UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

JULIO FRANCISCO ROCHA DE OLIVEIRA

Aplicação de Circuitos Somadores Aproximados em Filtros de Processamento de Imagens

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Engenharia Eletrônica e Computação.

Orientador: Prof. Dr. Eduardo Antonio César da Costa
Co-orientador: Prof. Dr. Sérgio Bampi

Pelotas
2016

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Oliveira, Julio Francisco Rocha de

Aplicação de Circuitos Somadores Aproximados em Filtros de Processamento de Imagens / Julio Francisco Rocha de Oliveira. – 2016.

55 f.:il.

Orientador: Dr. Eduardo Antonio César da Costa; Co-orientador: Dr. Sérgio Bampi.

Dissertação (Mestrado) – Universidade Católica de Pelotas. Mestrado em Engenharia Eletrônica e Computação. Pelotas, BR – RS, 2016.

1. Circuitos somadores aproximados. 2. Processamento de imagem 3. Detector de bordas de Canny.

I. Costa, Antonio César da. II. Bampi, Sérgio. III. Aplicação de Circuitos Somadores Aproximados em Filtros de Processamento de Imagens.

AGRADECIMENTOS

Seria difícil agradecer a todos que me apoiaram neste longo caminho, mas com certeza eles sabem da importância que tiveram, não sou muito bom com as palavra mas vou tentar fazer um breve agradecimento.

Primeiramente a Deus pela oportunidade recebida e por me dar forças para superar os desafios. Aos meus pais, pelo incansável apoio e por não medirem esforços para que eu chegasse até aqui. À minha companheira de todas as horas, por estar sempre ao meu lado me incentivando diante dos obstáculos que iam surgindo. Agradeço também ao meu grupo de pesquisa, o doutorando Leonardo Bandeira Soares, meu coorientador professor Dr. Ségio Bampi, e em especial ao meu orientador professor Dr. Eduardo Antonio César da Costa, que sempre esteve presente apoiando e propondo novos desafios a serem vencidos. Enfim, a todos os que contribuíram e me ajudaram a vencer mais essa etapa de minha vida.

"O caminho está aberto a todos, e se uns vencem e alcançam o que almejam, não é porque sejam predestinados, senão porque forçaram os obstáculos com arrojo e tenacidade."

Henrique Maximiliano Coelho Neto

RESUMO

Este trabalho propõe a exploração de circuitos somadores aproximados para a implementação de filtros eficientes em consumo de potência para Processamento de Imagem. O filtro Gaussiano é um operador de convolução que é usado para borrar as imagens e remover ruídos. Por outro lado, o Gradiente de uma imagem quantifica o quanto uma imagem está mudando. Ambos os blocos podem ser implementados em hardware usando apenas operações de deslocamento e somas/subtrações. Nesse trabalho, um conjunto de somadores aproximados é explorado para a implementação de filtros eficientes em termos de energia. As árvores de somadores dos filtros Gaussiano e Gradiente são implementadas usando um somador aproximado baseado na cópia de bits para a saída, bem como somadores tolerantes a erros (ETA - *Error-Tolerant Adders*). As arquiteturas aproximadas são comparadas com as implementações dos filtros com somadores precisos. Como os blocos Gaussiano e Gradiente são partes integrantes do algoritmo de detecção de bordas de Canny, logo as árvores de somadores dos filtros Gaussiano e Gradiente foram implementadas visando a esta aplicação. Em particular, um algoritmo foi proposto no âmbito deste trabalho para encontrar a melhor composição da árvore de somadores nos filtros Gaussiano e Gradiente. Os principais resultados mostram que, para a realização eficiente em potência desse algoritmo, as melhores estratégias consistem na implementação do filtro Gaussiano com o somador ETA I e a implementação do filtro Gradiente com o somador baseado em cópia de bits. Os filtros Gaussiano e Gradiente aproximados foram aplicados ao circuito completo de detecção de bordas de Canny. Os resultados mostraram que as arquiteturas de detecção de bordas de Canny aproximadas, com somadores baseado na cópia de bits e ETAI, na maioria dos casos possuem melhores resultados em relação às métricas de desempenho e precisão, com relação à arquitetura precisa. Os testes foram realizados usando um conjunto de imagens reais. Os resultados da síntese em ASIC mostraram que, as aproximações dos filtros Gaussiano e Gradiente com os somadores baseado em cópia de bits e ETA I trazem economia em área e energia ao circuito de detecção de bordas de Canny.

Palavras-chave: Circuitos somadores aproximados. Processamento de imagem. Detector de bordas de Canny. Redução do consumo de energia.

ABSTRACT

This work proposes the exploration of approximate adders circuits for the implementation of power-efficient for Image Processing. The Gaussian filter is a convolution operator which is used to blur images and to remove noise. On the other hand, the Gradient of an image measures how it is changing. Both blocks can be designed in hardware using only shifts and additions/subtractions. In this work we exploit a set of approximate adders in order to implement energy-efficient filters. The tree of adders of Gaussian and Gradient filters are implemented using one Copy of bits adder, as well as an Error-Tolerant Adders - ETA. The approximate architectures are compared to the best precise implementation of the filters. As the Gaussian and Gradient blocks are part of the Canny edge detector algorithm, we have implemented the tree of adders of the filters aiming this application. In particular, an algorithm was proposed in the scope of this work in order to achieve the best adder trees for the Gaussian and Gradient filters. The main results show that for an efficient power realization of this algorithm, the best strategy consists in the implementation of the Gaussian filter with ETA I adder, and the Gradient filter with the Copy of bits adder. The approximate Gaussian and Gradient filters were applied to the fully hardware of Canny edge detector. The main results showed that the approximate Canny edge detector architectures present the best performance and precision metrics results, for most of the cases, when using both the Copy of bits and ETA I adders. For these tests a set of true images were used. The synthesis results showed that the use of the Gaussian and Gradient filters including the Copy of bits and ETA I adders has been efficient to the hardwired Canny edge detector that presented both area and energy consumption reductions.

Keywords: Approximate arithmetic operators. Image processing. Canny edge detector. Energy consumption reduction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ETA	Error Tolerant Adder
RCA	Ripple Carry Adder
FA	Full Adder
CLA	Carry Look Ahead
VLSI	Very Large Scale Integration
CMOS	Complementary Metal Oxide Semiconductor
HA	Half Adders
MUX	Multiplexador
PSNR	Peak Signal to Noise Ratio
VHDL	Very High Speed Integrated Circuits Hardware Description Language
MATLAB	Matrix Laboratory
PR	Performance Ratio Metric
FPGA	Field Programmable Gate Array
ASIC	Application Specific Integrated Circuit
BSD	Berkeley Segmentation Dadaset

SUMÁRIO

1 INTRODUÇÃO

Hoje em dia vídeos e imagens digitais estão presentes em uma grande variedade de aplicações, tais como redes sociais, entretenimento, processamento de imagens biomédicas e telemedicina. Portanto, as formas com que as imagens e os vídeos são processados, são cruciais para a qualidade das informações. Da mesma forma, a largura de banda necessária para a transmissão das informações entre os diferentes dispositivos é um fator muito importante para viabilizar esta transmissão.

Por outro lado, o aumento do uso de dispositivos portáteis que lidam com imagens digitais, como câmeras digitais, telefones inteligentes, *tablets*, resultaram em uma mudança significativa no estilo de vida das pessoas. Logo, essas aplicações exigem um processamento intensivo, como por exemplo, multimídia, rodando em dispositivos alimentados por bateria. Portanto, abordagens visando uma eficiência energética para projetos VLSI (*Very Large Scale Integration)* são necessárias.

A partir de algoritmos para o nível de concepção lógica, dá-se ênfase na implementação de aplicações que demandam um maior processamento. Neste cenário, a computação imprecisa surge como uma alternativa promissora, onde as aplicações podem tolerar alguma perda de qualidade de vídeo ou imagem e, mesmo assim, produzir resultados finais aceitáveis, em favor de um projeto digital eficiente em termos de energia.

O paradigma da computação imprecisa (Han; Orshansky, 2013) lida com a questão da eficiência energética mencionada anteriormente, considerando que as aplicações que possuem um processamento computacional intensivo, como sinais de processamento de multimídia e de imagens, exigem uma aritmética de alta precisão para proporcionar uma experiência aceitável para o usuário final. Por exemplo, se o grau de brilho em um ou dois pixels de uma imagem não é tão preciso, a maioria dos usuários não vai perceber (Haider, 2013), e assim, não há necessidade de um processamento computacional de alta precisão.

Operadores aritméticos do tipo somadores, têm sido muito explorados no contexto da computação imprecisa uma vez que são blocos de construção básica e intensamente utilizados em várias aplicações computacionais. Assim, a economia de energia tem sido apresentada para estudo de caso em algumas aplicações, enquanto os somadores aproximados são incorporados no nível de projeto de circuitos digitais CMOS (*Complementary Metal Oxide Semiconductor*) (Kahng; Kang, 2012), (Gupta; Mohapatra; Raghunathan; Roy, 2013). O trabalho apresentado em (Albicocco; Cardarilli; Nannarelli; Petricca; Re, 2012) mostra maneiras mais simples de executar adição e multiplicação de uma forma imprecisa, com o objetivo de um alto desempenho computacional com menor área e dissipação de energia, a

custa de um erro tolerável pelas aplicações. Desta forma, este trabalho explora diferentes formas de aplicação de circuitos somadores aproximados em Processamento de Imagem.

A área de Processamento de Imagem surge como um bom estudo de caso para o uso da computação imprecisa, onde a filtragem de ruído é uma ferramenta essencial, cuja principal finalidade é remover o ruído e os seus efeitos a partir da imagem original, mantendo a distorção em um nível mínimo aceitável (Gonzales; Woods; Eddins, 2009). Essas operações de filtragem, geralmente exigem um uso intensivo de operações aritméticas, aumentando assim o nível de consumo de energia.

Nesse trabalho são implementados filtros para processamento de imagem, tais como, filtro Gaussiano, filtro Gradiente e filtro detector de bordas de Canny (He, Gerstlauer; Orshansky, 2011) que agregam uma grande quantidade de operadores aritméticos, principalmente circuitos somadores. O gaussiano é um operador de convolução que é usado para desfocar as imagens e remover ruídos (Monajati; Fakhrale; Kabir, 2015). Para esse operador foram implementados os filtros de kernels 3x3 e 5x5, com desvios padrão σ=1,0 e σ=1,4, respectivamente.

Outro filtro implementado nesse trabalho é o filtro gradiente que, juntamente com o filtro gaussiano, são partes integrantes do detector de bordas de Canny, que também é o foco desse trabalho. Para ambos os filtros foram realizadas convoluções com os kernels respectivos de cada filtro, resultando na implementação de arquiteturas dedicadas de hardware compostas de árvores de deslocamentos, operações de adição e subtração. Foram utilizados nos filtros implementados, circuitos somadores aproximados, com o objetivo de melhorar tanto a área quanto o consumo de energia nos circuitos implementados dos filtros propostos.

1.1 Objetivos

Os objetivos centrais deste trabalho são o estudo e a implementação de circuitos somadores aproximados para a utilização em filtros de processamento de imagens, com o intuito da redução do consumo de energia dos filtros sem comprometer a qualidade das imagens.

Os filtros implementados são o Gaussiano, Gradiente e o detector de bordas de Canny como estudos de caso para a avaliação do desempenho dos circuitos somadores aproximados, em termos de área, consumo de energia e frequência de operação.

1.2 Principais Contribuições

A partir do estudo realizado sobre circuitos somadores aproximados e sobre filtros de processamento de imagem, como estudos de caso para a aplicação dos somadores implementados, apresentam-se as seguintes contribuições centrais do trabalho:

• Implementação de somadores aproximados estado da arte e comparações entre eles;

• Metodologia de aplicação de circuitos somadores aproximados em filtros de processamento de imagem.

1.3 Organização do Trabalho

A estrutura desse trabalho é dividida em 5 capítulos. Os conceitos introdutórios da computação imprecisa, bem como as estruturas dos somadores aproximados são apresentados no Capítulo 2. O Capítulo 3 aborda os filtros de processamento de imagem usados como estudo de caso nesse trabalho. A metodologia de aplicação dos somadores aproximados nos filtros de processamento de imagem, bem como os principais resultados obtidos são apresentados no Capítulo 4. Finalmente, o Capítulo 5 apresenta as conclusões do trabalho, bem como os trabalhos futuros.

2 COMPUTAÇÃO IMPRECISA

A computação imprecisa surge como uma ferramenta muito importante para reduzir o consumo de energia e aumentar o desempenho nas aplicações que são mais tolerantes a erros, como já foi mencionado anteriormente.

Em geral, aplicações multimídia toleram algumas imprecisões e/ou erros aritméticos. Isto porque, erros de aproximação não produzem perda significativa na qualidade subjetiva deste tipo de sinal digital. Por exemplo, a visão humana possui menor sensibilidade para componentes de alta frequência em imagens e/ou vídeos digitais. Portanto, erros inseridos nestas componentes são de difícil percepção visual (Park; Choi; Roy, 2010). Dessa forma, surge o conceito de computação imprecisa ou aritmética aproximada cuja ideia principal é utilizar técnicas que reduzam o consumo energético ou aumentem o desempenho computacional de sistemas VLSI digitais inserindo determinadas taxas de erros e/ou imprecisões nos dados computados. Estes erros, por sua vez, devem ser tolerados pela aplicação em questão.

O nosso estudo foca nos circuitos somadores, que são amplamente utilizados em projetos de multimídia, processamento de imagem, TV digital, dispositivos móveis, etc. Inicialmente são testadas técnicas de aproximações para estes operadores aritméticos (somadores), que posteriormente, são aplicados em filtros de processamento de imagens para análise do desempenho destes em relação aos somadores precisos, bem como entre eles mesmos.

2.1 Somadores Aproximados

Esta seção aborda os circuitos somadores aproximados. As técnicas utilizadas, bem como os somadores estado da arte são apresentadas nessa seção.

2.1.1 Somador Baseado em Cópia de Bits

O esquema do somador aproximado baseado na cópia direta de bits para a saída usado neste trabalho (Zhu; Goh; Zhang; Yeo; Kong, 2010) é apresentado na figura 2.1. Esse somador foi inicialmente proposto em (Gupta; Mohapatra; Raghunathan; Roy, 2013), cuja aproximação é realizada considerando que, para a tabela da verdade desse somador, a simples atribuição de uma das entradas A (ou B) como sendo o resultado da soma imprecisa, e sem levar em conta o transporte de saída (*Carry-out*), afirma-se que o resultado será correto em 4 das 8 possíveis combinações operando 1-bit, ou seja 50%.

Na figura 2.1, uma vez que o parâmetro k é determinado, os respectivos k bits menos significativos de um dos operandos de entrada são copiados diretamente para a saída do somador. Este valor k mencionado é no número de bits que farão parte do bloco impreciso do somador. Este primeiro bloco, o impreciso, é muito importante para a redução do número de operações do somador.

Por outro lado, o transporte de entrada (*Carry-in*) do próximo bloco somador, ou seja, o bloco preciso, consiste em copiar o bit mais significativo da saída do operando anterior impreciso. Este procedimento tem 75% de probabilidade de acertar a resposta correta, conforme mostrado em (Albicocco; Cardarilli; Nannarelli; Petricca; Re, 2012). A metodologia utilizada para encontrar os melhores valores de k será apresentada posteriormente no Capítulo 4.

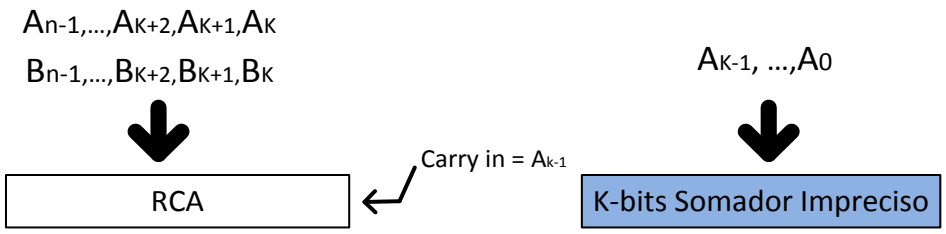


Figura 2.1 Esquema do somador baseado em Cópia de bits (Soares; Costa;Bampi,2015).

2.1.2 Somador ETA I

Os somadores ETA (*Error-Tolerant Adder*) apresentam estruturas diferentes, mas todos eles dividem o somador preciso em blocos mais curtos com o intuito de acelerar o pior caso da adição. A estrutura do somador ETA I (Zhu; Goh; Zang; Yeo; Kong, 2010) é apresentada na figura 2.2.

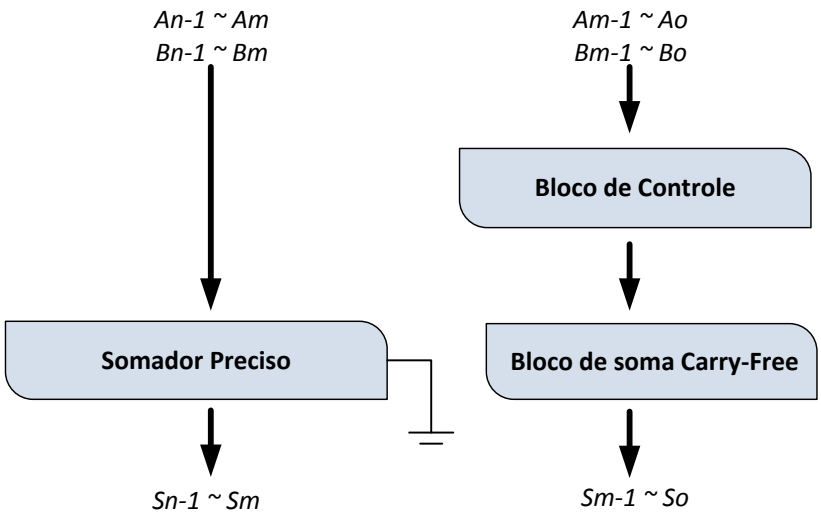


Figura 2.2 Estrutura do somador ETA I (Zhu; Goh; Zang; Yeo; Kong, 2010).

O somador ETA I é dividido em blocos aproximados e precisos. No bloco impreciso não há propagação de transporte (*carry*), e consiste no uso de meio somadores HA (*Half Adders*) e uma porta lógica *AND* para cada bit do somador. Os bits de soma são calculados a partir dos bits mais significativos em direção aos menos significativos. Cada bit é calculado por meio de somadores até que o valor "1"aparece em ambas as entradas do somador e assim, tem-se resposta igual a "1". Depois disso, todos os bits restantes são definidos com o valor lógico "1", como se pode observar no exemplo da figura 2.3.

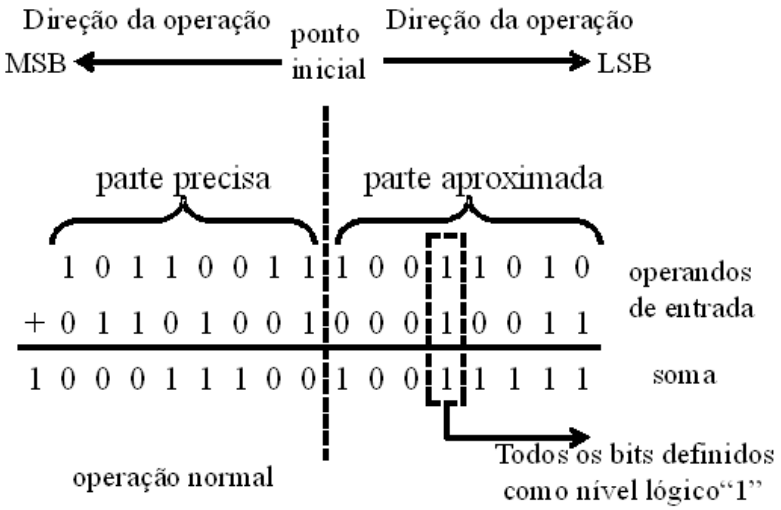


Figura 2.3 Exemplo de soma usando o esquema ETA I (Zhu; Goh; Zang; Yeo; Kong, 2010).

Como pode ser observado na figura 2.2, o bloco preciso é implementado sem qualquer técnica de aproximação para atenuar a magnitude do erro e calcula a soma considerando o *carry-in* com o valor definido como nível lógico "0".

2.1.3 Somador ETA II

O somador ETA II (Zhu; Goh; Yeo, 2009) surge para melhorar as respostas do somador ETA I, pois este apresenta um grande problema quando tem que realizar um cálculo com entrada de valores menores. Por exemplo, quando é realizada uma soma com números de 16 bits, os valores 0000000000001111 + 0000000000001111 apresentam como resultado, com a técnica ETA I, o valor igual a 0000000000001111, que é bem diferente do valor correto 0000000000011110. Logo, o somador ETA II tenta melhorar o resultado da aproximação.

A arquitetura do ETA II está representada na figura 2.4, onde para a soma de N-bits, utiliza-se a divisão desses bits por M ($M \geq 2$) blocos, onde cada bloco contém N/M bits e é composto por dois circuitos separados, sendo um gerador de *carry* (*Carry Generator* na figura

2.4) e outro gerador de soma (*Sum Generator* na figura 2.4). Um dos blocos irá gerar o *carry* de saída para o próximo bloco. O outro bloco irá gerar a soma, levando em conta o *carry* gerado pelo bloco anterior.

Deste modo, a propagação de *carry* só existe entre os dois blocos vizinhos, ao invés de se estender por toda a estrutura. Em outras palavras, o caminho de propagação do *carry* mais longo é de 2N/M bits como se pode constatar na parte marcada em cinza escuro na figura 2.4. Assim, o atraso maior do ETA II é 2/M vezes o atraso de um somador RCA com o mesmo número de bits. Desta forma, como o caminho de propagação do *carry* diminui, logo, o consumo de energia dinâmica também diminui, causado pelo menor número de comutações, de acordo com (Zhu; Goh; Yeo, 2009).

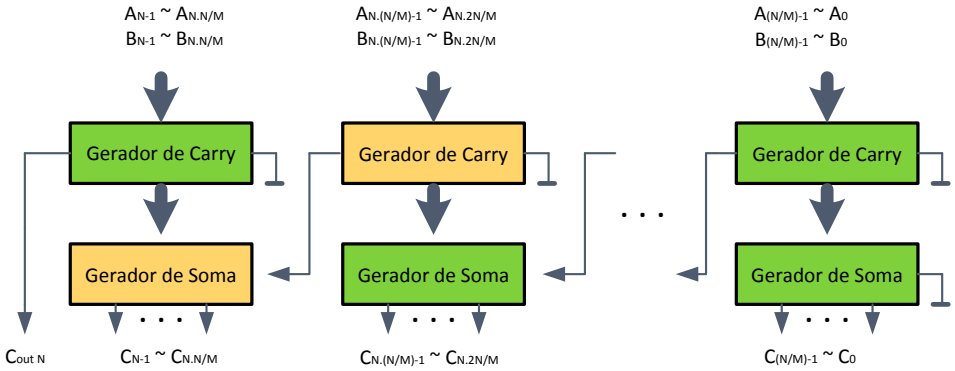


Figura 2.4 Diagrama de blocos do somador ETA II (Zhu; Goh; Yeo, 2009)

Embora a implementação em hardware do somador ETA II use circuitos precisos, ele é considerado um somador impreciso, pois todos os blocos geradores de *carry*, consideram o *carry* de entrada igual a zero. O bloco gerador de soma apresentado na figura 2.4 é um bloco RCA (*Ripple Carry Adder*), que é constituído por somadores completos (FA - *Full Adders*). O bloco gerador de *carry* da figura 2.4 é implementado com somador do tipo CLA (*Carry Look-Ahead*).

Os somadores CLA (*Carry Lookahead)* possuem o objetivo de aumentar a velocidade de propagação do *carry* em relação aos somadores RCA. Esta técnica se baseia em examinar todos os estágios de entrada do somador e, simultaneamente, produzir os *carries* apropriados para cada um destes estágios, ou seja, todos os *carries* são calculados ao mesmo tempo, paralelamente. Então estes *carries* são aplicados aos estágios posteriores onde foram gerados, para produzir o resultado final da operação de soma daqueles estágios.

2.1.3.1 Somador ETA II Modificado

Na literatura, foi proposta uma modificação no circuito ETA II (Zhu; Goh; Yeo, 2010) para aumentar a sua precisão. Neste circuito modificado, $\frac{N}{m} - (\frac{m}{2} + 1)$ blocos de geração de

carry usam carry-in igual a zero, onde *N* representa o número de bits, e *m* o número de bits em cada bloco. No exemplo da figura 2.5, cinco blocos geradores de *carry* (representados pelo somador CLA) apresentam *carry* de entrada igual a zero. Entretanto, esse somador modificado gera um problema de atraso no circuito, pois como pode ser visto na figura 2.5, o *carry* de entrada dos blocos restantes geradores de *carry* (2 blocos na figura 2.5), dependem da saída do bloco gerador de *carry* (somador CLA) do bloco anterior. Note que o último bloco de soma ($RCA_{25\text{-}31}$ no exemplo da figura 2.5) depende da cadeia de blocos geradores de *carry* anteriores (do bloco $CLA_{16\text{-}19}$ ao bloco $CLA_{27\text{-}24}$) . Desta forma, o somador ETA II modificado apresenta um maior atraso em relação ao somador ETA II, como mostrado em (Zhu; Goh; Yeo, 2009).

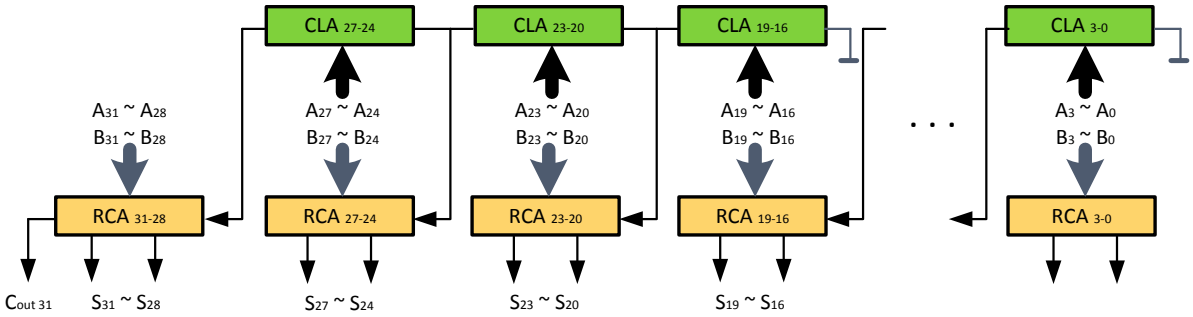


Figura 2.5 Diagrama de blocos do somador ETA II modificado (Zhu; Goh; Yeo, 2010).

2.1.4 Somador ETA IV

O somador ETA IV (Zhu; Goh; Yeo, 2010) surgiu como uma evolução do somador ETA II, visto que o mesmo não apresenta uma probabilidade de acerto muito boa para uma adição de 32 bits (cerca de 83% (Zhu; Goh; Yeo, 2010)), o que em algumas ocasiões, dependendo da aplicação, não é aceitável.

O somador ETA IV, apresentado na figura 2.6, destaca o relacionamento entre precisão e atraso de uma forma mais eficiente. A diferença neste somador, é a adição de circuitos multiplexadores (MUX 2 na figura 2.6), que são inseridos para quebrar a cadeia de transporte (*carry*) em duas fases. Dois geradores de *carry*, chamados *Carry Generator* II na figura 2.6, são criados, sendo um deles com *carry-in* igual a zero e outro idêntico com *carry-in* igual a um. Um outro gerador de *carry*, chamado de *Carry Generator* I na figura 2.6, foi criado para interligar estes blocos como se pode observar na figura 6.

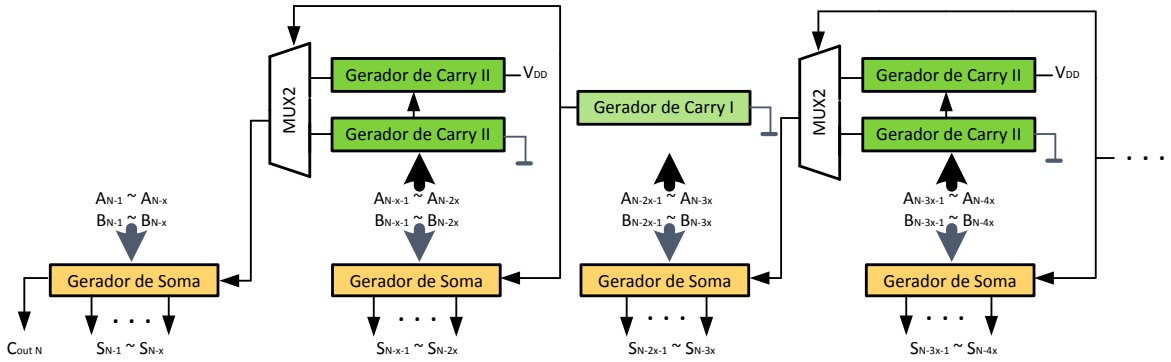


Figura 2.6 Diagrama de blocos do somador ETA IV (Zhu; Goh; Yeo, 2010).

O papel dos multiplexadores é o de selecionar o sinal de saída dos dois blocos geradores de *carry* II para a entrada do gerador de soma do bloco seguinte como se pode ver na figura 2.6.

Desta forma, a precisão na entrada do bloco somador seguinte aumenta pois ele calcula paralelamente os valores de *carry* para 0 e para 1, o que aumenta a frequência máxima de operação. Porém, este aumento de frequência de operação pode acarretar em um aumento na dissipação de potência do circuito, principalmente a potência dinâmica que é diretamente proporcional à frequência de operação (Chandrakasan; Brodersen, 1995). Entretanto, esta maior precisão é obtida ao custo do aumento do número de blocos do circuito. Assim, o somador ETA IV pode eliminar perfeitamente as desvantagens que surgiram nos ETA II e ETA II *modificado* e é considerado atualmente como o estado da arte dos somadores tolerantes a erros.

2.1.5 Comparações entre os Somadores Aproximados

Esta seção apresenta as arquiteturas dos somadores com 32 bits precisa e aproximadas, que foram implementadas. As arquiteturas foram descritas em linguagem de descrição de hardware VHDL e sintetizadas na tecnologia 45ηm, na frequência de 100 MHz, usando a ferramenta da Cadence RTL *Compiler*. Os resultados são apresentados na tabela 2.1, em termos de área, número de células, valores de potências e *Timing Slack*. Pode-se notar que o somador baseado em cópia de bits foi o que apresentou a menor área entre os somadores. Isto ocorre pelo fato desse somador ter uma estrutura mais simples, que simplesmente copia para a saída os bits menos significativos. Pelo fato de apresentar menor área, contribui para a menor potência de *leakage* apresentada pelo somador baseado em cópia de bits. O fato de apenas copiar os bits menos significativos para a saída também contribui para a redução do consumo de potência apresentado pelo somador baseado em cópia de bits. Isto pelo fato de reduzir a

atividade de chaveamento nos bits menos significativos do somador. Como apresenta menores potências de *leakage* e dinâmica, logo o somador baseado em cópia de bits apresenta a menor potência total entre os somadores apresentados.

Observando o *timing slack*, que é o caminha crítico definido como o espaço entre uma entrada e uma saída com o atraso máximo (Davare, Lwin, Kondratyev, 2004), nota-se que os somadores ETA II e ETA IV obtiveram os melhores resultados, pois tiveram maiores folgas na síntese. Isso mostra que esses somadores são mais rápidos e voltados para aplicações que necessitam maiores frequências de operação, visto que estes somadores agregam circuitos geradores de *carry* com mais eficiência do que os somadores baseados em cópia de bits e ETA I.

Como os somadores baseado em cópia de bits e ETA I foram os que apresentaram os melhores resultados, em termos de área e consumo de potência, logo estes somadores serão utilizados ao longo deste trabalho nos estudos de caso que serão apresentados.

Tabela 2.1 Análise dos somadores preciso e aproximados @ 100 MHz

	Preciso	Somador Baseado em cópia de bits	ETA I	ETA II	ETA IV
Nº de Células	**193**	**154**	**188**	**231**	**233**
Área	**736**	**592**	**714**	**866**	**873**
Potência Leakege (µm)	**12,83**	**10,35**	**12,13**	**13,47**	**13,53**
Potência Dinâmica (µm)	**953,22**	**826,91**	**943,43**	**1075,73**	**1021,17**
Potência Total (µm)	**966,06**	**837,27**	**955,56**	**1089,20**	**1034,70**
Timing Slack (ρs)	**7104**	**7725**	**7809**	**8762**	**8467**

2.2 Resumo do Capítulo

Este capítulo apresentou as principais características e estruturas internas dos somadores aproximados baseados na cópia de bits e tolerantes a erros (ETA). Na comparação entre os circuitos somadores foi mostrado que, enquanto os somadores baseados em cópia de bits e ETA I são mais indicados para projetos envolvendo redução de área e do consumo de

energia, por outro lado os somadores ETA II e ETA IV são mais indicados para projetos que necessitam maiores frequências de operação. Por apresentarem estruturas semelhantes em alguns aspectos e por apresentarem melhores resultados em termos de redução de área e consumo de energia, os somadores baseados na cópia de bits e ETA I serão utilizados no projeto de filtros de processamento de imagens.

3 FILTROS DE PROCESSAMENTO DE IMAGENS

No processamento de imagens, a filtragem de ruído é um componente essencial, cuja finalidade principal é a de remover os ruído a partir da imagem original, mantendo a distorção no mínimo possível (Gonzales; Woods; Eddins, 2009). Essas operações de filtragem exigem um uso intensivo de operações aritméticas, aumentando assim o consumo de energia no sistema.

Foi pensando nisso, que foi escolhido o processamento de imagens como alvo para aplicação e análise dos circuitos somadores aproximados estudados nesta dissertação. Os filtros em que se aplicaram esses somadores foram os filtros Gaussiano, Gradiente e o detector de bordas de Canny (*Canny Edge Detector*), onde os dois primeiros são partes integrantes do último filtro, isto é, integram o filtro detector de bordas de Canny.

3.1 Filtro Gaussiano

O filtro Gaussiano é um filtro passa-baixa não uniforme, que suaviza a imagem por cálculos de médias ponderadas. A filtragem gaussiana é utilizada para desfocar as imagens e para remover os ruídos e os detalhes. Em uma dimensão, a sua função é dada pela Equação 1, onde σ representa o desvio padrão e a distribuição assume que tem media igual a zero (isto é, centrado na linha de x=0)(Costa, 2002).

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

Note que para a obtenção dos coeficientes de kernel para o filtro gaussiano, estes são obtidos a partir da função gaussiana 2-D apresentada na Equação 2. Portanto, pode-se notar que estes coeficientes dependem do valor de σ. Neste trabalho foi usada uma distribuição com media (0,0) e com σ igual a 1, como exemplo para um filtro gaussiano 5x5 visto na figura 3.1. Este exemplo foi utilizado por estar presente na literatura (Benda; Mudry; Ijspeert, 2008).

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

$$\begin{bmatrix} A1 & A2 & A3 & A4 & A5 \\ A6 & A7 & A8 & A9 & A10 \\ A11 & A12 & A13 & A14 & A15 \\ A16 & A17 & A18 & A19 & A20 \\ A21 & A22 & A23 & A24 & A25 \end{bmatrix} * 1/273 \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 3.1 Kernel do filtro Gaussiano 5x5 com σ=1 (Benda; Mudry; Ijspeert, 2008)

A figura 3.2 mostra a implementação do hardware para obter a imagem processada pelo filtro gaussiano 5x5 com σ=1 a partir do kernel da figura 3.1.

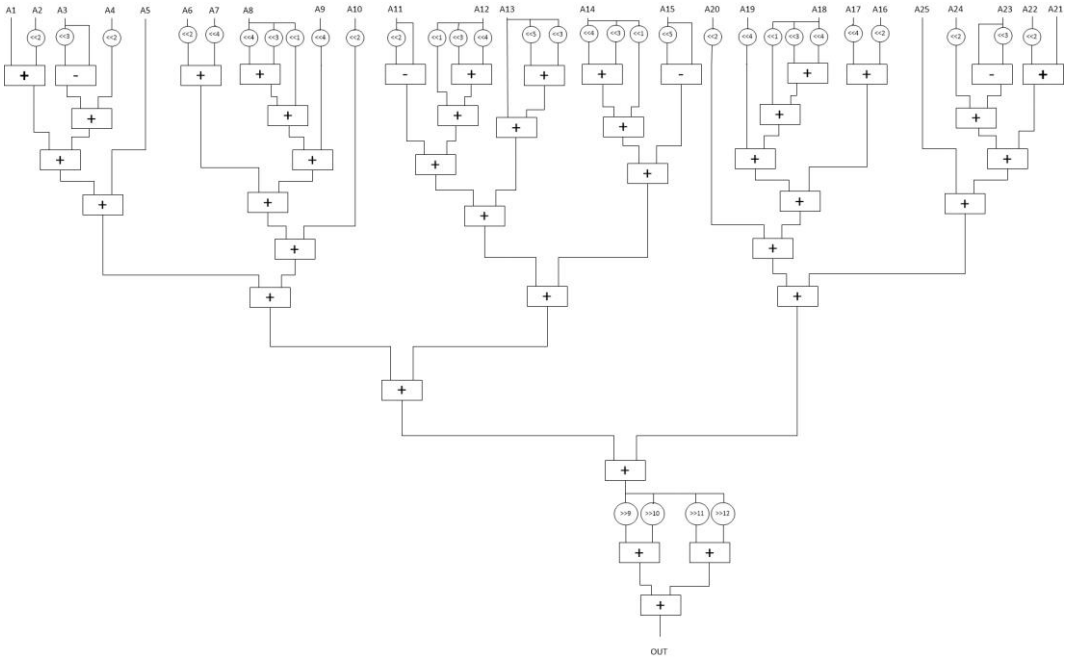


Figura 3.2 Estrutura de hardware do filtro Gaussiano 5x5 com σ=1

A figura 3.3 ilustra um exemplo de uma imagem original e a mesma imagem após passar pelo filtro gaussiano.

Figura 3.3 Imagens original e filtrada (Gaussiano) com σ=1

3.2 Filtro Gradiente

O gradiente de uma imagem é essencialmente utilizado para a detecção de bordas no processamento de imagens. A ideia consiste em primeiro realizar as derivadas horizontal e vertical da imagem, onde isto significa achar as regiões na imagem onde existem diferenças de intensidade dos elementos de imagem - pixels (*picture elements*) próximos. A equação do filtro Gradiente é mostrada na Equação (3) (Canny, 1986), onde o termo I representa as entradas do filtro e os termos H_x e H_y são as derivadas horizontais e verticais, respectivamente.

$$\bigtriangledown A = \frac{\delta I}{\delta x} + \frac{\delta I}{\delta y} = (H_x \times A) + (H_y \times A) \tag{3}$$

Pode-se obter a magnitude das entradas a partir da Equação (4) e a direção do gradiente a partir da Equação (5). Logo, a magnitude aproximada do gradiente é obtida pela Equação (6).

$$|\nabla A| = \sqrt{(H_x \times A)^2 + (H_y \times A)^2} \tag{4}$$

$$\theta(\nabla A) = arctan\left(\frac{H_x \times A}{H_y \times A}\right) \tag{5}$$

$$|\nabla A| = |H_x \times A| + |H_y \times A| \tag{6}$$

A equação 6 foi usada para a implementação do hardware do kernel de convolução, conforme está expresso na figura 3.4. Vale destacar que esse kernel foi utilizado por estar presente na literatura (Benda; Mudry; Ijspeert, 2008).

Vertical derivada

$$\begin{bmatrix} A1 & A2 & A3 \\ A4 & A5 & A6 \\ A7 & A8 & A9 \end{bmatrix} \times 1/4 \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Horizontal derivada

$$\begin{bmatrix} A1 & A2 & A3 \\ A4 & A5 & A6 \\ A7 & A8 & A9 \end{bmatrix} \times 1/4 \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figura 3.4 Kernel do filtro Gradiente 3x3 (Benda; Mudry; Ijspeert, 2008)

A figura 3.5 mostra a implementação do hardware para obter a imagem do filtro gradiente 3x3 a partir do kernel 3x3 da figura 10.

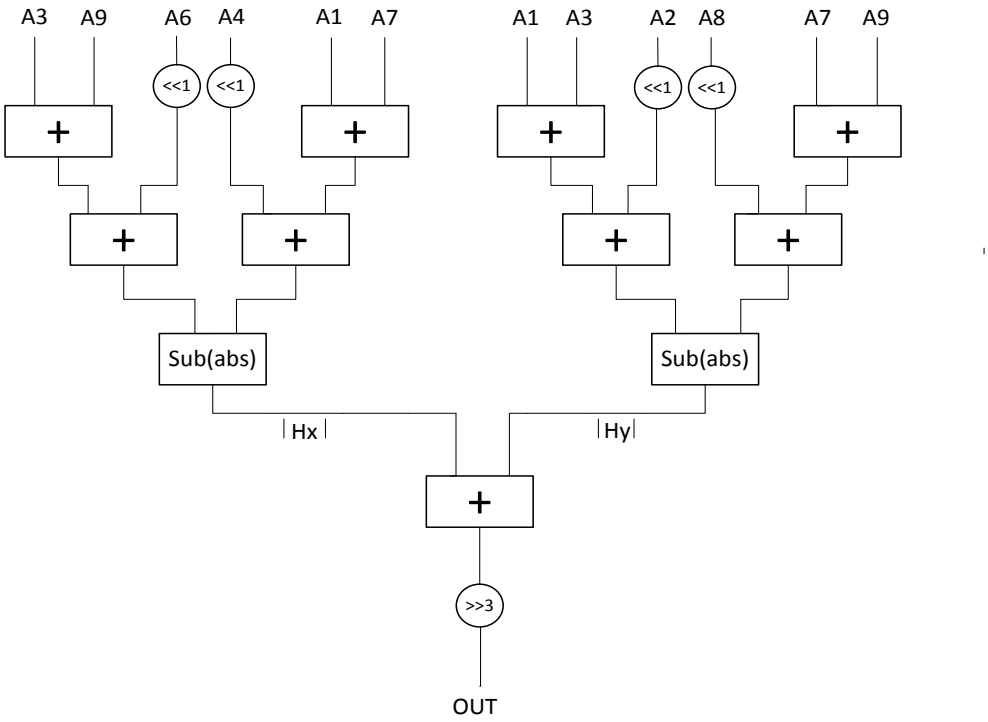


Figura 3.5 Estrutura de hardware do filtro Gradiente 3x3

A figura 3.6 ilustra um exemplo de uma imagem original e a mesma imagem após passar pelo filtro gradiente.

Figura 3.6 Imagens original e filtrada (Gradiente)

3.3 Algoritmo de Detecção de Bordas de Canny

O algoritmo de Canny foi proposto por John F. Canny em 1986 (Canny,1986)**.** No processamento de imagem, esse algoritmo representa um operador de detecção de bordas que utiliza um algoritmo de múltiplos estágios para detectar uma grande variedade de arestas em imagens (He; Gerstlauer; Orshansky, 2011), (Chandrashekar; Nataraj, 2013).

O objetivo principal do algoritmo de detecção de bordas de Canny é o de detectar bordas em uma imagem. O algoritmo é capaz de detectar e localizar bordas reais com resposta mínima quando os ambientes de ruído são considerados. O algoritmo de Canny é dividido em 5 etapas principais (Moeslund, 2009):

• Reduzir ruídos através da filtragem com um filtro gaussiano.

• Determinar os gradientes de uma imagem para destacar regiões com derivadas espaciais verticais e horizontais.

• Relacionar os gradientes das derivadas espaciais mais fortes.

• Traçar bordas válidas.

• Estabelecer o limiar de histerese (*Hysteresis thresholding*) para eliminar os rompimentos dos contornos das bordas.

Dois dos passos do algoritmo de detecção de bordas de Canny envolvem: i) aplicar o filtro gaussiano para suavizar a imagem, a fim de remover os ruídos espúrios, e ii) encontrar os gradientes das imagens e suas derivadas horizontal e vertical. Estes dois passos são alvo deste trabalho e estão marcados com círculos na figura 3.7, que mostra um diagrama de blocos dos passos do algoritmo de Canny (Oliveira; Soares; Costa; Bampi, 2015).

Neste trabalho foi utilizado um filtro gaussiano 5x5 com desvio padrão σ = 1,4 (Moeslund, 2009). Por outro lado, a fim de encontrar o gradiente da intensidade da imagem, foi usado um filtro gradiente 3x3 com base em um operador Sobel (Benda, 2008), que forneceu também as derivadas horizontal e vertical. O operador Sobel calcula o gradiente da intensidade de uma imagem em cada pixel, fornecendo uma variação de claro para escuro, e a quantidade desta variação é processada nas direções horizontais e verticais. Assim, obtém-se uma noção de como varia a luminosidade em cada ponto, podendo ser mais suaves ou mais abruptas. As variações claro-escuro mais intensas correspondem a fronteiras bem definidas entre os objetos. Assim, consegue-se fazer a detecção das bordas. Esse operador foi escolhido para ser utilizado neste trabalho por ser utilizado por vários autores no algoritmo de detecção de bordas de Canny, bem como pela simplicidade de implementação. Entretanto, os operadores de Prewitt, Roberts e laplaciano também são considerados como operadores clássicos do filtro gradiente (Oliveira; Soares; Costa; Bampi, 2016).

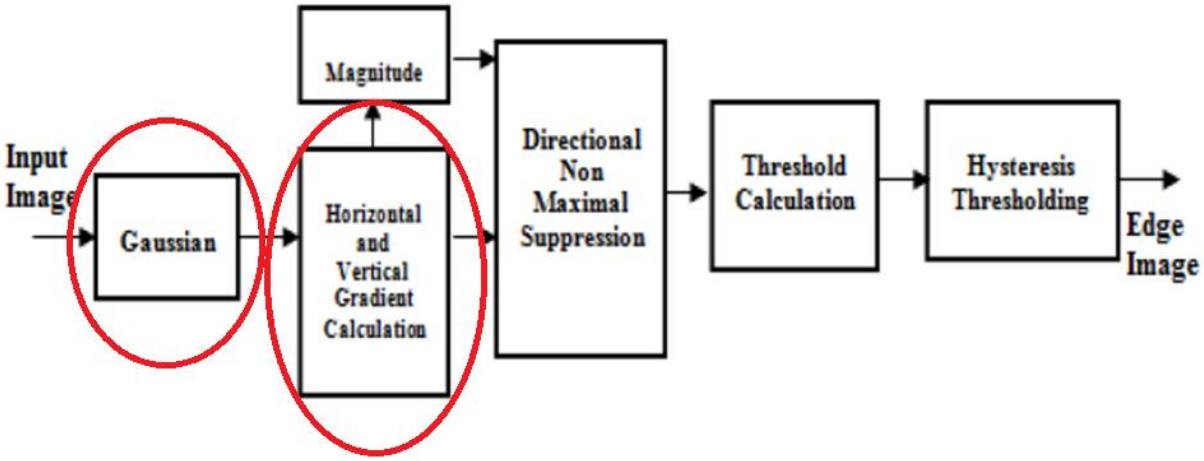


Figura 3.7 Diagrama de blocos do algoritmo de detecção de bordas de Canny (Oliveira; Soares; Costa; Bampi, 2015).

A figura 3.8 mostra um exemplo de uma imagem original, suas saídas intermediarias e a saída final do algoritmo de detecção de bordas de Canny, conforme os passos do diagrama de blocos da figura 3.7.

Figura 3.8 Etapas de processamento de uma imagem alvo usando o algoritmo de detecção de bordas de Canny

3.3.1 Aspectos de Implementação do Algoritmo de Canny

Como mencionado anteriormente, o objetivo principal do algoritmo de detecção de bordas de Canny é o de detectar as bordas e os gradientes em uma imagem. O algoritmo é capaz de detectar e localizar bordas reais, quando os ambientes de baixo ruído são considerados. O algoritmo Canny é dividido em 5 etapas: i) reduzir o ruído por filtragem através de um filtro Gaussiano; ii) determinar de gradientes de uma imagem para destacar regiões com derivadas espaciais significantes; iii) relacionar as bordas dos gradientes nas direções horizontal e vertical; iv) traçar bordas válidas ou supressão de vlores não máximos (*non-maximum suppression*); e v) aplicar um limiar de histerese (*hysteresis thresholding*) para eliminar rompimento de contornos de bordas. A figura 3.7 mostrou um diagrama de blocos das etapas do algoritmo Canny.

Ao avaliar o esforço computacional de cada etapa, pode-se notar que os filtros Gaussiano 5x5 e o Gradiente 3x3 são os blocos que requerem um esforço computacional mais intenso. Considerando como exemplo imagem na escala de cinza de 512x512 a serem filtradas pelo detector de bordas de Canny, a tabela 3.1 resume o número de operações aritméticas necessárias para a realização da filtragem dessa imagem.

Tabela 3.1. Número de operações aritméticas das etapas do detector de bordas de Canny para uma imagem de 512 x 512 na escala de cinza

	5x5 Gaussiano	3x3 Gradiente e magnitude	Determinação das direções	*Non-maximum suppression*	*Hysteresis thresholding*
Multiplicação	6,553,600	5,242,880	262,144	-	524,288
Adição	6,291,456	4,456,448	-	-	-
Comparações	-	-	1,048,576	786,432	1,048,576
Cálculo de raiz quadrada	-	262,144	-	-	-

O filtro gaussiano 5x5 é um operador de convolução que realiza 25 multiplicações e 24 adições por convolução. Uma vez que a convolução deve ser realizada 512x512 vezes, logo, chega-se ao número total de operações aritméticas superior a 12 milhões. Para cada convolução do filtro Gradiente 3x3, 9 multiplicações e 8 adições são necessárias. Uma vez que são utilizadas duas instâncias do filtro gradiente para calcular as derivadas horizontal e vertical, logo, são necessárias 18 multiplicações e 16 adições para a realização das convoluções. Na tabela 3.1 os cálculos de operações aritméticas são considerados tanto para a magnitude quanto para o gradiente. Para a magnitude, duas multiplicações e uma adição são necessárias para realizar o cálculo da raiz quadrada. Portanto, o número total de multiplicações e adições que serão necessárias tanto para o Gradiente quanto para a magnitude, são de 20 e 17, respectivamente.

Uma vez que o procedimento é repetido 512 x 512 vezes, logo, o total de operações aritméticas (isto é, multiplicações, adições e raiz quadrada) é superior a 9,5 milhões. Nos blocos restantes do detector de bordas de Canny, a soma de multiplicações representa apenas 6,67% da soma das multiplicações dos filtros Gaussiano 5x5 e Gradiente 3x3. Esta porcentagem mostra claramente que as operações de convolução exigem um processamento computacional intensivo, se comparado com as etapas restantes. Com base nisso, o objetivo foi explorar a aritmética aproximada nos filtros Gaussiano e Gradiente, como marcado na figura 3.7.

3.4 Resumo do Capítulo

Este capítulo apresentou as principais características dos filtros de processamento de imagens usados nesta dissertação como estudos de caso. Foram apresentadas as estruturas de hardware dos filtros Gaussiano e Gradiente a partir dos kernels destes filtros. Para o filtro detector de bordas de Canny foram mostrados aspectos de implementação, bem como a

justificativa para a utilização dos filtros Gaussiano e Gradiente para aproximações, visto que estes dois filtros são os que mais agregam operadores aritméticos entre os blocos do detector de bordas de Canny. O próximo capítulo aborda a aplicação de somadores imprecisos nos filtros usados como estudo de casos nesta dissertação.

4 APLICAÇÃO DE CIRCUITOS SOMADORES APROXIMADOS NOS FILTROS DE PROCESSAMENTO DE IMAGENS: ESTUDOS DE CASOS

Os estudos de caso apresentados neste capítulo estão baseados na metodologia de aplicação dos circuitos somadores aproximados apresentados anteriormente nos filtros Gaussiano, Gradiente e algoritmo de detecção de bordas de Canny. Em particular, é mostrada a metodologia de seleção dos melhores somadores aproximados a serem usados nas árvores de somadores dos filtros. Também é mostrado o relacionamento entre a qualidade de imagens e o grau de aproximação selecionado. Resultados de síntese dos filtros com o uso dos somadores aproximados também são apresentados neste capítulo. O primeiro estudo de caso aborda um estudo inicial para os filtros Gaussianos 3x3 e 5x5. Após, são mostrados resultados para filtros Gaussiano 5x5 e Gradiente 3x3, visando à aplicação no detector de bordas de Canny. Finalmente, o terceiro estudo de caso aborda resultados do detector de bordas de Canny completo, incluindo os filtros Gaussiano e Gradiente aproximados.

4.1 Aplicação de Somadores Aproximados nos Filtros Gaussianos 3x3 e 5x5

O estudo inicial consistiu na utilização de circuitos somadores aproximados para implementação dos filtros gaussianos 3x3 e 5x5 com σ=1. O objetivo do estudo é o de avaliar os níveis de aproximações e a perda de precisão no fluxo de dados aritméticos que os filtros podem tolerar para um conjunto de imagens. Nesta parte do trabalho apenas foram utilizados somadores precisos e somadores aproximados baseados em cópias de bits, com diferentes níveis de aproximação. Após, foram realizadas comparações dos filtros Gaussianos usando ambos os somadores onde se obtiveram resultados de consumo de energia e área para cada um dos filtros. Este estudo inicial serviu de base para avaliar o potencial de impacto da utilização dos somadores aproximados em filtros de processamento de imagens.

4.1.1 Metodologia de Obtenção dos Valores de k

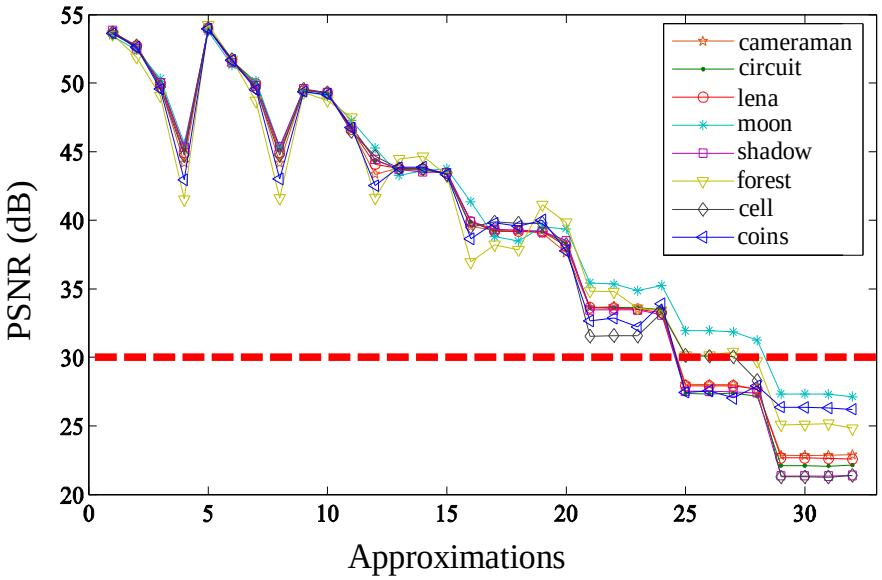
Como mostrado na seção 2.2.1 (Figura 2.1), no somador aproximado, baseado em cópia de bits, o bloco aproximado é composto pelos k bits menos significativos que são copiados diretamente das entradas para as saídas. Os valores mais apropriados de k são obtidos em simulações no software MATLAB, a partir das árvores de somadores dos dois filtros (3x3 e 5x5). A metodologia para a aproximação consiste nas seguintes etapas:

• Os somadores são separados em grupos, levando-se em conta o número de deslocamentos das entradas das árvores de somadores. Este agrupamento é realizado de

acordo com os somadores que possuem menos deslocamentos e os que possuem mais deslocamentos.

• Define-se a faixa em que vai se variar o k para cada grupo de somadores, levando-se em conta os seus deslocamentos nas suas entradas. O grupo em que os somadores possuem mais deslocamentos, irá variar o valor de k numa faixa maior. Por outro lado, para os somadores que apresentam menos deslocamentos, ou mesmo nenhum deslocamento, varia-se o k numa faixa menor.

• Os resultados dos melhores valores de k para os grupos de somadores são gerados no MATLAB para oito imagens definidas como padrão de teste (*cameraman*, *circuit*, *lena*, *moon*, *shadow*, *forest*, *cell*, *coins*), onde a relação sinal ruído de pico (PSNR - *Peak Signal-to-Noise Ratio*) é usada como métrica, como pode ser visto nas figuras 4.1, 4.2 e 4.3.



. Figura 4.1 PSNR para 8 imagens do filtro 3x3

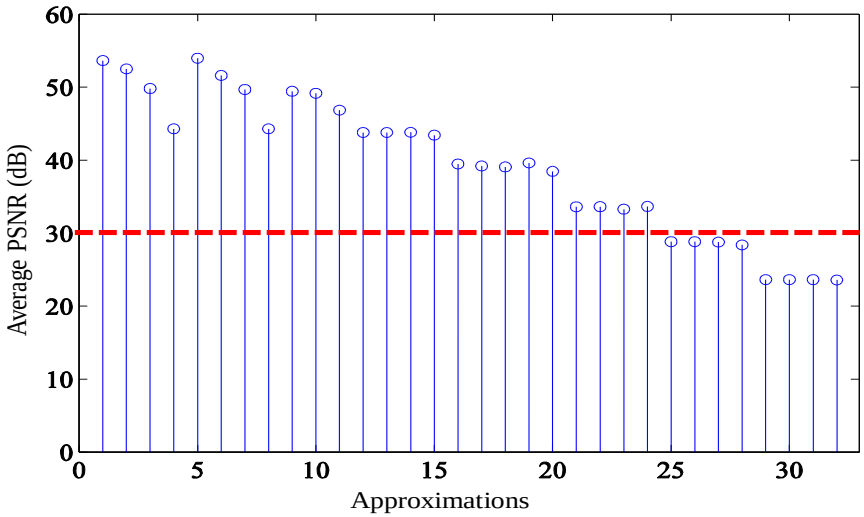


Figura 4.2 Média PSNR para filtro 3x3

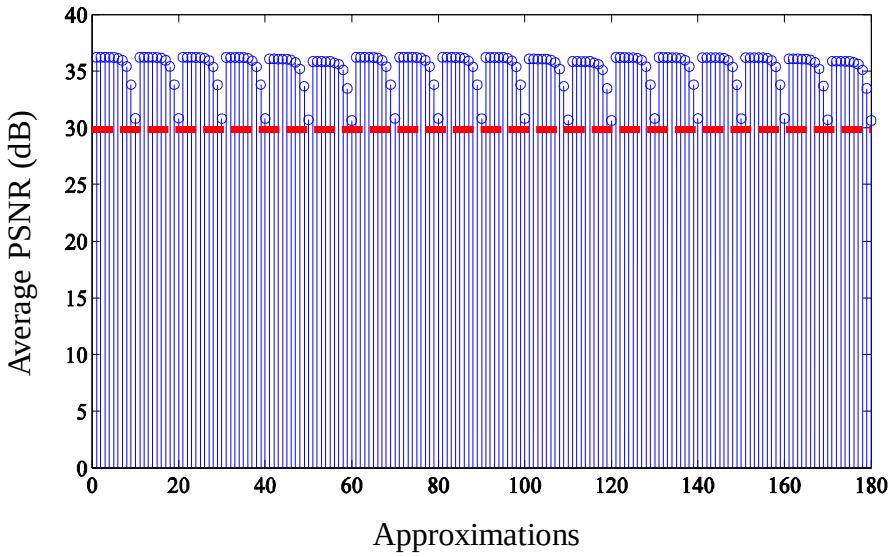


Figura 4.3 Média PSNR para filtro 5x5

Observa-se nas linhas tracejadas das figuras 4.1, 4.2 e 4.3, que o limite estabelecido para a qualidade das imagens foi de 30dB. Esse limite foi utilizado, pois de acordo com a literatura (Xu; Varadarajan; Chakrabarti, 2014) e (Sangeetha; Deepa, 2016), esse valor representa o mínimo de PSNR para que se tenha uma imagem com nível aceitável de qualidade. Com esse limite de qualidade das imagens de 30dB, foram estabelecidos os melhores valores de k para os grupos de somadores aproximados, conforme mostram as árvores de somadores das figuras 4.4 e 4.5 para os filtros Gaussiano 3x3 e 5x5, respectivamente.

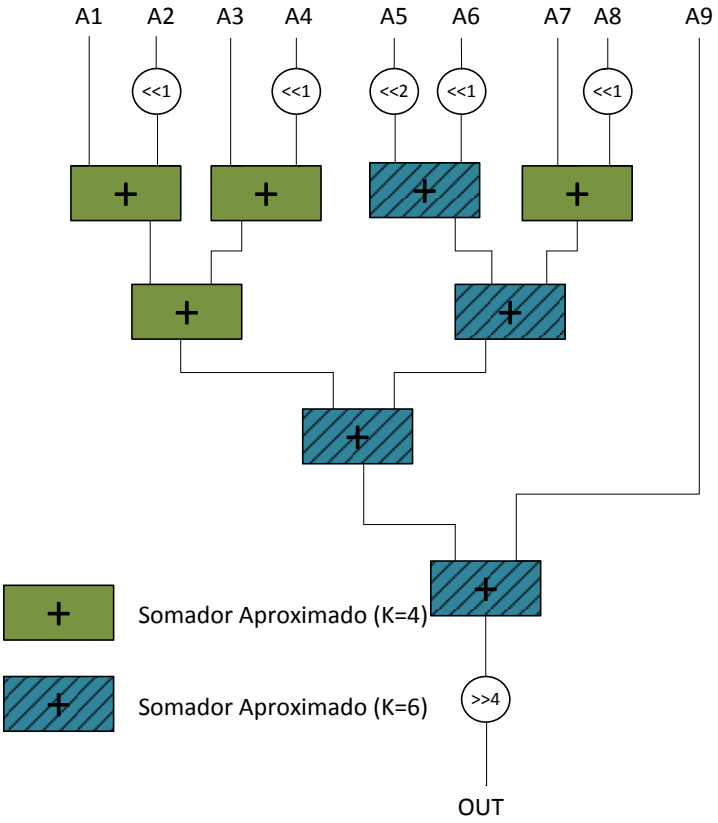


Figura 4.4. Filtro Gaussiano aproximado 3x3

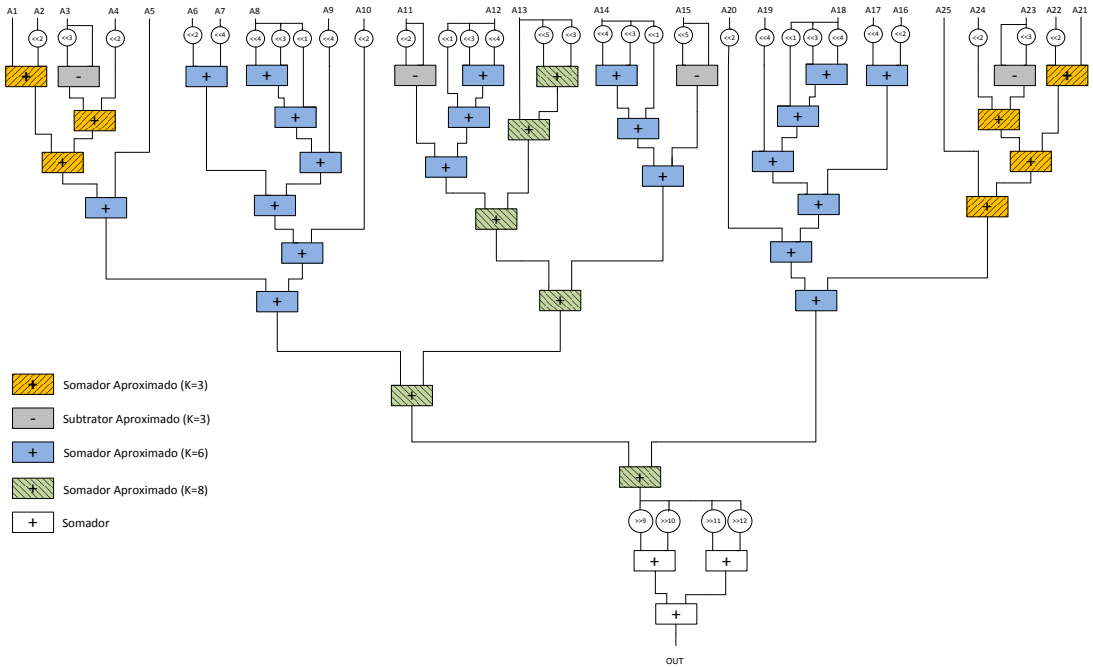


Figura 4.5 Filtro Gaussiano aproximado 5x5

4.1.2 Resultados de Síntese

As arquiteturas dos filtros 3x3 e 5x5 precisos e aproximados, das figuras 4.4 e 4.5, foram descritas em linguagem de descrição de hardware VHDL (*Very High Speed Integrated Circuits Hardware Description Language*). As estruturas foram validadas no ambiente Quartus II da ferramenta Altera (He; Gerstlauer; Orshansky, 2011) e sintetizadas na ferramenta *Encounter RTL Compiler* da Cadence na tecnologia CMOS de 45ηm (Xu, Varadarajan, Chakrabarti, 2014). O consumo de energia e área para os filtros gaussianos 3x3 (com k=4 e k=6) e 5x5 (com k=3, k=6 e k=8) são apresentados na figura 4.6. As frequências utilizadas para a realização das sínteses vão desde 100MHz até as frequências máximas alcançadas pelos filtros gaussianos 3x3 e 5x5 precisos, que foram de 700MHz e 350MHz, respectivamente.

Na figura 4.4(a), que representa o filtro gaussiano 3x3, mostra-se que a técnica de aproximação produz uma redução de área máxima de até 45% para um PSNR médio de 33,7 dB. Em um outro experimento, realizado para o filtro 3x3, foram realizadas sínteses para um nível menor de aproximações com valores k=5 e k=3 com um PSNR médio de 39,6dB, e para k=2 e k=3 com um PSNR médio de 49,7dB. Os valores de redução de área foram de 34% e 30%, respectivamente. Observa-se então o relacionamento entre o nível de aproximação com a qualidade das imagens. Quanto menor a qualidade da imagem requerida (mais próximo à 30 dB), torna-se possível utilizar uma maior aproximação, o que acarreta em uma menor área a ser utilizada (menos somadores).

Em termos de redução de energia os valores máximos atingidos foram de 34% para um PSNR médio de 39,6dB, 20% para um PSNR médio de 49,7dB e 42% para um PSNR médio de 33,7dB, conforme se pode observar na figura 4.6(b). Esses resultados comprovam que, quanto menor o nível da qualidade da imagem requerida, tem-se uma maior redução de energia, visto que menos circuitos somadores são utilizados (menos área).

Na figura 4.6(b), que representa o filtro gaussiano 5x5, foram mostrados dois níveis de aproximações, um com os valores de k=3, k=6 e k=8, com um PSNR médio de 35,1dB e outro com os valores de k=3, k=6 e k=10, com um PSNR médio de 30,6dB. A redução de área máxima alcançada foi de 28% para o PSNR médio de 30,6dB e de 26% para o PSNR médio de 35,1dB. Neste caso, como o PSNR é próximo entre os dois testes realizados, logo os valores de redução de área são praticamente os mesmos. Entretanto, com um PSNR menor (30,6dB) a redução de área é maior, visto que a aproximação é maior (menos somadores utilizados).

Em relação à redução do consumo de energia os resultados obtidos como melhores valores apresentaram para o PSNR médio de 30,6dB o valor de redução de 32% e para o PSNR de 35,1dB o valor de redução de 29%. Esses resultados só comprovam o que já tinha sido mostrado anteriormente, ou seja, menor qualidade da imagem, maior aproximação, menos somadores (menor área), logo menor consumo de energia.

Outro efeito importante observado é que ambos os filtros aproximados apresentaram aumento na frequência de funcionamento. Embora o filtro preciso 3x3 tenha obtido um valor máximo de 700MHz, o filtro aproximado 3x3 alcançou um valor de 840MHz para um PSNR médio de 33,7dB. Para os filtros 5x5 foram obtidos valores máximos de 360MHz e 380MHz para os circuitos preciso e aproximado, respectivamente, onde o filtro aproximado foi implementado com um PSNR médio de 35,1dB. Em outras palavras, os filtros aproximados 3x3 e 5x5 apresentaram um aumento na frequência de operação nos valores de 5,26% e 16,67%, respectivamente. Os picos representados nos gráficos da figura 4.6 ocorrem devido ao fato da ferramenta de síntese estar atingindo o máximo de frequência para os circuitos alvo. Neste caso, a ferramenta de síntese tenta otimizar ainda mais os circuitos, tentando atingir frequências ainda maiores, mas chega ao seu limite máximo.

Os resultados obtidos deste estudo inicial de aplicação de circuitos somadores aproximados nos filtros Gaussianos 3x3 e 5x5 foram publicados em (Oliveira; Soares; Costa; Bampi, 2015).

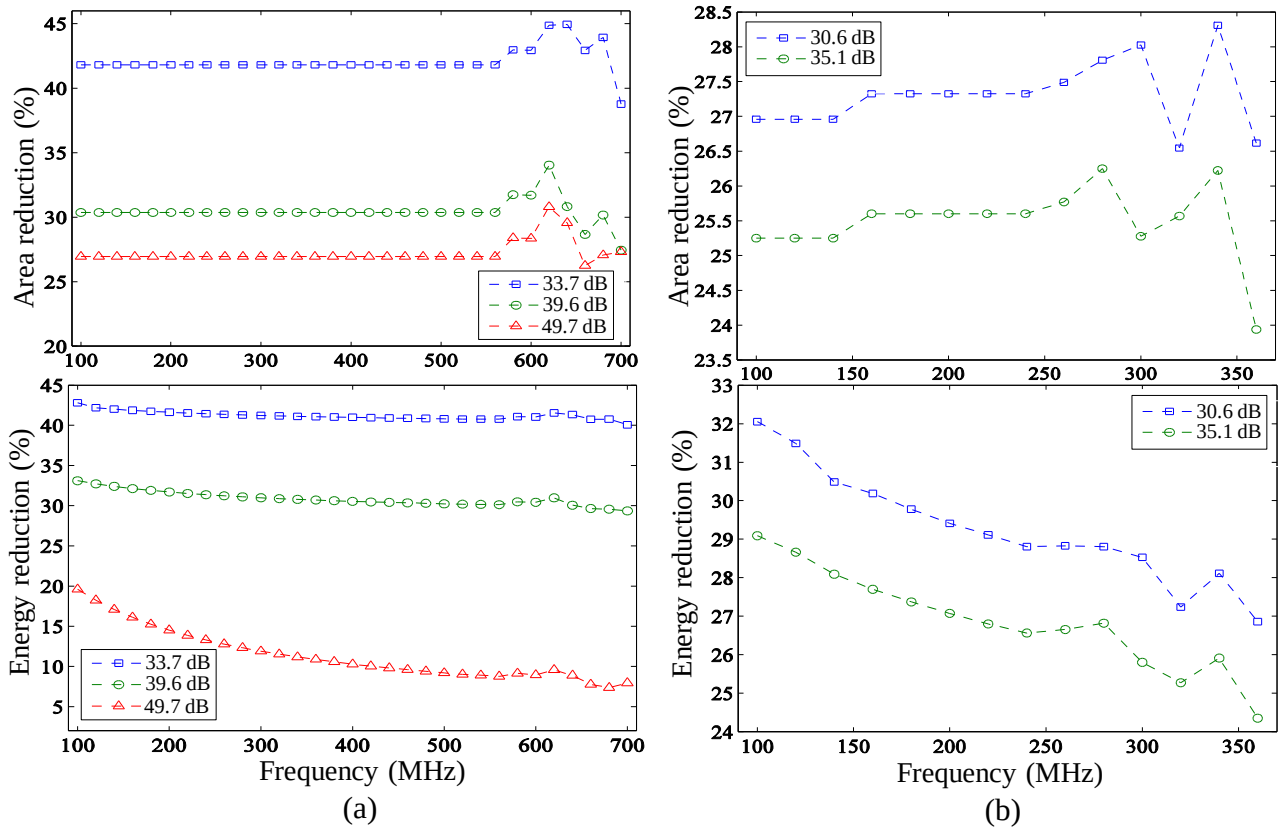


Figura 4.6 Resultados de redução de área e energia para os filtros aproximados: (a) Filtro Gaussiano 3x3 e (b) Filtro Gaussiano 5x5

A figura 4.7 mostra as imagens de referência da Lena após serem filtradas pelos filtros gaussianos 3x3 e 5x5 precisos e aproximados. Os resultados precisos são mostrados em (a) e (c) e os aproximados são apresentados em (b) e (d). Para a imagem filtrada aproximada 3x3 em (b) o PSNR utilizado para a aproximação foi de 33,3 dB, enquanto que para imagem aproximada 5x5 em (d), o PSNR foi de 36,6 dB, pois foram os valores mais próximos do valor mínimo de 30 dB adotado na análise dos valores de k.

(a) (b)

(c) (d)

Figura 4.7 (a) Filtro Gaussiano preciso 3x3 (b) Filtro gaussiano aproximado 3x3 (c) Filtro Gaussiano preciso 5x5 (d) Filtro Gaussiano aproximado 5x5

4.2 Aplicação de Somadores Aproximados em Filtros Gaussiano e Gradiente

Como mostrado anteriormente (Figura 3.7), o filtro detector de bordas de Canny é composto por 5 etapas, onde os blocos dos filtros Gaussiano e Gradiente agregam o maior número de operadores aritméticos. Desta forma, os filtros implementados nesta parte do trabalho utilizam os somadores aproximados para implementar o filtro Gaussiano 5x5 com σ=1.4 e o filtro Gradiente 3x3, com vistas à implementação do circuito completo do detector de bordas de Canny aproximado.

4.2.1 Obtenção das Árvores de Somadores

Nesta parte do trabalho, além do somador aproximado baseado na cópia de bits, também o somador ETA I foi utilizado em diferentes níveis de aproximação para a

verificação do impacto da redução do consumo de energia dos filtros aproximados com o uso destes somadores. A metodologia empregada foi muito semelhante a da seção 4.1.1.

Para o somador aproximado baseado no somador com cópia de bits, a metodologia de obtenção dos melhores valores de k para os filtros gaussiano e gradiente foi a mesma mostrada anteriormente na seção 4.1.1. As imagens utilizadas como referência também foram as mesmas da seção 4.1. A figura 4.6 mostra a análise PSNR para os filtros Gaussiano e Gradiente. Na figura 4.8 o eixo horizontal representa o número de combinações utilizado nos testes. Por exemplo, no gráfico do gradiente, o número de 24 combinações representa que o primeiro grupo de somadores teve o k variando de 1 até 4 e o segundo grupo de somadores teve o k variando de 1 até 6, totalizando assim 24 combinações para poder chegar nos valores otimos para cada K, respeitando o valor mínimo pré-determinado do PSNR de 30dB.

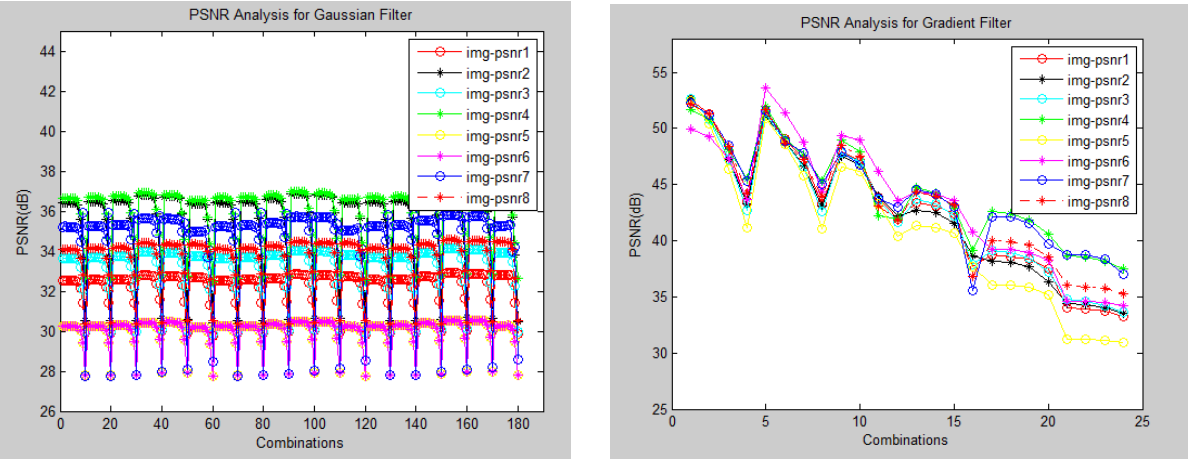


Figura 4.8 PSNR para 8 imagens. Filtros Gaussiano 5x5 com σ=1.4 e Gradiente 3x3

A partir dos resultados do PSNR, obtidos a partir de simulações no MATLAB, foram implementadas as árvores com os grupos de somadores aproximados com os melhores valores de K para os filtros gaussianos 5x5 e gradiente 3x3, como podem ser vistas nas figuras 4.9 e 4.10 respectivamente.

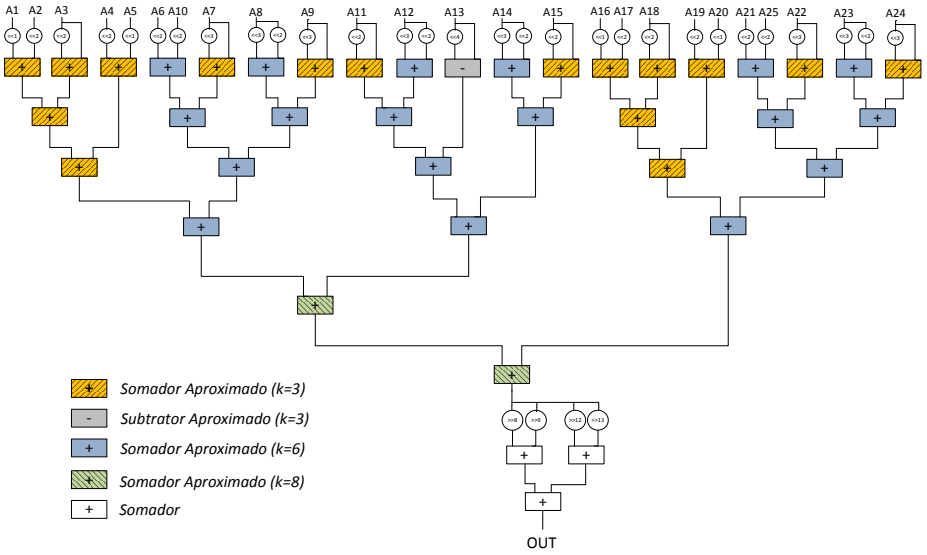


Figura 4.9 Filtro Gaussiano 5x5 com σ=1.4 com os grupos de somadores aproximados

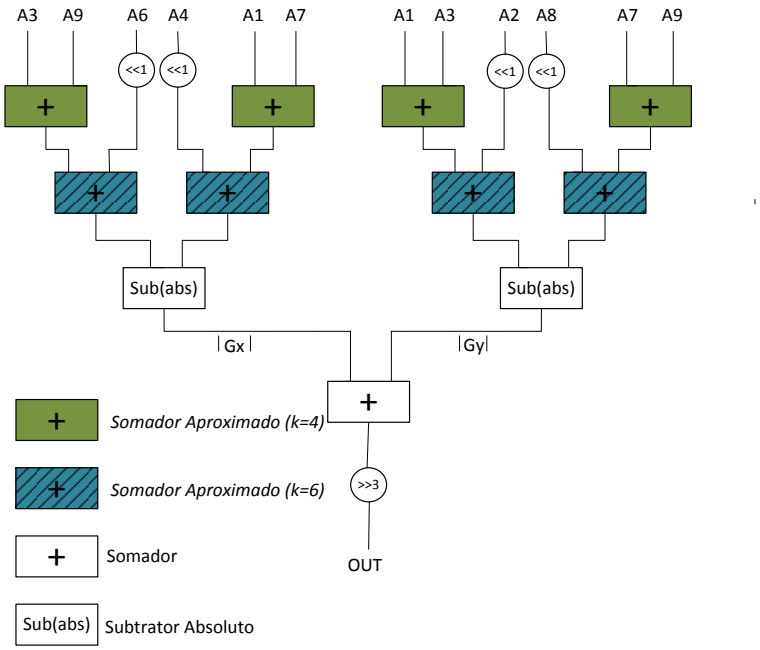


Figura 4.10 Filtro Gradiente 3x3 com os grupos de somadores aproximados

4.2.2 Resultados de Síntese

Após implementação das árvores das arquiteturas dos filtros aproximados e precisos, estes foram descritos em VHDL, e após as estruturas foram validadas no ambiente da ferramenta Quartus II da Altera (He; Gerstlauer; Orshansky, 2011). A validação foi realizada no sentido de verificar o funcionamento correto dos circuitos. Para tal, alguns valores aleatórios foram colocados nas entradas dos filtros e os resultados foram verificados no

ambiente de simulação da ferramenta Quartus II (*Simulation Waveform Editor*). Após, os circuitos foram sintetizados usando a ferramenta *Encounter RTL Compiler da Cadence* com uma tecnologia CMOS de 45ηm (Oliveira; Soares; Costa; Bampi, 2016) para as frequências de 100MHz, 200MHz e 300MHz. Vale destacar que a frequência de 300MHz ficou próxima da máxima frequência de síntese dos circuitos, tornando-se assim o limite superior de funcionamento correto dos circuitos. As duas outras frequências foram escolhidas para uma melhor análise dos resultados. As tabelas 4.1 e 4.2 apresentam os resultados dos filtros em relação à área, número de células, potência estática, potência dinâmica e potência total para os filtros com somadores utilizados neste estudo.

As tabelas abaixo mostram os resultados dos filtros Gaussiano e Gradiente, onde os somadores baseados em cópia de bits e ETA I foram utilizados. Estes dois somadores foram utilizados, pois apresentam estruturas parecidas, o que proporciona uma comparação justa. Como pode ser observado, ambos os filtros Gradiente e Gaussiano apresentaram melhores resultados usando o somador baseado em cópia de bits. Entretanto, o uso deste somador somente se mostrou eficiente no filtro Gradiente em termos de redução do consumo de energia, pois este filtro apresenta um menor número de somadores. Entretanto, o filtro Gaussiano, que apresenta um maior número de circuitos somadores, se mostrou mais eficiente, em termos de redução do consumo de energia, com o somador ETA I. Isso mostra uma tendência de que os somadores tolerantes a erro sejam mais eficientes em termos de redução do consumo de energia quando um maior número de circuitos somadores é utilizado.

Em uma análise mais criteriosa, chegou-se à conclusão que o somador ETA I é mais indicado para simplificações pela ferramenta do que o somador baseado em cópia de bits. Pelo fato de que o primeiro somador completo da parte precisa pode ser trocado por um meio somador sem problema nenhum, visto que o *carry* de entrada desse somador é sempre “0”, ao contrário do somador baseado na cópia de bits, que espera sempre o valor do bit mais significativo da parte imprecisa.

Analisando os resultados dos filtros Gradiente e Gaussiano observa-se que, no primeiro filtro (Gradiente) os ganhos em potência estão relacionados ao uso do somador baseado na cópia de bits. Isto porquê este filtro possui os menores valores de k, bem como menos somadores, o que não possibilita maiores simplificações para o somador ETA I. Por outro lado, no filtro Gaussiano, os ganhos estão relacionados ao uso do somador ETA I, pois o filtro agrega maiores valores de k, bem como mais somadores, oportunizando assim, uma maior simplificação pela ferramenta de síntese.

Os principais resultados deste estudo estão publicados em (Oliveira; Soares; Costa; Bampi, 2016).

Tabela 4.1 Resultados de síntese dos filtros Gaussianos 5x5 σ=1.4 com somadores preciso e aproximados

Filtro	Número de células	Area (μm^2)	Potência (ηW)		
			Estática	Dinâmica	Total
100MHz					
Preciso	870	5782	43449	1418712	1462161
Basedo em cópia de bits [7]	692	4509	34704	1091957	1126662
ETA I [8]	1022	4786	45284	1030231	1075515
200MHz					
Preciso	858	5765	43379	1946206	1989586
Basedo em cópia de bits [7]	680	4492	34639	1543507	1578146
ETA I [8]	1010	4769	45214	1454891	1500106
300MHz					
Preciso	880	5870	43841	2520643	2564484
Basedo em cópia de bits [7]	722	4609	35298	2036016	2071315
ETA I [8]	1017	4782	45383	1870627	1916011

Tabela 4.2 Resultados de síntese dos filtros Gradiente com somadores preciso e aproximados

Filtro	Número de células	Área (μm^2)	Potência (ηW)		
			Estática	Dinâmica	Total
100MHz					
Preciso	395	2090	15805	464240	480045
Baseado em cópia de bits [7]	321	1571	12089	317190	329279
ETA I [8]	581	2184	17742	390257	407999
200MHz					
Preciso	389	2081	15801	646848	662650
Baseado em cópia de bits [7]	315	1562	12076	456591	468668
ETA I [8]	575	2190	18258	573997	592255
300MHz					
Preciso	389	2081	15849	830765	846615
Basedo em cópia de bits [7]	315	1562	12107	596154	608261
ETA I [8]	571	2187	18393	757155	775549

4.3 Aplicação dos Somadores Aproximados no Filtro Detector de Borda de Canny Completo

O terceiro estudo de caso dessa dissertação envolve a implementação completa do detector de borda de Canny, visando avaliar os resultados com a aplicação dos filtros Gaussiano e Gradiente aproximados.

No processamento de imagens, o filtro Detector de Bordas de Canny é um operador que utiliza um algoritmo de múltiplos estágios para detectar uma vasta gama de bordas em imagens (Canny, 1986) (Zhu; Goh; Zhang; Yeo; Kong, 2010). É um dos algoritmos de detecção de borda mais utilizado devido ao seu ótimo desempenho na detecção de bordas, mas em comparação com outros algoritmos de detecção de borda, tais como o operador de Sobel (Blenda, 2008), apresenta um maior processamento computacional.

As duas etapas que apresentam maior processamento computacional do algoritmo detector de bordas de Canny são aplicar o filtro Gaussiano para suavizar as imagens e remover os ruídos, e encontrar os gradientes das imagens, como mostrado anteriormente na Tabela 1. Neste trabalho, a fim de lidar com o elevado processamento computacional e para alcançar uma eficiência energética na detecção de bordas, propõe-se a implementação em hardware do detector de bordas de Canny, com somadores aproximados para as arquiteturas dos filtros Gaussiano 5x5 e Gradiente 3x3, mostrados na seção anterior. Buscam-se aproximações sub-óptimas, uma vez que a busca de valores ótimos envolve um processamento exaustivo que tem uma grande probabilidade de exigir um tempo muito grande de simulação. Por isso, propõe-se no âmbito desse trabalho, em parceria com o doutorando Leonardo Bandeira Soares, da Universidade Federal do Rio Grande do Sul (UFRGS), um algoritmo para selecionar grupos de somadores que buscam formar sub-árvores, levando em conta a expectativa da magnitude da entrada. O algoritmo proposto transforma a busca por aproximações nos filtros Gaussiano e Gradiente em uma tarefa mais viável. Foi usado um filtro Gaussiano 5x5 com desvio padrão σ=1.4(Moeslund, 2009), e um filtro Gradiente 3x3 com base em um operador de Sobel (Blenda, 2008), em que as derivadas horizontal e vertical da imagem são calculadas. A heurística explora o uso de somadores aproximados baseado em cópia de bits e ETA I. Esses somadores apresentam semelhanças na estratégia de aproximação dos bits menos significativos. Baseado nisto fica viável controlar a magnitude da aproximação das sub-árvores de somadores que compõem a árvore principal.

O alvo principal do estudo é avaliar as respostas da detecção de borda e a eficiência energética utilizando as aproximações propostas. Foi proposta ainda uma arquitetura de um acelerador de hardware para todo o algoritmo Detector de Bordas de Canny. A arquitetura

proposta filtra a imagem na direção horizontal (fila por fila) proporcionando a reutilização de dados e produzindo uma saída de um pixel da imagem por ciclo de relógio, com uma latência de 8 ciclos de relógio. A maioria dos trabalhos relacionados, como em (Xu; Varadarajan; Chakrabarti, 2014)-(Sangeetha; Deepa, 2016), propõem arquiteturas para o detector de bordas de Canny, que são sintetizadas para FPGA. Nesse trabalho, foi projetada uma arquitetura ASIC (*Application Specific Integrated Circuit*) completa. Para conhecimento geral, nenhum trabalho anterior propôs uma arquitetura para o detector de bordas Canny empregando computação aproximada para implementação em ASIC. A importância de projetos em ASIC se deve à possibilidade da avaliação da dissipação de potência de maneira mais precisa. As métricas de qualidade de saída para avaliar o desempenho foram PR (*Performance Ratio*) e a medida F que indica quão "suficientemente bom" os resultados são em termos de detecção de borda, quando considerando a versão aproximada da arquitetura em comparação as versão precisa, ambas com o mesma estrutura de hardware.

4.3.1Algoritmo para aproximação dos filtros Gaussiano e Gradiente

O estudo inicial para a obtenção dos melhores valores de k nos filtros Gaussiano e Gradiente mostrou a necessidade do uso de uma heurística para o projeto das árvores de somadores, pois para se chegar aos resultados ótimos para cada situação leva-se um longo tempo de processamento. Por exemplo, para um conjunto de 10 somadores, todos eles serão simulados com um parâmetro k igual a 1. Depois disso, todos os somadores são testados para o k igual a 2, 3, 4, e assim por diante. Esta estratégia não representa uma resposta ideal com relação à eficiência energética e a qualidade da informação de saída. Portanto a busca exaustiva daria um melhor resultado. Porém, essa alternativa pode resultar num tempo de processamento computacional inviável. Por exemplo, considerando um filtro contendo 16 somadores de 16 bits para ser aproximado, e se tendo interesse na avaliação do parâmetro k, varia-se este valor de k desde 1 até 10 bits menos significativos para avaliar as aproximações. Baseado neste cenário, o número de diferentes configurações possíveis neste filtro seria de 10^{16}. Agora, considera-se que, para executar uma convolução leve 1ms (isto com base no tempo de processamento do MATLAB). Assim, quando se utiliza apenas 1 núcleo do processador, o tempo para processar uma imagem de 512x512 pixels de 8bits nesta configuração é 262,1ms. Como se têm 10^{16} configurações que levariam $2{,}62 \times 10^{15}$ segundos para serem testadas para uma imagem de 512x512 pixels, logo, pode-se concluir que, depois de $8{,}3 \times 10^{7}$ anos, seria impossível decidir qual configuração de somadores aproximados apresenta o melhor relacionamento entre eficiência energética e qualidade de saída.

Consequentemente, para um circuito simples de 16 somadores a simulação seria inviável. Além disso, este cenário considera apenas uma imagem. Entretanto, o cenário ideal é usar mais imagens para validar melhor a abordagem de aproximação. Mesmo considerando arquiteturas com muitos núcleos com 48 processadores e com velocidade alta de processamento, este tipo de simulação continuaria inviável. Em geral, o tempo de simulação de um conjunto de n somadores é dada pela equação 7.

$$stime = k^n \cdot comptime \tag{7}$$

Na equação 7, o parâmetro k se refere ao intervalo de valores que se deseja testar para o parâmetro k e *comptime* é o tempo computacional levado para realizar o conjunto de *n* somadores por cada configuração que está sendo testada. Com base nesta análise, foi prospota no âmbito deste trabalho de dissertação uma nova heurística para somadores aproximados, implementada em MATLAB, como mostrado no Algoritmo 1.

Algoritmo 1 Heurística para aproximações

Input: *S, R* – lists containing the initial weights *w* for each input operands in *level* 0
T – a list of threshold values *t* /
Q – a tree of adders to be approximated

Output: *P* – a tree of adders regarding the group of each adder
W – a tree of computed weights for each adder

1: *level 0 grouping*:
2: (*W, P*) := *Q*;
3: **for each** $r_j \in R, s_j \in S, p_{0,j} \in P, w_{0,j} \in W$ **do**
4: **for** *i* **in** 1 **to** *ngroups* **do**
5: **if** ($r_j \geq t_i \wedge s_j \geq t_i$) **then**
6: $p_{0,j} := i$;
7: $w_{0,j} := r_j + s_j$;
8: **end if**
9: **end for**
10: **end for**
11: *remaining levels grouping:*
12: i := 1;

```
13: for each p_{i,j} ∈ P, w_{i,j} ∈ W do
14: currentValue := child_left(w_{i,j}) + child_right(w_{i,j});
15: for k in 1 to ngroups do
16:         if (currentValue ≥ t_k ) then
17:                 p_{i,j} := k;
18:                 w_{i,j} := currentValue;
19:         else
20:                 p_{i,j} := max(children(p_{i,j}));
21:                 w_{i,j} := currentValue;
22:         end if
23: end for
24: end for
```

A heurística é aplicada ao hardware para implementar os filtros Gaussiano e Gradiente. Nas figuras 4.11 e 4.12 são apresentadas implementações em hardware, tanto para o filtro Gaussiano 5x5 como para o filtro Gradiente 3x3 desenvolvidos neste trabalho. Na figura 4.11, o filtro Gaussiano 5x5 apresenta 40 somadores e 29 deslocamentos, que são utilizados para implementar as multiplicações e somas do operador de convolução. Na figura 4.12, o filtro Gradiente 3x3 realiza multiplicações e somas para realizar as derivadas vertical e horizontal deste filtro. Pode-se observar que a arquitetura implementada apresenta 10 somadores e 4 deslocamentos. Com base nas arvores de somadores mostrados nas figuras 4.11 e 4.12, a heurística proposta se baseia na magnitude esperada dos operandos de entrada de cada somador. Nessa heurística, a grandeza da entrada esperada é classificada para cada somador, uma vez que que o operador da convolução dos pixels de entrada é multiplicado por uma constante.

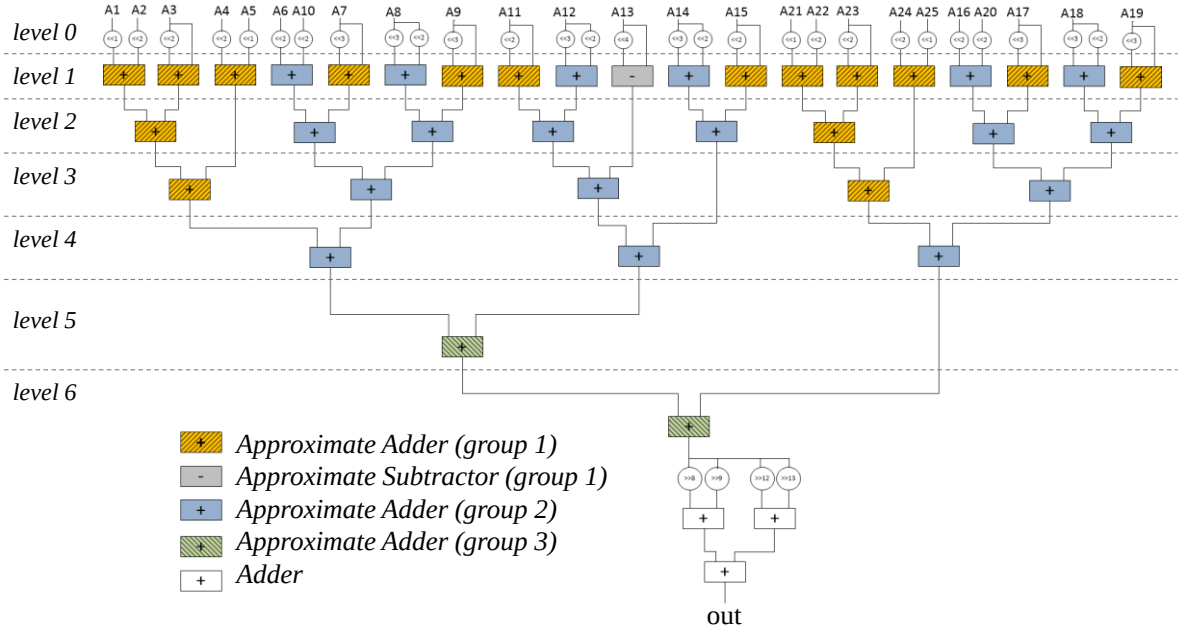


Figura 4.11 Arquitetura do Filtro Gaussiano 5x5 σ=1.4 com os grupos dos somadores aproximados.

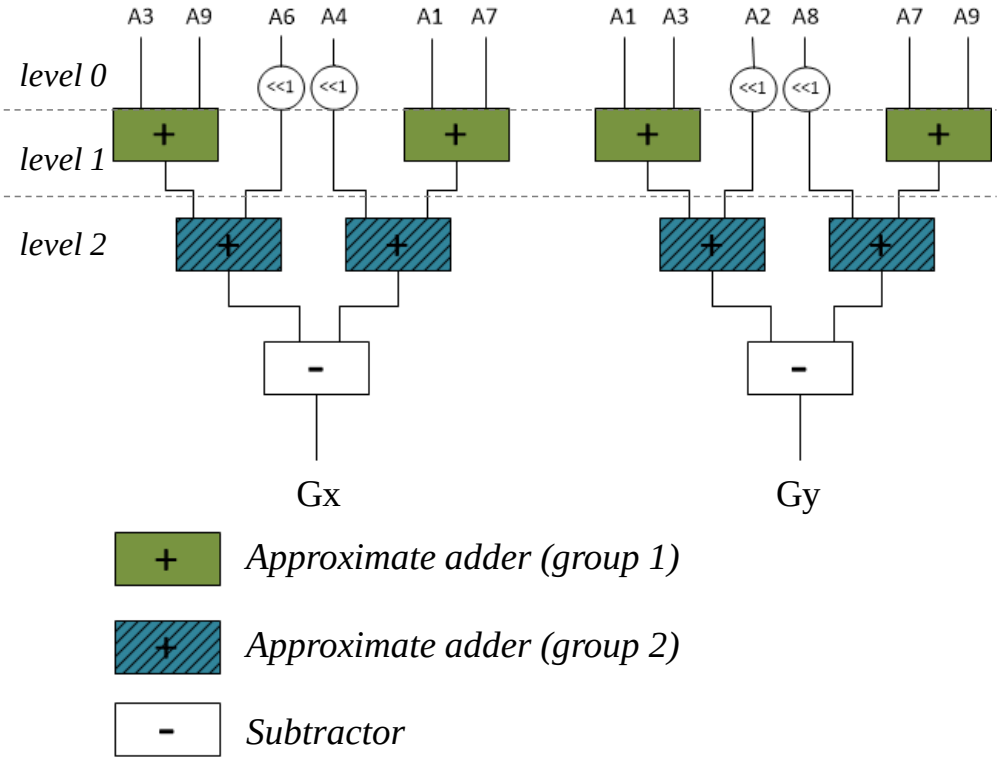


Figura 4.12 Arquitetura do Filtro Gradiente 3x3 com os grupos de somadores aproximados

O algoritmo 1 apresenta a heurística proposta para procurar um bom relacionamento com relação à eficiência energética e a qualidade de saída. As entradas são: i) a árvore de somadores Q contendo todas as ligações entre os somadores, que são os nós desta árvore; ii)

as listas S e R contendo todos os valores que multiplicam os operandos de entrada desta árvore Q; iii) uma lista com valores limites T para determinar a qual grupo cada somador pertence. As saídas são: i) a árvore de somadores P contendo o grupo que cada somador da árvore Q pertence; ii) a árvore de somadores W contendo os pesos computados que foram utilizados para formar os grupos.

O algoritmo começa inicializando duas novas árvores contendo as mesmas conexões fornecidas pela árvore de somadores Q. O primeiro *loop* é usado para agrupar todos os somadores que pertencem ao nivel 0 ou às folhas da árvore. Por exemplo, nas figuras 4.11 e 4.12, o nível 0 são os nós representados pela linha pontilhada na horizontal. Esses nós têm como entrada, os operandos que são multiplicados por constantes (implementadas por deslocamentos). Os pesos são organizados em duas listas R e S. A primeira tem a entrada à esquerda dos operandos, enquanto que a outra lista tem os operandos à direita dos somadores do nível 0. Assim, o processo de agrupamento para este nível específico é realizado como se segue: para cada somador j^{th}, se ambos os pesos *rj* e *sj* são maiores ou iguais do que o i^{th} limite para o valor de *ti* definido para cada grupo. Então o somador correspondente será do grupo i^{th}. Em outras palavras, os somadores sobre as folhas são agrupados pela maior magnitude de entrada. Isto é feito para um melhor controle da magnitude do erro inicial de toda a árvore de somadores.

Uma vez que o agrupamento é concluído para os somadores no nível 0 , em seguida, o próximo passo é determinar o processo de agrupamento para os níveis restantes da árvore de somadores. Para os somadores do nível superior os valores correntes são determinados pelos valores dos somadores do nível anterior que servem de entrada destes. Em outras palavras, estão sendo armazenadas informações sobre os somadores que fornecem operandos de entrada para o somador corrente a ser agrupado em um grupo. Em seguida, para cada possível valor de entrada, se o valor corrente é maior ou igual do que k^{th} de entrada, então o somador corrente será agrupado conforme o valor corrente. Caso contrário, o somador corrente será agrupado pelo valor máximo do grupo dos somadores que fornecem as entradas do somador.

Além disso o peso atual w_{ij} é atualizado com o *currentvalue* para ambas condições. Este procedimento é repetido até o último somador ser agrupado. Isto porque é possível determinar o número de somadores que se deseja aproximar. Na figura 4.11, os somadores não agrupados são os que correspondem à divisão por 159 que resulta da convolução do filtro Gaussiano, como podemos ver na figura 3.1 onde o valor da divisão é 273, pois o σ=1.0. Na figura 4.12, os subtratores não são aproximados a fim de evitar ou reduzir a possibilidade de sinal incorreto para as derivadas. Pode-se observar que, para o exemplo das figuras 4.11 e

4.12, são determinados três e dois grupos diferentes, respectivamente. Uma vez que todos os somadores agrupados pertencem a um grupo específico, em seguida a simulação pode ser realizada com a determinação dos valores de intervalos para cada grupo. Por exemplo, na figura 4.11, optou-se pelos seguintes intervalos para cada grupo: i) 1 a 3 para o grupo 1, ii) 1 a 6 para o grupo 2, e iii) 1 a 10 para o grupo 3. Isso é viável, visto que os grupos menores estão relacionados com somadores que eventualmente possuem operandos com magnitude menor. Em outras palavras, não há nenhuma necessidade de testar maiores valores de k para as aproximações destes somadores, porque a magnitude do erro iria degradar substancialmente a qualidade da aplicação. Durante as iterações nas simulações, todos os somadores que pertencem a um determinado grupo, irão variar uniformemente, de acordo com o intervalo estabelecido para eles. Esse procedimento reduz substancialmente o tempo de simulação a fim de viabilizar o processo pela busca das aproximações de um conjunto de somadores. Para o exemplo da figura 4.11, é necessário executar uma simulação com 180 configurações diferentes. Uma vez que o filtro Gaussiano é usado para borrar e remover os ruídos das imagens, então se utiliza a métrica o PSNR (*Peak Signal to Noise Ratio*) para avaliar a degradação introduzida pela aproximação no filtro. Por isso, nas 180 configurações diferentes, o PSNR é avaliado para a imagem filtrada aproximada, utilizando como referência a imagem filtrada precisa. Para executar a simulação, utilizam-se 8 imagens na escala de cinza com pixels de 8 bits. Este banco de imagens conta com 7 imagens genuínas do MATLAB e uma imagem de referência, ou seja “Lena”.

A figura 4.13 mostra as avaliações das configurações versus o PSNR médio, com respeito às aproximações usando os somadores baseados em cópia de bits em (a) e ETA I em (b). Pode-se observar que as aproximações com o somador ETA I apresenta melhores resultados de PSNR do que as aproximações com o somador baseado em cópia de bits, considerando-se o mesmo número de configurações. Foram selecionadas as configurações, que produzem um resultado de PSNR médio mais próximo a 30 dB. Esta escolha é fundamentada na literatura, conforme mencionado em (He; Gerstlauer; Orshansky, 2011) e (Park; Choi; Roy, 2010), que afirmam que com esse limite de 30 dB as imagens ainda mantêm uma qualidade na saída suficientemente boa. Portanto, as configurações selecionadas para os somadores baseado em cópia de bits e ETA I são as seguintes, respectivamente:

- grupo 1 (somadores baseados em cópia de bits) - somadores com k=3, grupo 2 - somadores com k=6, e grupo3 - somadores com k=8;
- grupo 1 (somadores ETA I) - somadores com k=3, grupo 2 - somadores com k=6, e grupo 3 - somadores com k=10.

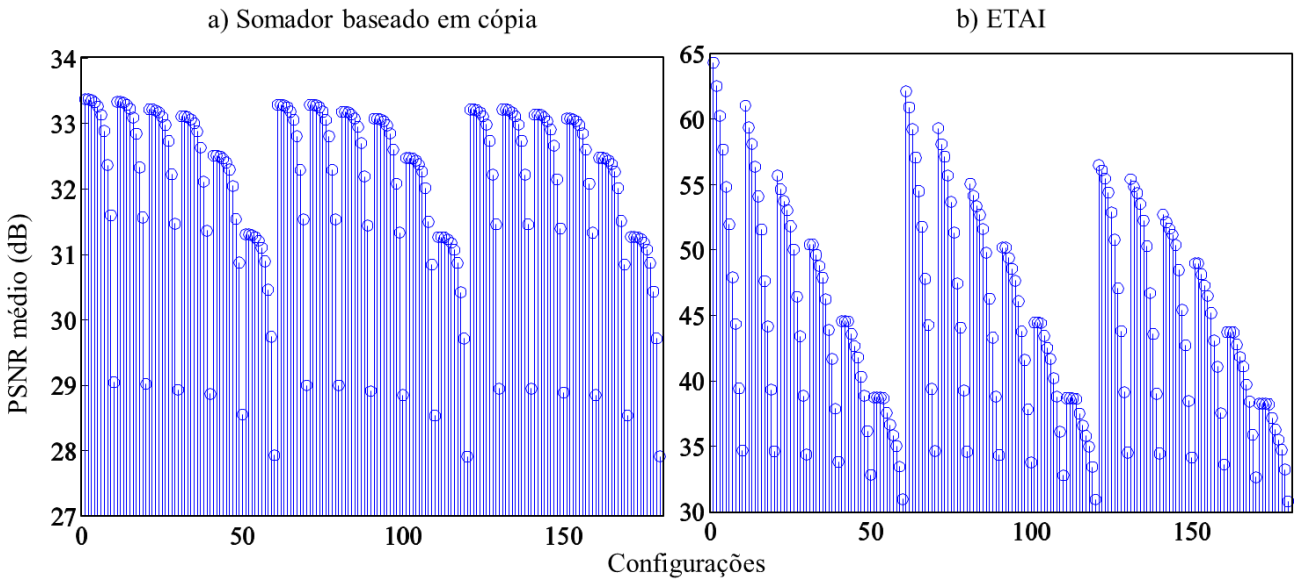


Figura 4.13 Configurações para os filtros Gaussianos 5x5 com: a)Somador baseado em cópia de bits e b) ETA I

A mesma metodologia de parametrização de k é utilizada para o filtro Gradiente. Desta forma, Foram selecionados para o grupo 1 e o grupo 2 de somadores, variação dos valores de k na faixa de 1 a 4 e 1 a 6, respectivamente.

Após a realização da avaliação das aproximações para o filtro Gradiente, foram analisadas as saídas das bordas da imagem após passar pelas etapas restantes do algoritmo de detecção de bordas de Canny. Foi considerada a imagem da borda filtrada pelo filtro Canny preciso como referência para calcular a diferença entre o filtro preciso e cada versão aproximada. Uma vez que uma borda da imagem é representada por um pixel branco (isto é, valor igual a 255), e a ausência de uma borda é representada por um pixel preto (isto é, valor igual a 0), então os valores de -255 e 255 indicam que uma borda foi suprimida e criada, respectivamente. Portanto, uma forma intuitiva de compreender esta informação é que quanto maior for o número de bordas excluídas ou criadas, menor a qualidade da saída. Se for executada a métrica do PSNR, irá resultar numa média que não dá qualquer informação sobre a qualidade da imagem com relação a detecção de bordas. Logo, adotou-se a métrica PR (*Performance Ratio Metric*) apresentada em (Soares; Costa; Bampi, 2015), e mostrada na Equação 7.

$$PR = \frac{ntrue}{nfalse}\ 100 \tag{7}$$

Na equação 7, *ntrue* se refere ao número de bordas detectadas na imagem filtrada precisa, que também é detectado pela imagem filtrada aproximada. O termo *nfalse* representa

o número de bordas que são apagadas ou criadas pela imagem aproximada. Observou-se que, um valor maior da métrica PR indica uma melhor qualidade de saída na detecção de bordas.

As figuras 4.14 e 4.15 mostram resultados para três diferentes configurações considerando o filtro Gradiente aproximado utilizando os somadores baseado em cópia de bits e ETAI, respectivamente. Os resultados são apresentados para ambos os somadores considerando as seguintes parametrizações: i) grupo 1 com somadores com k=4, e grupo 2 com somadores com k=4, ii) grupo 1 com somadores com k=4, e grupo 2 com somadores com k=5, e iii) grupo 1 com somadores com k=4, e grupo 2 com somadores com k=6. Pode-se notar que, para ambos os somadores quando se utiliza o k com o maior parâmetro, as bordas das imagens aproximadas ficam mais sensíveis à detecção de borda.

Apresentam-se também os resultados em termos de PR médio na Tabela 4.3, considerando-se as 8 imagens usadas como *benchmarking*. Considerando uma análise subjetiva, bem como a métrica PR, os filtros Gaussiano 5x5 e Gradiente 3x3, ambos implementados com o somador aproximado ETA I, apresentaram melhor qualidade de saída com relação aos implementados com o somador baseado em cópia de bits. Para ambos os filtros Gradientes aproximados, após analisar a tabela 4.3, decidiu-se adotar somadores aproximados do grupo 1 com k=4 e do grupo 2 com k=5 para avaliar a eficiência energética.

a) b) c) d)

Figura 4.14 Análise subjetiva para a detector de bordas de Canny quando utilizado o somador baseado em cópia de bits - a) Detector de bordas de Canny preciso; b) Aproximação com grupo 1 – k=4 e grupo 2 – k=4; c) Aproximação com grupo 1 – k=4 e grupo 2 – k=5; d) Aproximação com grupo 1 – k=4 e grupo 2 – k=6.

a) b) c) d)

Figura 4.15 Análise subjetiva para a detector de bordas de Canny quando utilizado o somador ETA I - a) Detector de bordas de Canny preciso; b) Aproximação com grupo 1 – k=4 e grupo 2 – k=4; c) Aproximação com grupo 1 – k=4 e grupo 2 – k=5; d) Aproximação com grupo 1 – k=4 e grupo 2 – k=6.

TABELA 4.3 ANÁLISE DO PR MÉDIO

Aproximação	com somador baseado em cópia de bits	com somador ETAI
4-4	115.72	113.85
4-5	75.20	92.39
4-6	27.71	41.71

4.3.2 Arquitetura proposta para o Detector de Bordas de Canny

A fim de avaliar os resultados de eficiência energética, foi proposta uma arquitetura de hardware, em conjunto com o doutorando da UFRGS Leonardo Bandeira Soares, para avaliar a arquitetura precisa e duas versões aproximadas para a aplicação do algoritmo de detecção de bordas de Canny completo. Alguns trabalhos da literatura propõem uma arquitetura para o detector de bordas de Canny baseada em FPGA (Xu; Varadarajan; Chakrabarti, 2014) e (Sangeetha, 2016). Entretanto, de acordo com a revisão bibliográfica realizada neste trabalho, não há nenhum trabalho que propõe uma arquitetura para o detector de bordas de Canny, que seja sintetizada para um projeto em ASIC e também que leve em consideração os conceitos de computação aproximada.

A arquitetura proposta processa linha por linha de qualquer tamanho de imagem e tem uma taxa de transferência de um pixel de 8 bits para cada ciclo de relógio. Para atingir esta taxa existem dependências entre os blocos dos algoritmos de detecção de bordas de Canny que foram otimizadas com o uso de bancos de registradores. A figura 4.16 mostra o caminho de dados da arquitetura proposta de detecção de bordas de Canny, onde a arquitetura recebe como entrada uma imagem de pixels de 8 bits e coloca na saída a imagem com as bordas processadas pelo algoritmo de detecção de bordas de Canny.

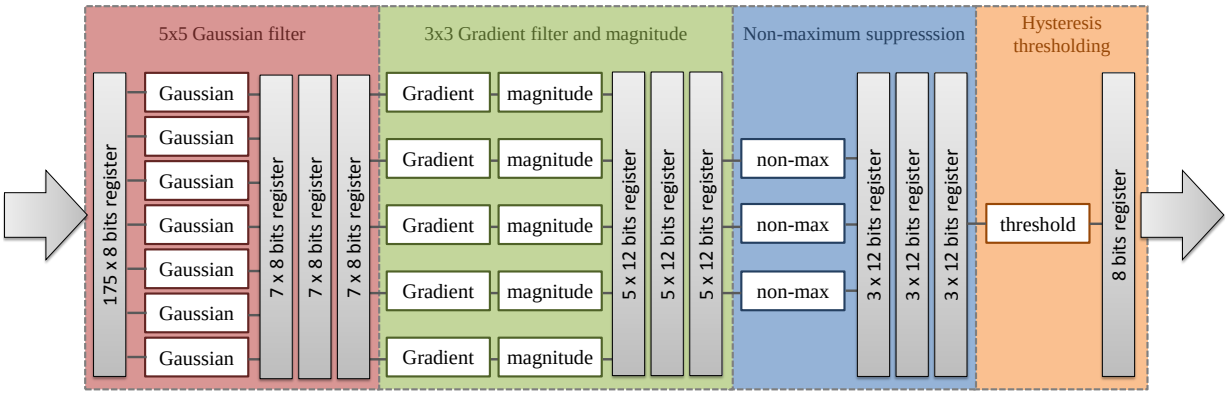


Figura 4.16 Arquitetura proposta para o algoritmo de detecção de bordas de Canny

4.3.2.1 Resultados do detector de bordas de Canny

A fim de avaliar o desempenho do algoritmo de detecção de bordas de Canny aproximado em comparação à solução precisa, adotou-se como imagem das bordas ideal, uma imagem que foi chamada de *Ground Truth*. A imagem *Ground Truth* foi adquirida de um banco de dados composto por imagens segmentadas por observadores humanos e foi usada na área de visão computacional para mensurar o desempenho de um dado algoritmo de detecção de bordas. Também se usa para análise de desempenho, o conjunto de imagens BSD (*Berkley Segmentation Dataset*) (Martin; Malik; 2001), fazendo-se uma comparação entre a arquitetura precisa e as versões aproximadas.

Foram descritos programas no ambiente MATLAB dos algoritmos de detecção de bordas de Canny aproximados e precisos a fim de analisar os resultados obtidos. Foram adotadas duas métricas objetivas nesta avaliação: i) a métrica PR mencionada anteriormente, ii) e medida F (Pontuação F1) como mostrado na equação 8 (Martin; Malik, 2004).

$$F\ measure = 2\frac{precision \cdot recall}{precision + recall} \tag{8}$$

Na equação 8, o parâmetro *precision* se refere ao número de bordas corretamente detectadas divididas pelo número total de bordas detectadas. O parâmetro *recall* se refere ao número de bordas detectadas corretamente dividido pelo número correto de bordas que deveriam ser detectadas. Portanto, a medida F é uma média harmônica de precisão e *recall.* Quanto maior for o valor de F melhor será o resultado.

Figura 4.17 Análise da detecção de bordas: a) Imagem da BSD database, b) *Ground truth*, c) Arquitetura precisa do detector de bordas de Canny, d) Arquitetura do detector de bordas de Canny com somador baseado em cópia de bits, e) Arquitetura do detector de bordas de Canny com somador ETA I, f) Detector de bordas de Canny preciso em ponto flutuante do MATLAB, g) Detector de bordas preciso baseado no operador Sobel do MATLAB.

Foram selecionadas sete imagens do banco de dados BSD, cuja análise subjetiva é mostrada por uma imagem na figura 4.17 da seguinte maneira: a) a imagem da base de dados BSD, b) o *Ground truth*, c) a arquitetura precisa do detector de bordas de Canny, d) a arquitetura do detector de bordas de Canny com somador baseado em cópia de bits, e) a arquitetura do detector de bordas de Canny com o somador ETA I, f) o detector de bordas de Canny, baseada no modelo em ponto flutuante no MATLAB, g) o detector de bordas com o operador Sobel realizado no MATLAB.

Com base apenas na análise subjetiva, pode-se observar que as arquiteturas precisas e aproximadas do detector de bordas de Canny, isto é, da figura 4.17c até a figura 4.17e, apresentam uma detecção de bordas semelhante. Como mencionado anteriormente as arquiteturas aproximadas tendem a ser mais sensíveis às bordas para um mesmo limiar de entrada. Portanto, a diferença entre as imagens nas figuras 4.17(c) até 4.17(e) é a presença de mais pontos detectados nas versões aproximadas.

A função de Canny do MATLAB na figura 4.17(f) detecta mais arestas do que as outras soluções. Qualquer diferença na luminância resulta na detecção de uma borda neste caso. Pode-se observar que as versões aproximadas apresentam mais bordas detectadas que são parecidas com as que foram detectadas pelo algoritmo detector de bordas de Canny preciso. O algoritmo de Sobel do MATLAB, na figura 4.17(g), apresenta menor qualidade subjetiva do que as outras abordagens, uma vez que algumas arestas não são detectadas ou ocorre uma degradação.

A tabela 4.4 mostra os resultados usando as métricas PR e F com o objetivo de avaliar a detecção de bordas. Na tabela as colunas "SCORE" se referem à métrica F. Para uma melhor análise das imagens, o detector de bordas de Canny do MATLAB, que usa ponto flutuante, apresenta os melhores resultados para ambas as métricas PR e F (*score*). Por outro lado, o detector de bordas Sobel do MATLAB possui os menores resultados para ambos os indicadores, quando em comparação com os outros detectores de bordas. Os resultados interessantes são relacionados com as arquiteturas de detecção de bordas de Canny aproximadas com somadores baseado na cópia de bits e ETA I, que na maioria dos casos possuem melhores resultados de PR e F, com relação à arquitetura precisa. A hipótese para esse resultado está relacionado com a maior sensibilidade em relação às bordas das versões aproximadas. Na verdade, como foi mostrado previamente nas fig.4.14, fig.4.15 e na tabela 4.3, existe um limite para realizar aproximações no filtro Gradiente. Para um maior valor de k de aproximação do somador, a imagem sofre uma substancial degradação. Os resultados mostram que a detecção de borda das imagens com aproximação, apresentam uma qualidade de saída semelhante ou boa suficiente.

TABELA 4.4 ANÁLISE DOS PARÂMETROS PR E F PARA OS DIFERENTES DETECTORES DE BORDA.

Imagem	Arquitetura Precisa		com somador baseado em cópia de bits		com ETAI		Canny Matlab		Sobel Matlab	
	PR	Score	PR	Score	PR	Score	PR	Score	PR	Score
11.jpg	17.00	0.011	17.98	0.012	18.80	0.013	17.56	0.009	9.63	0.005
14.jpg	18.70	0.008	19.24	0.009	19.60	0.019	21.93	0.008	15.42	0.004
22.jpg	13.89	0.012	14.43	0.014	14.83	0.016	16.74	0.014	11.17	0.001
24.jpg	12.44	0.013	13.55	0.014	13.01	0.013	16.31	0.017	9.98	0.001
26.jpg	19.69	0.003	19.29	0.003	19.60	0.004	20.22	0.002	16.79	0.003
31.jpg	8.17	0.006	9.58	0.011	10.02	0.008	12.29	0.012	6.12	0.004
32.jpg	11.18	0.009	12.07	0.011	12.31	0.011	14.06	0.014	8.90	0.006

4.3.2.2 Resultados de Consumo de Energia

As arquiteturas implementadas precisas e aproximadas foram totalmente descritas em linguagem de descrição de hardware VHDL e sintetizadas na tecnologia 45ηm usando a biblioteca *Nangate Open Cell Library* (Zhu; Goh; Yeo, 2010) na ferramenta da Cadence RTL Compiler. Para estimar a frequência máxima que foi realizada a síntese usou-se o método de *bisection* search (Bilal; Richard, 2016). A tabela 4.5 mostra os resultados de síntese das arquiteturas completas precisa e aproximadas do detector de bordas de Canny, usando os

somadores aproximados. Os resultados são expressos em termos de área (em termos do número de portas NAND2 X1, EG na tabela 4.5), número de células, potência estática, potência dinâmica, potência total, e média de energia para cada borda da imagem.

Tabela 4.5 Resultados de síntese para as arquiteturas de detecção de bordas de Canny @ 350 MHz

	Precisa	Com somador baseado em cópia de bits	Com somador ETAI
Area (EG)	60945	45758	49956
# Células	26605	20624	23023
Potência estática (mW)	1.2	0.9	1
Potência dinâmica (mW)	31.3	19.4	17.1
Potência total (mW)	32.5	20.3	18.1
Média de Energia por 512 x 512 pixels de 8-bits para cada borda da imagem (µJ)	24.73	15.45	13.77

Para os resultados de energia foi considerado um circuito lógico *switching activity extraction,* usado para extração de chaveamento em circuitos, para identificar quando o detector de bordas de Canny processa 5.000 pixels de saída a partir de imagens reais, este um limite prático para não sobrecarregar a simulação RTL. A arquitetura precisa em ASIC atinge uma frequência máxima de 362 MHz, enquanto que as arquiteturas com os somadores baseado em cópia de bits e ETA I, atingem frequências máximas de 372,5 MHz e 371,8 MHz, respectivamente. Se for considerada a frequência máxima de operação, a arquitetura precisa, e as arquiteturas aproximadas com somador baseado em cópia de bits e somador ETAI do detector de bordas de Canny, irão precisar de 0,74 ms, 0,71 ms e 0,72 ms para processar uma imagem 512 x 512 de 8bits, em 0,74 ms, 0,71 ms e 0,72 ms, respectivamente. Isto é devido aos oito ciclos de relógio de latência necessários para processar cada fila.

De acordo com os resultados apresentados na tabela 4.5, observa-se que houve uma redução de área de 24,92% e 18,03% para as arquiteturas aproximadas com os somadores baseados em cópia de bits e ETA I, respectivamente. Enquanto as arquitetura propostas (as versões precisas e aproximadas processando a 350 MHz) gastaram 0,761 ms para processar uma imagem de 512 x 512 e 8 bits, os trabalhos apresentados em (Xu; Varadarajan; Chakrabarti, 2014) e (Sangeetha; Deepa; 2016), gastaram 0,721 ms e 1 ms, respectivamente, para processar o mesmo tamanho de imagem. De acordo com (Sangeetha; Deepa; 2016), 1 ms para processar uma imagem de 512 x 512 é o suficiente para um processamento em tempo

real. Assim, a energia média para processar uma imagem de 512 x 512 de 8 bits, em μJ, também é mostrada na tabela 4.5. O consumo de energia foi avaliado a partir de simulações em nível lógico. As reduções de energia obtidas pelos circuitos aproximados com relação ao circuito preciso foram de 37,53% e 44,32%, para as arquiteturas com os somadores aproximados baseado em cópia de bits e ETA I, respectivamente.

Com base nos resultados de eficiência de consumo de energia, pode-se concluir que a heurística proposta para as aproximações da arquitetura do algoritmo de detecção de bordas de Canny traz uma substancial economia de energia, com um processamento em tempo real e sem degradar a qualidade da detecção de bordas. Os principais resultados obtidos da implementação completa do detector de bordas de Canny foram submetidos à revista Transactions on Circuits aand Systems (TCAS-I), que está em avaliação dos revisores.

4.4 Resumo do Capítulo

Este capítulo abordou o uso dos somadores baseados na cópia de bits e ETA I nos filtros Gaussiano, Gradiente e detector de bordas de Canny. Inicialmente foi mostrada a metodologia para a obtenção dos melhores valores de k (número de bits menos significativos que são copiados para a saída no somador baseado em cópia de bits). Esta metodologia foi testada inicialmente no filtro Gaussiano. Essa mesma metodologia foi aplicada para a seleção dos melhores valores de k para os filtros Gaussiano e Gradiente, visando à aplicação no detector de bordas de Canny. Como a seleção dos melhores valores de k envolvem um tempo de processamento elevado, foi proposto no âmbito deste trabalho um algoritmo baseado em heurística para uma melhor seleção dos valores de k. Este algoritmo foi apresentado neste capítulo. Para a análise do desempenho e da precisão na detecção de bordas, foi implementada em ASIC a arquitetura completa do detector de bordas de Canny. Os principais resultados mostraram que as arquiteturas de detecção de bordas de Canny aproximadas, com somadores baseados na cópia de bits e ETA I, na maioria dos casos possuem melhores resultados de relacionamento entre desempenho e precisão, com relação à arquitetura precisa. O capítulo também mostrou que as aproximações das arquiteturas dos filtros Gaussiano e Gradiente com os somadores baseados em cópia de bits e ETA I, trazem uma substancial economia de área e energia ao circuito detector de bordas de Canny,

5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou o estudo e a implementação de circuitos somadores aproximados para aplicação em filtros de processamento de imagens. Os resultados mostraram a eficiência da aplicação dos somadores aproximados baseados em cópia de bits e ETA I em filtros Gaussiano e Gradiente, com reduções de área e consumo de energia.

Em relação ao somador baseado na cópia de bits, foi apresentada uma metodologia para a obtenção dos melhores valores de k, que representa a quantidade de bits que são copiados para a saída. Foi proposto no âmbito deste trabalho um algoritmo para encontrar os melhores valores de k para a composição da árvore de somadores nos filtros Gaussiano e Gradiente. Para avaliação da eficiência energética versus a qualidade das imagens, foi implementado em ASIC o algoritmo completo de detecção de bordas de Canny. Foram utilizadas imagens reais para análise da detecção de bordas em relação às aproximações propostas. Os resultados mostraram que as arquiteturas de detecção de bordas de Canny aproximadas, com somadores baseados na cópia de bits e ETA I, na maioria dos casos possuem melhores resultados de PR e F (métrica de desempenho e métrica de precisão, respectivamente), com relação à arquitetura precisa.

Os resultados da síntese em ASIC mostraram que, as aproximações das arquiteturas dos filtros Gaussiano e Gradiente com os somadores baseado em cópia de bits e ETA I do algoritmo de detecção de bordas de Canny trazem uma substancial economia de área e energia, com um processamento em tempo real e sem degradar a qualidade da detecção de bordas.

5.1 Trabalhos Futuros

Os estudos realizados com a implementação do detector de bordas de Canny completo, oportunizou a submissão de um artigo para a revista TCAS-I (IEEE Transactions on Circuits and Systems I: Regular Papers), a partir de um convite para uma versão estendida da conferência LASCAS 2016. O artigo está em processo de revisão e, espera-se que em breve, os resultados possam estar publicados nesta conceituada revista internacional.

Embora esta dissertação tenha focado no estudo e implementação de somadores aproximados, o trabalho abre a possibilidade de estudo de outros operadores aritméticos aproximados, tais como somadores compressores e circuitos multiplicadores.

A implementação completa da arquitetura do detector de bordas de Canny abre uma perspectiva da aplicação deste circuito no Processamento de imagens médicas, que conta como um dos blocos principais, o algoritmo de detecção de bordas de Canny.

REFERÊNCIAS

J. HAN and M. ORSHANSKY, “Approximate computing: An emerging paradigm for energy-efficient design,” in Test Symposium (ETS), 2013 18th IEEE European, Avignon, 2013, pp. 1–6.

T. HAIDER, “Energy Efficient Computing for a Long Battery Life - Emerging Model of Approximating Computing,” in OSTI - A Gem of Scientific Repositories, October 2013.

A. B. KAHNG and S. KANG, “Accuracy-configurable adder for approximate arithmetic designs,” in Proceedings of the 49th Annual Design Automation Conference, San Francisco, 2012, pp. 820–825.

V. GUPTA, D. MOHAPATRA, A. RAGHUNATHAN, and K. ROY, “Low-Power Digital Signal Processing Using Approximate Adders,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

P. ALBICOCCO, G. CARDARILLI, A. NANNARELLI, M. PETRICCA, and M. RE, "Imprecise Arithmetic for Low Power Image Processing," in 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR), pp. 983-987, 2012.

R. GONZALEZ, R. WOODS, and S. EDDINS, Digital Image Processing Using Matlab. Gatesmark Publishing, 2009.

K. HE, A. GERSTLAUER, and M. ORSHANSKY, “Controlled timing-error acceptance for low energy IDCT design,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011, Grenoble, 2011, pp. 1–6.

M. MONAJATI, S. M. FAKHRALE, and E. KABIR, “Approximate Arithmetic for Low-Power Image Median Filtering,” *Circuits Systems and Signal Processing*, February 2015.

J. PARK, J. CHOI, K. ROY. "Dynamic Bit-Width Adaptation in DCT: An Approach to Trade Off Image Quality and Computation Energy," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 18, n. 5, p. 787-793, Mai 2010.

N. ZHU, W. L. GOH, W. ZHANG, K. S. YEO, and K. S. KONG, “Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 8, pp. 1225–1229, Aug. 2010.

L. SOARES, E. COSTA, and S. BAMPI. “Approximate Adder Synthesis for Area- and Energy- Efficient FIR Filters in CMOS VLSI” in 13th IEEE International NEW Circuits And Systems (NEWCAS) conference , Grenoble, 2015.

NING ZHU, WANG LING GOH, and KIAT SENG YEO, “An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application”,*International Symposium on Integrated Circuits (ISIC 2009), Singapore, 14 to 16 December 2009, Singapore, pp. 69-72.*

NING ZHU, WANG LING GOH, and KIAT SENG YEO, “An Enhanced Low-Power High-Speed Adder For Error-Tolerant Application”, *International SoC Design Conference (ISOCC 2010), Songdo Incheon, 22 to 23 November 2010, Korea, pp. 323-327.*

A. CHANDRAKASAN, R. BRODERSEN, "Low Power Digital CMOS Design," Dordrecht. Kluwer Academic, 1995.

COSTA, E. A. C., "Operadores Aritméticos de Baixo Consumo para Arquiteturas de Circuitos DSP". 2002. Tese (Doutorado em Ciência da Computação) — Programa de Pós-Graduação em Computação - UFRGS, Porto Alegre, RS.

L. BENDA, P. MUDRY, and A. IJSPEERT, "Hardware Acceleration for Image Processing," Technical report, EPFL, January 2008,

CANNY, J. A Computational approach to edge detection. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 8, n. 6, p.679-698, 1986.

N. CHANDRASHEKAR and K. NATARAJ, "NMS and Thresholding Architecture used for FPGA based Canny Edge Detector for Area Optimization, "Proceedings of International Conference on Control, Communication and Power Engineering, 2013, pp. 80-83.
T. MOESLUND, "Canny Edge Detection," Notes courtesy Prof. Thomas B. Moeslund, March 2009, pp.1-7.
OLIVEIRA, JULIO. R., SOARES, LEONARDO. B.; COSTA, E.A.C; BAMPI, SERGIO, "Energy-Efficient Gaussian Filter for Image Processing Using Approximate Adder Circuits",December 2015, icecs 2015.
L. BENDA, "Hardware Acceleration for Image Processing," Raport EPFL, I&C, BIRG, 2008, pp.1-45.
OLIVEIRA, JULIO R., SOARES, LEONARDO B.; COSTA, E.A.C; BAMPI, SERGIO,"Exploiting Approximate Adder Circuits for Energy-Efficient Gaussian and Gradient Filters for Canny Edge Detector Algorithm", February 2016, Lascas 2016.
J. CANNY, "tA computational approach to edge detection, " *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.6, 1986, pp. 679-698.
QIAN XU, SRENIVAS VARADARAJAN, CHAITALI CHAKRABARTI, "A Distributed Canny Edge Detector: Algorithm and FPGA Implementation", *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 2944 – 2960, July 2014.
SANGEETHA D and DEEPA P, "An Efficient Hardware Implementation of Canny Edge Detection Algorithm", International Conference on VLSI Design, pp. 457 – 462, 2016.
K. HE, A. GERSTLAUER, and M. ORSHANSKY, "Controlled timing-error acceptance for low energy IDCT design," *in Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2011, pp. 1–6.
D. MARTIN and C. FOWLKES and D. TAL and J. MALIK, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", Proc. 8th Int'l Conf. Computer Vision, vol. 2, pp. 416–423, July, 2001.
D. R. MARTIN, C. C. FOWLKES and J. MALIK. "Learning to detect natural image boundaries using local brightness, color, and texture cues", IEEE Transactions on Pattern Analysis and machine Intelligence, vol. 26, no. 5, pp. 530 – 549, May, 2004.
BILAL M. AYYUB, RICHARD H. MCCUEN. Textbooks in Mathematics. Numerical Analysis for Engineers - Methods and Applications. Second edition. CRC Press. 2016.
A. DAVARE, K. LWIN and A. KONDRATYEV. "The Best of Both Worlds: The Efficient Asynchronous Implementation of Synchronous Specifications", IEEE Transactions on Pattern Analysis and machine Intelligence, Proc. 41st anual Desing Automation Conference, pp. 588-591 , June, 2004.