficando como etapa futura no desenvolvimento desta pesquisa.

Em condições nominais, essa arquitetura trabalha com os dados de entrada integrais de 8 *bits* oferecendo a melhor qualidade de imagem possível. Se exigido pelo sistema, a arquitetura pode ser configurada para atender às restrições de energia ou qualidade, reduzindo ou aumentando-se o número de *bits* adotados durante o período de processamento do filtro gaussiano.

4.3 Modelagem do *hardware* em *software*

Esta seção descreve o processo metodológico da modelagem das arquiteturas propostas de filtro gaussiano 2D. A modelagem foi desenvolvida na linguagem de programação

python. Para isso, subdivide-se em:

1. Modelagem do comportamento das arquiteturas atuando na redução da largura de *bits* para análise da qualidade;
2. Modelagem para se observar o impacto da remoção do sinal de controle de *reset* durante a configuração da arquitetura.
3. Modelagem para analisar a degradação de imagens através da concatenação em diferentes larguras de *bits*;

Todas essas avaliações de qualidade simuladas via *software* mensuram o impacto visual das diferentes larguras de *bits* sob análise. Os códigos utilizados para tais experimentos encontram-se disponíveis no apêndice C.

4.3.1 Modelagem do comportamento das arquiteturas atuando na redução da largura de *bits* para análise da qualidade

A análise de qualidade dos filtros gaussianos 2D com aproximação via quantização dos *bits* menos significativos tem como objetivo fornecer uma visão qualitativa do impacto em imagens do comportamento decorrente da quantização. Neste processo, zeram-se os *bits* menos significativos durante a filtragem gaussiana. A Figura 7 apresenta as etapas adotadas para a elaboração deste ambiente de simulação em *software* das arquiteturas.

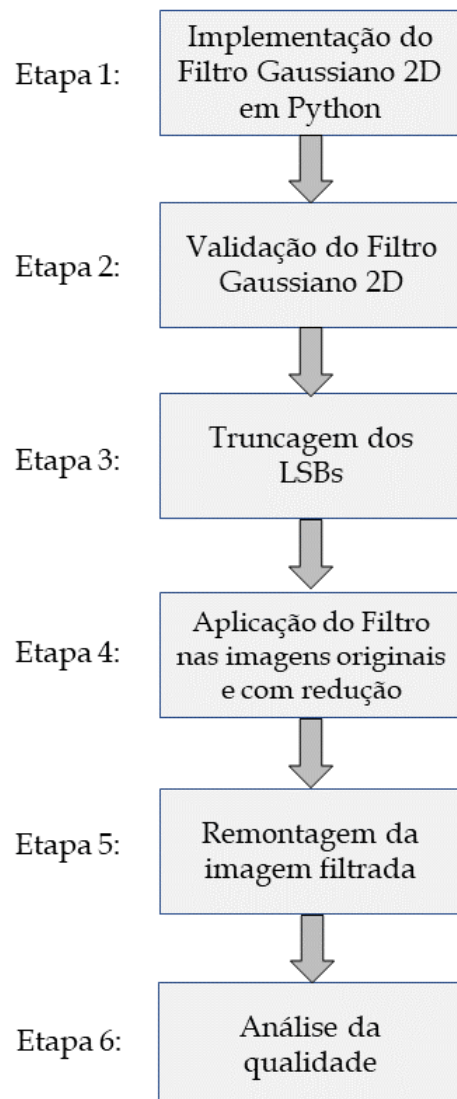
Etapas 1: Implementação do filtro gaussiano 2D estático

Inicialmente implementou-se a versão de 8 *bits* do filtro gaussiano 2D como proposto em Soares (SOARES et al., 2019). Logo após, foram descritas cinco versões da arquitetura original aplicando a redução de *bits* respectiva a quantização de 3 a 7 *bits* do filtro gaussiano 2D. Estas modelagens em *software* aceleram a validação do funcionamento real das arquiteturas propostas, e permitem estudos mais detalhados.

Etapas 2: Validação do filtro gaussiano 2D

Para a etapa de validação da arquitetura foi elaborado um código capaz de gerar vetores de entrada para alimentar o filtro. Este código transforma cada uma das 10 imagens do *benchmark* BSD em vetores de 25 amostras, listando as imagens e transformando-as para o formato .out. O código que realiza a validação do filtro gaussiano 2D encontra-se no Apêndice B.

Figura 7: Fluxo de etapas para avaliação de qualidade.

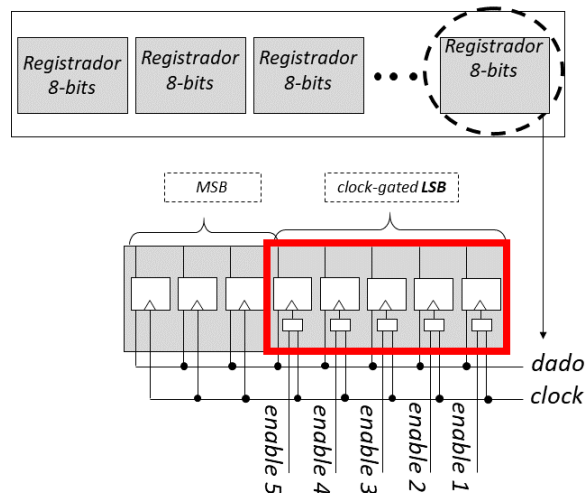


Fonte: A autora

Etapa 3: Truncagem dos LSBs

Após ter-se as versões da arquitetura do filtro gaussiano 2D implementadas e validadas, ocorreu a etapa 3, que vislumbra aplicar uma estratégia nos *bits* menos significativos dos dados de entrada do filtro, para zerá-los, a fim de se observar os níveis de qualidade gerados ao final desse procedimento. A Figura 8 apresenta um esquemático ilustrativo de como essa estratégia funciona.

Figura 8: Truncagem dos LSBs



Fonte: A autora

Etapa 4: Processamento de imagens com os Filtros Gaussianos

O filtro gaussiano possui coeficientes e tamanho de máscara idênticos ao utilizado na implementação de *hardware*. A aplicação do filtro gaussiano 2D ocorreu para cada uma das 10 imagens do *benchmark* BSD, primeiramente sem ruído e seguidamente com adição de ruído gaussiano. Nessa etapa, a saída da arquitetura gera uma imagem no formato de um arquivo .txt. O código que realiza a transformação da imagem filtrada para formato PNG, JPG ou BMP encontra-se no apêndice C.

Etapa 5: Remontagem da imagem filtrada

Para a etapa de remontagem da imagem para o formato BMP, PNG ou JPG utilizou-se a função *reshape* do python. Essa etapa é de suma importância para depois poder-se aplicar a função PSNR tendo como referência a arquitetura base, de 8 *bits* do filtro gaussiano 2D. O resultado do processo descrito pode ser analisado através de uma percepção visual a partir da Figura 9, na qual é possível observar a aplicação do filtro gaussiano 2D em ambas imagens, primeiramente a) sem ruído e b) com adição de ruído gaussiano.

Figura 9: Aplicação do filtro gaussiano em a) sem ruído e b) adição de ruído gaussiano.



Fonte: A autora

Etapa 6: Análise dos resultados de qualidade

Por fim, partiu-se para a etapa 6, referente à análise de qualidade através da métrica PSNR (HORE; ZIOU, 2010). A métrica PSNR é considerada um método objetivo baseado em comparações através da utilização de critérios numéricos. Dessa forma, obtém-se várias referências através de parâmetros estatísticos e testes realizados. Foi utilizada uma implementação via python para se obter o valor de PSNR entre a imagem base com 8 *bits* e suas variações de 3 a 7 *bits*.

4.3.2 Modelagem para avaliação da inativação do sinal *reset* entre mudanças de configurações

Buscou-se através dessa modelagem, aplicar uma técnica de compensação de erro intrínseca, que reduz o a magnitude do erro e o número de ciclos para permitir uma nova configuração da arquitetura. Para observar a degradação do conjunto de imagens do *benchmark* BSD, de uma maneira dinâmica fez-se uma avaliação do impacto de se utilizar ou não, o sinal *reset* nos registradores de entrada da arquitetura do filtro gaussiano 2D configurável em tempo de execução.

A hipótese baseia-se em ignorar o sinal *reset* antes de uma nova configuração. Este

procedimento mantém os valores nas posições dos *bits* que serão desativados, como uma solução para reduzir a resposta de erro com polarização positiva, gerado pelo truncamento e, para evitar ciclos extras relacionados ao sinal de *reset*. Dessa forma, geralmente precisa-se de apenas um ciclo de *clock* por nova configuração. Além disso, o sinal de *reset* não é utilizado quando não se tem aproximações.

Para simular o que aconteceria se o sinal *reset* fosse aplicado, forçou-se com máscaras específicas os *bits* menos significativos, antes de uma nova configuração do filtro gaussiano 2D ser selecionada. Para uma versão em *software* mais realista do funcionamento do filtro gaussiano 2D, com redução da largura de *bits*, avaliou-se *software* duas possibilidades:

1. Sinal *reset* ativo nos LSB com redução na quantidade de *bits*;
2. A manutenção dos valores pré-carregados de uma configuração anterior dos LSB nos registradores.

Para reproduzir o comportamento na arquitetura com o *reset* ativo nos LSBs a serem descartados, foi implementado um pré-processamento, para simular o *reset* ligado/desligado, juntamente com as funcionalidades da técnica de *clock gating* para todos os possíveis perfis aproximados da arquitetura. Para simular o cenário de reinicialização, adotamos operações com o uso de máscaras *bit a bit* com o objetivo de limpar as posições de *bits* onde o *clock gating* estivesse habilitado.

A técnica de *clock gating* possui a função de desabilitar esses *flip-flops* menos significativos. Pode-se considerar que o uso do sinal *reset* associado à técnica de *clock gating* realiza um truncamento nas operações aritméticas. Portanto, os resultados aproximados tendem a ser menores do que os precisos, uma vez que o truncamento produz uma característica de erro com tendência positiva. Ou seja, considerando o erro, há diferença entre os resultados precisos e aproximados.

Este trabalho realiza $2k$ simulações, onde k é o número de LSBs a serem desabilitados. Para cada simulação, forçou-se estaticamente os dados de entrada para valores que variam de 1 a $2^5 - 1$. Portanto, após a máscara AND limpar os k LSBs, uma máscara OR adicional é realizada para registrar o conteúdo estático na região desabilitada, simulando dados preexistentes nos registros antes que a técnica de clock-gating desabilite esses flip-flops.

A Tabela 2 apresenta cada valor de máscara correspondente à quantidade de *bits*, pois, para se desligar 1 *bit* da arquitetura, utiliza-se a máscara 11111110, desligar 2 *bits*, 11111100, e assim sucessivamente, conforme mostrado na Tabela 2.

Tabela 2: Máscaras utilizadas

Redução da largura de <i>bits</i>	Máscara
1	11111110
2	11111100
3	11111000
4	11110000
5	11100000

Fonte: A autora

A simulação foi realizada adotando a operação lógica AND para zerar as posições de *bits*, onde a técnica de *clock gating* estava ativada. O sinal *reset* ligado denota zerar os *bits* dos *flip-flops* menos significativos, pois eles não terão possibilidade de receber novos valores de entrada. Por outro lado, o *reset* desligado indica que as posições desses LSBs não serão apagadas antes que a nova configuração seja selecionada. Logo, nessa condição de *reset* desligado, esses *flip-flops* mantêm um valor pré-carregado da operação executada anteriormente.

Para avaliar os efeitos de desabilitar o *reset*, mantendo os valores anteriores, optou-se por implementar e testar uma avaliação combinatória que considerasse todos os valores possíveis para a aproximação de 5 *bits*, pois esta versão do filtro gaussiano 2D apresentará mais impacto de qualidade. Sendo assim, foram testadas as 32 combinações possíveis de valores para estes *bits*.

Usou-se a máscara 11100000 de forma a zerar os 5 *bits* menos significativos correspondentes à redução de 5 *bits* da arquitetura do filtro. Utilizou-se essa máscara juntamente com o operador lógico OU para forçar os *bits* menos significativos com valores 0 a 31. Após as manipulações para representar o *reset* em estado ativo ou desativado, foi calculado e anotado o valor de PSNR correspondente a cada arquitetura aproximada.

4.3.3 Modelagem para analisar a degradação de imagens através da concatenação em diferentes larguras de *bits*

Esta avaliação busca realizar uma análise objetiva, além de ser possível fazer uma percepção visual, do nível de degradação de um conjunto de imagens do *benchmark* BSD, com diferentes larguras de *bits* (3 a 8 *bits*), filtradas pelo filtro gaussiano 2D. A metodologia busca cortar uma dada imagem de largura de *bits* x em várias faixas de dimensão k , no qual essa dimensão possui x *pixels*.

Para elaboração deste experimento, primeiramente foi elaborado um código em python utilizando a biblioteca de imagens PILLOW (CLARK, A., 2021). Essa biblioteca oferece suporte a vários formatos de arquivos. Esse primeiro código tem a finalidade de subdividir a imagem em faixas horizontais de acordo com a dimensão dos cortes.

Após a imagem ser subdividida, ela é salva em uma pasta local com o nome correspondente ao processo que ela passou, por exemplo: Imagem2Filtrada3bitsSemRuido,

que significa que essa imagem pertence ao *benchmark* BSD, nomeada como Imagem 2, processada pelo filtro com redução de 3 *bits* e que não contém adição de ruído gaussiano. Depois disso, a imagem vai sendo cortada em faixas horizontais e estas faixas vão sendo nomeadas, como: Faixa1, Faixa2 e assim, sucessivamente até a última fatia que mostra o fim da imagem. Cada faixa contém o tamanho de 50 *pixels* de altura, mantendo o valor original de largura da imagem, sendo necessários esses valores para que depois seja possível mesclar as faixas.

Posteriormente a esse procedimento, fez-se outro código em python, ainda com o auxílio da biblioteca PILLOW, onde tendo-se várias faixas de largura de *bits* diferentes, concatena-se essas faixas, a fim de remontar a imagem para análise de qualidade. Essa imagem pode ser remodelada, aplicando-se maior redução na quantidade de *bits* onde se deseja obter melhor qualidade na região de interesse. A Figura 10 apresenta em a) imagem original do pássaro com 8 *bits*, em b) uma ilustração das faixas e em c) imagem resultante após o processo de concatenação das faixas.

Figura 10: a) Imagem Original do pássaro de 8 bits, b) Imagem ilustrativa das faixas e c) Resultado da concatenação



Fonte: A autora

Essa seção apresentou o desenvolvimento metodológico desta pesquisa. Foram apresentadas as arquiteturas do filtro gaussiano 2D desenvolvidas, no qual a proposta do filtro gaussiano 2D com precisão configurável em tempo de execução difere-se das versões com aproximação em tempo de projeto. Tal diferença se dá por ser possível controlar, de forma dinâmica, através do gerenciamento das configurações de aproximação, a largura de *bits* do filtro, a fim de oferecer vários perfis de qualidade, potência e desempenho. Através da modelagem da arquitetura do filtro em *software* será possível avaliar a qualidade, a partir de testes de diferentes características, de forma mais eficiente.

5 RESULTADOS

Neste capítulo serão apresentados os resultados referentes às estimativas de eficiência energética e dimensionamento da área das arquiteturas do filtro gaussiano 2D. A etapa de síntese foi realizada utilizando o *Register Transfer Level* (RTL) *Compiler* no filtro gaussiano 2D, utilizando a linguagem de descrição de *hardware* VHDL sintetizada para a tecnologia ST de 65nm, e considerando o fluxo de implementação ASIC. As frequências de *clock* são determinadas para filtrar quadros considerando um vídeo em tempo real. A síntese foi feita considerando os dois *clocks*. Elas estão listadas da seguinte forma:

1. 63 MHz para processar vídeos em alta definição (full-HD) a 30 fps;
2. 249 MHz para processar vídeos Ultra HD 4K a 30 fps.

Para determinar a frequência de operação deve-se considerar que:

1. i) a arquitetura proposta filtra um *pixel* em tons de cinza por ciclo;
2. ii) a resolução do vídeo, que neste caso são as resoluções Full HD (1920 x 1080) e Ultra HD (3840 x 2160);
3. iii) a taxa de quadros, que neste caso é de 30 quadros por segundo.

Portanto, as frequências para atender a essas restrições são 62,2 MHz e 248,8 MHz para ambas as resoluções de vídeo. Utilizou-se a função teto para selecionar 63 MHz e 249 MHz. Pode-se observar que essas frequências são mais altas do que o necessário para atender ao processamento em tempo real. Por exemplo, ao operar a 249 MHz, a arquitetura proposta processa 249 milhões de *pixels* por segundo, o que é suficiente para filtrar 3840 x 2160 x 30 *pixels* por segundo.

A técnica de computação aproximada a nível de circuitos poderá ser analisada através desses resultados, tanto para o projeto da arquitetura do filtro gaussiano 2D com aproximação definida em tempo de projeto como para a arquitetura com precisão configurável em tempo de execução. Serão mostrados os resultados de qualidade da aplicação para os diversos perfis do filtro gaussiano 2D. Adicionalmente, serão ilustradas as análises

de uma avaliação da remoção do sinal de controle de *reset* durante a configuração da arquitetura. As seções estão organizados da seguinte maneira:

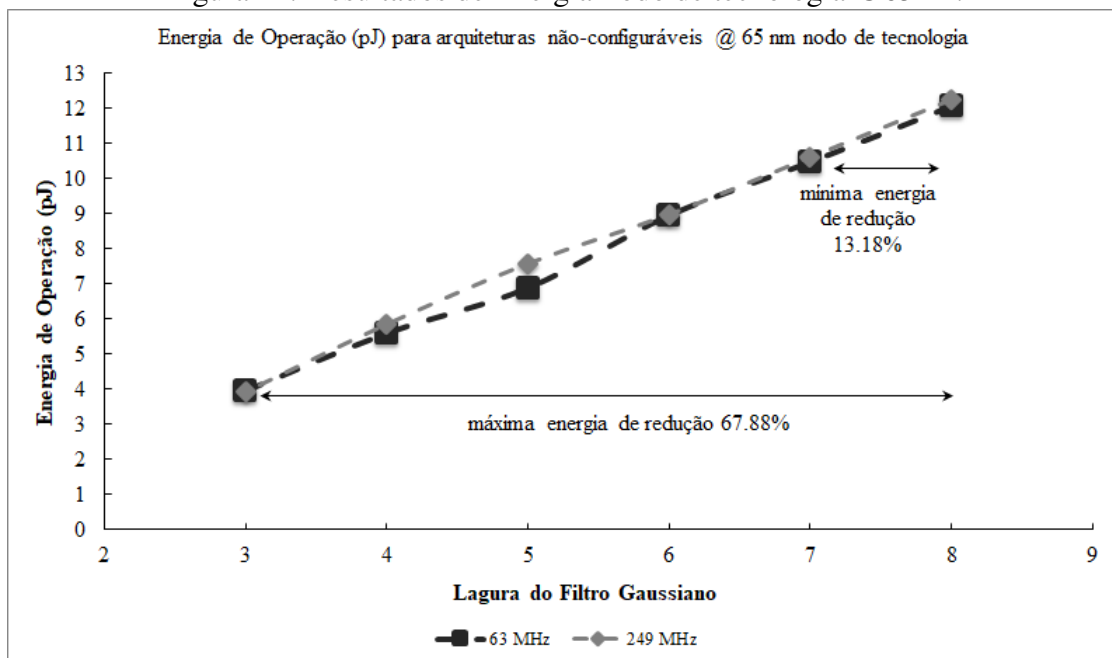
1. Resultados de síntese das arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto;
2. Resultados de síntese das arquiteturas do filtro gaussiano 2D com precisão configurável em tempo de execução;
3. Resultados da análise de qualidade da modelagem do *hardware* em *software*.

5.1 Resultados de síntese das arquiteturas do filtro gaussiano 2D com aproximação definida em tempo de projeto

O principal objetivo desta análise refere-se aos resultados da análise de potência dinâmica e área do circuito projetado. O gráfico da Figura 11 ilustra os resultados de eficiência energética para a tecnologia de 65 nm. O filtro aproximado apresenta reduções de 1 a 5 *bits*. Isso possibilitou redução de energia, variando de 13,18% a 67,88%, respectivamente.

Os resultados mostram que existe uma tendência de redução linear, pois a energia diminui proporcionalmente com a redução da largura dos *bits* do filtro. Isso ocorre devido ao circuito possuir contagem lógica reduzida e, portanto, menor dissipação de potência.

Figura 11: Resultados de Energia-nodo de tecnologia @65nm.



Fonte: A autora

A Tabela 3 apresenta os resultados da área para a tecnologia de 65 nm. São apresentados o número de células e a área (μm^2), onde as reduções variam de 11,7% a 59%. Quando observados os dois alvos de frequência de *clock*, as frequências mais altas resultam em projetos de área mais alta. Isso é esperado devido ao tamanho maior para alcançar um desempenho superior.

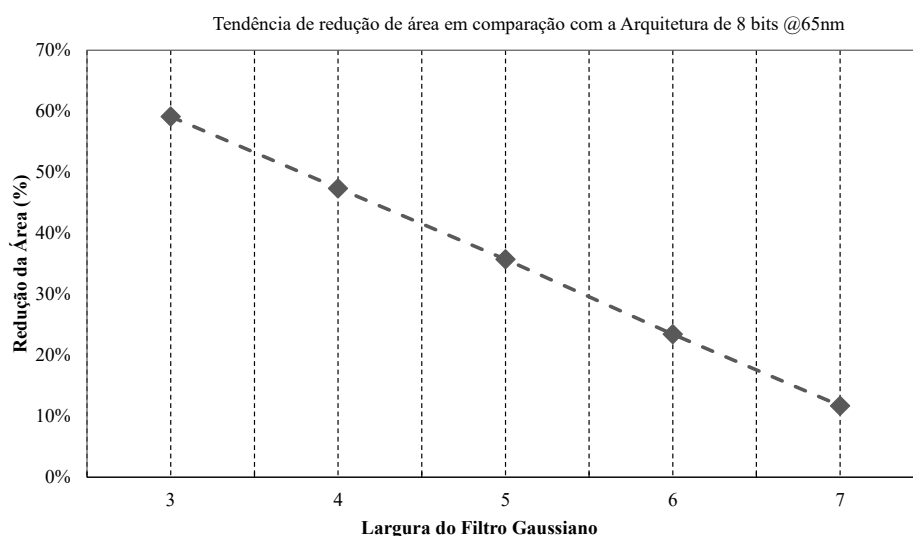
Tabela 3: Análise da área - tecnologia de @ 65 nm

Redução da largura de <i>bits</i>	63 MHz		249 MHz	
	#células	área (μm^2)	#células	área (μm^2)
0 bits	607	5156	621	5186
1 bits	543	4551	551	4574
1 bits	479	3946	494	3970
3 bits	400	3313	430	3385
4 bits	338	2713	357	2737
5 bits	272	2107	272	2111

Fonte: A autora

A Figura 12 destaca uma tendência observada para as duas frequências, ou seja, de 63 MHz e 249 MHz, em termos de redução de área. Observou-se uma redução de 59% alcançada quando as arquiteturas foram reduzidas para 3 *bits* e, para a versão de 7 *bits*, a redução é de cerca de 11%. Pode-se analisar que há uma resposta proporcional quando a largura de *bits* é alterada. Esse é um resultado substancial quando os projetistas da tecnologia CMOS decidem implementar arquiteturas não configuráveis e aproximadas.

Figura 12: Tendência de redução da área



Fonte: A autora

5.2 Resultados de síntese das arquiteturas do filtro gaussiano 2D com precisão configurável em tempo de execução

Esta seção irá apresentar os resultados da análise de potência dinâmica e área no *hardware* do filtro gaussiano 2D Configurável em tempo de execução. A síntese da arquitetura do filtro gaussiano 2D com precisão configurável em tempo de execução ocorreu para a tecnologia ST de 65nm.

As frequências exploradas e o conjunto de imagens do *benchmark* BSD foram os mesmos utilizados nas versões da arquitetura com aproximações definidas em tempo de projeto. A única diferença é que todas as imagens utilizadas, contém agora, a largura fixa de 8 *bits*. O processo de filtragem gaussiana é realizado considerando diferentes configurações desta arquitetura configurável em tempo de execução.

A simulação das arquiteturas foi realizada com base em 10.000 blocos 5x5 de imagens do *benchmark* BSD correspondente à janela do filtro implementado. Essa etapa foi realizada após a simulação do circuito gerado pela síntese lógica para capturar a atividade de chaveamento. A partir da atividade de chaveamento, faz-se através da ferramenta de síntese, uma estimativa de potência do circuito.

A Tabela 4 apresenta os resultados de potência estática e potência dinâmica para a tecnologia de 65nm. A potência dinâmica é substancialmente dominante sobre a potência estática, o que justifica a adoção da técnica de *clock gating* para reduzir a atividade de comutação do circuito.

Com base nisso, a redução da potência dinâmica é a abordagem principal para reduzir o consumo de energia por operação quando a qualidade da aplicação for baixa. Quanto aos resultados de potência estática, eles permaneceram constantes independentemente da quantização de *bits*. Por outro lado, a potência dinâmica variou de acordo com o número de *bits* acionado.

Tabela 4: Análise da potência estática e dinâmica na tecnologia de @ 65nm

Redução da largura de <i>bits</i>	63 MHz		249 MHz	
	Pot. Estática (μW)	Pot. Dinâmica (μW)	Pot. Estática (μW)	Pot. Dinâmica (μW)
0 bits	0,52	988,17	0,55	3984,89
1 bits	0,52	890,57	0,55	3559,39
2 bits	0,52	763,14	0,55	3056,25
3 bits	0,52	634,92	0,55	2524,74
4 bits	0,52	525,15	0,55	2081,49
5 bits	0,52	413,23	0,55	1634,99

Fonte: A autora

Ambas as frequências alcançam os mesmos níveis de redução de energia, no entanto, com diferenças na potência dinâmica absoluta. Contudo, a síntese com frequência de 249

MHz exige portas lógicas de tamanhos diferentes para otimização de tempo, refletindo, assim, em um aumento de 4x do consumo de potência dinâmica em comparação com a síntese da frequência de 63 MHz.

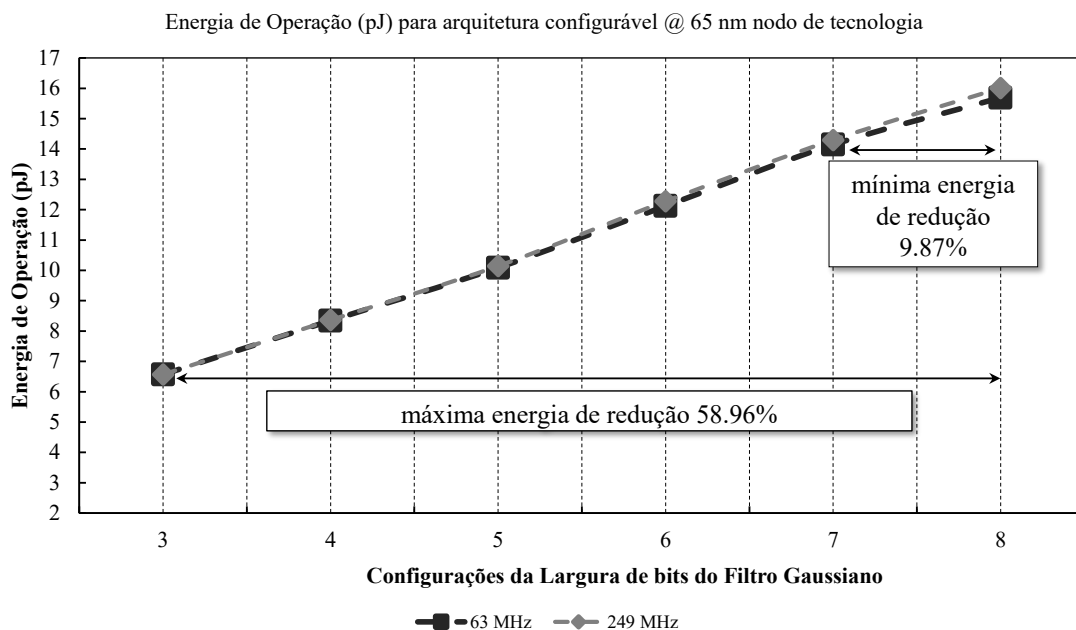
A arquitetura proposta permite melhor *trade-off* entre eficiência energética e qualidade da aplicação durante o tempo de execução. A Figura 13 mostra os resultados de energia por operação da arquitetura do filtro. A ferramenta *RTL Compiler* fornece os valores de potência dinâmica e área; logo para calcular a energia por operação faz-se necessário utilizar as frequências de operação, multiplicadas pelo período, através da fórmula :

$$Energia = (pot * T) \quad (7)$$

onde *pot* é a potência dinâmica oferecida pela ferramenta e *T* é período, ou seja, (1/frequência de operação). A energia por operação processa uma amostra por ciclo.

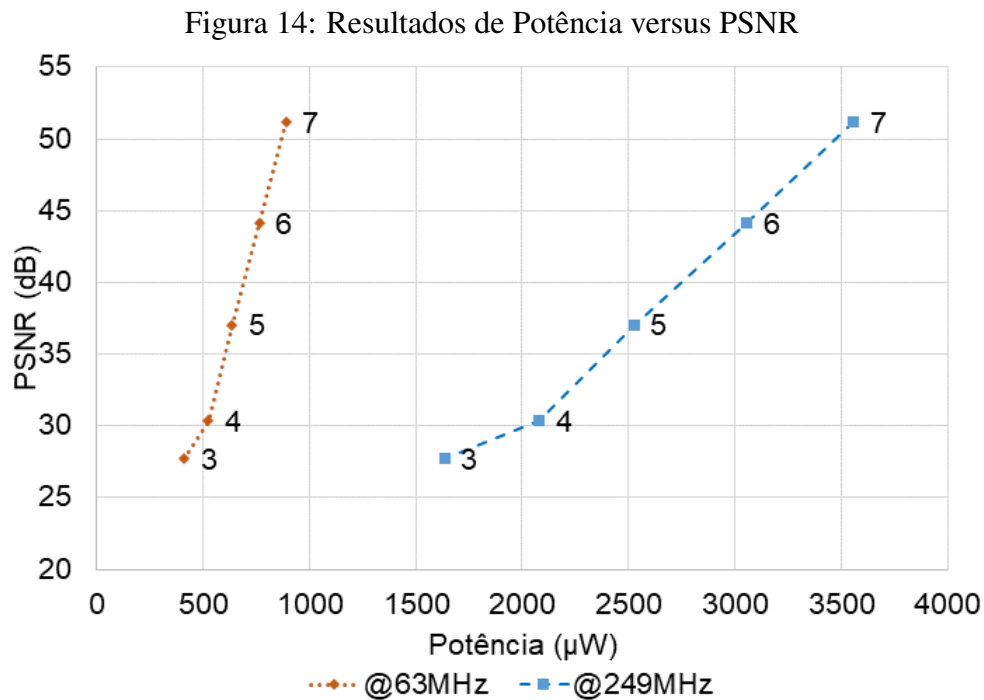
Em comparação com a solução de precisão exata de 8 *bits*, o acelerador do filtro gaussiano 2D com precisão configurável em tempo de execução apresenta reduções de energia que variam de 9,87% até 58,96% para os perfis de 7 a 3 *bits*, respectivamente. A energia diminui proporcionalmente com a redução da largura de *bits* do filtro, porque o circuito possui uma comutação reduzida de atividade devido a técnica de *clock gating* adicionada ao banco de registradores do filtro gaussiano 2D.

Figura 13: Resultados de energia da tecnologia de @65 nm da arquitetura com precisão configurável em tempo de execução



Fonte: A autora

A Figura 14 mostra a relação de qualidade entre o PSNR médio e o consumo de potência para ambos os alvos de frequências. Os resultados mostram uma tendência linear em que os valores de potência diminuem proporcionalmente, reduzindo a largura de *bits* do filtro. A principal razão para este comportamento é a adição da técnica de *clock gating* nos registradores de entrada. Ao se analisar a qualidade das imagens do *benchmark* em avaliação, o comportamento é diretamente proporcional: quanto maior a largura de *bits*, maior será a qualidade apresentada pela métrica PSNR.



Fonte: A autora

Com a adição da técnica de *clock gating*, na configuração exata da arquitetura, ou seja, em 8 *bits*, existe um *overhead* relacionado ao aumento do consumo de energia. Este consumo é devido aos circuitos adicionais que controlam as possíveis configurações de precisão. Por outro lado, através da técnica de *clock gating*, ao comparar todos os perfis analisados da arquitetura configurável em tempo de execução, existe uma redução substancial de energia toda vez que for selecionado um perfil que priorize a redução de consumo energético.

Observou-se que a arquitetura configurável em tempo de execução mostra áreas de 7.353,38 e 7.464,91 NAND2 *Equivalent Gates* (EGs) para as frequências de 63 MHz e 249 MHz, respectivamente. Para calcular o EGs, dividiu-se a área resultante pela área da NAND2 da tecnologia de 65nm a fim de se obter um parâmetro independente da tecnologia. As células adicionais de controle de *clock* resultam em um *overhead* de área variando de 13,8% a 14,8%, em comparação com o filtro gaussiano 2D tradicional para a tecnologia de 65 nm.

Quando observada a frequência de *clock*, as frequências mais altas resultam em pro-

jetos de área superior. Isso é esperado, pois alvos de frequência maiores demandam por células lógicas com maior dimensionamento. A Figura 12 mostra esta tendência linear de reduções de área apenas para a tecnologia de 65 nm. A Tabela 5 apresenta os resultados da análise de área do filtro gaussiano 2D com precisão configurável em tempo de execução.

Tabela 5: Comparação entre as área da Arq. em tempo de projeto x Arq. em tempo de execução - tecnologia de @65nm

	63 MHz		249 MHz	
	#células	área (μm^2)	#células	área (μm^2)
Arq. Tempo de Projeto	607	5156	621	5186
Arq. Tempo de Execução	925	5828	937	5957

Fonte: A autora

Ao comparar os resultados entre as arquiteturas com configurabilidade em tempo de projeto com aqueles apresentados pela arquitetura do acelerador do filtro gaussiano 2D configurável em tempo de execução, observa-se que os resultados de redução de energia ficaram próximos, o que já era esperado em função da similaridade da técnica de aproximação.

Além disso, foi observado na solução configurável em tempo de execução a ocorrência de um *overhead* de área devido à adição de *hardware* extra ao circuito, contrapondo as vantagens da arquitetura configurável de gerenciamento de diferentes perfis de qualidade e energia de acordo com definições do sistema. O próprio contexto de qualidade pode ser melhor explorado de forma a permitir diferentes perfis de aproximação para uma mesma imagem, habilitando uma configurabilidade orientada às características da imagem.

5.3 Resultados da análise de qualidade da modelagem do *hardware* em *software*

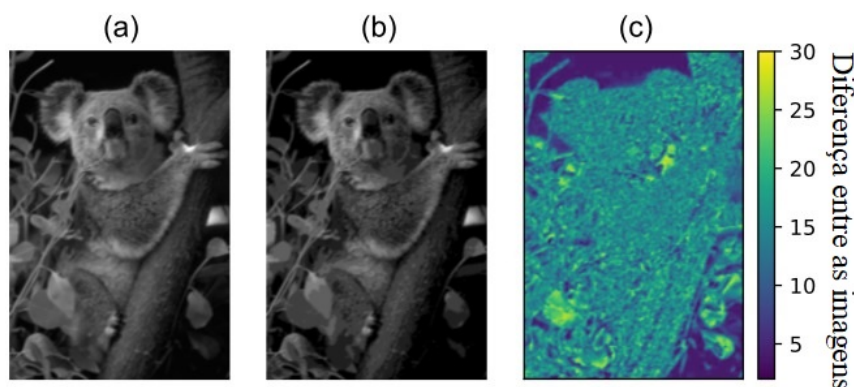
A avaliação da qualidade neste trabalho é composta por duas análises: objetiva e por percepção visual. A análise objetiva é realizada por meio da métrica PSNR, enquanto a percepção visual é feita através da experiência da pessoa que está analisando as imagens. Os resultados da análise de qualidade dinâmica buscaram avaliar diferentes níveis de qualidade nas imagens do *benchmark*, através do filtro gaussiano 2D. Esta seção está organizada da seguinte forma:

1. Resultados da análise das imagens inteiras sem ruído;
2. Resultados da análise das imagens inteiras com adição de ruído gaussiano;
3. Resultados da modelagem de avaliação da inativação do sinal de *reset*;
4. Resultados da modelagem para analisar a degradação de imagens através da concatenação em diferentes larguras de *bits*.

5.3.1 Resultados da análise de imagens inteiras sem ruído

Como um exemplo, ao analisar a Figura 15, uma percepção visual pode ser realizada da seguinte forma: a) imagem do filtro gaussiano preciso, b) uma imagem filtrada pelo perfil aproximado de 3 *bits* e em c) mapa de calor com as diferenças entre os filtros gaussianos 2D preciso e de 3 *bits*. Esse exemplo permite uma avaliação qualitativa dos resultados aproximados, apresentando o comportamento geral encontrado para as outras imagens do *benchmark* BSD em avaliação. Através do mapa de calor pode-ser observar as diferenças entre amostras da imagem precisa em relação à imagem com redução de 3 *bits*.

Figura 15: Análise objetiva para a imagem de entrada sem ruído: (a) Filtro gaussiano 2D preciso (b) Perfil aproximado de 3 *bits* de redução (c) Mapa de calor com as diferenças entre os filtros gaussianos 2D preciso e de 3 *bits*



Fonte: A autora

A Tabela 6 apresenta o resultado da média dos valores de PSNR das 10 imagens do *benchmark* BSD, primeiramente, sem a adição de ruído gaussiano. Nota-se que os valores variaram de 51,17 dB na configuração de 1 *bit* de redução a 27,73 dB para a redução de 5 *bits*.

Tabela 6: Média de valores do PSNR x N de *bits* - Imagens sem ruído

Redução da largura de <i>bits</i> (<i>bits</i>)	PSNR (dB)
1	51,17
2	44,11
3	37,05
4	30,36
5	27,73

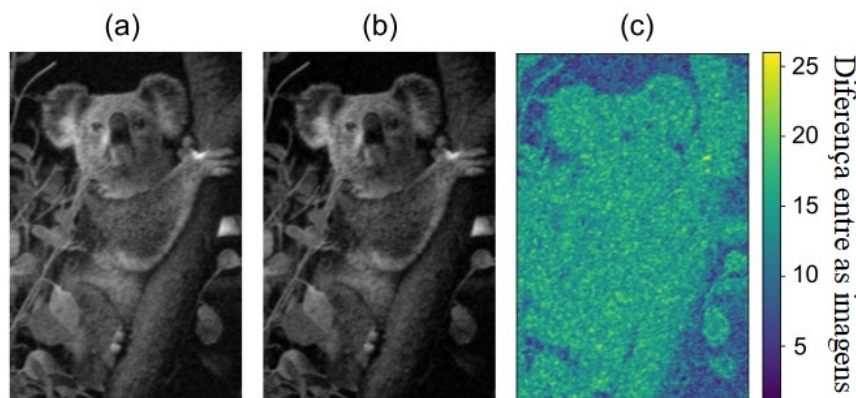
Fonte: A autora

5.3.2 Resultados da análise de imagens inteiras com adição de ruído gaussiano

A Figura 16 considera adição de ruído gaussiano com média $\mu = 0$ e variância $\sigma^2 = 0,01$. Observa-se que a maioria dos erros de magnitude são menores que 25, o que ratifica a dificuldade para se identificar através da percepção visual humana as diferenças entre as imagens.

A Figura 16 reflete o comportamento geral encontrado para outras imagens do *benchmark*, não havendo degradação e diferenças visuais entre as imagens filtradas. As análises qualitativas visuais indicam que a solução aproximada pode filtrar o ruído sem degradação indesejável. Além disso, indica que o cenário mais agressivo de computação aproximada pode ser adequado para aplicações de filtro gaussiano.

Figura 16: Análise objetiva para a imagem de entrada corrompida com ruído gaussiano com $\mu = 0$ e $\sigma^2 = 0,01$: (a) Filtro gaussiano 2D preciso (b) Perfil aproximado de 3 *bits* de redução (c) Mapa de calor com as diferenças entre os filtros gaussianos 2D preciso e de 3 *bits*



Fonte: A autora

A Tabela 7 mostra o resultado da média dos valores de PSNR as 10 imagens do *benchmark* BSD, com adição de ruído gaussiano. Observa-se que os valores variaram de 51,22 dB na configuração de 1 *bit* de redução à 27,5 dB para a redução de 5 *bits*. Quando comparados esses valores de PSNR com adição de ruído gaussiano com os valores da Tabela 6, nota-se que os valores são muito próximos e isso indica que o funcionamento do filtro gaussiano 2D projetado é capaz de remover os artefatos indesejados, conseguindo manter a mesma qualidade das imagens.

Tabela 7: Média de valores do PSNR x N de *bits* - Imagens com adição de ruído

Redução da largura de <i>bits</i> (<i>bits</i>)	PSNR (dB)
1	51,22
2	44,26
3	37,24
4	30,70
5	27,50

Fonte: A autora

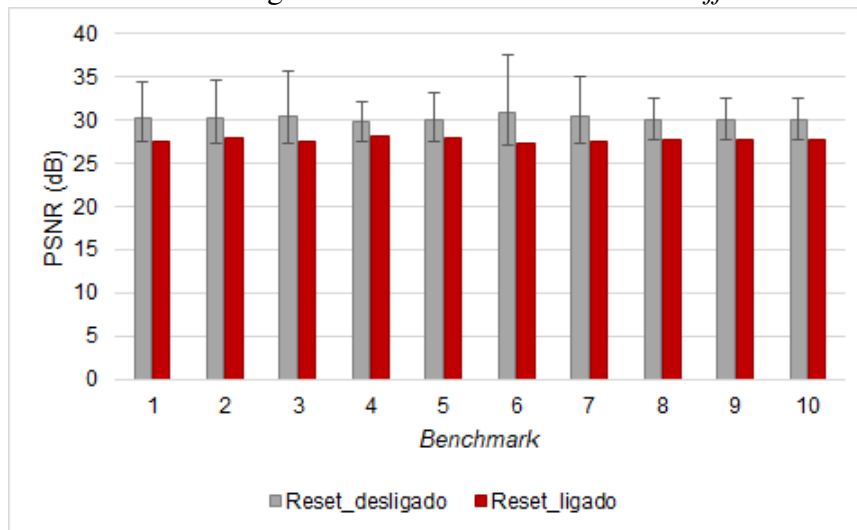
5.3.3 Resultados da modelagem de avaliação da inativação do sinal de *reset*

A análise dinâmica da qualidade no conjunto das 10 imagens do *benchmark* BSD ocorreu primeiramente sem adição de ruído nessas imagens e, posteriormente, com adição de ruído gaussiano nesse conjunto de imagens em avaliação. Conforme já mencionado, utilizou-se estaticamente as máscaras mostradas na Tabela 2 para se desligar a quantidade de *bits* que se desejava. Foram elaborados 5 códigos python, no qual cada um deles representava uma versão da arquitetura do filtro gaussiano 2D, com largura de *bits* variando de 3 a 7 *bits*. Após as manipulações desse procedimento, foram calculados via *script* python os valores de PSNR.

A seguir, as Figuras 17 e 18 apresentam uma análise combinatória para as 32 combinações das 10 imagens do *benchmark* BSD, a partir da versão de redução de 5 *bits* da arquitetura de filtro gaussiano 2D. O *Reset_desligado* é implementado por essa avaliação combinatória que possibilita uma análise mais detalhada do comportamento da reconfiguração, apresentando baixo *overhead* ao não utilizar o sinal *reset*. A Figura 17 apresenta os valores de PSNR, considerando os cenários:

1. *Reset_ligado*;
2. *Reset_desligado*;
3. Configuração de redução de 5 LSBs desabilitados.

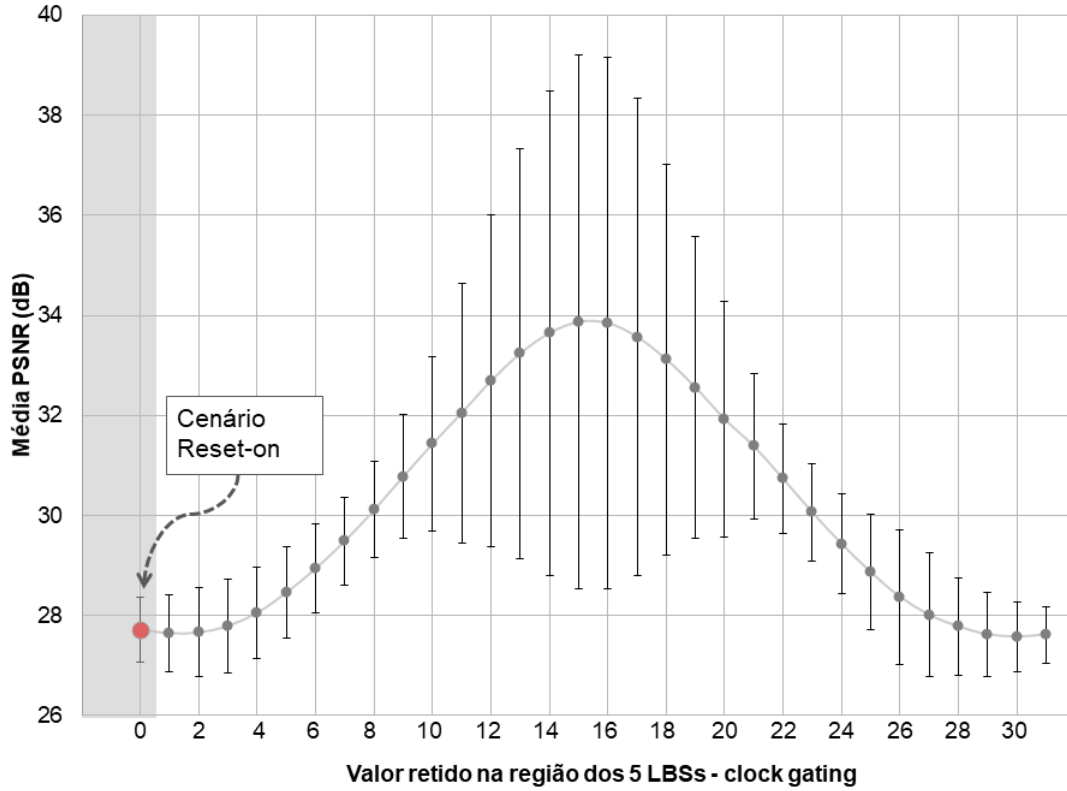
A barra *Reset_desligado* mostra a resposta média, enquanto a barra de erros indica os níveis mínimo e máximo de PSNR para todos os valores pré-carregados possíveis nos 5 LSBs. Pode-se notar que esta abordagem de não utilizar o *reset* atinge níveis de PSNR variando de 27,1dB a aproximadamente 40dB, que são valores de PSNR semelhantes ou mais altos do que o cenário *Reset_ligado*. Isso indica que o uso de redefinição impõe um ciclo de *clock* extra para a configuração, não resultando em melhor qualidade.

Figura 17: *Reset-On* versus *ResetOff*

Fonte: A autora

Portanto, pode-se observar que é preferível não utilizar o *reset* antes de cada nova configuração a ser escolhida. Assim, durante uma reconfiguração, o *reset* do LSB poderia ser evitado uma vez que a qualidade da aplicação seja preservada, reduzindo o *overhead* em termos de tempo de configuração e do *hardware* no projeto de controle lógico e tendo resultados de eficiência energética positivos no bloco de controle.

A Figura 18 mostra os valores médios de PSNR considerando as 10 imagens selecionadas do *benchmark*, além de todos os valores possíveis retidos nos 5 LSBs antes da configuração aproximada de 3 *bits* de redução ser selecionada. Observa-se que o cenário de *Reset_ligado* é o ponto mais à esquerda no gráfico de dispersão, enquanto que todos os 32 pontos (ou seja, valores que variam de 0 a 31) representam o cenário de *Reset_desligado*. Como esperado, o cenário de *Reset_desligado* permite a compensação do erro nessas imagens na maioria dos casos e uma resposta semelhante quando os 5 LSBs são todos valores "0" ou "1" (ou seja, valores 0 e 31). No primeiro valor retido, o erro representa uma tendência positiva enquanto no último, o erro representa uma tendência negativa.

Figura 18: Análise das combinações do sinal de *reset*

Fonte: A autora

Por outro lado, a maioria dos pontos na Figura 18 apresenta uma média substancialmente superior de qualidade. Além disso, as barras de erro na Figura 18 indicam a regra de variação 3-sigma. Pode-se observar uma variação maior para os valores centrais, mas os níveis mais baixos não são piores do que os valores “0” e “31”. Com base nisso, pode-se concluir que o cenário *Reset_desligado* contribui para a compensação do erro intrínseco, reduzindo ainda mais o número de ciclos para processar um conjunto de imagens. O número total de ciclos n é dado pela Equação 8.

$$n = nconf \cdot l + s + nrst \quad (8)$$

onde $nconf$ é o número de configurações usadas durante o tempo de execução, l é a latência, s é o número de amostras a serem processadas, e $nrst$ é o número de sinais de *reset* adicionais necessários para limpar os LSBs entre as configurações diferentes. A arquitetura configurável proposta tem dois estágios de *pipeline*, o que produz uma latência de dois ciclos. Para 30 quadros Full HD de banda única, 62.208.000 amostras precisam ser processadas pelo filtro. Isso indica que a arquitetura deve processar 2.488.320 blocos de entrada com 5x5 amostras, onde 10% deste valor sendo configurações resultariam em cerca de 62,95 milhões para o cenário *Reset_ligado*. O cenário *Reset_desligado* reduz o número total de ciclos para cerca de 62,71 milhões.

5.3.4 Resultados da modelagem para analisar a degradação de imagens através da concatenação em diferentes larguras de *bits*

Após as imagens do *benchmark* BSD passarem pelo filtro gaussiano 2D com aproximações definidas em tempo de projeto, nas diferentes versões (3 a 8 *bits*), elas foram recortadas. Posteriormente, houve a concatenação das faixas horizontais, remontando portanto as imagens, conforme mostra a Figura 19, onde é possível realizar uma análise através da percepção visual e também objetiva através da métrica PSNR.

A seguir, a Figura 19 apresenta uma imagem, onde a região de interesse é referente ao pássaro. Essa imagem passa por diferentes versões do filtro gaussiano 2D; logo, da esquerda para direita é possível ver resultados de: a) 8 *bits*, b) 7 *bits* e assim, e assim até f) 3 *bits*. A partir disso, constata-se pela percepção visual, que a região de interesse na imagem não sofre degradação perceptível; porém, ao redor do pássaro ocorrem oscilações conforme se diminui a quantidade de *bits* na imagem.

Figura 19: Percepção visual. a) 8 *bits*; b) 7 *bits*; c) 6 *bits*; d) 5 *bits*; e) 4 *bits* f) 3 *bits*.



Fonte: A autora

A Figura 20 apresenta uma imagem concatenada ampliada da versão de arquitetura com 5 *bits* de redução sem conter adição de ruído. Esta imagem obtém um PSNR de 35.075dB em relação à imagem do pássaro com largura de 8 *bits*. Já quando se calcula o PSNR da imagem original com uma imagem de 1 *bit* de redução, obtém-se um PSNR maior de 50,89 dB. Isso ocorre devido à redução de 1 *bit* apresentar pouca diferença de *pixels* em relação à original.

A Figura 21 apresenta uma imagem concatenada ampliada da versão de arquitetura com 5 *bits* de redução contendo adição de ruído gaussiano, com média $\mu = 0$ e variância $\sigma^2 = 0,01$. Esta imagem obtém um PSNR de 27,47 dB em relação à imagem do pássaro com largura de 8 *bits*. Observando-se a imagem é possível analisar que a qualidade da imagem sofreu pouca degradação na região de interesse, mas ainda assim, apresenta ruído nas regiões ao redor da imagem do pássaro.

Figura 20: Percepção visual - redução de 5 *bits* sem adição de ruído



Fonte: A autora

Figura 21: Percepção visual - redução de 5 *bits* com adição de ruído gaussiano



Fonte: A autora

A fim de analisar a imagem do pássaro com diferentes aproximações através das faixas, fez-se uma montagem conforme apresentado nas Figuras 22 e 23, no qual a imagem foi dividida em 10 faixas, sendo cortada de 50 em 50 *pixels*. As imagens representam em: a) pássaro com 8 *bits*, apresenta uma imagem com a versão da filtro gaussiano 2D de 8 *bits* nas faixas que contém as regiões de interesse e nas áreas homogêneas, ou seja, ao redor do pássaro aplicou-se a versão de redução de largura de 5 *bits*, enquanto que a imagem em b) pássaro com redução de 5 *bits*, representa o contrário, ou seja, nas regiões

de interesse utilizou-se a versão de redução de largura de 5 *bits* e nas regiões homogêneas usou-se a versão da arquitetura de 8 *bits*.

A Figura 22 apresentou os seguintes resultados: a imagem a) teve um PSNR de 31,053dB, enquanto a imagem b) obteve um PSNR de 31,072dB. Tais resultados demonstram que os valores de PSNR foram quase idênticos, comprovando neste caso, que utilizar uma versão com maior redução na quantidade de *bits* na área de interesse ou utilizar uma versão mais precisa, são equivalentes comprovado pela métrica, além de visualmente não apresentarem diferenças detectáveis a olho humano. Porém ao se observar ao redor do pássaro, nota-se claramente a diferença nas faixas através das tonalidades de cor.

Figura 22: Análise níveis de aproximação em uma mesma imagem. a) pássaro com 8 *bits* b) pássaro com 3 *bits*.

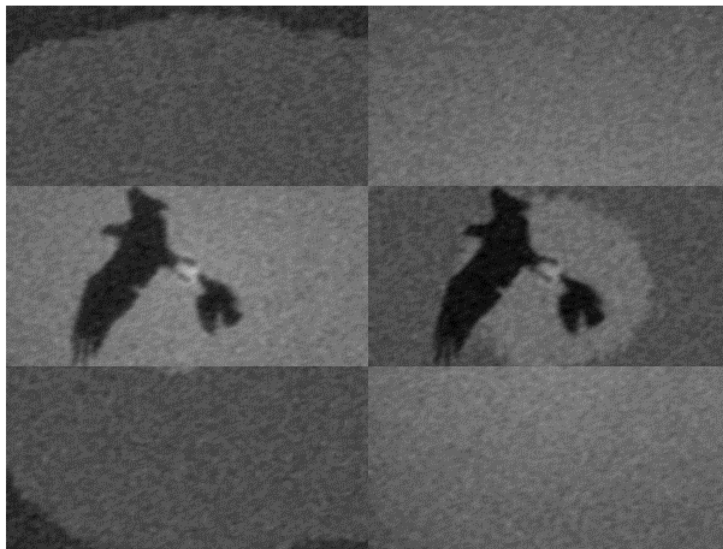


Fonte: A autora

A Figura 23 apresentou os seguintes resultados: a imagem a) teve um PSNR de 29,23dB, enquanto a imagem b) obteve um PSNR de 32,25dB, essa montagem obteve um valor melhor de PSNR. Ainda consegue-se, através da percepção visual, notar que a região do pássaro não sofre tanta degradação visual, mas ao seu redor ficam claramente visíveis as diferenças nas faixas através das tonalidades de cor.

Através da análise de qualidade dessa modelagem, constatou-se que a métrica PSNR não reproduz a sensação visual. Com isso torna-se adequado explorar, por exemplo, a métrica *structural similarity index measure* (SSIM), a fim de melhorar a análise quantitativa da região de interesse. Outro aspecto a ser analisado refere-se a realizar essa modelagem de concatenação em diferentes larguras de *bits* realizando cortes em formatos de blocos na imagem. Os códigos que realizam a modelagem em diferentes aproximações em uma mesma imagem, encontram-se disponíveis no apêndice C.

Figura 23: Análise níveis de aproximação em uma mesma imagem com ruído gaussiano. a) pássaro com 8 *bits* b) pássaro com 3 *bits*.



Fonte: A autora

5.4 Comparação com o Estado da Arte

Realizando uma comparação com os trabalhos relacionados, a arquitetura configurável elaborada neste trabalho atinge reduções semelhantes em termos de consumo de energia quando comparado ao estado da arte. A Tabela 8 resume os principais pontos comparativos deste trabalho com os trabalhos relacionados. O trabalho de Kahng (KAHNG; KANG, 2012) apresenta redução de potência dinâmica de 51,6%. A abordagem configurável apresentada por Soares (SOARES; COSTA; BAMPI, 2019) mostra reduções de energia que variam de 29,1% a 64%.

Na análise desta pesquisa, a comparação foi feita em termos de energia ou redução de potência, pois as arquiteturas e, também as tecnologias, são diferentes. Portanto, o foco refere-se à medir o impacto da energia ou as reduções de potência ao se considerar a arquitetura mais precisa de cada trabalho. A ampla gama de reduções de energia é encontrada por (SOARES et al., 2019), de 7,7% até 73,2%. Contudo, este é um sistema configurável em tempo de projeto. A melhor redução de energia relatada anteriormente para uma arquitetura de tempo de execução é de 64% de (KAHNG; KANG, 2012), ligeiramente superior aos melhores resultados desta pesquisa (aproximadamente 59%).

Considerando apenas imagens com adição de ruído, as referências (KAUSHIK; KUMAR, 2015), (SOARES; COSTA; BAMPI, 2019) e (KAHNG; KANG, 2012) obtiveram valores de PSNR variando entre 20 dB a 29,36 dB, enquanto o presente trabalho atingiu uma faixa de 27,5 dB a 51,2 dB. Os níveis mais baixos e mais altos estão relacionados às configurações aproximadas de 3 e 7 *bits*, respectivamente. Esta análise considera os novos resultados com base na avaliação *reset_ligado/reset_desligado*.

Tabela 8: Comparação geral com os trabalhos relacionados

Trabalho	Configurabilidade	Redução de Energia	Tecnologia	Qualidade
(SOARES et al., 2019)	Tempo de projeto	7,7% a 73,2%	ASIC 45 nm	30 a 50 dB
(JULIO et al., 2015)	Tempo de projeto	40% e 25%	ASIC 45 nm	30,6 a 49,7 dB
(KAUSHIK; KUMAR, 2015)	Tempo de execução	-	FPGA 45 nm	20 a 29,36 dB
(KAHNG; KANG, 2012)	Tempo de execução	12,4% a 51,6%	ASIC TSMC 65GP	24,5 dB
(SOARES; COSTA; BAMPI, 2019)	Tempo de execução	29,1% a 64%	ASIC 45 nm	20,6 a 25,7 dB
Este trabalho	Tempo de projeto	13,18% a 67,88%	ASIC 65 nm	30 a 45 dB
Este trabalho	Tempo de execução	9,87% a 59%	ASIC 65 nm	27,5 a 51,2 dB

Fonte: A autora

Por fim, os resultados desta pesquisa mostram que a exploração da largura de *bits* para aceleradores de *hardware* que processam imagens e vídeos, permite uma importante contribuição para o escopo da área de computação aproximada. Quando comparado com os trabalhos relacionados (SOARES et al., 2019) e (SOARES; COSTA; BAMPI, 2019), as reduções de energia demonstradas neste texto estão na mesma ordem. A partir disso, pode-se notar que este trabalho apresenta contribuições de pesquisa, para a área de processamento de imagens e de eficiência energética em circuitos CMOS.

6 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma exploração da técnica de computação aproximada, de redução da largura de *bits* para o projeto de um acelerador de *hardware* do filtro gaussiano 2D, com a finalidade de fornecer múltiplos perfis de qualidade, potência e desempenho. A partir disso, esta pesquisa encontra-se inserida na concepção de aceleradores de *hardware*, que utilizam o filtro gaussiano 2D, a fim de usar estratégias para aumentar a compensação entre eficiência energética e qualidade da aplicação, voltado para as áreas de processamento digital de imagens e computação aproximada.

Foram desenvolvidas seis versões, ou seja, de 3 a 8 *bits* da arquitetura de aproximação definida em tempo de projeto em *hardware*. Posteriormente, aconteceu o desenvolvimento da versão do filtro gaussiano 2D com precisão em tempo de execução, onde a configuração do número de *bits* ocorreu de forma dinâmica pelo *hardware* ou usuário. A partir disso, houveram análises de potência e área para essas duas arquiteturas do filtro gaussiano 2D, assim como a elaboração de experimentos de modelagem para avaliar a qualidade.

Os resultados para a versão do filtro gaussiano 2D com aproximação definida em tempo de projeto, mostraram uma redução considerável da energia. Para as tecnologias testadas, a redução de energia varia de 13,18% a 67,8%. Em termos de área, as reduções variam de 60% a 11%, sendo proporcional ao número de *bits* adotados nas arquiteturas do filtro gaussiano 2D. No quesito qualidade da aplicação, a análise dinâmica considerando o *benchmark* em avaliação, com adição de ruído gaussiano apresentou níveis de PSNR variando de 27,5 dB a 51,2 dB para as configurações aproximadas de 3 a 7 *bits*.

O filtro gaussiano configurável em tempo de execução alcançou redução energética por operação, variando de 9,87% até 58,96%, demonstrando uma tendência linear ao observar a relação entre consumo de energia e largura de *bits*. A arquitetura apresenta um *overhead* de até 15% em comparação com o filtro gaussiano de 8 *bits* 2D tradicional para a tecnologia de 65 nm. No entanto, é um preço baixo a se pagar em comparação com a economia de energia permitida.

Quanto à avaliação do impacto da redefinição do sinal de *reset* para avaliação da qualidade do filtro gaussiano 2D, o trabalho atingiu uma faixa de 27,1 dB a aproximadamente

40 dB para a configuração de 5 *bits* do filtro. Os níveis mais baixos e mais altos estão relacionados às configurações aproximadas de 3 e 7 bits, respectivamente. Esses resultados indicaram que não é necessário usar o sinal *reset*, antes de cada nova configuração a ser escolhida, reduzindo também a complexidade de controle do sistema configurável.

Por fim, esse trabalho insere-se no âmbito de trabalhos relacionados, trazendo uma contribuição de pesquisa para o processamento de imagens com melhor qualidade e compensação energética. Ademais, o trabalho está inserido no projeto de aceleradores de *hardware*, que utilizam o filtro Gaussiano 2D, buscando estratégias para potencializar o *trade-off* entre eficiência energética e qualidade da aplicação. Os resultados também indicam que o projeto de aceleradores de *hardware* com precisão configurável é viável para gerenciar a eficiência energética e a qualidade da imagem.

Como trabalhos futuros, pretende-se implementar um componente preditivo, onde o compromisso entre eficiência energética e qualidade da aplicação possa ser otimizado em função dos requisitos do sistema. Ademais, realizar testes com imagens com resolução em 4k (3840 x 2160 *pixels*) para se observar os níveis de degradação de qualidade. Adicionalmente, seria interessante validar o filtro gaussiano 2D em aplicações como detecção de bordas e visão computacional.

6.1 Publicações

A execução deste trabalho gerou as publicações listadas abaixo, considerando o tempo desde o início do mestrado em Engenharia de Computação-FURG até o presente momento. A publicação de (BORGES et al., 2019) foi referente ao trabalho de conclusão de curso em Engenharia de Computação na Unipampa, submetida anteriormente ao início do mestrado, mas apresentada durante o curso de mestrado. As demais são relacionadas às etapas de desenvolvimento desse projeto de dissertação.

1. BORGES, T.; MEINHARDT, C.; SOARES, L. Accuracy-configurable 2D Gaussian filter architecture design toward the energy-efficient image and video processing. Design&Test 2021, [S.l.],v.1, n.1, p.1–5, 2021. (Em revisão);
2. BORGES, T.; MEINHARDT, C.; SOARES, L. A Low-impact overhead technique for configurable and energy-efficient 2D Gaussian Filter Design. Simpósio Sul de Microeletrônica 2021, [S.l.],v.1, n.1, p.1–4, 2021.
3. BORGES, T.; MEINHARDT, C.; SOARES, L. A Fine-grained Methodology for Accuracy-configurable and Energy-efficient Gaussian Filters Design. SBCCI 2020, [S.l.],v.1, n.1, p.1–6, 2020.
4. BORGES, T.; MEINHARDT, C.; SOARES, L. Bit width Exploration for Energy-efficient Image and Video Processing. Simpósio Sul de Microeletrônica 2020, [S.l.],v.1, n.1, p.1–4, 2020.

5. BORGES, T.; MEINHARDT, C.; SOARES, L. Projeto de um filtro gaussiano 2D com precisão configurável energeticamente eficiente. Mostra de Produção Universitária 2020, [S.l.],v.1, n.1, p.1–3, 2020.
6. BORGES, T.; MEINHARDT, C.; SOARES, L. Explorando Computação Aproximada via redução de tamanho de imagens para eficiência energética. Mostra de Produção Universitária 2019, [S.l.],v.1, n.1, p.1–3, 2019.
7. BORGES, T. and RAMOS, F. Multiplier Architectures aiming Energy Efficiency within the Context of the HEVC Entropy Coding. Simpósio Sul de Microeletrônica 2019, [S.l.],v.1, n.1, p.1–4, 2019.

REFERÊNCIAS

AKBARI, O.; KAMAL, M.; AFZALI-KUSHA, A.; PEDRAM, M.; SHAFIQUE, M. Toward Approximate Computing for Coarse-Grained Reconfigurable Architectures. **IEEE Micro**, [S.l.], v.38, n.6, p.63–72, 2018.

ALEXANDRE, D. S.; TAVARES, J. Factores da percepção visual humana na visualização de dados. In: CMNE 2007-CONGRESSO DE MÉTODOS NUMÉRICOS EM ENGENHARIA, XXVIII CILAMCE-CONGRESSO IBERO LATINO-AMERICANO SOBRE MÉTODOS COMPUTACIONAIS EM ENGENHARIA, PORTO, PT, 2007. **Anais...** [S.l.: s.n.], 2007.

ALMOHAMMAD, A.; GHINEA, G. Stego image quality and the reliability of PSNR. In: INTERNATIONAL CONFERENCE ON IMAGE PROCESSING THEORY, TOOLS AND APPLICATIONS, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.215–220.

AMBROSE, G.; HARRIS, P. **Imagem**. [S.l.]: Porto Alegre: Bookman, 2009.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer networks**, [S.l.], v.54, n.15, p.2787–2805, 2010.

BARUA, H. B.; MONDAL, K. C. Approximate Computing: A Survey of Recent Trends—Bringing Greenness to Computing and Communication. **Journal of The Institution of Engineers (India): Series B**, [S.l.], p.1–8, 2019.

BENARA, V.; PURINI, S. Accurus: A fast convergence technique for accuracy configurable approximate adder circuits. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI (ISVLSI), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.577–582.

BORGES, T.; RAMOS, F.; MEINHARDT, C.; SOARES, L. Multiplier Architectures aiming Energy Efficiency within the Context of the HEVC Entropy Coding. **Simpósio Sul de Microeletrônica 2019**, [S.l.], v.1, n.1, p.1–4, 2019.

CANNY, J. A computational approach to edge detection. **IEEE Transactions on pattern analysis and machine intelligence**, [S.l.], n.6, p.679–698, 1986.

CHIPPA, V. K.; VENKATARAMANI, S.; CHAKRADHAR, S. T.; ROY, K.; RAGHUNATHAN, A. Approximate computing: An integrated hardware approach. In: ASILOMAR CONFERENCE ON SIGNALS, SYSTEMS AND COMPUTERS, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.111–117.

CISCO, U. **Cisco annual internet report (2018–2023) white paper.**

CLARK, A. **Documentation Pillow.** Access date: 08 jun. 2021.

GONZALEZ, R. C.; WOODS, R. C. **Processamento digital de imagens .** [S.l.]: Pearson Educación, 2009.

GUPTA, V.; MOHAPATRA, D.; RAGHUNATHAN, A.; ROY, K. Low-power digital signal processing using approximate adders. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.32, n.1, p.124–137, 2012.

HORE, A.; ZIOU, D. Image quality metrics: PSNR vs. SSIM. In: INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.2366–2369.

IMANI, M.; GARCIA, R.; HUANG, A.; ROSING, T. Cade: Configurable approximate divider for energy efficiency. In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.586–589.

JESUS, E. O.; COSTA JR, R. A utilização de filtros gaussianos na análise de imagens digitais. **Proceeding Series of the Brazilian Society of Computational and Applied Mathematics**, [S.l.], v.3, n.1, 2015.

JIANG, H.; SANTIAGO, F. J. H.; MO, H.; LIU, L.; HAN, J. Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications. **Proceedings of the IEEE**, [S.l.], v.108, n.12, p.2108–2135, 2020.

JULIO, R. O.; SOARES, L. B.; COSTA, E.; BAMPI, S. Energy-efficient Gaussian filter for image processing using approximate adder circuits. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS, AND SYSTEMS (ICECS), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.450–453.

KAHNG, A. B.; KANG, S. Accuracy-configurable adder for approximate arithmetic designs. In: ANNUAL DESIGN AUTOMATION CONFERENCE, 49., 2012. **Proceedings...** [S.l.: s.n.], 2012. p.820–825.

KAMILARIS, A.; PITSILLIDES, A. Mobile phone computing and the internet of things: A survey. **IEEE Internet of Things Journal**, [S.l.], v.3, n.6, p.885–898, 2016.

KANG, Y.; KIM, J.; KANG, S. Novel approximate synthesis flow for energy-efficient FIR filter. In: IEEE 34TH INTERNATIONAL CONFERENCE ON COMPUTER DESIGN (ICCD), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.96–102.

KAUSHIK, S.; KUMAR, N. V. Energy-efficient approximate 2D Gaussian smoothing filter for error tolerant applications. In: IEEE INTERNATIONAL ADVANCE COMPUTING CONFERENCE (IACC), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p.789–794.

KHANDELWAL, S.; CHOUDHURY, Z.; SHRIVASTAVA, S.; PURINI, S. Accelerating Local Laplacian Filters on FPGAs. In: INTERNATIONAL CONFERENCE ON FIELD-PROGRAMMABLE LOGIC AND APPLICATIONS (FPL), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.109–114.

LEE, J.; TANG, H.; PARK, J. Energy efficient canny edge detector for advanced mobile vision applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.28, n.4, p.1037–1046, 2016.

MARQUES FILHO, O.; NETO, H. V. **Processamento digital de imagens**. [S.l.]: Brasport, 1999.

MARTIN, D.; FOWLKES, C.; TAL, D.; MALIK, J. et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: 2020 30TH , 2001. **Anais...** [S.l.: s.n.], 2001.

MITTAL, S. A survey of techniques for approximate computing. **ACM Computing Surveys (CSUR)**, [S.l.], v.48, n.4, p.1–33, 2016.

NOGUEIRA, L. **Dados mostram o crescimento impressionante da internet em 10 anos**. Access date: 29 mai. 2020.

OLIVEIRA, J. de; SOARES, L.; COSTA, E.; BAMPI, S. Exploiting approximate adder circuits for power-efficient Gaussian and Gradient filters for Canny edge detector algorithm. In: IEEE 7TH LATIN AMERICAN SYMPOSIUM ON CIRCUITS & SYSTEMS (LASCAS), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.379–382.

OTSUKA, T.; FUKUSHIMA, N.; MAEDA, Y.; SUGIMOTO, K.; KAMATA, S.-i. Optimization of Sliding-DCT Based Gaussian Filtering for Hardware Accelerator. In: IEEE INTERNATIONAL CONFERENCE ON VISUAL COMMUNICATIONS AND IMAGE PROCESSING (VCIP), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p.423–426.

PEDRINI, H.; SCHWARTZ, W. R. **Análise de imagens digitais: princípios, algoritmos e aplicações**. [S.l.]: Thomson Learning, 2008.

PHAM-QUOC, C.; THINH, T. N. An efficient runtime adaptable floating-point Gaussian filtering core. In: NAFOSTED CONFERENCE ON INFORMATION AND COMPUTER SCIENCE, 2017., 2017. **Anais...** [S.l.: s.n.], 2017. p.183–188.

QIQIEH, I.; SHAFIK, R.; TARAWNEH, G.; SOKOLOV, D.; DAS, S.; YAKOVLEV, A. Significance-driven logic compression for energy-efficient multiplier design. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, [S.l.], v.8, n.3, p.417–430, 2018.

QIQIEH, I.; SHAFIK, R.; TARAWNEH, G.; SOKOLOV, D.; YAKOVLEV, A. Energy-efficient approximate multiplier design using bit significance-driven logic compression. In: DESIGN, AUTOMATION & TEST IN EUROPE CONFERENCE & EXHIBITION (DATE), 2017, 2017. **Anais...** [S.l.: s.n.], 2017. p.7–12.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. **RITA**, [S.l.], v.13, n.2, p.11–42, 2006.

RAMYA, R.; MOORTHY, S. Design and implementation of accuracy configurable multi-precision multiplier architecture for signal processing applications. In: IEEE RECENT ADVANCES IN INTELLIGENT COMPUTATIONAL SYSTEMS (RAICS), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.89–93.

REDA, S.; SHAFIQUE, M. **Approximate Circuits: Methodologies and CAD**. [S.l.]: Springer, 2018.

RODRIGUES, G. S.; FONSECA, J.; BENEVENUTI, F.; KASTENSMIDT, F.; BOSIO, A. Exploiting approximate computing for low-cost fault tolerant architectures. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN (SBCCI), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.1–6.

RUBIO-GONZÁLEZ, C.; NGUYEN, C.; NGUYEN, H. D.; DEMMEL, J.; KAHAN, W.; SEN, K.; BAILEY, D. H.; IANCU, C.; HOUGH, D. Precimonious: Tuning assistant for floating-point precision. In: SC'13: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING, NETWORKING, STORAGE AND ANALYSIS, 2013. **Anais...** [S.l.: s.n.], 2013. p.1–12.

SEKANINA, L. Introduction to approximate computing: Embedded tutorial. In: IEEE 19TH INTERNATIONAL SYMPOSIUM ON DESIGN AND DIAGNOSTICS OF ELECTRONIC CIRCUITS SYSTEMS (DDECS), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p.1–6.

SEKANINA, L.; VASICEK, Z.; MRAZEK, V. Approximate Circuits in Low-Power Image and Video Processing: The Approximate Median Filter. **Radioengineering**, [S.l.], v.26, n.3, 2017.

SOARES, L. B.; COSTA, E. A. C. da; BAMPI, S. A Configurable Pruning Gaussian Image Filter for Energy-Efficient Edge Detection. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS (ICECS), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p.666–669.

SOARES, L. B.; ROSA, M. M. A. da; DINIZ, C. M.; COSTA, E. A. C. da; BAMPI, S. Design Methodology to Explore Hybrid Approximate Adders for Energy-Efficient Image and Video Processing Accelerators. **IEEE Transactions on Circuits and Systems I: Regular Papers**, [S.l.], v.66, n.6, p.2137–2150, 2019.

STROLLO, A. G. M.; ESPOSITO, D. Approximate computing in the nanoscale era. In: INTERNATIONAL CONFERENCE ON IC DESIGN TECHNOLOGY (ICICDT), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p.21–24.

VENKATARAMANI, S.; CHAKRADHAR, S. T.; ROY, K.; RAGHUNATHAN, A. Approximate computing and the quest for computing efficiency. In: ANNUAL DESIGN AUTOMATION CONFERENCE, 52., 2015. **Proceedings...** [S.l.: s.n.], 2015. p.120.

WU, Q.; PEDRAM, M.; WU, X. Clock-gating and its application to low power design of sequential circuits. **IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications**, [S.l.], v.47, n.3, p.415–420, 2000.

XU, Q.; MYTKOWICZ, T.; KIM, N. S. Approximate computing: A survey. **IEEE Design & Test**, [S.l.], v.33, n.1, p.8–22, 2015.

YE, R.; WANG, T.; YUAN, F.; KUMAR, R.; XU, Q. On reconfiguration-oriented approximate adder design and its application. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD), 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p.48–54.

ZHU, N.; GOH, W. L.; WANG, G.; YEO, K. S. Enhanced low-power high-speed adder for error-tolerant application. In: INTERNATIONAL SOC DESIGN CONFERENCE, 2010., 2010. **Anais...** [S.l.: s.n.], 2010. p.323–327.

7 APÊNDICE A - CONJUNTO DE IMAGENS COM LARGURA DE 8 BITS DO BENCHMARK BSD

Figura 24: Imagem 01 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 25: Imagem 02 do Benchmark BSD.



Resolução: 321x481. Fonte: (MARTIN et al., 2001)

Figura 26: Imagem 03 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 27: Imagem 04 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 28: Imagem 05 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 29: Imagem 06 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 30: Imagem 07 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 31: Imagem 08 do Benchmark BSD.



Resolução: 321x481. Fonte: (MARTIN et al., 2001)

Figura 32: Imagem 09 do Benchmark BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 33: Imagem 10 do *Benchmark* BSD.



Resolução: 481x321. Fonte: (MARTIN et al., 2001)

Figura 34: Imagem 11 do *Benchmark* BSD.



Resolução: 321x481. Fonte: (MARTIN et al., 2001)

8 APÊNDICE B - CÓDIGOS UTILIZADOS PARA A IMPLEMENTAÇÃO EM *PYTHON* DA MODELAGEM DE AVALIAÇÃO DE INATIVAÇÃO DO RESET ENTRE MUDANÇAS DE CONFIGURAÇÕES

Figura 35: Exemplo de código do filtro gaussiano 2D-redução de 3 bits

File - C:\Users\55539\PycharmProjects\Arquitetura-5-bits\Arquitetura_5_bits.py

```

1 # Arquitetura do Filtro Gaussiano (3 bits de redução)
2 import numpy as np
3 arq = open("passaro.out", "r") # Ler arquivo input
4 linhas = [] # lista para adicionar valores
5 file = open("passaro_5_bits.txt", "w") # colocar resultado num
    arquivo txt
6 for line in arq.readlines(): # Ler as linhas do arquivo
7     linhas.append([int(x) for x in line.split(",")]) # Line.
    split divide uma string em uma lista
8 for x in linhas:
9     a = [] #lista para adicionar valores mascarados
10    count = 0
11    for num in x: # pega-se cada número do arquivo para fazer o
        and com a máscara
12        tmp = num & 248 #aplicando a máscara para desligar 1
        bit menos significativo
13        a.append(tmp) #adiciona resultado dos elementos
        mascarados para uma lista a
14        numBin = format(tmp, '08b') #converte o 0b da frente do
        número
15        print(numBin) #imprime número em binário corrigido
16        numInt = int(numBin[-3::])
17        count = count + numInt # verifica se count contém zero
        , afirmando que a máscara está correta
18        print(count)
19    if count != 0:
20        exit(1) # máscara incorreta
21    # 8 somadores da entrada
22    # somador 1
23    x1 = (a[0] * (2 ** 2))
24    somador1 = (x1 + a[0])
25    # somador 2
26    x2 = (a[1] * (2 ** 2))
27    somador2 = (x2 + a[1])
28    # somador 3
29    somador3 = (somador1 + somador2)
30    # somador 4
31    x3 = (a[2] * (2 ** 2))
32    somador4 = (x3 + a[2])
33    # somador 5
34    x4 = (a[3] * (2 ** 2))
35    somador5 = (x4 + a[3])
36    # somador 6
37    somador6 = (somador4 + somador5)
38    # somador 7
39    x5 = (a[4] * (2 ** 3))
40    somador7 = (x5 + a[4])
41    # somador 8
42    x6 = (a[5] * (2 ** 3))
43    somador8 = (x6 + a[5])

```

Page 1 of 3

Figura 36: Continuação do código do filtro gaussiano 2D-redução de 3 bits

File - C:\Users\55539\PycharmProjects\Arquitetura-5-bits\Arquitetura_5_bits.py

```

44     # somador 9
45     somador9 = (somador7 + somador8)
46     # somador 10
47     somador10 = (somador6 + somador9)
48     # somador 11
49     x7 = (a[6] * (2 ** 3))
50     somador11 = (x7 + a[6])
51     # somador 12
52     x8 = (a[7] * (2 ** 3))
53     somador12 = (x8 + a[7])
54     # somador 13
55     somador13 = (somador11 + somador12)
56     # subtrator 1
57     subtrator1 = (a[8] * (2 ** 4)) - a[8]
58     # somador 14
59     somador14 = ((a[9] * (2 ** 3)) + (a[9] * (2 ** 2)))
60     # somador 15
61     somador15 = ((a[10] * (2 ** 3)) + (a[10] * (2 ** 2)))
62     # somador 16
63     somador16 = (somador14 + somador15)
64     # somador 17
65     somador17 = ((a[11] * (2 ** 3)) + (a[11] * (2 ** 2)))
66     # somador 18
67     somador18 = ((a[12] * (2 ** 3)) + (a[12] * (2 ** 2)))
68     # somador 19
69     somador19 = (somador17 + somador18)
70     # somador 20
71     somador20 = (somador16 + somador19)
72     # somador 21
73     somador21 = (somador13 + subtrator1)
74     # somador 22
75     somador22 = (somador21 + somador20)
76     # somador 23
77     somador23 = ((a[13] * (2 ** 2)) + (a[14] * (2 ** 2)))
78     # somador 24
79     somador24 = ((a[15] * (2 ** 2)) + (a[16] * (2 ** 2)))
80     # somador 25
81     somador25 = (somador23 + somador24)
82     # somador 26
83     somador26 = ((a[17] * (2 ** 2)) + (a[18] * (2 ** 2)))
84     # somador 27
85     somador27 = ((a[19] * (2 ** 2)) + (a[20] * (2 ** 2)))
86     # somador 28
87     somador28 = (somador26 + somador27)
88     # somador 29
89     somador29 = ((a[21] * (2 ** 1)) + (a[22] * (2 ** 1)))
90     # somador 30
91     somador30 = ((a[23] * (2 ** 1)) + (a[24] * (2 ** 1)))
92     # somador 31
93     somador31 = (somador29 + somador30)

```

Page 2 of 3

Figura 37: Continuação do código do filtro gaussiano 2D-redução de 3 bits

File - C:\Users\55539\PycharmProjects\Arquitetura-5-bits\Arquitetura_5_bits.py

```

94     # somador 32
95     somador32 = (somador3 + somador25)
96     # somador 33
97     somador33 = (somador28 + somador31)
98     # somador 34
99     somador34 = (somador32 + somador33)
100    # somador 35
101    somador35 = (somador10 + somador34)
102    # somador 36
103    somador36 = (somador22 + somador35)
104    # somador 37
105    somador37 = ((somador36 / (2 ** 8)) + (somador36 / (2 ** 9
106    )))
107    # somador 38
108    somador38 = ((somador36 / (2 ** 12)) + (somador36 / (2 **
109    13)))
110    # somador 39
111    somador39 = (somador37 + somador38)
112    # registrador de saída de 8 bits
113    reg = np.uint8(somador39)
114    file.write(str(reg) + '\n')
115    arq.close()

```

Figura 38: Transformação arquivo JPG/BMP para .out.

```

1 import numpy
2 import cv2
3 import os
4
5 #lista todos os nomes de arquivos no diretório
6 fileList = [f for f in os.listdir('images') if os.path.isfile(os.path.join('images', f))]
7 for nameImage in fileList:
8     img=cv2.imread(os.path.join('images',nameImage),0)
9     nRows = img.shape[0]
10    nColumns = img.shape[1]
11
12    #efetua o padding nas imagens
13    img1=numpy.append(img,img[:nColumns-2:nColumns],axis=1)
14    img2=numpy.append(img1[:,0:2],img1,axis=1)
15    img3=numpy.append(img2,img2[nRows-2:nRows:],axis=0)
16    img=numpy.append(img3[0:2:],img3,axis=0)
17
18    #deleta variáveis desnecessárias
19    del img1
20    del img2
21    del img3
22    imgFile = numpy.uint8(numpy.zeros((1,25)))
23
24    #efetua a reorganização dos dados para o formato esperado para filtragem
25    for i in range(0,nRows):
26        for j in range(0,nColumns):
27            line = numpy.reshape(img[i:i+5,j:j+5],(1,25),order='C')
28            imgFile = numpy.append(imgFile,line,axis=0)
29    numpy.savetxt(nameImage[:-4]+'.out', imgFile[-(nRows*nColumns):], fmt='%u', delimiter=',')

```

Fonte: Autora

Figura 39: Transformação arquivo txt para JPG ou BMP.

```
Imagem_Filtrada.py x
1 import numpy as np
2 import cv2
3
4 #Cria uma lista
5 lista = []
6
7 #Abre arquivo fonte
8 arqFonte = open('passaro_3_bits.txt', 'r')
9
10 #Popula a lista com os inteiros
11 for linha in arqFonte:
12     lista.append(int(linha))
13
14 #Converte para array
15 npArray = np.asarray(lista)
16
17 #Aplica método reshape no array para criar a matriz
18 reshaped = npArray.reshape(481, 321)
19
20 #Salva a saída
21 np.savetxt('saidaProcessada.txt', reshaped, fmt='%d')
22 cv2.imwrite('C:/Users/55539/PycharmProjects/Imagem_Filtrada/passaro_3_bits.bmp', reshaped)
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()
```

Fonte: Autora

9 APÊNDICE C - CÓDIGOS *PYTHON* UTILIZADOS PARA A IMPLEMENTAÇÃO DA MODELAGEM DE ANÁLISE DE DIFERENTES CONFIGURAÇÕES EM UMA MESMA IMAGEM E INSERÇÃO DE RUÍDO GAUSSIANO

Figura 40: Recorte das fitas das imagens filtradas

```
Recorte_imagem.py x
1 from PIL import ImageOps
2 from PIL import Image
3 img = Image.open("passaro_3_bits.bmp")
4 H=481
5 W=321
6 border = (0, 450, 0, H-500) # left (X), up (Y), LARGURA (L), ALTURA RETIRAR (A)
7 cropped = ImageOps.crop(img, border)
8 teste = cropped.save("FITA10.bmp")
```

Fonte: Autora

Figura 41: Colagem das versões de imagens filtradas

```
colagem.py x
1 from PIL import Image
2
3 im1 = Image.open('FITA1.bmp')
4 im2 = Image.open('FITA2.bmp')
5 im3 = Image.open('FITA3.bmp')
6 im4 = Image.open('FITA4.bmp')
7 im5 = Image.open('FITA5.bmp')
8 im6 = Image.open('FITA6.bmp')
9 im7 = Image.open('FITA7.bmp')
10 im8 = Image.open('FITA8.bmp')
11 im9 = Image.open('FITA9.bmp')
12 im10 = Image.open('FITA10.bmp')
13
14
15
16 dst = Image.new('L', (321, 481))
17 dst.paste(im1, (0, 0))
18 dst.paste(im2, (0, im1.height))
19 dst.paste(im3, (0, ((im1.height) + (im2.height))))
20 dst.paste(im4, (0, ((im1.height) + (im2.height) + (im3.height))))
21 dst.paste(im5, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height))))
22 dst.paste(im6, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height) + (im5.height))))
23 dst.paste(im7, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height) + (im5.height) + (im6.height))))
24 dst.paste(im8, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height) + (im5.height) + (im6.height) + (im7.height))))
25 dst.paste(im9, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height) + (im5.height) + (im6.height) + (im7.height) + (im8.height))))
26 dst.paste(im10, (0, ((im1.height) + (im2.height) + (im3.height) + (im4.height) + (im5.height) + (im6.height) + (im7.height) + (im8.height) + (im9.height))))
27 # return dst
28 dst.save('Imagem_inicio_8_bits_teste02_sem_ruido.bmp')
29
```

Fonte: Autora

Figura 42: Código da implementação do Filtro Gaussiano.

```
PSNR.py x
1  # Cálculo do PSNR
2
3  import numpy
4  import math
5  import cv2
6
7  borboleta = cv2.imread('passaro_8_bits.bmp')
8  contraste = cv2.imread('passaro_com_3_bits.png', 1)
9  def psnr(img1, img2):
10     mse = numpy.mean((img1 - img2) ** 2)
11     if mse == 0:
12         return 100
13     PIXEL_MAX = 255.0
14     return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
15  d = psnr(borboleta, contraste)
16  print(d)
```

Fonte: A autora