UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ismael Seidel

Exploiting SATD Properties to Reduce Energy in Video Coding

Florianópolis
2020

Ismael Seidel

Exploiting SATD Properties to Reduce Energy in Video Coding

Tese submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de doutor em Ciência da Computação.
Orientador: Prof. José Luís Almada Güntzel, Dr.
Coorientador: Prof. Luciano Volcan Agostini, Dr.

Florianópolis
2020

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

Seidel, Ismael
Exploiting SATD Properties to Reduce Energy in Video Coding / Ismael Seidel ; orientador, José Luís Almada Güntzel, coorientador, Luciano Volcan Agostini, 2020.
254 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2020.

Inclui referências.

1. Ciência da Computação. 2. Video coding. 3. Hadamard based Sum of Absolute Transformed Differences. 4. Energy efficiency. 5. Hardware architecture. I. Güntzel, José Luís Almada. II. Agostini, Luciano Volcan. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Ismael Seidel

Exploiting SATD Properties to Reduce Energy in Video Coding

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Nuno Filipe Valentim Roma, Dr.
Universidade de Lisboa e INESC-ID Lisboa

Prof. Guilherme Ribeiro Corrêa, Dr.
Universidade Federal de Pelotas

Prof. Claudio Machado Diniz, Dr.
Universidade Católica de Pelotas

Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Ciência da Computação.

Profª. Vania Bogorny, Dr.
Coordenadora do Programa

Prof. José Luís Almada Güntzel, Dr.
Orientador

Florianópolis, 2020.

Aos meus pais, Lelinha, demais familiares e ao pessoal do ECL.

ACKNOWLEDGEMENTS

Aos meus pais, Leonice e Edgar, muito obrigado! Sempre contei com o apoio de vocês, sempre compartilhando as partes boas e as partes difíceis da caminhada. Terminar essa tese foi um trabalho e tanto, e sei que vocês ficavam tão aflitos quanto eu enquanto o tempo passava e eu não terminava. Ouvi muitas vezes: "quase terminando a tese?", e parecia que quanto mais fazia, mais faltava fazer... Bom, aqui esta ela. "Terminada." Qual será o próximo desafio agora? Sei que não importa qual seja, sempre continuarei com o apoio e o amor de vocês me ajudando no caminho. Então, novamente, muito obrigado!

À Rafaela, muito obrigado! Por me ajudar tanto nessa jornada, no crescimento acadêmico e pessoal. Por me amar quando eu estava animado trabalhando incessantemente, mas também enquanto estava sem vontade de sair da cama. "Só mais 5 minutos"... Quantas alucinações sobre a tese, sobre os artigos para apresentar. Obrigado por ser meu porto seguro, na loucura e também na calmaria. Muito obrigado, por me ajudar mesmo fazendo sua tese durante esse tempo! Por me apoiar nas decisões mais difíceis, por estar aqui do meu lado enquanto escrevo as últimas linhas deste texto. Agora é minha vez de contribuir mais. Dos dias que passaram, só posso agradecer: Obrigado!

Ao meu orientador, Güntzel, muito obrigado! Agradeço pela orientação na inciciação científica, TCC, mestrado e agora doutorado. Basicamente foi uma orientação de toda minha vida acadêmica! Obrigado pelas oportunidades que me proporcionastes e principalmente pela amizade ao longo dos anos que passei na UFSC. Também agradeço à Margarida e ao Matias, por me receberem na família. Foram uns quantos jantares e viagens compartilhadas... Obrigado!

Ao meu coorientador, Luciano, muito obrigado! Mesmo achando inicialmente que "só isso talvez não dê uma tese" sempre me deu um grande apoio! Minha ida para Pelotas, passando uma semana imerso no então GACI foi excelente para dar o impulso necessário para transformar as ideias nesta tese. Obrigado!

Aos amigos do ECL, muito obrigado! Foram alguns anos de crescimentos juntos! Com alguns mais tempo, com alguns menos. Agradeço em especial aos que também foram meus orientados, André, Bonotto, Luiz, Marcio e Vânio! Cada um contribuiu com essa tese, e ela não seria como é sem a ajuda e o esforço de vocês! Obrigado!!!

Aos membros da banca, Nuno, Cláudio, Guilherme e Willrich, muito obrigado! Agradeço por todas as contribuições ao texto e durante a defesa! Ao Guilherme agradeço duplamente, pois foi membro da banca de qualificação, além da defesa da tese. Agradeço também ao Luiz Cláudio, que também contribuiu na qualificação, e ao Djones e Laércio que contribuíram no seminário de andamento. Obrigado!

Agradeço também a todos que direta ou indiretamente participaram deste trabalho. Obrigado!

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

"The great enemy of the truth is very often not the lie-deliberate, contrived and dishonest – but the myth—persistent, persuasive and unrealistic... We enjoy the comfort of opinion without the discomfort of thought".
John F. Kennedy (1962).

" [...] the merit of every mathematical tool is in the simplicity and ease at which it facilitates calculations for a given class of problems [...]".
JUDEA PEARL, 'Application of Walsh Transform to Statistical Analysis' (1971).

RESUMO

O contínuo surgimento de novas aplicações de vídeo demanda também o aumento da eficiência da codificação. Entretanto, a lei de Moore é insuficiente para compensar o aumento da complexidade e, portanto, não se pode mais confiar que consiga sozinha viabilizar tais aplicações. Paradoxalmente, a maioria dos métodos para diminuir a complexidade reduzem a eficiência de codificação. A principal contribuição desta tese reside no aumento da eficiência energética da codificação de vídeo sem redução da eficiência de codificação. Por suas complexidades, adotamos a soma das diferenças transformadas absolutas (SATD), uma métrica de distorção baseada na transformada de Hadamard, e a estimação de movimento fracionária (FME), que pode utilizar a SATD em sua execução. Inicialmente, analisamos a eficiência da codificação de ambas através do modelo Bjøntegaard Delta (BD) no software de referência do padrão "High Efficiency Video Coding" (HEVC). Mostramos que o uso da SATD resulta em Bjøntegaard Delta Bitrate (BD-Rate) de -1,38%, em média e que o uso da FME é de suma importância, uma vez que quando desabilitada aumenta o BD-Rate em 10,89%, em média. A chave para as contribuições principais desta tese é a exploração de propriedades matemáticas da SATD quando usada durante a predição, reduzindo a energia necessária para o seu cálculo. Para tanto, exploramos três técnicas: reúso de cálculo, eliminação sucessiva (SEA), e eliminação da distorção parcial (PDE). Dentre elas, o reúso proporciona simplicidade e efetividade máximas, economizando 2/3 das operações de transformadas. A SEA baseada em SATD é mais simples em sua forma de nível único, mas apresenta uma efetividade inferior àquela de sua versão multinível. Em geral, ambas são capazes de eliminar até 80% dos blocos candidatos. Finalmente, mostramos que a PDE é capaz de reduzir em até 70% o número de blocos necessários. Ainda, apresentamos uma exploração do espaço de projeto de arquiteturas de hardware para a SATD, culminando com uma versão que explora simultaneamente o reúso e a organização do hardware. Tal exploração simultânea resultou em 28% menos área e 32% menos energia consumida do que a exploração somente do reúso. Propomos também, uma arquitetura para FME inicialmente com a soma das diferenças absolutas (SAD), métrica de distorção mais simples do que a SATD. Tal arquitetura é até 46% mais eficiente do ponto de vista energético do que as arquiteturas do estado da arte para o padrão HEVC. Modificamos a arquitetura, e demonstramos que a FME usando SATD demanda 2,6 vezes mais energia do que requerida pela FME usando SAD. Finalmente, desenvolvemos uma versão da arquitetura para FME que usa a SEA e a PDE, a qual resultou em reduções de energia de até 25% com um pequeno sobrecusto de área. Concluímos que as técnicas propostas provêm economias suficientes quando isoladas. Porém, o paralelismo proporcionado pelo hardware dedicado compromete a eficiência das técnicas. Embora as técnicas propostas não sejam suficientes para permitir projetos completos de codificadores com alta eficiência energética, elas certamente contribuem para melhorar a eficiência energética e, portanto, devem ser empregadas.

Palavras-chave: Codificação de vídeo. Soma das diferenças transformadas absolutas baseadas na Transformada de Hadamard. Eficiência energética. Arquitetura de hardware.

RESUMO EXPANDIDO

Introdução

Não há dúvidas de que os dispositivos móveis, em especial os "smartphones", mudaram o estilo de vida das pessoas no início do século XXI. Em especial, as aplicações de vídeo passaram a ter um grande impacto em nosso cotidiano, uma vez que tais dispositivos facilitam a captura, codificação e reprodução de vídeos. Nós nos comunicamos através de vídeos, aprendemos assistindo tutoriais e video aulas e muitas vezes armazenamos e compartilhamos momentos especiais. Não é de se surpreender que cerca de 75% dos dados que circulam através da Internet sejam oriundos de vídeos (CISCO, 2018). O futuro se aproxima com ainda mais aplicações para vídeos: estamos na iminência da imersão em tal mídia. Portanto, tudo indica que haverá uma quandidade cada vez maior de dados para comprimir, e comprimir de maneira cada vez mais eficiente. Afinal, até mesmo a contínua adoção de vídeos com resoluções cada vez maiores tem resultado em forte demanda por aumento da eficiência da codificação, que é a relação de custo-benefício entre taxa e distorção. Entretanto, a lei de Moore já não é mais suficiente para compensar o aumento exponencial da complexidade e portanto, não se pode mais confiar que ela consiga sozinha viabilizar tais novas aplicações. Assim, arquiteturas de hardware dedicado para codificação de vídeo são cada vez mais necessárias. Em tal cenário, a complexidade computacional está intimamente relacionada com a eficiência energética. Em geral, quanto maior a complexidade computacional de uma determinada ferramenta de codificação, maior será seu consumo energético. Paradoxalmente, a maioria dos métodos para diminuir a complexidade computacional causam a redução da eficiência de codificação. Particularmente, os codificadores de vídeo embarcados em dispositivos portáteis devem ser energeticamente eficientes a fim de prolongar a vida útil da bateria. Portanto, codificadores embarcados devem ter a relação entra a eficiência de codificação e eficiência energética bem equilibrada, uma vez que a satisfação do usuário depende tanto da qualidade da câmera (e dos algoritmos de compressão) quanto da duração da bateria. Além da satisfação do usuário, devemos considerar que um consumo exacerbado de energia reduz o tempo de vida das baterias, resultado em mais lixo eletrônico. Dado este jogo de compromissos, **a principal contribuição desta tese reside no aumento da eficiência energética da codificação de vídeo sem redução da eficiência de codificação**. Para tanto, foram identificadas ferramentas complexas do codificador, como a soma das diferenças transformadas absolutas (em inglês, Sum of Absolute Transformed Differences - SATD) baseada na transformada de Hadamard e a estimação de movimento fracionária (em inglês, Fractional Motion Estimation - FME). A SATD é uma métrica de distorção, ao passo que a FME é uma ferramenta de predição que pode utilizar a SATD em sua execução.

Objetivos

Considerando a demanda por aceleradores de codificadores de vídeo que apresentem alta eficiência energética e alta eficiência de codificação, levantamos a seguinte hipótese nesta tese: **É possível aplicar, de maneira eficiente, técnicas baseadas em propriedades matemáticas da SATD quando usada nas ferramentas de predição para aumentat a eficiência energética de arquiteturas de hardware dedicado para a FME e SATD sem reduzir a eficiência de codificação.** Logo, o objetivo geral desta tese é investigar, propor e avaliar algoritmos que exploram as propriedades matemáticas da SATD quando usada nas ferramentas de predição. Tais algoritmos visam reduzir a complexidade sem trocar seu ganho por redução na eficiência de codificação. Assim, através de tais algoritmos temos como objetivo o aumento da eficiência energética de aceleradores em hardware para codificadores de vídeo. Adicionalmente, para

validar a hipótese apresentada, outra parte do objetivo geral desta tese consiste em responder as seguintes perguntas: 1) Vale a pena, em termos de eficiência de codificação, usar a SATD durante a predição entre quadros? 2) Qual é o ganho, em termos de eficiência de codificação, obtido através da adoção da FME? 3) É possível superar as limitações impostas pela transformada no cálculo da SATD e aplicar técnicas que são utilizadas com os mesmos fins quando a Sum of Absolute Differences (SAD) é adotada como métrica de distorção? e 4) Caso a resposta à pergunta anterior seja positiva, tais técnicas, quando aplicadas na SATD, demonstram benefícios em termos de eficiência energética?

Metodologia

Há dois métodos de trabalho fundamentais que foram utilizados nesta tese. O primeiro método foi utilizado na análise da eficiência de codificação e validação dos algoritmos propostos, e consiste dos seguintes passos: 1) Implementação dos algoritmos propostos no software de referência do padrão High Efficiency Video Coding (HEVC), chamado de HEVC Model (HM); 2) Execução do HM sem os algoritmos propostos ou ferramentas em estudo para servir como base de comparação. Foram obtidos relatórios de taxa (bits) e distorção (Peak Signal-to-Noise Ratio (PSNR)); 3) Execução do HM com os algoritmos propostos ou ferramentas em estudo para obter sua eficiência em relação à base. Foram obtidos relatórios de taxa (bits) e distorção (PSNR); e 4) Comparação das duas execuções através de suas curvas de Rate-Distortion (RD), resumidas utilizando o modelo Bjøntegaard Delta (BD). Deste modelo, o Bjøntegaard Delta Bitrate (BD-Rate) foi escolhido por facilitar a comparação entre os resultados; No caso de análise de eficiência de codificação de ferramentas, tais quais a SATD e FME, reportamos e comparamos os resultados. No caso da validação dos algoritmos propostos, tal método permitiu verificar que tais algoritmos não reduziam a eficiência de codificação. O segundo método de trabalho foi utilizado para obtermos os resutados de área e potência, e através deste último calcular o consumo de energia das arquiteturas propostas. Tal método contém os seguintes passos: 1) Descrição e validação funcional das arquiteturas propostas; 2) Síntese das arquiteturas utilizando a ferramenta Synopsys® Design Compiler (DC®) no modo "Topographical". A definição do período alvo depende das arquiteturas (número de ciclos por tarefa) e busca atender demandas temporais para a codificação em tempo real de vídeos em resoluções atuais, tais quais Full HD (FHD) e Ultra High Definition (UHD); 3) Simulação funcional das arquiteturas sintetizadas ("netlists") na ferramenta DC® para garantir que a funcionalidade da arquitetura continua a mesma após a síntese. Este passo também possibilita a obtenção da atividade de chaveamento da arquitetura; 4) Execução da ferramenta de síntese (DC®) para obter relatórios de potência atualizados com base na atividade de chaveamento obtida no passo anterior; e 5) Execução da ferramenta de análise estática Synopsys® PrimeTime (PT®) para obter estimativas de potência detalhadas para cada ciclo de execução; Os passos 1 até 3 são fundamentais para obter estimativas de área e potência das arquiteturas, permitindo assim a comparação com outros trabalhos. Já os demais passos auxiliam a aumentar a acurácia das estimativas de potência e também permitem uma análise mais fina de tais resultados. Finalmente, para permitir a reprodutibilidade dos experimentos, as descrições dos algoritmos e arquiteturas foram disponibilizadas publicamente.

Resultados e Discussão

Como contribuição inicial, analisamos a eficiência da codificação considerando a SATD e a FME no HM. Mostramos que o uso da SATD resulta em BD-Rate de -1,38%, em média. Também mostramos que o uso da FME é de suma importância, em termos de eficiência de codificiação, uma vez que sua quando não habilitada aumenta o BD-Rate em 10,89%, em média. A chave para se chegar às contribuições principais desta tese residiu na exploração de

propriedades matemáticas da SATD quando ela é usada durante a predição, de modo a reduzir a energia necessária para o seu cálculo. Para tanto, exploramos três técnicas: reúso de cálculo, eliminação sucessiva (em inglês, Successive Elimination Algorithm (SEA)), e eliminação da distorção parcial (em inglês, Partial Distortion Elimination (PDE)). Estas três técnicas foram avaliadas em termos de simplicidade (sobrecusto), efetividade (capacidade de reduzir o número de operações) e economia. Dentre tais técnicas, o reúso é aquela que proporciona a simplicidade e a efetividade máximas, economizando 2/3 das operações de transformadas. A SEA baseada em SATD é mais simples em sua forma de nível único, mas apresenta uma efetividade inferior àquela de sua versão multinível. Em geral, ambas as adaptações da SEA para usar SATD são capazes de eliminar até 80% dos blocos candidatos, mesmo considerando apenas a FME. Finalmente, mostramos que a PDE é capaz de reduzir em até 70% o número de blocos necessários. A economia de PDE de baixa granularidade para SATD é praticamente equivalente à economia de PDE de alta granularidade porque ambas calculam o mesmo número de transformadas. Também apresentamos uma exploração do espaço de projeto de arquiteturas de hardware para a SATD, culminando com uma versão que explora simultaneamente o reúso e a organização do hardware. A exploração simultânea destes dois quesitos resultou em 28% menos área e 32% menos energia consumida do que a exploração somente do reúso. Propomos, também, uma arquitetura para FME que é eficiente tanto em codificação quanto em consumo de energia, inicialmente com a soma das diferenças absolutas (em inglês, Sum of Absolute Differences - SAD), métrica de distorção mais simples do que a SATD. Tal arquitetura é até 46% mais eficiente do ponto de vista energético do que as arquiteturas do estado da arte para o padrão HEVC. Também modificamos a arquitetura proposta para a FME, demonstrando que a energia demandada pela SATD é 2,6 vezes maior do que aquela requerida pela SAD. Finalmente, desenvolvemos uma versão da arquitetura para FME que usa a SEA e a PDE, a qual resultou em reduções de energia de até 25% com um pequeno sobrecusto de área.

Considerações Finais

Neste trabalho, demonstramos que as técnicas propostas para a SATD usada durante a etapa de predição são simples e mantêm o mesmo resultado de predição que é obtido sem o uso das técnicas propostas. Em outras palavras, demonstramos que as técnicas propostas não afetam, e portanto não reduzem, a eficiência de codificação. Ainda, concluímos que as técnicas propostas apresentam alta efetividades (até 80%) quando consideradas isoladamente, demonstrando assim alto potencial de redução do consumo energético. No entanto, o paralelismo proporcionado pelo hardware dedicado compromete a efetividade das técnicas. Embora as técnicas propostas não sejam suficientes para permitir projetos completos de hardware de codificadores de vídeo de alta eficiência energética, elas certamente contribuem para melhorar a eficiência energética e, portanto, devem ser empregadas. Além disso, qualquer técnica de economia de energia que não imponha perdas de eficiência de codificação deve receber a devida atenção. Afinal, mesmo quando combinadas com outras técnicas com perdas, elas não resultam em mais degradações na eficiência da codificação.

Palavras-chave: Codificação de vídeo. Soma das diferenças transformadas absolutas baseadas na Transformada de Hadamard. Eficiência energética. Arquitetura de hardware.

ABSTRACT

The continued emergence of new video applications with ever-increasing resolutions results in a dramatic demand for increased coding efficiency. Meanwhile, embedded video encoders must be energy-efficient to prolong portable devices' battery lifetime. However, Moore's law is no longer sufficient to compensate for the snowballing complexity increase and thus it cannot be expected anymore to be the sole enabler of those new applications. Moreover, most methods to reduce the computational complexity end up decreasing the coding efficiency. Given such game of trade-offs, the main contribution of this thesis is to improve the energy efficiency of video encoding without reducing coding efficiency. To achieve that, we identified two complex parts of the encoder: the Hadamard-based Sum of Absolute Transformed Differences (SATD), a distortion metric, and the Fractional Motion Estimation (FME), a prediction tool that may adopt the former during its execution. We then analyzed their coding efficiency, in terms of Bjøntegaard Delta Bitrate (BD-Rate), in the High Efficiency Video Coding (HEVC) reference software. We show that using SATD results in -1.38% BD-Rate, on average and that the use of FME is a must, as disabling it increases by 10.89% the average BD-Rate. The key to achieving the main contributions of this thesis was to exploit mathematical properties of the SATD when used during prediction, as to make SATD calculation less energy demanding. For that, we exploited three techniques: calculation reuse, Successive Elimination Algorithm (SEA), and Partial Distortion Elimination (PDE). We evaluated them in terms of simplicity (overhead), effectiveness (capacity do reduce the number of operations), and savings. Among those three techniques, the reuse provides the maximum simplicity and effectiveness, resulting in about 2/3 transform operations saved. The SATD-based SEA is simpler in its single-level form but has smaller effectiveness than its multi-level counterpart. In general, they are able to eliminate up to 80% of candidate blocks, even considering FME solely. Finally, we exploited PDE, that is able to reduce the number of blocks required in up to 70%. Furthermore, we present a design space exploration of SATD hardware architectures that culminate in the design of a hardware that exploits reuse together with the hardware organization. This results in 28% less area and consumes 32% less energy than exploiting reuse alone. We also propose a coding- and energy-efficient hardware architecture for FME, initially for the Sum of Absolute Differences (SAD), a distortion metric simpler than SATD. Such architecture is up to 46% more energy-efficient than state-of-the-art ones for the HEVC standard. We also modified our FME architecture, showing that the energy demands of SATD are $2.6\times$ higher than that of SAD. Finally, we adopted the proposed SEA and PDE in such FME hardware, showing that they reduce energy in up to 25% with a small area overhead. We conclude that the proposed techniques provide enough savings when considered isolated. However, dedicated hardware parallelism jeopardizes the efficiency of the techniques. Also, while the proposed techniques are not sufficient to achieve the desired energy efficiency in a video encoder design, they certainly help to improve energy efficiency and should be employed. Moreover, any energy saving technique that does not impose coding efficiency losses deserves high consideration. After all, even when combined with other lossy techniques, they will not result in further degradations to the coding efficiency.

Keywords: Video coding. Hadamard-based Sum of Absolute Transformed Differences. Energy efficiency. Hardware architecture.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

1D	1-Dimension	61
2D	2-Dimensions	19, 39
AdaMSEA	Adaptive Multilevel Successive Elimination Algorithm	128
ADS	Absolute Difference of Sums	109, 110
AFD	Absolute First Difference	111
AI	All Intra	86
ASIC	Application Specific Integrated Circuit	47
ASIP	Application Specific Instruction set Processor	47
AVC	Advanced Video Coding	40
B	Bi-Predicted	66
BD	Bjøntegaard Delta	11, 14, 23, 40
BD-PSNR	Bjøntegaard Delta PSNR	90
BD-Rate	Bjøntegaard Delta Bitrate	11, 14, 17, 23, 84
BM	Block Matching	164, 166
BMA	Block Matching Algorithm	60
bpp	bits per pixel	39, 42
BRAM	Block RAM	140
CAGR	Compound Annual Growth Rate	48
Cb	“Difference to blue”	58
CIE	Commission Internationale de l’Éclairage	59
CODEC	Coder/decoder	23, 40
CPU	Central Processing Unit	46
Cr	“Difference to red”	58
CTC	Common Test Conditions	19, 84

CTU	Coding Tree Unit	19, 62
CU	Coding Unit	62
DC	Discrete Current	91
DC®	Synopsys® Design Compiler	14, 152
DCT	Discrete Cosine Transform	19, 59
DFT	Discrete Fourier Transform	61
DPCM	Differential Pulse-Code Modulation	59
DRAM	Dynamic Random Access Memory	162
DSP	Digital Signal Processing	47
EMSEA	Extended Multilevel Successive Elimination Algorithm	128
ESEA	Extended Successive Elimination Algorithm	128
EVC	Essential Video Coding	63
FBMA	Fullsearch Block Matching Algorithm	20, 67
FFT	Fast Fourier Transform	73
FGSE	Fine Granularity Successive Elimination	128
FHD	Full HD	14, 58
FHT	Fast Hadamard Transform	20, 23, 73
FME	Fractional Motion Estimation	11, 13, 17, 19–21, 23, 35, 50
FOVS	First-Order Vertical Samples	76
FPGA	Field-Programmable Gate Array	140
fps	frames per second	41
FR	Full Reference	239
FSM	Finite State Machine	20, 149
FUHD	Full Ultra HD	42
GOP	Group of Pictures	62

MOS Metal-Oxide-Silicon 46

MPEG Moving Pictures Experts Group 43

MSB Most Significant Bit 178

MSE Mean Squared Error 84

MSEA Multi-level Successive Elimination Algorithm 19, 120

MV Motion Vector 66

MVP Motion Vector Prediction 67

NR No Reference 240

P Predicted 66

PDE Partial Distortion Elimination 11, 15, 17, 20, 101, 102

PMD Portable Mobile Device 47

PMV Predicted Motion Vector 67

POC Picture Order Count 199

PSAD Partial SAD 129

PSATD Partial SATD 20, 131

PSNR Peak Signal-to-Noise Ratio 14, 84

PT® Synopsys® PrimeTime 14, 198

PU Prediction Unit 19, 62

QP Quantization Parameter 19, 20, 61

RA Random Access 23, 86, 87

RD Rate-Distortion 14, 40

RDO Rate-Distortion Optimization 19, 66

RMD Rough Mode Decision 74

RR Reduced Reference 239

RTL Register Transfer Level 139

SAD Sum of Absolute Differences 11, 14, 17, 19, 20, 23, 35, 50

SAIF Switching Activity Interchange Format 154

SATD Sum of Absolute Transformed Differences 11, 13, 17, 19, 20, 23

SDF Standard Delay Format 154

SEA Successive Elimination Algorithm 11, 15, 17, 19, 21, 23, 24, 101, 102

SI Spatial Information 19, 88

SIMD Simple Instruction Multiple Data 46

SOC System on Chip 23, 47

SOVS Second-Order Vertical Samples 77

SRAM Static Random Access Memory 140

SS Sums and Shifts 20, 166

SSD Sum of Squared Differences 102

SSE Sum of Squared Errors 69

TB Transpose Buffer 20, 73, 102

TC Transform Coding 59

TE Transform Exempted 139

TI Temporal Information 19, 88

TSMC Taiwan Semiconductor Manufacturing Company Limited 152

UHD Ultra High Definition 14, 42

VBSME Variable Block Size Motion Estimation 85

VCS® Synopsys® Verilog Compiler Simulator 153

VLSI Very-large-scale Integration 44

VR Virtual Reality 39

VVC Versatile Video Coding 40

Y Luma 58

NOTATION

a Lower case italic character is a scalar

$f(\ldots)$ is a function that returns an scalar

$\mathbf{A}$ Upper case bold character is a matrix

$\mathbf{A}_{i,j}$ is an element in the i^{th} row and j^{th} column

$\mathbf{F}(\ldots)$ is a function that returns a matrix

$\mathbf{A}^{\text{name}}$ is a named matrix

$\mathbf{A}_{m\times n}$ is an $m\times n$ sized matrix

$\mathbf{A}_{m}$ is an $m\times m$ sized matrix

$\mathbf{A}_{m\times n;m'\times n'}$ is an $m\times n$ sized matrix with $p\times q$ partitions where $p = m/m'$ and $q = n/n'$.

$\mathbf{A}_{m;m'}$ is an $m\times m$ sized matrix with $p\times p$ partitions where $p = m/m'$. For instance, an 8×8 matrix, partitioned in 2×2 submatrices, each with size 4×4:

$$\mathbf{A}_{8;4}=\left[\begin{array}{ccc|ccc} \mathbf{A}_{1,1} & \cdots & \mathbf{A}_{1,4} & \mathbf{A}_{1,5} & \cdots & \mathbf{A}_{1,8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{4,1} & \cdots & \mathbf{A}_{4,4} & \mathbf{A}_{4,5} & \cdots & \mathbf{A}_{4,8} \\ \hline \mathbf{A}_{5,1} & \cdots & \mathbf{A}_{5,4} & \mathbf{A}_{5,5} & \cdots & \mathbf{A}_{5,8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{8,1} & \cdots & \mathbf{A}_{8,4} & \mathbf{A}_{8,5} & \cdots & \mathbf{A}_{8,8} \end{array}\right]=\left[\begin{array}{c|c} \mathbf{A}_{;1,1} & \mathbf{A}_{;1,2} \\ \hline \mathbf{A}_{;2,1} & \mathbf{A}_{;2,2} \end{array}\right]$$

A Upper case italic character is a set

$\overrightarrow{v}$ Lower case character with an arrow above is a vector

`a` Lower case character with typewriter font represents an RTL signal

`A` Upper case character with typewriter font represents an RTL state

LIST OF SYMBOLS

qp Quantization Parameter Value . . . . . 69, *Acronyms:* QP

S Set of candidate blocks 64

σ Standard deviation 89

$\mathbf{T}$ Transform 60

$\mathbf{T}^{-1}$ Inverse Transform 62

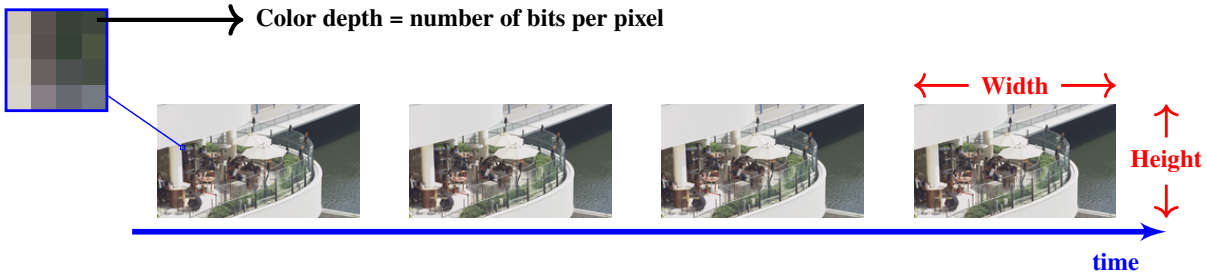
td a scalar value from the transformed differences matrix; 115

CONTENTS

1 INTRODUCTION

Once restricted to science fiction realms, applications like Virtual Reality (VR) and holography are now at our reach. According to Chou (2018), "holograms are the next video." Holograms may be represented by means of volumetric media, such as light fields[1] and point clouds[2] (CHOU, 2018). Notwithstanding, both imaging types can be mapped to 2-Dimensions (2D) video representations, like the one presented in Figure 1, to be compressed. Therefore, the compression[3] advancements needed to allow the adoption of those technologies may rely on the development of ever more efficient 2D video standards. For that reason, this thesis focuses on 2D video compression.

Figure 1 – Example of a 2D video, which is a sequence of 2D (width × height pixels) frames (images) through time.



Source: the author. Frames from "BQTerrace" sequence (BOSSEN, 2012).

1.1 A GAME OF TRADE-OFFS

Even for today's applications, it is always desirable to deliver maximum video quality with minimum data bitrates (LIN et al., 2018; CHEN, C. et al., 2018). It is of no surprise that both creators and viewers (consumers) of video have ever-increasing expectations for the perceived quality (WANG; INGUVA; ADSUMILLI, 2019). However, in many cases the bandwidth limitations put the perceptual visual quality in check because of the aggressiveness of bitrate reduction. Therefore, video compression standards must provide an acceptable trade-off between:

1. **Rate** (*bits*): the resulting bitrate, i.e., number of bits per pixel in the encoded video, which determines how much it compresses;

2. **Distortion**: a measure of errors in the frames that were introduced by lossy compression;

[1] For more information on this type of imaging, please refer to Ihrke, Restrepo, and Mignard-Debise (2016) and Wu, Masia, et al. (2017).

[2] For more information on this type of imaging, please refer to Schwarz, Preda, et al. (2019).

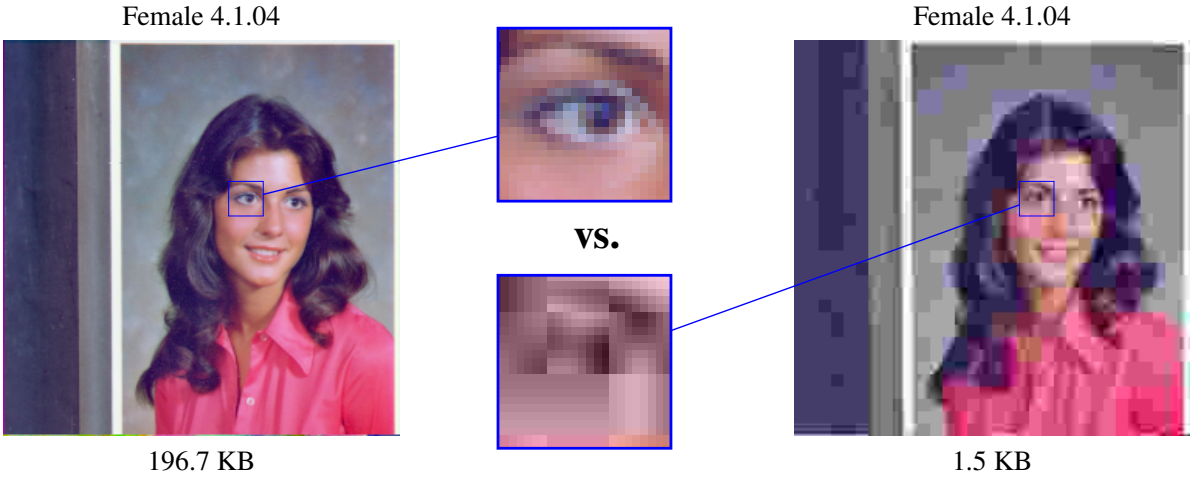
[3] In this thesis, "compression is the process of compacting data into a smaller number of bits" (RICHARDSON, 2003, p. 27).

Figure 2 illustrates such Rate-Distortion (RD) dichotomy. The RD trade-off achieved by a Coder/decoder (CODEC) is referred to as its **coding efficiency**[4]. Thus, a CODEC is said to be more coding-efficient than another when:

1. It keeps the same bitrate while improves quality;

2. It keeps the same quality while reduces bitrate;

Figure 2 – Example of the bitrate vs. quality (RD) trade-off. The lossless compression of static and dynamic images achieves only a modest ratio (RICHARDSON, 2003, p. 3). With the ever-increasing resolutions, the required compression ratios are also ever higher and thus lossy compression must be applied. Therefore, a loss in visual quality is expected when reducing too much the bitrates. In this example, the quality drop may be easily[5]perceived in the eye zoom but also the remaining of the image, such as in the wall.



Source: the author. Image "Female" from (USC, 2019), using "ImageMagick 6.9.7-4 Q16 x86_64 20170114" to convert the original `tiff` to `jpeg` with parameter `-quality 5` .

Although the evolution of new video coding standards like the Advanced Video Coding (AVC) (ITU-T, 2009), the High Efficiency Video Coding (HEVC) (ITU-T, 2013), and the Versatile Video Coding (VVC) continuously improves coding efficiency (WANG; INGUVA; ADSUMILLI, 2019), the current market for video applications still produces a massive amount of data.

Digital video content is the lion's share of today's Internet traffic: in 2017 videos were responsible for as much as 75% from the 77 Exabytes[6]/month of consumer traffic (CISCO, 2018). Cisco (2018) also predicts that this share may reach 82% by 2022 when about 5 million years[7]

[4] An objective model to analyze coding efficiency, called Bjøntegaard Delta (BD) model (BJØNTEGAARD, 2001, 2008), is presented inSection B.1 from Appendix B.

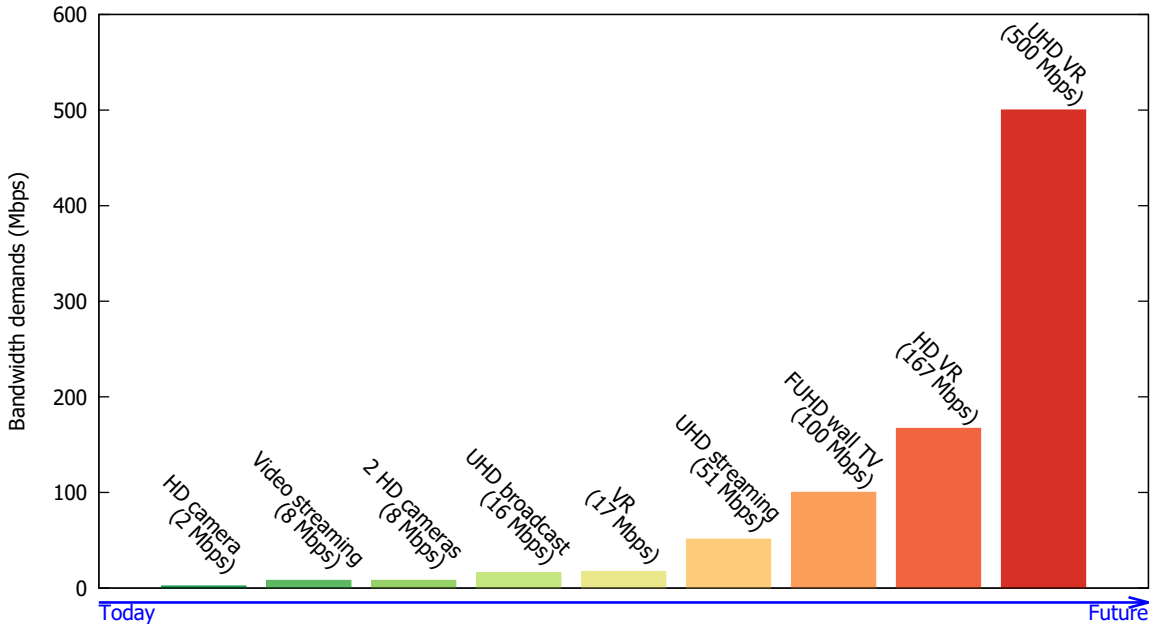
[5] Depending on the pdf visualizer, the poor quality of the rightmost image may partially be masked by embedded filter tools that makes the image smooth.

[6] 1 Exabyte (EB) = 10^{18} bytes.

[7] For comparison, the human-chimpanzee speciation is estimated to have occurred about 6.3 million years ago (PATTERSON; RICHTER, et al., 2006), whereas the universe is estimated to be between 10 to 14 billion years old (LIDDLE, 2015, p. 62; SERGE PARNOVSKY, 2018, p. 27).

of video content will cross the Internet every month, corresponding to about 240 EB/month. Although today's video applications have such enormous demand, in the future (even after 2022) the demand is expected to be even higher because of the emerging video technologies, such as VR. Figure 3 shows the expected bandwidth requirements of a few current and future video applications.

Figure 3 – Bandwidth demands of current and future video coding applications.



Source: the author. Data from Cisco (2018).

To understand why video has such high data demands (RICHARDSON, 2002; BOVIK, 2005, p. 12), lets us first analyze its needs before compression. Bovik (2005, p. 12) points out that the data requirements in visual signals increase geometrically with the dimensionality of the data. As shown in Figure 1, a video is a sequence of frames that are quickly captured and then further presented, creating the illusion of movement (RICHARDSON, 2002, p. 8). The total amount of data to represent a digital video before compression is linearly proportional to its color depth[8], to its image spatial resolution (its width times its height, shown in Figure 1), and to the video temporal resolution, which is the number of frames per second (fps). Equation 1.1 shows how to compute the bitrate of a video without compression, called "raw" (RICHARDSON, 2003, p. 3), in bits/s.

$$\text{bitrate}_{\text{raw}} = \text{color depth} \times \underbrace{\text{width} \times \text{height}}_{\text{Frame Resolution}} \times \text{frame rate} \tag{1.1}$$

Besides the bitrate, the space required to store a raw digital video also depends on the

[8] The number of bits required to represent a picture element, i.e, a pixel. See Figure 1.

video sample duration, as shown in Equation 1.2:

$$\text{space}_{\text{raw}} = \text{bitrate}_{\text{raw}} \times \text{sample duration} \tag{1.2}$$

Table 1 illustrates the required raw data rate to transmit Blu-ray video resolutions, as well as the required space for its storage. From the bitrate values shown in Table 1, it is possible to notice that an efficient compression method must be employed to achieve the bitrates depicted in Figure 3. Supposing the High Definition (HD) Camera from Figure 3 captures a video using the format HD 720 24p (first data row in Table 1) the compression rate must be of about 271:1. An Ultra High Definition (UHD) Broadcast, with a raw bitrate of about 5971.97 Mbps, is expected to use only 16 Mbps (Figure 3) and thus the compression ratio increases to about 373:1.

Table 1 – Bitrate and storage space for Blu-ray video formats without compression ("raw"). The last three lines were included to illustrate the use of higher spatial resolutions, like UHD and Full Ultra HD (FUHD). All estimates assume using 24 bits per pixel (bpp), i.e., 8 bits for each color channel.

Format	**resolution**	**frame rate**	**bitrate$_{\text{raw}}$**	**space$_{\text{raw}}$ (90 min)**
HD 720 24p	1280×720	24fps	0.53 Gbps	333.71 GiB
HD 720 50p	1280×720	50fps	1.11 Gbps	695.23 GiB
HD 1080 24p	1920×1080	24fps	1.19 Gbps	750.85 GiB
4K UHD 24p	3840×2160	24/s	4.78 Gbps	2.93 TiB
8K FUHD 24p	7680×4320	24/s	17.8 Gbps	11.73 TiB
8K FUHD 50p	7680×4320	50/s	39.8 Gbps	24.44 TiB

Source: adapted from (BENNETT, 2011).

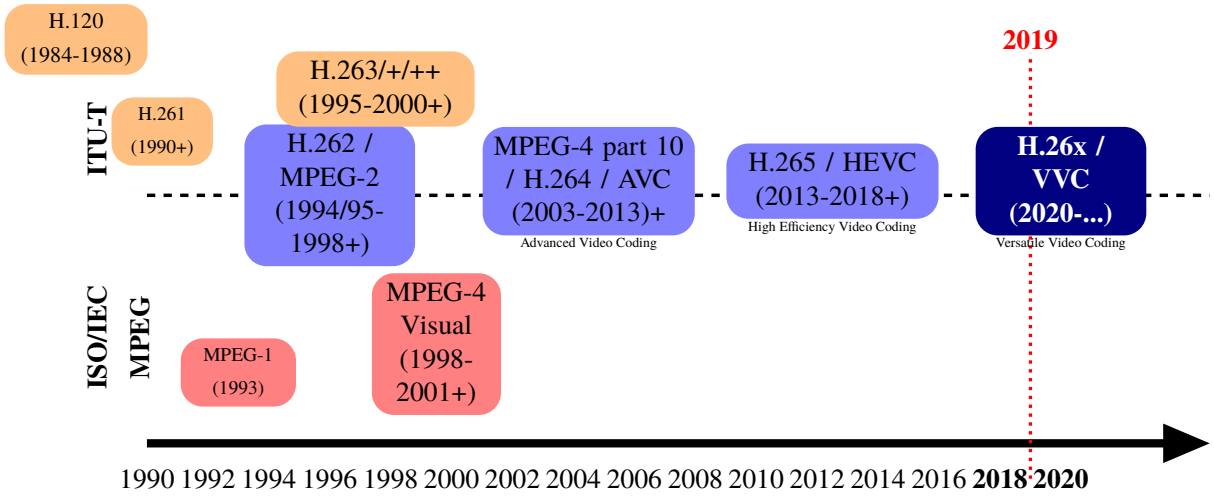
As the need for video compression becomes evident, one needs to exploit video redundancies[9] to achieve the required compression (RICHARDSON, 2003) within acceptable levels (BOVIK, 2005, p. 13). Such redundancy exploitation is one of the roles of video coding standards which, as mentioned, is to deal with the **bitrate vs. quality tradeoff** intrinsic to video compression. The other role of a video coding standard is to keep the compatibility among video decoding applications and devices. Therefore, the format by which videos are encoded, i.e., their bitstreams, must be standardized. In other words, the standard only defines the syntax and decoding tools (SULLIVAN, 2005, p. 3): the standard *per se* does not define the encoding tools.

The International Telecommunication Union – Telecommunication Standardization Sector (ITU-T), the International Standardization Organization (ISO) and the International Electrotechnical Commission (IEC) are three sister organizations in charge of defining new video

[9] The presence of redundancy in an information-carrying signal usually means that it can be represented in a more compact way (RICHARDSON, 2003, p. 3, 2002, p. 29), i.e., it can be compressed. For instance, pixels close together (in a frame or consecutive frames) tend to contain similar (redundant) data. To exploit such kind of redundancy, encoders rely upon prediction tools, as presented in Chapter 2.

coding standards. The extent of those standards is such, that according to Chiariglione (2019, p. 47) the total amount paid for Moving Pictures Experts Group (MPEG) patents is probably in excess of 1 billion USD/year. The evolution of these standards over time, depicted in Figure 4, is driven mainly by the adoption of increasing spatio-temporal resolutions (TROCHIMIUK, 2015). After all, as per the Jevons paradox[10], the improvements in video coding efficiency results in a broader adoption of video coding applications, thus increasing demand and requiring even higher coding efficiency. Richardson (2002, p. 292) mentions that "As digital video increases its share of the market, consumer demands for higher-quality, richer multimedia services will continue to increase." Therefore, the coding efficiency must continue to increase to cope with the larger compression ratios continually needed even at the expectancy of broader bandwidth (Figure 3).

Figure 4 – ITU-T, ISO and IEC international video coding standards over time.



Source: adapted from Sullivan (2005, p. 2) and Ohm and Sullivan (2018, p. 6).

Therefore, during standardization, a significant increase in the coding efficiency is desired to justify the adoption of the new standard (ISO/IEC, 2015). All standards listed in Figure 4 follow the same hybrid model (RICHARDSON, 2003, p. 72) that performs block-based prediction and transform (SULLIVAN; OHM, et al., 2012), which is presented in Chapter 2; they differ mainly by their final bitstream. The higher coding efficiency arises by the successive adoption of new coding tools and structures.

Much like Moore's law[11], there is also a general observed rule for the coding efficiency of consecutive video standards (KARWOWSKI et al., 2017): "About each 9 years we have a new video compression generation that provides halved bitrates" (DOMAŃSKI, 2015). For instance, the H.264/AVC (ITU-T, 2009) allowed a bitrate reduction of about 50% for the same quality with respect to its predecessors, like MPEG-2 (KAMACI; ALTUNBASAK, 2003, p. 347) and

[10] Such paradox, from 1865 by William Stanley Jevons regarding the usage of coal, states that an efficiency increase in the use of a resource tends to increased its use rather than reduce the demands of such resource (POLIMENI, 2012, p. x; CLEVELAND; MORRIS, 2009, p. 278). It is also known as "rebound effect" (MISSEMER, 2012).

[11] Moore's law observes that the number of transistor on a single chip doubles roughly every one (MOORE et al., 1965, p. 115) or two years (MOORE et al., 1975, p. 13) at constant cost.

MPEG-4 part 2 (OSTERMANN et al., 2004, p. 26). Analogously, HEVC (ITU-T, 2013) presents ~50% lower bitrates relative to H.264/AVC (SULLIVAN; OHM, et al., 2012) while keeping equivalent quality. However, such an increase in coding efficiency brought by HEVC over AVC was only possible at the cost of a substantial increase in the computational complexity[12]. The same increase in complexity was observed when comparing AVC to MPEG-2 (KALVA; FURHT, 2005, p. 513).

Indeed, the increase in computational complexity is characteristic of new encoder tools to achieve improved coding efficiency. For instance, an AVC encoder is about ten times more complex than one for the MPEG-4 part 2 (OSTERMANN et al., 2004, p. 26)[13]. The many differences between the AVC reference software, called Joint Model (JM), and the HEVC Model (HM), make comparisons inaccurate. Even so, considering similar configurations, the latter is almost four times slower than the former (BOSSEN et al., 2012; FRÖJDH; NORKIN; SJÖBERG, 2013). Furthermore, enabling every encoding tool available in HM increases its complexity by around 500% (CORREA et al., 2012). The estimate is that HEVC is 2-10× more complex than AVC (GROIS et al., 2013; KIM; RHEE, et al., 2017). Figure 5 shows projections of the increased coding efficiency and complexity throughout the years as new ITU-T/ISO/IEC standards are released. As can be seen, there is also a trend in the complexity increase: each standard is about 10× more complex than its predecessor.

The law of diminishing returns can be applied to this context: after achieving a given coding efficiency, the computational complexity will increase too much for ever-small gains. Nevertheless, the combination of such small gains often is what gives rise to new standards. While in the past the complexity has perhaps been put aside as a secondary factor, the ever-higher encoding complexity became a critical concern (ORTEGA, 2007). Hence, video coding became an even larger game of trade-offs (PATTERSON, 2012); besides the compromise in terms of RD, one must add to the list of trade-offs (from p. 39):

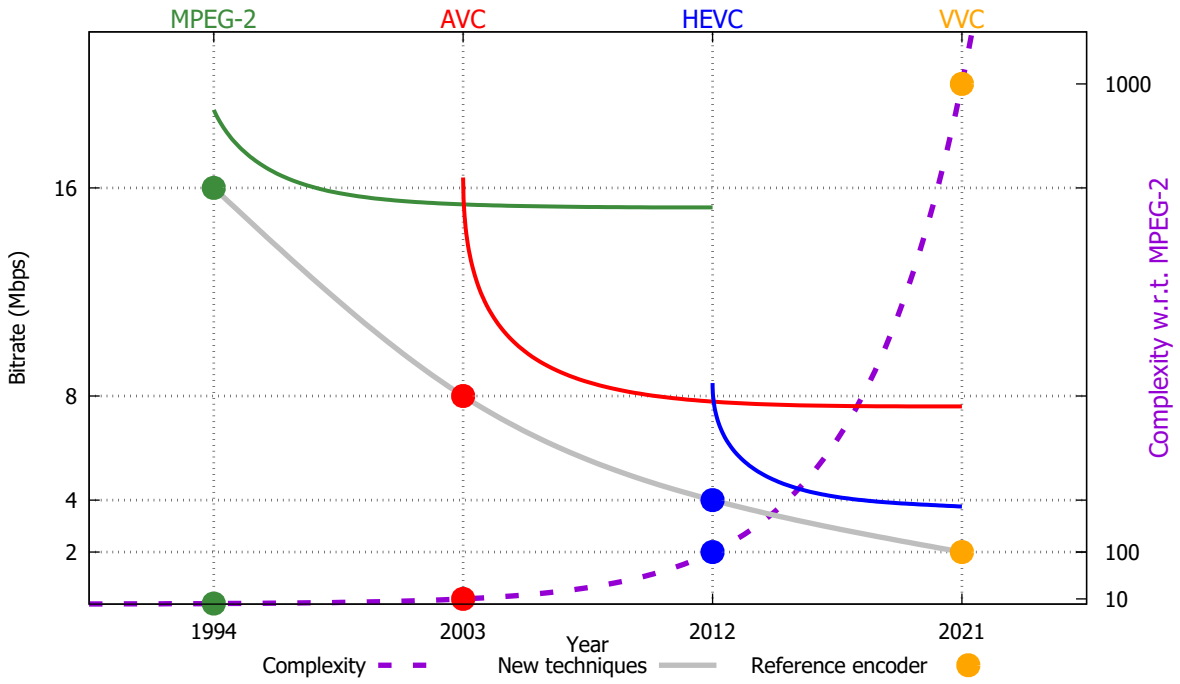
3. **Computational complexity**: the computational complexity of the tools used to encode/decode;

Most of the time, the large leap in complexity from one standard to another is somewhat compensated by Moore's law (MOORE et al., 1965). Kalva (2006, p. 90) concludes that the increased performance brought after Moore's law had begun to mitigate AVC complexity. For instance, when considering the performance increase of Very-large-scale Integration (VLSI) circuits, in 2004 AVC had about the same relative complexity that MPEG-2 had in 1994 (OSTERMANN et al., 2004, p. 26). In fact, the mass availability and emergence of video coding was due to a convergence of a number of areas (RICHARDSON, 2002, p. 1); As pointed out by Schafer and Sikora (1995, p. 907), both video coding standardization efforts and the fast

[12] In this thesis "computational complexity" is used as equivalent to "execution time", as it is common in the video coding community.

[13] The 10× increase is reported for the "Simple Profile". Comparing with AVC "Main Profile" with MPEG-4 part 2 "Simple Profile", Ostermann et al. (2004, p. 24) report a increased complexity larger than one order of magnitude.

Figure 5 – Approximate behavior of the coding efficiency (solid lines) and complexity (dashed line) increasings of ITU-T/ISO/IEC standards over time. The coding efficiency is presented in terms of bitrate for the same quality level, whereas the complexity is an estimate complexity increase with respect to the complexity of the first standard presented in this figure (MPEG-2). Each circle represent a given standard coding efficiency (i.e., from their reference implementation) near the time they are finished. The curves with the same color than a circle represent the improving of the coding efficiency of that standard along time considering commercial implementations. The actual figures are impossible to obtain since all figures depend on a vast set of parameters. Neither assessing the complexity of new standards is a straightforward task because it depends on the platform characteristics (OSTERMANN et al., 2004, p. 22). Moreover, video content also plays a role in coding efficiency (SOLE et al., 2018), as well as in the computational complexity of the encoder.

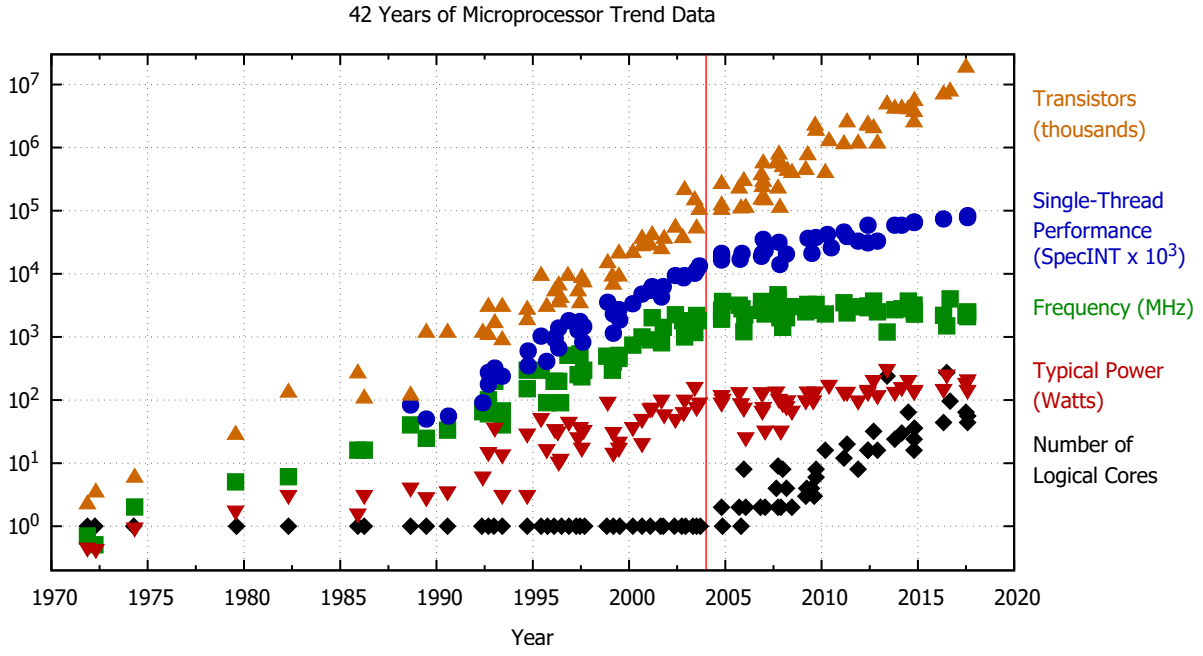


Source: adapted from Karwowski et al. (2017, p. 7).

advances in microelectronics were together the main drivers for the success in the massive adoption of video content in our daily life. Such performance increases of VLSI circuits are depicted in Figure 6, along with other relevant data on VLSI circuits through time.

Therefore, the increase in complexity becomes more acceptable as time goes by (OSTERMANN et al., 2004, p. 21), providing that the supporting technologies improve accordingly. A dependency on the performance growth exists in several areas, like anticipating the performance improvements to create computationally intensive applications even before the required performance is available (FULLER; MILLETT, 2011, p. 32). Such way to cope with increased complexity by relying upon improved hardware is also present in the software development industry, giving rise to numerous "rules" and "laws", for instance: "Software is getting slower more rapidly than hardware becomes faster" (WIRTH, 1995, p. 64); "Software efficiency halves every 18 months, compensating for Moore's Law" (May's law) (EADLINE, 2011). All of these can also be related to the aforementioned Jevons paradox. In this sense, solely relying

Figure 6 – 42 Years of Microprocessor Trend Data. Notice that the growth is exponential, following Moore's law (see note 11). Also, the vertical red line by 2004 marks the end clock frequency increases due to the power wall imposed by the end of Dennard scaling (see note 14). From then on, the number of logical cores increased, keeping Moore's law. However, although the single-thread performance continued to increase, such increase have slowed down. In 2010 the performance gap was of about a factor of 10, while by 2020 such gap is expected to have a factor of 1000 (FULLER; MILLETT, 2011, p. 33).



Source: `https://github.com/karlrupp/microprocessor-trend-data`.

on the exponential growth in performance due to transistor technology improvements may be insufficient to deal with ever-larger complexity.

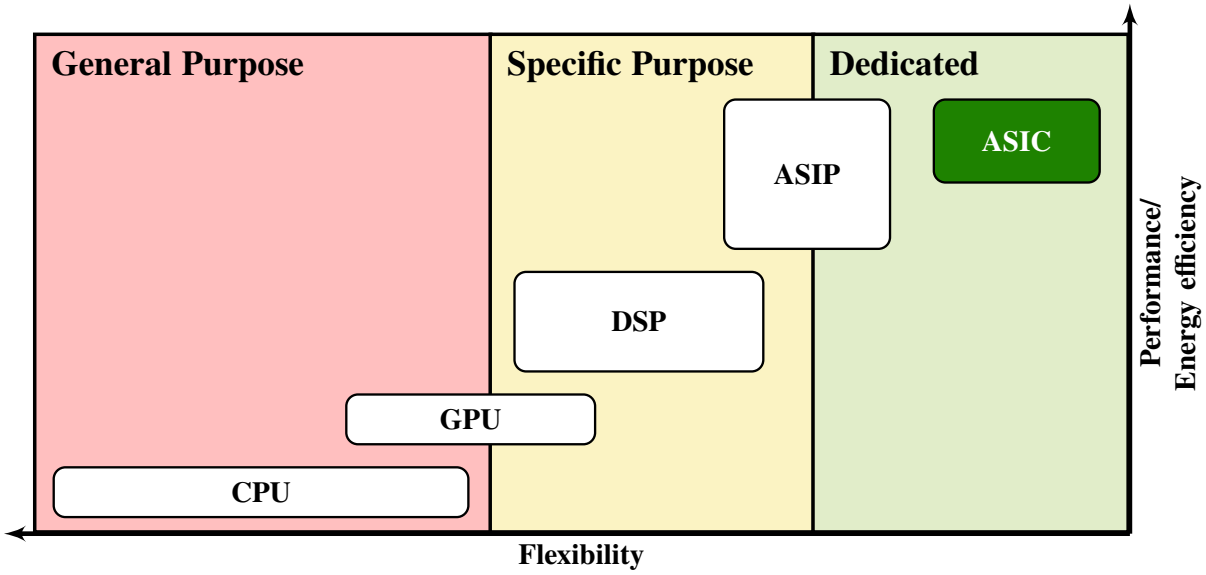
Particularly, every exponential growth must come to an end (COLWELL, 2013, p. 4; TAYLOR, 2014, p. 73), and so does Moore's law. Although transistor density may continue to increase following Moore's law (Figure 6), there is expected to be comparatively little improvement in their speed and energy consumption (BORKAR; CHIEN, 2011, p. 77). It is possible to see in Figure 6 a slowing down in core performance gains from 2004 on, imposed by power limits (FULLER; MILLETT, 2011, p. 32) due to the end of Dennard scaling[14].

Increasing parallelism by providing multiple cores and improved instruction sets was the choice to keep Moore's law going. For instance, some Intel® processors provide Simple Instruction Multiple Data (SIMD) instructions for distortion metrics used during encoding (PATTERSON; HENNESSY, 2017, p. 326; COCKSHOTT; RENFREW, 2013, p. 87). However, even such a level of parallelism has its limitations, also imposed by data dependencies. Another way to avoid this slowing down in performance is by transitioning from generic processors (e.g., Central Processing Units (CPUs)) to more specialized ones (also called accelerators) (AITKEN,

[14] As resumed by Ning (2007, p. 27), it is a theory for scaling down the physical dimensions of Metal-Oxide-Silicon (MOS) transistors presented by Dennard et al. (1974). Briefly, it shows that it is possible to scale the sizes and voltages of the transistors by the same factor (HASIB, 2018, p. 11), while keeping the power constant.

2019). In this aspect, there as another trade-off between flexibility and performance/energy efficiency when considering different hardware platforms, illustrated in Figure 7. The least flexible ones, the accelerators, are processors tailored to specific computations such as machine learning, cryptocurrency and, of course, video coding.

Figure 7 – Flexibility vs. performance/energy efficiency trade-off at different hardware platforms. Although CPUs provide the highest flexibility among the presented platforms, they also have the poorest performance due to their generality. Graphics Processing Units (GPUs) have increased performance by exploiting parallelism at a higher level than the CPUs. Digital Signal Processings (DSPs) processors exploit the intrinsic parallelism in several digital signal processing algorithms but lose flexibility by having such a specific purpose. Both Application Specific Instruction set Processors (ASIPs) and Application Specific Integrated Circuits (ASICs) present the highest efficiency, but with decreased flexibility. The former is less flexible than the latter, as the latter can still be used by a broader set of applications (although limited within its purpose).



Source: adapted from (MOONS; VERHELST, 2017, p. 905).

There is a substantial market share that already adopts video coding hardware accelerators: the so-called Portable Mobile Devices (PMDs), *e.g.*, smartphones, another marvel of the contemporary days enabled by microelectronics (MOORE et al., 1965, p. 114). Although there is no publicly available data on the size of VLSI industry that develops Intellectual Property (IP) cores[15] implementing MPEG standards (CHIARIGLIONE, 2019), some mobile processors are known to integrate them. Table 2 presents some embedded SOCs that implement video CODECs. The 3rd generation of Snapdragon 800 (810) processors, for instance, are capable of encoding UHD videos at 30fps by using a HEVC hardware accelerator that supports a color depth of up to 14 bits per channel.

[15] IP cores are predesigned and preverified hardware accelerators for complex functions that can be embedded in larger designs (REORDA; PENG; VIOLANTE, 2005, p. 57; JERVAN et al., 2005, p. 121; MOHAMED, 2015, p. 2). Designers of a System on Chip (SOC) can often customize and connect IP cores from different sources (SMITH; PARR, 2004, p. 286; REORDA; PENG; VIOLANTE, 2005, p. 57), thus reducing their costs and time-to-market (KAMAT et al., 2011, p. 93).

Table 2 – Some SOCs with embedded hardware CODEC accelerators.

Manufacturer (Model)	Qualcomm (Snapdragon)	Samsung (Exynos 9820)	Samsung (Exynos 990)
CODECs	AVC, HEVC, VP9	AVC, HEVC, VP9	AVC, HEVC, VP9
Technology	10nm FinFET	Samsung 10nm LPP	7nm EUV
Year	2016	2017	2019

Source: Qualcomm (2016, p. 2), Samsung (2016, p. 3) and Samsung (2019, p. 6).

As the cost of image acquisition and displays have been dropping drastically (ORTEGA, 2007), video capable devices such as PMDs are becoming pervasive. Commercially, the PMDs were so successful that they are now an elemental part of everyday life for billions of people (CONTI et al., 2018, p. 2658). To be more precise, according to GSMA (2019, p. 6), in 2017 about 5 billion people worldwide had at least one smartphone; such estimate grows from 61% up to 71% of the world population by 2025 (GSMA, 2019, p. 6). The camera quality (therefore the encoder quality) is a valuable asset when people are deciding which smartphone to purchase: 86% of US citizens participating in a poll (CONSULT, 2018, p. 72)[16] considered the camera quality "very important" (51%) or at least "somewhat important" (35%). Consult (2018, p. 60) also shows that for 95% of the respondents the battery life is "very important" (74%) or at least "somewhat important" (21%), which makes this feature voted as the most important one in new smartphones.

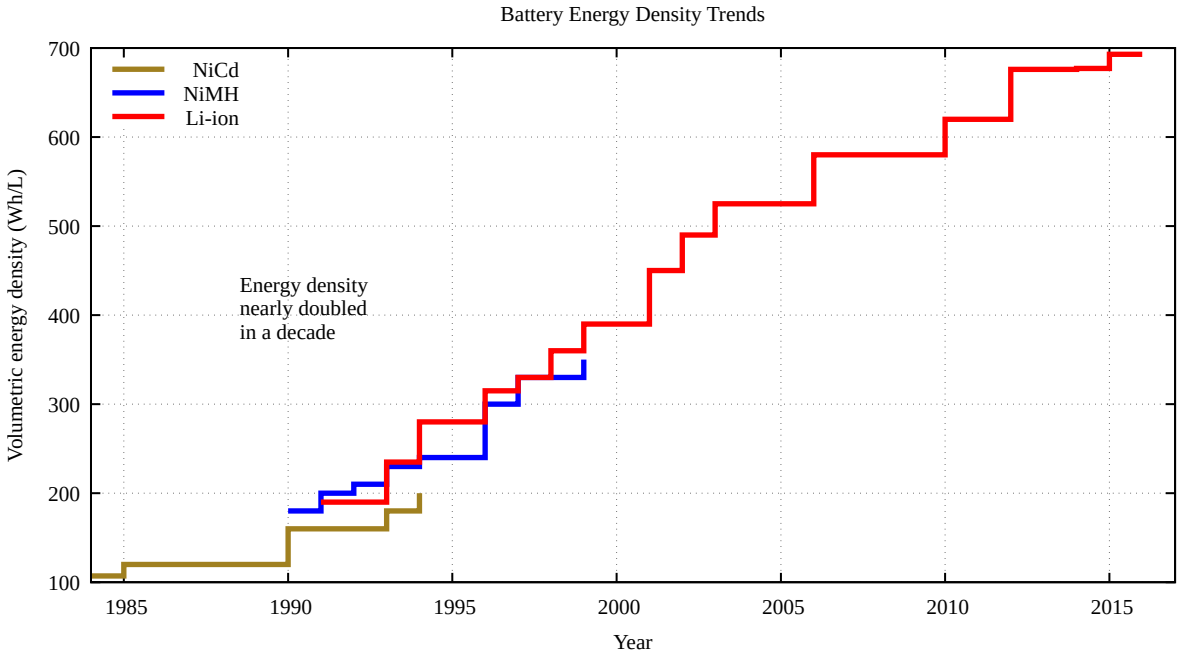
Lead-acid batteries have the largest market share by far (about 75%) (PILLOT, 2018, p. 4), in spite of a small energy density (from 25 to 50 Wh/Kg) (DIVYA; ØSTERGAARD, 2009, p. 514). It is also the oldest and most mature technology, being used for most power system applications (DIVYA; ØSTERGAARD, 2009, p. 513). However, PMDs have a high volumetric energy density requirement (PLACKE et al., 2017, p. 1953), i.e., their batteries must provide the largest possible amount of energy whilst having the most compact form (BAES et al., 2018, p. 8).

In this sense, the Lithium-ion (Li-ion) batteries offer high energy density in addition to having small size and low weights (DIVYA; ØSTERGAARD, 2009, p. 513). Therefore, Li-ion batteries have been widely adopted as energy sources for PMDs (A. LUNDGREN et al., 2017, p. 449). Also, amongst other battery technologies, Li-ion had the highest growth and investments since their introduction in the '90s (PLACKE et al., 2017, p. 1939), presenting a Compound Annual Growth Rate (CAGR) of +26% from 2010-2017 (PILLOT, 2018, p. 4). Nevertheless, battery energy has the slowest trend in mobile computing (PARADISO; STARNER, 2005, p. 18). Figure 8 presents such trend and although steady advances in energy density are visible (about twice in 10 years), such growth has remained relatively stagnant (LIAO; SODANO, 2008, p. 1) when compared to the advances in Figure 6. Although there will be continuous improvements to Li-ion battery technology for the upcoming years, there is also an agreement that the used materials will soon reach their limits (PLACKE et al., 2017, p. 1961).

Therefore, batteries impose an even lower power wall for PMDs and thus mobile/embedded

[16] Poll conducted in November, 2018 among a sample of 2201 adults from the United States. The results are expected to have a margin of error of plus or minus two percentage points (CONSULT, 2018, p. 1).

Figure 8 – Li-ion batteries had a meaningful improvement in energy density since their introduction and dominated the market for PMDs (PLACKE et al., 2017, p. 1939). Yet, they are soon reaching their limits in energy density so that the expected energy density limit is of about 1.2 KWh/L (PLACKE et al., 2017, p. 1961) to 3 KWh/L (WHITTINGHAM, 2014, p. 11439). Also, notice that while Figure 6 trends have exponential growth (logarithm scale on y-axis), here the trends grow linearly.



Source: adapted from Thangavelu and Chau (2013, p. 15), Straubel (2015, p. 4), and Placke et al. (2017, p. 7).

computing drives energy efficiency requirements (BORKAR; CHIEN, 2011, p. 70). In the context of video coding trade-offs, there are other conflicting effects in play (ZHIHAI HE et al., 2005, p. 645): improving coding efficiency may save energy by reducing transmission overheads over wireless networks, but achieving higher coding efficiency demands for more energy due to the increased complexity. Hence, the game of trade-offs has yet another key player:

4. **Energy efficiency**: dependents not only on the computational complexity (time), but also on the power requirements of the chosen platform;

In the midst of a such game, there is a wide field for research that has been active for decades (RICHARDSON, 2003, p. 5; CHIARIGLIONE, 2019, p. 17, 55). Thus, before presenting this thesis contribution within this game, let us sum up what was presented so far.

1.1.1 Trade-offs summary

The ever-increasing resolutions and the rise of new video applications demand ever more coding-efficient video compression. Thus, several new tools have been proposed within new standards to improve compression, which increases the computational complexity of video

CODECs, mostly within the encoders. Moore's law is no longer sufficient to compensate for the complexity increase and thus cannot be expected anymore to be the sole enabler of those technologies. In addition, the increasing complexity is reflected upon video coding applications for PMDs, which demands high compression due to their limited storage and transmission capabilities. Moreover, video encoders targeted for PMDs must be as energy-efficient as possible to improve battery lifetime. Paradoxically, most methods to reduce the computational complexity end up decreasing the coding efficiency.

1.2 ENTER THIS THESIS: CONTRIBUTIONS

Given the presented game of trade-offs presented in the previous section, the main contribution of this thesis is **to improve energy efficiency without reducing coding efficiency** (i.e., a win-win situation). To achieve that, we have identified in related works some of the main culprits for the increased complexity within video encoders. They are the SATD and the Fractional Motion Estimation (FME)[17]. The former is a distortion metric similar to the Sum of Absolute Differences (SAD), but with expected higher coding efficiency (SAPONARA et al., 2003 apud OHM et al., 2012, p. 1675) and also higher complexity due to the extra transform step needed. The FME is a prediction tool within the encoder that may use the former as a distortion metric. Details on both SATD and FME, including their complexity analysis that endorses their choice as key to improve overall encoding energy efficiency, are given in Sections 2.3 and 2.4, respectively.

Most works aiming complexity reduction also end up reducing coding efficiency. However, a few of them are able to reduce complexity without affecting coding efficiency. To achieve that, they rely on reducing the number of operations required to compute the demanding distortion metrics during predictions, without affecting their accuracy. Therefore, the key to achieving the main contributions of this thesis was to exploit mathematical properties of the SATD when used during prediction. Some of the proposed techniques are improvements upon known methods used for the SAD, but aiming to render the transform within SATD less energy demanding. Also, some of properties can only be exploited if appropriate hardware organizations for SATD and FME are adopted.

Moreover, according to Chiang et al. (2017), video CODECs rely heavily on FME to achieve higher coding efficiency. Given the complexity of FME, there is a high demand within consumer applications to increase the energy efficiency of Motion Estimation (ME) as a whole (SINGH; AHAMED, 2018). Also, the most complex encoding tools of video coding should be off-loaded to dedicated hardware accelerators (VANNE et al., 2012). After all, as Figure 7 shows, the most energy-efficient hardware platforms are ASIC implementations. Thus, this work brings dedicated hardware designs for both SATD and FME. In fact, those hardware accelerators were designed to be as energy-efficient as possible. On top of that, we improved

[17] Details on both SATD and FME are given in Chapter 2, and so is the analysis of their complexity.

the architectures with the explored complexity reduction techniques.

Table 3 shows a summary of the main contributions achieved while pursuing the goals of this thesis. The totality of contributions is highlighted within each chapter. Moreover, Appendix D presents a list of peer-reviewed scientific papers published so far as a result of the present work.

Table 3 – Summary of the contributions of this thesis. The color codes are related to the organization of this thesis, as presented in Figure 9 (Section 1.6).

Contribution	**SATD**	**SATD+FME**	**FME**
Identification of complexity culprits	X		X
Coding efficiency analysis	X		X
Complexity reduction algorithms	X		X
Hardware exploration*	X	X	X
Joint complexity reduction		X	

*Hardware exploration includes the design and synthesis, but also the analysis in terms of area, power and energy.

Because the contributions of this thesis are not so much dependent on the standard we will avoid using standard dependent names. However, in some cases there are some normative steps in the decoder that will affect the way the encoder executes (and is implemented). In these cases, we will use the HEVC because it is the current stable (fully defined) state-of-the-art encoder.

1.3 HYPOTHESIS

Considering the need for energy- and coding-efficient video coding hardware accelerators, the basic hypothesis of this work can be stated as follows:

> **It is possible to efficiently apply techniques based on mathematical properties of both the SATD and the prediction tools to improve the energy efficiency of dedicated FME and SATD hardware architectures without reducing coding efficiency.**

1.4 GOALS AND RESEARCH QUESTIONS

The main goal of this thesis is to explore, propose, and evaluate algorithms that exploit mathematical properties of SATD when used within predictions to reduce complexity without trading coding efficiency off and through those algorithms enable energy-efficient video coding hardware accelerators. Additionally, to validate the presented hypothesis, another part of the main goal of this thesis consists in answering the following fundamental research questions:

RQ-1. Is it worthy, in terms of coding efficiency, to use SATD during inter predictions?

RQ-2. Which is the coding efficiency gain brought by the FME?

RQ-3. Is it possible to overcome the limitations imposed by the transform part of SATD and apply to it techniques that are used to reduce complexity without affecting coding efficiency when SAD is adopted as distortion metric?

RQ-4. If possible, do those techniques, applied when using SATD, provide improvements in terms of energy efficiency?

1.4.1 Specific goals

The following specific goals were devised to answer the research questions:

SG-1. Describe and evaluate the coding efficiency of SATD and FME within a state-of-the-art video codec;

SG-2. Propose, describe, demonstrate, and evaluate new algorithms that exploit SATD properties to reduce the average number of operations, but without reducing coding efficiency;

SG-3. Design and evaluate SATD and FME hardware architectures, aiming to improve their energy efficiency;

SG-4. Identify and exploit opportunities within the hardware architectures to improve energy efficiency, also without reducing coding efficiency;

SG-5. Interpret and explain the area, power, and energy results from the designed architectures in light of the adopted strategies (work method) and complexity reduction algorithms;

SG-6. Compare the achieved results with results reported by state-of-the-art works;

1.5 KNOWN LIMITATIONS

This work is not intended to be the only and one solution to achieve real-time and energy-efficient video coding. It is, however, intended to propose means to reduce encoding complexity, even by small amounts, but with no impact on coding efficiency. Therefore, the techniques proposed here can be used along with other, possibly lossy, techniques found in the literature, but without having a combined effect in coding efficiency. I.e., there is no risk in combining the proposed techniques with others in a way that the total coding efficiency loss may be worse than the sum of the combining parts. After all, some complexity reduction can be achieved at a small coding efficiency cost because the encoder can compensate for the imposed restrictions with other tools. However, combining several lossy techniques may render the encoders unable to compensate elsewhere.

1.6 ORGANIZATION OF THIS THESIS

This thesis is organized as follows. The present chapter provided an overall look on video coding research, from the need for coding-efficient compression to the needs of energy efficiency on battery-powered devices. Apart from this introduction and the conclusions, this work is divided into three parts. Figure 9 shows an overview of those parts and the chapters contained within each one of them.

Part I provides concepts about video coding and several definitions needed, along with a coding efficiency analysis that is intended to support the algorithm/design choices in the remaining of this thesis. Within such part, Chapter 2 introduces the notation used throughout this manuscript and lays down several needed definitions and some properties used in subsequent chapters. Also, in Chapter 2 the inner workings of a hybrid video CODEC are explained, pinpointing the scope of this thesis and bringing up the complexity problem. That chapter also holds details about the SATD computation and the FME. The mentioned coding efficiency analysis is presented in Chapter 3.

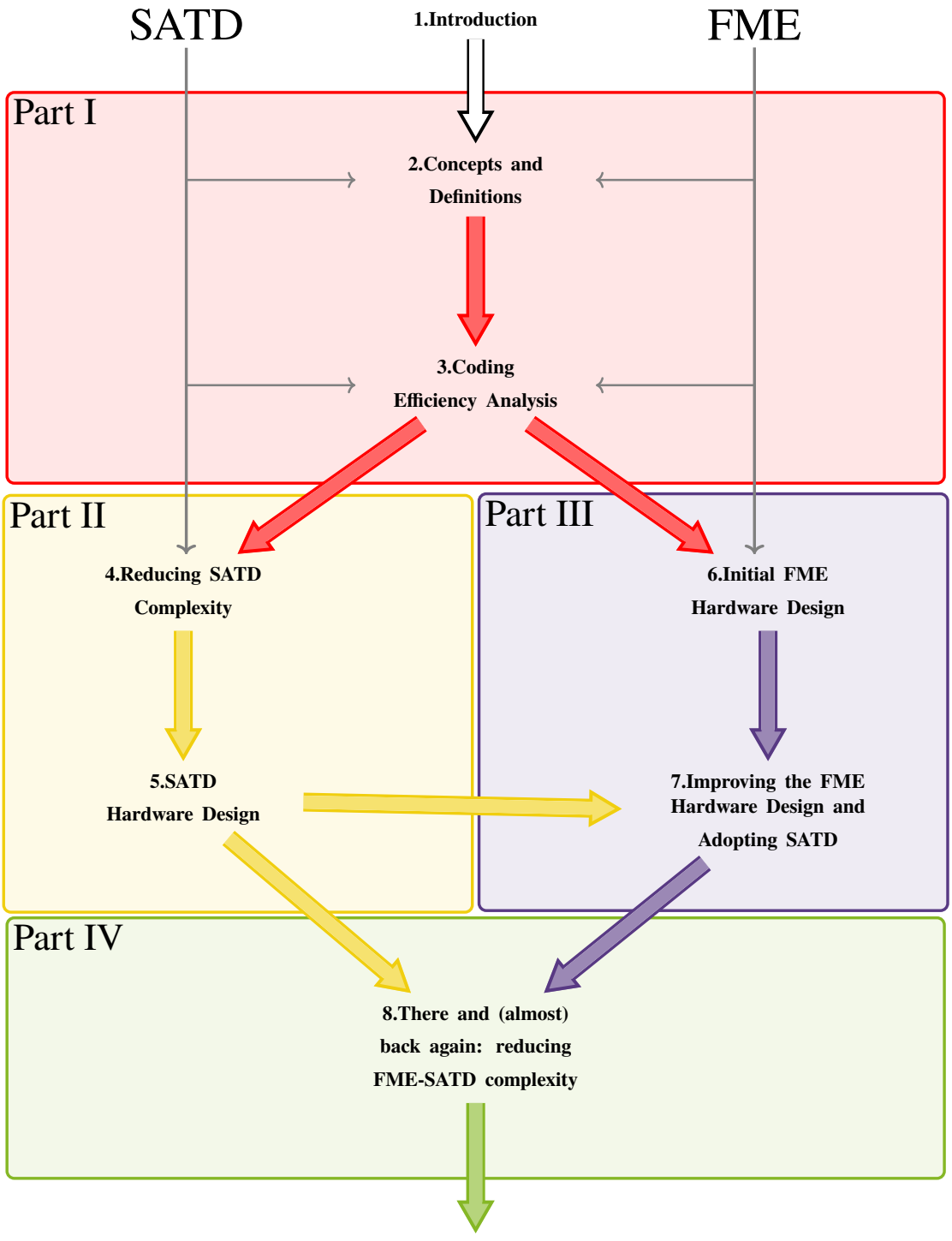
Part II presents the contributions of this work that are related to the SATD alone. In such part, Chapter 4 presents three techniques to reduce the complexity of SATD without reducing coding efficiency and their demonstration. Chapter 5, by its turn, presents SATD hardware architectures. There are a few of them devised to compute the SATD as energy-efficient as possible without using the proposed techniques, and the identification of architectures among those that are the most suitable to adopt techniques present in Chapter 4. Chapter 5 also holds one significant contribution of this thesis, that is a novel architecture for SATD that exploits jointly the hardware design and properties of SATD to reduce energy consumption.

In Part III the FME is brought up, both as a part of the encoding process that needs energy-efficient hardware architectures, as well as one encoding tool that uses SATD. Thus, Chapter 6 first presents a competitive hardware architecture in terms of coding and energy efficiency, while adopting the SAD instead of the SATD. In the same chapter, such architecture is improved by using knowledge of encoding choices. Then, the improved version of the FME architecture is further changed to accommodate the adoption of the SATD, which is the subject of Chapter 7. By doing so, Chapter 7 provides insights on how much more energy is demanded in FME by the adoption of SATD instead of SAD.

Part IV sets the convergence point for Parts II and III. Its unique chapter shows how the techniques from Chapter 4 can be adopted within the architecture from Chapter 7. In that sense, Chapter 8 provides results on the presented techniques in a practical use of SATD, thus showing to what extent the theory meets the practice.

Finally, Chapter 9 draws the conclusions of this work and brings up a few possible future works.

Figure 9 – Map of the parts and chapters of this thesis, showing the content flows and the direct relations between parts and chapters.



Source: the author.

Part I

Concepts and Coding Efficiency

2 CONCEPTS AND DEFINITIONS

This chapter introduces very briefly the notation[18] used throughout this manuscript. A more detailed explanation, containing partitioned matrices notation and useful definitions and properties is presented in Appendix A. Still in this chapter, we bring some more details about concepts that are only briefly mentioned in Chapter 1, along with other concepts which should be useful in the following chapters. In this sense, we tried to condense most of the definitions that are common to other chapters. Moreover, this chapter provides more details into the SATD and the FME, that are fundamental in this work. By the end of this chapter, the complexity of both is unveiled. Thus, this chapter provides reasoning for exploring these two parts of the encoding process, only touched superficially in Section 1.2.

In this thesis, we adopt the following notation. As so far, acronyms are represented in uppercase. When presenting hardware architectures, digital signals are marked as typeface lowercase (`signal`), whereas states are marked as typeface uppercase (`ST`). A set is represented in italic uppercase (S). Scalars are represented in italic lowercase (s). Vectors are represented in lowercase with an arrow ($\vec{v}$). Matrices are represented in bold uppercase ($\mathbf{A}$). When representing a matrix with related semantics, we put a name in the form ($\mathbf{A}^{\text{name}}$). For instance, an original frame (before being coded) can be represented in a matrix named as $\mathbf{F}^{\text{original}}$ or $\mathbf{F}^{\text{ori}}$, for short[19].

2.1 A BRIEF OVERVIEW ON VIDEO CODING

Video compression can be achieved by reducing the redundancies[20] within its data. Shi and Sun (2008, p. 4–20) classify video redundancies as:

1. **Psychovisual**: related to characteristics of the Human Visual System (HVS);
2. **Statistical**: related to the repetitions of symbols in the video; they are subdivided in:
   a) **Interpixel**: pixels within a frame or a sequence of frames are not statistically independent; such class can be further divided into:
      i. **Intra-frame**: spatial redudancies, i.e., the correlation existing in pixels within the same frame; inside each frame;
      ii. **Inter-frame**: temporal redudancies, i.e., the correlation existing in pixels within a sequence of frames; between frames close in time;
   b) **Coding**: the statistical redundancy associated with coding techniques, i.e., the entropy;

[18] For convenience, a condensed index of such notation is also provided in p. 31.
[19] The recurring named matrices are shown in the "List of Symbols" (page 33).
[20] C.f., Note 9.

Albeit using slightly different terms, the leaves of such classification falls within the same categories defined by Kalva (2006): perceptual (psychovisual), spatial (intra-frame), temporal (inter-frame), and statistical (coding). Corrêa et al. (2015, p. 8) classify the redundancies into three categories, spatial, temporal, and entropic. Although psychovisual redundancies directly considered in this classification of redundancies, they make a distinction between data redundancy and data irrelevance. Thus, the latter data category includes data that are not relevant to the HVS, and thus such data may be removed (CORRÊA et al., 2015, p. 8).

The goal in reducing redundancies is also to reduce video entropy, enabling the entropy encoder to generate almost optimal compressed codes. Given an alphabet[21] A, with n symbols and a string $\mathbf{a}$ over this alphabet such that their probability density is $\{p_1, p_2, ...p_n\}$, the entropy of this message $h(\mathbf{a})$, given in bits, can be defined as:

$$h(\mathbf{a}) = \sum_{i=1}^{n} \left(p_i \times log_2 \left(\frac{1}{p_i} \right) \right) \tag{2.1}$$

Considering a variable-length binary encoding of such message, its entropy imposes a lower bound on the total number of bits necessary to represent this message without loss of information. Therefore, reducing the entropy reduces this lower bound and allows for more efficient compression. The encoder part of a CODEC is thus a set of tools able to reduce redundancies as efficiently as possible conforming to a bitstream. By its turn, the decoder part is responsible for reconstructing the frames again, adding back the redundancies.

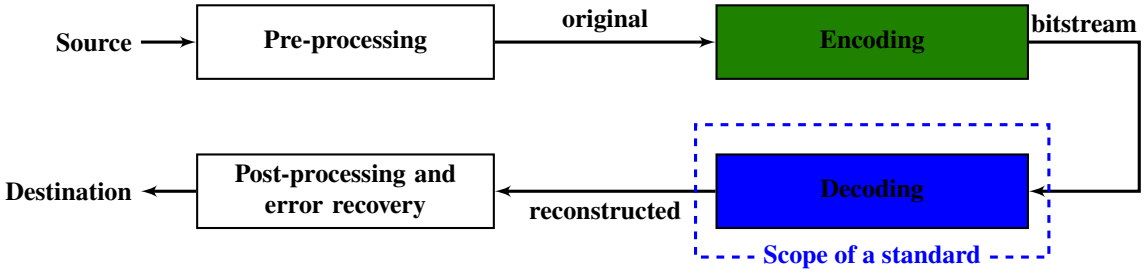
As introduced, a video coding standard (like AVC and HEVC) only defines the bitstream syntax and semantics and the operation of the decoder (WIEGAND; SULLIVAN, et al., 2003; STANKOWSKI et al., 2015). In this sense, the encoder tools that do not change the bitstream syntax and semantics are said to be non-normative. Thus, the encoder is granted to decide which tools to use and their configuration, constrained only by the desired compression/quality and by the adopted standard capabilities. This freedom in the encoder is essential to enable coding efficiency to improve within a standard[22] or to be trade off by a less complex implementation. Figure 10 illustrates the scope of a video coding standard within the source encoding/decoding part of a digital communication system.

Different from the standardization scope, the scope of this thesis is within the encoding part. The highest level structure within a video to be encoded is the frame, as illustrated in Figure 1. Video frames are composed of at least one color channel, represented by a matrix of samples (e.g., an original Full HD (FHD) frame may be denoted as $\mathbf{F}^{\text{ori}}{}_{1080\times1920}$).

In the YCbCr color space (ITU-T, 2011, 2008a, 2012), for instance, a frame has three channels: Luma (Y), "Difference to blue" (Cb) and "Difference to red" (Cr). In this luminance-chrominance format, the Y-channel contains brightness information of all color signals in the image (MONTAÑA, 2017, p. 192). Additionally, such color space has known **Psychovisual**

[21] A non-empty finite set whose members are called symbols of the alphabet (SIPSER, 2012, p. 13).
[22] This means that a standard has no guarantees on quality either (SULLIVAN, 2005, p. 3, 2014, p. 4).

Figure 10 – Block diagram of the source encoding/decoding part of a digital communication system, highlighting the scope of a video coding standard. Examples of source and destination in such context are video acquisition and display, respectively (WIEN, 2014a, p. 23–24). Pre- and post-processing include operations in the raw data, such as sub/resampling, color conversion/correction, and denoising. Wien (2014a, p. 23) includes a "transmission" box between encoder and decoder, emphasizing the channel encoder/decoding and the actual channel. Shi and Sun (2008, p. 34) provide a block diagram for a complete visual communication system.



Source: adapted from Wiegand, Sullivan, et al. (2003, p. 560) and Wien (2014a, p. 24).

redundancies because the HVS is more sensitive to luminance (Y)[23] than to chrominance (Cb and Cr) (SHI; SUN, 2008, p. 19). In this case, the spatial resolution of chroma channels may be reduced while Y-channel spatial resolution is kept (CUBITT, 2014, p. 247). Such a process is called chroma subsampling (POYNTON, 2012, p. 124).

In general, data representing characteristics almost unseen by the HVS, such as these psychovisual color redundancies, are removed and so they cannot be completely reconstructed. Therefore, such kind of compression is called **lossy**[24]. The key idea is to avoid introducing losses that are visually perceptible, thus the exploitation of characteristics of HVS. On the other hand, statistical redundancies can be reduced during encoding without introducing errors. Therefore, the encoded data can be completely restored to its original form and so this compression is said to be **lossless**[25].

Within statistical redundancies, the interpixel ones can be reduced by exploiting differential (predictive) coding (such as Differential Pulse-Code Modulation (DPCM)) or Transform Coding (TC), applied over blocks within frames. Video coding benefits from the former because samples close together tend to present similar intensities in the same frame as well as across consecutive frames. The Joint Photographic Experts Group (JPEG) committee proposed the homonym JPEG (ITU, 1992; ISO/IEC, 1994) standard that relies on the latter technique (TC) by applying the Discrete Cosine Transform (DCT) (AHMED; NATARAJAN; RAO, 1974) over

[23] Poynton (2012, p. 122) points out the conflict in the use of luminance term in video and its original definition by the Commission Internationale de l'Éclairage (CIE), the standards body for color science. To avoid confusion, some authors adopt a prime in Y' to indicate luma as in the video signal. In this thesis, Y is used to refer to the luma video signal, being equivalent to Y' from (POYNTON, 2012, p. 122).

[24] Chapter 3 presents objective metrics to evaluate the errors introduced by lossy coding. Basically, the errors are the differences between the original signal and the reconstructed one (see Figure 10). The **distortion**, listed as Item 2 in the list of video coding trade-offs (p. 39), is therefore a measure of such errors.

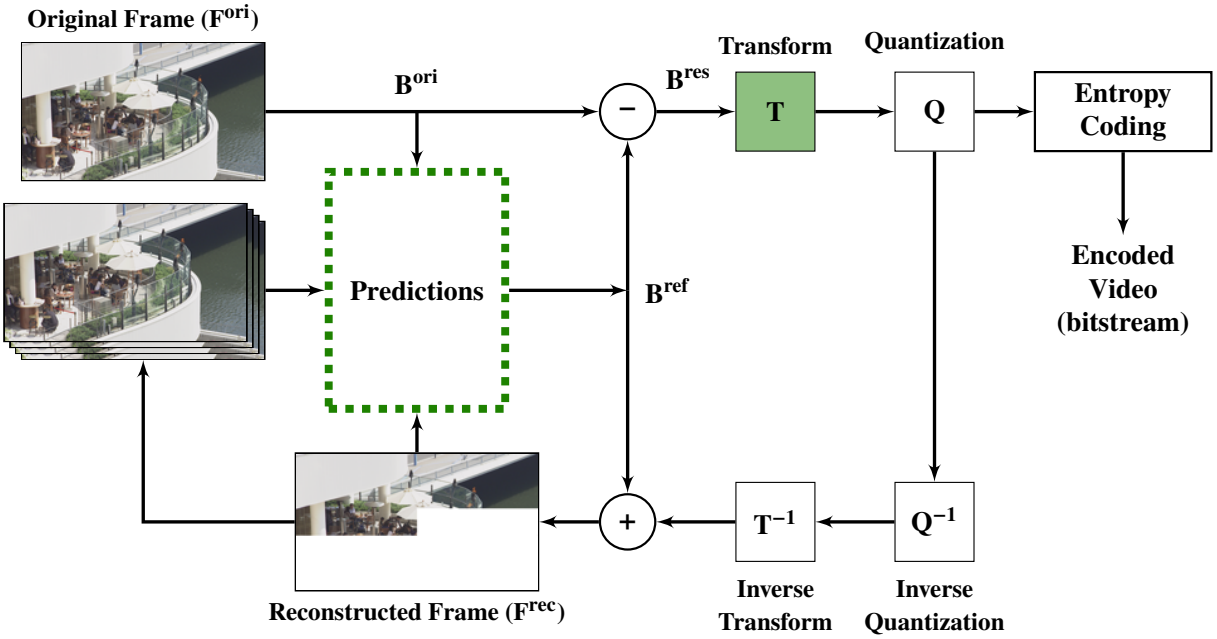
[25] According to Montaña (2017, p. 202), "there is no such thing as a lossless processing of video, neither in the capturing of light nor in the processing of the data". However, considering each encoding tool alone, when the process is fully reversible (i.e., the inverse of the output data is the same as the input data), then such tool is, in fact, lossless.

8×8 pixel blocks to encode still images (WALLACE, 1992, p. xx). Both DPCM and TC used to reduce intra-frame (spatial) redundancies go way back: Habibi (1971) compares DPCM to TC using three different transforms. Although DPCM is simpler than TC in terms of complexity, it is also more sensitive to changes in the signal statistics (HABIBI, 1971, p. 954).

2.1.1 The hybrid model

Consequently, a hybrid coding model was created to combine the merits of both techniques (SHI; SUN, 2008, p. 109). Such a model was successful and adopted in international video coding standards since H.261 (see Figure 4). One reason for its success was its ability to exploit inter-frame (temporal) redundancies (SHI; SUN, 2008, p. 109) through **inter-prediction**. A simplified model of a hybrid video encoder is depicted in Figure 11, having the original frame ($\mathbf{F}^{ori}$) as input and a compliant bitstream as output, defined by a standard such as the HEVC.

Figure 11 – Simplified model of a block-based hybrid video encoder. Notice that the prediction step occurs right before Transform (**T**) and Quantization (**Q**). On the path from the $\mathbf{F}^{rec}$ to the picture buffer a filter step is commonly used to improve predictions.



Source: adapted from Richardson (2003, p. 73). Frames from "BQTerrace" sequence (BOSSEN, 2012).

In the presented model, intra- and inter-frame predictions are the responsible for reducing the spatial and temporal redundancies, respectively, through differential coding (represented by the minus sign in Figure 11). As in TC, the prediction steps of the hybrid model operate on a block-by-block basis. Section 2.2 explains the Block Matching Algorithm (BMA) (CHAKRABARTI et al., 2015), used to find a predictor (also called reference block $\mathbf{B}^{ref}$) for an original block $\mathbf{B}^{ori}$, which is a partition of an $\mathbf{F}^{ori}$. The difference between the

$\mathbf{B}^{ori}$ and $\mathbf{B}^{ref}$ is called residual block ($\mathbf{B}^{res}$). Such $\mathbf{B}^{res}$ is expected to present lower entropy than its related $\mathbf{B}^{ori}$.

After prediction, each $\mathbf{B}^{res}$ is sent to the transform step (**T**). The **T** step changes the residue sample domain from space to frequency, often using the DCT (AHMED; NATARAJAN; RAO, 1974). The DCT can be interpreted as a "simplification" of the Discrete Fourier Transform (DFT) where the imaginary sines terms get eliminated, and only cosine terms remain (WIEGAND; SCHWARZ, 2010, p. 203). There are several DCT types, the most known and used one in both image and video coding is the DCT-II (WIEGAND; SCHWARZ, 2010, p. 204). According to Wiegand and Schwarz (2010, p. 204), the advantages of using DCT:

- It provides a good approximation of the signal-dependent Karhunen-Loève Transform (KLT), which is the optimal transform in terms of energy compaction[26] (AKRAMULLAH, 2014, p. 39);
- Its independence on the input signal (unlike the KLT, where the transform basis is dependent on signal statistics (DIAMANTARAS; STRINTZIS, 1999, p. 1508));
- There are fast algorithms to compute its forward and inverse versions;
- Its 2D version is separable, i.e., a 2D transform may be computed using two separated 1-Dimension (1D) transforms (see Equations 2.19 and 2.20);

The next step after **T** is called quantization (**Q**). Quantization consists in a reduction of the signal representation interval, after integer divisions by quantization coefficients: losses may arise by relinquishing the division remainder, which will not be recovered in the inverse process. In this sense, the space-to-frequency domain change induced by the **T** step helps to reduce the visual impact of the quantization. This is because the HVS is usually less sensitive to higher spatial frequencies than to lower ones (RICHARDSON, 2002, p. 17, 53), and thus the former can be more aggressively quantized.

The quantization is the largest responsible for quality losses[27] while bringing significant compression (MALVAR et al., 2003). Increasing the quantization coefficients, through a Quantization Parameter (QP), increases compression at a quality price. Therefore, the quantization allows for a certain degree of control over the RD ratio.

The quantization output goes, along with prediction side information (e.g., motion information), to the entropy encoder. Such final step generates the bitstream exploiting the reduction of entropy due to the previous steps to generate an almost optimal bitstream. For instance, the entropy in the binary representation can be exploited using binary arithmetic coding (WIEN, 2014a, p. 255).

[26] I.e., in the ideal scenario, the KLT packs most of the signal energy in the least amount of transform coefficients (DIAMANTARAS; STRINTZIS, 1999, p. 1508).

[27] While in AVC the transform was an integer approximation of the DCT with exact inverse, this is not true for HEVC that, in some cases, presents truncation and discarded coefficients during the transform step. Therefore, while in AVC the only source of compression errors was the quantization, HEVC has also the transform step as a source of errors.

To avoid drifting errors[28], there is also a decoding loop in the encoding process that is the source for new predictors to be used for encoding other frames. Part of the decoding loop are the inverse quantization ($\mathbf{Q}^{-1}$) and inverse transform ($\mathbf{T}^{-1}$). They return the residues to space domain which are then added back to the used predictor to reconstruct the original frame, with possible quantization errors.

The quantization errors are the difference between the original input and the reconstructed output (from the decoder, also present in the decoding loop of the encoder). In fact, these errors may be perceived as quality decrease due to compression. Moreover, the better the prediction, the lower are the quantization errors: with lower magnitude residue to be quantized after the transform, smaller will be the differences after reconstruction.

Given all these presented tools within a hybrid video encoder, it is evident that the advance of lossless compression by using predictions must be perfected, while the impact of lossy compression lessened.

2.1.2 A look at video coding standards: where lies their complexity

In HEVC, each distinct partition of a frame is represented in a quad-tree called Coding Tree Unit (CTU) (SULLIVAN; OHM, et al., 2012), exemplified in Figure 12. The CTU area may range from 64×64 down to 16×16 samples, and it may be further split down into Coding Units (CUs) with a minimum area of 8×8 samples. To reduce frame entropy, the CUs undergo through intra- and inter-frame predictions. A CU will be partitioned using one among eight possible Prediction Unit (PU) types for each prediction mode, i.e., intra and inter, explained in Sections 2.2.1 and 2.2.2, respectively. The choice of whether the CU will be encoded using intra or inter mode belongs to the so-called mode decision step in the encoder. Such a step is also responsible for deciding the partitioning mode of each CTU, and its goal is to ensure a good RD trade-off.

All this partitioning structure contributes to HEVC reaching a good efficiency for the encoding of homogeneous areas, using large blocks, as well as for encoding regions with more details, using small blocks. On the other hand, such an intricate partitioning scheme adopted in HEVC is one of the main culprits for its huge encoding complexity. While its predecessor, H.264/AVC (ITU-T, 2009) had only 41 possible size combinations for each frame partition (called Macroblock (WIEGAND; SULLIVAN, et al., 2003)), a 64×64 CTU can be partitioned in up to 83,500 distinct ways.

The mode by each CTU is partitioned will have an impact on its neighbors due to data dependencies and therefore a chain of bad decisions may ripple throughout the entire Group of Pictures (GOP). The pictures (frames) in a GOP are, in general, independent to the remaining frames of the sequence. To achieve the maximum efficiency of a standard, all

[28] Drift is caused if successive predictions are made in the encoder using original samples as references (open-loop) instead of reconstructed ones (closed-loop) (WIEGAND; SCHWARZ, 2010, p. 164). Using an open-loop for prediction may result in increasing errors over time during decoding (reconstruction). Therefore, to avoid drift, both encoder and decoder must use reconstructed samples (WIEGAND; SCHWARZ, 2010, p. 164).

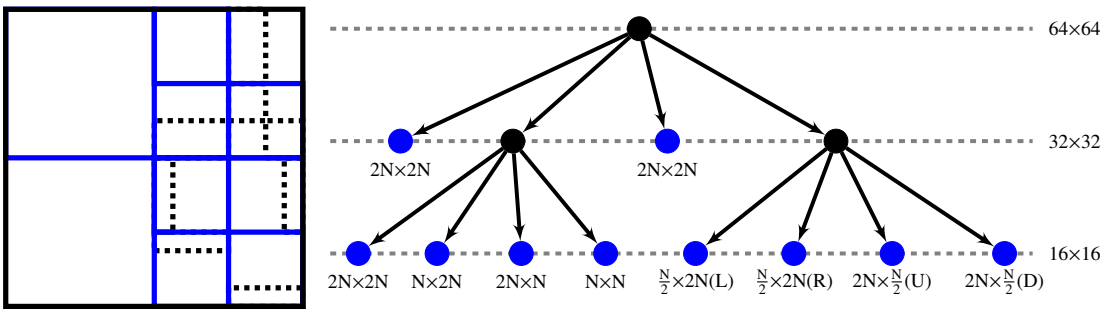
possible combination of partitions in each frame from a GOP would have to be evaluated. It is straightforward that the computational complexity for that would be enormous and thus such process is unfeasible. To keep complexity lower, the encoders often assume a certain statistical independence among neighbor blocks. Therefore, although the standard has an underlying maximum coding efficiency, the set of tools adopted by the encoder may have a significant impact on such efficiency.

As mentioned in Chapter 1, the law of diminishing returns can be applied to this context: after achieving a given coding efficiency, the computational complexity will increase too much for ever small gains. Such an effect is apparent in Figure 5: a huge increase in complexity after each new standard with smaller absolute gains in coding efficiency. Delagi (2010) and Sinangil et al. (2013) estimated that between 2002 and 2020, the complexity of a video encoder would increase by more than ten times. This matches the expected tendency shown in Figure 5.

The efforts for future video coding standards, beyond HEVC, points out that an encoding complexity increase of ~10× over HEVC is still acceptable for many applications (ISO/IEC, 2015). Such efforts are converging to be the VVC, the next ISO/IEC/ITU-T video coding standard[29]. According to Karczewicz and Alshina (2017), VVC achieves 30% better compression than HEVC, but it is 10× more complex. Although VVC simplifies the partitioning nomenclature, it has even more possible combinations: instead of a quad-tree, there is a quad/binary/ternary tree structure whereas the maximum partition size may be 128×128 samples which can be partitioned down to 4×4 (WIEN, 2018).

[29] There is also another future video coding standard currently being elaborated by ISO experts, the so-called Essential Video Coding (EVC). The main goal of EVC is to be licensable while providing at least the same coding efficiency of HEVC (REQUIREMENTS, 2018).

Figure 12 – Example of a 64×64 CTU partitioned using a quad-tree where each of its leaves represents a CU. (a) Partitioned CTU. (b) Quad-tree representing the partitioned CTU. The dashed lines in (a) represent CU partitions, the PUs. In (b) the respective PU types are marked below the CU nodes. The 16×16 CUs were partitioned to show the eight possible PU sizes: the leftmost four in (b) are the symmetric ones at upper right in (a) and the four rightmost in (b) are the asymmetric ones at bottom right in (a).



Source: adapted from Sullivan, Ohm, et al. (2012).

2.2 DIVING DEEPER INTO THE PREDICTIONS

All the partitioning scheme presented in Section 2.1.2 aims at improving coding efficiency. Large blocks, especially from homogeneous regions, benefit more from larger transform sizes (OHM et al., 2012): most of the quantized residues would be in the higher frequency transformed coefficients. For the same reason, the coding efficiency arising from the use of larger transforms are more significant for videos with large spatial resolutions (LAINEMA et al., 2012). Moreover, by using such quite flexible partitioning scheme, the encoder is able to improve its predictions. Thus, this section describes with more details the prediction process.

During encoding, each non-overlapping block from the original frame ($\mathbf{F}^{\text{ori}}$) is referred to as original block (${\mathbf{B}^{\text{ori}}}_{m\times n}$). To reduce its entropy, a reference block (${\mathbf{B}^{\text{ref}}}_{m\times n}$) is selected as the predictor of $\mathbf{B}^{\text{ori}}$. Such predictor is used to obtain the residual block (Equation 2.2) that will be then transformed and quantized (see Figure 11).

$$ {\mathbf{B}^{\text{res}}}_{m\times n} = {\mathbf{B}^{\text{ori}}}_{m\times n} - {\mathbf{B}^{\text{ref}}}_{m\times n} \tag{2.2} $$

As presented, to find $\mathbf{B}^{\text{ref}}$, the encoder uses the BMA (CHAKRABARTI et al., 2015). The goal of BMA is to find a candidate block ($\mathbf{B}^{\text{can}}$), among a set of candidates (S), that minimizes a cost function ($cost$) to be used as $\mathbf{B}^{\text{ref}}$, that is:

$$ {\mathbf{B}^{\text{ref}}}_{m\times n} = \underset{{\mathbf{B}^{\text{can}}}_{m\times n}\in S}{\arg\min}\ cost({\mathbf{B}^{\text{ori}}}_{m\times n}, {\mathbf{B}^{\text{can}}}_{m\times n}) \tag{2.3} $$

Once the obtained reference block $\mathbf{B}^{\text{ref}}$ will be used to compute the residue $\mathbf{B}^{\text{res}}$, $cost$ may be obtained after the difference $\mathbf{D}$ between the original and candidate blocks:

$$ \mathbf{D}_{m\times n} = {\mathbf{B}^{\text{ori}}}_{m\times n} - {\mathbf{B}^{\text{can}}}_{m\times n} \tag{2.4} $$

During BMA, the often large cardinality of S (in Equation 2.3) usually brings the need for a simple distortion metric such as the SAD (RICHARDSON, 2010; AKRAMULLAH, 2014, p. 48) to be used as $cost$. The SAD is defined in (2.5):

$$ sa(\mathbf{D}_{m\times n}) = \sum_{i=1}^{m}\sum_{j=1}^{n} |\mathbf{D}_{i,j}| \tag{2.5} $$

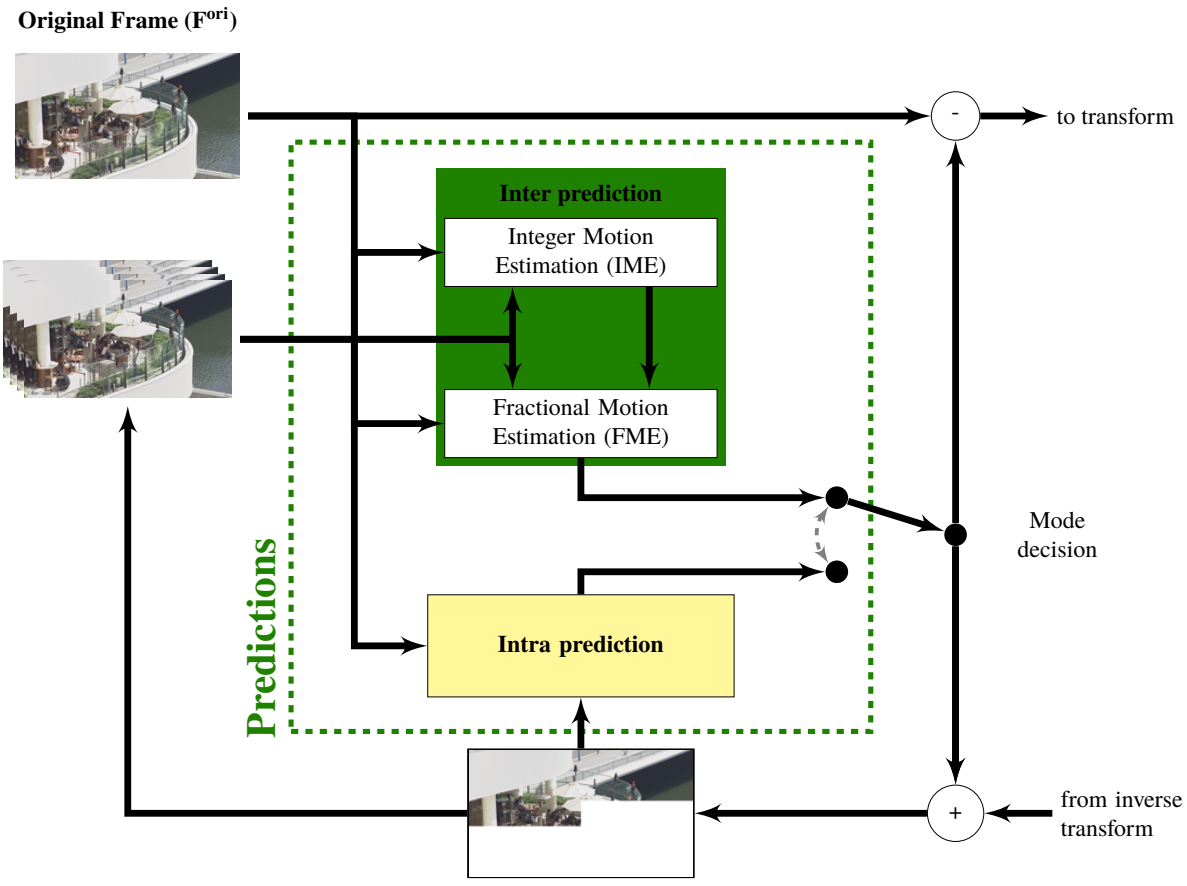
The complexity of BMA, in terms of the total number of operations, depends on how large is the cardinality of set S and on the size of each block. The former defines how many times a SAD will be computed, and the latter defines the number of operations in each SAD computation. Algorithm 1 presents a BMA using SAD as $cost$.

Although simple in its essence, running BMA trying to minimize distortion alone may increase the rate. To better understand why, it is necessary first to define how intra- and inter-frame predictions work. Furthermore, the encoder also needs to decide whether to use intra or inter prediction for a block. Figure 13 shows a block diagram of the prediction steps, which are explained in the following sections.

Algorithm 1: BMA using SAD as *cost*.

Input : $\mathbf{B}^{\text{ori}}$, S
Output $\mathbf{B}^{\text{ref}}$
:
1 $sad^{\min} \leftarrow \infty$;
2 **foreach** $B^{can} \in S$ **do**
3 $\quad \mathbf{D} = \mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}$; // Equation 2.4
4 $\quad sad^{\text{curr}} \leftarrow sa(\mathbf{D})$; // Equation 2.5
5 $\quad$ **if** $sad^{curr} < sad^{min}$ **then**
6 $\quad\quad sad^{\min} \leftarrow sad^{\text{curr}}$;
7 $\quad\quad \mathbf{B}^{\text{ref}} \leftarrow \mathbf{B}^{\text{can}}$
8 $\quad$ **end**
9 **end**

Figure 13 – Detailed view of the prediction steps within the hybrid model from Figure 11.



Source: the author.

2.2.1 Intra prediction

In the intra-frame prediction, a set of $\mathbf{B}^{\text{can}}$ is artificially created by interpolating pixel samples from the borders of other blocks, outside the $\mathbf{B}^{\text{ori}}$. The direction by which the pixel data is interpolated for the generation of each $\mathbf{B}^{\text{can}}$ is called mode, i.e., each mode defines how a $\mathbf{B}^{\text{can}}$ is generated. To allow decoding, besides the residue, the encoder must also include in

the bitstream which mode was selected. Not surprisingly, BMA is used to select the best mode, finding the $\mathbf{B}^{\text{ref}}$ over such a set of candidate blocks (S), one for each mode[30]. The standard determines a possible cardinality of S, by defining the number of intra modes.

Considering AVC, there are nine possible modes for 4×4 luma (Y-channel) blocks (eight directionals plus one planar), four modes for 16×16 luma blocks, and four modes for the chroma components (RICHARDSON, 2003, p. 177). In the case of HEVC, there are as many as 33 directional modes in addition to the DC and the planar modes (WIEN, 2013) for each PU size evaluated, which can be $2n \times 2n$ or $n \times n$. The number of directional candidates in VVC is expected to be even more expressive: about 65 (SULLIVAN; OHM, 2018, p. 18; ZHAO; ZHAO, et al., 2019, p. 53) to 59 modes are reported (FU et al., 2019, p. 55). Moreover, VVC also supports intra prediction of non-square blocks (XU; LIU, 2019, p. 3), namely wide angular intra prediction (ZHAO; ZHAO, et al., 2019).

Intra prediction must be used when there is no reference frame available (i.e., the first frame from a sequence). A frame that only contains blocks that were intra-predicted is called Intra (I)-frame. Apart from the first frame in a sequence being an I-frame, from times to times it is essential to have other I-frames to allow for randomly initiate the decoding of a sequence (in a live streaming, for instance). Also, I-frames help to avoid using references that are too different from the current one, which would reduce coding efficiency.

2.2.2 Inter prediction

The inter-frame prediction is one of the biggest responsible for the high compression rates of today's encoders, thus being one of the most critical steps during encoding (TROCHIMIUK; ABRAMOWSKI, 2014). Such a prediction is based on Motion Estimation (ME). For each $\mathbf{B}^{\text{ori}}$ during ME, a search area is defined in an already encoded frame, called candidate frame ($\mathbf{F}^{\text{can}}$). The set of candidate blocks used by BMA are then obtained from this search area, assuming a translational model of motion (GHANBARI, 2003, p. 240), represented by the so-called Motion Vectors (MVs). Similarly to intra-prediction needing to encode the chosen mode, inter prediction needs to encode which $\mathbf{F}^{\text{can}}$ was used as reference ($\mathbf{F}^{\text{ref}}$) and the MV, that signals the position of the $\mathbf{B}^{\text{ref}}$ in the $\mathbf{F}^{\text{ref}}$.

The need to signal the $\mathbf{F}^{\text{ref}}$ arises because the ME may also select among a set of available $\mathbf{F}^{\text{can}}$s. Also, a frame that contains only intra and inter-predicted blocks that use only one $\mathbf{F}^{\text{ref}}$ is called Predicted (P)-frame. If a frame also contains inter-predicted blocks that use two[31] $\mathbf{F}^{\text{ref}}$s, it is called Bi-Predicted (B)-frame.

During ME, the size of S for the BMA is defined by the used search algorithm, which

[30] A "two-step" step BMA is often adopted, first performing a rough mode decision that selects a subset of the candidate blocks using a simplified Rate-Distortion Optimization (RDO). Then, on the second step a full RDO is executed to find the $\mathbf{B}^{\text{ref}}$.

[31] This is a case where part of the prediction comes from one $\mathbf{B}^{\text{ref}}$ in a frame whereas other part comes from another $\mathbf{B}^{\text{ref}}$ possibly in another frame. Also, it is possible to reorder the sequence as to allow for some predictions use future frames (in sequence time).

some times also defines the evaluation ordering of each $\mathbf{B}^{\text{can}}$. The Fullsearch Block Matching Algorithm (FBMA), illustrated in Figure 14, is used to obtain the optimum result for the given search area (BARNEA; SILVERMAN, 1972). Once every $\mathbf{B}^{\text{can}}$ in the search area is evaluated in FBMA, the size of such area determines the cardinality of S. Figure 14 shows the large number of evaluated $\mathbf{B}^{\text{can}}$s when using FBMA, even for a small search area (16×16). Generalizing to a $w\times h$ sized search area and $m\times n$ sized blocks, in the FBMA $(w-m+1)\times(h-n+1)$ $\mathbf{B}^{\text{can}}$s are evaluated. Thus, it is possible to see why ME is regarded as one of the main culprits for the huge complexity of video encoders, taking about 40% up to 90% of encoding time (BOSSEN et al., 2012; KUMAR; KUMAR; PANDIT, 2019, p. 698).

The ME algorithm presented in Figure 14 takes into consideration discrete displacements (pixel-by-pixel) within the search area. Therefore, when only integer values are considered for MV, the ME becomes known as Integer Motion Estimation (IME). By its turn, Section 2.4 brings details about the Fractional Motion Estimation (FME), where pixels are interpolated between integer pixel positions, thus creating new candidates to perform BMA.

As mentioned, the MVs need to be encoded into the bitstream so the decoder can find the $\mathbf{B}^{\text{ref}}$ again. Because neighboring blocks tend to present similar motion, ever since AVC the hybrid encoders also perform the so-called Motion Vector Prediction (MVP). By doing so, only the residue between the Predicted Motion Vector (PMV) and the actual MV need to be encoded. Also, it is common to centralize the search area at the PMV. Given that, in the example shown in Figure 14, the residual motion vector is $\overrightarrow{mv}^{\text{res}} = (3,-2)$.

As shorter MV residues are expected, they require fewer bits to be encoded than the actual MV. Therefore, this is one of the reasons why only minimizing a distortion metric in BMA may increase the rate. For instance, if a block that minimizes distortion is far from its predicted position, a longer MV will be encoded, spending extra bits. Perhaps a candidate closer to the PMV would provide a good enough reference in terms of distortion, with a small added cost to encode its MV residue.

2.2.3 Rate-distortion optimization

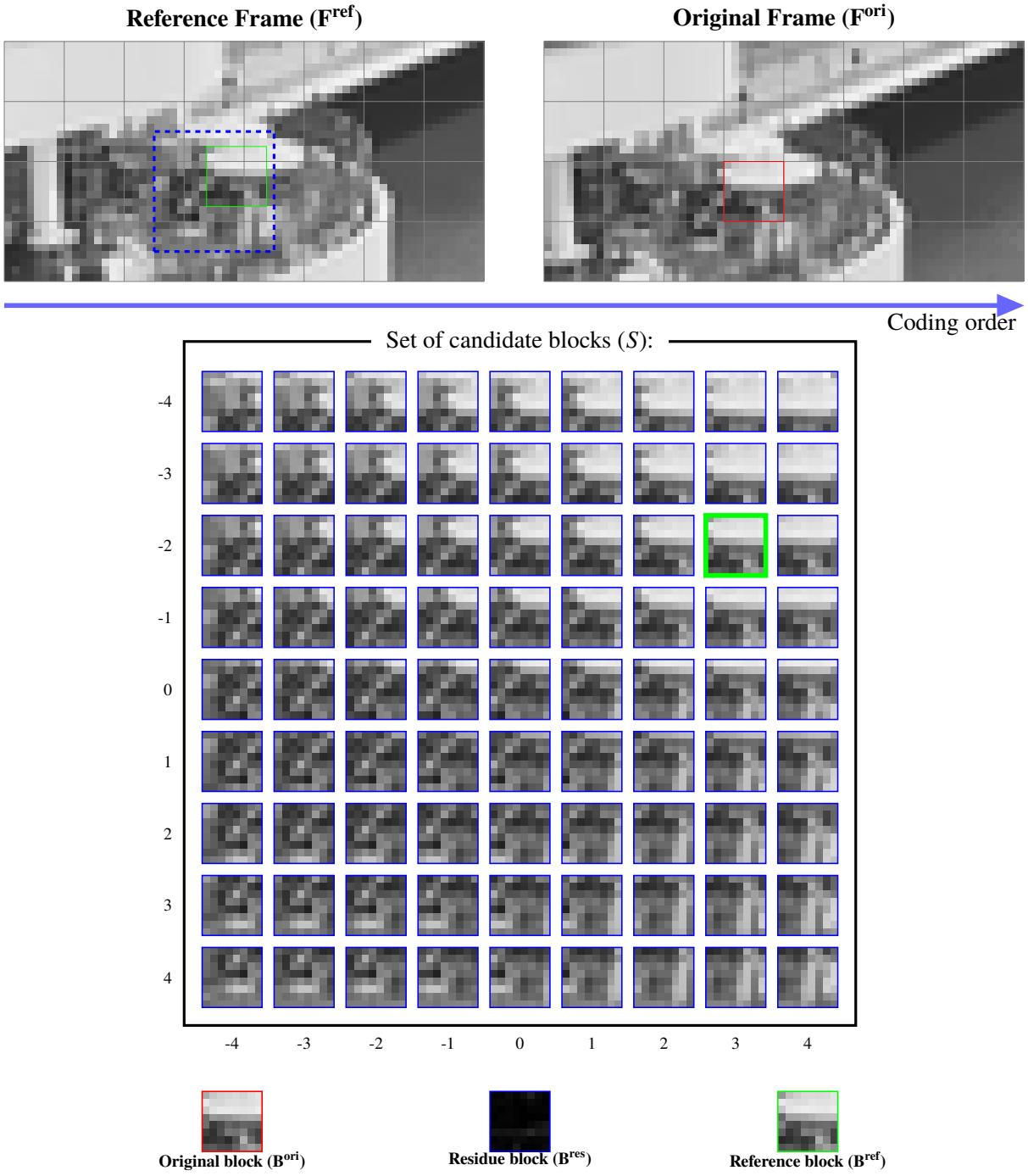
To improve coding efficiency, encoders often rely on the RDO, which uses the Lagrangian rate-distortion cost (j_{cost}) (SULLIVAN; WIEGAND, 1998; ORTEGA; RAMCHANDRAN, 1998) shown in (2.6), where λ is a pre-calculated Lagrange Multiplier.

$$j_{\text{cost}}(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}) = distortion\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right)+\lambda\times rate\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right) \tag{2.6}$$

Let us analyze each term of Equation 2.6.

Firstly, $distortion\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right)$ is a distortion metric, such as the SAD presented in Equation 2.5. However, the actually perceived distortion is caused by the difference between the original and reconstructed frame (Figure 10). Therefore, to be truly optimal in terms of RD a full RDO needs to compute distortion using the reconstructed block ($\mathbf{B}^{\text{rec}}$), as defined in

Figure 14 – Example of FBMA considering a 16×16 pixel search area and block size 8×8 pixels, resulting in 9×9 $\mathbf{B}^{\text{can}}$s $\in S$. Therefore, in this example, the cardinality of S is 81. The numbers surrounding S are the x and y displacements relative to the search area center. The selected $\mathbf{B}^{\text{ref}}$ is the $\mathbf{B}^{\text{can}}$ at coordinate $(3,-2)$. After $\mathbf{B}^{\text{ref}}$ is selected, a $\mathbf{B}^{\text{res}}$ is computed, as defined in Equation 2.2.



Source: the author.

Equation 2.7, instead of directly using $\mathbf{B}^{\text{can}}$ (Equation 2.4).

$$\mathbf{B}^{\text{rec}} = \underbrace{\mathbf{T}^{-1}(\mathbf{Q}^{-1}(\overbrace{\mathbf{Q}(\mathbf{T}(\mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}))}^{\text{Transformed and quantized difference}}))}_{\text{Reconstructed difference}} + \mathbf{B}^{\text{ori}} \tag{2.7}$$

Therefore, for each $\mathbf{B}^{\text{can}} \in S$ from Equation 2.3, a corresponding $\mathbf{B}^{\text{rec}}$ must be computed. This means computing a huge amount of transform, quantization (that introduces most of the losses), and their inverses, resulting in much-increased complexity. Besides such increase in complexity for obtained each reconstructed candidate, other distortion metrics may be used. For instance, the SAD gives the same weights for all errors, no matter how large they are. However, small errors are often less perceptive then larger ones. Given $\mathbf{E}_{m\times n}$ as an error matrix defined as:

$$\mathbf{E}_{m\times n} = {\mathbf{B}^{\text{ori}}}_{m\times n} - {\mathbf{B}^{\text{rec}}}_{m\times n} \tag{2.8}$$

the distortion may be evaluated giving a more significant weight for larger errors. This can be achieved, for instance, by computing the square errors $\mathbf{SE}_{i,j} = (\mathbf{E}_{i,j})^2$. This partially defines another commonly used distortion metric, the Sum of Squared Errors (SSE) (WIEN, 2014d, p. 63), shown in Equation 2.9:

$$\text{SSE} = su(SE) = \sum_{i=1}^{m}\sum_{j=1}^{n}\mathbf{SE}_{i,j} = \sum_{i=1}^{m}\sum_{j=1}^{n}(\mathbf{E}_{i,j})^2 \tag{2.9}$$

Secondly, $rate\left(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}\right)$ from Equation 2.6 is the number of bits needed to encode such $\mathbf{B}^{\text{ori}}$ when considering that the given $\mathbf{B}^{\text{can}}$ will be used as its predictor. Therefore, this includes the number of bits for encoding the transformed and quantized difference (see Equation 2.7), as well as the needed side information. Concerning the rate needed for side information coding, it depends on whether intra or inter prediction is being evaluated. For the former, the intra prediction mode that generated such $\mathbf{B}^{\text{can}}$ must be taken into account. On the other hand, the rate for encoding the MVP needs to be evaluated in the case of inter prediction. All this information is only available after entropy encoding (see Figure 11).

Finally, the Lagrange multiplier (λ) from Equation 2.6 can be seen as a kind of "knob" (TABATABAI et al., 2014, p. 283) that regulates the RD trade-off:

- Increasing λ gives more weight to $rate\left(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}\right)$, and thus a $\mathbf{B}^{\text{can}}$ that minimizes the required rate has a higher probability of being chosen by the BMA;
- Decreasing λ gives more weight to $distortion\left(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}\right)$, and thus a $\mathbf{B}^{\text{can}}$ that minimizes the required distortion has a higher probability of being chosen by the BMA;

For some encoders, such as the HM, there is a distinction between the λ used for mode decision (λ^{mode}) and the one used for BMA during prediction (λ^{pred}). Furthermore, finding the right λ to use is challenging, because there is not one single value that would work well for all kinds of sequences. Anyhow, some models relate λ with the quantization levels (SULLIVAN; WIEGAND, 1998; ORTEGA; RAMCHANDRAN, 1998). Thus, λ^{mode} in HM (ROSEWARNE et al., 2016, p. 41) is computed as:

$$\lambda^{\text{mode}} = \alpha \times w \times 2^{\frac{qp-12}{3}} \tag{2.10}$$

where w represents a weighting factor that depends on several factors[32], such as QP and other encoding configuration, as defined in (ROSEWARNE et al., 2017, p. 38), and α is:

$$\alpha = \begin{cases} 1 - min(\#\text{ of B-frames} \times 0.05, 0.5) & \text{, for referenced frames} \\ 1 & \text{, for non-referenced frames} \end{cases} \tag{2.11}$$

Notice that there is a direct dependency between λ^{mode} and QP. Thus, higher QPs results in higher λ^{mode}, giving more weight to the rate term in BMA. This relationship makes sense in Equation 2.6: a higher QP value is used when lower rate is desired (at a cost in quality) and so choosing a $\mathbf{B}^{\text{can}}$ with lower rate (because of the higher λ) follows the expected behavior.

However good may be using the full RDO as described, the huge complexity it demands (HASHIMOTO et al., 2006, p. 618) makes its use for predictions prohibitive. Thus, most of the time, a simplified RDO is adopted for intra and inter predictions. In general, simplified RDO is adopted when a large number of candidates are evaluated, and then the full RDO is used for final decisions (e.g., selecting between intra or inter coding, block partition sizes) (TSAI; TSAI; CHEN, 2014, p. 360).

The simplified RDO also relies on Equation 2.6. However, it computes distortion directly over $\mathbf{D}$ (Equation 2.4). In this case, the SAD is a well-suited distortion metric given its simplicity. The HM, for instance, adopts SAD in Integer Motion Estimation (IME). For this reason, λ^{pred} is defined as[33]:

$$\lambda^{\text{pred}} = \sqrt{\lambda^{\text{mode}}} \tag{2.12}$$

Also, for simplified RDO, the rate is only an estimate and depends on the prediction type. For instance, HM implements rate estimation for ME considering only the $\overrightarrow{v} = (x, y)$ position of the $\mathbf{B}^{\text{can}}$ relative to the PMV (that coordinate, if such $\mathbf{B}^{\text{can}}$ is chosen as $\mathbf{B}^{\text{ref}}$ will become the residual MV). Such a rate estimate is obtained as:

$$rate\left(\overrightarrow{v}\right) = rate(x, y) = g(x << p) + g(y << p) \tag{2.13}$$

where $g(a)$ is the number of bits to encode the given a using exponential Golomb codes (TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C., 2014, p. 3177). Such a function is defined as:

$$g(a) = 2 \times \lfloor \log_2(2 \times |a| + 1) \rfloor + 1 \tag{2.14}$$

The two left shifts performed in Equation 2.13 are due to the fixed point representation of motion vectors that allow for $1/4$ accuracy in FME, as Section 2.4 shows. Because such

[32] For instance, w is said to be multiplied by 0.95 when SATD is used for FME (ROSEWARNE et al., 2017, p. 38).

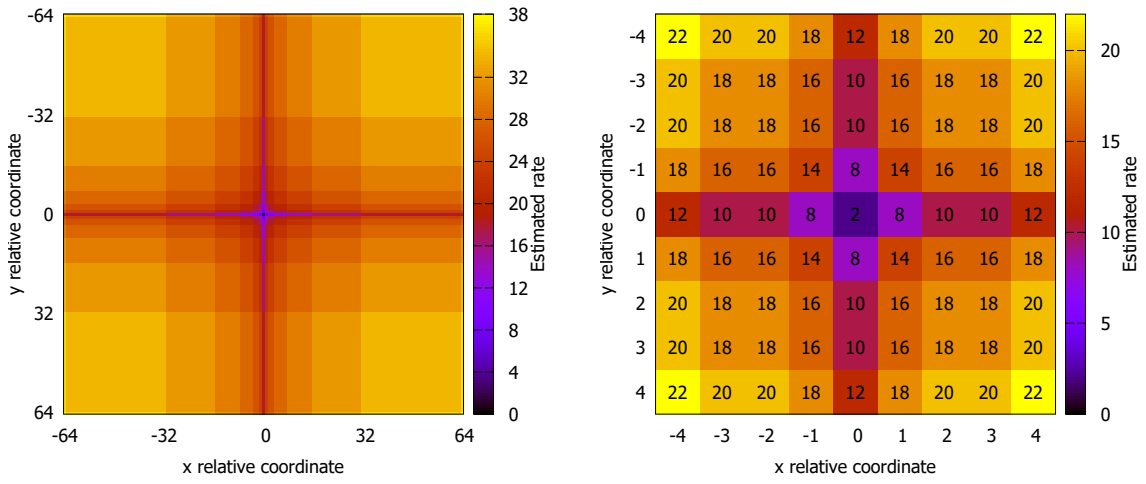
[33] Equation 2.12 is justified by the square operation in SSE that is not present in SAD (SULLIVAN; WIEGAND, 1998, p. 87).

quarter pixel accuracy, two extra bits are needed to represent the fractional movement in MVs. Thus, p in Equation 2.13 is the precision shift amount, that can be defined as:

$$p = log_2\left(\frac{1}{max\_precision}\right) - log_2\left(\frac{1}{current\_precision}\right) \tag{2.15}$$

In the case of HEVC, $max\_precision = 1/4 = 0.25$. Also, as in IME the MV displacements are only in integer positions, $current\_precision = 1$ Therefore, during IME of HEVC, $p = 2$ in Equation 2.13. Figure 15 shows the results of Equation 2.13 in the form of a heatmap[34].

Figure 15 – Heatmap of estimated rate (Equation 2.13) used in simplified RDO when running IME, according the HM implementation. The coordinates (0,0) are indicates the PMV. (a) Estimated rate considering a window with 128×128 positions. (b) Estimated rate considering a window with 9×9 positions, corresponding to the ones from S in Figure 14.



(a) (b)

Source: the author.

Although using a simplified RDO gives satisfactory results with much lower computational complexity than full RDO, the decisions taken in the prediction step become more apart from the remainder encoder steps. In fact, even though the encoder model is a hybrid of prediction and transform, in simplified RDO, often both parts are too isolated. In this scenario, the SATD arises as a distortion metric with the capability to tie both together again. Moreover, SATD does so without all the extra complexity burden of the full RDO.

[34] The pattern formed in Figure 15 helps to understand some of the $\mathbf{B}^{ref}$ choices when using the fast ME algorithm in HM. In fact, there are other fast ME algorithms that exploit those choices, such the one proposed by Gonçalves et al. (2018).

2.3 ON THE SATD CALCULATION

What unites the SATD (used in prediction) with the transform step of the encoder model is the transform part of the metric calculation, defined as:

$$sa(\mathbf{T}(\mathbf{D}_{2^n})) = \frac{1}{2^{n-1}} \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} | \mathbf{T}(\mathbf{D}_{2^n})_{i,j} | \tag{2.16}$$

The main difference between SAD (2.5) and SATD (2.16) is, as one might expect, that the latter computes a transform over $\mathbf{D}$. As mentioned, the main reason for this extra computation during prediction is to account for the forthcoming transform step (see Figure 11), over the $\mathbf{B}^{\text{res}}$. In light of that, the SATD serves as metric that measures both residual error and rate (WIEN, 2014d, p. 64). Although the transform step after prediction usually adopts the DCT, the most widely used one in SATD the Hadamard Transform (HT). The reason is that HT is simpler than DCT (LEE; PARK; JEONG, 2014, p. 513) while providing a good-enough approximation. Acording to Akramullah (2014, p. 43) the HT shows modest decorrelation capabilities, but it is a popular transform given its simplicity (WIEN, 2014d, p. 47). Thus, to some extent, the SATD can account for the differences (Equation 2.4) in the frequency domain (LIAO; CHEN, 2018, p. 141) by adopting the HT. Omitting normalization[35] , HT is defined as:

$$\mathbf{HT}(\mathbf{D}_{2^n}) = \mathbf{H}_{2^n} \times \mathbf{D}_{2^n} \times \mathbf{H}_{2^n} \tag{2.17}$$

where

$$\mathbf{H}_{2^n} = \begin{cases} \begin{bmatrix} 1 \end{bmatrix} & \text{, if } n = 0 \\ \left[\begin{array}{c|c} \mathbf{H}_{2^{n-1}} & \mathbf{H}_{2^{n-1}} \\ \hline \mathbf{H}_{2^{n-1}} & -\mathbf{H}_{2^{n-1}} \end{array}\right] & \text{, otherwise} \end{cases} \tag{2.18}$$

For the sake of simplicity, we will denote the matrix function $\mathbf{HT}(\mathbf{D})$ throughout this thesis as $\mathbf{T}(\mathbf{D})$, whereas the transform will continue to be addressed as HT. Because $\mathbf{H}$ is composed only of plus and minus ones, HT does not require any scalar multiplication, which allows for efficient implementations (TANG et al., 2018). By its turn, the separability property allows for solving first the leftmost multiplication in Equation 2.17, which, by the definition of matrix multiplication can be expressed as:

$$(\mathbf{H}_{2^n} \times \mathbf{D}_{2^n})_{:,k} = \mathbf{H}_{2^n} \times (\mathbf{D}_{2^n})_{:,k} = (\mathbf{F}_{2^n})_{:,k} \tag{2.19}$$

Similarly, the rightmost multiplication can be expressed as:

$$(\mathbf{F}_{2^n} \times \mathbf{H}_{2^n})_{k,:} = (\mathbf{F}_{2^n})_{k,:} \times \mathbf{H}_{2^n} = (\mathbf{T}(\mathbf{D}_{2^n}))_{k,:} \tag{2.20}$$

Given the recursive construction of the $\mathbf{H}$ matrix (in Equation 2.18), each vector multiplication represented in Equation 2.19 and Equation 2.20 can be computed using only $n2^n$

[35] For normalization, $\mathbf{HT}(\mathbf{D}_{2^n})$ must be multiplied by $\sqrt{2}^{-n}$ (WIEN, 2014d, p. 47).

operations, instead of $2^{2n} - 2^n$. Therefore, the whole transform can be computed with $n2^{2n+1}$ operations, instead of $2^{3n+1} - 2^{n+1}$. Such decomposition allowed by the separability and the recursive nature of HT is known as Fast Hadamard Transform (FHT), which resembles the well known Fast Fourier Transform (FFT). In fact, the HT, DFT, and DCT can be implemented after the FFT algorithm (SHI; SUN, 2008, p. 101).

However, even adopting FHT the SATD is still more complex than SAD which requires $3{\times}2^{2n} - 1$ operations, also required for the remaining computation of SATD. Table 4 shows the number of operations required by SAD and SATD for some typical n and also highlights the savings provided by FHT[36].

Table 4 – Number of operations required by SAD, HT (before and after using FHT) and SATD using FHT.

n	2^n	$sa(\mathbf{D}_{2^n})$	$\mathbf{T}(\mathbf{D}_{2^n})$	$\mathbf{T}(\mathbf{D}_{2^n})$ (FHT)	$sa(\mathbf{T}(\mathbf{D}_{2^n}))$
2	4	47	120	64	111
3	8	191	1008	384	575
4	16	767	8160	2048	2815
5	32	3071	65472	10240	13311

Source: the author.

To overcome the impact of this large number of operations as n grows and thus keep an acceptable complexity of the software, only two "small" SATD sizes are available in HM: $n = 2$ or $n = 3$. Also, one may notice that Equation 2.16 only defines the SATD for square sized multiples of power of two. This is because of the definition of HT (Equation 2.17). Therefore, as in HM (JCT-VC, 2013), the SATDs of large, non-square, and non-power-of-two sized partitions may be defined over partitioned matrices as:

$$sa\big(\mathbf{T}\big(\mathbf{D}_{l\times m;2^n}\big)\big) = \sum_{i=1}^{\frac{l}{2^n}} \sum_{j=1}^{\frac{m}{2^n}} sa\big(\mathbf{T}\big(\mathbf{D}_{;i,j}\big)\big) \tag{2.21}$$

where $n = 3$, if $l\%8 = m\%8 = 0$, or $n = 2$ otherwise. For instance, be $\mathbf{D}_{4\times 8}$ then $n = 2$ and:

[36] Although using FHT saves a lot of operations, a structure called Transpose Buffer (TB) is required for the transposition of the partially transformed matrix in a hardware implementation of the SATD. The TB also consumes energy. Chapter 5 shows such structure with more details.

$$
\begin{aligned}
sa\left(\mathbf{T}\left(\mathbf{D}_{4\times 8;4}\right)\right) &= sa\left(\mathbf{T}\left(\left[\begin{array}{ccc|ccc} \mathbf{D}_{1,1} & \dots & \mathbf{D}_{1,4} & \mathbf{D}_{1,5} & \dots & \mathbf{D}_{1,8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{4,1} & \dots & \mathbf{D}_{4,4} & \mathbf{D}_{4,5} & \dots & \mathbf{D}_{4,8} \end{array}\right]\right)\right) && \text{(Def. of } \mathbf{D}_{4\times 8} \text{ after Eq. A.3)} \\
&= sa\left(\mathbf{T}\left(\left[\mathbf{D}_{;1,1} \mid \mathbf{D}_{;1,2}\right]\right)\right) && \text{(changing notation)} \\
&= sa\left(\left[\mathbf{T}\left(\mathbf{D}_{;1,1}\right) \mid \mathbf{T}\left(\mathbf{D}_{;1,2}\right)\right]\right) && \text{(Eq. A.6)} \\
&= sa\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right) + sa\left(\mathbf{T}\left(\mathbf{D}_{;1,2}\right)\right) && \text{(associativity)} \\
&= \sum_{i=1}^{1}\sum_{j=1}^{2} sa\left(\mathbf{T}\left(\mathbf{D}_{;i,j}\right)\right) && \text{(Eq. 2.21)}
\end{aligned}
$$

Algorithm 2 shows such SATD computation as it is implemented in HM. As can be seen, the complexity of SATD depends on the block size, that defines the used HT size as well as the number of partitions.

Algorithm 2: Calculation of larger and non-square SATDs in HM.

Input : $\mathbf{B}^{\text{ori}}{}_{l\times m}, \mathbf{B}^{\text{can}}{}_{l\times m}$
Output $satd^{\text{curr}} = sa\left(\mathbf{T}\left(\mathbf{D}_{l\times m;2^n}\right)\right)$
:

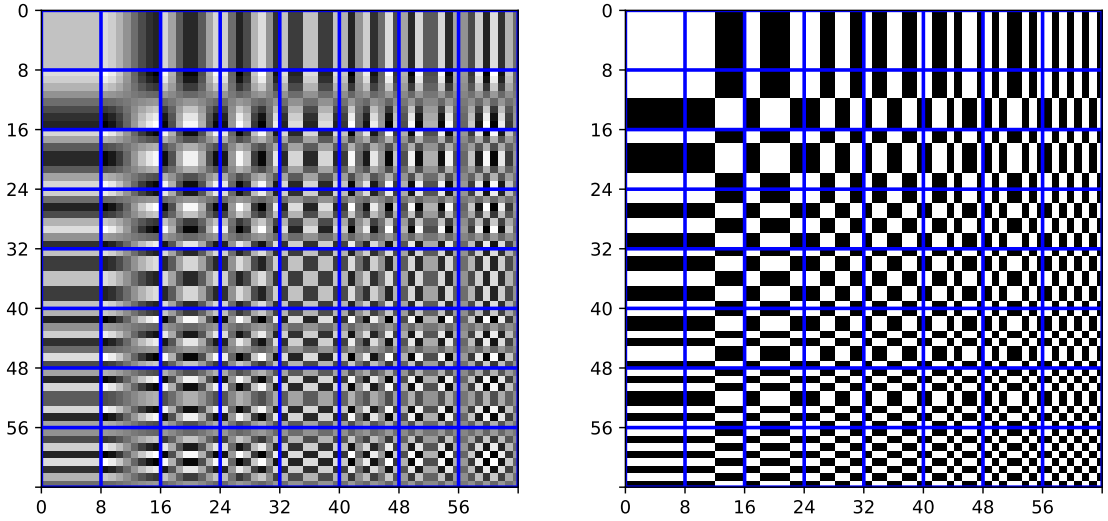
```
if l%8 = m%8 = 0 then
    p ← l/8 ;                          // number of vertical partitions
    q ← m/8 ;                          // number of horizontal partitions
else
    p ← l/4 ;                          // number of vertical partitions
    q ← m/4 ;                          // number of horizontal partitions
end
D ← B^ori − B^can;
satd^curr ← 0;
for 1 ≤ i ≤ p do
    for 1 ≤ j ≤ q do
        satd^curr ← satd^curr + sa(T(D_{;i,j}));
    end
end
```

Using 8×8 size increases the number of operations (see Table 4), but also ensures the similarity between the HT and DCT. Figure 16 presents a graphical representation of the basis functions of the DCT and HT that shows the similarity between them.

In HM, the SATD is used in the Rough Mode Decision (RMD) for intra prediction (LAINEMA et al., 2012, p. 1798). RMD tests all possible intra modes using the SATD as distortion and mode bits for rate (ZHAO; ZHANG, et al., 2011, p. 1). However, in RMD the BMA is modified as to select k best modes[37] to be evaluated in a new BMA using full

[37] Where k depends on the block size (PIAO; MIN; CHEN, 2010).

Figure 16 – DCT and HT basis functions. The HT basis functions are obtained from the HT with the **H** in the so-called Sequency order. The sequency-ordered **H** can be obtained by ordering its rows in a crescent number of sign changes in each row (GEADAH; CORINTHIOS, 1977, p. 436). Although the transform is different, considering the sum in SATD, the same elements are added and therefore the SATD result will be the same, regardless of the positions of the elements in the transformed matrix.



Source: the author.

RDO. According to Vanne et al. (2012), the SATD during the intra prediction in a complete HEVC encoder may be responsible for up to 9% of the total encoder complexity. Figure 17a presents the profiling results for the Kvazaar HEVC encoder (GROUP, 2016) in the all intra configuration[38] (LEMMETTI et al., 2016), showing an even higher time share for the SATD computation. Considering the HM (v10.0) profiling presented by Silveira et al. (2015), it is also possible to notice the large share of SATD in encoding time: Figure 17b shows that SATD alone is responsible for up to 19% of the total encoding time.

Another step in HM that uses the simplified RDO with SATD by default is the FME. In fact, the time share of SATD presented in Figure 17b is from an HM configuration that uses the SATD in FME.

2.4 ON THE FME

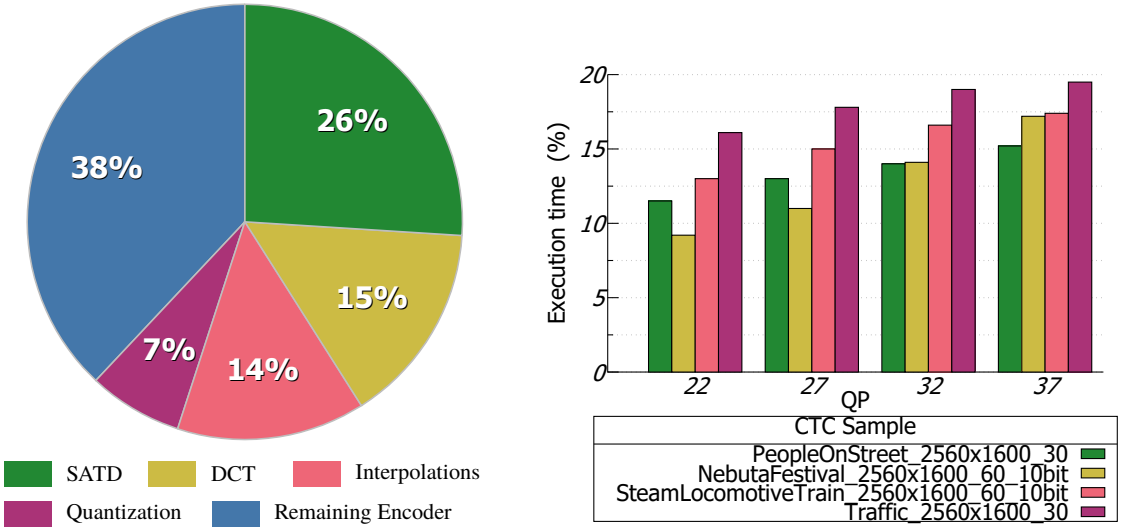
As Section 2.2.2 shows, the ME can be divided into two main steps:

1. Integer Motion Estimation (IME);

2. Fractional Motion Estimation (FME);

[38] Chapter 3 shows such configuration. It is basically an encoder configuration that only exploits spatial redundancies through intra prediction. Thus, all frames are I-frames.

Figure 17 – Impact of SATD in the total HEVC encoding time. (a) Complexity distribution (%) of Kvazaar HEVC encoder tools in all intra configuration. Therefore, the interpolation part corresponds to the generation of $\mathbf{B}^{can}$s for BMA of intra prediction (explained in Section 2.2.1). (b) SATD execution time (%) of HM (v10.0). The presented results are for 8×8 sized SATDs. The SATD 4×4 is responsible for another 2% of the total execution time (SILVEIRA et al., 2015). Notice that the time share depends on the sequence and QP.



(a) (b)

Source: (a) adapted from Lemmetti et al. (2016); (b) adapted from Silveira et al. (2015).

While the BMA in IME gets its $\mathbf{B}^{can}$s at integer positions within the search area, in FME the BMA gets candidates from fractional positions. Such positions are between the pixels at integer positions and thus have no pixels. To obtain pixels there, an interpolation is performed. As presented in Figure 13, the FME is a subsequent step after the IME. In fact, FME is a refinement of IME: the explored fractional positions are around the position of the $\mathbf{B}^{ref}$ selected by the BMA in IME. The $\mathbf{B}^{ref}$ position is represented by the MV from IME, i.e., the $\overrightarrow{mv}^{ime}$.

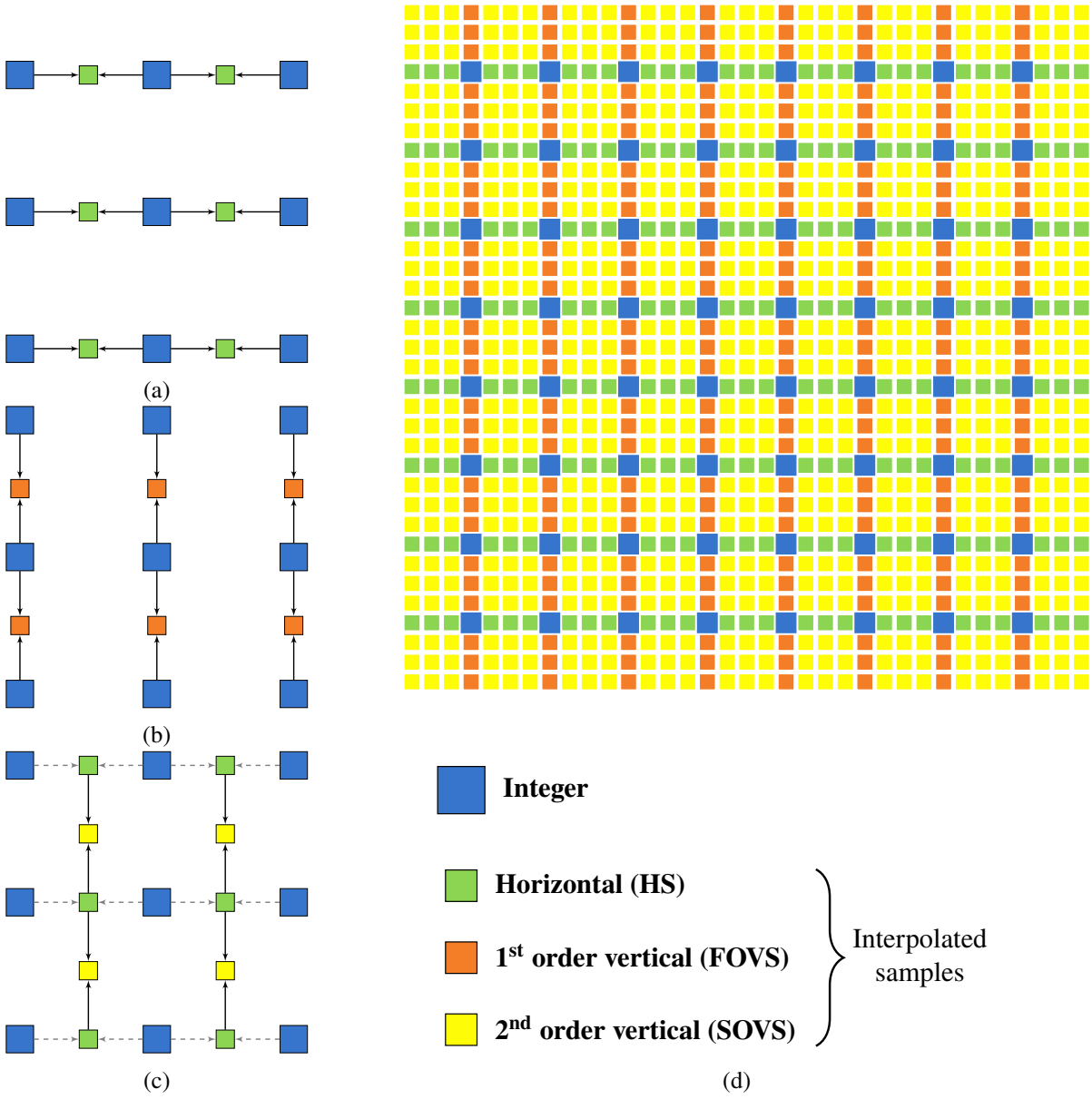
The FME is composed of two steps: 1) the interpolation of samples around the ones pointed to by the $\overrightarrow{mv}^{ime}$ and 2) the search, i.e., the BMA using the interpolated samples that form new $\mathbf{B}^{can}$s. Although the search step is non-normative, the interpolation is normative because the decoder must know how to interpolate new pixel samples between the reconstructed ones at integer positions. Therefore, we will describe the interpolation as defined by the HEVC standard, that allows a 1/4 pixel position to be represented in its MVs for the luma component[39].

In HEVC, the interpolation is performed in both horizontal and vertical directions, as illustrated in Figure 18. The horizontal interpolation uses only pixels in integer positions as input (Figure 18a). On the other hand, the vertical interpolation uses the integer position pixels (Figure 18b) as well as the pixels generated in the horizontal direction (Figure 18c). Therefore, we call the vertical pixels interpolated from integer position samples as First-Order Vertical Samples (FOVS), while the vertical pixels interpolated from horizontal samples are named as

[39] Both VVC and EVC allow 1/8-precision positions for luma component. Both future standards share the same interpolation coefficients among them. Moreover, the interpolation coefficients for 1/4 and 1/2 positions are the same from HEVC, which are covered in this section.

Second-Order Vertical Samples (SOVS) because they are dependent on the horizontals which in turn were obtained from integer position samples. Figure 18d shows all fractional pixels that surround one 8×8 block.

Figure 18 – Directions of the interpolation performed during FME and all interpolated samples surrounding a block with pixels in integer positions. (a) Horizontal interpolation. (b) First-Order Vertical Samples (FOVS) interpolation. (c) Second-Order Vertical Samples (SOVS) interpolation. (d) All pixel samples that may be interpolated at 1/4-precision positions around a 8×8 block. All those pixels form a total of 48 possible $\mathbf{B}^{\text{can}}$s.



Source: the author.

The used interpolation coefficients are independent of the interpolation direction but depend on the proximity of the input samples. HEVC defines two sets of coefficients used for luma component interpolation, depending on the symmetry in the number of the inputs. Both sets are listed in Table 5. If a sample is to be generated halfway between two inputs, its interpolation uses an 8-tap filter with symmetric coefficients (h_i from Table 5). However, if

a sample is to be interpolated closer to one of the inputs (in a quarter position relative to the inputs), it uses a 7-tap filter with asymmetric coefficients (q_i from Table 5). In this case, the larger coefficient should multiply the sample closer to the position where the interpolated sample will occupy. Figure 19 exemplifies this for horizontal pixels.

Table 5 – Coefficients used for HEVC interpolation of luma samples for FME.

i	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**
h_i	-1	4	-11	40	40	-11	4	1
q_i	-1	4	10	58	17	-5	1	

Source: Sullivan, Ohm, et al. (2012).

In some cases, the filter result may be larger than the maximum value that can be represented in a given color depth. When this happens, it is necessary to use a clipping function, such as the one in Equation 2.22.

$$clip(low, high, value) = \begin{cases} low & \text{, if } value < low \\ high & \text{, if } value > high \\ value & \text{, otherwise} \end{cases} \tag{2.22}$$

As shown in Figure 19, there are three types of interpolation: up, middle and down. Considering pixel samples with bit-depth (b), and coefficients q_i and h_i defined in Table 5, up, down and middle samples near position k are obtained after Equations 2.23-2.25, respectively. Notice that up and down filters use the same coefficients but in the reverse order.

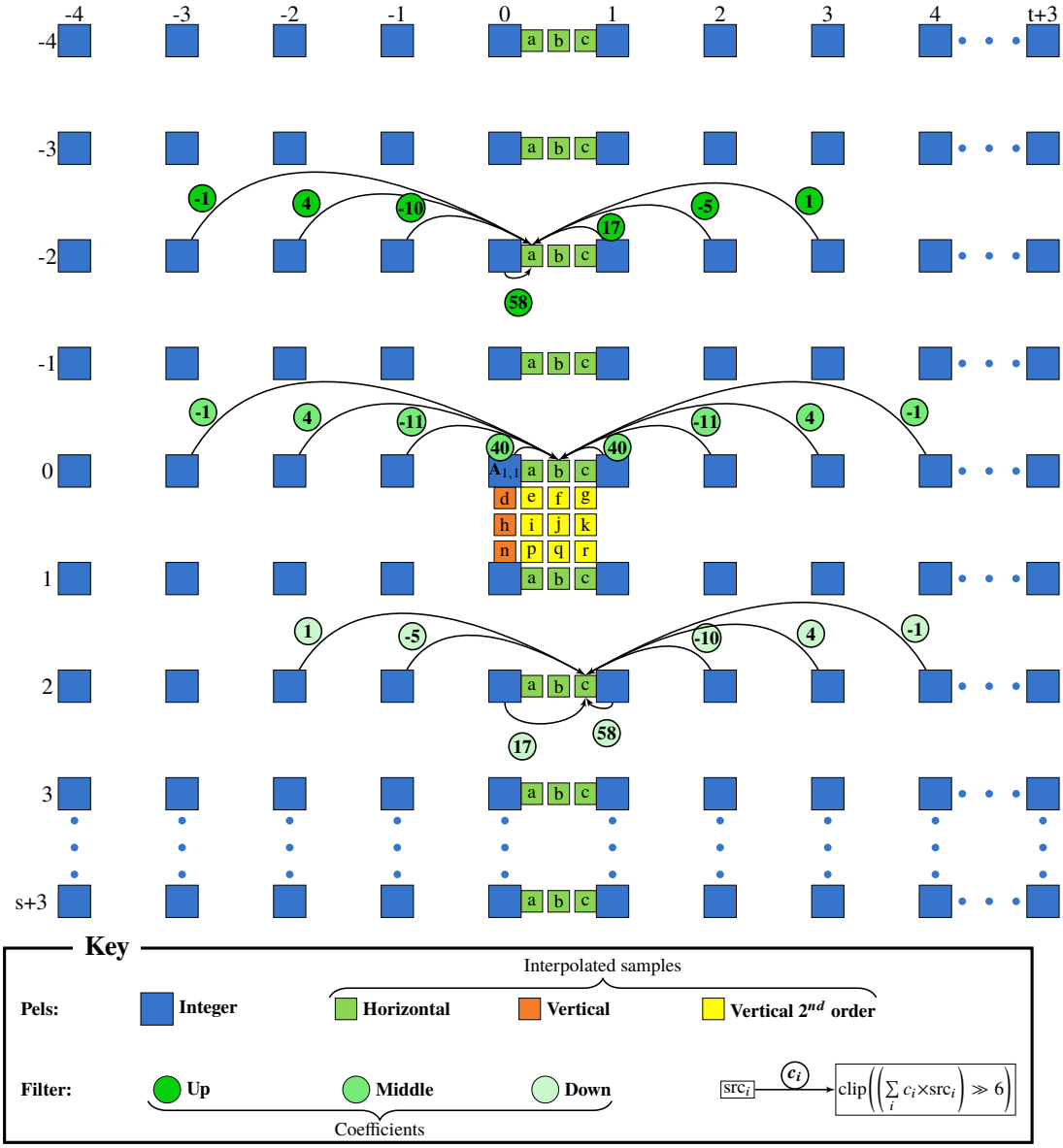
$$u_k = clip\left(0, 2^b - 1, \sum_{i=0}^{6} (q_{i-3} \times in_{k+i-3}) >> 6\right) \tag{2.23}$$

$$m_k = clip\left(0, 2^b - 1, \sum_{i=0}^{7} (h_{i-3} \times in_{k+i-3}) >> 6\right) \tag{2.24}$$

$$d_k = clip\left(0, 2^b - 1, \sum_{i=0}^{6} (q_{3-i} \times in_{k+i-3}) >> 6\right) \tag{2.25}$$

Given the interpolated pixels, the blocks are defined considering unitary integer distances between fractional position pixels. Each horizontal position has two overlapping $\mathbf{B}^{\text{can}}$s, different only by the first and last pixel columns. Thus, as there are three horizontal positions represented by a, b, and c (see Figure 19), there are six horizontal candidates. Similarly, each FOVS position has two overlapping $\mathbf{B}^{\text{can}}$s, different only by the first and last pixel rows. Thus, as there are three FOVS positions represented by d, h, and n (see Figure 19), there are six FOVS candidates. Considering the SOVS, there are four overlapping $\mathbf{B}^{\text{can}}$s for each position (e, f,

Figure 19 – Interpolation of luminance samples in HEVC, considering blocks with $s \times t$ samples. The three types of filters are depicted generating horizontal samples. As most fractional pixels will be interpolated in the vertical direction, we named the filters which generate them considering the position of the larger number of pixels used for their interpolation: up (7-tap), middle (8-tap) and down (7-tap).



Source: adapted from Claudio M Diniz et al. (2015), V. Afonso et al. (2015) and Vladimir Afonso et al. (2016).

g, i, j, k, p, q, and r from Figure 19). For instance, considering the $\mathbf{E}$ defined pixels, of an hypothetical 2×2 block:

$$\mathbf{E} = \begin{bmatrix} e_{1,1} & e_{1,2} & e_{1,3} \\ e_{2,1} & e_{2,2} & e_{2,3} \\ e_{3,1} & e_{3,2} & e_{3,3} \end{bmatrix} \tag{2.26}$$

The four 2×2 $\mathbf{B}^{\text{can}}$s that may be obtained from $\mathbf{E}$ are:

$$\mathbf{B}^{\text{can1}} = \begin{bmatrix} e_{1,1} & e_{1,2} \\ e_{2,1} & e_{2,2} \end{bmatrix}, \text{ at } \overrightarrow{mv}^{\text{ime}} + (-0.75,-0.75) = \overrightarrow{mv}^{\text{can1}} \tag{2.27}$$

$$\mathbf{B}^{\text{can2}} = \begin{bmatrix} e_{1,2} & e_{1,3} \\ e_{2,2} & e_{2,3} \end{bmatrix}, \text{ at } \overrightarrow{mv}^{\text{ime}} + (0.25,-0.75) = \overrightarrow{mv}^{\text{can2}} \tag{2.28}$$

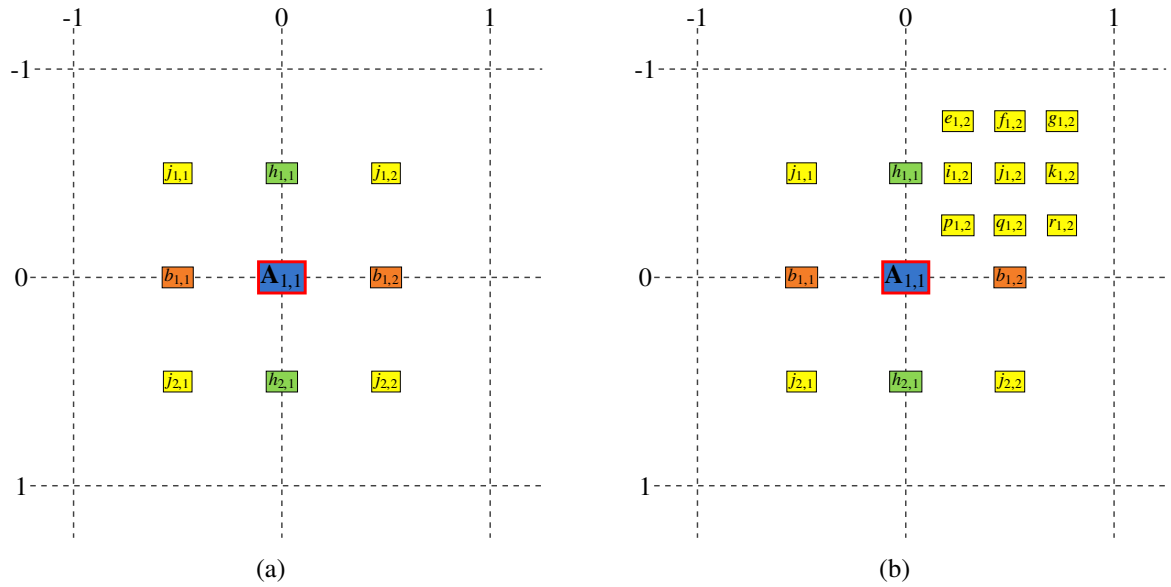
$$\mathbf{B}^{\text{can3}} = \begin{bmatrix} e_{2,1} & e_{2,2} \\ e_{3,1} & e_{3,2} \end{bmatrix}, \text{ at } \overrightarrow{mv}^{\text{ime}} + (-0.75,0.25) = \overrightarrow{mv}^{\text{can3}} \tag{2.29}$$

$$\mathbf{B}^{\text{can4}} = \begin{bmatrix} e_{2,2} & e_{2,3} \\ e_{3,2} & e_{3,3} \end{bmatrix}, \text{ at } \overrightarrow{mv}^{\text{ime}} + (0.25,0.25) = \overrightarrow{mv}^{\text{can4}} \tag{2.30}$$

Therefore, there are four candidates per position and nine positions of SOVS totaling 36 candidates that, together with the 12 from horizontal and FOVS adds up to 48 possible $\mathbf{B}^{\text{can}}$s to be included in S for BMA in FME. However, HM limits its search to 18 candidates (including two repeated candidate evaluations to adjust p for rate estimation, see Equation 2.15). Algorithm 3 shows the way BMA in FME is performed in HM by default.

In those two steps, first in 1/2 positions and then in 1/4 positions around the best 1/2 one, p for rate estimation (Equation 2.13) is 1 and then 0, respectively. Figure 20 shows an example of Algorithm 3 execution. The limitation in the number of candidates have as a downside the possibility to find a suboptimal candidate to use as reference. After all, a better candidate may lie within the opposite direction found by the first step in 1/2 positions as the best.

Figure 20 – Example of FME search in HM. A total of 18 candidates are evaluated, nine in each of the two steps. Assuming, for instance, that the MV in IME was (0,0), the first step chooses $\overrightarrow{mv}^{\text{half}} = (0.5,-0.5)$. If in the second step the candidate starting with sample $r_{1,2}$ is chosen, the final $\mathbf{B}^{\text{ref}}$ is in the position pointed by $\overrightarrow{mv}^{\text{ref}} = (0.75,-0.25)$. (a) Initial search points of HM FME. Lines 2-14 from Algorithm 3. (b) Second step of HM FME considering that the block started at $j_{1,2}$ presented the lower j_{cost}. Lines 16-28 from Algorithm 3.



Source: the author.

Algorithm 3: Search of FME using SATD

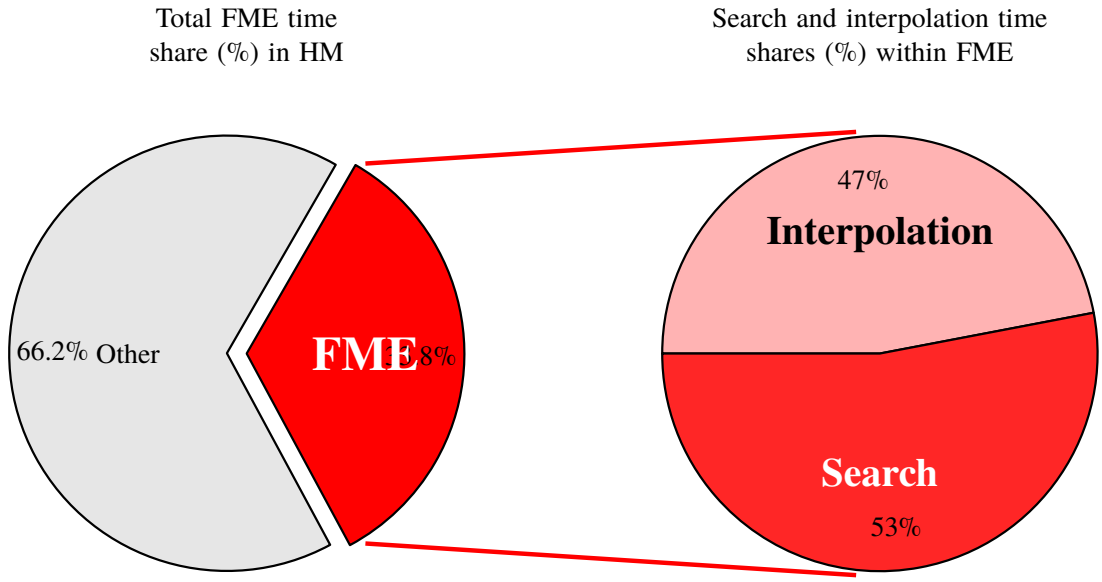
Input : $\mathbf{B}^{\text{ori}}, S, \overrightarrow{mv}^{\text{ime}}$
Output $\mathbf{B}^{\text{ref}}, \overrightarrow{mv}^{\text{ref}}$
:
$j^{\text{min}} \leftarrow \infty$;
for $x \in [-0.5, 0, 0.5]$ **do**
  for $y \in [-0.5, 0, 0.5]$ **do**
    $\overrightarrow{curr} \leftarrow \overrightarrow{mv}^{\text{ime}} + (x, y)$;
    $\mathbf{B}^{\text{can}} \leftarrow$ **GET_CANDIDATE_AT**$\left(\overrightarrow{curr}, S\right)$;
    $\mathbf{D} = \mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}$;
    $j^{\text{curr}} \leftarrow sa(\mathbf{T}(\mathbf{D})) + \lambda^{\text{pred}} \times rate(\overrightarrow{curr})$;
    if $j^{curr} < j^{min}$ **then**
      $j^{\text{min}} \leftarrow j^{\text{curr}}$;
      $\mathbf{B}^{\text{ref}} \leftarrow \mathbf{B}^{\text{can}}$;
      $\overrightarrow{mv}^{\text{half}} \leftarrow \overrightarrow{curr}$;
    end
  end
end
$\overrightarrow{mv}^{\text{ref}} \leftarrow \overrightarrow{mv}^{\text{half}}$;
for $x \in [-0.25, 0, 0.25]$ **do**
  for $y \in [-0.25, 0, 0.25]$ **do**
    $\overrightarrow{curr} \leftarrow \overrightarrow{mv}^{\text{half}} + (x, y)$;
    $\mathbf{B}^{\text{can}} \leftarrow$ **GET_CANDIDATE_AT**$\left(\overrightarrow{curr}, S\right)$;
    $\mathbf{D} = \mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}$;
    $j^{\text{curr}} \leftarrow sa(\mathbf{T}(\mathbf{D})) + \lambda^{\text{pred}} \times rate(\overrightarrow{curr})$;
    if $j^{curr} < j^{min}$ **then**
      $j^{\text{min}} \leftarrow j^{\text{curr}}$;
      $\mathbf{B}^{\text{ref}} \leftarrow \mathbf{B}^{\text{can}}$;
      $\overrightarrow{mv}^{\text{ref}} \leftarrow \overrightarrow{curr}$;
    end
  end
end

Even with this reduced number of candidates, FME is responsible for up to 33.88% of inter prediction encoding time in HM (BLASI et al., 2015). Moreover, as shown in Figure 21, 46.9% of FME time is spent interpolating the pixels at fractional positions, while the remaining 53.1% is spent calculating the SATD over the generated pixels (BLASI et al., 2015).

2.5 CHAPTER CLOSING REMARKS

This chapter contains an overview of video coding, exploring its tools down to the distortion metrics calculation, like the SATD, and to the normative interpolations used in the HEVC FME. Thus, in this chapter, we pinpoint in the well known hybrid encoder model these

Figure 21 – HM execution time shares (%) highlighting the FME. Within FME, almost half of the time is spent in interpolation, whereas the remaining share of the time is spent computing the SATD.



Source: the author with data from Blasi et al. (2015).

two main parts that are exploited in this thesis, i.e., the SATD and the FME. Moreover, the present chapter brings a motivation/justification to exploit both as means to reduce the overall encoding complexity.

As this chapter brought information on the formulation and complexity of those steps, the next chapter will provide, as motivation, information on the coding efficiency that both may provide. After all, a possible way to decrease their complexity during encoding ought to be simply not using them. However, even in a complexity constrained encoder, using the SATD for FME is highly recommended (CORREA et al., 2012) due to its underlying coding efficiency.

Also, as mentioned in Chapter 1, the most complex encoding tools of a video encoder should be off-loaded to dedicated hardware accelerators (VANNE et al., 2012). Unfortunately, dedicated hardware architectures to implement the SATD are more complex than those generally used to calculate the SAD, resulting in silicon area and power consumption overheads. Therefore, to allow for taking advantage of their coding efficiency, SATD and FME hardware architectures must be as energy-efficient as possible. Such architectures are presented in Parts II and III, respectively.

3 CODING EFFICIENCY ANALYSIS

In this chapter, we present a coding efficiency analysis of SATD, FME, and also on the use of the rate term from RDO. Such an analysis serves as a motivation for the adoption of SATD and for evaluating the importance of FME in the compression flow. Moreover, this chapter is also meant to support and justify some hardware design decisions taken in Chapter 6. Thus, we have drawn five main experiments divided into two case studies, listed in Section 3.1.

As presented in Chapter 1, an encoder must provide a good trade-off between rate and distortion. Such a trade-off defines the coding efficiency of the encoder. The rate part is straight-forward to obtain: we only need to get the video sample size (number of bytes) after encoding using a given set of parameters that are being tested. On the other hand, distortion measurement is a broader subject that still demands research efforts.

Section 3.2 presents the method adopted to perform the present coding efficiency analysis. By its turn, Section 3.3 presents related work on the coding efficiency of SATD and FME, whereas Section 3.4 presents the results. Finally, Section 3.5 draws some conclusions about the coding efficiency analysis.

3.1 DEFINITION OF CASE STUDIES

Given the goals of this chapter, we have defined two main case studies, related mainly to SATD and FME, respectively. To assess the coding efficiency that may result from the adoption of the SATD instead of the SAD, we devised the first case study (CS-3.1), that comprises two experiments:

CS-3.1. Coding efficiency provided by the distortion metric: SAD vs. SATD. The default in HM is to use SAD in IME and SATD in FME.

E-3.1a. SAD vs. SATD in IME;

E-3.1b. SAD vs. SATD in FME;

To assess the coding efficiency of FME, as well as some different approaches within FME, we devised the second case study (CS-3.2), that comprises three experiments:

CS-3.2. Coding efficiency provided by the FME

E-3.2c. ME with IME+FME vs. ME without FME (i.e., only IME);

E-3.2d. Only distortion vs. simplified RDO in FME;

E-3.2e. Searching 18 vs. 48 candidate blocks in FME. The default in HM is to use 18 (exemplified in Figure 20), whereas there are 48 possible candidates around one integer position (exemplified in Figure 18);

To our best knowledge, there is no other work that presents results related to Experiments E-3.2d and E-3.2e.

Before presenting the used method, we recommend to readers that are not familiar with image/video quality assessment also read the brief introduction to such theme in Appendix B. Considering that this may be known to the reader, it is possible to go directly to Section 3.2, where we present the evaluation method used in this chapter.

3.2 EVALUATION METHOD

This section presents the coding efficiency assessment method. According to Sole et al. (2018), several factors impact the assessment of CODECs coding efficiency, such as:

- Metrics;
- Encoder implementation;
- Encoder settings;
- Content;

All of the listed factors must be thoroughly described by any assessment, as a small difference in any of these factors may lead to different conclusions on the encoder coding efficiency (SOLE et al., 2018). A discussion on quality assessment is presented in Appendix B, culminating in the introduction of the BD model. The Bjøntegaard Delta Bitrate (BD-Rate) will be adopted as a metric to measure coding efficiency given that it is widely adopted and it provides a result that summarizes a RD curve in terms of percentages, which are straight-forward to interpret. In addition, according to (WINKLER, 2005b, p. 150), Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) (both described in Appendix B) are only well defined for the Y-channel, while there is no agreement on the computation of these metrics for color. Given that the BD model uses RD curves with PSNR results, we will only report BD-Rate for the Y-channel. Moreover, it is necessary to have in mind that **a negative BD-Rate means reduction in the bitrate for the same quality (which is a good result)**, while positive BD-Rate means increase in the bitrate for the same quality.

The next section brings details about the encoder implementation, whereas Section 3.2.2 presents the Common Test Conditions (CTC), that defines the encoder settings and the test content.

3.2.1 Encoder implementation

Regarding the encoder implementation, the conducted coding efficiency analysis relied on HM version 16.16[40]. A few modifications were included to create the devised tests, as

[40] Available at `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.16/`.

replacing SAD by SATD in IME and testing 48 candidates instead of 18 in FME. Table 6 shows the new command line parameters (flags) included in HM to control the tests.

Table 6 – New flags included in HM. To allow reproducibility, we provide patches to the HM 16.16 code.

New flag	Disabled ("0")	Enabled ("1")
`--HadamardFS`	IME (FBMA) uses SAD	IME (FBMA) uses SATD
`--eclFme`	FME searches 18 $\mathbf{B}^{\text{can}}$s	FME searches 48 $\mathbf{B}^{\text{can}}$s
`--eclUseLR`	FME uses only d^* in RDO	FME uses $d+\lambda r^{**}$ in RDO

* Distortion (d) only (as if $\lambda = 0$ in Equation 2.6).

** Equation 2.6 given that $\lambda \neq 0$.

Source: the author.

The first one, `--HadamardFS` sets the SATD to be used as the metric in FBMA for IME (see Figure 14). Whenever it is enabled, we also set the center of the search window to the co-localized block in the reference frame. This is done to ensure the overlapping windows necessary to allow reuse in Variable Block Size Motion Estimation (VBSME)[41]. Notice that the use of FBMA is key for sharing previously distortion computations during VBSME (HU et al., 2017). Moreover, FBMA is usual for hardware implementations of IME due to its regular data flow (HU et al., 2017) and capability of reducing memory access (CHEN, C.-Y. et al., 2006). For the tests involving FBMA, we also disabled the adaptive search range and set the search range to 64 as in previous HM versions.

The second parameter from Table 6, `--eclFme`, enables the use of all 48 candidates in FME (Figure 18d). Algorithm 4 shows a possible search algorithm[42] that evaluates these 48 candidate blocks ($\mathbf{B}^{\text{can}}$s). The alternative, shown in Algorithm 3, is the default HM implementation that searches for the best match among 18 $\mathbf{B}^{\text{can}}$s. Comparing both alternatives is the goal of E-3.2e.

The last new parameter shown in Table 6 is `--eclUseLR`. Contrary to the other two new parameters, this one is enabled by default. Disabling it prevents the use of the rate term from Equation 2.6. In other words, if `--eclUseLR`== 0, then $\lambda = 0$ in FME. However, if `--eclUseLR`== 1, then λ has the values that are defined in HM implementation.

Concerning the remaining encoder settings and the used test contents, we followed the Common Test Conditions (CTC) (BOSSEN, 2012; SHARMAN; SUEHRING, 2017) as explained in Section 3.2.2.

3.2.2 The Common Test Conditions (CTC)

The CTC (BOSSEN, 2012, p. 1) define a total of eight configurations for testing the HM encoder. Half of them are for 8-bit/channel encoder (called main configurations), and the other

[41] The reuse of previous computations is one of the techniques exploited in Section 4.1.

[42] Our actual implementation first searches the six horizontal candidates (Figure 18a), then the six first order vertical candidates (Figure 18b), and finally the remaining 36 second order vertical candidates (Figure 18c).

Algorithm 4: Search of 48 candidate blocks in FME using SATD in simplified RDO.

Input : $\mathbf{B}^{\text{ori}}, S, \overrightarrow{mv}^{\text{ime}}$
Output $\mathbf{B}^{\text{ref}}, \overrightarrow{mv}^{\text{ref}}$
:

$j^{\text{min}} \leftarrow \infty$;
for $x \in [-0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75]$ **do**
  for $y \in [-0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75]$ **do**
    $\overrightarrow{curr} \leftarrow \overrightarrow{mv}^{\text{ime}} + (x, y)$;
    if $\overrightarrow{curr} == \overrightarrow{mv}^{ime}$ **then**
      continue;
    end
    $\mathbf{B}^{\text{can}} \leftarrow$ **GET_CANDIDATE_AT**$\left(\overrightarrow{curr}, S\right)$;
    $\mathbf{D} = \mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}$;
    $j^{\text{curr}} \leftarrow sa(\mathbf{T}(\mathbf{D})) + \lambda^{\text{pred}} \times rate(\overrightarrow{curr})$;
    if $j^{curr} < j^{min}$ **then**
      $j^{\text{min}} \leftarrow j^{\text{curr}}$;
      $\mathbf{B}^{\text{ref}} \leftarrow \mathbf{B}^{\text{can}}$;
      $\overrightarrow{mv}^{\text{ref}} \leftarrow \overrightarrow{curr}$;
    end
  end
end

half are for 10-bit/channel encoder (called high efficiency configurations). The configuration names are:

1. All Intra (AI);
2. Random Access (RA);
3. Low Delay (LD);
4. Low Delay with P slices only (LD-P) (optional);

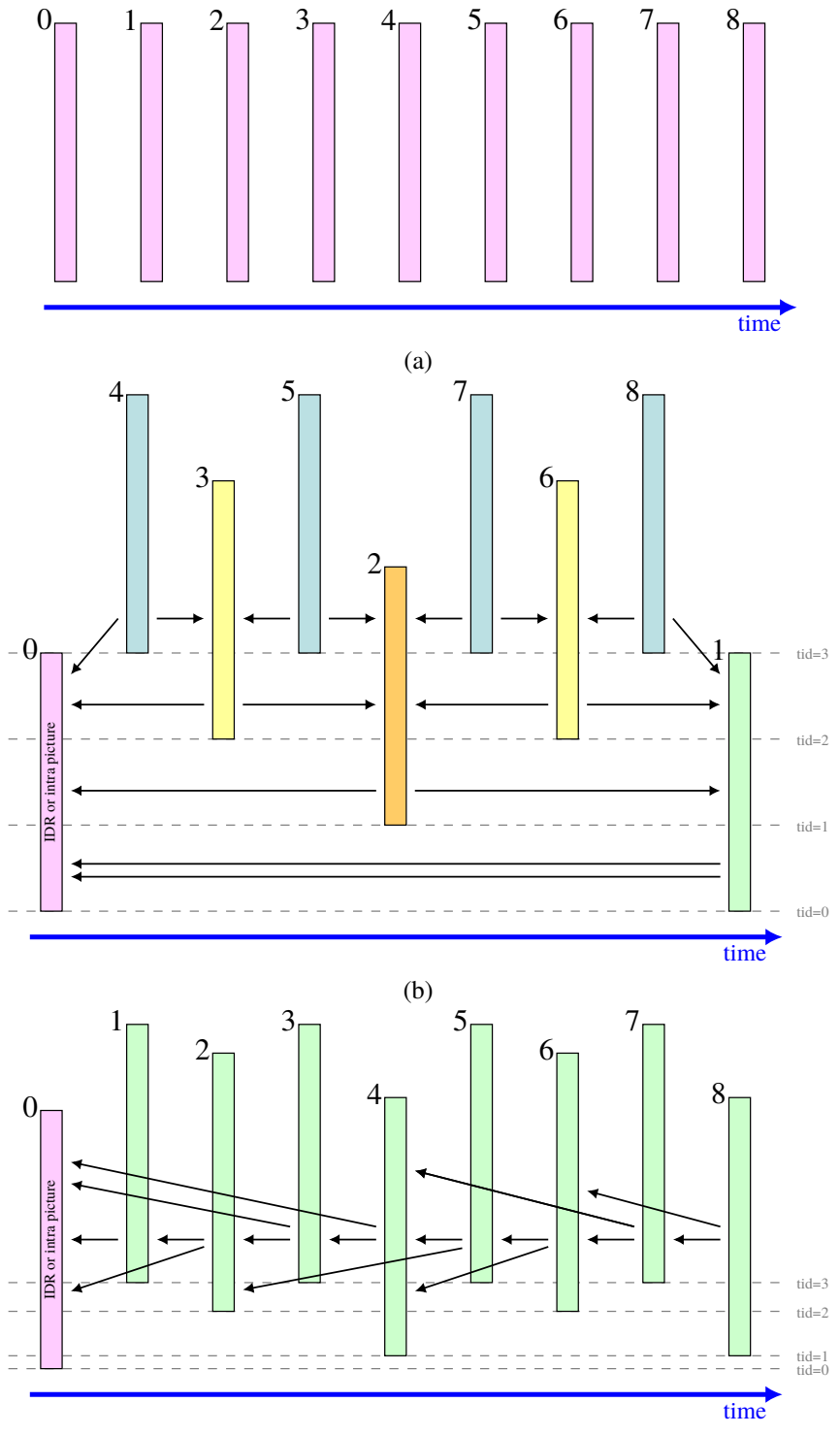
Such configurations define, among other parameters, the frame structure of the video sequence. Figure 22 shows examples of those frame structures. Figure 22a exemplifies the All Intra (AI) frame structure. In such a configuration, all frames are I-frames[43], i.e., they adopt only intra prediction. Therefore, we will not present results for this configuration, since it does not use inter prediction (and thus no ME).

Figure 22b exemplifies the frame structure of Random Access (RA) configuration. Random access is achieved by regularly inserting keyframes[44], such as the first and last frames shown in Figure 22b. The use of such keyframes (mainly I-frames) also helps controlling error

[43] See Section 2.2.1.

[44] "A picture is called a key picture when all previously coded pictures precede this picture in display order." ((SCHWARZ; MARPE; WIEGAND, 2005, p. 2)).

Figure 22 – Example of the frame structures defined by the CTC. The frames are organized in temporal order, while the numbers in their upper left corner represent their encoding order. Also, note that frames with higher temporal id (tid) are less referenced. Thus, HM adopts higher QPs for frames with higher temporal ids. (a) All Intra (AI) frame structure. (b) Random Access (RA) frame structure. (c) Low Delay (LD) frame structure.



Source: adapted from Rosewarne et al. (2017, p. 37) and Rosewarne et al. (2016, p. 37).

propagation (TABATABAI et al., 2014, p. 277). Besides that, such configuration adopts a hier-

archical organization of B-frames[45], which has been found to provide superior coding efficiency than previously adopted B-frame organizations (SCHWARZ; MARPE; WIEGAND, 2005, p. 1). Although suffering from higher delays due to frame reordering, RA configuration also provides the highest coding efficiency among the configurations defined in the CTC (TABATABAI et al., 2014, p. 277).

In the Low Delay (LD) structure (Figure 22c), there is only one initial I-frame, and all the following frames allow inter prediction. As expected, LD has small coding delay because no frame reordering is used (TABATABAI et al., 2014, p. 278). Contrary to RA, in LD configuration, sub-optimal encoder decisions during prediction may cause a snowball-like effect in the upcoming frames. This is because the references may have more encoding artifacts, making more difficult future predictions and so on. The effect is similar to the drift effect[46] explained in Chapter 2. In this case, however, instead of accumulating visible distortions during encoding, the encoder will try to compensate using higher rates.

The CTC (BOSSEN, 2012, p. 2) also lists a set of 24 video sequences, divided into six categories (classes). Such classes contain videos that represent different use cases and characteristics (BROSS et al., 2014, p. 136). Furthermore, those sequences span from high to low spatial and temporal resolutions. Table 7 presents the spatial-temporal characteristics of such test sequences.

A more recent CTC document (SHARMAN; SUEHRING, 2017) defines two new classes (A1 and A2) of UHD video sequences to be used to replace those from class A of (BOSSEN, 2012). Table 8 presents the spatial-temporal characteristics of the new sequences from classes A1 and A2. All the new sequences have a color depth of 10 bpp. However, because the architectures that are presented in this thesis (from Chapter 5 on) operate considering 8-bit samples, our coding efficiency analysis will focus mainly on 8-bit video samples and configurations (i.e., main configurations). Also, all sequences defined in both CTC have 4:2:0 chroma subsampling[47].

Because it is unfeasible to cover all kinds of video sequences with different characteristics, sequences used for evaluation shall have distinct Temporal Information (TI) and Spatial Information (SI) (ITU-T, 2008b, p. 15). These two information features (SHI; SUN, 2008, p. 25) provide objective measurements of temporal and spatial frame complexities, respectively. According to ITU-T (2008b, p. 15), the compression difficulty (in terms of coding efficiency) is directly related to both features.

Let I^{sequence} be an index set containing frame indexes from a given sequence with n frames. Thus, $F^{\text{sequence}} = \{\mathbf{F}^i : i \in I^{\text{sequence}}\}$ is an indexed set (CUNNINGHAM, 2012, p. 157) containing the frames from a given video sequence. Equation 3.1 shows TI calculation for a

[45] See Section 2.2.2.

[46] See Note 28.

[47] 4:2:0 chroma subsampling means that both the Cb and Cr color channels are subsampled by a factor of two in the horizontal and vertical directions (POYNTON, 2012, p. 125). Thus, for each 4 luma pixels (from Y-channel) there is one pixel from Cb and another from Cr channel. See also page 58.

Table 7 – Video sequences used to analyze coding efficiency. The use of these sequences are defined in the Common Test Conditions (CTC) (BOSSEN, 2012, p. 2).

Class	Video	Resolution	Number of frames	fps
A	Nebuta*	2560×1600	300	60
A	SteamLocomotive*	2560×1600	300	60
A	Traffic	2560×1600	150	30
A	PeopleOnStreet	2560×1600	150	30
B	Kimono	1920×1080	240	24
B	ParkScene	1920×1080	240	24
B	Cactus	1920×1080	500	50
B	BQTerrace	1920×1080	600	60
B	BasketballDrive	1920×1080	500	50
C	RaceHorsesC	832×480	300	30
C	BQMall	832×480	600	60
C	PartyScene	832×480	500	50
C	BasketballDrill	832×480	500	50
D	RaceHorses	416×240	300	30
D	BQSquare	416×240	600	60
D	BlowingBubbles	416×240	500	50
D	BasketballPass	416×240	500	50
E	FourPeople	1280×720	600	60
E	Johnny	1280×720	600	60
E	KristenAndSara	1280×720	600	60
F	BasketballDrillText	832×480	500	50
F	ChinaSpeed	1024×768	500	30
F	SlideEditing	1280×720	300	30
F	SlideShow	1280×720	500	20

* 10 bpp sequences.

Source: Bossen (2012, p. 2).

Table 8 – Video sequences used to analyze coding efficiency. The use of these sequences are defined in the Common Test Conditions (CTC) (SHARMAN; SUEHRING, 2017, p. 2).

Class	Video	Resolution	Number of frames	fps
A1	Tango	4096×2160	294	60
A1	ToddlerFountain	4096×2160	300	60
A1	Drums	3840×2160	300	100
A1	CampfireParty	3840×2160	300	30
A2	RollerCoaster	4096×2160	300	60
A2	CatRobot	3840×2160	300	60
A2	DaylightRoad	3840×2160	300	60
A2	TrafficFlow	3840×2160	300	30

Source: Sharman and Suehring (2017, p. 2).

given sequence, where σ is the standard deviation, as defined by ITU-T (2008b, p. 5).

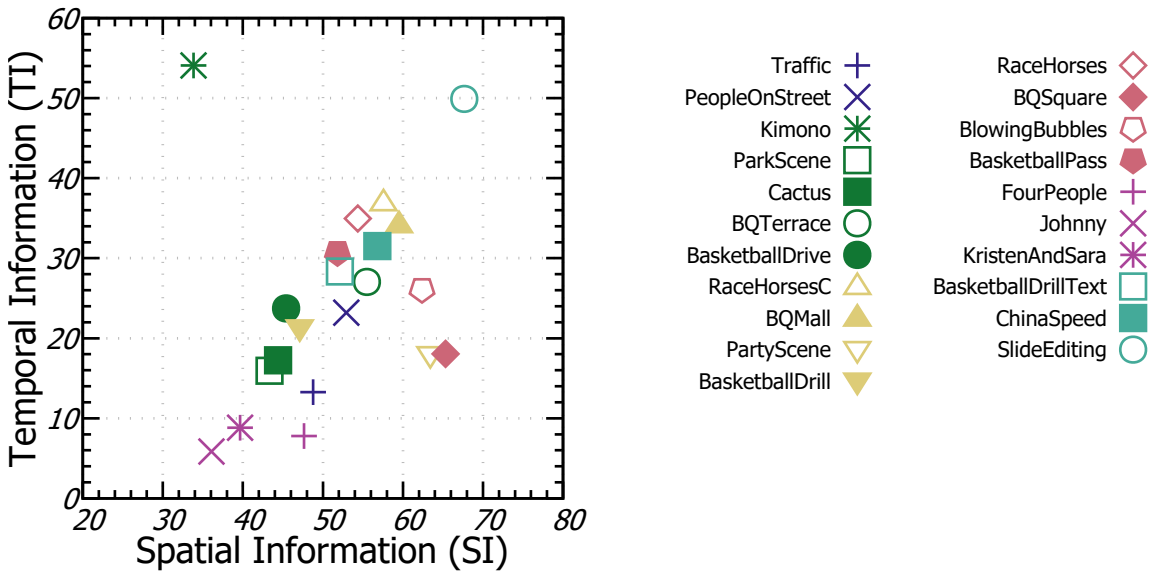
$$ti(F^{\text{sequence}}) = max\left(\sigma\left(\mathbf{F}^{i} - \mathbf{F}^{i-1}\right), \forall i \in \{2,...,n\}\right) \tag{3.1}$$

While TI is measured over the residues of consecutive frames (and thus gives an objective metric of temporal complexity), SI is measured over Sobel filtered frames[48], giving an idea of spatial complexity. As so, TI is usually higher for scenes with more complex motion while SI is usually higher for scenes that are more spatially complex (ITU-T, 2008b, p. 1), i.e., scenes with more edges. Equation 3.2 shows SI calculation for a given sequence, as defined by ITU-T (2008b, p. 5).

$$si(F^{\text{sequence}}) = max\left(\sigma\left(\mathbf{SOBEL}\left(\mathbf{F}^{i}\right), \forall i \in \{1,\ldots,n\}\right)\right) \tag{3.2}$$

We computed TI and SI for the sequences from Table 7 by using Equations 3.1 and 3.2. Figure 23 shows the results[49]. In fact, as can be seen in Figure 23, our coding efficiency evaluation covers a broad range of both features.

Figure 23 – Temporal Information (TI) and Spatial Information (SI) of the 8-bit video sequences from the Common Test Conditions (CTC).



Source: the author.

To generate the RD pairs from which the BD-Rate is computed, we utilized the QPs from the so-called main tier (SHARMAN; SUEHRING, 2017, p. 3): 22, 27, 32, and 37. Also, as presented in Equation B.5, to obtain the BD-Rate, each sequence has to be executed at least two times (baseline and test). Consequently, for each sequence, there are at least eight executions for each experiment/configuration if they are considered independent. Finally, we used our own BD-Rate calculation tool[50] that uses the GNU Scientific Library (GOUGH, 2009) to compute

[48] The Sobel operator is the most common linear filter used to perform edge detection within images (BOVIK, 2005, p. 131).

[49] To compute the TI and SI presented in Figure 23 from the raw samples, we developed a set of tools that we made available at `https://gitlab.com/ismaelseidel/yuv_tools`.

[50] Available at `https://gitlab.com/ismaelseidel/video_quality_analysis`. Given the same input data, our tool provides the same BD-Rate and Bjøntegaard Delta PSNR (BD-PSNR) then the ones presented in (BJØNTEGAARD, 2001).

the polynomial interpolation required in Equation B.4.

3.3 RELATED WORK

This section presents a few works from the literature that provide insights into the coding efficiency of SATD or FME.

3.3.1 Coding efficiency provided by the distortion metric: SAD vs. SATD (CS-3.1)

According to Akramullah (2014, p. 157), using SATD provides better video quality. However, there are no coding efficiency results presented to support such a claim. Ohm et al. (2012, p. 1675) mention that using SATD instead of the SAD usually improves the coding efficiency, but no analysis is presented. To support such a claim, the authors cite (SAPONARA et al., 2003). However, it is not clear that the analysis in the latter work was related to the SATD. Considering that Saponara et al. (2003, p. 10) mention that using HT increases decoding time by 5%, it is probable that their analysis of HT was related to the transform of Discrete Current (DC) coefficients in AVC macroblocks (WIEN, 2014c, p. 225). Nonetheless, Saponara et al. (2003, p. 2) also mention that HT can be used to improve the coding efficiency of distortion metrics.

A brief analysis of the effects of using Hadamard-based SATD in IME and FME was presented in (ABDELAZIM; VARLEY; AIT-BOUDAOUD, 2010). However, their analysis comprised only QCIF (176×144) and CIF (352×288) test sequences and was performed in the AVC reference software. They concluded that the SATD is capable of improving compression if applied to IME, but further research should be done to obtain a more appropriate λ (from Equation 2.6). Such a need for a better λ or even an adaptive λ has been investigated by several authors considering AVC (ZHANG, J. et al., 2010; GONZÁLEZ-DE-SUSO et al., 2014) and also for HEVC (ZHANG; BULL, 2018). Nevertheless, to our best knowledge, the selection of an optimal λ for SATD in RDO remains an open problem that will not be addressed in this thesis.

Correa et al. (2012) evaluated several tools from HM (version 7.0), still during the HEVC standardization. In HM, the use of SATD in FME can be enabled by setting a parameter called `--HadamardME`. According to Correa et al. (2012, p. 1908), `--HadamardME` is among the first tools to be enabled in a complexity-constrained video encoder. When enabled, `--HadamardME` was the tool that provided the highest bitrate savings (up to 2.2%) with a small cost in complexity (CORREA et al., 2012, p. 1905).

3.3.2 Coding efficiency provided by FME (CS-3.2)

Saponara et al. (2003, p. 103) present a coding efficiency evaluation of AVC if it were to adopt 1/8-precision positions in FME. However, their analysis considered only four low-

resolution sequences, two with QCIF (176×144) resolution and two CIF (352×288). Moreover, they did not provide results following the BD model. Their main conclusion on this topic is that 1/8-precision should be enabled only for high-bitrate encodings (SAPONARA et al., 2003, p. 103).

Bross et al. (2014, p. 136) show that the FME in HM (version 6.0) contributes with average BD-Rates of −4.0%, −4.9%, and −2.6%, for RA, LD, and Low Delay with P slices only (LD-P) configurations, respectively. The authors also adopted the CTC, and the baseline for computing their BD-Rate estimates is an implementation of AVC interpolation filters (BROSS et al., 2014, p. 135). The losses are reportedly more substantial for sequences with more high-frequency content (BROSS et al., 2014, p. 136–137), and more prominent on chroma channels, which presents up to −14.3% BD-Rate.

Blasi et al. (2015, p. 145) reported BD-Rate for disabling FME entirely in HM, corresponding to our experiment E-3.2c. However, they evaluated sequences that are not specified in the CTC. In their findings, BD-Rate increases of 7.01%, 6.27%, and 16.33% when encoding UHD, FHD, and HD sequences, respectively. Vladimir Afonso et al. (2016, p. 110) also provided results for such an experiment, considering the sequences defined in the CTC (BOSSEN, 2012). Their results were obtained from executing HM version 13.0rc1. On average, they report an increase of 10.66% and 12.55% of BD-Rate for RA and LD, respectively.

3.4 RESULTS

This section presents the results for case studies CS-3.1 and CS-3.2.

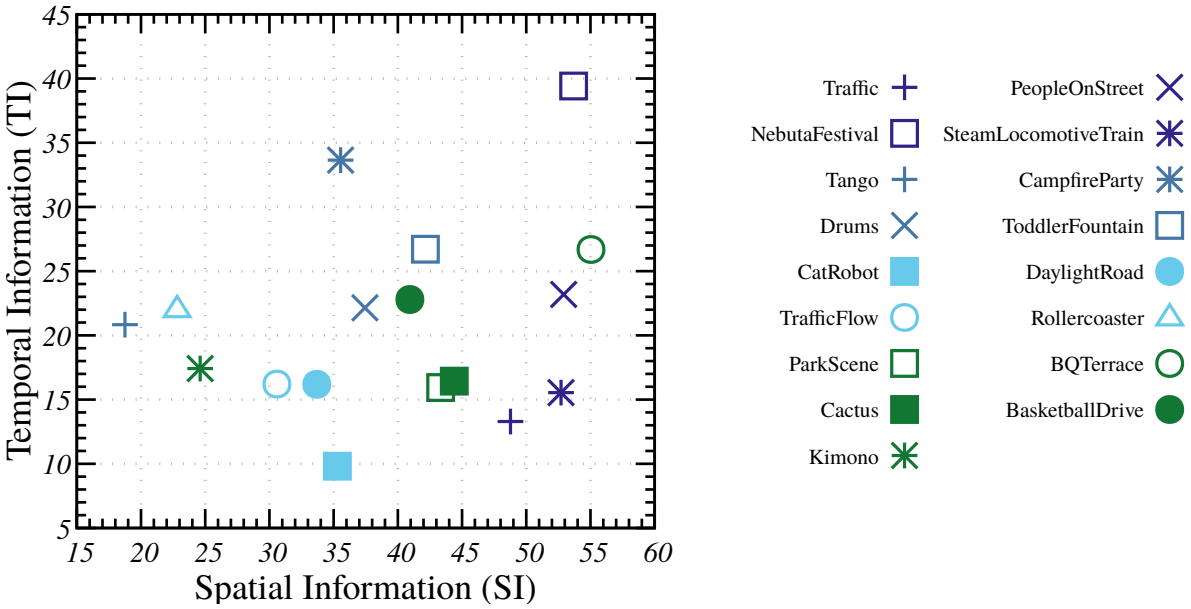
3.4.1 Coding efficiency provided by the distortion metric: SAD vs. SATD (CS-3.1)

As the main goal of this thesis is related to the SATD (see Section 1.4), we decided to provide broader results for the case study CS-3.1, we tested jointly experiments E-3.1a and E-3.1b. Moreover, we tried to cover the highest resolution video sequences in such a test. For that reason, we tested all 17 video sequences (including the 10-bit ones) from classes A, B (from Table 7), A1, and A2 (Table 8).

To limit the execution time, we adopted just the RA configuration and only encoded 128 frames of each tested sequence[51]. RA configuration was used because it is the most realistic one for general-purpose video coding applications, while providing the highest coding efficiency among the recommended configurations from the CTC. Moreover, Bossen (2012, p. 2) and Sharman and Suehring (2017, p. 3) recommend that classes A, A1, and A2 sequences should not be tested with any of the low delay configurations. Figure 24 shows the TI and SI for the 128 frames of the 17 tested sequences.

[51] Even with such limitations, the total execution time was almost 650 days. The load was distributed within a heterogeneous multicore environment.

Figure 24 – Temporal Information (TI) and Spatial Information (SI) of the first 128 frames from each of the 17 tested video sequences from the Common Test Conditions (CTC) defined in (BOSSEN, 2012; SHARMAN; SUEHRING, 2017).



Source: the author.

Table 9 shows the average BD-Rate results, while Table 10 shows the BD-Rate results for each tested sequence with respect to the use of SAD vs. SATD in both IME and FME.

Table 9 – Average BD-Rate* (%) results for Random Access (RA) encoding of all 17 tested sequences (BOSSEN, 2012; SHARMAN; SUEHRING, 2017).

IME/FME	SAD/SATD	SATD/SAD	SATD/SATD
SAD/SAD	-0.53	-0.77	**-1.38**
SAD/SATD	–	-0.25	-0.86
SATD/SAD	–	–	-0.61

*The baselines are in the leftmost column and the tested configurations are in the top row.

It is possible to notice in Table 9 that the best option is to enable SATD in both IME and FME, which results in -1.38% BD-Rate, on average. In particular, such configuration achieves up to −2.99% BD-Rate for CatRobot sequence and −2.53% BD-Rate for Cactus. Table 9 also shows that using SATD in IME alone is slightly better, on average, than using it alone in FME.

As the results presented so far were limited due to their too long execution time, we also ran experiment E-3.1b for all frames of the sequences from classes B to F of the CTC defined in (BOSSEN, 2012). Thus, we have a new set of coding efficiency results comparing the use of SAD and SATD in FME only. Table 11 shows such results for all three tested configurations.

The use of SATD in FME presents superior coding efficiency than using SAD for all tested sequences except ChinaSpeed and SlideEditing. A possible reason for the positive BD-Rate results for those two class F sequences is that they tend to present less high-frequency information than the other sequences. In general, the results are better for LD-P configuration

Table 10 – BD-Rate (%) results by sequence w.r.t SAD in both IME/FME in Random Access (RA) configuration.

Class	Sequence	SAD/SATD	SATD/SAD	SATD/SATD
A	SteamLocomotive*	-0.14	-0.23	-0.40
A	NebutaFestival*	-0.39	-0.10	-0.43
A	Traffic	-0.55	-0.38	-0.95
A	PeopleOnStreet	-0.72	-1.60	-2.37
A1	Tango*	-0.19	-1.03	-1.39
A1	Drums*	-0.35	-0.63	-1.04
A1	CampfireParty*	-0.91	-0.83	-1.94
A1	ToddlerFountain*	-0.21	-0.32	-0.60
A2	CatRobot*	-1.22	-1.66	**-2.99**
A2	TrafficFlow*	-0.19	-0.38	-0.79
A2	DaylightRoad*	-0.30	-1.06	-1.41
A2	Rollercoaster*	-0.20	-0.66	-0.86
B	Kimono	-0.69	-0.48	-1.19
B	ParkScene	-0.52	-0.60	-1.08
B	Cactus	-0.90	-1.29	-2.53
B	BQTerrace	-0.84	-0.65	-1.40
B	BasketballDrive	-0.61	-1.24	-2.08

*10-bit color-depth sequences. The encoder was configured to operate with 8-bit mode, thus each 10-bit source sample s was converted by the encoder to an 8-bit value s' as $s'=(s+2)/4$ and clipped to [0,255], prior to the encoding (SHARMAN; SUEHRING, 2017).

Source: the author.

than for LD, which are in turn better than RA. This is expected since the latter configuration has already the best coding efficiency, which makes it even harder to improve by only changing the distortion metric. Even so, the results are significant (up to -4.33% BD-Rate) given that no change to the standard is necessary. Considering that using SATD in both IME and FME gives even better results than using it only in FME, as seen in Tables 9 and 10 for RA configuration, the results for LD and LD-P would probably improve accordingly.

The main reason to use SATD in FME is its small number of candidates in HM, 18. However, hardware FME architectures may increase such value to up to 48 candidates (the coding efficiency results of such increase are presented in Section 3.4.2), as the one presented in Chapter 6. Finally, it is worth reinforcing that the λ from Equation 2.6 was not optimized for SATD as it is for SAD. A proper λ may increase even more the coding efficiency resulting from the adoption of SATD in both IME and FME (ABDELAZIM; VARLEY; AIT-BOUDAOUD, 2010). The next subsection also presents results considering a scenario where $\lambda = 0$, i.e., disregarding the rate cost during the simplified RDO.

Table 11 – BD-Rate (%) results by sequence when changing the distortion metric within FME: SAD (baseline) vs. SATD (test).

Class	Sequence	RA	LD	LD-P
B	Kimono	-0.85	-0.85	-0.55
B	ParkScene	-0.78	-1.43	-1.28
B	Cactus	-1.59	-1.93	-1.67
B	BQTerrace	-1.19	-3.17	-4.33
B	BasketballDrive	-1.47	-1.54	-1.29
C	RaceHorses	-2.31	-2.49	-2.45
C	BQMall	-1.45	-1.59	-1.53
C	PartyScene	-1.35	-1.85	-1.85
C	BasketballDrill	-1.87	-1.91	-1.64
D	BQSquare	-0.53	-2.50	-2.72
D	BlowingBubbles	-1.41	-2.27	-2.20
D	BasketballPass	-1.78	-1.77	-1.82
E	FourPeople	-0.94	-2.00	-1.73
E	Johnny	-0.91	-2.27	-2.79
E	KristenAndSara	-1.01	-2.18	-2.07
F	BasketballDrillText	-1.80	-2.21	-1.95
F	ChinaSpeed	1.37	1.07	0.54
F	SlideEditing	0.15	0.61	-0.14
F	SlideShow	-1.35	-1.06	-1.45
-	Average	-1.17	-1.69	-1.77

Source: the author.

3.4.2 Coding efficiency provided by FME (CS-3.2)

Table 12 shows the BD-Rate results of E-3.2c, i.e., the inter prediction adopts only the IME and therefore the FME is not even executed.

When considering the subset of corresponding sequences, the results in Table 12 are similar to the ones presented by Blasi et al. (2015, p. 148). For instance, they reported about 21.71% and 5% BD-Rate for BQTerrace and Kimono sequences, considering RA. The RA and LD average results are also similar to the ones reported by Vladimir Afonso et al. (2016, p. 110). However, we included RA results for class E, despite the CTC recommendation (BOSSEN, 2012, p. 2). If such recommendation is followed, the average result of RA is close to -12% BD-Rate, which is closer to the LD average result. By the presented results, it is quite clear that FME is an essential tool for keeping the high coding efficiency of current encoders.

Similarly to the CS-3.1, we considered a joint evaluation of experiments E-3.2d and E-3.2e. Table 13 shows the results of both experiments.

The second column of Table 13 shows specifically the results of E-3.2d, while the fourth column shows specifically the results of E-3.2e. In experiment E-3.2d, the BD-Rates are positive, i.e., there is a decrease in coding efficiency when not using the rate term from Equation 2.6. On the other hand, evaluating 48 candidate blocks instead of only 18 in FME presents negative BD-Rates, i.e., there is an increase in coding efficiency. In fact, from Table 13,

Table 12 – BD-Rate (%) results of skipping the FME by sequence.

Class	Sequence	RA	LD	LD-P
B	Kimono	5.04	4.07	8.03
B	ParkScene	14.26	12.80	27.23
B	Cactus	7.21	7.11	12.57
B	BQTerrace	22.13	25.20	45.92
B	BasketballDrive	10.21	7.75	14.74
C	RaceHorses	17.02	17.58	29.61
C	BQMall	13.53	13.56	24.05
C	PartyScene	16.09	20.70	35.00
C	BasketballDrill	10.53	8.04	14.84
D	RaceHorses	17.02	17.58	29.61
D	BQSquare	20.99	39.26	67.09
D	BlowingBubbles	15.23	18.22	31.27
D	BasketballPass	9.44	9.55	15.65
E	FourPeople	3.90	8.45	14.12
E	Johnny	7.29	13.54	22.28
E	KristenAndSara	4.84	8.66	15.84
F	BasketballDrillText	9.88	7.07	15.76
F	ChinaSpeed	2.15	2.33	3.82
F	SlideEditing	0.04	0.14	0.76
F	SlideShow	11.03	13.33	19.99
–	Average	10.89	12.75	22.41

Baseline: FME enabled. Test: FME disabled. Source: the author.

one can come to two main conclusions:

1. Regardless of the number of candidates evaluated, not using the rate term increases the BD-Rate by 0.26% on average;

2. Regardless of the use of the rate term from RDO, searching in all 48 interpolated $\mathbf{B}^{\text{can}}$ during FME reduces the BD-Rate by -0.14%, on average.

To illustrate the differences resulting from the use of the rate term of Equation 2.6, Figure 25 shows heatmaps of the fractional positions that are chosen in FME for the lowest and highest tested QPs, 22 and 37, respectively.

The results are very similar when considering the lowest QP (22). This is due to a lower λ value, which is dependent on the QP, as presented in Equation 2.10. Such lower λ gives higher importance to the distortion than to rate term, hence the similar results. On the other hand, for the highest QP (37), it is possible to notice that if the rate term is used (Figure 25b), the results are more concentrated at the center, i.e., closer to the result chosen in IME. For the same highest QP, if the rate term is not used (Figure 25d), the results are more dispersed. More variability tends to require a higher rate, thus reducing coding efficiency.

Table 13 – Joint evaluation of λR and number of evaluated candidates in FME.

Class	Baseline: Test:	18 w/ λR 18 w/o λR	48 w/ λR 48 w/o λR	18 w/ λR 48 w/ λR	18 w/o λR 48 w/o λR
B	Kimono	0.21	0.20	-0.06	-0.08
B	ParkScene	0.23	0.23	-0.02	-0.02
B	Cactus	0.40	0.40	-0.11	-0.12
B	BQTerrace	0.16	0.20	-0.18	-0.15
B	BasketballDrive	0.35	0.44	-0.19	-0.11
C	RaceHorses	0.35	0.33	-0.21	-0.23
C	BQMall	0.38	0.41	-0.19	-0.16
C	PartyScene	0.21	0.20	-0.11	-0.11
C	BasketballDrill	0.44	0.44	-0.11	-0.11
D	RaceHorses	0.35	0.33	-0.21	-0.23
D	BQSquare	0.09	0.06	-0.14	-0.17
D	BlowingBubbles	0.24	0.20	-0.11	-0.15
D	BasketballPass	0.36	0.35	-0.14	-0.15
E	FourPeople	0.20	0.18	-0.04	-0.06
E	Johnny	0.17	0.14	-0.29	-0.32
E	KristenAndSara	0.13	0.16	-0.21	-0.18
F	BasketballDrillText	0.52	0.53	-0.13	-0.12
F	ChinaSpeed	0.25	0.31	-0.03	0.02
F	SlideEditing	0.02	0.01	-0.05	-0.06
F	SlideShow	0.14	0.36	-0.26	-0.05
–	Average	0.26	0.27	-0.14	-0.13

Without (w/o) λR means that the rate term is not used during simplified RDO. It is equivalent to $\lambda = 0$ in Equation 2.6.

Source: the author.

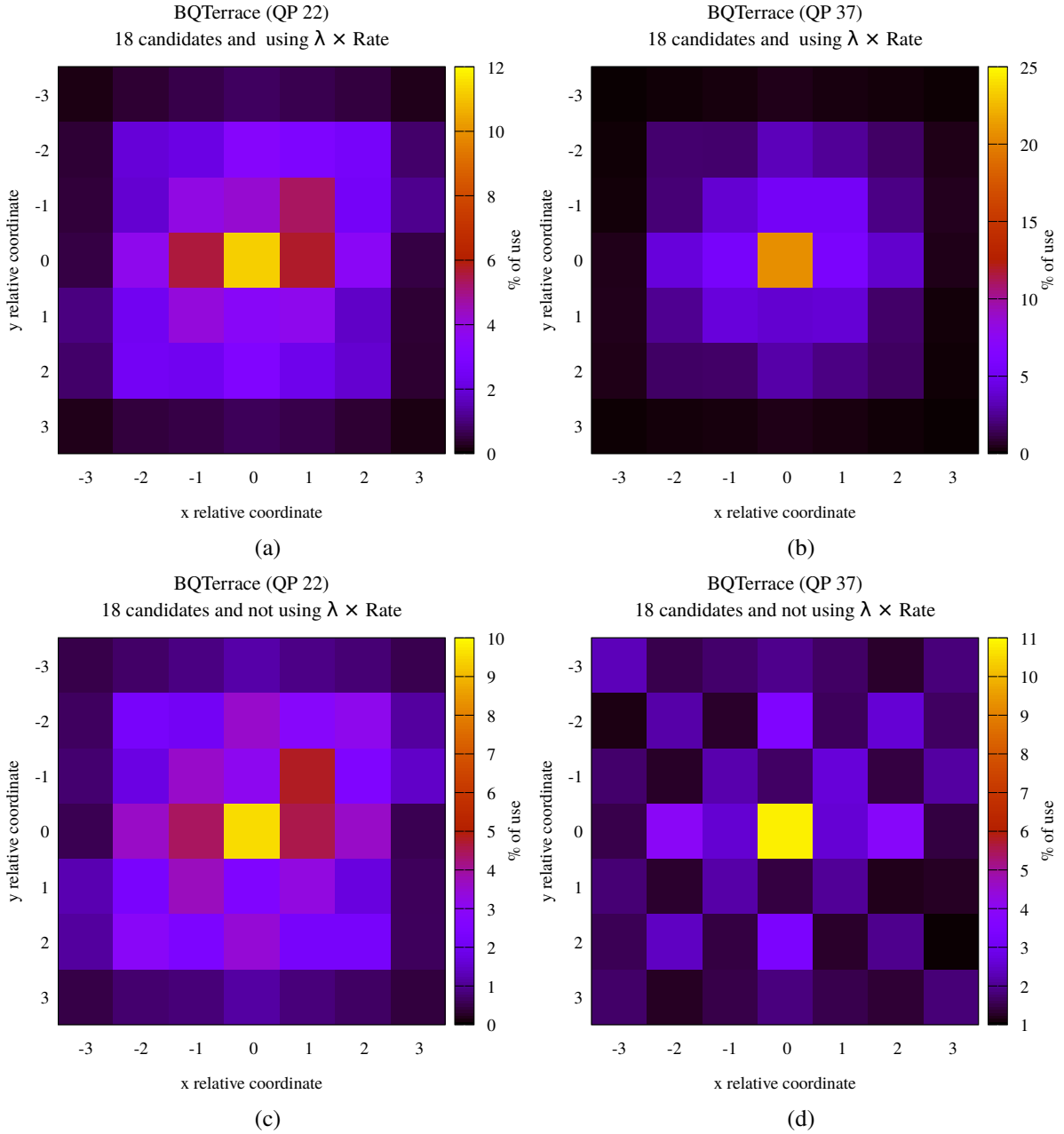
3.5 CHAPTER CLOSING REMARKS

In this chapter, we presented two case studies that serves as motivation for this thesis. The first one is related to the distortion metric used in the simplified RDO, where we compared SAD and SATD. The second one is related to the FME, more specifically evaluating its efficiency and the number of search candidates.

The presented results show a certain degree of variation, as expected when evaluating encoding tools, given the variation in the test sequences content. Exploring further the execution reports, we detected that, as expected, there are variations in the BD-Rate in every frame of every sequence. This means that the BD-Rate variations highly depend on frame content. In this sense, we tried to correlate each frame BD-Rate with "frame" TI, "frame" SI, and QP variations. However, we found no correlation. Such difficulty in explaining the variations in the BD-Rate results is due to the very complex set of variables. These variable influence the massive number of decisions in each frame of each sequence.

Although the variation is not clearly explained, considering the myriad of decisions done inside the encoder while it is trying to optimize the rate-distortion relation, it is important to emphasize that the reached results are strong enough to motive this thesis and to support the adoption of SATD as distortion metric. Therefore, it is worth to seek ways to reduce the SATD

Figure 25 – Choices from RDO vs. only SATD during FME in sequence “BQTerrace”. The topmost heatmaps are using the rate term, while the bottom ones are not. The two at the left are for QP 22 and the two at the right are resulting from QP 37.



Source: the author.

complexity, as in Chapter 4, and to design energy-efficient architectures for that, as presented in Chapter 5.

Also, the results are strong to support the use of FME and show that a search in all 48 candidates using the rate term from simplified RDO is beneficial to increase even more the coding efficiency. In fact, we show that the best FME configuration in terms of coding efficiency is a search in all 48 $\mathbf{B}^{\text{can}}$s using the simplified RDO with rate term. An architecture for the FME with these characteristics is presented in Chapter 6.

Part II

SATD: Complexity Reduction and Hardware Design

4 REDUCING THE SATD COMPLEXITY

As presented in Chapter 1, the increasing resolutions demand more efficient video encoders. Thus, several new tools have been proposed within new standards to improve compression, which increases the computational complexity of encoders. Paradoxically, most methods to reduce the computational complexity end up by decreasing the coding efficiency. However, there are a few which are able to reduce complexity without affecting coding efficiency. They rely on reducing the average number of operations required for the cost computation during BMA (Equation 2.3) without changing its outcome. This means that for the same search order, BMA will result in the same reference block.

One way to achieve that is through calculation reuse, that avoids performing redundant operations, i.e., if the exact same calculation is performed over the same data more than once, it can be reused. Other techniques exploit the minimization of the cost in BMA. When sequentially searching for the best candidate block to become a reference, the BMA keeps track of the current block with the smallest cost until it finds another one with a smaller cost, as in Algorithm 1. The BMA then updates the current minimum and continues the search, repeating this procedure until the cost of all candidates in S are evaluated. Therefore, apart from the reuse, those other techniques can be divided in two groups:

1. **Elimination of impossible candidates**: a simplified criterion is used to determine if the current candidate block is able to provide smaller distortion than the current best one. The so-called Successive Elimination Algorithm (SEA) (LI; SALARI, 1995) lies in this first group: it was proposed as a means of skipping SAD computations during BMA.

2. **Early-termination of the metric computation**: during the cost computation, its current partial value may be evaluated. If such value is larger than the current minimum, the remaining computation can be skipped. Such kind of early-termination[52] is called Partial Distortion Elimination (PDE), which is a means of reducing the computational load of SAD (CHIU; SIU, 2006) without introducing losses (CHAN; SIU, 2002).

To be able to reduce complexity, the methods in both groups must be **simple** but also **effective**. By simple we mean that the overhead must be small, while by effective we mean that their capability of skipping operations must be large. If a method is simple and effective enough, it shall be able to provide savings in terms of number of operations performed. On the other hand, if a technique is not simple or effective enough, its overhead will surpass the gains it could provide. Thus, instead of reducing the complexity, such overhead would introduce a complexity increase. The **simplicity** is indeed critical, as it can render the techniques useless.

[52] Another kind of early-termination consists on terminating the metric computation as soon as it is larger then a predefined threshold. However, such method is not optimal with respect to FBMA.

Furthermore, even so these methods were first introduced to deal with SAD (LI; SALARI, 1995), the same key ideas can be explored for other distortion metrics. For instance, Brunig and Niehsen (2001) show how Successive Elimination Algorithm (SEA) can be formulated for using the MSE and thus, can be extended for Sum of Squared Differences (SSD). The idea of partial computations and Partial Distortion Elimination (PDE) can be also easily extended for SSD. On the other hand, to our best knowledge, there is no work in the literature that efficiently exploited such methods in the context of SATD. Considering PDE for instance, the transform must be computed prior to the sum. Thus, although the same formulation can be applied, PDE is not able to reduce the SATD complexity since it struggles with the complexity of the transform. The same reasoning holds for the elimination of impossible candidates. Therefore, the efficient exploitation of these methods to reduce the SATD complexity presents a room for novelty.

Besides that, even though not being normative, the SATD is a key artifact for reducing the overall encoder complexity[53] (see Figure 17). The overhead of adopting SATD instead of SAD as distortion metric, presented in Section 2.3, is expected when considering the larger number of operations required by the former metric (Table 4). Moreover, the use of a structure called Transpose Buffer (TB) also demands a large share of energy consumption (see Note 36). In this chapter, we introduce those three techniques mentioned, which are usually adopted as means to reduce the number of operations in BMA when the SAD is used as distortion metric. Moreover, as contribution of this chapter, we show how such techniques can be adapted for using SATD, also aiming to reduce the number of operations with no impact in coding efficiency. The underlying assumption is that "Determining the SATD quicker provides the benefits of better coding performance without suffering the drawbacks of longer computation times" (KRISHNAN, 2011). The proposed techniques are evaluated in terms of **effectiveness**, **simplicity**, and % of operations saved (**savings**, for short).

The first technique is the calculation reuse, which is presented in Section 4.1. Calculation reuse is a technique commonly adopted for hardware implementations of VBSME that adopt the SAD as distortion metric[54]. In the case of SATD, the key idea is to reuse the transform computation of small blocks to compute the transform of the larger ones. In other words, through calculation reuse we avoid computing the same part of the transform twice. Because the same operations are performed, there is no coding efficiency losses. This technique has no overhead also, so it is the most promising one, in terms of **simplicity**. Moreover, Section 4.1 brings as contribution a formal proof on how to re-use already computed HTs to save operations, and thus save energy, when computing the SATD of larger blocks that overlap smaller ones. Such technique will be then exploited in Chapter 5 to create a area- and energy-efficient SATD hardware architecture.

The remaining two techniques avoid redundant operations by skipping (stopping) the

[53] Although the distortion metric used in BMA during encoding is not normative, the SATD is able to increase the coding efficiency and thus its use is recommended (CORREA et al., 2012), as presented in Chapter 3.

[54] C.f. mentioned in p. 85.

cost computation over a candidate block as soon as the technique detects that such candidate will not be the one that minimizes the *cost* (Equation 2.3). Because the technique has to determine when this condition happens, there is an underlying overhead, which depends upon the **simplicity** of the technique. The savings provided by these techniques are directly dependent on the number of evaluated candidates and how many operations can be saved when a candidate is eliminated. Both techniques present no coding efficiency losses because a skip only occurs if it is guaranteed that a block will not minimize *cost* and thus the same reference block will be chosen.

Section 4.2 presents the elimination of impossible candidates as the second technique in this chapter. It seeks to eliminate the candidates before the actual *cost* is computed for such candidate according to an elimination criterion which should be simpler than the *cost*. On the other hand, the third technique is the PDE, which saves operations by early-terminating the *cost* computation. Section 4.3 presents PDE in SATD calculation.

4.1 CALCULATION REUSE

This section contains results of our works published in LASCAS2018 and TCAS-I2019[55].

When variable block sizes are considered during encoding, a possible approach to reduce redundant operations is by means of calculation reuse. Such reuse is feasible because larger original blocks are composed by smaller ones, as illustrated in Figure 26.

In Figure 26 is also possible to see that the smaller neighboring original blocks share part of their search window[56]. This overlap of the search windows is exploited to reduce memory read costs during IME, such as in the Level C+ memory reuse scheme (CHEN, C.-Y. et al., 2006). Moreover, besides the reduced memory costs, it is possible the reuse the computed distortion values, such as the SADs.

4.1.1 Reusing SADs

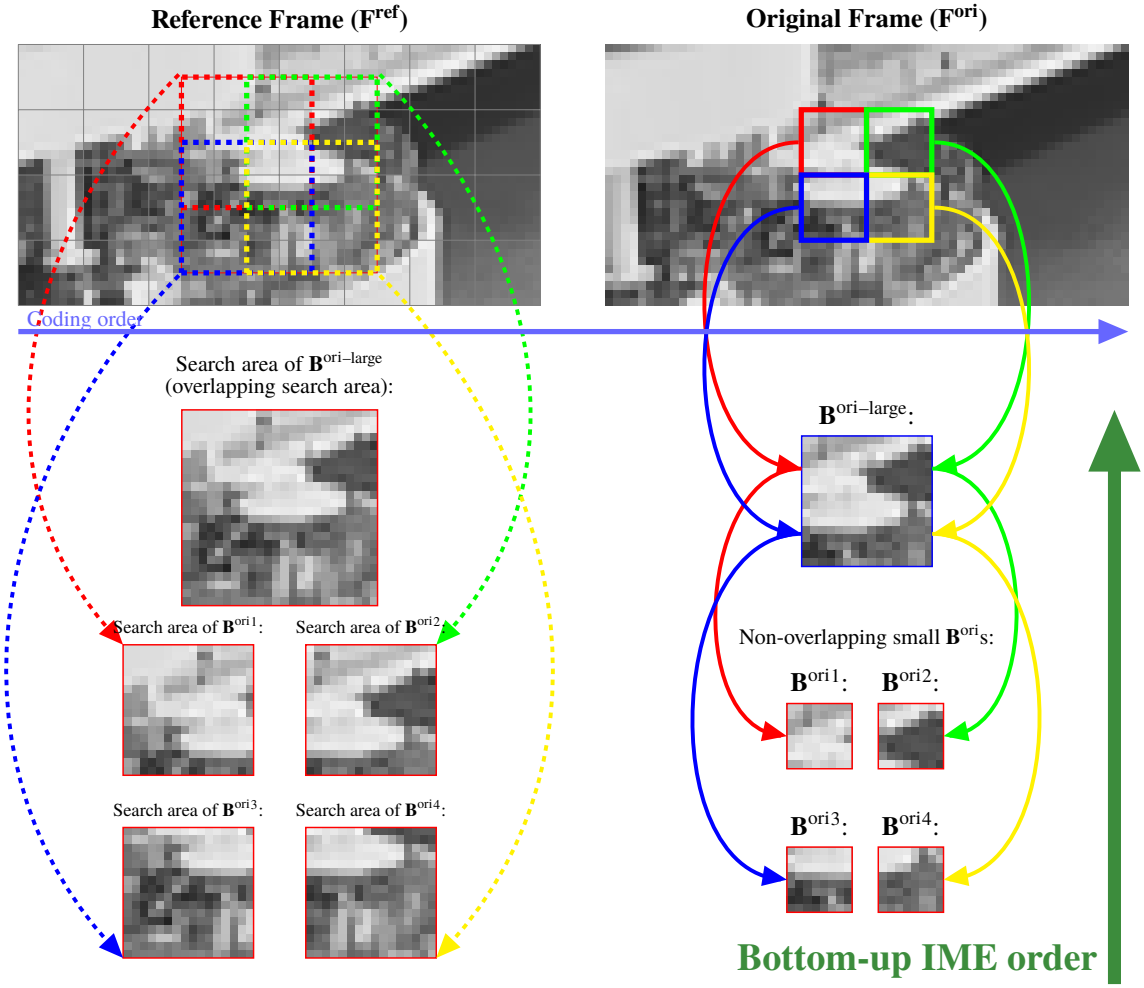
The reuse is straightforward for the SAD (KIM; PARK, 2009) given its recursive nature. Let us define $\mathbf{D}_{m\times n;m'\times n'}$ as the differences block (Equation 2.4) formed by the composition of neighbor $\mathbf{D}_{m'\times n'}$, where $p = m/m'$ and $q = n/n'$:

$$\mathbf{D}_{m\times n;m'\times n'} = \left[\begin{array}{c|c|c} \mathbf{D}_{;1,1} & \cdots & \mathbf{D}_{;1,q} \\ \hline \vdots & \ddots & \vdots \\ \hline \mathbf{D}_{;p,1} & \cdots & \mathbf{D}_{;p,q} \end{array}\right] \tag{4.1}$$

[55] Monteiro, Seidel, and Güntzel (2018) and Seidel, Monteiro, et al. (2019).

[56] However, this "regularity" is only ensured if the search window is centralized in the co-localized position. I.e., the center of the search window in the candidate frame is in the same position of the center of the original block in the original frame. L. Trudeau, S. Coulombe, and C. Desrosiers (2016, p. 2004) named this requirement as "constant search area assumption".

Figure 26 – Example of bottom-up IME order. Performing IME of the smaller partitions first enables the reuse of already computed distortions. This is only possible because the larger partition is formed by the smaller ones and they share a common search area and hence the same neighboring candidate blocks.



Source: the author. Frames from "BQTerrace" sequence (BOSSEN, 2012).

In Equation 4.1, $\mathbf{D}_{m\times n;m'\times n'}$ is the differences matrix from a large partition and each sub-partition $\mathbf{D}_{;i,j}$ is a differences matrix of a smaller neighboring block. By using SAD as *distortion* during VBSME it is possible to re-use previously computed SADs from smaller partitions to compose SADs of larger ones (KIM; PARK, 2009), as follows:

$$sa\left(\mathbf{D}_{m\times n;m'\times n'}\right) = \sum_{i=1}^{p=\frac{m}{m'}} \sum_{j=1}^{q=\frac{n}{n'}} \underbrace{sa\left(\mathbf{D}_{;i,j}\right)}_{\substack{\text{previously}\\ \text{computed}}} \tag{4.2}$$

By Equation A.9, Equation 4.2 can be also written as:

$$sa(\mathbf{D}_{m\times n}) = s\left(\mathbf{SA}\left(\mathbf{D}_{m\times n;m'\times n'}\right)\right) \tag{4.3}$$

Notice that if $m' = n' = 1$, then each partition is an element and Equation 4.2 is the proper SAD definition (Equation 2.5). Given that the blocks are most commonly divided into

two or four partitions (Figure 12), that may in turn be subdivided, the most common reuse is of two or four partitions, which may also be reused in an even larger block. Equations 4.4 to 4.8 show a simple example with a $\mathbf{D}_{2^{n+1};2^n}$ where $n = 1$, thus reusing four partitions.

$$sa\left(\begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{bmatrix}\right) = s\left(\mathbf{SA}\left(\left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ \hline 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{array}\right]\right)\right) \tag{4.4}$$

$$= s\left(\begin{bmatrix} sa\left(\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right) & sa\left(\begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}\right) \\ sa\left(\begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}\right) & sa\left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\right) \end{bmatrix}\right) \tag{4.5}$$

$$= sa\left(\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right) + sa\left(\begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}\right) \tag{4.6}$$

$$+ sa\left(\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\right) + sa\left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\right) \tag{4.7}$$

$$= 10 + 10 + 6 + 6 = 32 \tag{4.8}$$

There is also a similarity among such reuse formulation (Equation 4.2) and the block composition used to compute the SATD of non-square and non-power-of-two partitions (Equation 2.21 and Algorithm 2). However, due to the HT, the reuse of SATDs is not as straightforward when the computed transform is the same size of the block. This is the case in HM when $n = 2$ or $n = 4$ in Equation 2.16.

4.1.2 Reuse in the SATD: reusing HTs

Considering the example from Equations 4.4 to 4.8, with a $\mathbf{D}_{2^{n+1};2^n}$ where $n = 1$. Let us first compute the sum of the SATDs of the small partitions:

$$s\left(\begin{bmatrix} sa\left(\mathbf{T}\left(\begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix}\right)\right) & sa\left(\mathbf{T}\left(\begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}\right)\right) \\ sa\left(\mathbf{T}\left(\begin{bmatrix} 3 & 4 \\ 2 & 1 \end{bmatrix}\right)\right) & sa\left(\mathbf{T}\left(\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\right)\right) \end{bmatrix}\right) = sa\left(\begin{bmatrix} 10 & 0 \\ 4 & 2 \end{bmatrix}\right) + sa\left(\begin{bmatrix} 10 & 0 \\ 4 & 2 \end{bmatrix}\right) \tag{4.9}$$

$$+ sa\left(\begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix}\right) + sa\left(\begin{bmatrix} 6 & 0 \\ 0 & 2 \end{bmatrix}\right)$$

$$= 16 + 16 + 8 + 8 = 48 \tag{4.10}$$

Now, let us compute the SATD of the larger partition:

$$sa\left(\mathbf{T}\left(\begin{bmatrix}1&2&3&4\\4&3&2&1\\1&2&2&1\\2&1&1&2\end{bmatrix}\right)\right) = sa\left(\begin{bmatrix}32&0&0&0\\0&4&8&4\\8&0&0&0\\0&4&8&4\end{bmatrix}\right) = 72 \tag{4.11}$$

The resulting sum in Equation 4.10, 48, is different from the result in Equation 4.11, 72. Therefore, as can be seem by this example, the SATD of the larger partition is not simply the sum of the SATDs of the smaller partitions. Because the HT, one does not simply reuse the SATDs.

Yet, as many other transforms, the HT is recursive (Equation 2.18). The recursive property is often explored because it allows for resource sharing among different size transforms and thus to reduce the area of this kind of hardware architectures (LIU; ZHOU, et al., 2011; HE et al., 2015b; GOEBEL et al., 2016; BONATTO et al., 2017; SILVEIRA; FERREIRA, et al., 2017). Moreover, such property may be adopted to partially achieve calculation reuse in the SATD, at least for the transform part (HE et al., 2015b), which has the largest amount of operations anyway (Table 4). Thus, nevertheless the impossibility of reusing SATDs, we will show that it is possible to re-use already computed HTs of smaller partitions. Such reuse is able to save several operations, since, as just mentioned, most of SATD complexity lies on HT computation.

In both examples in this section, the differences $\mathbf{D}_{2^{n+1};2^n}$ is formed by the composition of four neighbor $\mathbf{D}_{m' \times n'}$, as follows:

$$\mathbf{D}_{2^{n+1};2^n} = \left[\begin{array}{c|c}\mathbf{D}_{;1,1} & \mathbf{D}_{;1,2}\\\hline \mathbf{D}_{;2,1} & \mathbf{D}_{;2,2}\end{array}\right] \tag{4.12}$$

Theorem 1

Let $\mathbf{R}_{2^{n+1}}$ *be the matrix of re-usable HTs:*

$$\mathbf{R}_{2^{n+1};2^n} = \left[\begin{array}{c|c}\mathbf{T}(\mathbf{D}_{;1,1}) & \mathbf{T}(\mathbf{D}_{;1,2})\\\hline \mathbf{T}(\mathbf{D}_{;2,1}) & \mathbf{T}(\mathbf{D}_{;2,2})\end{array}\right] \tag{4.13}$$

Then, $\mathbf{T}(\mathbf{D}_{2^{n+1}})$ *can be obtained as:*

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) = \mathbf{H}_2 \times \mathbf{R}_{2^{n+1};2^n} \times \mathbf{H}_2 \tag{4.14}$$

Proof. We will prove by contradiction that Theorem 1 holds. Thus, first we assume that:

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) \neq \mathbf{H}_2 \times \mathbf{R}_{2^{n+1};2^n} \times \mathbf{H}_2 \tag{4.15}$$

Now, we can rewrite Equation 4.15 using the definition of $\mathbf{H}_2$ (Equation 2.18).

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) \neq \underbrace{\begin{bmatrix}1&1\\1&-1\end{bmatrix} \times \left[\begin{array}{c|c}\mathbf{R}_{;1,1} & \mathbf{R}_{;1,2}\\\hline \mathbf{R}_{;2,1} & \mathbf{R}_{;2,2}\end{array}\right] \times \begin{bmatrix}1&1\\1&-1\end{bmatrix}}_{\mathbf{P}_{2^{n+1};2^n}} \tag{4.16}$$

Solving the right hand side of Equation 4.16, using Equation A.4 and Equation A.5 respectively for the left and rightmost multiplications, gives rise to $\mathbf{P}_{2^{n+1};2^n}$, where each partition is:

$$\mathbf{P}_{;1,1} = [\mathbf{R}_{;1,1} + \mathbf{R}_{;2,1}] + [\mathbf{R}_{;1,2} + \mathbf{R}_{;2,2}] \tag{4.17}$$

$$\mathbf{P}_{;1,2} = [\mathbf{R}_{;1,1} + \mathbf{R}_{;2,1}] - [\mathbf{R}_{;1,2} + \mathbf{R}_{;2,2}] \tag{4.18}$$

$$\mathbf{P}_{;2,1} = [\mathbf{R}_{;1,1} - \mathbf{R}_{;2,1}] + [\mathbf{R}_{;1,2} - \mathbf{R}_{;2,2}] \tag{4.19}$$

$$\mathbf{P}_{;2,2} = [\mathbf{R}_{;1,1} - \mathbf{R}_{;2,1}] - [\mathbf{R}_{;1,2} - \mathbf{R}_{;2,2}] \tag{4.20}$$

Changing each $\mathbf{R}_{;i,j}$ by their definition (Equation 4.13):

$$\mathbf{P}_{;1,1} = [\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1})] + [\mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})] \tag{4.21}$$

$$\mathbf{P}_{;1,2} = [\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1})] - [\mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})] \tag{4.22}$$

$$\mathbf{P}_{;2,1} = [\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1})] + [\mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})] \tag{4.23}$$

$$\mathbf{P}_{;2,2} = [\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1})] - [\mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})] \tag{4.24}$$

Now, let us expand the left hand side of Equation 4.15. By the definition of Hadamard Transform (Equation 2.17), we know Equation 4.25 to be true.

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) = \mathbf{H}_{2^{n+1}} \times \mathbf{D}_{2^{n+1}} \times \mathbf{H}_{2^{n+1}} \tag{4.25}$$

The leftmost $\mathbf{H}_{2^{n+1}}$ can be replaced by its definition (Equation 2.18):

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) = \underbrace{\left[\begin{array}{c|c} \mathbf{H}_{2^n} & \mathbf{H}_{2^n} \\ \hline \mathbf{H}_{2^n} & -\mathbf{H}_{2^n} \end{array}\right] \times \left[\begin{array}{c|c} \mathbf{D}_{;1,1} & \mathbf{D}_{;1,2} \\ \hline \mathbf{D}_{;2,1} & \mathbf{D}_{;2,2} \end{array}\right]}_{\mathbf{F}_{2^{n+1};2^n}} \times \mathbf{H}_{2^{n+1}} \tag{4.26}$$

Solving first the leftmost block matrix multiplication results in the matrix $\mathbf{F}_{2^{n+1};2^n}$:

$$\mathbf{F} = \left[\begin{array}{c|c} \mathbf{H}_{2^n} \times \mathbf{D}_{;1,1} + \mathbf{H}_{2^n} \times \mathbf{D}_{;2,1} & \mathbf{H}_{2^n} \times \mathbf{D}_{;1,2} + \mathbf{H}_{2^n} \times \mathbf{D}_{;2,2} \\ \hline \mathbf{H}_{2^n} \times \mathbf{D}_{;1,1} - \mathbf{H}_{2^n} \times \mathbf{D}_{;2,1} & \mathbf{H}_{2^n} \times \mathbf{D}_{;1,2} - \mathbf{H}_{2^n} \times \mathbf{D}_{;2,2} \end{array}\right] \tag{4.27}$$

We can now substitute $\mathbf{F}$ in Equation 4.26 and the rightmost $\mathbf{H}_{2^{n+1}}$ by its definition (Equation 2.18), as shown in Equation 4.28. Solving the remaining block matrix multiplication results in the matrix $\mathbf{S}_{2^{n+1};2^n}$.

$$\mathbf{T}(\mathbf{D}_{2^{n+1}}) = \underbrace{\left[\begin{array}{c|c} \mathbf{F}_{2^n} & \mathbf{F}_{2^n} \\ \hline \mathbf{F}_{2^n} & \mathbf{F}_{2^n} \end{array}\right] \times \left[\begin{array}{c|c} \mathbf{H}_{2^n} & \mathbf{H}_{2^n} \\ \hline \mathbf{H}_{2^n} & -\mathbf{H}_{2^n} \end{array}\right]}_{\mathbf{S}_{2^{n+1};2^n}} \tag{4.28}$$

The resulting partitions of $\mathbf{S}_{2^{n+1};2^n}$ are as follows:

$$\begin{aligned}\mathbf{S}_{;1,1} =&[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,1}\times\mathbf{H}_{2^n})+(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,1}\times\mathbf{H}_{2^n})]\\ &+[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,2}\times\mathbf{H}_{2^n})+(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,2}\times\mathbf{H}_{2^n})] \quad (4.29)\\ \mathbf{S}_{;1,2} =&[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,1}\times\mathbf{H}_{2^n})+(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,1}\times\mathbf{H}_{2^n})]\\ &-[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,2}\times\mathbf{H}_{2^n})+(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,2}\times\mathbf{H}_{2^n})] \quad (4.30)\\ \mathbf{S}_{;2,1} =&[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,1}\times\mathbf{H}_{2^n})-(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,1}\times\mathbf{H}_{2^n})]\\ &+[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,2}\times\mathbf{H}_{2^n})-(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,2}\times\mathbf{H}_{2^n})] \quad (4.31)\\ \mathbf{S}_{;2,2} =&[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,1}\times\mathbf{H}_{2^n})-(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,1}\times\mathbf{H}_{2^n})]\\ &-[(\mathbf{H}_{2^n}\times\mathbf{D}_{;1,2}\times\mathbf{H}_{2^n})-(\mathbf{H}_{2^n}\times\mathbf{D}_{;2,2}\times\mathbf{H}_{2^n})] \quad (4.32)\end{aligned}$$

Applying the Hadamard transform definition (Equation 2.17), in Equations 4.29-4.32 results in:

$$\mathbf{S}_{;1,1} = [\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1})]+[\mathbf{T}(\mathbf{D}_{;1,2})+\mathbf{T}(\mathbf{D}_{;2,2})] \quad (4.33)$$

$$\mathbf{S}_{;1,2} = [\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1})]-[\mathbf{T}(\mathbf{D}_{;1,2})+\mathbf{T}(\mathbf{D}_{;2,2})] \quad (4.34)$$

$$\mathbf{S}_{;2,1} = [\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1})]+[\mathbf{T}(\mathbf{D}_{;1,2})-\mathbf{T}(\mathbf{D}_{;2,2})] \quad (4.35)$$

$$\mathbf{S}_{;2,2} = [\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1})]-[\mathbf{T}(\mathbf{D}_{;1,2})-\mathbf{T}(\mathbf{D}_{;2,2})] \quad (4.36)$$

Comparing Equations 4.21-4.24 to Equations 4.33-4.36 one can conclude that:

$$\mathbf{P}_{2^{n+1};2^n} = \left[\begin{array}{c|c}\mathbf{P}_{;1,1} & \mathbf{P}_{;1,2}\\ \hline \mathbf{P}_{;2,1} & \mathbf{P}_{;2,2}\end{array}\right] = \left[\begin{array}{c|c}\mathbf{S}_{;1,1} & \mathbf{S}_{;1,2}\\ \hline \mathbf{S}_{;2,1} & \mathbf{S}_{;2,2}\end{array}\right] = \mathbf{S}_{2^{n+1};2^n} \quad (4.37)$$

By Equation 4.16, $\mathbf{T}(\mathbf{D}_{2^{n+1}}) \neq \mathbf{P}_{2^{n+1};2^n}$. However, by Equation 4.28, $\mathbf{T}(\mathbf{D}_{2^{n+1}}) = \mathbf{S}_{2^{n+1};2^n}$, which contradicts Equation 4.37, where $\mathbf{P}_{2^{n+1};2^n} = \mathbf{S}_{2^{n+1};2^n}$. Thus, Equation 4.14 holds. ■

By re-using four HTs, instead of requiring $(n+1)\times 2^{2\times n+3}$ operations, $\mathbf{T}(\mathbf{D}_{2^{n+1}})$ requires only $2^{2\times n+3}$ extra operations. Table 14 summarizes the operation savings provided by the reuse of HTs.

Table 14 – Number of saved operations (**savings**) – Large re-using small

small		$\mathbf{T}(\mathbf{D}_{2^n})$	large	$\mathbf{T}(\mathbf{D}_{2^{n+1}})$	remaining	saved
n	2^n	$n2^{2n+1}$	$n+1$	$(n+1)2^{2n+3}$	2^{2n+3}	$n2^{2n+3}$
2	4	64	3	384	128	**256**
3	8	384	4	2048	512	**1536**
4	16	2048	5	10240	2048	**8192**

Notice that there is no overhead (in terms of number of operations) for this technique, thus it is the most simple possible. Its effectiveness is constant for a pair of large and small sizes, as shown in Table 14. The only limitation is that a bottom-up approach must be used, if

this technique is to be adopted. Yet, there are other applications for the reuse, like evaluating multiple SATD sizes for the same ($\mathbf{B}^{\text{ori}}$, $\mathbf{B}^{\text{can}}$) pair. The ideia is to detect the best transform size. Because its relation to the transform step in encoding (Figure 11), the SATD may be used to determine if it is possible to bypass transform and quantization to save energy (DEGUCHI; TREE, 2001). Nevertheless the application, we provided an initial view on how one can reuse HTs for SATD, thus saving in terms of number of operations.

4.2 ELIMINATION OF IMPOSSIBLE CANDIDATES

The main contribution of this section is the proposal of a method to eliminate impossible candidates when using Hadamard-based SATD. With that method it is possible to reduce the computational complexity with no cost in BD-Rate, i.e, without compromising coding efficiency. The proposed method is a formulation of SEA to use the SATD instead of the SAD.

4.2.1 The Successive Elimination Algorithm (SEA)

The most known method for elimination of impossible candidates is the SEA. SEA is based on the following properties:

Property 1
Subadditivity. *A given function $f : A \rightarrow B$ is said to be subadditive when it presents the following property:*

$$\forall x, y \in A, f(x+y) \leq f(x) + f(y) \tag{4.38}$$

Property 2 (Schechter (1996))
Subaditivity of the absolute value. *The absolute value function is subadditive, thus:*

$$|a+b| \leq |a| + |b| \tag{4.39}$$

Property 2 can be used in the context of the SAD. Considering pair of original and candidate blocks with two pixels each so that their difference matrix ($\mathbf{D}$) have two values, a and b. Let us assume a BMA execution as presented in Algorithm 1. If $sad^{\min} \leq |a+b|$, then, by Equation 4.39, $sad^{\min} \leq |a| + |b| = sa(\mathbf{D})$. This means that the current candidate block will not be selected as reference. More generally, the so-called Absolute Difference of Sums (ADS)[57] is defined as:

$$ads\left(\mathbf{B}^{\text{ori}}, \mathbf{B}^{\text{can}}\right) = |s\left(\mathbf{B}^{\text{ori}}\right) - s(\mathbf{B}^{\text{can}})| \tag{4.40}$$

[57] Also known as projected sum absolute differences (CAI; PAN, 2010).

By Property 4 and Equation 2.4, Equation 4.40 can be written as:

$$ads\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right) = \left|s\left(\mathbf{B}^{\text{ori}} - \mathbf{B}^{\text{can}}\right)\right| = \left|s(\mathbf{D})\right| \tag{4.41}$$

After Property 1 , Equation 4.42, using the Absolute Difference of Sums (ADS) holds true (TRUDEAU, Luc; COULOMBE, Stephane; DESROSIERS, Christian, 2015).

$$ads\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right) = \left|\sum_{i=1}^{m}\sum_{j=1}^{n} d_{i,j}\right| \leq \sum_{i=1}^{m}\sum_{j=1}^{n}\left|d_{i,j}\right| = sa\left(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}\right) = sa(\mathbf{D}) \tag{4.42}$$

As in the example, let $sad^{\min}$ be the current minimum SAD of a sequential BMA execution (Algorithm 1). The SEA skips the calculation of SAD with a given $\mathbf{B}^{\text{can}}$ when its ADS is equal or larger than $sad^{\min}$ because, as shown in Equation 4.42, its SAD is guaranteed to be larger than $sad^{\min}$. Such $\mathbf{B}^{\text{can}}$ is called an impossible candidate, once it will not be able to minimize the SAD and will not be selected by BMA (Equation 2.3) as $\mathbf{B}^{\text{ref}}$. Therefore, skipping such $\mathbf{B}^{\text{can}}$ does not reduce the coding efficiency. Such transitive elimination (TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C., 2018, p. 924) gives rise to the SEA, presented in Algorithm 5.

Algorithm 5: Successive Elimination Algorithm (SEA)

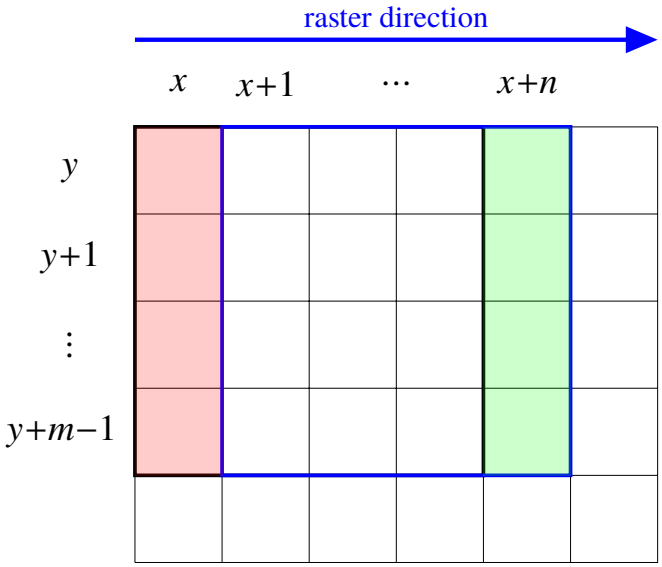
```
Input  : B^ori, S
Output : B^ref
:
1  sad^min ← ∞;
2  foreach B^can ∈ S do
3      D = B^ori − B^can;
4      ads ← |s(D)|;
5      if ads < sad^min then
6          sad^curr ← sa(D);
7          if sad^curr < sad^min then
8              sad^min ← sad^curr ;
9              B^ref ← B^can;
10         end
11     end
12 end
```

However, there seems to be a very small advantage computing the ADS instead of the SAD. The number of sums is basically the same, only a smaller number of absolute value computations will be performed when computing the ADS. Moreover, whenever the ADS is smaller than the $sad^{\min}$ a SAD must be computed to be compared with $sad^{\min}$.

The advantage of SEA lies in a fast computation of the ADS that is possible for neighboring candidate blocks. In this case, each ADS can be obtained with only $2^{n+1}+1$ operations (LI; SALARI, 1995), instead of $2^{2n+1}-1$ operations of an SAD. Figure 27 illustrates

the fast computation of the ADS. Notice that such a fast estimation can only be obtained for a raster-like search order[58]. Fast search algorithms would not benefit much from the SEA because the overhead of computing ADS.

Figure 27 – Efficient method for the computation of successive ADSs of neighbor candidate blocks. Each square represents a pixel. Firstly, it is necessary to obtain the $m \times n$ ADS of the block in position (x, y). Then, the next ADS in the same line can be obtained by adding the pixel values of the next column (rightmost filled) and subtracting the ones from the first row (leftmost filled). Such process can be repeated until horizontal limit of the search area.



Source: the author.

As shown in Figure 27, the number of operations required to compute the ADS depends on the side size of the block which is orthogonal to the raster direction. In the figure, the such side has m pixels. Thus, m additions and m subtractions are needed, after the first Absolute First Difference (AFD) computation, to obtain the sum of the new candidate block. A pair of subtractions is needed to compute the absolute difference[59] with the $s(\mathbf{B}^{\text{ori}})$, as in Equation 4.40. Another subtraction is needed to compare with $sad^{\min}$. Thus, the overhead of SEA is $2{\times}m{+}3$ operations. Considering only blocks with size $2^n \times 2^n$, each $\mathbf{B}^{\text{can}} \in S$ have an overhead of $2^{n+1}{+}3$ operations. If a skip occurs, $3{\times}2^{2n}{-}1$ operations from the SAD (Table 4) can be saved. According to Li and Salari (1995, p. 106), SEA is able to reduce in 85% the number of SAD calculations when compared to FBMA.

So far we presented the SEA algorithm considering only the distortion in BMA. Coban and Mersereau (1998) included the $\lambda \times rate$ term from Equation 2.6 in the SEA formulation. The rate-constrained SEA is almost the same, but the rate cost is added to ADS and instead of comparing to $sad^{\min}$, now the elimination criterion is compared to $j^{\min}$. Because the rate term is required for computing j^{curr}, we can assume that the multiplication has no extra cost. Thus, only one new sum is added to the overhead of ADS.

[58] Some examples are raster, snake, and spiral search orders. Some search orders are presented in Figure 29.

[59] Chapter 5 presents a hardware datapath for the absolute difference operation with only two subtractions and a multiplexer.

The number of saved operations can be obtained as follows. As previously defined, S is the set of all candidate blocks. Let I be the set of candidates for which was possible to skip the SAD, such that $I \subset S$. The **effectiveness** of SEA (ef^{sea}) can be defined as:

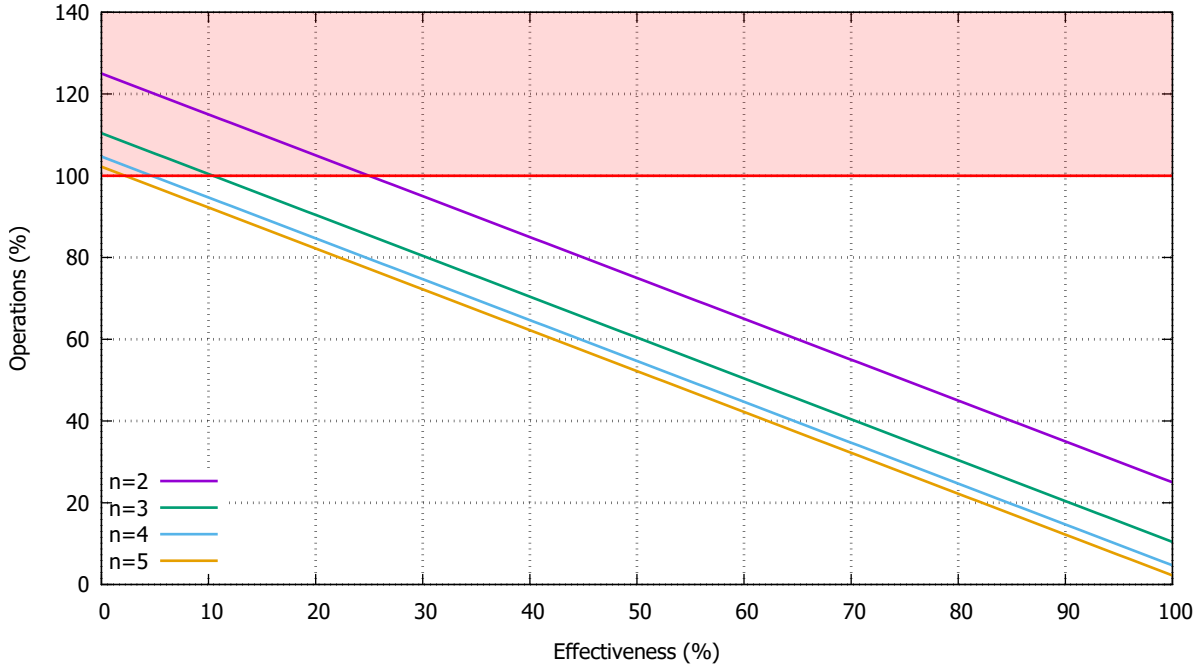
$$ef^{\text{sea}} = \frac{|I|}{|S|} \tag{4.43}$$

The relative number of operations (%) when using SEA (ro^{sea}), with respect to the BMA without ADS, depends on the effectiveness and can be defined as:

$$ro^{\text{sea}} = \frac{ef^{\text{sea}} \times \overbrace{(2^{n+1}+4)}^{\text{ADS overhead}} + (1-ef^{\text{sea}}) \times (\overbrace{3\times 2^{2n}}^{\text{SAD operations}} + \overbrace{2^{n+1}+4}^{\text{ADS overhead}})}{\underbrace{3\times 2^{2n}}_{\text{SAD operations}}} \times 100\% \tag{4.44}$$

Figure 28 presents a plot of the SEA savings relative to the effectiveness. Notice that if the effectiveness is too low, instead of saving operations, using SEA may require more operations than not using it. For instance, if less than 10% of the SADs of 4×4 blocks are skipped, the SEA would require more than 10% extra operations would be executed. Hence the importance of reducing the overheads and maximizing the effectiveness.

Figure 28 – Relative number of executed operations in SEA with respect to a BMA without SEA, as a function of its effectiveness in skipping candidates. The horizontal line in 100% delimits the region of overheads, i.e., the shaded region where the use of SEA requires more operations than a simple BMA.



Source: the author.

A possible way to reduce the overhead, proposed by Li and Salari (1995, p. 106), is to compute the sums of candidate blocks for the whole frame beforehand and at once. This brings two main benefits:

1. Saves operations by not computing again the sum of overlapping candidate blocks for different original blocks (see Figure 26);

2. The whole frame calculations can be done following a raster order and the sum values can be accessed at random, thus enabling the use of SEA along with other fast search algorithms.

So, disregarding the initial overhead, for each candidate the only overheads are the ADS sum with the rate estimate and comparison with the current best cost.

To maximize effectiveness, the ideal case would be to first evaluate the best matching candidate[60] and then to eliminate all other candidates. Therefore, the faster the value of $j^{\min}$ is reduced, the more candidates are likely to be eliminated (CAI; PAN, 2010, p. 1191). One strategy is to adopt a better search order. Figure 29 shows some search order examples.

Figure 29 – Examples of search order. The number indicate the evaluation order of the candidate blocks. Each square represents 1 pixel and the shaded one represents the search area center. (a) Raster search order. (b) Snake search order (c) Spiral search order (d) JM search order (e) Rate-constrained search order

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

(a)

0	1	2	3	4
9	8	7	6	5
10	11	12	13	14
19	18	17	16	15
20	21	22	23	24

(b)

20	19	18	17	16
21	6	5	4	15
22	7	0	3	14
23	8	1	2	13
24	9	10	11	12

(c)

23	10	12	14	24
21	7	2	8	22
19	5	0	6	20
17	3	1	4	18
15	9	11	13	16

(d)

22	14	6	15	24
18	10	2	12	17
7	3	0	1	5
19	11	4	9	13
23	16	8	20	21

(e)

Source: (b) the author; (a)(c)(e) (TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C., 2014, p. 3177); (c)(d) (TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C., 2018, p. 924).

[60] Such candidate has the lowest possible threshold and thus the highest possibility to eliminate the other ones (CAI; PAN, 2010, p. 1191).

The spiral search order (Figure 29c) consists in initializing the search at the center and expanding to the borders following an spiral-like path. The search order used in JM (Figure 29d) is also a kind of spiral search order, as it starts at the center and expands towards the borders. For instance, Coban and Mersereau (1998, p. 771) centralized the search area (and thus the spiral) at the zero MV, i.e., the co-localized $\mathbf{B}^{\text{can}}$. The underlying assumption is that the possibility of a candidate minimizing the cost decreases proportionally to the magnitude of the motion vector (CAI; PAN, 2010, p. 1191). Such assumption can be verified by analyzing the position of the reference blocks with respect to the search area center. Gonçalves et al. (2018) illustrates that such assumption holds. A possible explanation is that the rate estimates increase towards the outside of the search area, as can be seen in Figure 15.

L. Trudeau, S. Coulombe, and C. Desrosiers (2014) exploit the fact that rate cost is based on exponential Golomb codes for MVs (Equation 2.13), and define a search order that is rate-constrained (Figure 29e). In such search oder, the candidates are ordered by their estimated rates, just as shown in Figure 15. Luc Trudeau, Stephane Coulombe, and Christian Desrosiers (2015, p. 209) define the concept of SEA-optimal: only those blocks that have their rate-constrained ADS lower than the j^{cost} of the chosen reference block are evaluated using the SAD. Luc Trudeau, Stephane Coulombe, and Christian Desrosiers (2015) then propose a new search algorithm that improves the effectiveness by ordering the candidates by their rate-constrained ADS before the actual search. By doing so, they ensure that candidates with rate-constrained ADS larger than the j^{cost} of the soon-to-be reference block are placed after such blocks in the evaluation order. Therefore, their algorithm is SEA-optimal.

Even tough already reaching optimality, considering the search order, L. Trudeau, S. Coulombe, and C. Desrosiers (2016) proposed a SEA-based algorithm for VBSME that deals with the reuse of SADs of smaller partitions (as presented in Section 4.1.1). Under a constant search area assumption (see Note 56) it is possible to use the sum of the minimum SADs fo the smaller partitions as a lower bound to the best SADs of the larger one. Compared to their algorithm in (TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C., 2014), such new approach saved 20% of the SAD computations.

Although such myriad of works exploit the SEA, we only found such formulation for using the SAD. In the next section we will present our proposal of the rate-constrained SEA considering the SATD as distortion metric.

4.2.2 Rate-constrained single-level SEA using SATD

The present subsection contains results published in ICIP2016[61].

Although SEA is originally defined considering the SAD as distortion metric, it is possible to formulate an SATD-based SEA, as it will be presented in the sequel. By following the same arguments for the definition of ADS in the SAD-based SEA, which relies on Property

[61] Seidel, Cancellier, et al. (2016).

1, we propose a new metric, named AFD, calculated as:

$$af(\mathbf{D}_{2^n}) = \frac{1}{2^{n-1}} \times |s(\mathbf{T}(\mathbf{D}_{2^n}))| = \frac{1}{2^{n-1}} \times \left| \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} td_{i,j} \right| \tag{4.45}$$

By such definition, the AFD can be used on the right-hand side of Equation 4.39. Similarly to Equation 4.42, the following holds true (after Property 1):

$$af(\mathbf{D}_{2^n}) = \frac{1}{2^{n-1}} \times \left| \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} td_{i,j} \right| \leq \frac{1}{2^{n-1}} \times \sum_{i=1}^{2^n} \sum_{j=1}^{2^n} \left| td_{i,j} \right| = sa(\mathbf{T}(\mathbf{D}_{2^n})) \tag{4.46}$$

Therefore, the AFD can be used to eliminate impossible candidates when SATD is applied during BMA in the same manner as ADS is used when SAD is applied. Nevertheless, the definition of AFD in Equation 4.45 still requires the HT calculation. Thus, we need to find an expression able to simplify the right-hand side of Equation 4.45. Moreover, such a simplification should not require the HT. Ignoring the constant multiplication part ($1/2^{n-1}$ from Equations 2.16 and 4.45) at first, we state the following theorem:

Theorem 2

$$\forall n \in \mathbb{N}^*, \quad s(\mathbf{T}(\mathbf{D}_{2^n})) = 2^{2n} \times d_{1,1} \tag{4.47}$$

Proof. We will prove by induction that Theorem 2 holds.

Base case: If $n = 1$, by

$$s(\mathbf{T}(\mathbf{D}_2)) = (\mathbf{T}(\mathbf{D}_2))_{1,1} + (\mathbf{T}(\mathbf{D}_2))_{1,2} + (\mathbf{T}(\mathbf{D}_2))_{2,1} + (\mathbf{T}(\mathbf{D}_2))_{2,2} \tag{4.48}$$

Using $\mathbf{H}_{2^n} = \mathbf{H}_2$ to transform $\mathbf{D}_2$, we can describe each $\mathbf{T}(\mathbf{D}_2))_{i,j}$ with its $d_{i,j}$ components:

$$\begin{aligned} \mathbf{T}(\mathbf{D}_2)) &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} d_{1,1} & d_{1,2} \\ d_{2,1} & d_{2,2} \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ &= \begin{bmatrix} (d_{1,1} + d_{2,1} + d_{1,2} + d_{2,2}) & (d_{1,1} + d_{2,1} - d_{1,2} - d_{2,2}) \\ (d_{1,1} - d_{2,1} + d_{1,2} - d_{2,2}) & (d_{1,1} - d_{2,1} - d_{1,2} + d_{2,2}) \end{bmatrix} \end{aligned} \tag{4.49}$$

Substituting the right side of Equation 4.48 with the right side of Equation 4.49 gives:

$$\begin{aligned} s(\mathbf{T}(\mathbf{D}_2)) =& (d_{1,1} + d_{2,1} + d_{1,2} + d_{2,2}) + (d_{1,1} + d_{2,1} - d_{1,2} - d_{2,2}) \\ &+ (d_{1,1} - d_{2,1} + d_{1,2} - d_{2,2}) + (d_{1,1} - d_{2,1} - d_{1,2} + d_{2,2}) \end{aligned} \tag{4.50}$$

Except for $d_{1,1}$ all other terms canceled out, thus leading to:

$$\begin{aligned} s(\mathbf{T}(\mathbf{D}_2)) &= d_{1,1} + d_{1,1} + d_{1,1} + d_{1,1} \\ &= 4 \times d_{1,1} \\ &= 2^{2\times 1} \times d_{1,1} \end{aligned} \tag{4.51}$$

Hence, Equation 4.51 shows that Equation 4.47 is true for $n = 1$.

Induction step: Let $n = k + 1$. Then, first we need to obtain $\mathbf{T}(\mathbf{D}_{2^{k+1}})$:

$$s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = s(\mathbf{H}_{2^{k+1}} \times \mathbf{D}_{2^{k+1}} \times \mathbf{H}_{2^{k+1}}) \tag{4.52}$$

By considering the Hadamard transform recursive definition, presented in Equation 2.18, Equation 4.52 can be rewritten using partitioned matrices (ANTON; RORRES, 2010), where $\mathbf{D}_{2^{k+1}}$ is partitioned in four 2^k sized matrices, i.e., $\mathbf{D}_{2^{k+1};2^k}$:

$$s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = s\left(\left[\begin{array}{c|c} \mathbf{H}_{2^k} & \mathbf{H}_{2^k} \\ \hline \mathbf{H}_{2^k} & -\mathbf{H}_{2^k} \end{array}\right] \times \left[\begin{array}{c|c} \mathbf{D}_{;1,1} & \mathbf{D}_{;1,2} \\ \hline \mathbf{D}_{;2,1} & \mathbf{D}_{;2,2} \end{array}\right] \times \left[\begin{array}{c|c} \mathbf{H}_{2^k} & \mathbf{H}_{2^k} \\ \hline \mathbf{H}_{2^k} & -\mathbf{H}_{2^k} \end{array}\right]\right) \tag{4.53}$$

$$= s\left(\left[\begin{array}{c|c} \mathbf{T}(\mathbf{D})_{;1,1} & \mathbf{T}(\mathbf{D})_{;1,2} \\ \hline \mathbf{T}(\mathbf{D})_{;2,1} & \mathbf{T}(\mathbf{D})_{;2,2} \end{array}\right]\right) \tag{4.54}$$

$$= s(\mathbf{T}(\mathbf{D})_{;1,1}) + s(\mathbf{T}(\mathbf{D})_{;1,2}) + s(\mathbf{T}(\mathbf{D})_{;2,1}) + s(\mathbf{T}(\mathbf{D})_{;2,2}) \quad \text{(associativity of } s(\ldots))$$

Solving the block multiplications from Equation 4.53 results in:

$$s(\mathbf{T}(\mathbf{D})_{;1,1}) = s(\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1}) + \mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})) \tag{4.55}$$

$$s(\mathbf{T}(\mathbf{D})_{;1,2}) = s(\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1}) - \mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})) \tag{4.56}$$

$$s(\mathbf{T}(\mathbf{D})_{;2,1}) = s(\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1}) + \mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})) \tag{4.57}$$

$$s(\mathbf{T}(\mathbf{D})_{;2,2}) = s(\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1}) - \mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})) \tag{4.58}$$

Notice the equivalence in the sign patterns between Equations 4.21-4.24 and Equations 4.55-4.58. In fact, besides $s()$ functions, they represent the same step of the computation of a $\mathbf{T}(\mathbf{D}_{2^{k+1}})$, achieved through different formulations. Using the values from Equations 4.55-4.58 results in:

$$\begin{aligned} s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = {} & s(\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1}) + \mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})) \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1}) + \mathbf{T}(\mathbf{D}_{;2,1}) - \mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})) \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1}) + \mathbf{T}(\mathbf{D}_{;1,2}) - \mathbf{T}(\mathbf{D}_{;2,2})) \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1}) - \mathbf{T}(\mathbf{D}_{;2,1}) - \mathbf{T}(\mathbf{D}_{;1,2}) + \mathbf{T}(\mathbf{D}_{;2,2})) \end{aligned} \tag{4.59}$$

Applying Property 4, i.e., the associativity of $s()$) yields:

$$\begin{aligned} s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = {} & s(\mathbf{T}(\mathbf{D}_{;1,1})) + \cancel{s(\mathbf{T}(\mathbf{D}_{;2,1}))} + \cancel{s(\mathbf{T}(\mathbf{D}_{;1,2}))} + \cancel{s(\mathbf{T}(\mathbf{D}_{;2,2}))} \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1})) + \cancel{s(\mathbf{T}(\mathbf{D}_{;2,1}))} - \cancel{s(\mathbf{T}(\mathbf{D}_{;1,2}))} - \cancel{s(\mathbf{T}(\mathbf{D}_{;2,2}))} \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1})) - \cancel{s(\mathbf{T}(\mathbf{D}_{;2,1}))} + \cancel{s(\mathbf{T}(\mathbf{D}_{;1,2}))} - \cancel{s(\mathbf{T}(\mathbf{D}_{;2,2}))} \\ & + s(\mathbf{T}(\mathbf{D}_{;1,1})) - \cancel{s(\mathbf{T}(\mathbf{D}_{;2,1}))} - \cancel{s(\mathbf{T}(\mathbf{D}_{;1,2}))} + \cancel{s(\mathbf{T}(\mathbf{D}_{;2,2}))} \end{aligned} \tag{4.60}$$

Notice that the same sign pattern presented in Equation 4.50 for each $d_{i,j}$ now presents itself for each $s\left(\mathbf{T}\left(\mathbf{D}_{;2,2}\right)\right)$ in Equation 4.60. Thus, we can cancel those elements as done with the ones in Equation 4.50. The remaining elements are:

$$s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right)+s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right)+s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right)+s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right) \tag{4.61}$$

They can be grouped:

$$s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) = 4\times s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right) \tag{4.62}$$

In Equation 4.53 it is possible to observe that each partition $\mathbf{D}_{;i,j}$ has size $2^k\times 2^k$. Also, $\mathbf{D}_{;1,1}$ includes the first element, $d_{1,1}$. By the induction hypothesis, we assume Equation 4.47 true for $n = k$, therefore:

$$s\left(\mathbf{T}\left(\mathbf{D}_{;1,1}\right)\right) = 2^{2k}\times d_{1,1} \tag{4.63}$$

Using Equation 4.63 in Equation 4.62 yields:

$$\begin{aligned} s(\mathbf{T}(\mathbf{D}_{2^{k+1}})) &= 4\times 2^{2k}\times d_{1,1} && (4.64)\\ &= 2^2\times 2^{2k}\times d_{1,1} && (4.65)\\ &= 2^{2+2k}\times d_{1,1} && (4.66)\\ &= 2^{2(k+1)}\times d_{1,1} && (4.67) \end{aligned}$$

This completes the inductive step. By the principle of mathematical induction, Theorem 2 holds.

■

After Theorem 2, Equation 4.45 can be rewritten as Equation 4.68. Notice that AFD calculation is O(1), thus its calculation has a constant complexity and its evaluation poses a small overhead when the SATD calculation cannot be skipped.

$$af(\mathbf{D}_{2^n}) = \frac{2^{2n}}{2^{n-1}}\times|d_{1,1}| \tag{4.68}$$

Before evaluating the AFD effectiveness, our model must be adjusted to explore the full potential of a current video standard. As presented, in a rate-constrained encoder, either using SAD or SATD, the rate cost of selecting a given $\mathbf{B}^{\text{can}}$ must be also considered by the *cost*, along with its distortion. Such rate-constrained cost was previously presented in Equation 2.6. Following the rate-constrained SEA using SAD (COBAN; MERSEREAU, 1998), the relation between AFD and SATD must be considered as:

$$\text{AFD}(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}})+\lambda\times\text{Rate}(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}) \leq \text{SATD}(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}})+\lambda\times\text{Rate}(\mathbf{B}^{\text{ori}},\mathbf{B}^{\text{can}}) \tag{4.69}$$

Also, up to this point, we have considered only square blocks. However, rectangular partitions are also considered in current encoders, such as H.264/AVC and HEVC. Even so, the HT is only defined for square matrices. Thus, the SATD of a rectangular block is defined as the sum of the SATDs of its square partitions. Moreover, the HM adopts the composition of 4×4 or 8×8 sized SATDs even when calculating an SATD with larger block sizes, as shown in Equation 2.21 and Algorithm 2. In the case of AFD, the first element of each partition must be considered both in the case of rectangular blocks and composition of small SATDs. Algorithm 6 presents the pseudo-code of the AFD as we implemented in HM.

Algorithm 6: AFD calculation compatible with the SATDs present in HM.

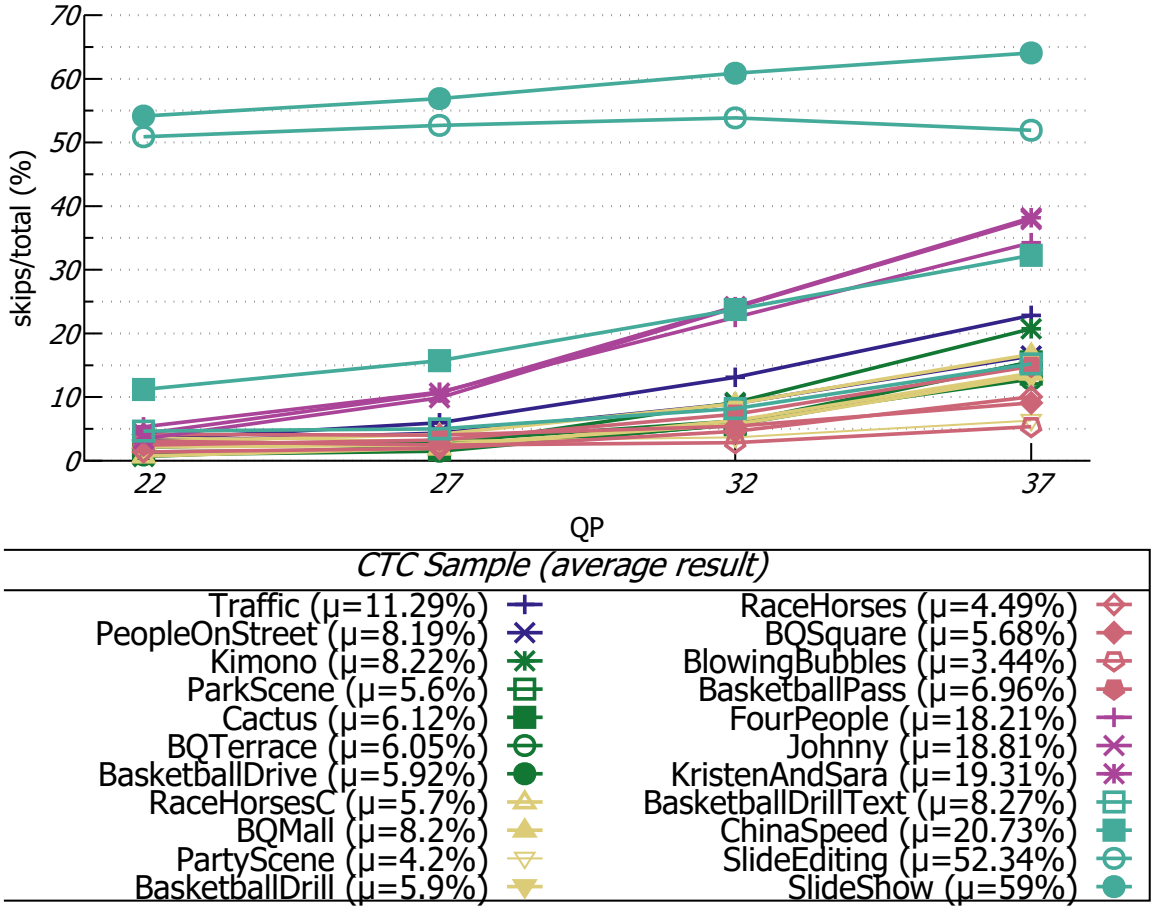
Input : $\mathbf{B}^{\text{ori}}{}_{m\times n}, \mathbf{B}^{\text{can}}{}_{m\times n}$
Output $afd^{\text{curr}} = af(\mathbf{D}_{m\times n})$
:
$afd^{\text{curr}} \leftarrow 0;$
if *(*$m\%8 = 0$*) and (*$n\%8 = 0$*)* **then**
  for $i \leftarrow 1; i < m; i \leftarrow i+8$ **do**
    for $j \leftarrow 1; j < n; j \leftarrow j+8$ **do**
      $d \leftarrow \mathbf{B}^{\text{ori}}{}_{i,j} - \mathbf{B}^{\text{can}}{}_{i,j}$ $afd^{\text{curr}} \leftarrow afd^{\text{curr}} + ((((|d| << 6) + 2) >> 2)$
    end
  end
end
else
  for $i \leftarrow 1; i < m; i \leftarrow i+4$ **do**
    for $j \leftarrow 1; j < n; j \leftarrow j+4$ **do**
      $d \leftarrow \mathbf{B}^{\text{ori}}{}_{i,j} - \mathbf{B}^{\text{can}}{}_{i,j}$ $afd^{\text{curr}} \leftarrow afd^{\text{curr}} + ((((|d| << 4) + 1) >> 1)$
    end
  end
end

To check the actual effectiveness, our method using AFD was implemented within the HM, version 16.6 (JCT-VC, 2013), and tested following the CTC (BOSSEN, 2012), using the Low Delay with P slices only (LD-P) configuration. Therefore, we tested each of the CTC 8-bit color depth video sample using QPs 22, 27, 32 and 37.

Once our method of successive elimination for SATD only eliminates impossible candidates, as demonstrated in the previous subsection, there is no change in coding efficiency. Hence, it is pointless to present the RD performance evaluation against a reference encoder (such as the analysis presented in Chapter 3), since both RD curves are the same. It suffices to say that we performed such tests to evaluate our implementation correctness. On the other hand, Figure 30 shows the effectiveness of our method, in terms of filtered SATD with respect to QP and the video sample.

In general, the relative number of skipped SATDs (effectiveness) when using the proposed method increases with the increase in QP. These results meet the findings of Coban and Mersereau (1998) and Luc Trudeau, Stephane Coulombe, and Christian Desrosiers (2015),

Figure 30 – Resulting % of skipped SATDs relative to the total number of candidate blocks, with respect to the used QP.



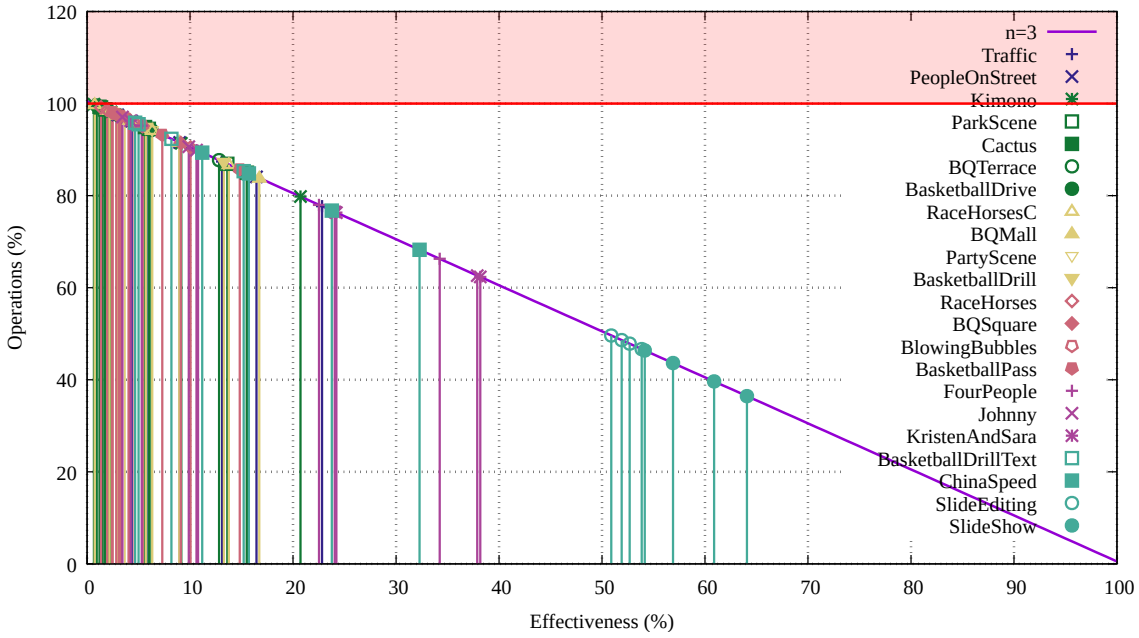
Source: the author.

and are justified by the latter as caused by an increase in the Lagrange multiplier value (λ from Equation 2.6). Therefore, there is an increase in the rate weight, allowing more $\mathbf{B}^{\text{can}}$s to be filtered. The best cases were for the two 720p screen content, SlideEditing and SlideShow, to which our method provided up to 64% skips. Following those cases, the results presented in Figure 30 show a good performance for the three sequences from class E, which present few movements, similar to those from video conferencing.

Even with the small number of candidate blocks evaluated during FME and the restrictions concerning the SATD size imposed by HM (only 4×4 and 8×8), our method for the successive elimination of SATDs presented acceptable performance. In average, 13.52% of the SATDs are skipped. Our method presents even better performance for low motion scenes: for class E of CTC almost 20% of SATDs are skipped. For screen content, the average effectiveness rises up to 44.02%. In the best case, for SlideShow encoded with QP 37, the number of skipped SATDs was 64%. However, some of the cases that had largest elimination amounts also resulted in the worst coding efficiencies, as presented in Section 3.4.1.

Figure 31 shows the number of operations needed according to the effectiveness, taking into account the overhead of AFD. The % number of operations is shown as a function of the effectiveness (0% to 100%), just as in Figure 28, for the SAD-based SEA. Notice that the overheads are smaller than the ones from SAD-based SEA, if we consider that the ADS is computed during BMA in a raster scan. The effectiveness is the same shown in Figure 30. However, now the idea is to identify the potential savings from the proposed technique.

Figure 31 – Relative number of executed operations in SATD-based SEA with respect to a BMA without SEA, as a function of its effectiveness in skipping candidates. The horizontal line in 100% delimits the region of overheads, i.e., the shaded region where the use of SEA requires more operations than a simple BMA. The dots indicate the actual effectiveness results of each sample, shown in Figure 30. The vertical lines connecting the dots to the abscissa are present to ease visualization.



Source: the author.

It is possible to observe in Figure 31 that most of the video samples, paired with the used QPs, only obtain about 20% of savings in terms of % of the number of operations. One way to improve the savings is by providing improved elimination criteria, thus trying to increase the effectiveness. The best way to be more effective is by having elimination criteria that generates larger values (closer to the actual distortion measure). A known method for that is the so-called multi-level SEA (GAO; DUANMU; ZOU, 2000), presented in the sequel.

4.2.3 Rate-constrained multi-level SEA using SATD

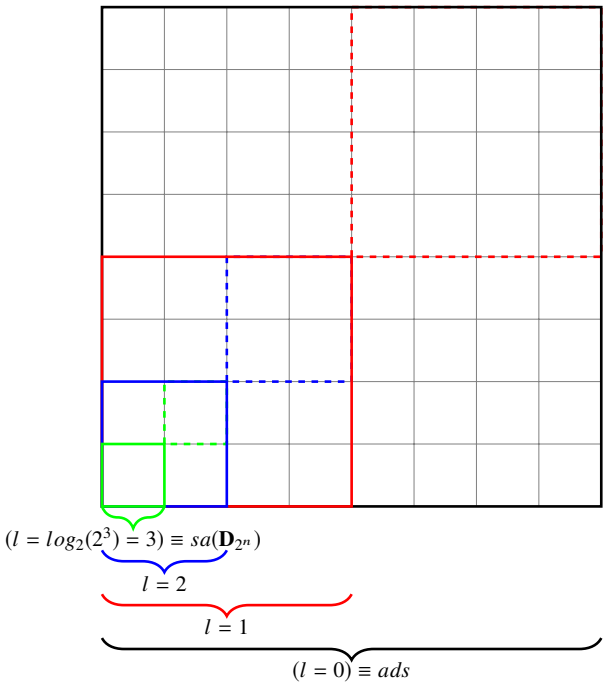
The main drawback of SEA is that the difference of block sums is not close enough to the true SAD (LIU; WEI; LAI, 2008, p. 263). This is also true when comparing the AFD with the SATD. One possible solution, when considering the SAD is the so-called Multi-level Successive Elimination Algorithm (MSEA) (GAO; DUANMU; ZOU, 2000). Such algorithm

relies on most restrictive elimination criteria constructed when considering more than 2 elements in Property 2, such as:

$$|a+b+c+d| \leq |a+b|+|c+d| \leq |a|+|b|+|c|+|d| \tag{4.70}$$

Therefore, after Equation 4.70, the SEA can be extended to successively increase the value of the elimination criteria. One approach that generates values potentially closer to the value of the SAD is by dividing the blocks in sub-blocks and compute the ADS of each one. This is the main essence of MSEA (GAO; DUANMU; ZOU, 2000). The levels consist in partitioning the block in a quad-tree like manner, in a way that all nodes are equally partitioned for a given depth which, in turn, corresponds to the level. Figure 32 illustrates this recursive quad-tree partitioning scheme for an 8×8 block.

Figure 32 – Multi-level partitioning of an 8×8 block considering the SAD as distortion metric. For each level (l), the sub-partitions are being only illustrated on the bottom left partition, however they also sub-divide the other partitions as well. The key idea is that the absolute operation is applied over the sum of the elements of each partition (Equation 4.71). For the lowest level ($l = 0$), all elements are first added and then the absolute difference is computed, which is equivalent to the ADS. On the other hand, for the highest level ($l = 3$), the SAD is computed because each partition corresponds to exactly an element of the block.



Source: Cancellier (2016).

Level 0 ($l = 0$) is equivalent to the ADS of single-level SEA, while level n ($l = n$) is equivalent to the SATD of a $2^n \times 2^n$ block. The elimination criterion, in this case is:

$$sa\left(\mathbf{S}\left(\mathbf{D}_{2^n;2^{n-l}}\right)\right) \tag{4.71}$$

Such a criterion can be used to eliminate impossible candidates, given the following relationships:

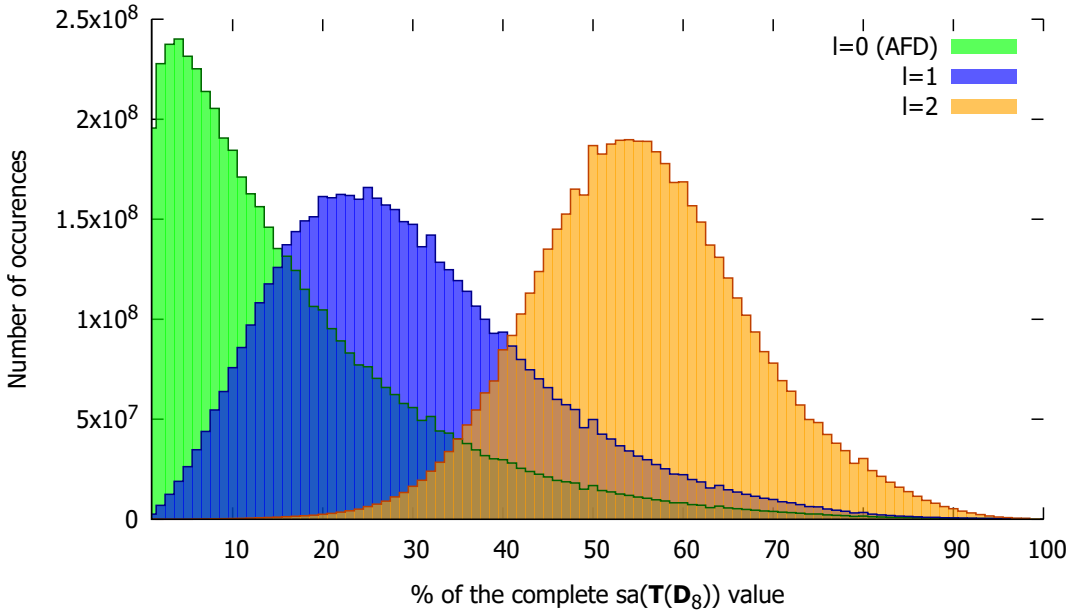
$$ads = \overbrace{sa\left(\mathbf{S}\left(\mathbf{D}_{2^n;2^n}\right)\right)}^{l=0} \leq sa\left(\mathbf{S}\left(\mathbf{D}_{2^n;2^{n-1}}\right)\right) + sa\left(\mathbf{S}\left(\mathbf{D}_{2^n;2^{n-2}}\right)\right) \leq$$
$$\leq \cdots \leq \underbrace{sa\left(\mathbf{S}\left(\mathbf{D}_{2^n;2^0}\right)\right)}_{l=n} = sa(\mathbf{D}_{2^n}) \tag{4.72}$$

By Equation 4.72, one may notice that MSEA is a more general case of SEA. A similar MSEA approach can be used considering the SATD, by subdividing the matrix of the transformed differences, as follows:

$$sa\left(\mathbf{S}\left((\mathbf{T}(\mathbf{D}_{2^n}))_{;2^{n-l}}\right)\right) \tag{4.73}$$

As mentioned, the MSEA main assumption is that elimination effectiveness can be improved by generating larger values for the elimination criterion. Figure 33 shows an histogram of the values computed at each level for the first 64 frames of "BQTerrace" sequence. It is possible to notice that higher levels (higher l) results in a increasing number (y axis) of larger values (x axis). Therefore, the chances of eliminating impossible candidates improve.

Figure 33 – Histogram showing the % distribution of the value of each level with respect to the total SATD value. To obtain these results we used the first 64 frames of the sequence "BQTerrace" and the values were collected from the SATD in FME. HM v16.16, RA configuration.



Source: the author.

Yet, the same problem of the initial formulation of AFD (Equation 4.45) presents itself: the need to compute the entire HT. Cancellier (2016) proposes another formulation that is as follows:

$$sa\left(\mathbf{T}_{\mathbf{2^L}}\left(\mathbf{S}\left((\mathbf{T}\left(\mathbf{D}_{2^n;2^{n-l}}\right))\right)\right)\right) \tag{4.74}$$

Using Equation 4.45, it is possible to substitute the innermost transform by the AFD of each partition:

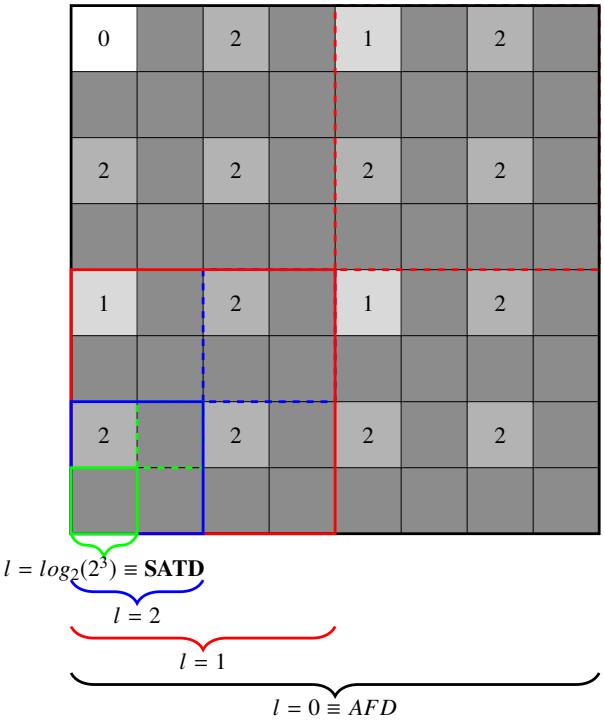
$$sa\left(\mathbf{T}\left(\mathbf{AF}\left(\mathbf{D}_{2^n;2^{n-l}}\right)\right)\right) \tag{4.75}$$

where

$$\left(\mathbf{AF}\left(\mathbf{D}_{2^n;2^{n-l}}\right)\right)_{;i,j} = af\left(\mathbf{D}_{;i,j}\right) \tag{4.76}$$

By doing so, only a small transform is needed. The needed transform, however, increases in size as the level increases.

Figure 34 – Multi-level partitioning of an 8×8 block considering the SATD as distortion metric. Each small square represents a sample of the transformed matrix. The colored and dashed lines represent the partitions on which the absolute value of the sum of the elements within each partition is computed, for each level. The numbered samples represent the ones used before the actual complete transform. However, except by the sampled marked as zero (correspondent to level 0, i.e., the AFD), the other values form a smaller block that is transformed, by a transform with the size of such smaller block. The samples from a higher level includes also the samples from the other smaller level.

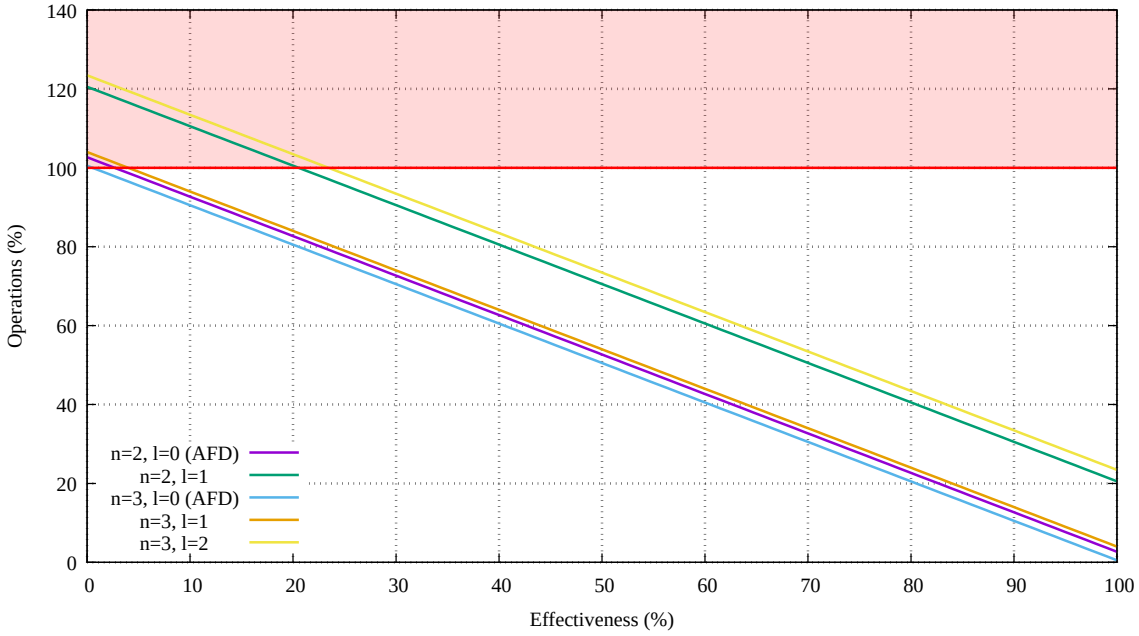


Source: Cancellier (2016).

Figure 35 shows the potential savings in number of operations according to the effectiveness in each level.

Higher levels (larger l) have large overheads and thus less potential savings. Therefore, it seems to be no advantage in going to higher levels. However, the idea to go into higher levels is to increase the effectiveness. Moreover, the algorithm only increases the level if the current ones does not eliminate. As shown in the previous section, the effectiveness of the rate-constrained AFD is relatively low for most samples, at least when considering the FME and small QPs.

Figure 35 – Relative number of executed operations in SATD-based SEA with respect to a BMA without SEA, as a function of its effectiveness in skipping candidates. The horizontal line in 100% delimits the region of overheads, i.e., the shaded region where the use of SEA requires more operations than a simple BMA. Notice that the overheads are smaller than the ones from SAD-based SEA.



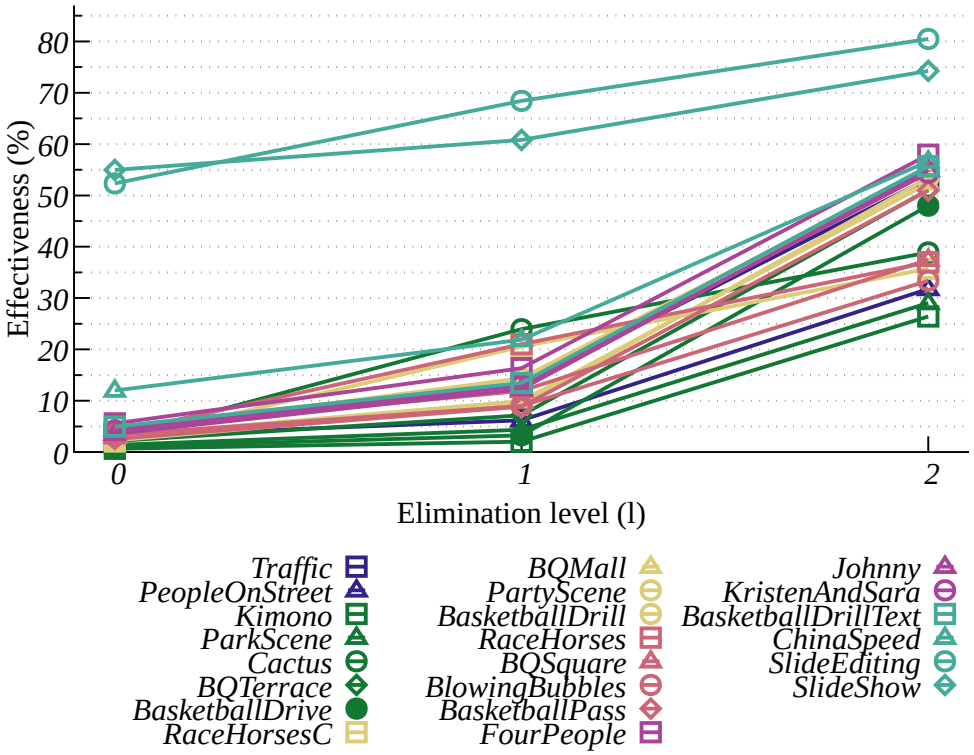
Source: the author.

Cancellier (2016) evaluated the effectiveness by implementing the MSEA using the SATD (MSEA-SATD) within the HM, version 16.6 (JCT-VC, 2013), and tested following the CTC (BOSSEN, 2012), using the Low Delay with P slices only (LD-P) configuration. Therefore, we tested each of the CTC 8-bit color depth video sample using QPs 22, 27, 32 and 37. Also, as for the single-level implementation, the rate term was considered.

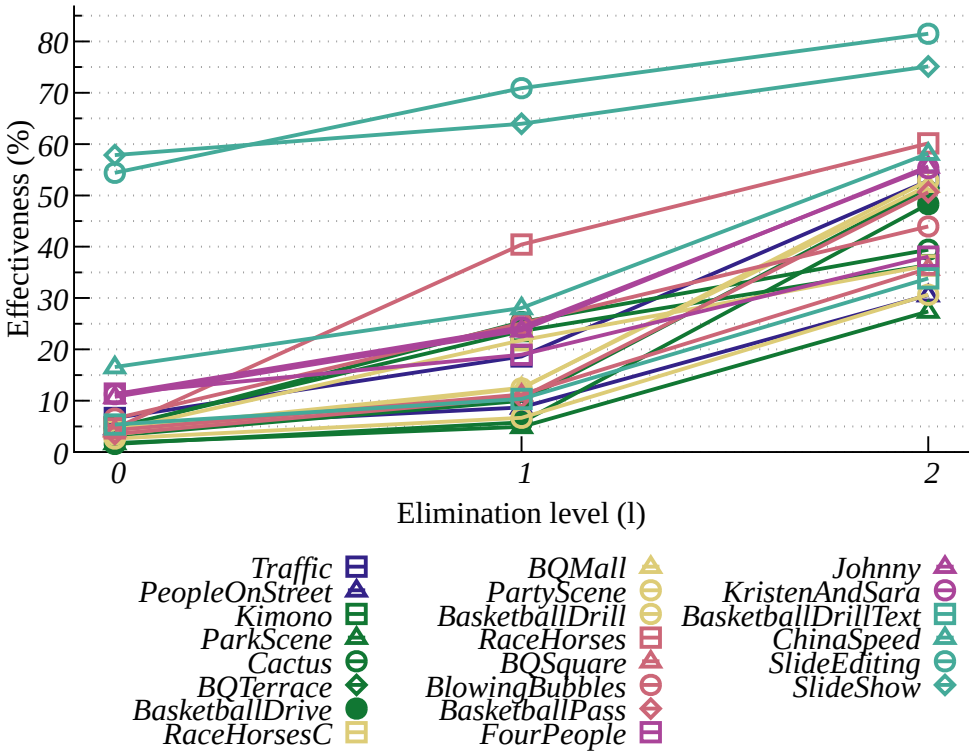
Similarly to the single-level, MSEA-SATD only eliminates impossible candidates, thus there is no change in coding efficiency. The rate-distortion was tested to evaluate the implementation correctness. Figure 30 shows the effectiveness of our MSEA-SATD method, in terms of filtered SATD with respect to QP and the video sample.

It is possible to notice that the accumulated effectiveness increases as the level increases. Moreover, the effectiveness also increases as increases the QP. Notice, also, that the effectiveness was obtained in the default FME from HM. This means that only 18 candidate blocks are evaluated, as shown in Figure 20. Each candidate thus corresponds to about 5.55% of the total cardinality of S. As the first block cannot be eliminated, because the minimum distortion initializes with ∞, the maximum effectiveness in this scenario is 94.44%. Moreover, the candidate with minimum distortion is evaluated twice in HM FME, lowering the maximum effectiveness to 88.8%. In the best case, level 2 is close to the maximum effectiveness in HM. Furthermore, in such level, on average, more than 12 blocks are skipped, among the 18. This means that, on average, only six blocks need the complete SATD computation, including the two

Figure 36 – Effectiveness of MSEA-SATD according to level and QP. (a) QP 22. (b) QP 27.



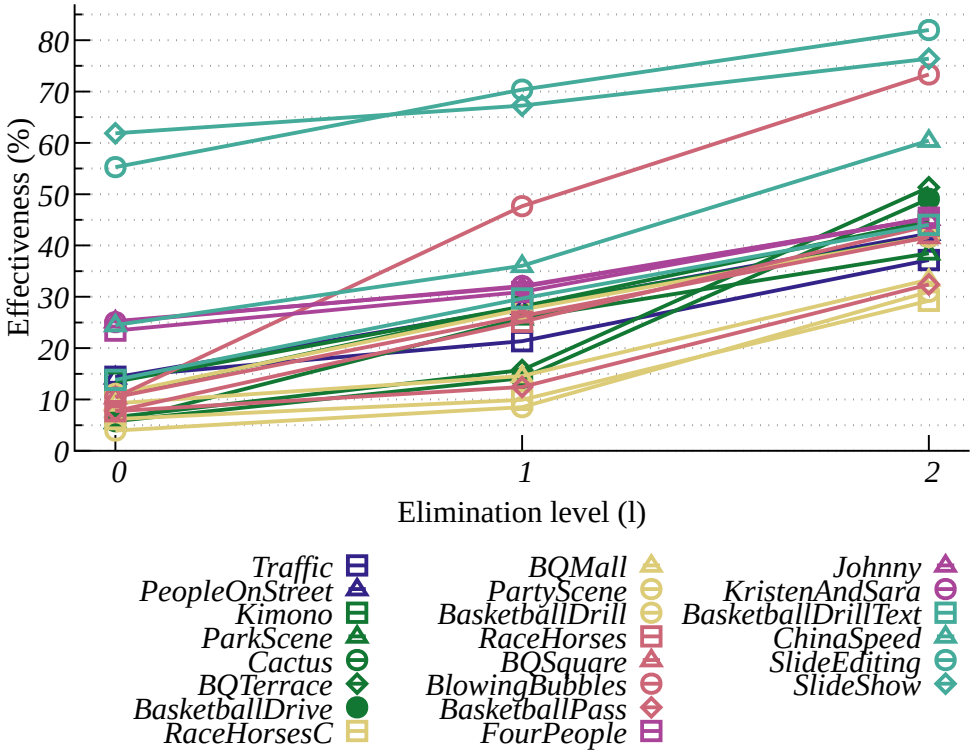
(a)



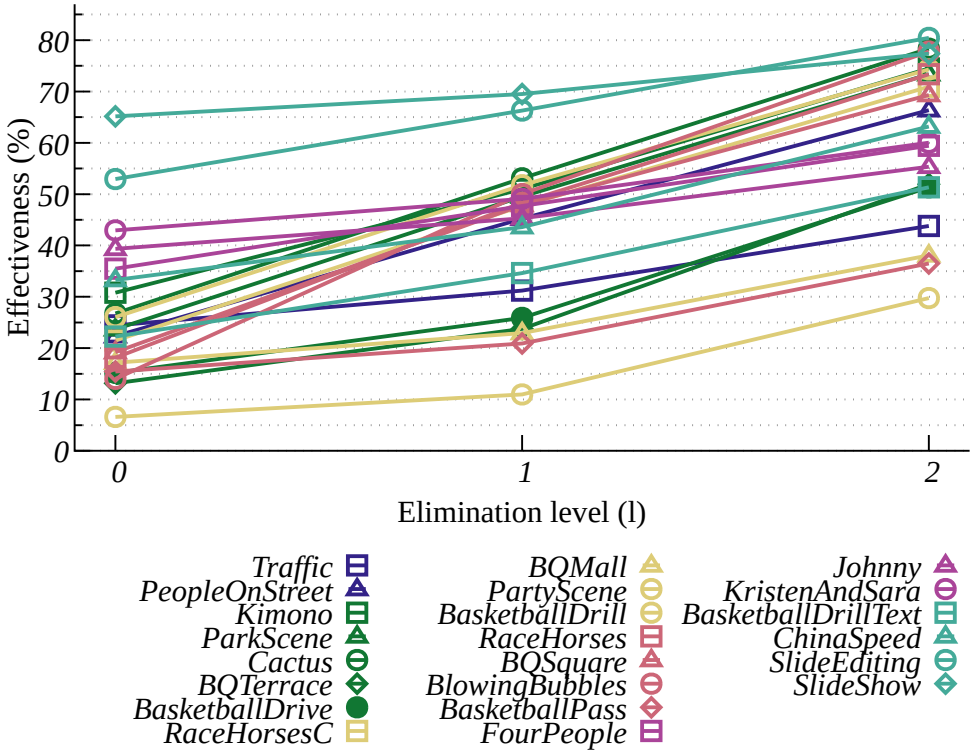
(b)

Source: adapted from Cancellier (2016, p. 49–50).

Figure 36 – Effectiveness of MSEA-SATD according to level and QP. (cont) (c) QP 32. (d) QP 37.



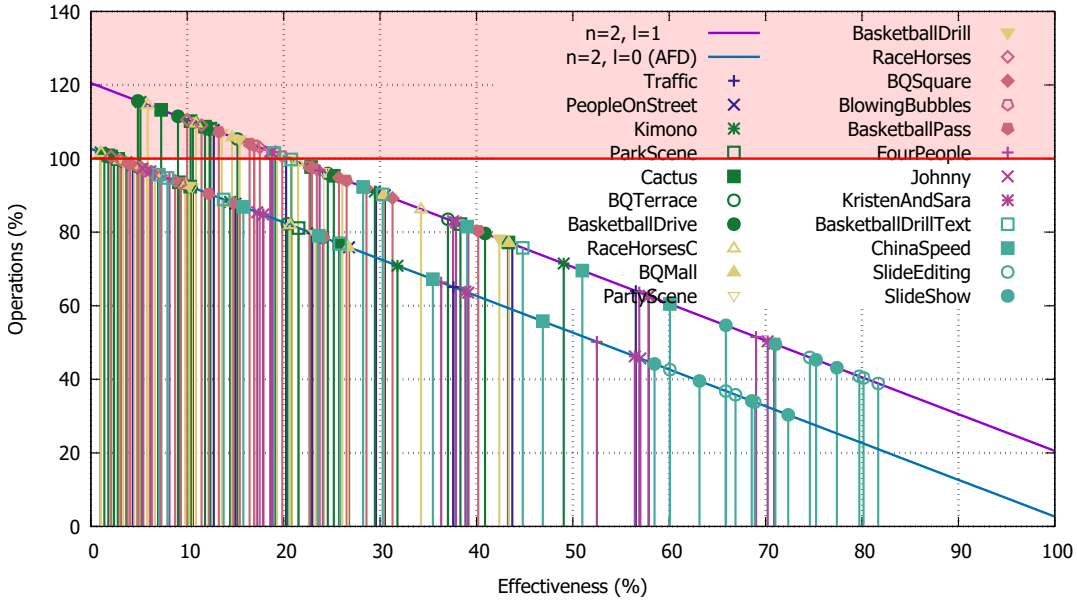
(c)



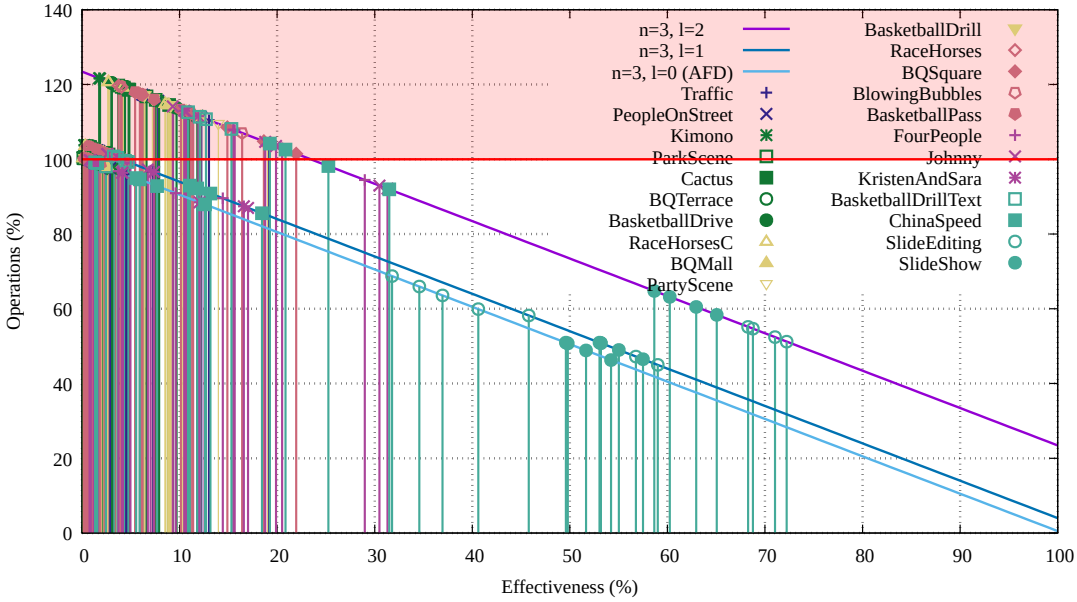
(d)

Source: adapted from Cancellier (2016, p. 49–50).

Figure 37 – Savings of MSEA-SATD according to level and QP. (a) 4×4. (b) 8×8.



(a)



(b)

Source: the author. Data from (CANCELLIER, 2016).

that are always evaluated. Yet, although the eliminations for the higher levels indeed increase, Figure 37 shows that the required extra computations in higher levels may reduce the percentage of saved operations.

As we show in the previous section, it is possible to reuse the calculations of small transform in the calculation of a larger one. The same idea can be applied to the MSEA-SATD. If a given level is not sufficient to eliminate, the remaining transform of that partitions must be computed and all values of the partitions can be used to compute the next level elimination criteria, until computing the entire SATD.

4.2.4 Notes on other SEA related works

In MSEA, when going from one level to the next $(l+1)$, all partitions are divided into four new ones. However, the subdivisions do not exploit a full quad-tree partitioning capability of successively generating tighter and tighter boundaries for eliminating candidates. Ce Zhu, Wei-Song Qi, and Wee Ser (2004) described a Fine Granularity Successive Elimination (FGSE) algorithm considering a broader range of the quad-tree possible subdivisions. Thus, while MSEA have five levels for a 16×16 block, the FGSE has 86, including the five from MSEA. In this sense, the MSEA is a special case of FGSE. Moreover, FGSE has increase chance to eliminate non-optimal $\mathbf{B}^{\text{can}}$ than MSEA (LIU; WEI; LAI, 2008, p. 264).

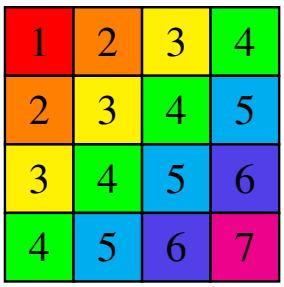
Other approaches includes the so-called Extended Successive Elimination Algorithm (ESEA) (BRUNIG; NIEHSEN, 2001) and Extended Multilevel Successive Elimination Algorithm (EMSEA) (TAE GYOUNG AHN; YONG HO MOON; JAE HO KIM, 2004). While the former uses the last level value to compute the SAD, the latter uses a previous level to compute the next one. Tae Gyoung Ahn, Yong Ho Moon, and Jae Ho Kim (2004) claims that EMSEA is up to 40% faster than MSEA. By its turn, the so-called Adaptive Multilevel Successive Elimination Algorithm (AdaMSEA) reduces the number of operations of FGSE adaptively by further partitioning the partitions with larger sum norm value (LIU; WEI; LAI, 2008). The underlying assumption is that those regions are less homogeneous and thus have a larger probability of eliminating the $\mathbf{B}^{\text{can}}$.

The strict MSEA (SONG et al., 2006) and enhanced strict MSEA (SONG et al., 2007) increase the effectiveness by multiplying the elimination criteria by some constants. However, by doing so, such algorithms cease to be optimal.

All of the above could potentially be adapted for using SATD, but depending on the granularity it could be hard to find simple elimination criteria. Also, all these methods rely on the fast computation of the sums norms upfront (one initial frame computation). However, such initial computation cannot be done for FME, otherwise it would be necessary to interpolate the whole frame. Moreover, we may exploit other means to subdivide the block instead of a quad-tree.

Dong and Pan (2017) proposed a multi-level approach that computes the distortion after the HT. Yet, they assumed a search using SSD as distortion metric and their approach was only defined for $2^n \times 2^n$ blocks. Such blocks are partitioned down until having only 4×4 sub-blocks. For each of the sub-blocks a HT is computed. Then they sum the transformed values according to their frequencies, in 7 different levels, shown in Figure 38. So, besides their target distortion metric not being the SATD, they introduce the need for computing the HT. Also, no matter the level, the HT needs to be initially computed. Furthermore, their evaluation was outside a reference encoder (no standard was defined), and only tested with 16×16 blocks.

Figure 38 – Defined levels



Source: adapted from Dong and Pan (2017, p. 26–27).

4.3 PARTIAL DISTORTION ELIMINATION (PDE)

As mentioned at the beginning of this chapter, PDE is a means of reducing the computational load of SAD (CHIU; SIU, 2006) without introducing losses (CHAN; SIU, 2002). Such SAD-based PDE is presented in Section 4.3.1. Althought simple in its formulation and applicability, we did not found any work that adopted or evaluated PDE in the SATD. Our proposal has two approaches. The first one is called coarse-grain PDE of SATD and it is based on the SATD computation defined in Equation 2.21[62].

As it is, the SATD of large block partitions (up to 64×64) is obtained after the sum of 4×4 or 8×8 SATDs. This drives our proposal of using PDE in a coarse grain level, during block compositions. Therefore, such PDE approach is able to skip SATD computations in a block-by-block basis. The second one is called fine-grain PDE of SATD and it is based on the SATD formulation from Equation 2.16, considering the sums of the lines of the transformed differences matrix. Thus, the second one is the most similar to the known PDE for SATD.

The contributions of this section are:

1. The proposal of coarse- and fine-grain PDE techniques to be used during SATD computation; and

2. The analysis of simplicity, effectiveness and savings of the coarse-grain approach;

This section contains results published in PATMOS2015, ICECS2016, and SBCCI2017[63]

4.3.1 PDE of SAD

Such technique consists of the SAD early termination when obtained in a line-by-line fashion, i.e., a successive sum of Partial SADs (PSADs). The k^{th} PSADs (PSAD^k) can be

[62] Although the similarity metric used during encoding is not normative, our coarse-grain approach focus on such SATD implementation present in HM.

[63] Seidel, Bräscher, and Güntzel (2015), Seidel, Güntzel, and Agostini (2016) and André Beims Bräscher, Ismael Seidel, and José Luís Güntzel (2017).

defined as:

$$sa^k(\mathbf{D}_{l\times m}) = \sum_{i=1}^{k}\sum_{j=1}^{m}|d_{i,j}| \tag{4.77}$$

for $1 \leq k \leq l$.

Equation 4.77 can be recursively defined as:

$$sa^k(\mathbf{D}_{l\times m}) = \begin{cases} \sum_{j=1}^{m}|d_{1,j}|, & \text{if } k = 1 \\ sa^{k-1}(\mathbf{D}_{l\times m}) + \sum_{j=1}^{m}|d_{k,j}|, & \text{otherwise} \end{cases} \tag{4.78}$$

Notice that $\forall k > 1, sa^{k-1}(\mathbf{D}) \leq sa^k(\mathbf{D})$. Thus, PSAD is a monotonically increasing function, i.e., increasing k never decreases this function value. As BMA aims at minimum *cost* (Equation 2.3), when PSAD^k exceeds the current minimum (sad_{min}), the remaining lines can be skipped (MAHMOOD, 2011). Algorithm 7 shows the BMA with SAD using PDE.

Algorithm 7: Block Matching Algorithm (BMA) using SAD with PDE.

```
Input  : B^ori_{l×m}, S
Output : B^ref_{l×m}
:
1  sad^min ← ∞;
2  foreach B^can ∈ S do
3      sad^curr ← 0;
4      for 1 ≤ k ≤ l do                                   // line-by-line
5          sad^curr ← sad^curr + sa(B^ori_{k,:} − B^can_{k,:}) ;
6          /* see Equation A.1 for colon ":" notation                  */
7          if sad^curr > sad^min then
8              break;
9          end
10     end
11     if sad^curr < sad^min then
12         sad^min ← sad^curr ;
13         B^ref ← B^can
14     end
15 end
```

As one might expect, the number of saved operations depends on how large is k when a skip may occur. The key benefit of PDE is its ability to reduce SAD complexity without introducing further losses (CHAN; SIU, 2002). We also exploited the PDE to reduce the energy when pel decimation[64] is applied to the SAD (SEIDEL; BRÄSCHER; GÜNTZEL, 2015). That architecture had as overheads an extra register for keeping the minimum SAD and a comparison unit. Because such comparison was on the critical path, for high frequencies the area overhead was of about 11%. We showed that PDE combined with 2:1 pel decimation ratio was able to

[64] A subsampling of the pixels in both original and candidate blocks (KUHN, 1999, p. 74).

save up to 48% energy of each SAD computation. Yet, such pel decimation causes a increase in BD-Rate: for class B, for instance, the increase was of 5.44%, on average.

On the other hand, without resorting to pel decimation, Abreu, Santana, et al. (2018) reported 16.4% and 11.6% energy savings for FHD (1080×1920) and UHD (3840×2160), respectively. Their architecture assumes a sequential execution of SADs and uses the comparator for loading the $sad^{\min}$ and also for PDE. So they do not assess the possible PDE overhead, claiming that only a nor gate is added. However, their elimination does not consider the rate cost, as the original PDE. Abreu, Grellert, et al. (2019) shows that the inclusion of the rate term before the begining of SAD calculation increases the energy efficiency of PDE by 17.5%.

Yet, to our best knowledge, there is no work that adopts PDE for SATD calculation. In the next two subsections we will show how to use PDE for SATD in two different ways.

4.3.2 Coarse-grain PDE of SATD: block-by-block

We propose a method to reduce the complexity of SATD computations that relies on PDE during the composition of large SATDs. Equation 2.21 shows the calculation of SATDs for such large SATDs. Given that such formulation has a sum of smaller SATDs, we can stop the computation earlier if the current sum is larger than the current minimum cost from BMA execution. Such method was baptized as **Coarse Grain PDE for Hadamard ME**. Following the definition of PSAD in Equation 4.77, we can define the block-wise Partial SATD (PSATD) computation, for $1 \leq k \leq l \times m$, as follows:

$$sa^k(\mathbf{T}(\mathbf{D}_{l\times m})) = \sum_{i=1}^{\left\lfloor \frac{k\times 2^n}{m} \right\rfloor} \sum_{j=1}^{\left\lfloor \frac{m}{2^n} \right\rfloor} sa\left(\mathbf{T}\left(\mathbf{D}_{;i,j}\right)\right) + \sum_{j=1}^{k\%(\frac{m}{2^n})} sa\left(\mathbf{T}\left(\mathbf{D}_{;(\left\lfloor \frac{k\times 2^n}{m} \right\rfloor+1),j}\right)\right) \tag{4.79}$$

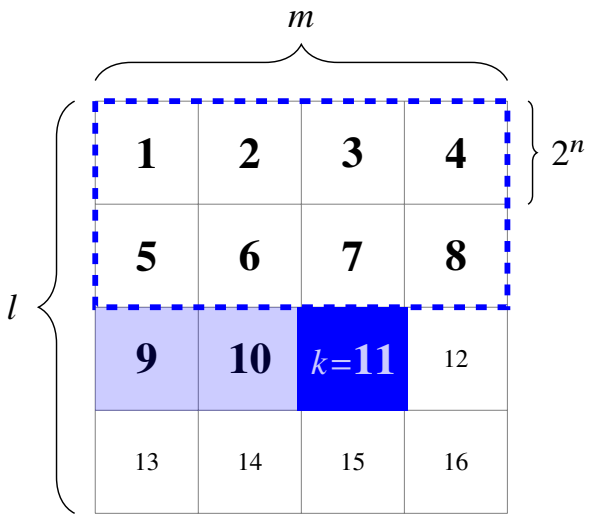
If $\left\lfloor \frac{k\times 2^n}{m} \right\rfloor = 0$, the first term of Equation 4.79 must be ignored. Such a term is responsible for iterating over the SATDs in the block lines above the one containing k. By its turn, the second term of Equation 4.79 sums the SATDs of the block line containing the k^{th} block, halting after computing the SATD of the k^{th} block. Figure 39 demonstrates the computation of a $\text{PSATD}^k_{M\times N}$.

Similarly to PDE in SAD, since each $\text{PSATD}^k \geq 0$, it follows that $\forall k > 1, \text{PSATD}^{k-1} \leq \text{PSATD}^k$. Thus, when a PSATD^k exceeds the current minimum, the remaining computation can be skipped.

Finally, in a rate-constrained encoder, the rate cost of selecting a given $\mathbf{B}^{\text{can}}$ must be considered by the Cost, along with its distortion. To that, the Lagrangian rate-distortion cost (j_{cost}) is used as Cost, as previously defined in Equation 2.6. Therefore, when using PDE in a rate-constrained encoder, $\lambda \times \text{Rate}$ must be taken into consideration. The adopted strategy is to use $\lambda \times \text{Rate}$ as the initial value in the sum, instead of initializing the current SATD as zero.

The proposed coarse grain PDE method was implemented within the FME of HM version 16.9 (JCT-VC, 2013) and tested following the CTC (BOSSEN, 2012), using the LD-P

Figure 39 – Example of the computation of a PSATD^k, with $k = 11$. The dashed blocks have their SATD computed by the first term of Equation 4.79, whereas the filled blocks have their SATD computed by the second term of Equation 4.79.



Source: the author.

Algorithm 8: Coarse-grain PDE for SATDs as implemented in HM.

Input : $\mathbf{B}^{\text{ori}}{}_{l\times m}, \mathbf{B}^{\text{can}}{}_{l\times m}, j^{\text{min}}$
Output $j^{\text{curr}} \leq sa\left(\mathbf{T}\left(\mathbf{D}_{l\times m;2^n}\right)\right) + \lambda^{\text{motion}} \times R$
:

```
j^curr ← λ^motion × R ;          // increases the chances of early termination
if l%8 = m%8 = 0 then
    p ← l/8, q ← m/8;
else
    p ← l/4, q ← m/4;
end
for 1 ≤ i ≤ l do
    for 1 ≤ j ≤ m do
        j^curr ← j^curr + sa(T(D_{;i,j}));
        if j^curr > j^min then
            /* in this case, there is no need to continue j^curr computation
               since it is not minimum one in BMA                          */
            return;
        end
    end
end
```

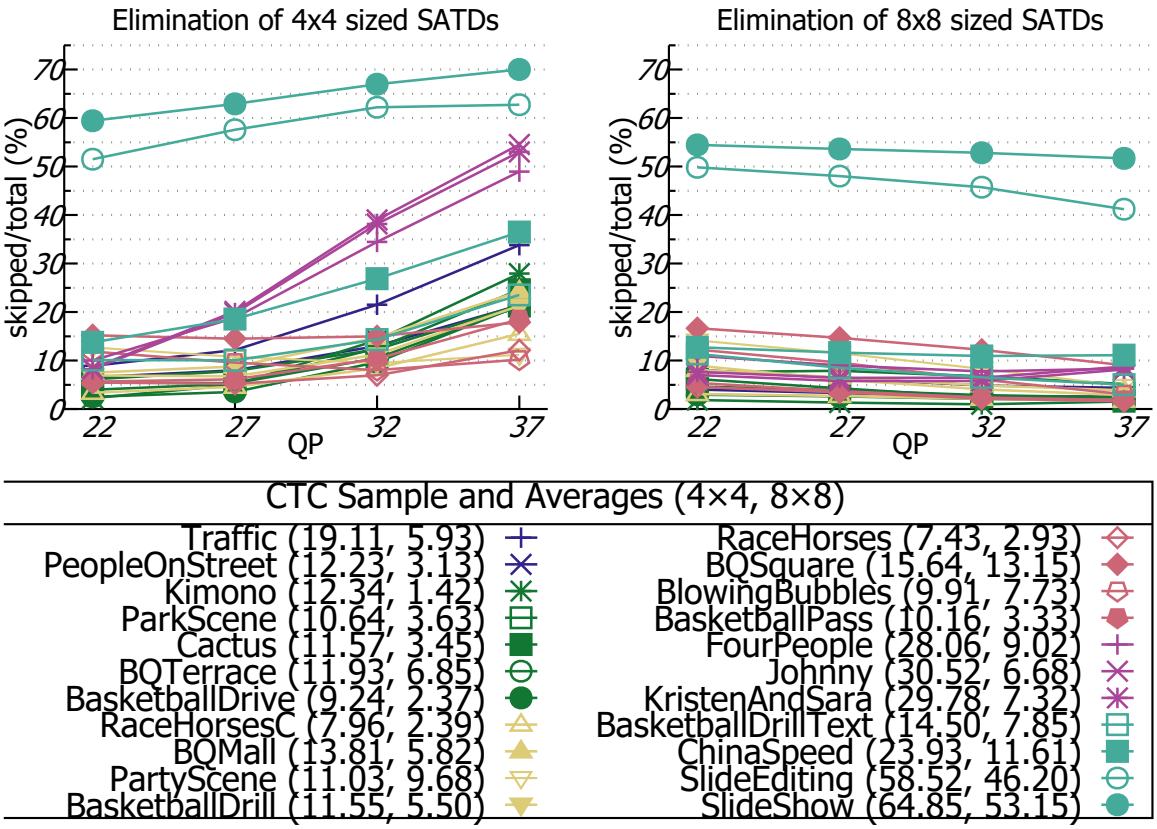
configuration and QPs 22, 27, 32 and 37. Since the developed architectures, presented in the sequel, use 8-bit input pixels, all of the CTC (BOSSEN, 2012) 8-bit color depth video samples were tested.

Once PDE only skips SATD calculations when the current computation exceeds the current minimum, there is no change in coding efficiency. Thus, there is no point in presenting RD performance evaluation against a reference encoder: both RD curves are the same.

Nevertheless, we performed such tests to guarantee our implementation correctness.

The effectiveness of our method, in terms of filtered SATD with respect to QP and video sample, is presented in Figure 40. Using S as small as 18 in FME (see Figure 20), the proposed method presents an acceptable elimination ratio. The results show that 4×4 presents a larger % of eliminations than 8×8, with averages of 19.30% and 9.96%, respectively. Also, it is possible to see that in 4×4 the elimination ratio grows as the QP increases.

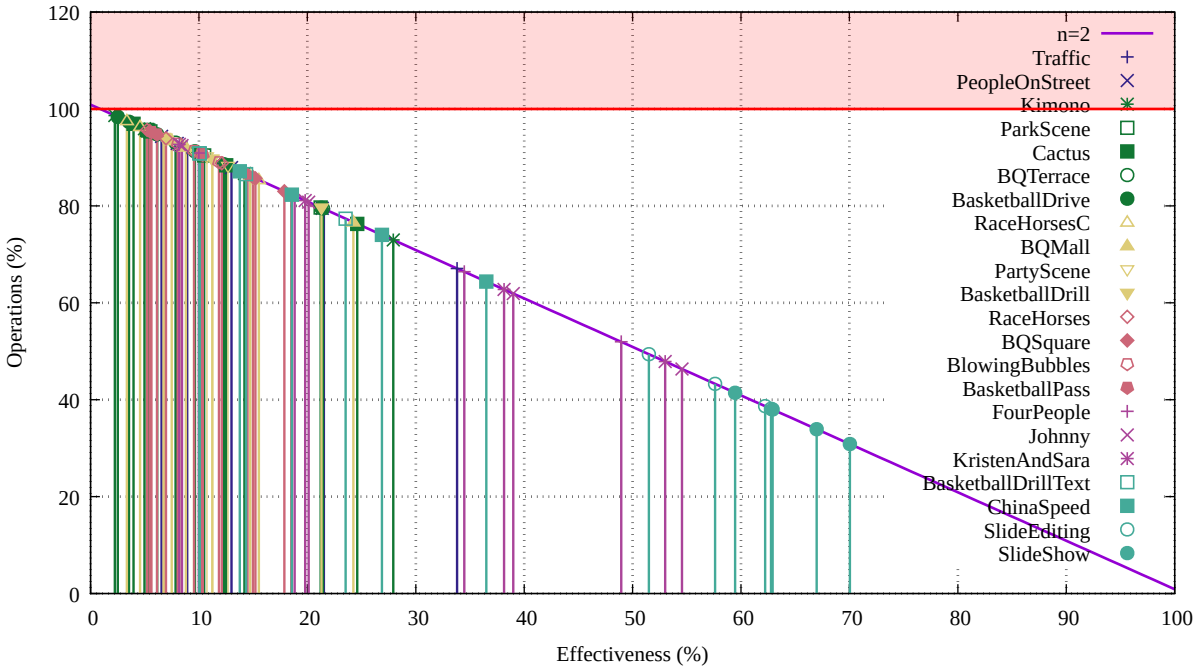
Figure 40 – Effectiveness of coarse-grain PDE according to level and QP.



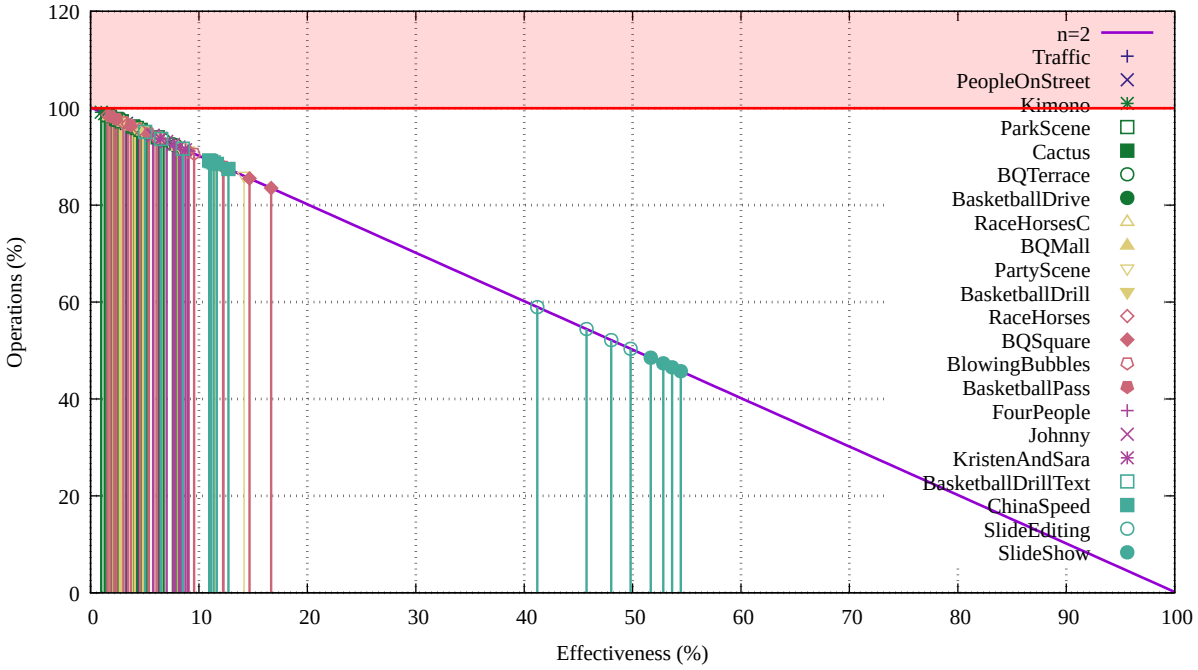
Source: the author.

The method is simple, as it only needs one extra comparison for each one of the small blocks composing a larger one. Figure 41 shows the savings results, considering the overhead. It is possible to observe that the savings are more prominent for 4×4 blocks than for 8×8. In the latter case, most of the savings were below 20%, with a few screen content presenting about 50% savings. As for the SATD-SEA, the best cases in terms of savings, are the worst ones in terms of coding efficiency for SATD.

Figure 41 – Savings obtained by coarse-grain PDE according to video simple and effectiveness.



(a)



(b)

Source: the author.

4.3.3 Fine-grain PDE: line-by-line

The second method, baptized as **Fine-grain PDE for Hadamard ME** is straightforward given the formulation of PSAD. Similar to Equation 4.77, the k^{th} PSATD can be defined as:

$$sa^k(\mathbf{T}(\mathbf{D}_{2^n})) = \sum_{i=1}^{k}\sum_{j=1}^{m}\left|(\mathbf{T}(\mathbf{D}))_{i,j}\right| \text{ , for } 1 \leq k \leq 2^n \tag{4.80}$$

As for the SAD PDE, such function is also monotonically increasing with regards to k. Thus, $\forall k > 2$, $sa^{k-1}(\mathbf{T}(\mathbf{D}_{2^n})) \leq sa^k(\mathbf{T}(\mathbf{D}_{2^n}))$.

Considering the PSAD, one approach to increase the effectiveness is to try to sum first the rows with larger distortion. Yet, for the SAD this is not easy to detect. On the other hand, because the **T**, the PSATD computed line-by-line from top to bottom already has the optimal sum order. Figure 42 shows such behavior with heatmaps of the values of each pixel distortion with respect to the total distortion sum, for QPs 22 and 37 (the smallest and highest from the CTC). The percentages were computed as $(\left|\mathbf{D}_{i,j}\right| / sa(\mathbf{D}_{2^n})) \times 100\%$ for SAD and $(\left|\mathbf{T}(\mathbf{D})_{i,j}\right| / sa(\mathbf{T}(\mathbf{D})_{2^n})) \times 100\%$ for SATD.

The problem with fine-grain PDE is that the transform still needs to be computed. Therefore, the number of saved operations will be close to the ones presented in the case of coarse-grain PDE. After all, in the best case scenario, we will only be able to save the last sums of a block already transformed. We exploited such approach to save energy in a low area SATD hardware that used a Linear Buffer (LB) instead of a Transpose Buffer (TB) in (BRÄSCHER, André Beims; SEIDEL, Ismael; GÜNTZEL, José Luís, 2017). Although fine-grain PDE was able to reduce the energy, it was still less energy-efficient than the TB-based SATD architecture. The biggest advantage of using a LB is its reduced area: the TB-based architecture is about five times larger than the LB one. Therefore, if area is a major concern than energy, then fine-grain PDE provides means to also reduce the energy.

4.4 CHAPTER CLOSING REMARKS

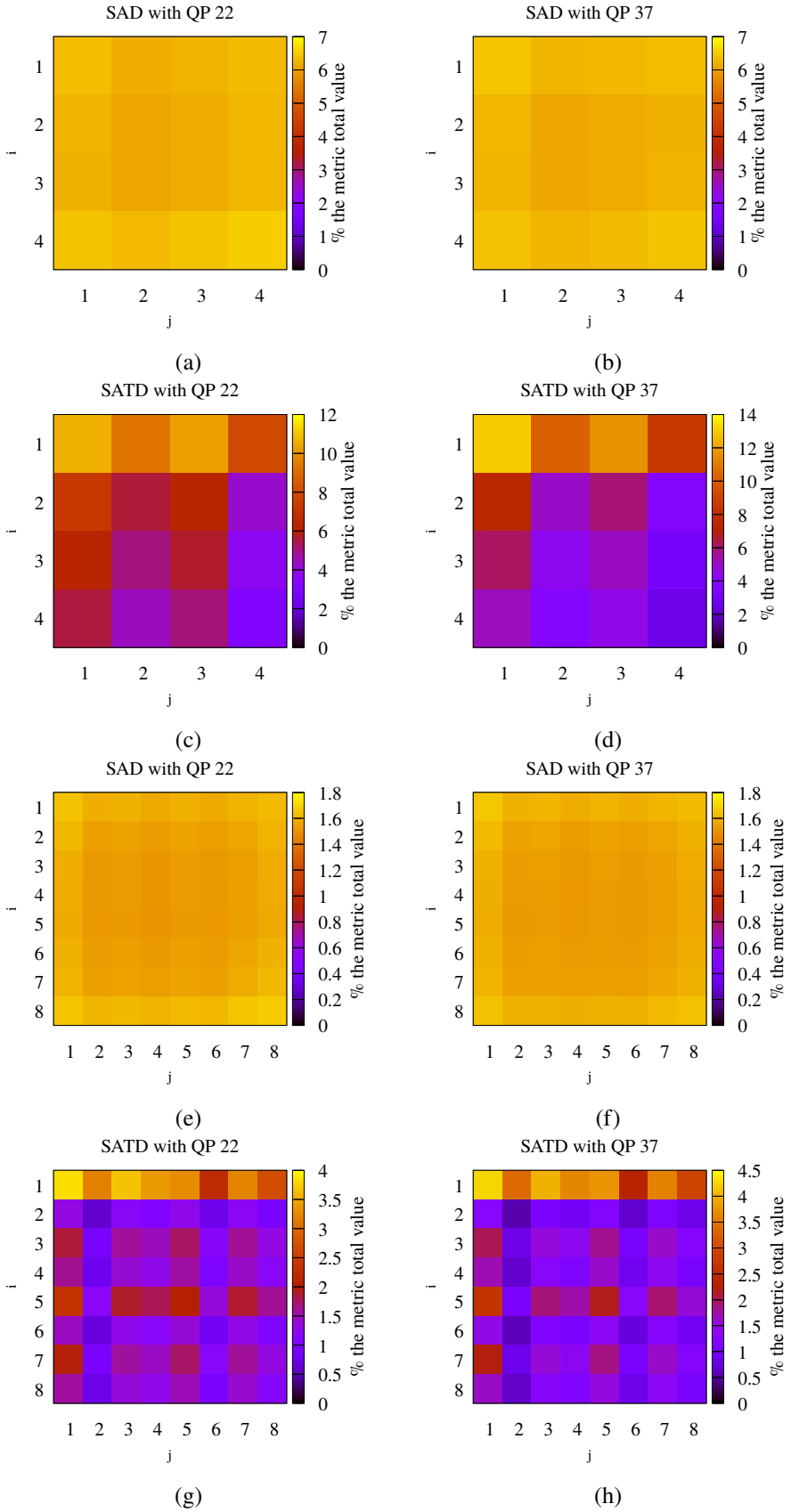
In this chapter we exploited and adapted three techniques to save operations in SATD. Among those three techniques, the reuse provides the maximum simplicity and effectiveness, resulting in about 2/3 transform operations saved. The SATD-based SEA is simple in its single-level form but achieves smaller effectiveness than its multi-level counterpart, which is less straightforward. In general, they are able to eliminate up to 80% of candidate blocks, even considering FME solely. Finally, we exploited the PDE, in coarse and fine grains. The coarse-grain PDE for SATD relies on the block-wise SATD calculation, achieving up to 70% reduction in the number of 4×4 blocks required and up to 55% of 8×8 blocks. The fine-grain PDE for SATD savings are almost the same as for the coarse-grain, because it still needs to compute the same number of transforms.

We also presented some related works that exploit those techniques in the SAD. As mentioned by L. Trudeau, S. Coulombe, and C. Desrosiers (2018, p. 922), our SEA-based works can be combined, instead of competing with each other. After all, the SATD main application scenario is in FME, whereas their proposal is applied to IME with SAD.

Also, there are some similarities between SEA and PDE, like all roads lead to Rome. For instance, one may observe that ADS (Equation 4.40) serves as an elimination criterion for SATD. However, the ADS is exactly the DC coefficient of HT, i.e., the $(\mathbf{T}(\mathbf{D}_{2^n}))_{1,1}$. Therefore, if $(\mathbf{T}(\mathbf{D}))_{1,1} = \text{ADS} > satd^{\min}$, because the $sa()$ is a monotonically increasing function, then certainly $sa(\mathbf{T}(\mathbf{D})) > satd^{\min}$. Nevertheless, using ADS as an elimination criterion for the SATD would be, at most, as effective as the fine-grain PDE (row-by-row). Another example is that the PDE can be applied to stop the SEA criterion calculation earlier, if the value is already larger than the minimum. On the other hand, this may increase the testing overhead.

Finally, these techniques will be further adopted in SATD hardware architectures, in Chapters 5 and 8. It is worthy remember that whenever needed (i.e., during BMA), the rate term of RDO must be present in the j_{cost} computation, since it gives improved coding efficiency, as shown in Section 3.4.2.

Figure 42 – Average share of the total value of the distortion metric that each position in a block contributes to. 64 frames from sequence BQTerrace encoded with HM v.16.16 and RA configuration.



Source: the author.

5 SATD HARDWARE DESIGN

As mentioned in Chapter 1, the most complex encoding tools of video coding should be off-loaded to dedicated hardware accelerators (VANNE et al., 2012). Unfortunately, dedicated hardware architectures to implement the SATD are more complex than those generally used to calculate the SAD, resulting in silicon area and power consumption overheads. On the other hand, Chapter 3 demonstrates that disabling the SATD in FME decreases the coding efficiency in HEVC. Even in a complexity constrained encoder, using the SATD for FME is highly recommended (CORREA et al., 2012). Thus, to maintain the coding efficiency, SATD hardware architectures must be as energy-efficient as possible and so their design must be carefully evaluated. Moreover, albeit High Level Synthesis (HLS) tools may greatly reduce design time, they still produce poorer quality designs than those obtained by solely using the Register Transfer Level (RTL) flow (LAHTI et al., 2018). Therefore, the most demanding encoder modules must be carefully tailored at the RTL to achieve high energy efficiency (CHAKRABARTI et al., 2015).

A possible way to improve the energy efficiency in SATD computation is by exploiting its mathematical properties, which is the underlying assumption of the present thesis. For instance, the so-called Transform Exempted (TE) method relies on a property that allows for reducing the number of arithmetic operations in the SATD (JOU, 2007; ZHU; XIONG, 2009). Yet, most energy consumption in a 2D transform comes from needing a so-called Transpose Buffer (TB) (RITHE; CHENG; CHANDRAKASAN, 2012), which will be presented in the sequel. Due to the TB, that property by itself has limited benefits in SATD hardware designs (CANCELLIER; SEIDEL, et al., 2015).

When variable block sizes are considered during encoding, a possible approach to reduce redundant operations is by means of calculation reuse. Section 4.1 demonstrates how the transform part in SATD may be reused to save operations. The idea is to also save energy, as in the SATD architecture that we proposed in (MONTEIRO; SEIDEL; GÜNTZEL, 2018). However, reusing HTs also has limitations, related to the TB. In fact, when considering the HTs reuse, because more computations are performed before the data enter in the TB, more bits are required and thus more energy is spent.

But only when the two mentioned properties are combined and the architecture components are rearranged it is possible to unleash their full potential in a novel generic SATD hardware architecture. Therefore, the main contribution of this chapter lies in exploiting both properties together to design such SATD architecture that occupies about 28% less area and requires 32% less energy than the state-of-the-art ones. Other contributions include:

1. A case study, presented in Section 5.3, where we described and synthesized our design to obtain realistic estimates of area and energy and compared our novel architecture with others from the literature; and

2. An area and power breakdown, presented in Section 5.3.3, to show the most demanding mod-

ules in our case study design, providing further insight to allow future improvements.

This chapter contains results of our works published in ISCAS2016, ICECS2016, LASCAS2018, and TCAS-I2019[65].

5.1 RELATED WORK

Although capable of improving coding efficiency, as shown in Chapter 3, the SATD is also more complex than tha SAD, as shown in Section 2.3. That is why a few works that focus on reducing the execution time of software-based encoders through SIMD include the SATD amongst the optimizations (CHEN, K. et al., 2012; ZHANG, Y. et al., 2017). In this sense, the correspondence between SIMD and RTL (LAHTI et al., 2018) seems very appropriate: as the software implementation of SATD demands optimization due to the large share of encoding time, its hardware optimization throughout the RTL flow is mandatory.

Considering hardware implementations of SATD, most of the complexity lies in the HT computation. The most common hardware implementations of HT are based upon the FHT and the separability property (thus requiring a transposition buffer) (BOLAÑOS-JOJOA; ESPINOSA-DURAN; VELASCO-MEDINA, 2014). FHT can be directly mapped to hardware as combinations of additions and subtractions, as shown in Figure 43a. The data flow of these combinations forms a butterfly-like diagram, just as in the FFT (BOLAÑOS-JOJOA; ESPINOSA-DURAN; VELASCO-MEDINA, 2014). In fact, HT is an example of a generalized 2D Fourier transform, with the advantage of requiring only integer additions and subtractions (TANG et al., 2018). The data flow shown in Figure 43a corresponds to the composition required for $n = 3$. A generic datapath is illustrated in Figure 43b.

The need for a transposition buffer can be traced back to Equation 2.19 and Equation 2.20. While in the former the operations are performed over columns, in the latter they are computed having rows as inputs. Figure 44 illustrates such process, which is also applied to DCT hardware architectures. Once HT was adopted in AVC as a secondary transform over the DC coefficients resulting from the DCT (MALVAR et al., 2003), multi-transform architectures to compute both DCT and HT have been proposed in the literature (AGOSTINI; PORTO, et al., 2006).

Vannerson and Lippincott (2007) propose an efficient transposition buffer targeting Static Random Access Memories (SRAMs), where each element of the first transformed matrix ($\mathbf{F}$) may be stored as a word. A dual buffer baptized as "ping-pong" was adopted in the DCT design from Agostini, Silva, and Bampi (2001) to increase the throughput while using the Block RAM (BRAM) of Field-Programmable Gate Arrays (FPGAs). Porto et al. (2005) presents five HT hardware alternatives targeting AVC, but only FPGA synthesis is considered. Their ping-pong buffer design was the one requiring less FPGA resources. Tumeo et al. (2007) present a

[65] Seidel, Bräscher, Güntzel, and Agostini (2016), Seidel, Güntzel, and Agostini (2016), Monteiro, Seidel, and Güntzel (2018), and Seidel, Monteiro, et al. (2019).

Figure 43 – FHT datapaths. (a) An 1D-HT considering a row/column with 8 elements (n=3). (b) Generic datapath. Adapted from: (SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016)

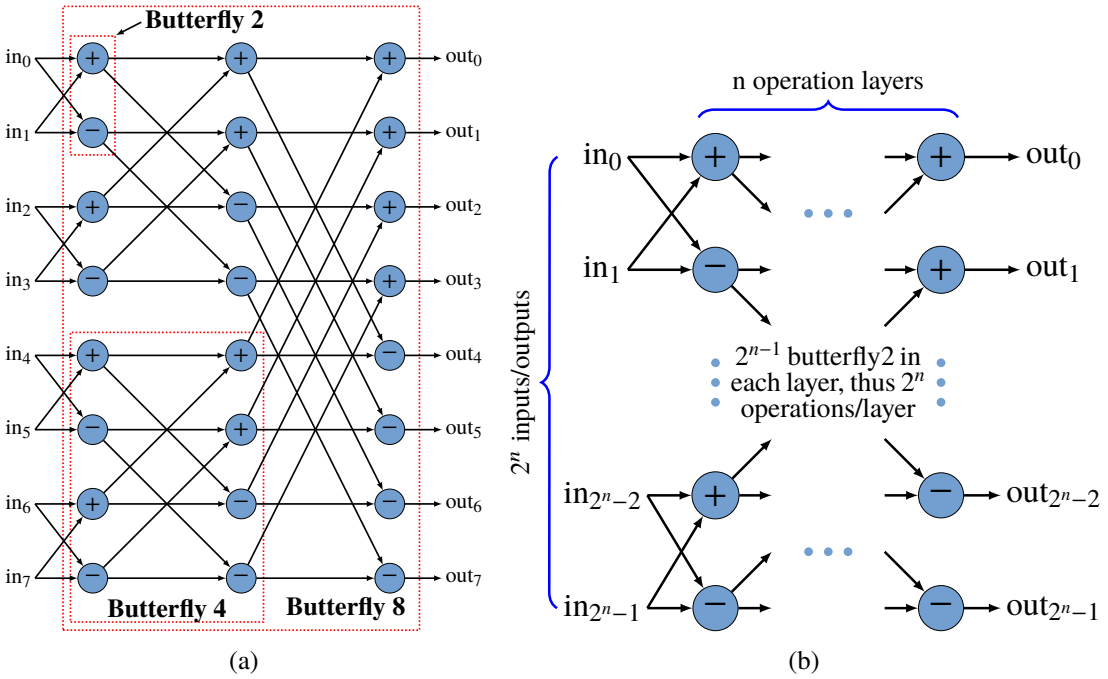
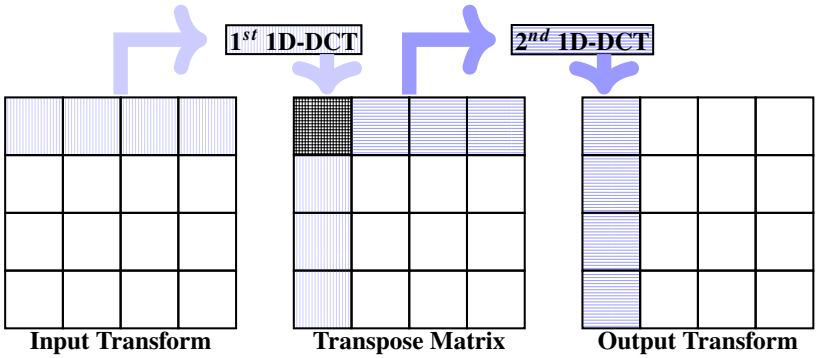


Figure 44 – Application example of the separability and the decomposition of transform computation in rows and columns such as in Equation 2.19 and Equation 2.20.



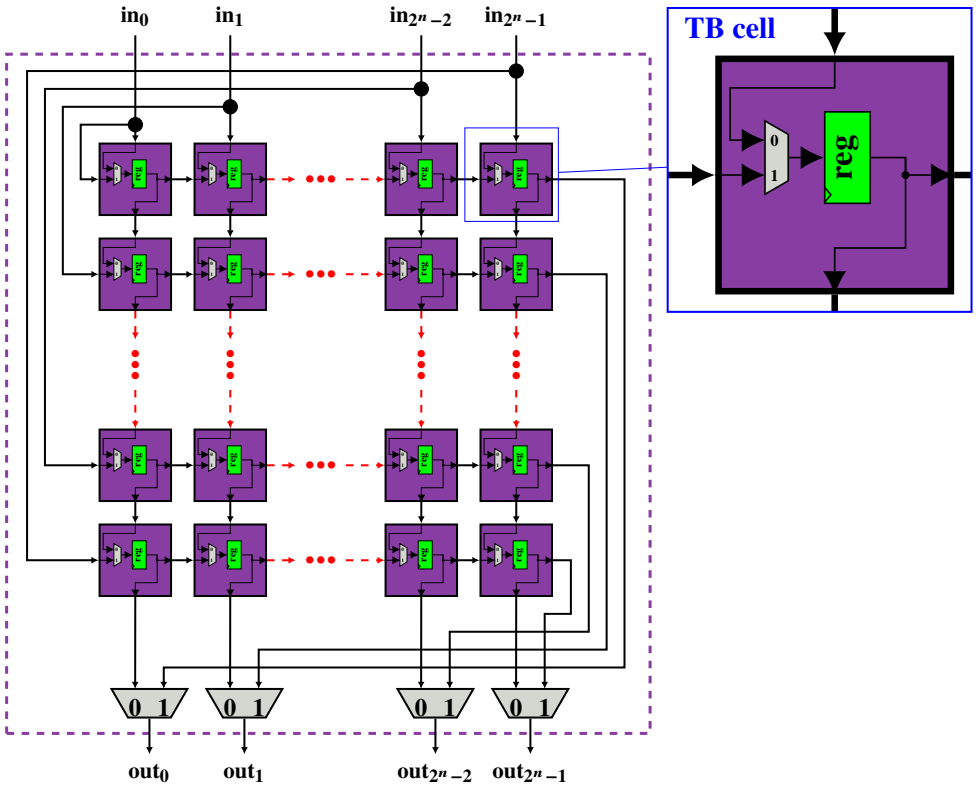
Source: adapted from Goebel et al. (2016).

similar approach for FPGA implementation of DCT, which also uses two buffers to implement transposition while using only one 1D DCT instance. El-Hadedy et al. (2010) present a proposal of a register array composed of shift registers capable of changing the shifting direction. A change in direction causes the transposition of the stored matrix. Such approach is similar to the one adopted by Liu, Zhou, et al. (2011) for their HT and SATD hardware architecture. The datapath of such transposition buffer is shown in Figure 45.

We presented a design space exploration of SATD hardware architectures in (CANCELLIER; BRÄSCHER, et al., 2014; CANCELLIER; SEIDEL, et al., 2015). The evaluated designs are:

1. $sa(\mathbf{T}(\mathbf{D}_4))$ using a TB as the one in Figure 50;

Figure 45 – Transposition buffer with configurable shift registers, also called TB. Inside each TB cell there is a multi-bit register (reg) to store the data. Such design is used by Liu, Zhou, et al. (2011), Cancellier, Bräscher, et al. (2014), Cancellier, Seidel, et al. (2015), Seidel, Bräscher, Güntzel, and Agostini (2016), A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017) and Monteiro, Seidel, and Güntzel (2018). The TB may be responsible for a significant share of the power consumed by a 2D transform (RITHE; CHENG; CHANDRAKASAN, 2012).



Source: adapted from Seidel, Bräscher, Güntzel, and Agostini (2016).

2. $sa(\mathbf{T}(\mathbf{D}_4))$ using 2D FHT (i.e., without the TB), called "Tree";

3. Tree with four pipeline stages.

By such exploration we were able to conclude that, for the adopted target period (31.25 ns) the most energy efficient design is the so-called Tree. However, this design required 2.74× larger area than the TB-based one. While for that target period TB uses 22% more energy than Tree, for maximum frequency, TB spends half the energy of Tree.

We also explored the behavior of SATD hardware architectures adopting the TE method (JOU, 2007; ZHU; XIONG, 2009). Such method was originally proposed for software implementation of the SATD. TE-SATD relies on Property 3 (Equation 5.1) to merge the last transform layer with the $sa()$ computation, saving operations without compromising the final SATD result.

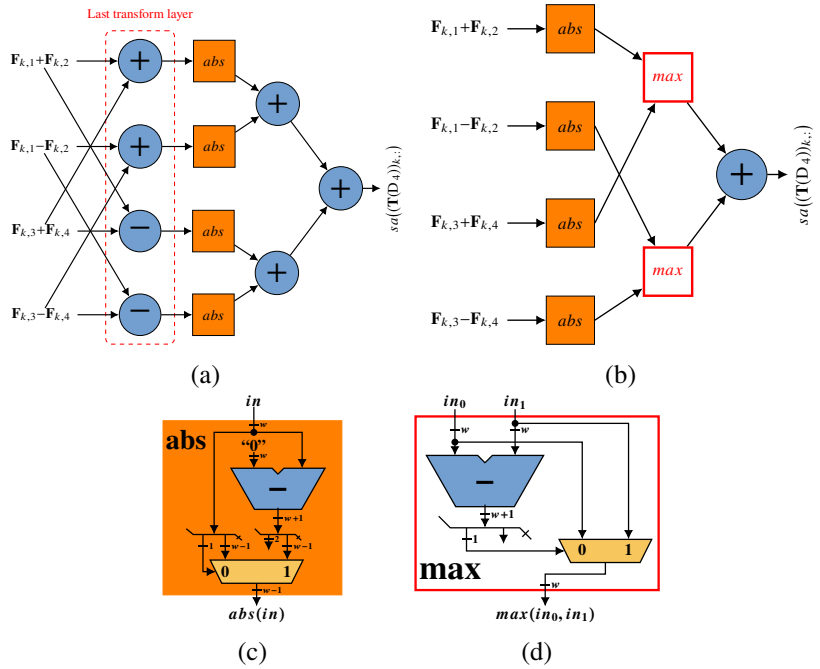
Property 3

(proved in (ZHU; XIONG, 2009))

$$|a+b|+|a-b| = 2\times max(|a|,|b|) \tag{5.1}$$

Figure 46 shows the datapaths for the last layer of the second 1D transform and the initial computation of the sum, compared to the same operation described after Property 3. Concerning the TB-based $sa(\mathbf{T}(\mathbf{D}_4))$ hardware, adopting TE occupied 9.8% smaller area and reduced energy by 15% (CANCELLIER; SEIDEL, et al., 2015).

Figure 46 – (a) Datapath of the last layer of the second 1-D HT and the *sa*, for obtaining the $sa(\mathbf{T}(\mathbf{D}_4))$. (b) Datapath for the same operation after applying Property 3 in the datapath show in (a). (c) hardware datapath of the absolute operation using only one adder and one mux. (d) "maximum of two" datapath, also using only one adder and one mux.

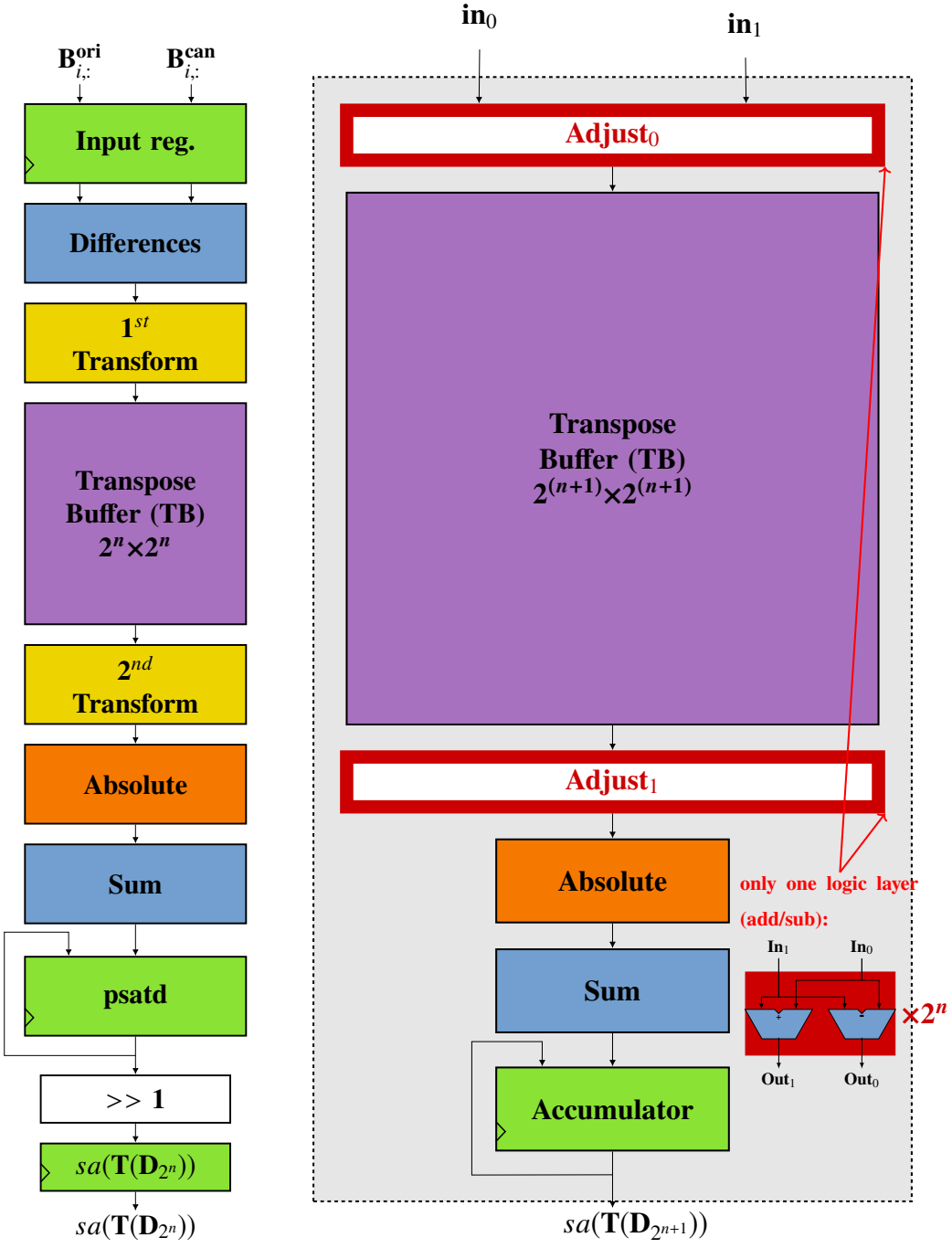


Source: the author ((c) and (d)). Adapted from Cancellier, Seidel, et al. (2015) ((a) and (b)).

We evaluated the use of TB vs. LB (PEREIRA et al., 2014) for $sa(\mathbf{T}(\mathbf{D}_{2^n}))$ with $n\in\{2,3,4,5\}$ in (SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016). Our TB-SATD datapath is presented in Figure 47a. Although LB occupies smaller area than TB (38.5% and 53.7% smaller for $sa(\mathbf{T}(\mathbf{D}_4))$ and $sa(\mathbf{T}(\mathbf{D}_8))$, respectively), it consumes more energy (124.9% and 70.6% more).

Silveira et al. (2015) present a $sa(\mathbf{T}(\mathbf{D}_8))$ and provide an insightful area and power breakdown of their architecture for standard-cell logic synthesis, showing that more than half of its area was occupied by registers. Also, such architecture requires twice the number of registers of our TB architecture from (SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016). Silveira, Ferreira, et al. (2017) proposed an architecture that resembles our Tree architecture, with a datapath reuse scheme similar to (LIU; ZHOU, et al., 2011). The novelty lies in the use of adder compressors, which are also explored in a TB-SATD architecture (SILVEIRA; ABREU,

Figure 47 – (a) The TB-based SATD architecture presented in (SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016). The 1st and 2nd Transforms are composed of butterflies such as the ones in Figure 43. (b) Datapath designed to reuse already computed HTs for $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ as presented in (MONTEIRO; SEIDEL; GÜNTZEL, 2018). It uses two instances of a $sa(\mathbf{T}(\mathbf{D}_{2^n}))$, such as the one in (a). Therefore, in_0 receives the outputs of the 2nd Transform of one instance, whilst in_1 receives the outputs of the 2nd Transform of the second instance. The so-called "Adjusts" are equivalent to the last layer of a 2^{n+1} butterfly.



Source: the author.

et al., 2017). Their use is able to reduce SATD energy consumption in about 14.4%, with 12.9% increase in area (SILVEIRA; ABREU, et al., 2017).

Although (LIU; ZHOU, et al., 2011) provides power estimates of their TB-based architecture with datapath reuse, their bit truncation strategy to reduce area and power by about 32.4% also reduces coding efficiency (2.42% BD-Rate with respect to AVC reference software). Another truncation scheme for $sa(\mathbf{T}(\mathbf{D}_4))$ is proposed by Soares et al. (2016), which reduces by up to 70% the energy with respect to the precise SATD, but also results in about 1% BD-Rate.

We proposed a coarse-grain (block-by-block) PDE scheme in (SEIDEL; GÜNTZEL; AGOSTINI, 2016), to reduce even more the energy consumption of Tree architecture, which still occupies a huge area. On the other hand, we adopted a fine-grain PDE scheme (row-by-row) to reduce the energy consumption of LB architecture in (BRÄSCHER, A. Beims; SEIDEL, I.; GÜNTZEL, J. L., 2017). However, TB-based architecture still consumes less energy for most cases.

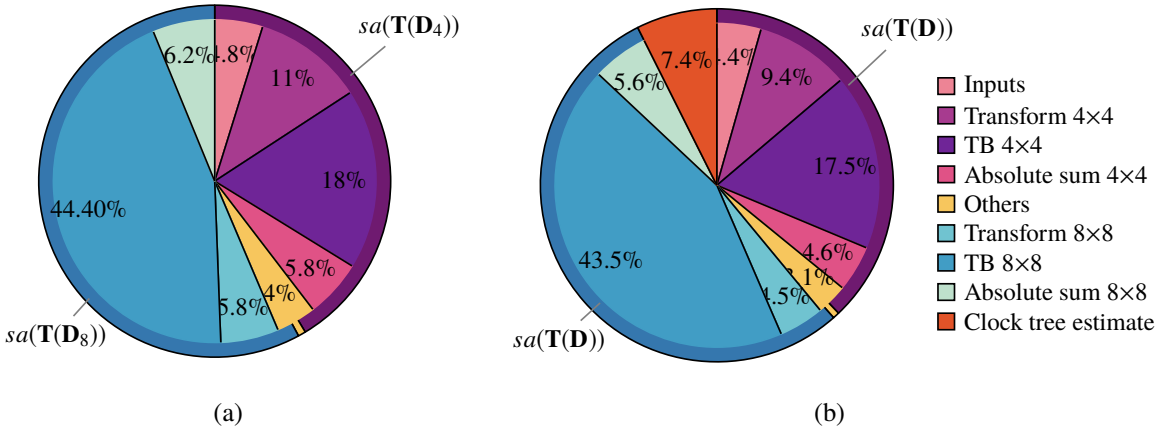
5.1.1 Works that reuse HT in SATD calculation

None of the works presented so far reuses previously computed HTs. On the contrary, He et al. (2015b) adopts Equation 4.14 in their FME hardware design for HEVC, which performs the Exhaustive HADs algorithm. Such algorithm computes the SATD with more than one size of transform and chooses the HT size which minimizes the SATD (computed as in Equation 2.21) to be the size adopted by the transform step. In their design, four $\mathbf{T}(\mathbf{D}_8)$ are reused in one $\mathbf{T}(\mathbf{D}_{16})$, and four $\mathbf{T}(\mathbf{D}_{16})$ are reused in one $\mathbf{T}(\mathbf{D}_{32})$. However, they used an SRAM to store the transformed values and their architecture requires 64 cycles solely for the transposition of $\mathbf{F}_8$ (see Equation 2.19).

Based on Property 1, we proposed a hardware design capable of reusing four previously computed $\mathbf{T}(\mathbf{D}_{2^n})$ to compute a $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ (MONTEIRO; SEIDEL; GÜNTZEL, 2018). Such design aims at a good compromise between area and energy efficiency and thus relies on the separability property (needs TBs). The reuse datapath for the $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ calculation is presented in Figure 47b. It also uses two instances of the datapath depicted in Figure 47a (connected to in_0 and in_1) to compute the four $sa(\mathbf{T}(\mathbf{D}_{2^n}))$.

As a proof of concept, we described and synthesized such architecture for n=2. Figure 48 shows area and power breakdowns for the target period adopted in (MONTEIRO; SEIDEL; GÜNTZEL, 2018). The TB $2^{n+1}\times 2^{n+1}$ is the most dominant module in both area and power estimates, confirming that the TB is the main culprit of power consumption in a TB-based SATD hardware architecture, as pointed out by Silveira et al. (2015). The reuse may have aggravated such dominance because more operations are done before the TB, increasing in 2^{2n+1} the total number of bits stored in such buffer. Therefore, it is evident that to reduce area and power we have to deal with the $2^{n+1}\times 2^{n+1}$ TB, the largest one in the architecture.

Figure 48 – Area and power breakdowns of our SATD reuse architecture presented in (MONTEIRO; SEIDEL; GÜNTZEL, 2018). To obtain the reports we resynthesized the architecture from (MONTEIRO; SEIDEL; GÜNTZEL, 2018) following the same method described in Sections 5.3.1 and 5.3.3. The slice marked as "Others" includes the control logic and the interconnections between modules. (a) Area breakdown. (b) Power breakdown.



(a) (b)

Source: the author.

5.2 A NOVEL HARDWARE ARCHITECTURE TEMPLATE

To achieve a reduction in the number of operations greater than that obtained by solely exploiting reuse, we applied Property 3 in the larger transform. Let $b_{i,j}$ be an element from the $2^{n+1}\times2^{n+1}$ TB before the module called "Adjust$_1$" (shown in Figure 47b). Such module operates as the last layer of a 2^{n+1} butterfly, thus its equidistant output values are the sum and the difference of the equidistant input values. After all $b_{i,j}$ go through Adjust$_1$, $\mathbf{T}(\mathbf{D}_{2^{n+1}})$ is completely computed. Then, each element from $\mathbf{T}(\mathbf{D}_{2^{n+1}})$ can be defined as:

$$td_{i,j} = \begin{cases} b_{i,j}+b_{i,j+2^n}, & \text{if } 1 \leq j \leq 2^n+1 \\ b_{i,j}-b_{i,j+2^n}, & \text{if } 2^n+1 < j \leq 2^{n+1} \end{cases} \tag{5.2}$$

Now, let us rewrite Equation 2.16 as:

$$sa(\mathbf{T}(\mathbf{D}_{2^{n+1}})) = \frac{1}{2^n}\sum_{i=1}^{2^{n+1}}\sum_{j=1}^{2^n} \mid td_{i,j} \mid + \mid td_{i,j+2^n} \mid \tag{5.3}$$

Applying Property 3 to Equation 5.3 after Equation 5.2:

$$sa(\mathbf{T}(\mathbf{D}_{2^{n+1}})) = \frac{1}{2^n}\sum_{i=1}^{2^{n+1}}\sum_{j=1}^{2^n} 2\times max\left(\mid b_{i,j} \mid, \mid b_{i,j+2^n} \mid\right) \tag{5.4}$$

From Equation 5.4 it is possible to notice that the absolute may be computed directly on the elements of the TB (i.e., $b_{i,j}$). Therefore, the module called "Adjust$_1$" (shown in Figure 47b) can be completely changed by an initial layer of absolute values of the TB outputs followed by the maximum computation. Moreover, we can save a few bits from the TB by computing the absolute before those elements enter the buffer (2^{2n+2} bits).

Far beyond a merely question of saving a few bits, Equation 5.4 also makes possible the main novelty of our proposed architecture, based on the following observation: 2^n cycles after $b_{i,j}$ is computed, $b_{i,j+2^n}$ becomes ready to enter the buffer. However, by Equation 5.4 it is clear that only the maximum value between these two will be used in the sum. Thus, instead of keeping $b_{i,j+2^n}$ for 2^n more cycles until transposition, as soon as $b_{i,j+2^n}$ is ready we can compare it to $b_{i,j}$ and discard the smallest one. This is only possible thanks to reuse, since all the other layers of the larger transform are computed before the larger TB. Figure 49 shows a datapath tailored for this operation.

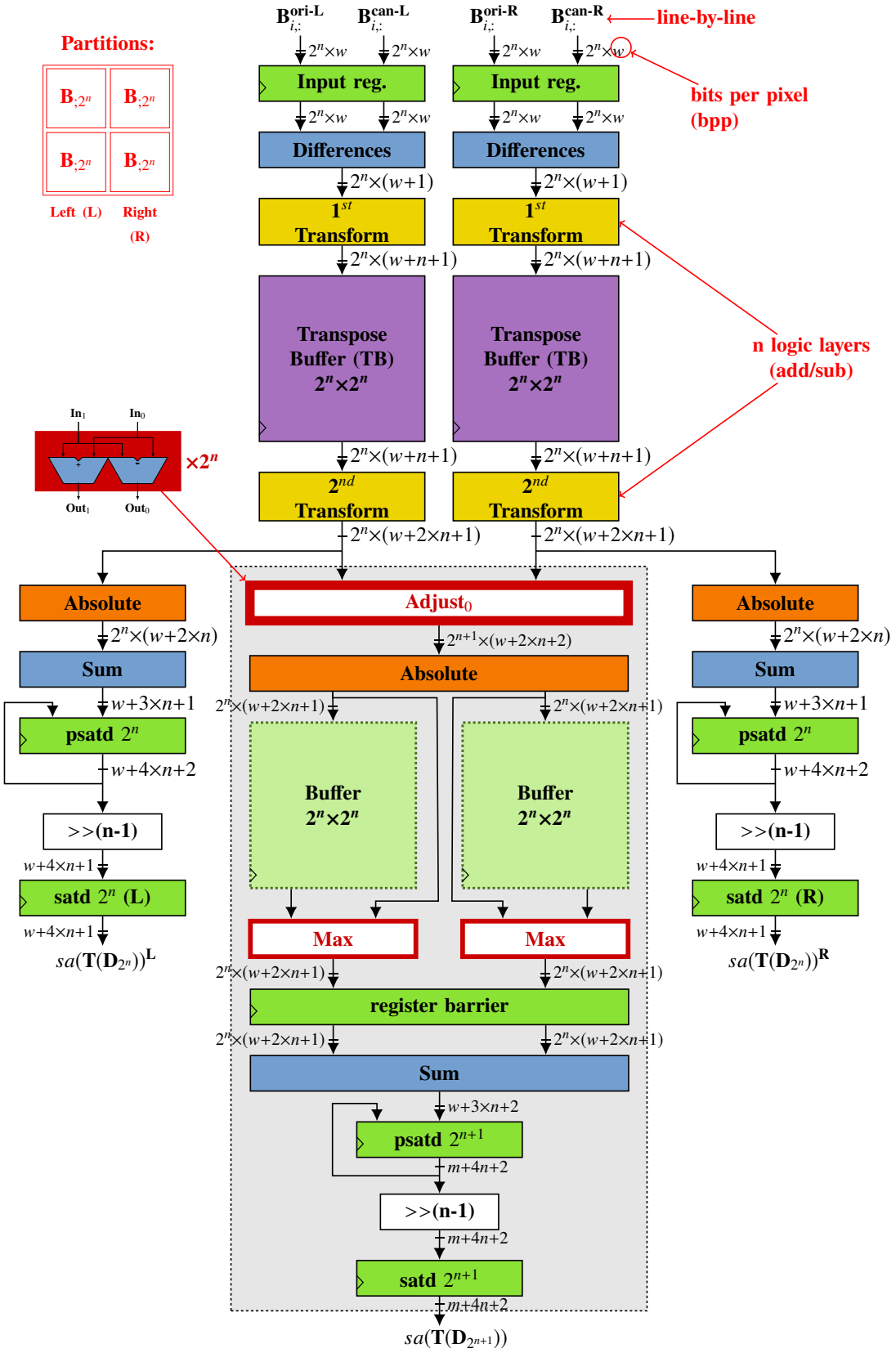
Moreover, no transposition is required because the comparisons are made directly between the same positions as the values are generated. The design of the adopted buffer is presented in Figure 50. As can be seen in Figure 49, two instances of this addressable buffer are required. Yet, these two instances together have half the total amount of registers than required by the $2^{n+1}\times2^{n+1}$ TB, shown in Figure 47b. An extra advantage of using addressable buffers is the reduction in the switching activities of the used registers, since their values may change only once every 8 cycles, instead of possibly changing every cycle as in the transpose buffer design in Figure 45.

To sum up, the novelty of our hardware architecture template lies in removing half of the $2^{n+1}\times2^{n+1}$ TB and replacing the other half by two $2^n\times2^n$ addressable buffers, as the one shown in Figure 50. Therefore, beyond the savings achieved by reuse Equation 4.14 and Property 3, our design saves half the larger TB, which was the module with the largest area and power dissipation in (MONTEIRO; SEIDEL; GÜNTZEL, 2018), as shown in Figure 48. Notice that none of the proposed hardware optimizations jeopardize coding efficiency because they do not change the SATD result itself.

Figure 51 shows the FSM used to control the datapath from Figure 49. The control signals issued by such FSM are shown by the timing diagram in Figure 52. An asynchronous reset has as arriving state `IDLE`, where the FME will stay until the input signal `enable`=1. This changes the current state to $\mathtt{I}_0$, which makes the input registers enabled. There are 2^{n+1} initialization states, named $\mathtt{I}_\omega$ ($0 \leq \omega < 2^{n+1}$), during which the inputs are loaded and the internal architecture registers are gradually enabled. Similarly, there are 2^{n+1} main execution loop states ($\mathtt{C}_\omega$) and the same number of finalization states ($\mathtt{F}_\omega$).

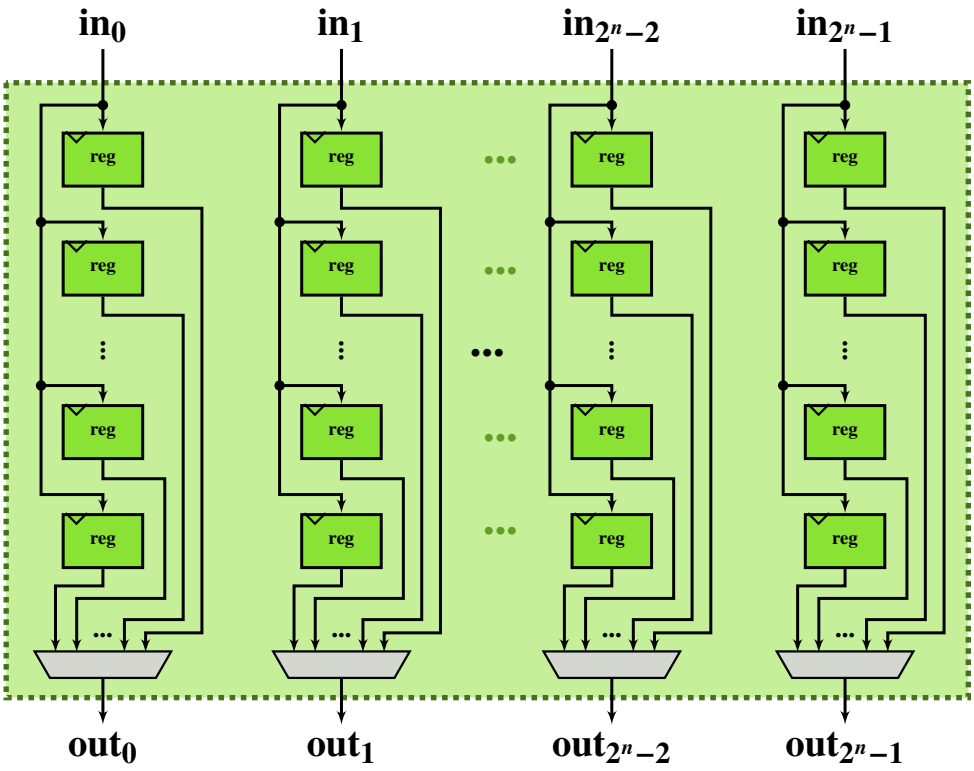
If signal `enable` is active after 2^{n+1} cycles from the beginning of the execution, the FSM enters the main execution loop. The first pair of $sa(\mathbf{T}(\mathbf{D}_{2^n}))$ are ready in state $\mathtt{C}_1$ and the FSM raises signal `done_2`^{n}, as shown in Figure 52. Then, after another 2^n cycles the second pair of $sa(\mathbf{T}(\mathbf{D}_{2^n}))$ becomes available in the outputs, raising again signal `done_2`^{n}. After only one cycle the first $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ finishes, now raising signal `done_2`$^{n+1}$. Thus, while `enable` is '1' in state $\mathtt{C}_{2^{n+1}-1}$ another loop begins, issuing two pairs of $sa(\mathbf{T}(\mathbf{D}_{2^n}))$, one in $\mathtt{C}_1$ and another in $\mathtt{C}_{2^{n+1}-3}$ while issuing one $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ every 2^{n+1} cycles in state $\mathtt{C}_{2^{n+1}-2}$.

Figure 49 – RTL datapath schematic of the proposed template to reuse HTs in SATD. Notice the datapath in the shaded area containing from Adjust_0 to the register satd 2^{n+1} performs the same computation as the datapath shown in Figure 47b, but using a smaller buffer.



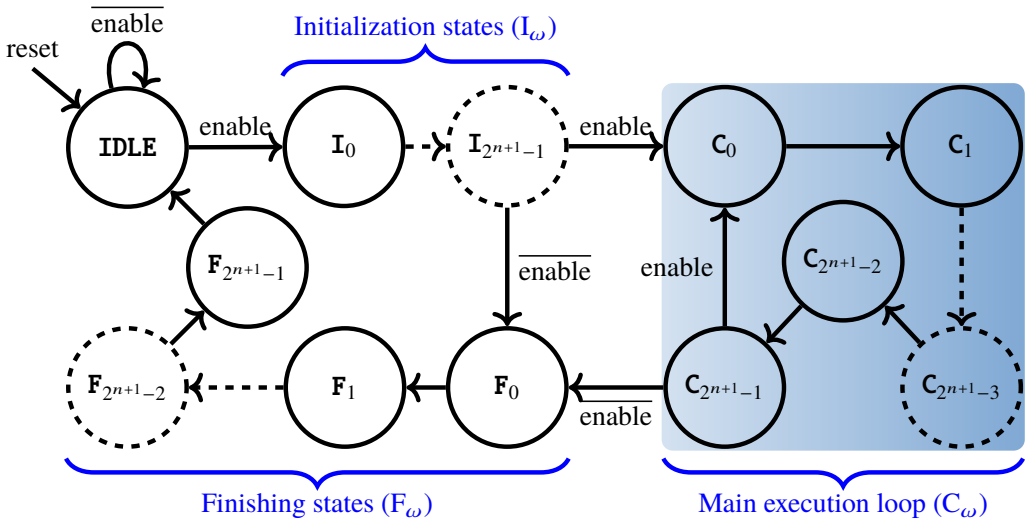
Source: the author.

Figure 50 – New $2^n \times 2^n$ buffer design. It works similarly to an SRAM, although in our evaluation it was synthesized as a register bank. Indeed, for a larger n it would be possible to use an SRAM instead. Because two instances are used, it can be seen as a 4×8 buffer.



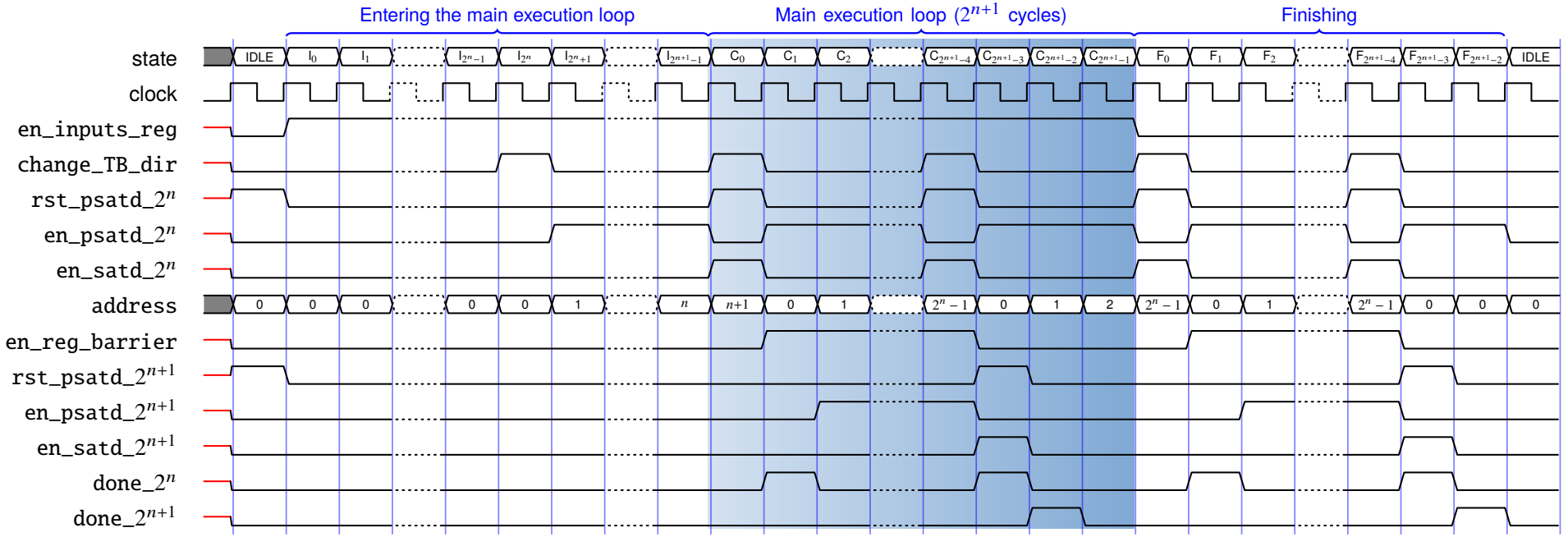
Source: the author.

Figure 51 – Control (Finite State Machine (FSM)) of the architecture. Each dashed node represents a set of states, the number of states being equal to the difference between the index of the dashed node set and the index of the previous node. Within the sets of states there are no conditions for state transition and thus each state lasts exactly a single clock cycle.



Source: the author.

Figure 52 – Timing diagram of our design template. It shows the signals generated by the FSM depicted in Figure 51, used to control the datapath from Figure 49. The signals initiating in `en` and `rst` are enable and reset signals, respectively. Also, every 2^n cycles the direction of the $2^n \times 2^n$ TBs is changed. The register barrier is disabled during 2^n cycles, during which the four "adjusted" lines enter the 4×8 buffer. Then, for the next 2^n cycles the values from the 4×8 buffer are read and compared to the absolute values from "Adjust$_0$" to decide which will be written to the register barrier, now enabled.



Source: the author.

The finishing states begin when enable is '0' in state $\mathtt{C}_{2^{n+1}-1}$ (or in state $\mathtt{I}_{2^{n+1}-1}$). In $\mathtt{F}_0$ the input registers are disabled thus no new data are read in. A pair of $sa(\mathbf{T}(\mathbf{D}_{2^n}))$ will be done in state $\mathtt{F}_1$ whereas the last pair of $sa(\mathbf{T}(\mathbf{D}_{2^n}))$ will become available in state $\mathtt{C}_{2^{n+1}-2}$. Finally, the last $sa(\mathbf{T}(\mathbf{D}_{2^{n+1}}))$ will be finished in state $\mathtt{F}_{2^{n+1}-1}$ so leaving this state in the next clock cycle arriving back to IDLE.

5.3 CASE STUDY: $sa(\mathbf{T}(\mathbf{D}_4))$ AND $sa(\mathbf{T}(\mathbf{D}_8))$

We created a case study of our hardware template by assuming n=2 (i.e., it is able to compute $sa(\mathbf{T}(\mathbf{D}_4))$ and $sa(\mathbf{T}(\mathbf{D}_8))$) to compare with architectures from the literature, including our own from (MONTEIRO; SEIDEL; GÜNTZEL, 2018). These are the same sizes used in the coding efficiency analysis presented in Chapter 3. Furthermore, as such analysis considered the use of SATD in both IME and FME, our target periods will be based on such use scenarios. Yet, as we aim to evaluate the proposed SATD micro-architecture, we introduce only one organization for each scenario.

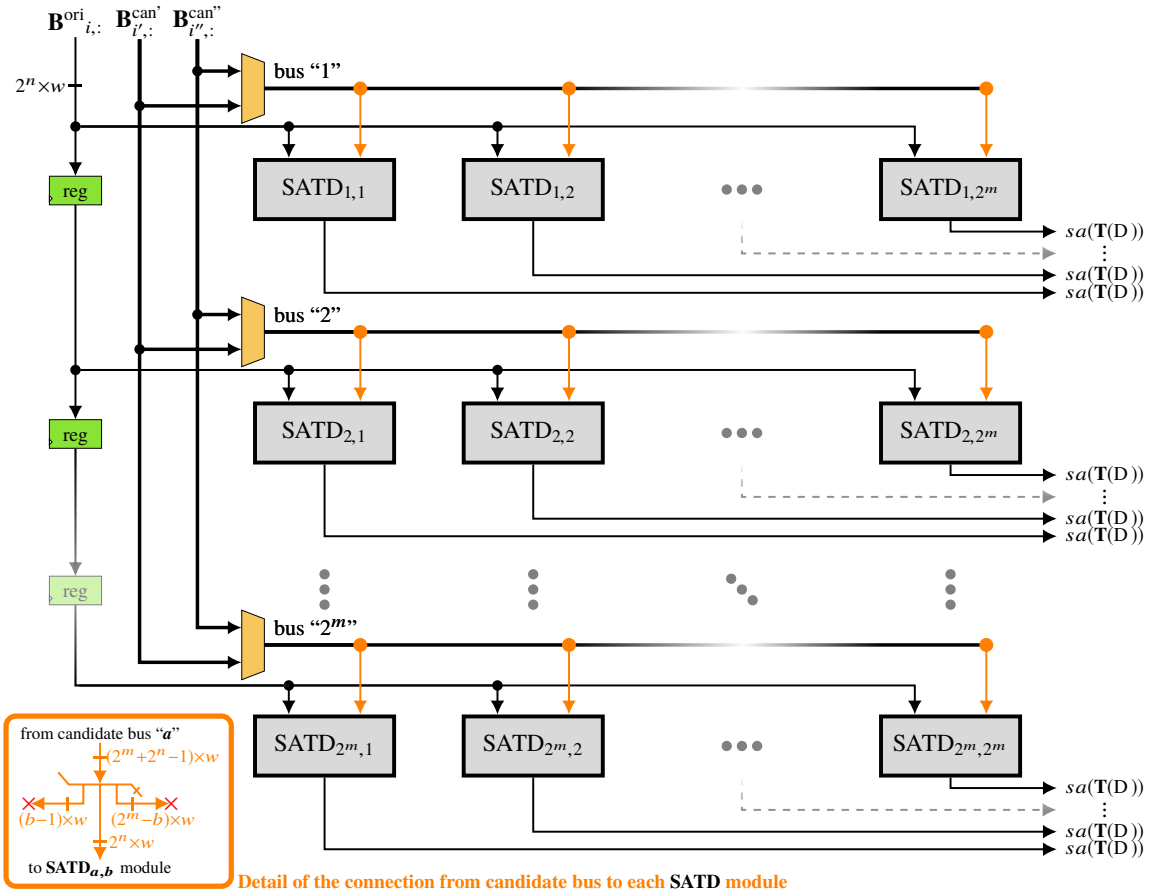
Figure 53 depics a possible organization build up from $2^m \times 2^m$ exemplars of the proposed SATD architectures to perform the FBMA in IME. It resembles the one proposed by Kim and Park (2009) and thus, it may be used to efficiently perform the VBSME. The original block ($\mathbf{B}^{\text{ori}}$) is read line-by-line and the candidates pixels are read in groups, depending on how large is m. For an encoder configuration with a search window similar to the one used in Section 3.2.1, m=3 thus 64 SATD modules are instantiated. In this case, after 16 initialization cycles, the FBMA architecture will finish 8 SATD calculations every cycle in one line of 8 SATDs at a time.

Concerning the FME, one energy-efficient approach is to compute the distortion of all 48 candidates surrounding the one selected in IME (AFONSO, Vladimir et al., 2016; SEIDEL; FILHO, et al., 2018). In the organization adopted by Vladimir Afonso et al. (2016) and Seidel, Filho, et al. (2018), six or twelve SADs are computed in parallel, depending on the number of neighbor candidates generated by the interpolation filter. During the total of 51 cycles to finish the FME of an 8×8 block, the architecture first searches in parallel the 6 so-called horizontal candidates, then 6 vertical candidates and the remaining 36 candidates are computed in parallel groups of 12, when the architecture achieves maximum utilization. Details of such FME hardware architectures are presented in Chapter 6.

5.3.1 Evaluation method

We synthesized the architectures for four distinct periods: 31.25 ns, 20 ns, 2.5 ns and 2 ns. The first one (31.25 ns) is to achieve a target throughput of 16 million 4×4 blocks/s, as used by Walter, Diniz, and Bampi (2012), Seidel, Bräscher, Güntzel, and Agostini (2016), A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017) and Monteiro, Seidel, and Güntzel (2018). The second one (20 ns) is the same used to achieve real time encoding of 1920×1080@30fps sequences in FME hardware designs such the one we present in Chapter 6.As mentioned, each

Figure 53 – Possible SATD organization for FBMA. The execution flow works by enabling the SATDs line-by-line, from the top ones to the bottom ones. The number of candidate buses connected to the multiplexers inputs will depend upon the search area. They are required to allow for computing the top lines of the search area while finishing the bottom ones.

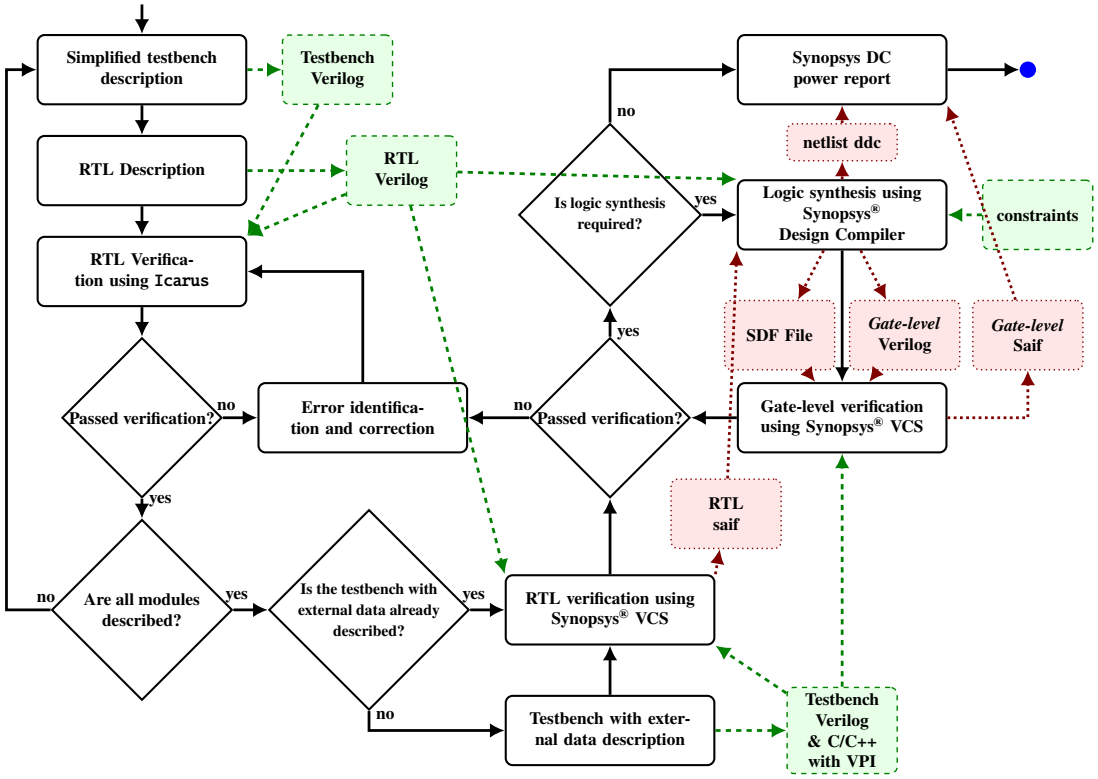


Source: the author.

FME takes 51 cycles. Thus, in each 1920×1080 original frame (see Figure 11) there are 32,400 8×8 original blocks. So, their FME for one reference frame takes 32,400×51×30=49.572 million cycles/s. The third (2.5 ns) also considers FME, but to achieve a target throughput of 3840×2160@60fps. Notice that 12 parallel SATD architectures are needed in that FME design. Finally, the fourth period (2 ns) is the necessary to achieve 1920×1080@30fps encoding using FBMA in a 64×64 candidates search area, which is similar to the FBMA configuration shown in Section 3.2.1. By adopting the organization shown in Figure 53 for m=3, 64 parallel SATD modules are necessary.

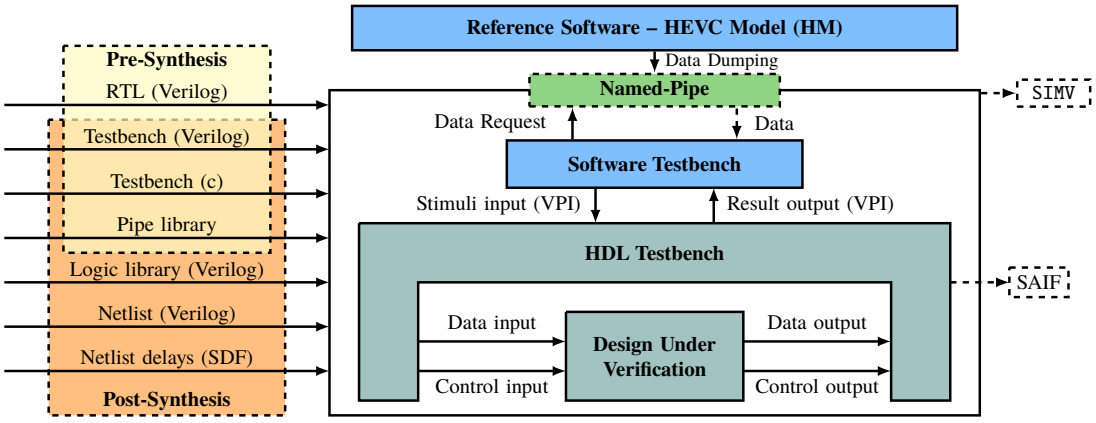
The main steps we followed to obtain area and power estimates are shown in Figure 54. The same method was also adopted to compare with the SATD hardware architectures from Seidel, Bräscher, Güntzel, and Agostini (2016) and Monteiro, Seidel, and Güntzel (2018), since we have access to those RTL descriptions. All architectures were synthesized with Synopsys® Design Compiler (DC®) (SYNOPSYS, 2016) in Topographical mode to estimate routing parasitics and thus tightly correlating with post-layout timing, area and power. All syntheses used a 45 nm standard cell library from Taiwan Semiconductor Manufacturing Company Limited

Figure 54 – Design and synthesis method to ensure reliable area and power estimates. The dashed boxes contents are files described by the designer whereas dotted boxes are files generated by commercial tools.



Source: the author.

Figure 55 – Adopted simulation model using the framework proposed by Bonotto et al. (2018). It was used for RTL and Gate-level verifications using Synopsys® Verilog Compiler Simulator (VCS®) (Figure 54), before and after the logic synthesis with DC.



Source: adapted from Bonotto et al. (2018).

(TSMC) (TSMC, 2011). For all syntheses, input and output delays were conservatively limited to 60% of the clock period. The maximum primary input capacitance was set to 10 times a 2-input AND gate whereas the maximum primary output capacitance was set to 30 times a 2-input AND gate.

We simulated our architecture using the framework that we proposed in (BONOTTO

et al., 2018), illustrated in Figure 55. Such simulation, compiled using VCS®, is a means for two ends: 1) to perform a functional verification of our architecture both prior and after synthesis and 2) to obtain realistic switching activity. Notice in Figure 54 that we used the gate-level Switching Activity Interchange Format (SAIF) to obtain the final power estimates. This is to capture the synthesized netlist switching activity using real stimuli from a video sequence. Moreover, we use the netlist Standard Delay Format (SDF) generated by DC®, thus accounting for temporal glitches in the netlist during simulation which also influence power estimates (SILVEIRA; PAIM, et al., 2017).

We used 100,000 8×8 original blocks and related candidates from BQTerrace sequence for each simulation and QP 22. This is the same amount used to evaluate the SAD architectures by Silveira, Paim, et al. (2017), that adopt a similar method than ours. They found through a sensitivity test that 50,000 tests are sufficient to stabilize the power estimates and doubled such figure to ensure reliability. BQTerrace was used as source for the simulation stimuli because it has higher TI and SI than the other class B (BOSSEN, 2012) tested sequences (see Figure 23). Thus, it should provide us with a reliable upper bound for power estimates. Moreover, we chose it instead of any other higher resolution ones because the prediction is that UHD will only represent 22.3% of IP video traffic by 2022, while HD resolution will represent about 56.8% (CISCO, 2018).

5.3.2 Results and comparison with related works

Table 15 shows the results presented in the related works that synthesized SATD architectures to standard cells and have no reduction in coding efficiency along with our obtained results. The presented power estimates of this work are all considering realistic switching activity, obtained as specified in Section 5.3.1. The average energy for each $sa(\mathbf{T}(\mathbf{D}_8))$ and four $sa(\mathbf{T}(\mathbf{D}_4))$ was computed as:

$$\text{Energy} = \text{Period}\times\text{Cycles}\times\text{Total Power} \tag{5.5}$$

The target periods, number of cycles and total power figures to estimate energy are in Table 15. The main execution loop (see Figure 51) of our case study takes 8 cycles to compute one $sa(\mathbf{T}(\mathbf{D}_8))$ and four $sa(\mathbf{T}(\mathbf{D}_4))$. The smallest energy estimate was for 2.5 ns period, because in this case the relative increase in power was smaller then the decrease in execution time (Period×Cycles) when compared to longer periods. On the other hand, the increase in power (+1739.36 µW) when the target period is reduced from 2.5 ns to 2 ns is larger than the benefit of reducing time (-4 ns) and thus the energy estimate increases.

It is possible to notice that the largest increase in the occupied area was from 2.5 ns to 2 ns synthesis. This is because more critical paths required gates with larger driving strengths (i.e., larger gates) to meet the tighter timing constraints. Such larger increase in area also reflected in the larger increase in the static power, as expected.

Table 15 – Synthesis results, as reported by the related works that synthesized SATD hardware architectures for standard cells and do not present coding efficiency losses, and our current work.

Work	Library	2^n	Target Period (ns)	Cycles/ $sa(\mathbf{T}(\mathbf{D}_{2^n}))$	Area (μm^2)	Stat. (μW)	Dyn. (μW)	Total (μW)	Energy (pJ/$sa(\mathbf{T}(\mathbf{D}_{2^n}$	Simul.
Cancellier, Seidel, et al. (2015)	TSMC 45 nm	4	20.80	3	5036.93	65.36	152.51	217.87	13.60	Yes
Silveira et al. (2015)	NanGate 45nm	8	1.69	8	12231.00	383.20	3382.40	3765.60	50.85	Yes*
Seidel, Bräscher, Güntzel, and Agostini (2016)	TSMC 45 nm	4	15.625	4	2943.23	32.09	125.89	157.98	7.8681	No
		8	31.25	8	9468.98	112.79	311.80	424.59	77.95	No
Seidel, Güntzel, and Agostini (2016)	TSMC 45 nm	4	31.25	variable	6711.67	86.23	73.36	159.60	8.09	No
		8	125.00	variable	41837.85	587.81	130.29	718.10	161.60	No
Silveira, Ferreira, et al. (2017)	NanGate 45nm	4	8.00	4	70321.00	596.93	6816.13	7412.77	237.20	Yes
		8	8.00	8	70321.00	596.93	6816.13	7412.77	474.41	Yes
A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017)	TSMC 45 nm	4	2.84	variable	2129.15	–	–	343.25	7.88-18.49	No
		8	3.38	variable	4661.90	–	–	563.03	56.60-129.09	No
Silveira, Abreu, et al. (2017)	NanGate 45nm	8	3.00	8	16729.00	146.10	4855.77	5001.86	120.04	Yes
Monteiro, Seidel, and Güntzel (2018)	TSMC 45 nm	4 and 8	31.25	8	14977.90	143.90	306.50	450.40	112.60†	No**
This work	TSMC 45 nm	4 and 8	31.25	8	11251.85	170.68	454.92	625.60	156.40†	Yes
			20.00	8	11253.61	170.46	683.58	854.04	136.65†	Yes
			2.50	8	11935.58	180.52	5257.60	5438.12	108.76†	Yes
			2.00	8	13973.70	247.78	6929.70	7177.48	114.84†	Yes

* Simulated with random stimuli; ** Provided simulation results, but the ones in this table are without simulation. For fair comparison, we re-simulate this architecture.

† pJ/$(sa(\mathbf{T}(\mathbf{D}_8)) + 4 \times sa(\mathbf{T}(\mathbf{D}_4)))$

To provide fair comparisons, we resynthesized our previous architectures from Seidel, Bräscher, Güntzel, and Agostini (2016) and Monteiro, Seidel, and Güntzel (2018) following the same method, using the same tool version and under the same constraints as the ones used for the architecture of this work. Although we have the RTL from Seidel, Güntzel, and Agostini (2016) and A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017), we do not resynthesized the Tree architecture from Seidel, Güntzel, and Agostini (2016) because it occupies much larger area and the LB-based one from A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017) spends much more energy than a TB-based one, as shown in (SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016; BRÄSCHER, A. Beims; SEIDEL, I.; GÜNTZEL, J. L., 2017). For comparison purposes, we also modified the reuse architecture from Monteiro, Seidel, and Güntzel (2018) to adopt Property 3 only in replacement of Adjust_1, but keeping the larger TB. In Table 16 such architecture appears as "Modified Monteiro, Seidel, and Güntzel (2018)". Also, for such direct comparisons we considered only one instance of $sa(\mathbf{T}(\mathbf{D}_4))$ from the related works when possible. The left hand side of Table 16 shows a relative area comparison of our case study architecture with respect to the resynthesized architectures (as baselines). It is possible to notice that our novel architecture is the smaller one, saving from 10% up to 38% in area.

The direct area comparison with the other designs from Table 15 may be biased by the used synthesis tool, constraints and standard-cell library. Therefore they may reflect these differences as well as the differences in the architecture. Considering the $sa(\mathbf{T}(\mathbf{D}_8))$ architecture from Silveira et al. (2015), our architecture is 14% larger. Yet, our design is able to compute four $sa(\mathbf{T}(\mathbf{D}_4))$ during the 8 cycles it computes $sa(\mathbf{T}(\mathbf{D}_8))$. On the other hand, our architecture is smaller than the $sa(\mathbf{T}(\mathbf{D}_8))$ architecture from Silveira, Abreu, et al. (2017). It is because their architecture is completely parallel, similar to the Tree-based one from Seidel, Güntzel, and Agostini (2016), and occupies more area than a TB-based based SATD architecture. Also, as expected, the area occupied by the LB-based architectures from A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017) is much smaller than ours. Yet, as A. Beims Bräscher, I. Seidel, and J. L. Güntzel (2017) show, it may also demand twice as power as a TB-based one.

Table 16 – Improvements in terms of area and energy over related works.

Related architecture (as baseline)	**Relative area[a] for each target period**				**Relative energy[e] for each target period**			
	31.25 ns	**20 ns**	**2.5 ns**	**2 ns**	**31.25 ns**	**20 ns**	**2.5 ns**	**2 ns**
Seidel, Bräscher, Güntzel, and Agostini (2016)	0.84	0.84	0.68	–*	0.78	0.75	0.60	–*
Seidel, Bräscher, Güntzel, and Agostini (2016)**	0.68	0.68	0.62	0.67	0.74	0.72	0.67	0.66
Monteiro, Seidel, and Güntzel (2018)	0.72	0.72	0.74	0.84	0.71	0.69	0.69	0.68
"Modified Monteiro, Seidel, and Güntzel (2018)"	0.76	0.76	0.79	0.90	0.80	0.79	0.82	0.82

[a] Relative area computed as $\frac{\text{area from the proposed architecture}}{\text{area from the } sa(\mathbf{T}(\mathbf{D}_8))+sa(\mathbf{T}(\mathbf{D}_4)) \text{ related architectures}}$.
[e] Relative energy computed as $\frac{\text{energy of the proposed architecture}}{\text{energy of the } sa(\mathbf{T}(\mathbf{D}_8))+4\times sa(\mathbf{T}(\mathbf{D}_4)) \text{ from related architectures}}$.
*violated timing in the $sa(\mathbf{T}(\mathbf{D}_4))$ architecture; **considering two instances of $sa(\mathbf{T}(\mathbf{D}_4))$.

The right hand side of Table 16 shows an energy comparison with related works. It is

possible to see in the results that our architecture is also the most energy efficient among the listed ones. Also, the modified version of (MONTEIRO; SEIDEL; GÜNTZEL, 2018) saves less energy than our novel architecture. Among all comparisons, our novel architecture is the most energy-efficient one: saving from 18% up to 40% in energy.

Similar to the area comparison, energy comparisons with different tools and library are also biased by these differences besides architectural ones. Moreover, as the energy depends on the power estimates, which by its turn depends on the circuit switching activity, the used simulation data may influence the energy estimates. To allow for reliable and fair future comparisons with our current case study architecture we made its RTLs available in Gitlab: `https://gitlab.com/eclufsc/videocoding/satd_4_8_rtl_description`.

5.3.3 Area and Power Breakdown

To get detailed area and power reports, we re-synthesized our case study architecture while restricting DC® to keep the RTL hierarchy during syntheses. Forcing the synthesis tool to keep hierarchy resulted in 0.95% larger area and 5% more power, on average. These detailed breakdown results are presented in Figure 56 for the lowest and highest frequencies.

The "TB 4×4" and "Buffer 4×8" shares are close because they have almost the same size. However, although the 4×8 buffer has a slightly larger number of bits per element than TB 4×4, its switching activity is smaller. Yet, we must consider the need of the register barrier, which also has considerable toll in area and power. Such barrier may be moved by the synthesis tool when hierarchy boundaries are removed and thus save power through register retiming (SAPATNEKAR, 2004).

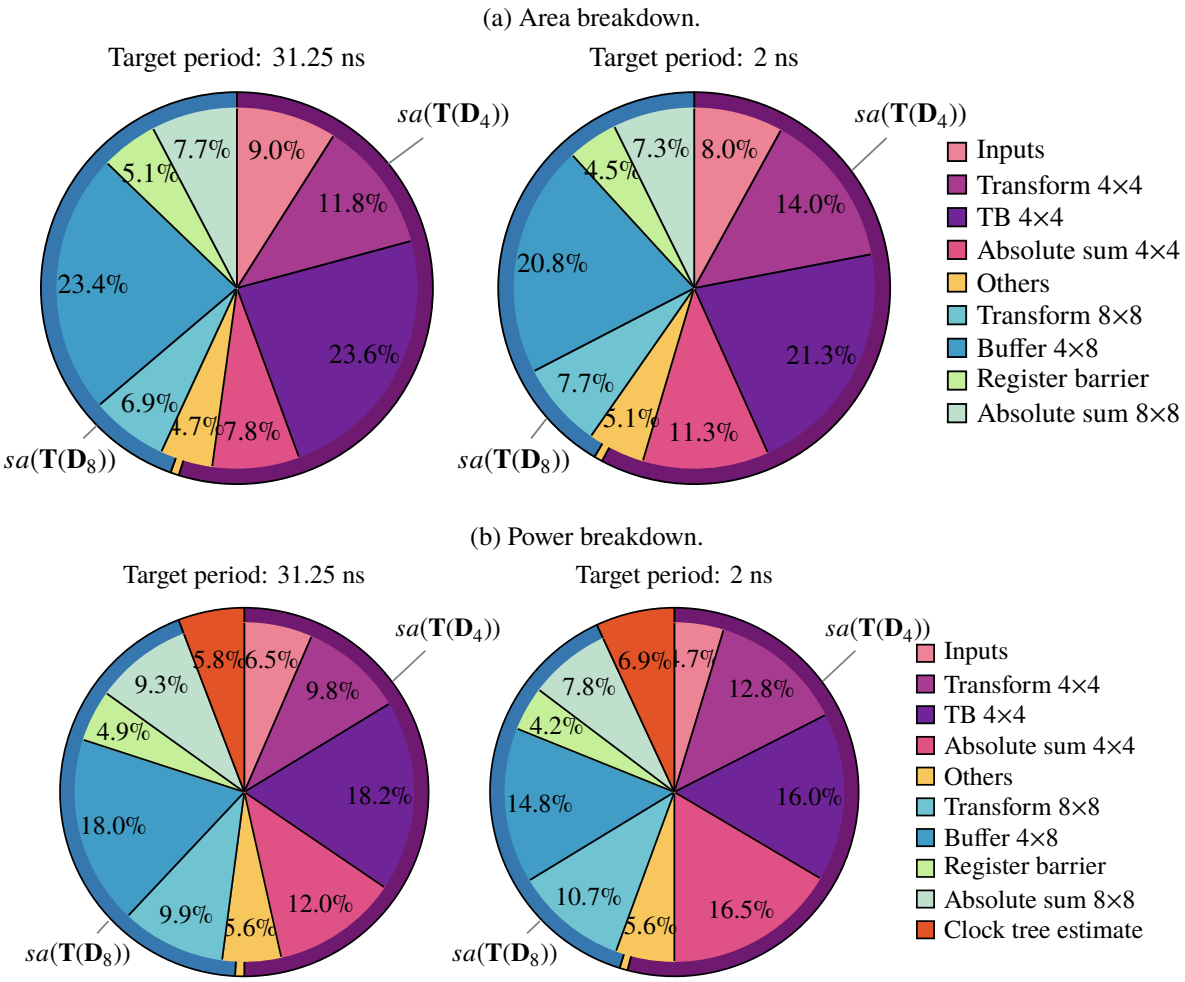
Comparing the results for 31.25 ns with the ones from Monteiro, Seidel, and Güntzel (2018) shown in Figure 48 it is possible to notice that we successfully reduced the share of the 8×8 buffer from 44.4% and 43.5% to 23.4% and 18% of the 4×8 buffer in area and power, respectively. Moreover, the absolute estimates of our half buffer design are smaller than the absolute estimates from the reuse design from Monteiro, Seidel, and Güntzel (2018), as can be seen in Table 16.

5.4 CHAPTER CLOSING REMARKS

In this chapter we proposed a novel SATD hardware architecture template that was designed by exploiting together two mathematical properties featured by the SATD to reduce area and energy consumption. By doing so, we could remove half of the larger buffer, which is responsible for the largest share of area and power consumption in our previous SATD architecture based on reuse solely.

To allow for comparison with related works, we described, synthesized and evaluated a case study of our architecture template tailored to compute the SATD of the two sizes more commonly found in literature, i.e., 4×4 and 8×8. Compared to our SATD architecture based

Figure 56 – Area and power breakdown of the SATD architecture. The outer ring represents the share of each main block (i.e., how much each SATD size contributed for the total).



on HT reuse solely, the proposed one occupies up to 28% less area and saves up to 32% in energy. Compared to other SATD hardware architectures, depending upon configuration, our architecture occupies 38% smaller area with 33% energy reduction or 32% smaller area with 40% energy reduction. For reproducibility purposes, we made our RTL description publicly available

By analyzing the number of reused operations, presented in Table 14, and buffer size reduction in our novel SATD hardware architecture, it is possible to conclude that the relative savings grow for larger HTs. Since the relative savings are more prominent when larger SATD are used, the proposed design has the ability to further improve energy efficiency. This makes it a candidate to be adopted in implementations of upcoming video encoders, such as the VVC and EVC, should larger block sizes be adopted[66].

[66] In this case, proper evaluation of the coding efficiency provided by using SATDs with larger size must be done.

Part III

FME Hardware Design

6 INITIAL FME HARDWARE DESIGN

Considering its importance, in terms of coding efficiency and its high complexity[67], the FME of current and forthcoming standards is a natural candidate to be implemented in hardware. As main contribution, this chapter proposes and evaluates a design strategy for FME hardware accelerators which may be used in future video CODECs. The resulting architecture has the following features.

1. **Coding efficiency**: it searches in all fractional positions while considering the weighted rate, in a simplified RDO;

2. **Energy efficiency**:

    - it has a regular dataflow and thus a high degree of parallelism,
    - performs no redundant filter operations,
    - avoids extra Motion Compensation (MC),
    - is memory-aware, thus avoiding unnecessary memory accesses,
    - keeps only useful data in internal buffers, which are designed to minimize wire-length.

As secondary contribution, we present area and power breakdowns of the architecture, identifying the largest modules and the sources of power consumption.

This chapter contains results published in ISCAS 2018[68] and consists in an initial design using the SAD as distortion metric. This chapter is organized as follows. Section 6.1 introduces a few related works that propose hardware designs for the HEVC FME. Section 6.2 presents our proposal of design strategy for FME, including a case study of the 8×8 HEVC FME. The architecture described in Section 6.2.1 was synthesized following the method in Section 6.2.2. Section 6.2.3 presents the synthesis results, while Section 6.3 concludes this chapter.

6.1 RELATED WORK

Guo et al. (2012) propose a configurable interpolation architecture able to realize the three HEVC filters. However, only one filter type (up, middle, or down[69]) can be used at a time. To reduce the energy consumption of the interpolation filters, Kalali and Hamzaoglu (2014) uses a Multiplierless Constant Multiplication (MCM) algorithm and explores common sub-expressions among source samples. Applying both techniques, the architecture from Kalali and Hamzaoglu (2014) was able to reduce in 48% the energy with respect to a baseline HEVC FME architecture. Both previously works present only interpolation filters, which require them

[67] See Figure 21.
[68] Seidel, Filho, et al. (2018).
[69] See Section 2.4. Figure 19 illustrates up, middle, and down filters.

to keep internally the interpolated samples or to write them in an external memory until they are used.

Sinangil et al. (2013) present several hardware design considerations for the HEVC ME. Although their designs include FME, no data is presented for such specific part. Vladimir Afonso et al. (2016) present a hardware design dedicated for the HEVC FME. Their design integrates interpolation and search, saving area and power. However, such design uses a filter structure with a dedicated datapath for each interpolated sample, resulting in redundant operations. Moreover, the authors assume an external source for the samples in integer positions and do not evaluate the impacts of such decision. Finally, most works that consider the whole FME adopt only a simple distortion metric that disregards rate constraints. As presented in Section 3.4.2, not using the rate term of Equation 2.6 increases the BD-Rate by 0.26%, on average.

6.2 FME HARDWARE DESIGN

A possible approach for FME consists in interpolating and performing BMA with candidates from all fractional positions surrounding $\mathbf{B}^{\text{ref-ime}}$. Such approach complies with the two main goals of our strategy: **coding** and **energy** efficiency. If S^{fme} contains all possible fractional candidates, BMA will find the optimal MV around $\mathbf{B}^{\text{ref-ime}}$. Considering the CTC (BOSSEN, 2012), this approach reduces BD-Rate by up to 0.37 % and 0.77 % for RA and LD profiles[70], respectively. Such approach for FME is also hardware-friendly, since it leads to a regular dataflow (XU; YIN, et al., 2015; HU et al., 2017). This regularity leaves more room for data-level parallelism which, by its turn, is able to reduce the required number of cycles. A high degree of parallelism reduces delay at the expense of area and power overheads. Nevertheless, if properly exploited, such time reduction is well worthy the overheads, resulting in better energy efficiency (RABAEY, 2009).

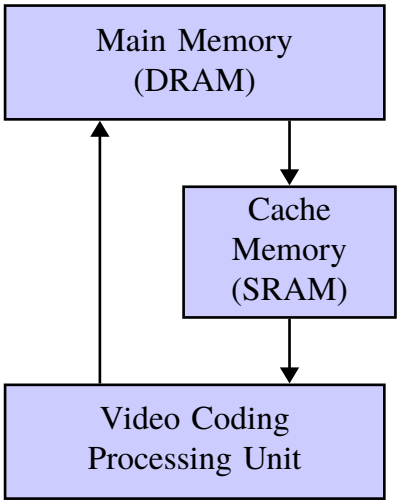
Using the interpolated samples as soon as they are generated (AFONSO, Vladimir et al., 2016) is a clever way to reduce power by getting rid of intermediary buffers. Thus, it is necessary that interpolation and BMA are both integrated in the same design (hardware). This restriction does not render MC unable to share the same interpolation, but it just requires a path from the interpolation output to the top level output. Another advantage of an integrated design is the possibility of implementing both steps in a pipelined fashion.

As Kalali and Hamzaoglu (2014) propose, no redundant operations should be performed. Also, the architecture must use MCM, once interpolation filter coefficients are constant. Apart these arithmetic optimizations, to achieve energy efficiency the design must be memory-aware, otherwise memory accesses may exceed 90% of total ME energy (ZATT et al., 2011). Most video encoders use a memory hierarchy composed of an external Dynamic Random Access Memory (DRAM) (to store the video frames) and internal SRAMs (usually to store the

[70] Using the SAD as distortion metric. This evaluation followed the same method presented in Section 3.2, but RA and LD configurations where slightly modified: `--HadamardME` was disabled so as to use the SAD instead of the SATD during FME.

search area) (LIU; LEE, 2008; LOPES; SILVA; AGOSTINI, 2012; AMARAL et al., 2014), depicted by Figure 57.

Figure 57 – Memory hierarchy for a video coding system.



Source: adapted from Amaral et al. (2014).

To avoid memory power overheads, it is necessary to keep the required data internally and use it as soon as possible. Moreover, some data require matrix transposition, which can be very demanding of external sources and thus must be done internally using TBs (S. LIVRAMENTO et al., 2012; SEIDEL; BRÄSCHER; GÜNTZEL; AGOSTINI, 2016), such as the ones from Figure 45. Even using memory hierarchies, data buses and on-chip buffers can be responsible for almost 70% of total power (YANG; WOLF; VIJAYKRISHNAN, 2005). Therefore, another power reduction technique consists in reducing interconnect distances. It can be induced by designing buffers where data moves in layers, thus avoiding long wires at the expense of more switching activity.

Finally, alongside the common outputs (best cost and related MV), our strategy includes issuing also the $\mathbf{B}^{\text{res-fme}}$. Hence, if the FME candidate is the one chosen by mode decision, the transform can be applied directly on the outputs, instead of interpolating again the $\mathbf{B}^{\text{res-fme}}$ (thus avoiding unnecessary MC).

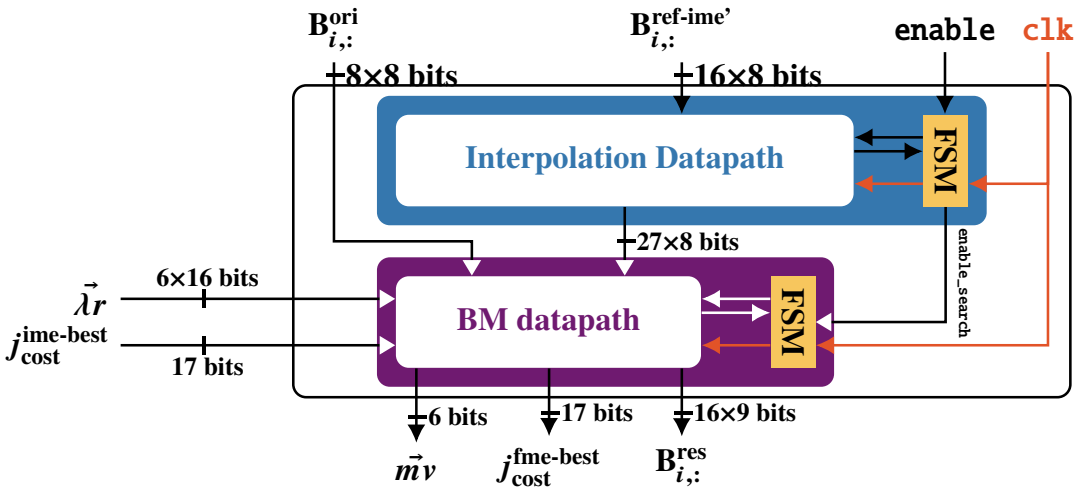
6.2.1 Case study: The HEVC 8x8 FME

While video coding standards define only the decoder side (SINANGIL et al., 2013), ME is part of the encoder. However, the interpolation is standardized, since it is also part of the decoder. Thus, to implement a hardware that implements the interpolation module, we need to adopt some standard for the interpolation. In this case we will use the HEVC FME to evaluate our strategy. To compare with state-of-the-art designs, our case study assumes 8×8 blocks. Such a block size is amongst the most chosen ones during HM encoding (AFONSO, Vladimir et al., 2016, p. 111). Moreover, the FME of larger blocks may be computed by using more than one

instance of the 8×8 FME module (AFONSO, Vladimir et al., 2016, p. 114). In such a case, to be more energy-efficient, the architecture should avoid storing twice the same samples.

The hardware architecture resulting from our strategy (Figure 58) interpolates and searches all fractional samples around $\mathbf{B}^{\text{ref-ime}}$. Such architecture is highly parallel and completes interpolation and search in 51 cycles.

Figure 58 – Main modules of our FME hardware architecture. The interpolation and Block Matching (BM) datapaths are presented in Figures 59 and 61, respectively.



Source: the author.

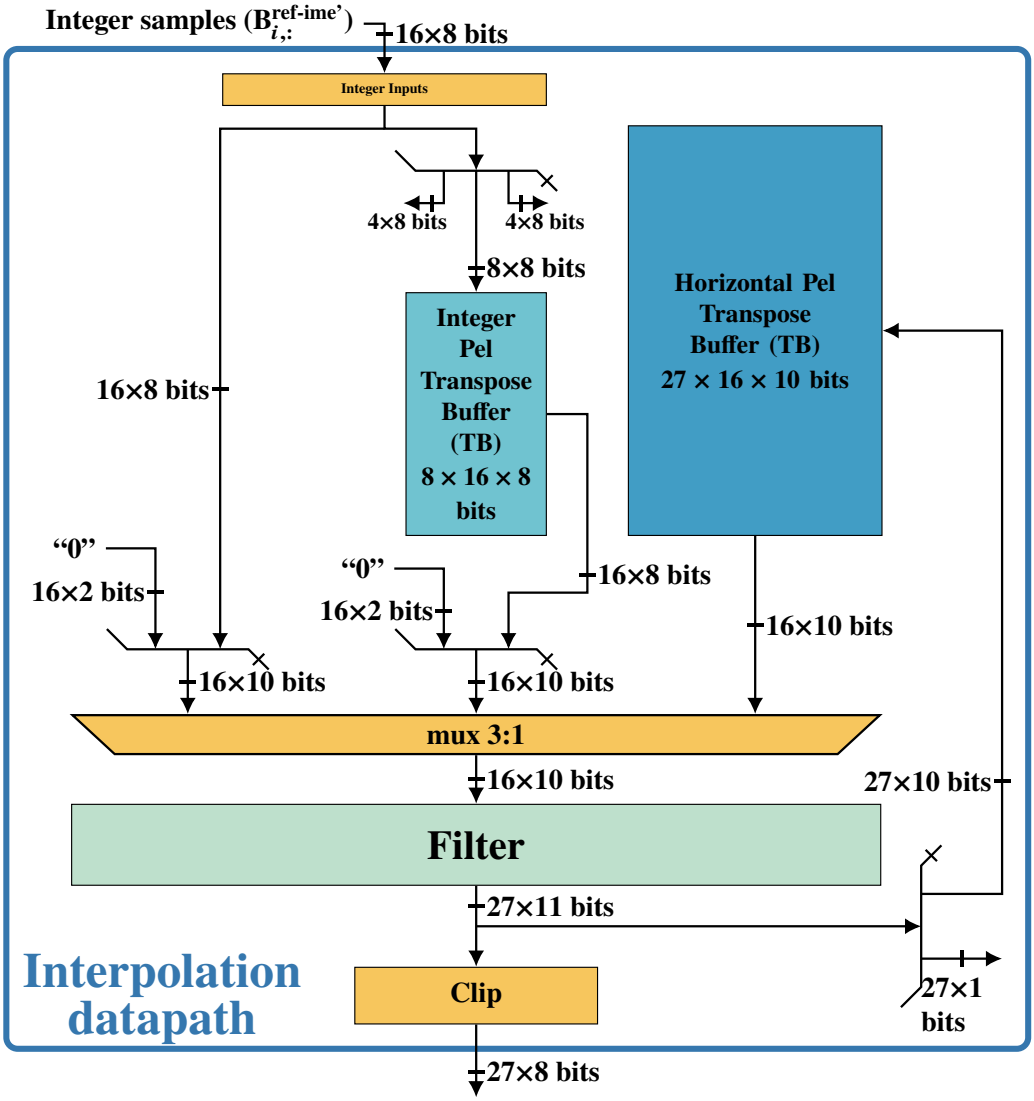
Although Vladimir Afonso et al. (2016) adopt the strategy of combining the interpolation and search in the same architecture, in their case integer samples are assumed as coming from an external source. However, to interpolate vertical samples the architecture must receive the transposed integer samples. Unfortunately, the complexity overhead from transposition was not considered and neither was the energy to access the external samples. Transposition is also an issue with regards to $\mathbf{B}^{\text{ori}}$. But in this case, the problem is still more stringent, since $\mathbf{B}^{\text{ori}}$must be read from an external source four times! In this sense, our design is **memory-aware** because each sample is read from an external memory only once, while transpositions are performed internally. Therefore, the interpolation datapath (Figure 59) includes the Filter (Figure 60a), clipping and two TBs, for integer and horizontal samples.

To avoid redundant filter operations, we exploited coefficient sharing. Since there are eight distinct filter coefficients (see Table 5 and Figure 19), their multiplications can be described by a vector function $\overrightarrow{ss}(x) : \mathbb{Z} \rightarrow \mathbb{Z}^8$, defined as:

$$\overrightarrow{ss}(x) = [-x; 4\times x; -10\times x; -5\times x; -11\times x; 40\times x; 58\times x; 17\times x] \tag{6.1}$$

Given $\overrightarrow{x} = \mathbf{B}^{\text{ref-ime'}}_{i,:}$, i.e $\overrightarrow{x}$ is the i^{th} line of $\mathbf{B}^{\text{ref-ime'}}$, the application of Up, Middle and Down filters (Figure 19) over $\overrightarrow{x}$ results in three vectors, namely $\overrightarrow{u}$, $\overrightarrow{m}$ and $\overrightarrow{d}$. These vectors have 9 interpolated samples each, and their values can be obtained by using $\overrightarrow{ss}(x)$, as shown in Equations 6.2 to 6.4, where $n \in [-1,8]$.

Figure 59 – Datapath of the FME Interpolation module



Source: Seidel, Filho, et al. (2018).

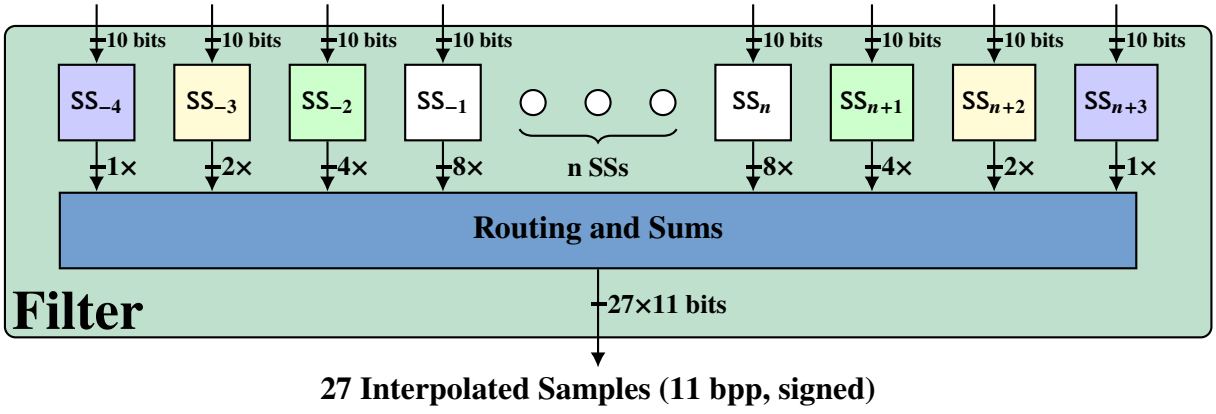
$$\vec{u}_n = \left(\vec{ss}\left(\vec{x}_{n-3}\right)_0 + \vec{ss}\left(\vec{x}_{n-2}\right)_1 + \vec{ss}\left(\vec{x}_{n-1}\right)_2 + \vec{ss}\left(\vec{x}_n\right)_6 + \vec{ss}\left(\vec{x}_{n+1}\right)_7 + \vec{ss}\left(\vec{x}_{n+2}\right)_3 + \vec{x}_{n+3}\right) \gg 6 \tag{6.2}$$

$$\vec{m}_n = \left(\vec{ss}\left(\vec{x}_{n-3}\right)_0 + \vec{ss}\left(\vec{x}_{n-2}\right)_1 + \vec{ss}\left(\vec{x}_{n-1}\right)_4 + \vec{ss}\left(\vec{x}_n\right)_5 + \vec{ss}\left(\vec{x}_{n+1}\right)_5 + \vec{ss}\left(\vec{x}_{n+2}\right)_4 + \vec{ss}\left(\vec{x}_{n+3}\right)_1 + \vec{ss}\left(\vec{x}_{n+4}\right)_0\right) \gg 6 \tag{6.3}$$

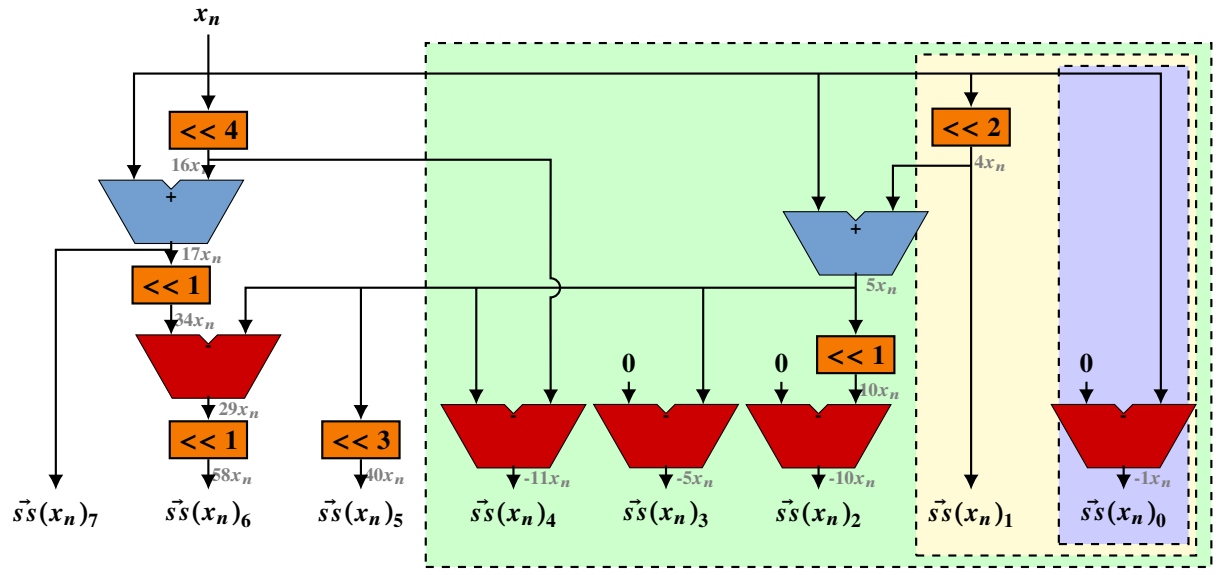
$$\vec{d}_n = \left(\vec{x}_{n-2} + \vec{ss}\left(\vec{x}_{n-1}\right)_3 + \vec{ss}\left(\vec{x}_n\right)_7 + \vec{ss}\left(\vec{x}_{n+1}\right)_6 + \vec{ss}\left(\vec{x}_{n+2}\right)_2 + \vec{ss}\left(\vec{x}_{n+3}\right)_1 + \vec{ss}\left(\vec{x}_{n+4}\right)_0\right) \gg 6 \tag{6.4}$$

Similarly to Kalali and Hamzaoglu (2014), we used Spiral tool (YEVGEN VORONENKO, 2017) to generate the filter datapaths corresponding to Equation 6.1. The resulting

Figure 60 – RTL datapath for the HEVC 8×8 FME filters. Some bitwidths are ommited for simplification. (a) Filter, which uses $n+8$ SS modules SS_i for $i \in [-4, n+3]$, where n is one of the dimensions of the block. Index i represents the coordinates surrounding Figure 19. Although the SS modules are represented generically, the output is considering $n = 8$, hence the 27 samples (which are $(n+1)\times 3$), up, down, and middle). (b) Sums and Shifts (SS) modules (see Equation 6.1). Notice the color correspondence between (a) and (b). For instance, each outermost SS module (SS_{-4} and SS_{n+3}) is composed of only the rightmost subtractor from (b). On the other hand, each of the 10 innermost SS modules (SS_{-1} up to SS_n) contains all operations from (b).



(a)



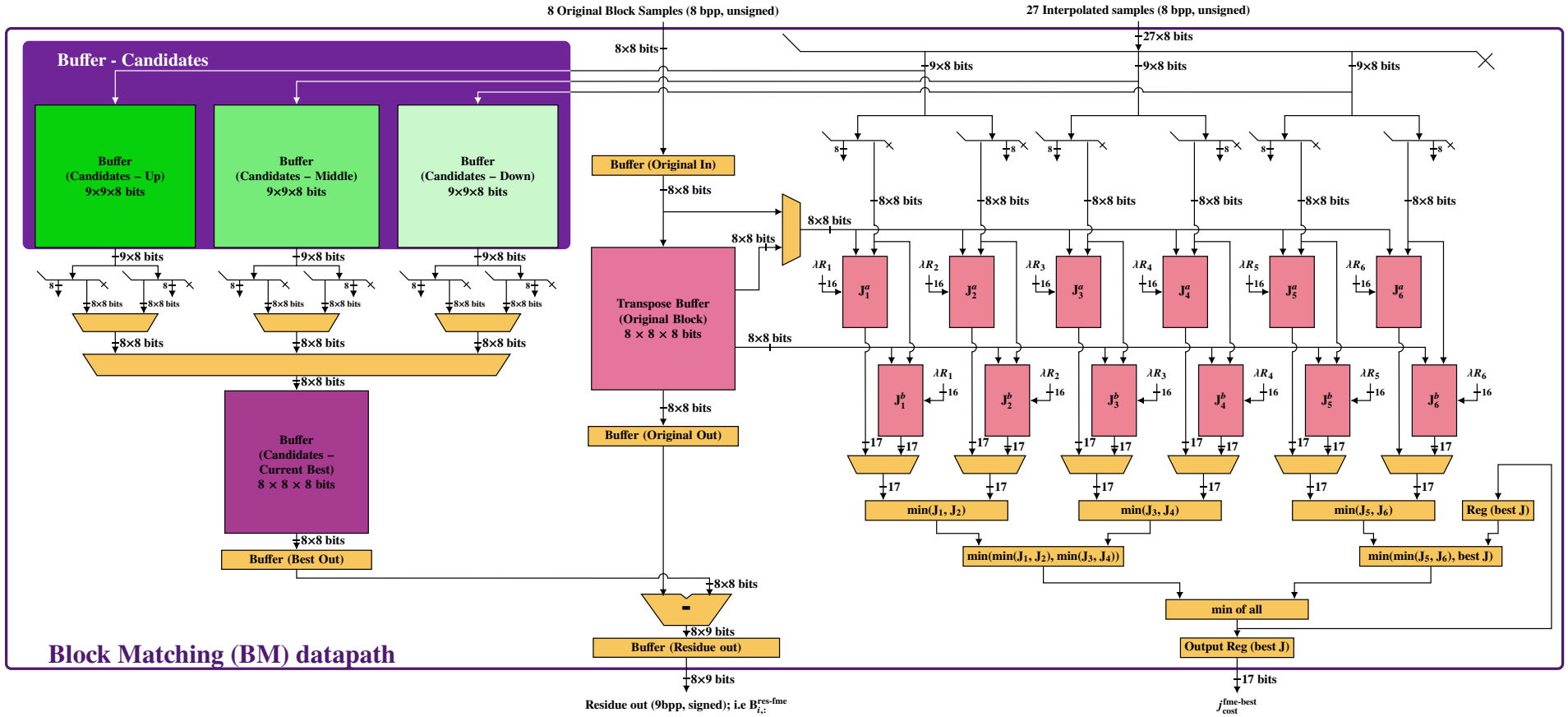
(b)

Source: Adapted from Seidel, Filho, et al. (2018).

datapath (Figure 60b) was named Sums and Shifts (SS). To increase parallelism, each of the 16 samples in $\overrightarrow{x}$ has its own dedicated SS datapath (Figure 60a). As the number of dependencies of samples near the borders of $\mathbf{B}^{\text{ref-ime'}}$ reduces, we adopted simplified versions of SS accordingly.

As shown in Figure 58, both interpolation and BMA are integrated in a single design. One cycle after finishing the interpolation of every row from the candidate samples, their values are available to the Block Matching (BM) module, presented in Figure 61.

Figure 61 – Block Matching (BM) datapath for FME.



Source: Seidel, Filho, et al. (2018).

Each J module performs the SAD (Equation 2.5), row-by-row, between $\mathbf{B}^{\text{ori}}$and each interpolated $\mathbf{B}^{\text{can}}$, which is added to the appropriate $\lambda\times$rate to obtain j_{cost}. The weighted rates must come from an external source, which are loaded to each J accumulator without overheads after 12 extra inputs. To be able to provide the $\mathbf{B}^{\text{res}}$, intermediary $\mathbf{B}^{\text{can}}$ must be kept in Candidate Buffers until their evaluation using j_{cost} finishes. After the decision, only the current best candidate is loaded line-by-line into another buffer, while the Candidate Buffers are overwritten with new candidates. Also, the current ${j_{\text{cost}}}^{\text{best}}$ is kept in a register to be compared to the remaining $\mathbf{B}^{\text{can}}$. All buffer controls are simplified using the decision tree of minimum j_{cost} values.

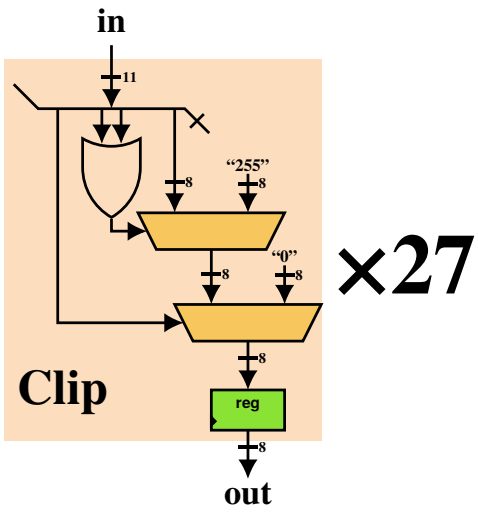
In the first four cycles, the four lines above $\mathbf{B}^{\text{ref-ime}}$ are loaded. During each cycle, there are two parallel operations:

1. The eight central samples are selected as inputs to Integer Pel TB;

2. One line of 27 interpolated samples is generated, while the 10 Least Significant Bits (LSBs) of each sample are made available as input to Horizontal Pel TB.

The next eight cycles will keep almost the same operation as before, but now the registers in the Clip module are also enabled, and the horizontal interpolated samples (see Figure 18a) are available to the BM module. Figure 62 shows the datapath of the Clip module, used to ensure that all interpolated samples are in the range $[0, 2^b - 1]$ for $b = 8$, i.e., it implements Equation 2.22 as it is used in Equations 2.23-2.25.

Figure 62 – Clip module datapath



Source: the author.

The remaining four cycles needed to load the entire area surrounding the $\mathbf{B}^{\text{ref-ime}}$ work just like the first four. Now that both Integer Pel TB and Horizontal Pel TB are fully loaded, FOVS may be generated (see Figure 18b). Therefore, the eight cycles that follow are required to interpolate FOVS from the transposed integer samples, which are used in BM module. The

final 27 cycles are used to interpolate the SOVS (see Figure 18c), also used during BMA. This concludes a FME, and thus our architecture requires 51 cycles to load and interpolate all samples.

During the first four cycles after the FME architecture is enabled the search is kept inactive, since there are no interpolated samples yet that are part of a candidate block. After these initial cycles, the $\mathbf{B}^{\text{ori}}$must be available as input, one line each cycle, for the next eight cycles. Also, during FME initialization, the best J cost from IME may be loaded in the architecture, allowing to be compared against the costs of each $\mathbf{B}^{\text{can-fme}}$. Thus, after the central horizontal samples are loaded into the Clip registers, they will be available as inputs to the J *cost* computations as BM module will execute the BMA.

Notice in Figure 61 that there are two rows of J modules; During the search in Horizontal samples, only the first row (the one above the others) is enabled. Each cycle a new line from $\mathbf{B}^{\text{ori}}$ is read into the Original In Buffer, it is fed both to Original TB and to the first row of J modules. When the SOVS are received, both rows of J modules are used. The second row receives the samples from the Original TB from the second layer of registers in the TB, i.e., delayed one cycle. This counteracts for the displacement in the position of the generated $\mathbf{B}^{\text{can}}$, which permits such high level of parallelism. When a row of J finishes the *cost* computations, the smallest j_{cost} is selected in a tree of minimum values. One cycle latter, the second row will finish, and their results will be already compared with the best one from the first J row. While selecting the best J, the choices are used to select from which Candidate Buffer the $\mathbf{B}^{\text{can}}$ was selected. The Current Best Candidate Buffer will only be loaded if one current candidate was smaller than the best J. In this case, the next eight cycles will load the samples in such buffer accordingly.

During the 51 interpolation cycles, a total of 48 candidate blocks are generated. Because we adopt an scheme of keeping only the required $\mathbf{B}^{\text{can}}$, the maximum of 13 $\mathbf{B}^{\text{can}}$ are stored at any given time (12 being compared plus the best one so far). Each candidate buffer maintains the $\mathbf{B}^{\text{can}}$ generated by one filter type: up, middle, and down.

6.2.2 Evaluation method

The hardware design case study was synthesized with DC® (SYNOPSYS, 2015) in Topographical mode to estimate routing parasitics and thus, obtain realistic timing, area and power estimates. All syntheses used a 45 nm standard cell library from TSMC (TSMC, 2011). Also, input and output delays were conservatively limited to 60% of the clock period. The maximum primary input capacitance was set to 10 times a 2-input AND gate whereas the maximum primary output capacitance was set to 30 times a 2-input AND gate. Finally, to get detailed area and power reports, we restricted DC® to keep the RTL hierarchy during syntheses[71].

Such evaluation method is only a part of the design and synthesis method illustrated in Figure 54. We used a reference software for the FME that was implemented based on FME

[71] Forcing the synthesis tool to keep RTL hierarchy resulted in 2.5% more power and 10.7% larger area, on average. Also, when hierarchy was not preserved, the tool was unable to meet the required constraints for 2160p@120fps.

descriptions from the literature, instead of using the HM[72]. Also, as we did not use HM in this test, this architecture was not simulated with realistic input data. Therefore, we only generated the power reports directly from the synthesis tool, after verification.

6.2.3 Results and comparison with related works

We performed four distinct syntheses, each one aiming a pair of resolution and fps, as shown in Table 17. Given the FME of a video with resolution of $w \times h$ and f frames per second, the required throughput in $\mathbf{B}^{\text{ori}}_{m \times n}$/second is $(w \times h \times f)/(m \times n)$. The presented architecture finishes an FME each 51 cycles which, multiplied by the throughput, gives the number of cycles per second (its inverse being the target period). Thus, the clock period was set accordingly for each synthesis.

Table 17 – Synthesis results for TSMC 45 nm @ 0.9 V

Target	**1080@30fps**	**2160@30fps**	**2160@60fps**	**2160@120fps***
Period (ns)	20	5	2.5	1.25
Area (μm^2)	93738.3	93826.5	94439.1	100733.4
Dynamic P. (mW)	3.05	10.78	21.21	44.04
Static P. (mW)	0.91	0.91	0.92	1.04
Total P. (mW)	3.96	11.69	22.13	45.07

* 2160@120fps is equivalent in terms of throughput to 4320@30fps. Source: The author.

The largest increase area, from one target period to the next smaller one, was only 6.6%, observed when comparing 2160@60fps and 2160@120fps cases. On the other hand, total power increases about 2× when doubling the frequency, which is expected since dynamic power increases with frequency. The share of dynamic power due to nets also increases with frequency: from 18% to up to 23%. This shows the importance of reducing wire-length, even at 45 nm. To provide some insightful evaluation, Figure 63 shows area and power breakdowns for the lowest and highest target throughputs.

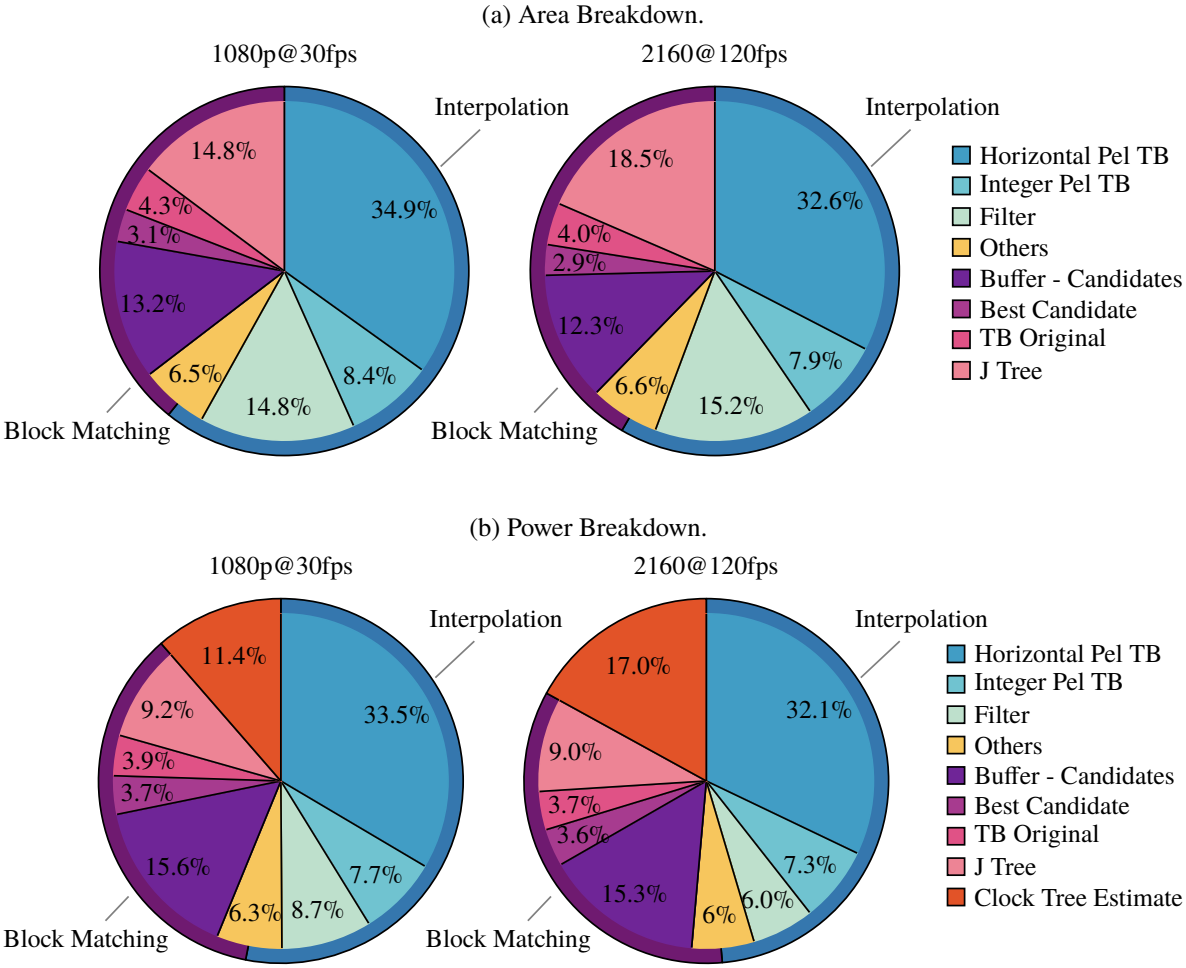
The increase in area share of Filter and J Tree comes from the critical path optimization performed by the synthesis tool to achieve the higher throughput[73]. In both area and power Horizontal Pel TB is the dominant part of the design. Considering Filter and TB shares, the former is larger and demands more power, even considering the large amount of power consumed by the clock tree (up to 17% @ 2160@120fps).

Among the related works, the architecture and syntheses presented by Vladimir Afonso et al. (2016) are the most similar to ours. They used a different synthesis tool and Nangate standard cell library, but the technology was also 45 nm (0.9 V). Since that architecture requires the same number of cycles than ours, a direct comparison of energy consumption can be done

[72] As a consequence, although the verification passed, we latter found out that the description of FME was underestimating the precision used to store horizontal samples, as it will be explained on Section 7.1. Nevertheless, to our best knowledge, our Horizontal Pel TB stores horizontal samples with the same precision as the most similar related work by Vladimir Afonso et al. (2016).

[73] As mentioned in p. 154, faster gates are inserted, which are also larger.

Figure 63 – Area and Power Breakdown. The outer ring represents the share of Interpolation and Block Matching (BM). For both area and power, exactly one third of "Buffer - Candidates" share corresponds to each internal buffer: Up, Middle and Down.



Source: the author.

by evaluating total power. For 1080@30fps our design consumes 20,1% less power than the one from Vladimir Afonso et al. (2016), while performing more operations. Moreover, their results do not include the power spent by external pixel sample source. On the other hand, for 2160@60fps their design requires 28% less power than ours. Particularly, when raising throughput from 1080@30fps to 2160@60fps (8× the frequency), (AFONSO, Vladimir et al., 2016) reported an increase of only 3.19× without justifying such unexpected behavior. For a fair power comparison with the architecture proposed by Vladimir Afonso et al. (2016), we disregarded the power share of the blocks that are exclusive to our design. By doing so, our design consumes 6.4% and 46.5% less power than theirs for 2160@60fps and 1080@30fps, respectively. Finally, it is worth noting that Vladimir Afonso et al. (2016) reports lower power than those in (DINIZ, C. M. et al., 2013; PASTUSZAK; TROCHIMIUK, 2016; HE et al., 2015a; AFONSO, V. et al., 2015).

6.3 CHAPTER CLOSING REMARKS

In this chapter, the proposed design strategy for FME was evaluated by means of a case study consisting of the HEVC 8×8 FME. The hardware architecture of such case study is more energy-efficient than state-of-the-art similar ones. This is because the proposed strategy leads to a BMA that considers all pixel positions to maximize parallelism and adopts a fast RDO during search to guarantee that the optimal fractional motion vector is found. Also, the synthesized design has no redundant filter operations, avoids extra MC, is memory-aware and the internal buffers were designed to minimize wire-length. As result, both coding and energy efficiency are improved.

The obtained power and area breakdowns show that buffers are responsible for up to 64% of both area and power. However, the largest one (the Horizontal Pel TB, with more than 30% of area and power) is intrinsic of all memory-aware parallel FME architectures, which is the case of our design. In addition, when enforcing shorter clock periods during synthesis, the largest amounts of area increase were observed in the sequential parts (like the Filters and J Trees), revealing that the critical paths are in those parts, and not in the large buffers.

7 IMPROVING THE FME HARDWARE DESIGN AND ADOPTING SATD

This chapter presents a hardware architecture for using SATD during FME. Therefore, the main contributions of this chapter are:

- A proposal of a complete FME hardware architecture that uses the SATD as distortion metric;
- The comparison of the overheads caused by the adoption of SATD instead of SAD during FME.

To our best knowledge, this is the first comparison between using SAD or SATD in the same hardware architecture. Also, instead of merely using the SAD-based FME architecture presented in Chapter 6, we first improve it. The improvements resulted from a re-evaluation of that architecture with an SATD implementation in mind. Thus, the new architecture includes new features, such as the computation of the rate term (from Equation 2.6), which is a novelty by itself, as most works do not include such part or do not provide details about it.

7.1 MODIFICATIONS TO OUR SAD-BASED FME ARCHITECTURE

While adapting our FME architecture for using SATD instead of SAD as distortion metric for the simplified RDO, we encountered some opportunities to improve further the design presented in Chapter 6. The modifications were:

M.1. Increase of the precision for storing horizontal pixels used to generate the SOVSs;

M.2. Design of a new hybrid TB architecture for reducing the switching activity without increasing too much the wire-length when applicable;

M.3. Reduction of the number of arithmetic operations in the filter datapaths;

M.4. Design of a new module that computes the rate term of the simplified RDO (Equation 2.6);

M.5. Removal of the large candidate buffers used for computing the residues;

Modification M.1 was done to ensure the architecture produces the same results as HM. We tested the architecture of Chapter 6 with an FME model following the descriptions provided on related works from the literature[74]. However, after integrating the testbench with HM[75], we found out that such a model was not fully compatible with the interpolations as described in the standard. The difference was in the precision of horizontal samples used as inputs for generating

[74] See Note 72.

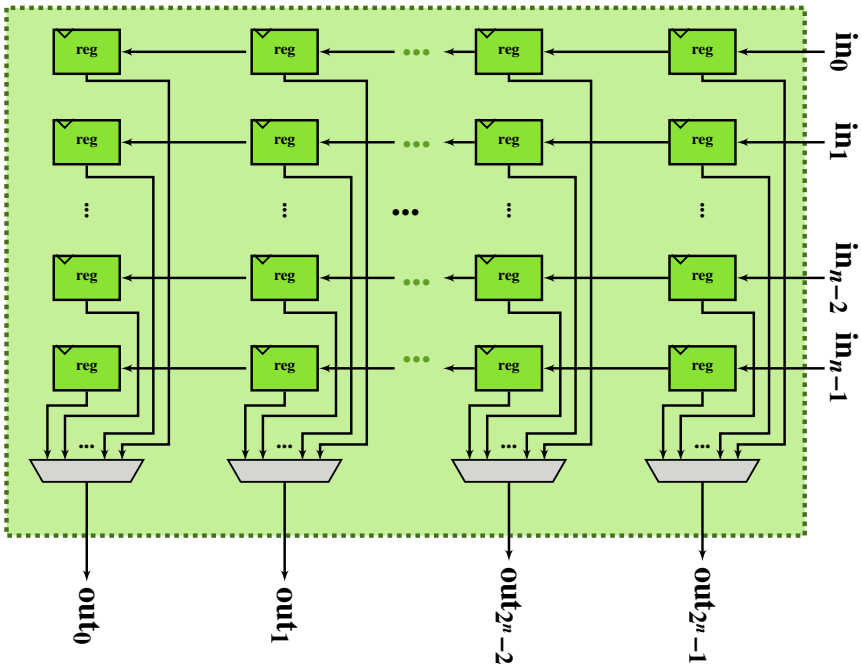
[75] Such integration was done by adopting the simulation model presented in Figure 55. This allowed us to guarantee that our architecture was simulated using real input data from HM.

the SOVS. HM is more precise than this initial model, i.e., it stores the horizontal samples with more bits.

In Figure 59, it is possible to notice that the 10 LSBs of each interpolated pixel serves as input to the Horizontal Pel TB. Also, those values are not clipped back to 8-bit precision. However, the standard interpolation considers that these values must be saved before the "»6" operation from Equations 2.23-2.25. Therefore, instead of storing each interpolated pixel with 10-bit, we need to store 16-bits per pixel to be compatible with HM. This represents a 60% increase in the number of bits in the Horizontal Pel TB, having a significant impact in the already area demanding and power-hungry TB (see Figure 63). Such modification also increases the number of bits in the filters. After all, they now have to support inputs with 16-bits instead of 10.

Given that TBs in our FME architecture are the main responsible for power consumption, they were selected to be optimized. Unlike the TBs used in the SATD, that must support concurrent writing and reading, the TBs in the FME operate either writing or reading data. Therefore, in the context of Modification M.2, we created a hybrid version of such TBs. Figure 64 shows its datapath.

Figure 64 – Hybrid TB design: compared to the TB design from Figure 45, the present one has reduced switching activity during reading cycles. The column to be read is obtained by providing its address to the output muxes.



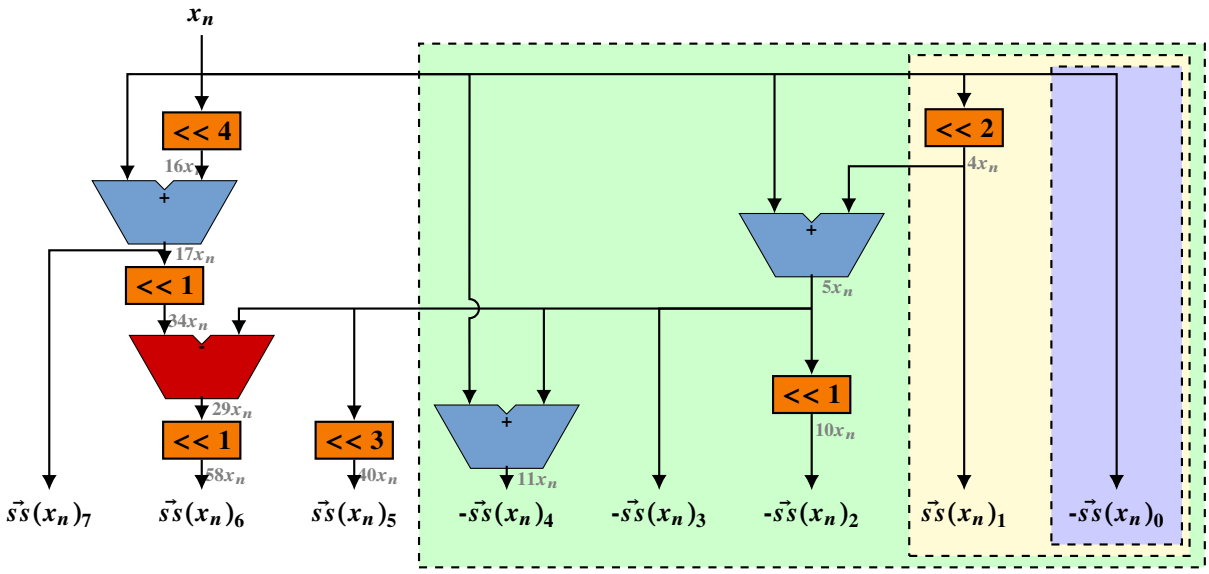
Source: the author.

Besides reducing the switching activity while still keeping relatively short wire-lengths, the new hybrid TB also allows a more effective use of clock-gating, since during reading states only a portion (column) of the buffers are active. Nevertheless, during writing states the hybrid TB works as the TB from Figure 45: the data enters in one layer of registers and is shifted to the next layer, cycle after cycle, until the buffer is filled. Thus, it achieves maximum switching

activity in the last cycle, when the last row of data is written (every row will shift to the next layer in such cycle). Although having higher switching activity during these cycles, this approach avoids long wire-lengths and the large capacitance that would rise if the inputs should be wired to every layer of registers.

Concerning Modification M.3, as the bitwidth of the filter operations increased by 60% because of Modification M.1, an increase in area and power is expected in its datapath. Thus, after carefully analyzing the filter datapaths generated by the Spiral tool (YEVGEN VORONENKO, 2017), we noticed that some signal inverting operations (i.e., "multiplications" by -1) were not really necessary and could be promptly merged to the sum trees (that are located within the Routing and Sums module in Figure 60a). Figure 65 shows the updated SS modules datapath. Compared to Figure 60b, which has seven arithmetic operations, the updated SS modules keep only four.

Figure 65 – Updated SS datapaths for HEVC FME.



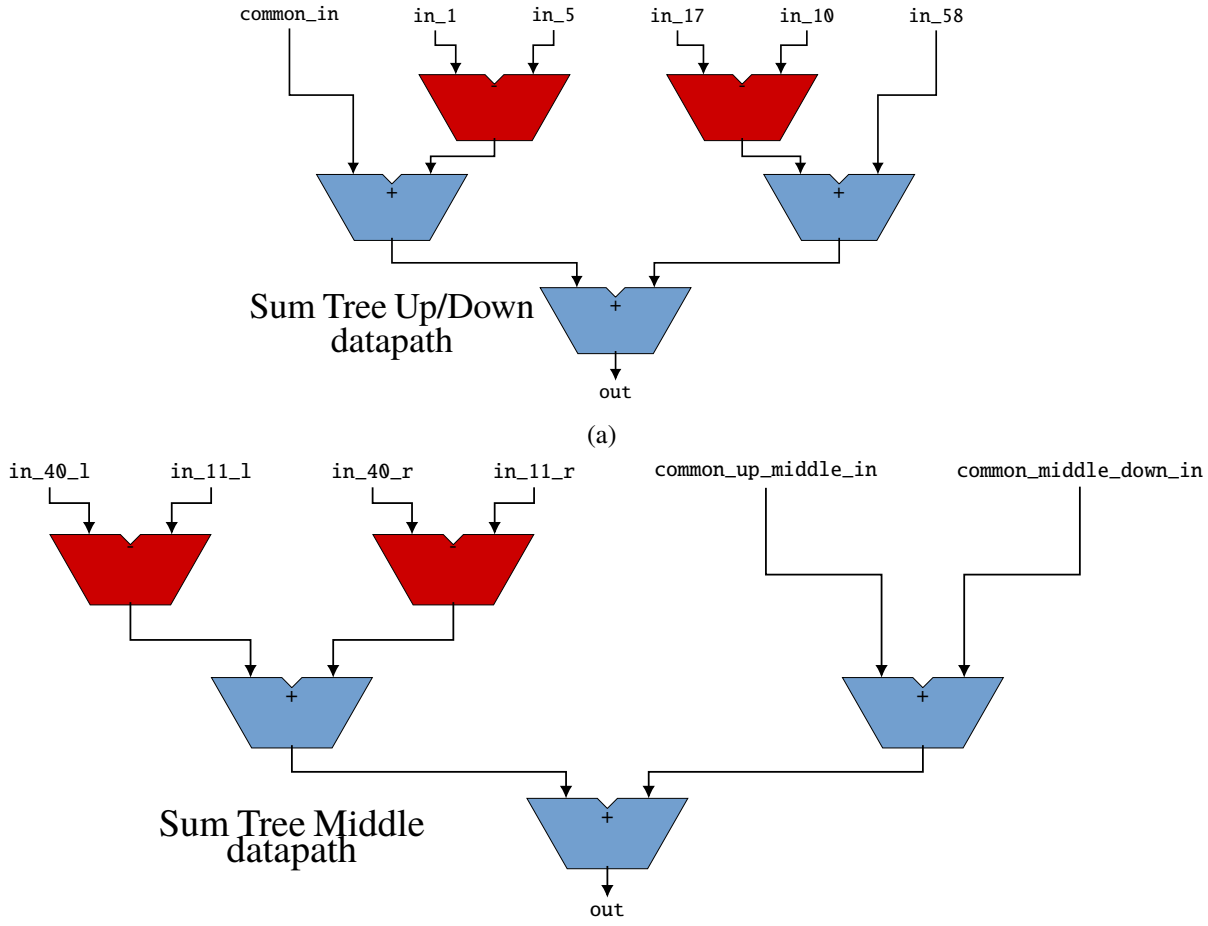
Source: the author.

With such modification, the outermost SS modules (SS_{-4} and SS_{n+3}) from Figure 60a are now just wires, since their only function was to change the signal of the input. Such operation was merged into the sum tree. The same was done for the other negative SS module outputs. Therefore, now all the SS outputs are positive, so their operation is the multiplication for positive constants with minimum cost. Therefore, Modification M.3 is a quite natural enhancement of the architecture, after perceiving such redundancies in the operations executed within the filters. In fact, as presented in Section 6.2, one of the premises for guaranteeing energy efficiency in the FME architecture is to avoid redundant operations.

There were even two shared operations that are inputs to up, middle, and down filters. One is shared among up and middle, the other among middle and down. Equations 6.2 and 6.3 share the operation $\overrightarrow{ss}\left(\overrightarrow{x}_{n-3}\right)_0 + \overrightarrow{ss}\left(\overrightarrow{x}_{n-2}\right)_1$, while Equations 6.3 and 6.4 share the operation $\overrightarrow{ss}\left(\overrightarrow{x}_{n+3}\right)_1 + \overrightarrow{ss}\left(\overrightarrow{x}_{n+4}\right)_0$. Figure 66 shows the two modified datapaths for the sum trees that

compensates for the absence of negative values in the SS modules.

Figure 66 – Updated filter datapaths for HEVC FME. The values shown in the input names are correspondent to the coefficients that multiplied such inputs in the SS modules. Such datapaths implement Equations 6.2-6.4 sums, while the SS modules implemented the constant multiplications defined in Equation 6.1.



Source: the author.

The datapath from Figure 66a is used for both up and down filters. On the other hand, the datapath in Figure 66b is used only for the middle filter. The mentioned shared operations are performed outside the sum trees and are the inputs marked as `common_*`.

Modification M.4 consists in the design of a new hardware for computing the rate term from Equation 2.6. Such a term consists of λ, which is a real value, multiplied by the rate estimate (see p. 70). To avoid using complex floating-point arithmetic and thus keep the hardware as simple as possible, we have chosen to represent λ using fixed-point. To find out the needed precision, we considered the maximum value of the rate estimate. The maximum range of each coordinate of an MV in HEVC is in the interval $[-2^{15}, 2^{15}-1]$ (WIEN, 2014b, p. 189; BROSS et al., 2014, p. 120), and so it can be represented by a pair of 16-bit signed integers. Given that the absolute value of -2^{15} cannot be represented in such kind of variable, we assumed the minimum value to be $-(2^{15}-1)$ for obtaining the rate estimate[76]. Assuming

[76] This would pose minimum error in the estimates, since it is very unlikely to have an actual MV pointing this far

maximum and minimum values for an MV, Equation 2.13 results in:

$$rate\left(\overrightarrow{v}\right) = rate(x,y) = 62 \quad ,\forall x,y \in \{-(2^{15}-1), 2^{15}-1\} \tag{7.1}$$

Therefore, the maximum rate estimate is 62. Rounding up to the nearest power of two, 64, we know that the six first fractional digits of λ could be shifted to become an integer value because $\lambda \times 64 = \lambda \ll 6$. As j_{cost} is accumulated with distortion as an integer value, if we keep the 6 first fractional digits of λ, we will keep the required precision to be equivalent to HM. Table 18 presents all the possible rate estimates, according to Equation 2.13, and their multiplication by λ, expressed in terms of sums and shifts.

Table 18 – Operations over λ using only sums and shifts. The largest value (64×) is not required, but is shown to establish a higher bound.

2× **to** 16×	18× **to** 32×	34× **to** 48×	50× **to** 64×
$2\lambda = \lambda \ll 1$	$18\lambda = 10\times\lambda + 8\times\lambda$	$34\lambda = 32\times\lambda + 2\times\lambda$	$50\lambda = 32\times\lambda + 18\times\lambda$
$4\lambda = \lambda \ll 2$	$20\lambda = 10\lambda \ll 1$	$36\lambda = 32\times\lambda + 4\times\lambda$	$52\lambda = 32\times\lambda + 20\times\lambda$
$6\lambda = 4\times\lambda + 2\times\lambda$	$22\lambda = 16\times\lambda + 6\times\lambda$	$38\lambda = 32\times\lambda + 6\times\lambda$	$54\lambda = 32\times\lambda + 22\times\lambda$
$8\lambda = \lambda \ll 3$	$24\lambda = 12\lambda \ll 1$	$40\lambda = 20\lambda \ll 1$	$56\lambda = 32\times\lambda + 24\times\lambda$
$10\lambda = 8\times\lambda + 2\times\lambda$	$26\lambda = 16\times\lambda + 10\times\lambda$	$42\lambda = 40\times\lambda + 2\times\lambda$	$58\lambda = 32\times\lambda + 26\times\lambda$
$12\lambda = 6\lambda \ll 1$	$28\lambda = 14\lambda \ll 1$	$44\lambda = 22\lambda \ll 1$	$60\lambda = 30\lambda \ll 1$
$14\lambda = 8\times\lambda + 6\times\lambda$	$30\lambda = 16\times\lambda + 14\times\lambda$	$46\lambda = 32\times\lambda + 14\times\lambda$	$62\lambda = 60\times\lambda + 2\times\lambda$
$16\lambda = \lambda \ll 4$	$32\lambda = \lambda \ll 5$	$48\lambda = 24\lambda \ll 1$	$64\lambda = \lambda \ll 6$

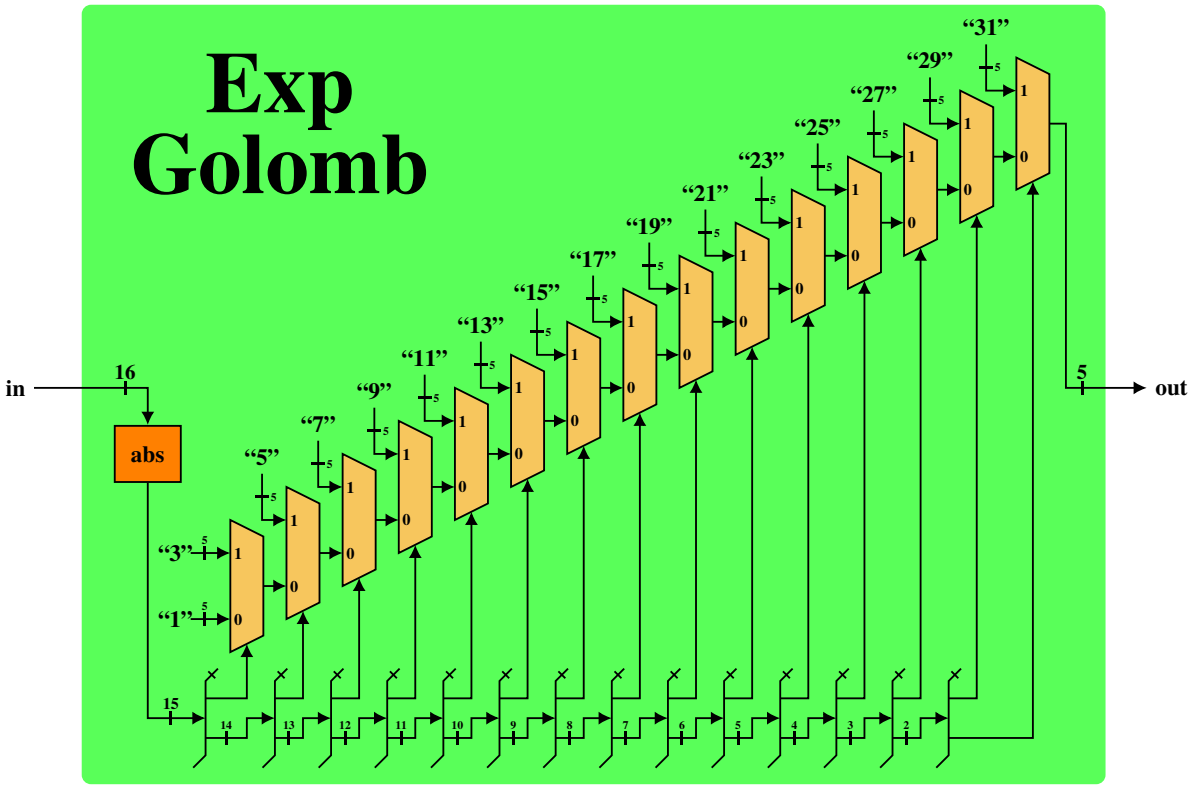
Therefore, we created a datapath that generates all those 31 possible rate terms in one cycle, in parallel with the operations from Table 18. The change in λ commonly happens every frame, given that it depends on the QP (Equation 2.10), and the QP may change according to the frame temporal id (Figure 22). Therefore, the circuit implementing the sums and shifts for computing all 32 multiplications of λ is expected to have a very low switching activity and, hence, a low power consumption. The outputs of such a module is a bus with $31\times(7+6) = 403$ bits, that will be used by six rate term modules.

Each rate term depends on the rate estimation of the candidate MV, as described in Equation 2.13. Such equation depends on Equation 2.14, which is computed for each element of the MV. Figure 67 shows our design of a datapath that corresponds to Equation 2.14. Basically, the outputs are related to the position of the most significant bit that is set, i.e., the leftmost bit that equals to one.

Finally, for completing Modification M.4, we implemented a datapath to obtain the six rate terms used for the simplified RDO in our FME architecture. Figure 68 presents the datapath that implements the integration of Equation 2.13 with the multiplication of the rate estimate in Equation 2.6. As mentioned, the MV coordinates are signaled with maximum precision (1/4). Hence, $p = 0$ in Equation 2.13 (after Equation 2.15 with the mentioned precision). Therefore, there is no need for the shift.

from the PMV. Also, the difference in the movement would be of 0.25 pixel in each dimension, since the MVs are represented with 1/4 precision, i.e., fixed-point representation with 2 bits for the fractional part (see p. 70).

Figure 67 – Exp Golomb datapath that implements Equation 2.14. Notice that such datapath is simple, as it only computes one absolute value (Figure 46c), and then a value is selected by each bit of the input. If the selection signal is 1, the mux outputs the value defined above. If the selection signal is 0, the mux outputs the value from the antecedent mux, with the only exception being the leftmost mux, which has a preset custom value for each of the inputs. The selection in such a mux is given by the LSB of the abs module output, while the Most Significant Bit (MSB) of abs output gives the selection of the rightmost mux. As the maximum value to be selected is 31, the output of Exp Golomb module requires only 5 bits.

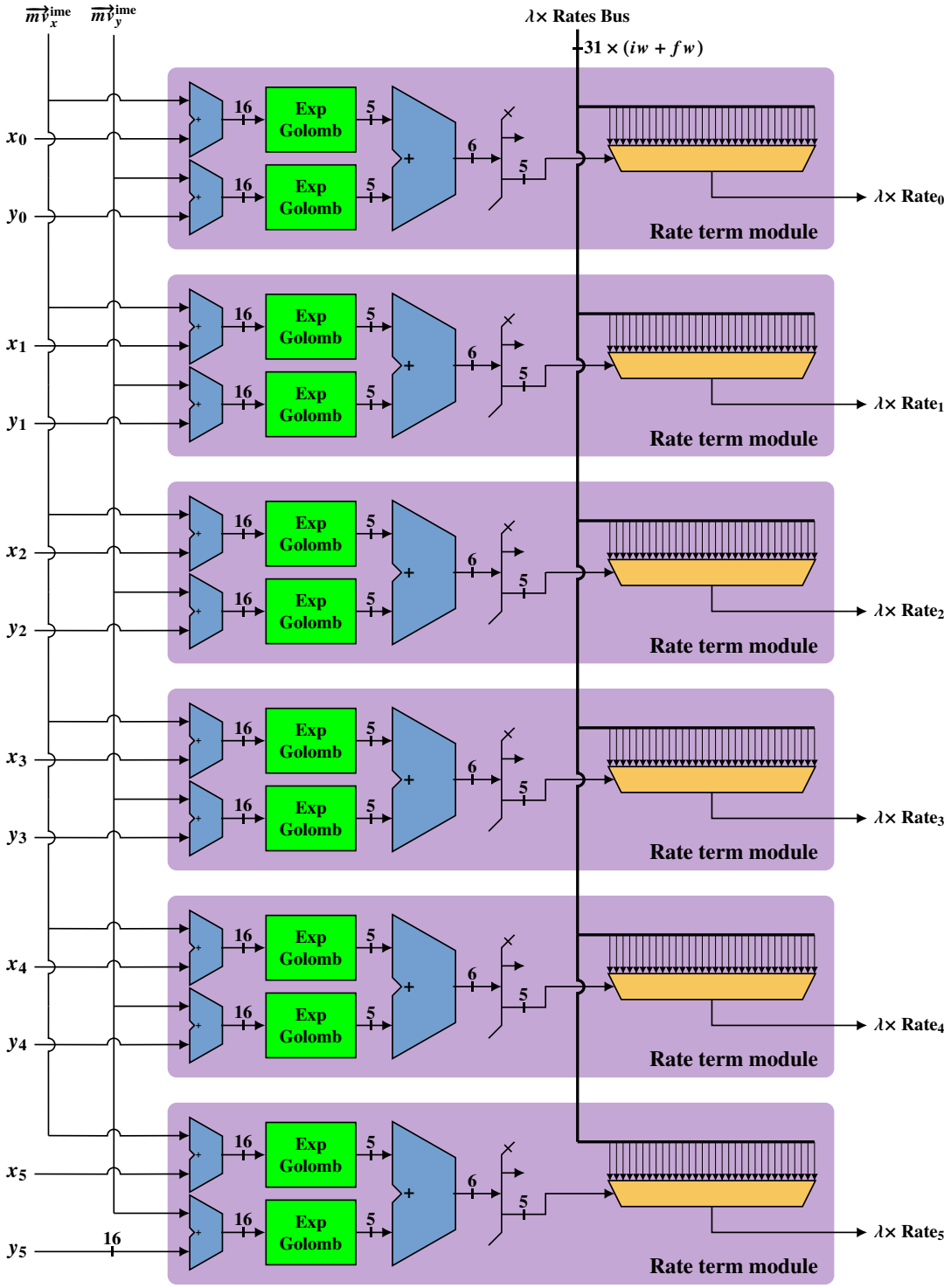


Source: the author.

The inputs for this datapath are the λ×Rates bus, the $\overrightarrow{mv}^{\text{ime}}$ (the MV obtained during IME), and the six pairs representing the displacements of each fractional position being computed, with respect to $\overrightarrow{mv}^{\text{ime}}$. There are only six coordinate pairs as inputs because the 12 modules that compute j_{cost} in our FME architecture have two distinct cycles for requesting the rate term. This is because the six bottom J modules initializes one cycle after the other six J modules (at the top). During the initial 12 candidate (horizontal and FOVS) evaluations, this datapath updates the fractional displacement coordinates only twice, as only the top J modules are being used. After that, for the SOVS candidates, this datapath updates the fractional displacements twice, in the first and second cycles of groups of 9 cycles. The $\overrightarrow{mv}^{\text{ime}}$ inputs may be updated at the beginning of each FME execution.

The last modification consists in removing the FME architecture datapath responsible for computing the residues. Hence, Modification M.5 saves both area and power by getting rid of the Buffer Candidates and Best Candidate Buffer (to the left of the TB original in Figure 61) that, together, were responsible for about 16% of area and power (Figure 63). Figure 69 shows

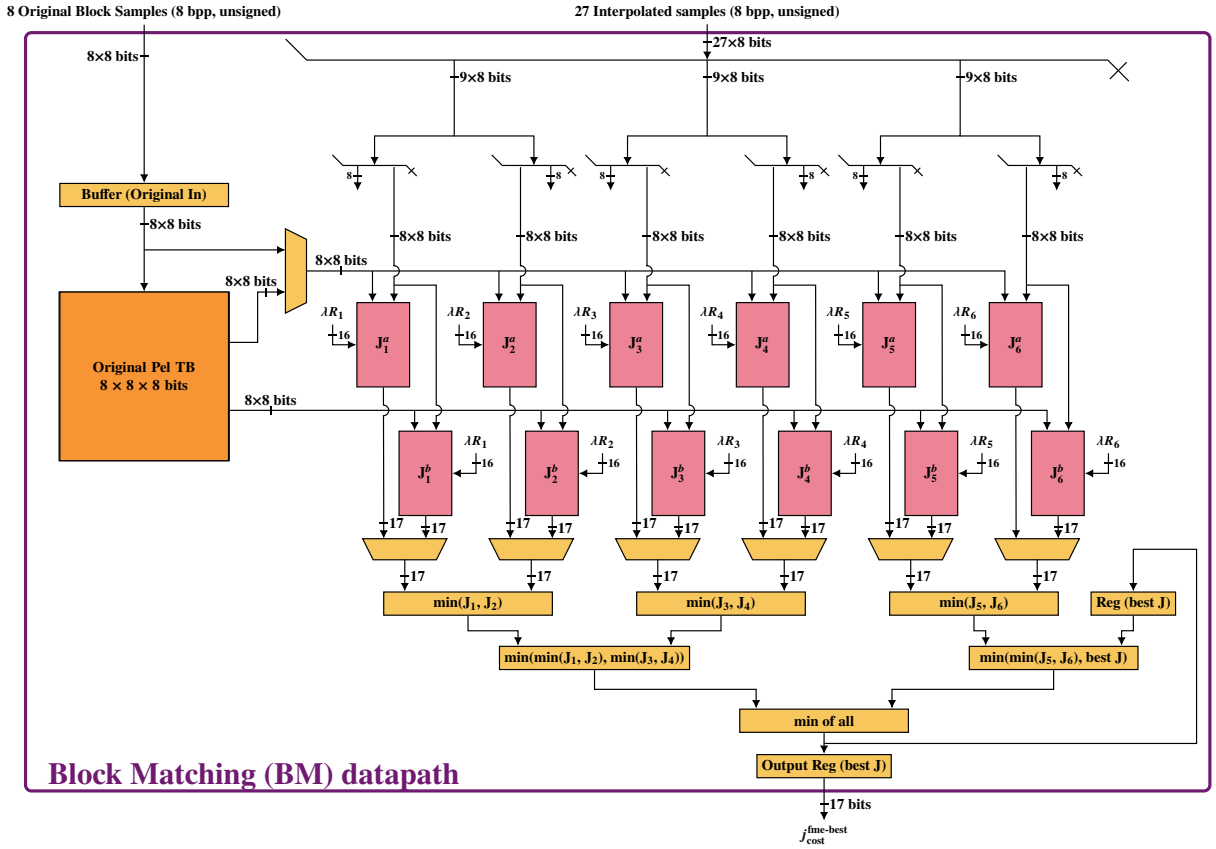
Figure 68 – Rate term (λR) datapath. As the sum of the outputs of two Exp Golomb modules is always even, we discarded the LSB before selecting the correct multiplication from the $\lambda\times$Rates bus. The part to the left of the $\lambda\times$Rates, within each rate term module, computes the rate estimate (Equation 2.13) of each MV. The multiplication of such an estimate by λ is done by selecting the precomputed value from the $\lambda\times$Rates bus. Therefore, no actual generic multiplication operator is used.



Source: the author.

the new BM module datapath.

Figure 69 – New block matching datapath for FME, after Modification M.5. By removing the candidate buffers, the architecture loses its ability to compute the residues.



Source: the author.

Of course, there will be an overhead of interpolating samples again if a fractional position is chosen. The overhead would be, however, much lower than our interpolation datapath share, because more simple interpolation architecture could be used to generate only the required pixels from the candidate block. Moreover, by our experiments, most of the times, when the rate term is considered, the integer position is chosen. This can be observed, for instance, in Figures 25a and 25b. Also, it is possible to observe that the central concentration (i.e., at the $\overrightarrow{mv}^{\text{ime}}$) is higher for a higher QP. This can be explained by the rate term: given that a higher QP results in larger λ and, hence, a higher weight for the rate term. The overhead in energy can be trade off by using a larger area and a ping-pong buffer to keep the integer and horizontal samples (that have a higher probability to be chosen), and only interpolating the vertical ones (FOVS and SOVS)[77].

Finally, without this modification, the adoption of SATD (presented in the sequel), would have required even larger Candidate buffers (near twice the size), because it takes twice the number of cycles to decide which one would be the best for each. Thus, considering the SATD, the benefit of not keeping all candidates (for possibly obtaining the residue) is even more

[77] Such an alternative method is being currently exploited by Monteiro (2019) in his work entitled "An energy-efficient FME architecture using SAD combined with a novel approach for residue calculation" (original title in Portuguese, our translation).

significant than for the SAD case.

7.2 ADOPTING THE SATD TO REPLACE THE SAD

Within each of the 12 J modules from Figure 69 lies one SAD module that is replaced by one SATD module. As the interpolation architecture can generate the candidates row-by-row (horizontals and FOVS) and column-by-column (SOVS), we used the TB-based SATD architecture, shown in Figure 47a[78], whereas Figure 45 presents the TB organization.

However, some small changes to the SATD architecture had to be made made, as now the hardware modules that are connected to its inputs and outputs are known. For instance, the candidate inputs are connected to the Interpolation datapath (Figure 59), that has a register barrier right before its outputs (in Clip module). Once there is no logic after the register barrier, there is no point in having another register barrier at the inputs of the SATD. Also, the interpolation datapath generates horizontal candidates (Figure 18a) during 2^n cycles (for 2^n sized blocks), but then it generates four extra lines of horizontal interpolations that are needed for obtaining the SOVS. This breaks the pipeline of the SATD architecture, which expects new data to fill the TB while the already contained data is being read for the second 1D transform. Thus, we modified the architecture control for waiting during these four cycles before continuing the current SATD computation.

Despite the large number of cycles for computing the SATD, our SATD-based FME architecture still finishes one FME every 51 cycles. However, there is a larger delay between the beginning of an FME and its finish. While for the SAD such delay was of 57 cycles, it is 62 cycles for the SATD. These extra cycles account for emptying the TBs from the SATDs, i.e., using the remaining data, and then deciding the best j_{cost}. Also, during each FME execution, half of the SATD modules stay unused during horizontal and FOVS candidate evaluations. This is the same as in the SAD-based FME architecture.

7.3 EVALUATION METHOD

We adopted the method presented in Section 6.2.2, with a few small changes. Instead of using a custom reference software to simulate the FME, we used the HM with the modifications presented and evaluated in Chapter 3, that evaluates all the 48 candidate blocks for FME. Also, we included an extra target frequency, which is able to achieve a throughput of 1080@60fps. Such cadence, 60 fps, is the one from the sequence "BQTerrace" from the CTC (BOSSEN, 2012). This was done to present initial results on such throughput, used in the next chapter to simulate three architectures with "BQTerrace" sequence: the two architectures from this chapter and a new one that uses some of the techniques proposed in Chapter 4.

We also synthesized each architecture in two versions, with and without clock gating, the former by allowing the automatic clock gating insertion option from the synthesis tool,

[78] Described by Seidel, Bräscher, Güntzel, and Agostini (2016).

which is the simplest strategy. Finally, to present the area and power breakdowns of the new architectures, we synthesized considering both cases, merging and preserving hierarchy. These cases were named “flat” and “hierarchy”, respectively. Therefore, there is a total of 5 target periods (20ns, 10ns, 5ns, 2.5ns, and 1.25ns), 2 clock-gating related syntheses (with and without it), and two related to RTL hierarchy, resulting in 20 synthesis cases.

7.4 RESULTS

All syntheses for 1.25ns period have not met the timing constraints. Therefore, we will not present those results. The complete set of results is condensed in Appendix C, from Table 22 to 29.

Table 25 contains results under the same synthesis configurations than those used for obtaining the results present in Table 17. Therefore, we rely on such results to compare the two SAD-based FME architectures (the one from Chapter 6 with the new one from the present chapter).

Thus, considering the results of the “flat” synthesis without clock gating (Table 25), the area of the new SAD-based FME architecture is larger than the one from the previous chapter (Table 17) by 2.51%, 4.36%, and 15.93% for 20ns, 5ns, and 2.5ns, respectively. The increase in these percentages as the period decreases demonstrates that there are longer critical paths after the improvements to the architecture. Therefore, these are expected by the larger number of bits in the filters (due to Modification M.1). The same argument justifies the inability to achieve 1.25ns period. The overall area overheads also show that Modification M.1 demanded a larger area than was occupied by the datapaths removed after Modifications M.3 and M.5.

The same modifications responsible for the area increase are expected to be increase power. Power increases are larger than the area increases for 20ns, 5ns, but smaller for 2.5ns (7.07%, 5.21%, and 7.59%). On the other hand, Modification M.2 was expected to cause a reduction in power consumption. Nevertheless, the large portion of the Horizontal Pel TB demands is expected to be only diminished by the adoption of such modification once clock gating is enabled.

However, the automatic clock-gating insertion for the “flat” synthesis was not successful in reducing power. On the contrary, the reported power increased up to 8.5× for the SAD-based FME architecture and more than 4.5× for the SATD-based one. This is because by breaking the boundaries between the RTL hierarchy, the common enable signals may have been spread over the design, therefore demanding a larger number of clock-gating cells. On the other hand, the adoption of automatic clock-gating was successful in reducing power, given the clustering of cells with the shared control logic, i.e., cells that are within the boundaries of a given RTL module. This allows a sparse placement of clock-gating cells that are able to gate larger parts of the design at once.

The area overhead of preserving hierarchy is between 0.3% and 3.7% (without clock gating) and 2.5% to 8% (with clock gating). Such increases are larger for longer clock periods

than for shorter periods. In terms of power, only considering the syntheses without clock-gating, the overhead is 2.4% and 6.9% In such a case, the increases are smaller for longer clock periods than for shorter ones. Given that, in terms of power (and thus energy), it is worth keeping the hierarchy and adopting clock gating. Therefore, in this section we focus on the results of the synthesis using clock gating while preserving the RTL hierarchy.

Table 19 presents the synthesis results for the new SAD-based FME architecture, considering the synthesis configurations that preserve hierarchy and adopt clock gating.

Table 19 – Synthesis results for the SAD-based FME, preserving hierarchy and using clock-gating (equivalent to Table 22).

Target	**1080@30fps**	**1080@60fps**	**2160@30fps**	**2160@60fps**
Period (ns)	20	10	5	2.5
Area (μm^2)	86302.28	86364.03	88867.67	96400.30
Dynamic P. (μW)	1392.70	2821.90	5500.00	11767.10
Static P. (μW)	1344.80	1345.40	1481.00	1814.80
Total P. (μW)	2737.50	4167.30	6981.00	13581.90
Energy (μJ)	2.79	2.13	1.78	1.73

Source: the author.

It is possible to notice a considerable increase in area for 2160@60fps synthesis, which hints that the synthesis tool is using larger gates to meet the tighter timing constraints. Almost half of the total power for 1080@30fps is static, so it is possible to adopt standard cell libraries that trades off speed for a lower static power. The speed, in this case, is not critical, given the large period, while static power is way too representative. Therefore, there is room to improve the energy of the architecture by having libraries with a better fit to such a case. On the other hand, as the area results show, there is not much room for slower cells in the target period of 2.5ns.

Table 20 presents the synthesis results for the SATD-based FME architecture, considering the synthesis configurations that preserve hierarchy and adopt clock gating. The dynamic power of the SATD-based FME architecture nearly double when the period is halved, which is expected given their relation. On the other hand, the increase in total power is a bit smaller in this context, 1.72×, on average ($\sigma = 0.16$), due to the share of static power. Also, the switching activity of this architecture is higher than for SAD-based one, which can be observed in the higher share of dynamic power within the total power. This is also expected, and it is due to the TBs in the SATD datapaths.

Figure 70 shows the area overheads resulting from the adoption of SATD instead of the SAD in the FME architecture, considering all synthesis cases. It is possible to see that the use of SATD occupies almost double the area in all cases. This means that each SATD architecture has almost twice the area of the SAD. After all, the increase within the FME architecture is amortized by the other (quite large) parts of the architecture, such as the Horizontal Pel TB.

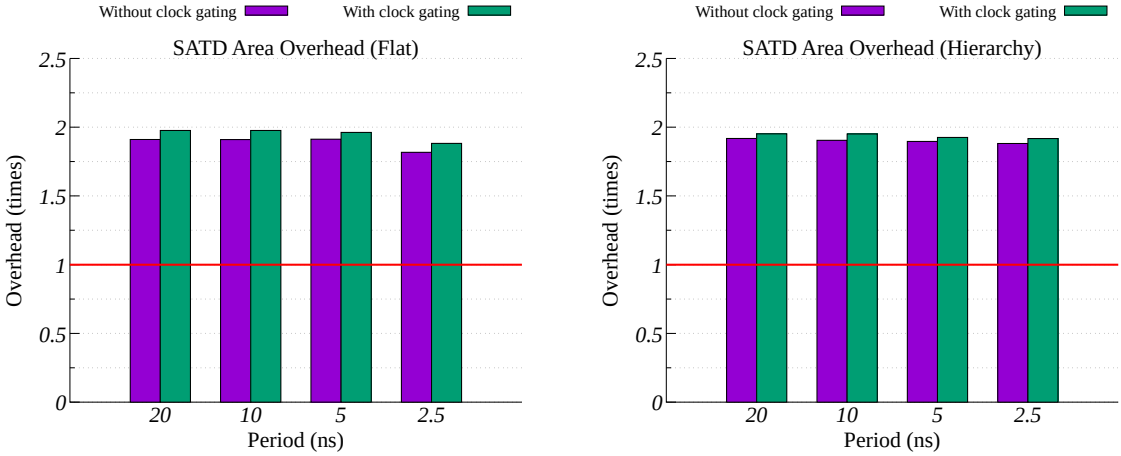
Similarly, Figure 71 shows the total power overheads after adopting SATD instead of the SAD, also considering all synthesis cases. The average total power increase is 1.90×

Table 20 – Synthesis results the SATD-based FME, preserving hierarchy and using clock-gating (equivalent to Table 26).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	168425.48	168512.62	171093.71	184858.20
Dynamic (mW)	3188.80	6171.70	12388.40	24690.80
Static (mW)	2298.60	2298.70	2712.10	3224.30
Total (mW)	5487.40	8470.40	15100.50	27915.10
Energy (μJ)	5.60	4.32	3.85	3.56

Source: the author.

Figure 70 – Area overheads of adopting SATD instead of the SAD. Computed as (area of SATD)/(area of SAD), for each period. The average area overhead is 1.89× (σ = 0.03) for the syntheses without clock-gating, and 1.94× (σ = 0.03) with clock-gating. (a) shows the overheads for "flat" syntheses, while (b) shows the overheads for syntheses that preserve hierarchy.



Source: the author.

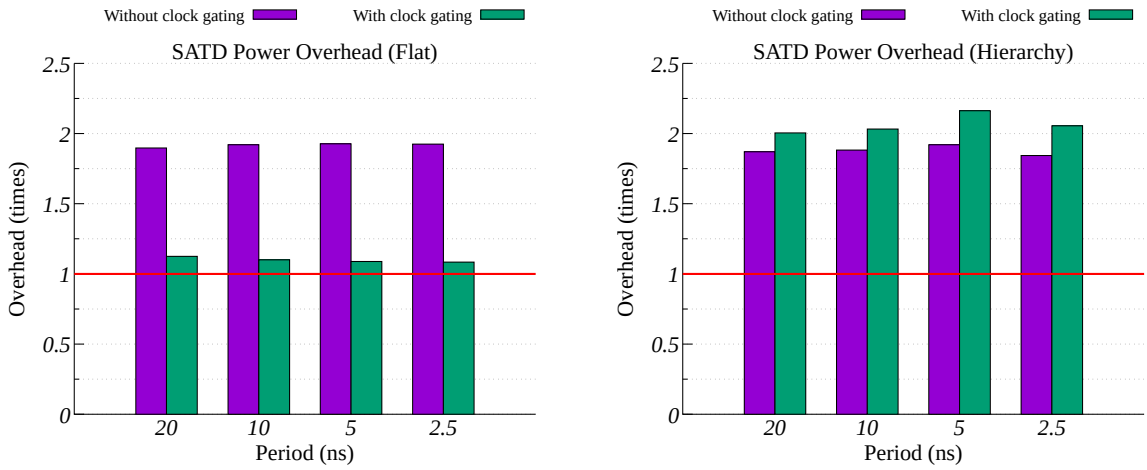
(σ = 0.03) when not using clock gating, and 2.06× (σ = 0.07) when using it (considering only the hierarchy preserving syntheses for clock gating).

7.4.1 Area and Power Breakdown

The detailed results, for the lowest and highest frequencies (20ns and 2.5ns), are presented in Figures 72 and 73 for area and power, respectively. They correspond to the "hierarchy" syntheses with clock gating enabled.

It is clear that for the SAD-based FME architecture, the biggest module is still the Horizontal Pel TB. Also, it is possible to notice that there is a increase in the share of J Tree and Filter modules when the target throughput is increased, indicating the presence of critical paths. Comparing Figure 72a with Figure 63a, particularly for 1080@30fps, which is more directly comparable by having at least the same period, we can see that the share of the Horizontal TB increased from 34.9% to 41.7% and for the Filter it increased from 14.8% to 22%. Both are due

Figure 71 – Power overheads of adopting SATD instead of the SAD. Computed as (total power of SATD)/(total power of SAD), for each period. This gives a hint that, in terms of power, the adoption of clock-gating was more effective in the SAD architecture. This can be justified by the TBs from the SATD datapaths, that cannot be clock-gated while computing a SATD, since all the data contained in it shifts one direction or another, every cycle. (a) shows the overheads for "flat" syntheses, while (b) shows the overheads for syntheses that preserved hierarchies. The small SATD power overhead in (a) when using clock gating looks promising, however, in this case, the clock gating itself was the cause of a large increase in power that so happened to be larger for the SAD-based architecture than for the SATD one (see p. 182).



(a) (b)

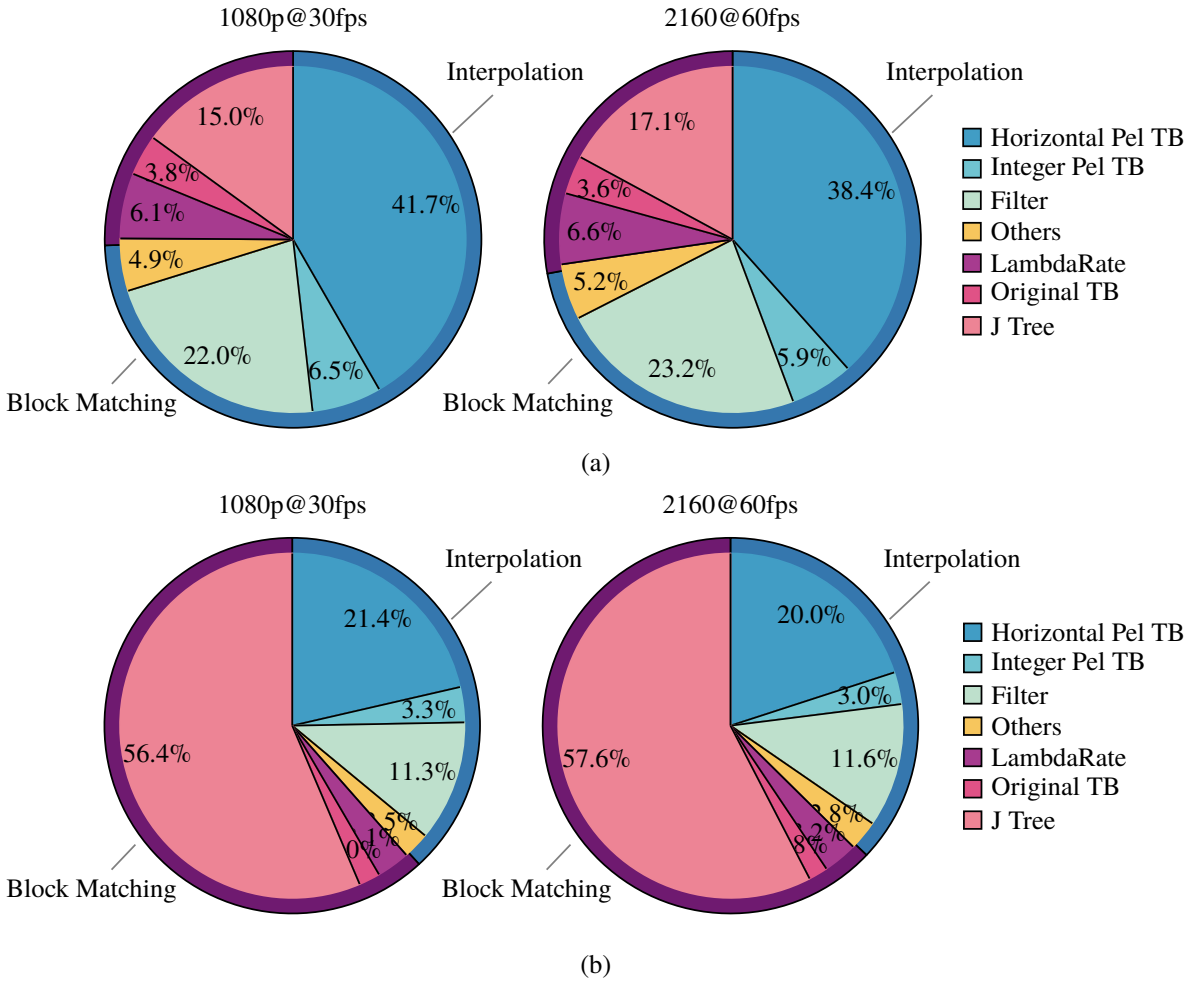
Source: the author.

to Modification M.1.

Also, it is quite evident that the largest area of the SATD-based FME architecture is occupied by the SATDs (J Tree). Each of the 12 SATD architectures occupies about 4.6% of the entire FME architecture. Within each SATD, about 2.8% represents the share of a TB in the FME. Therefore, about 33.6% of the FME architecture is occupied by these TBs, which have thus a larger share than the Horizontal Pel TB. About 7.2% is for the first 1D transform step, and 9.6% for the second 1D transform step. This large area demands by the SATD modules makes the BM module larger than the Interpolation. While when using SAD the BM module occupied only 25.5% to 27.9% of area, now it occupies about 61.8% to 62.9%, for smaller and higher throughput, respectively.

Not unlike the area shares, the adoption of the SATD instead of the SAD also results in a game-changing share of power considering the distribution between interpolation and search. The main difference considering the power shares is that now the Horizontal and Integer Pel TBs show increased power when the frequency increases. This is because now they have larger capacitances to drive on their outputs, making them more power hungry. Although the Filter datapath is part of the critical path, its switching activity seems to be smaller, as its share decreases with higher frequency.

Figure 72 – Area breakdown of the FME architectures proposed in this chapter. The outer ring represents the share of Interpolation and BM. (a) SAD-based FME architecture. (b) SATD-based FME architecture.



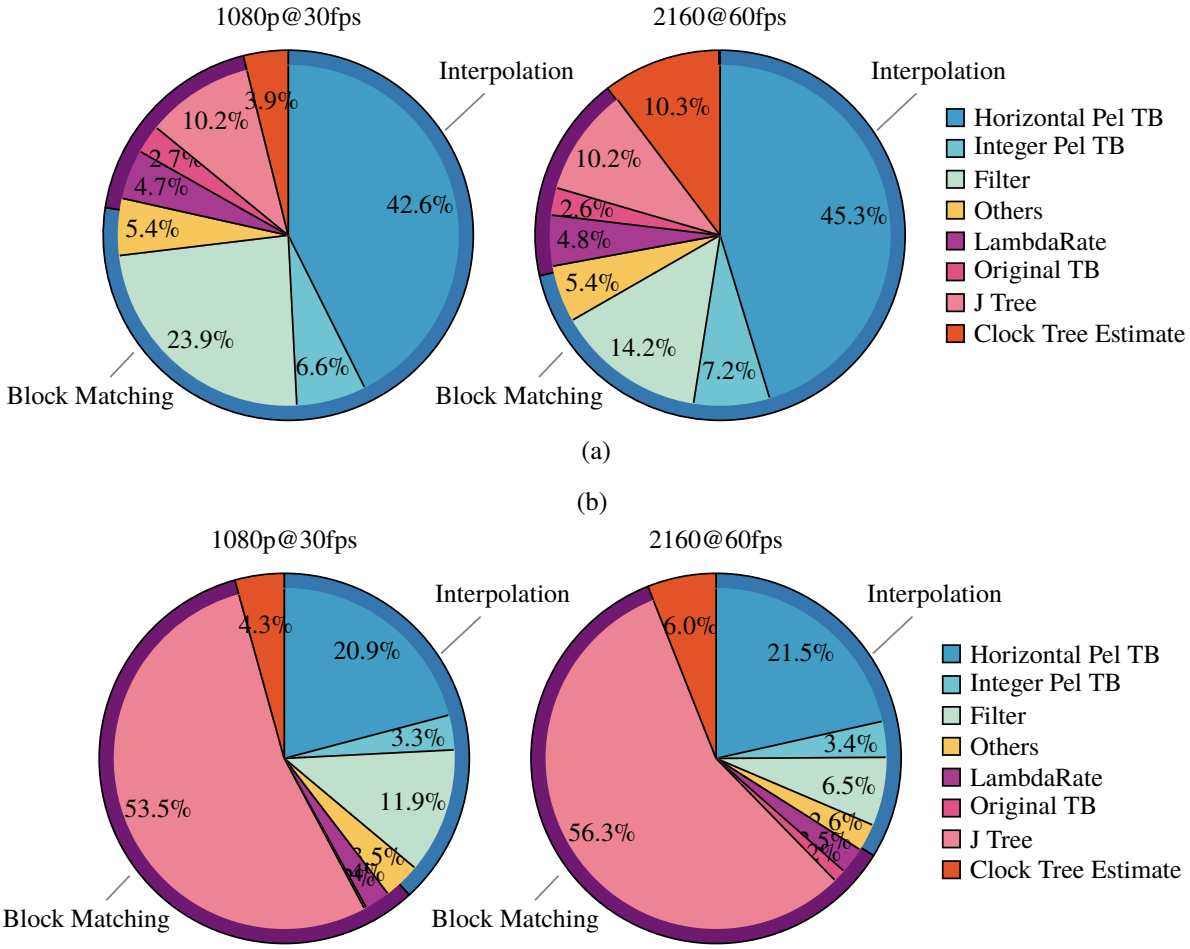
Source: the author.

7.5 CHAPTER CLOSING REMARKS

In this chapter, we proposed, designed, and evaluated several modifications to improve the FME architecture that was presented in Chapter 6. These modifications were in order to abide by FME design recommendations, also presented in Chapter 6 and for make our architecture conformant to HM, allowing it to be tested using HM. Furthermore, we described in detail a novel Rate term (λR) datapath that is multiplier free. To our best knowledge, this is the first work that describes this relevant module in detail.

More importantly, we show that the adoption of SATD in a FME hardware design occupies 2× more area than a similar FME design that used the SAD. The power overheads are similar (also about 2×), but are a bit larger when clock gating is adopted. In fact, the impact of the SATD adoption is so large that it changes the balance between interpolation and search withing the FME. Such impact is also higher than expected when considering only the increase in the number of operations. However, according to Silveira, Porto, and Bampi (2017, p. 1568), in

Figure 73 – Power breakdown of the FME architectures proposed in this chapter. The outer ring represents the share of Interpolation and BM. (a) SAD-based FME architecture. (b) SATD-based FME architecture.



Source: the author.

their analysis of x265 (an open-source CODEC implementation for HEVC), every 1% of bitrate reduction squeezed out of the encoder results in 145% energy consumption increase. This is a bit higher than what is shown here, as we have about 100% increase in energy consumption for about -1.17% BD-Rate (shown in Section 3.4.1), after the adoption of SATD in FME. Nevertheless, as every Joule of saved energy favors the adoption of SATD as a mean of improving coding efficiency, it is of utmost importance to reduce its hardware power consumption.

Finally, while with SAD most of the complexity lied within interpolation step (57.5% *vs.* 38.2% from search, in terms of power), the SATD weights heavy on the search, making it more complex than the interpolation (77.4% *vs.* 18.7% from interpolation). Such shares are quite different from the share presented in Figure 21, with data from Blasi et al. (2015). Even though such a data was collected from the HM implementation that searches within 18 candidates instead of 48 (as our architecture does), both implementations have a 1:1 relation on the number of interpolated and searched candidates. Therefore, this brings evidences that

the Interpolation module is optimized to reduce as much as possible its power needs, while the SATD architectures are more demanding in such a platform (compared to the SAD and to the other operations, like the FME interpolation filters).

Part IV

Applying the techniques to SATD and FME

8 THERE AND (ALMOST) BACK AGAIN: REDUCING FME-SATD COMPLEXITY

To a certain extent, this chapter is a convergence point for all the previous chapters in this thesis. Chapter 1 brought out the problem of the ever increasing complexity when higher coding efficiency is desired. The paradox is that most of the times such increase in complexity is dealt with by trading off coding efficiency. After presenting in Chapter 2 the concepts and the complexity of both SATD and FME, their clear benefit in terms of coding efficiency was was put in evidence through a thorough analysis in Chapter 3. Three methods to reduce the complexity of SATD were exploited in Chapter 4. One method consists in reusing already computed data, whereas the other two are elimination methods. Their finest characteristic is that they do not affect coding efficiency. In Chapter 5, the reuse was exploited further for defining a hardware design for a SATD with multiple sizes. Also, such a chapter presented the TB-based SATD architecture. As the most common application for the SATD is during FME, in Chapter 6 we defined a few guidelines for designing coding- and energy-efficient FME hardware architectures. In Chapter 7, the modifications proposed in the FME architecture from Chapter 6 paved the way for a novel FME architecture which uses SATD as distortion metric. As presented, adopting SATD instead of the SAD in FME increased twofold the power consumption. Such an overhead is significant, and thus is the problem addressed in the present chapter.

Given the embasement provided by the previous chapters, by all of these former chapters, the current chapter brings as main contribution the analysis of how the elimination methods, proposed in Sections 4.2 and 4.3, can be applied to the SATD-based FME hardware architecture presented in Chapter 7. Our detailed evaluation shows where are the benefits, and what are the caveats of using these methods in a hardware implementation of FME. Particularly, for the design of another SATD-based FME architecture that uses the proposed elimination methods to improve energy efficiency, there are two main challenges:

1. The elimination criteria must be designed to avoid large overheads;

2. The SATD-based architecture proposed on Chapter 7 relies on the parallel evaluation of candidate evaluation. This characteristic may reduce the effectiveness of the eliminations;

Therefore, just as in Chapter 4, the designed version of the elimination algorithms must be **simple** and **effective** to achieve maximum **savings**.

8.1 ADAPTING SEA

The Multi-level Successive Elimination Algorithm (MSEA) presented in Section 4.2.3 divides the prediction blocks recursively into four square partitions, similarly to a quad-tree. For instance, considering 8×8 blocks as in Figure 34, level 1 uses a 2×2 HT from four differences, one in each of the four partitions of the initial block. Level 2 takes a 4×4 HT, using as inputs the differences spread over the 16 partitions from the initial block. However, in general the

SATD computation follows a row-by-row or column-by-column order. This is the case of our SATD-based FME architecture. Therefore, instead of relying on the multi-level partitioning scheme presented in Figure 32, we will show that it is possible to compute other elimination criteria that fits better the already designed hardware dataflow.

To ease verify the correctness of all elimination criteria, and thus ensure that there will be no loss in coding efficiency, we implemented a Python script that relies on symbolic math[79]. The key idea is to define two expressions. The first one consists in applying the desired partitioning directly on the transformed differences matrix. For that, our Python script contains functions to symbolically compute the HT, then apply some partitioning over the resulting matrix, add the values from within each partition, and finally perform the absolute sum of the resulting values. By doing so we have an elimination criterion based on that given partitioning strategy. The second expression departs from the differences matrix and computes what we define as the simplified version of the same elimination criterion, i.e., avoiding as much as possible the computation of the transform. If both expressions are equal, we have ensured that the simplified elimination criteria is valid and thus will not change the outcome of BMA. The used library permits to simplify both generated expressions and to check if they are equal[80].

The way we defined new elimination criteria for multi-level in Section 4.2.3 was through successive partitioning of the transformed differences matrix ($\mathbf{T}(\mathbf{D})$). Fine Granularity Successive Elimination (FGSE) (CE ZHU; WEI-SONG QI; WEE SER, 2004) exploits better the quad-tree partition possibilities while seeking to improve the effectiveness. Yet, such a work still relies on a quad-tree structure for computing the tighter and tighter elimination criteria. Nevertheless, it is another example that Property 2 can be further exploited. One possible approach to create another elimination criterion is by dividing $\mathbf{T}(\mathbf{D})$ into rows, instead of partitions. Figure 74a shows such a division considering an 8×8 transformed block.

With such a way of partitioning the matrix in rows, the new obtained elimination criterion corresponds to the first 1D transformed row of the differences matrix (Figure 74b). We named such elimination criterion as $crit^{\text{1st\_row}}$, defined as:

$$crit^{\text{1st\_row}}(\mathbf{D}_8) = 8 \times sa\left(\mathbf{D}_{1,:} \times \mathbf{H}_8\right) \tag{8.1}$$

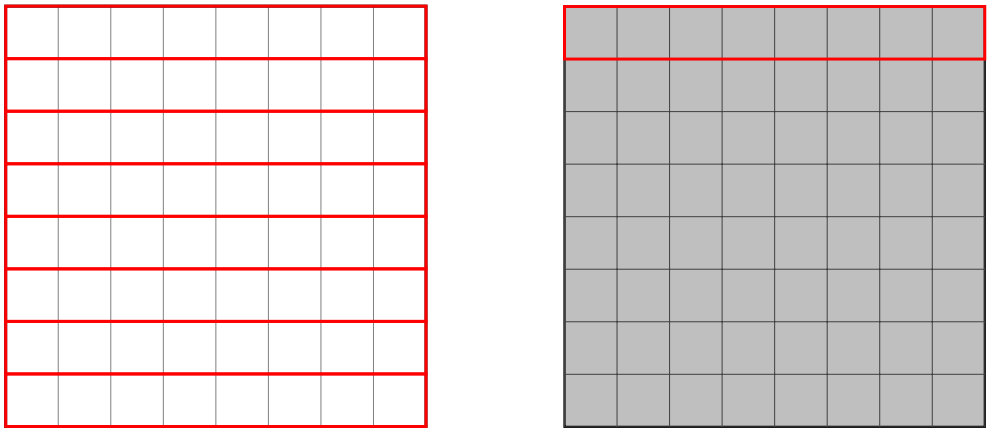
One advantage of such criterion is that it relies on the already computed 1D transform of the first row. As we can reuse such operation, this criterion is simple. On the other hand, it uses only eight values and thus its effectiveness lies within levels 1 and 2 of the multi-level approach from Section 4.2.3, that uses four and 16 values, respectively. To make things worse, the effectiveness will be also lowered because the parallelism of the architecture and by the larger delay taken to find the new j^{best}. Therefore, to improve the effectiveness, let us divide again the transformed differences matrix. Figure 75 shows the adopted partition strategy were each row was divided by half.

The obtained criterion, baptized as $crit^{\text{1st\_5th\_rows}}$, is defined as:

[79] Using SymPy, a Python library for symbolic math (MEURER et al., 2017).

[80] Given two expressions "Exp1" and "Exp2", their equality can be checked by verifying if "Exp1" – "Exp2" = 0.

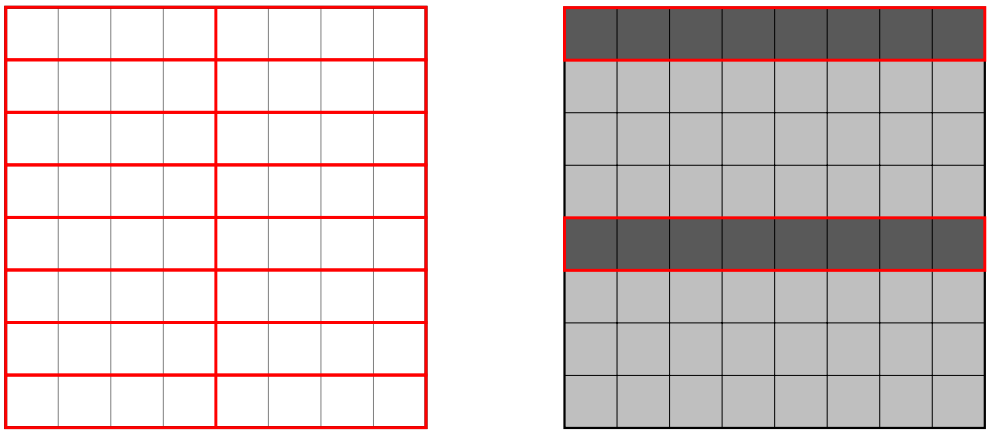
Figure 74 – Divisions of a 8×8 block to obtain new elimination criteria. (a) The block represents the 2D transformed values, while the red rectangles are the division of such transformed block in rows. The elements of each row are summed up, after which their absolute values are also summed, given rise to a new elimination criterion. (b) the gray squares represent each position of the 1D transformed differences matrix. The first row contains the elements used for obtaining the elimination criterion.



(a) (b)

Source: the author.

Figure 75 – Divisions of a 8×8 block into half rows to obtain new elimination criteria. (a) The block represents the 2D transformed values, while the red rectangles are the division of such transformed block in half rows. The elements of each half row are summed up, after which their absolute values are also summed, given rise to a new elimination criterion. (b) the gray squares represent each position of the 1D transformed differences matrix. The first and second rows contain the elements used for obtaining the elimination criterion. The darker tone means that they first needed to be partially transformed.



(a) (b)

Source: the author.

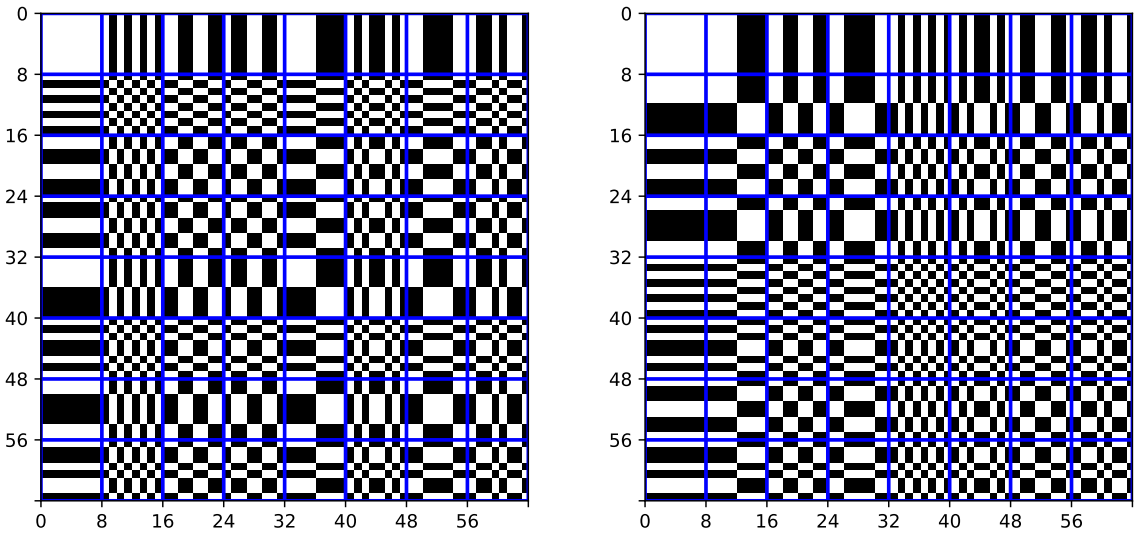
$$\begin{aligned} crit^{\text{1st\_5th\_rows}}(\mathbf{D}_8) =& 4 \times \left(sa\left(\left(\mathbf{D}_{1,:} \times \mathbf{H}_8 \right) + \left(\mathbf{D}_{5,:} \times \mathbf{H}_8 \right) \right) \right) \\ &+ 4 \times \left(sa\left(\left(\mathbf{D}_{1,:} \times \mathbf{H}_8 \right) - \left(\mathbf{D}_{5,:} \times \mathbf{H}_8 \right) \right) \right) \end{aligned} \tag{8.2}$$

Using $crit^{\text{1st\_5th\_rows}}$ may provide a tighter approximation to the SATD than $crit^{\text{1st\_row}}$ because it doubles the number of values considered. If a sequential execution of BMA was to

be considered, its effectiveness would be closer to the one from level 2, that considers the same number of values. However, the complete computation of the elimination criteria on this level would occur only after the fifth line of a block is first transformed. Thus, almost all pixels are already interpolated and almost half of the SATD-TB is already full. This would provide limited savings.

Ideally, the second row should be used instead of the fifth. As presented in Section 2.3 (Figure 16b), the Hadamard matrix can be reordered without affecting the SATD result. After all, this changes the transform as the basis functions are rearranged, i.e., the same elements are still present in the transformed matrix, only in different positions. So far, in our formulations, we used the Hadamard order (GEADAH; CORINTHIOS, 1977, p. 436). Such order is also called Natural, given that this is the order the Hadamard matrix is recursively constructed by Equation 2.18. Figure 76a presents the basis functions of such order. Figure 16b shows the basis functions of the so-called Sequence order of the $\mathbf{H}$ for an 8×8 matrix. Finally, another common order present in the literature is the so-called Dyadic (GEADAH; CORINTHIOS, 1977, p. 436), shown in Figure 76b.

Figure 76 – Basis functions of the HT for 8×8 matrices when using Natural and Dyadic orders of $\mathbf{H}_8$ (GEADAH; CORINTHIOS, 1977, p. 436).



(a) (b)

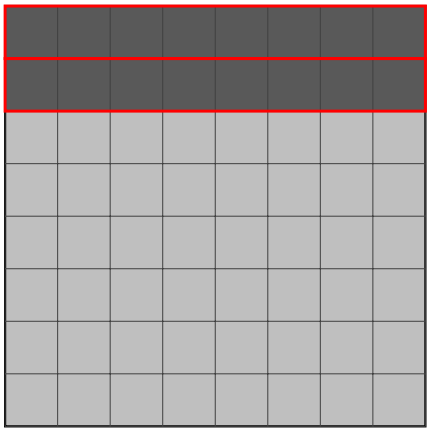
Source: the author.

Indeed, considering $\mathbf{H}_8$, the criteria that uses first and second transformed rows can be obtained by using either sequency or dyadic order. This criterion is called $crit1st\_2nd\_rows$ and is obtained as:

$$\begin{aligned} crit^{\text{1st\_2nd\_rows}}(\mathbf{D}_8) =& 4\times\left(sa\left(\left(\mathbf{D}_{1,:}\times\mathbf{H}_8\right)+\left(\mathbf{D}_{2,:}\times\mathbf{H}_8\right)\right)\right) \\ &+4\times\left(sa\left(\left(\mathbf{D}_{1,:}\times\mathbf{H}_8\right)-\left(\mathbf{D}_{2,:}\times\mathbf{H}_8\right)\right)\right) \end{aligned} \tag{8.3}$$

Figure 77 illustrates this elimination criterion. It shall have similar effectiveness than $crit^{\text{1st\_5th\_rows}}$ given that it also considers 16 values from the $\mathbf{T}(\mathbf{D})$. In fact, both $crit^{\text{1st\_2nd\_rows}}$ and $crit^{\text{1st\_5th\_rows}}$ can be evaluated, as the values present in the latter may be able to eliminate a candidate even if the values from the former are not. Also, although we defined this third elimination criterion after just reordering the elements in $\mathbf{H}_8$, we are not able to follow the same strategy to obtain a criterion similar to $crit^{\text{1st\_row}}$. In such a case, when considering the rows of the transformed differences matrices, all three $\mathbf{T}()$ orders results in the same elimination criteria that uses the first row of 1D transformed differences.

Figure 77 – Elimination obtained after the dyadic ordering.



Source: the author.

These two criteria defined in Equations 8.2 and 8.3 may have increased effectiveness, but they also require extra computations. However, both terms of Equation 8.2 are similar except for the sign between transformed rows. Also, both terms have their absolute values added. Because of these two characteristics, yet another opportunity to save resources presents itself. We can, once again, rely on Property 3 to simplify the architecture as Equation 8.2 can be re-written as shown in Equation 8.4, where $\mathbf{MAX}()$ and $\mathbf{A}()$ are defined in Equations 8.5 and 8.6 , respectively.

$$\begin{aligned} crit^{\text{1st\_5th\_rows}}(\mathbf{D}_8) &= 4 \times \left(2 \times s\left(\mathbf{MAX}\left(\mathbf{A}\left(\mathbf{D}_{1,:} \times \mathbf{H}_8\right), \mathbf{A}\left(\mathbf{D}_{5,:} \times \mathbf{H}_8\right)\right)\right)\right) \\ &= 8 \times s\left(\mathbf{MAX}\left(\mathbf{A}\left(\mathbf{D}_{1,:} \times \mathbf{H}_8\right), \mathbf{A}\left(\mathbf{D}_{5,:} \times \mathbf{H}_8\right)\right)\right) \end{aligned} \tag{8.4}$$

$$(\mathbf{MAX}(\mathbf{B}, \mathbf{C}))_{i,j} = max\left((\mathbf{B})_{i,j}, (\mathbf{C})_{i,j}\right) \tag{8.5}$$

$$(\mathbf{A}(B))_{i,j} = |(B)_{i,j}| \tag{8.6}$$

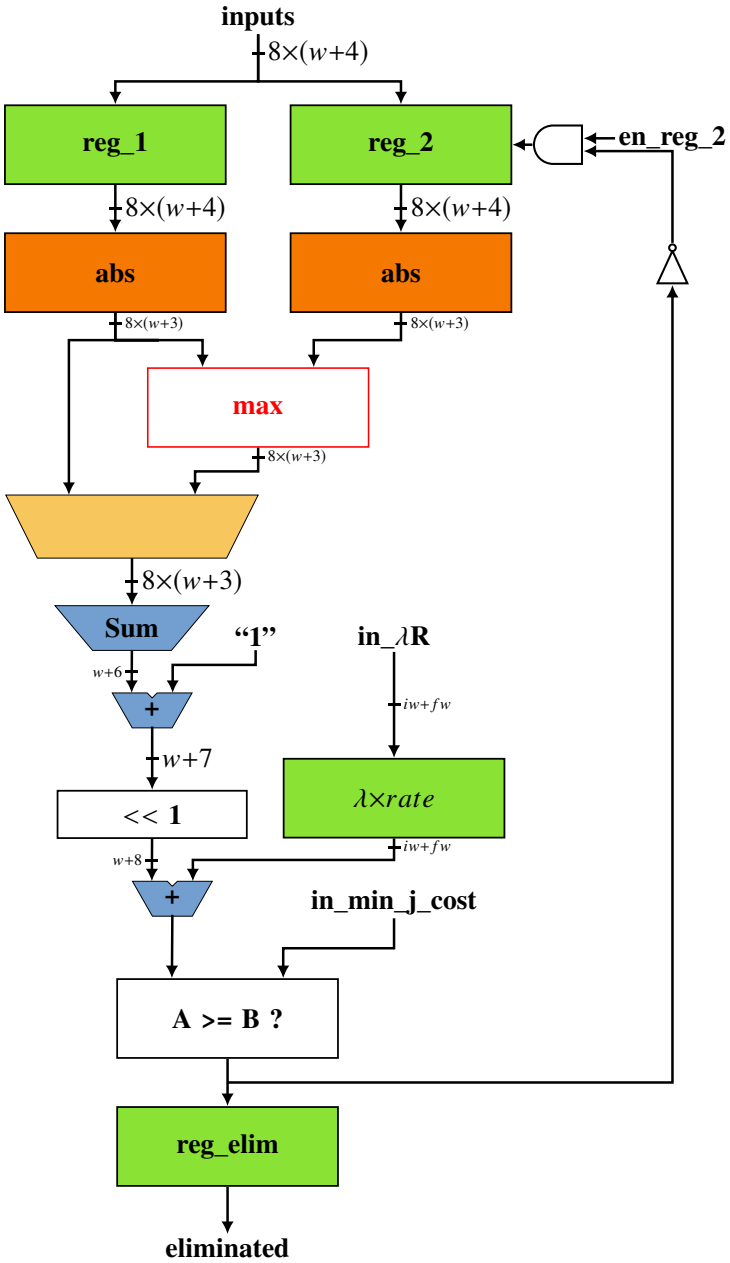
The same is valid for Equation 8.3, that can be written as:

$$crit^{\text{1st\_2nd\_rows}}(\mathbf{D}_8) = 8 \times s\left(\mathbf{MAX}\left(\mathbf{A}\left(\mathbf{D}_{1,:} \times \mathbf{H}_8\right), \mathbf{A}\left(\mathbf{D}_{2,:} \times \mathbf{H}_8\right)\right)\right) \tag{8.7}$$

8.2 ADOPTING SEA

Figure 78 shows our proposed datapath for the elimination criteria computation. Notice that there is no butterflies for transform computation in the elimination datapath. This is because the transform part is computed by the already existing first 1D transform datapath on each SATD architecture. In fact, the elimination architecture is quite simple considering its components. Such simplicity was possible by reusing the already computed transforms and also by relying on Property 3, as shown in Equations 8.4 and 8.7.

Figure 78 – Datapath of the elimination part of the architecture. The register $\lambda \times rate$ is shared with the SATD datapath, so it does not increase the overhead. Figure 46c shows a possible datapath for the absolute operation while Figure 46d presents a possible datapath for "max". The mux serves to select between using $crit^{1st\_row}$ or one of the other two.



Source: the author.

The first 1D transformed row is used in all three criteria that are adopted and so such a row is loaded into `reg_1`. The second and fifth 1D transformed rows are then loaded into `reg_2`. Therefore, there is a register barrier in the Elimination module inputs. Without this, a longer critical path would be created between the interpolation register barrier and the elimination criteria computation. Moreover, such input register barrier avoids continuous switching of the elimination datapath and thus saves power. They can be also used to disable entirely the elimination testing, if some method could detect that the elimination ratio is too low. However, they have a considerable cost in area, since such a barrier has the same number of register bits than two rows of the TB, which has eight rows in total. Therefore, the overhead of solely such a barrier is close to 25% of the TB size.

The evaluation order of the criteria is defined by the generation of the transformed values needed by each criteria. Therefore, first $crit^{\text{1st\_row}}$ is evaluated, then, on the next cycle, $crit^{\text{1st\_2nd\_rows}}$. To compute $crit^{\text{1st\_2nd\_rows}}$, the values of the second 1D transformed row is loaded into reg_2 and the mux selects the second rightmost input. Finally, if none criteria eliminated the current candidate, $crit^{\text{1st\_5th\_rows}}$ is computed three cycles after $crit^{\text{1st\_2nd\_rows}}$, when the fifth 1D transformed row is loaded also into reg_2.

There is a small logic to control the loading of reg_2, since it should only be enabled to load new values if the current candidate block was not eliminated yet. Otherwise, it could generate a criterion value that could be lower than the already computed one. Thus, it would be possible for the new value to be lower than the minimum cost and the elimination would be false again.

To increase the benefits of the elimination before the TB of each SATD, the 27 registers in the register barrier within the clipping module are segmented in three parts, containing 9 registers each. Each of these three parts corresponds to the inputs of a group of four SATDs modules. Whenever one of such group raises the `eliminated` signal, the correspondent segment of clipping registers is disabled, thus avoiding switching activity in such registers and in the first 1D transforms of the group of SATDs. If the three groups have all SATDs eliminated, an `eliminated_all` signal is raised, that stops the counter that issues the addresses used in the Horizontal Pel TB. This is the best case, because it reduces the switching of all the interpolation filters and of all wires from the Horizontal Pel TB (which is clock-gated).

During the evaluation of horizontal and FOVS, the elimination datapath is disabled to save power. This is because in this cases only half of the 12 SATDs are active, so the benefit of eliminations are lower. Also, even if all active SATDs signalized elimination in the horizontal candidates evaluation, the interpolation could not stop, as those values are needed to obtain the SOVS. Moreover, the effectiveness of these cases is expected to be lower, since the probability to chose one of these candidates is higher, which can be seen in the heatmaps from Figure 25, and justified by the heatmaps in Figure 15. Therefore, to avoid incurring in unnecessary overheads, our strategy was to keep the elimination module disable during the evaluation of the first 12 candidates, and enable during the evaluation of the remaining 36 candidates.

For each one of the 12 SATD modules in the BM module, we integrated one Elimination

module. Also, we created a simple logic in the accumulation of each SATD to implement Fine-grain PDE, as presented in Section 4.3.3. Such a logic has minimum overhead, and its savings are limited because in the first cycle it may early terminate one SATD, the transpose buffer is already getting inputs of the next candidate block. Nevertheless, if an elimination is detected in such small module, the switching of the second 1D transform may be saved.

We baptized the SATD-based FME architecture that adopts both of these algorithms (SEA and PDE) as SEA-FME architecture. For simplicity, the two FME architectures presented in Chapter 7 will be called SAD-FME and SATD-FME.

8.3 EVALUATION METHOD

A similar synthesis method from Section 5.3.1 is adopted. There are only a few differences, like clock periods, the adoption of more data for simulations and a step further to obtain even more detailed power consumption reports.

We synthesized our three FME architectures (SAD-FME, SATD-FME, and SEA-FME) for the same periods described in Section 7.3: 20ns, 10ns, 5ns, 2.5ns, and 1.25ns. To follow the method presented in Figure 54 to obtain realistic power estimates we used the netlist synthesized for 10ns target period that is able to achieve a throughput of 1080@60fps. This is because our main simulation case will adopt "BQTerrace" sample (see Table 7). The adoption of such a sample is also justified in Section 5.3.1.

The three FME architectures were simulated using the executables compiled by VCS®, considering the simulation model from Figure 55. However, instead of providing just one final report on the power consumption, we collected a fresh report for each frame[81]. By doing so, we were able to account for the variations in QP for hierarchical B-frames, as shown in Figure 22b.

The simulations were performed for the four QPs from the CTC, 22, 27, 32, and 37. Also, we adopted a random[82] 1:100 sampling FMEs from each frame. This means that for each 100 FMEs executed in HM, about one is used to simulate the architecture, i.e., 1%. Such a measure is necessary because of the too-long execution times of a simulation with 100% of the FMEs, which would limit the simulation to only a few frames. Therefore, for each frame about 324 FMEs were executed, limited to 32 frames. Thus, three architectures with four QP and 324 FMEs/frame and 32 frames lead to about 124.416 FMEs simulated and about 5.971.968 candidates evaluated for a single video sample.

To provide a more in depth analysis of the benefits and caveats of the SEA-FME architecture, we went further in the power evaluation method. We used Synopsys® PrimeTime (PT®) to generate cycle-accurate power estimates for all three FME architectures.

[81] A new SAIF is generated after each frame during simulation. Each SAIF is then used to obtain more realistic power reports on DC®.

[82] Using C++ `rand()` function from `<cstdlib>` with seed 42.

8.4 RESULTS

Similarly to the architectures from Chapter 7, the synthesis tool was unable to meet the 1.25ns timing constraints for SEA-FME architecture. Table 21 shows the results for the SEA-FME architecture when synthesized for the remaining target clock periods, adopting clock gating and preserving hierarchy. The results for the other synthesis setups are present in Appendix C, from Table 30 (equivalent to Table 21) to Table 33. Compared to the results for the SATD-FME from Table 20 it is possible to see an increase in all results. Although the increase in area is expected, the goal of using SEA was to reduce power. However, these results do not consider yet the switching activities.

Table 21 – Synthesis results for SEA with clock-gating and preserving hierarchy (equivalent to Table 30).

Target	**1080@30fps**	**1080@60fps**	**2160@30fps**	**2160@60fps**
Period (ns)	20	10	5	2.5
Area (μm^2)	198026.28	198154.88	201118.93	217917.85
Dynamic P. (μW)	3319.50	6433.30	12751.40	25602.50
Static P. (μW)	2691.50	2698.50	3138.00	3787.00
Total P. (μW)	6011.00	9131.80	15889.40	29389.50
Energy (μJ)	6.13	4.66	4.05	3.75

Source: the author

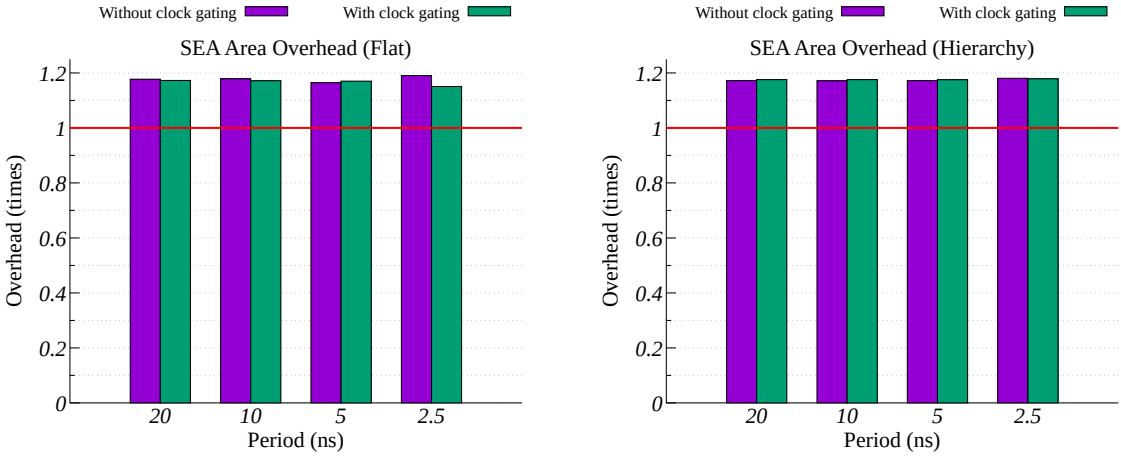
Figure 79 shows the area overheads of SEA-FME with respect to the SATD-FME architecture, for all syntheses. The average area overhead is about 17.37% (σ=0.84%), which is almost as large overhead as predicted in Section 8.2, considering the number of extra registers to store the 1D transformed values used to compute the elimination criteria. Even considering all efforts to keep the Elimination modules simple, it still occupies about 14.4% of the FME total area after hierarchy preserving syntheses.

Comparing all power results before the simulation to obtain the switching activities shows a power increase of 3.39% (σ=4.08%) with clock gating and 12.62% (σ=2.29%) without it. The large variance for clock gating results, when considering the mean, is mostly due to the differences from "flat" and "hierarchy" syntheses. Thus, only for clock gating result, the "flat" synthesis presents a small average power reduction of -0.18% (σ=0.42%) and when preserving the RTL hierarchy there is a increase of 6.96% (σ=1.71%) in the estimated power.

Figure 80 provide more reliable power estimates that account for the switching activity, thus showing the power results for the three architectures obtained after the simulation of 32 frames of the "BQTerrace" sequence. The results are presented in crescent order of Picture Order Counts (POCs), i.e., they are in the correct temporal ordering. Appendix C also presents realistic power results considering the simulation, from Figure 84 to Figure 86. Figure 84, for instance, presents the total power savings (%) of the SEA-FME with respect to the SATD-FME of "BQTerrace" sequence.

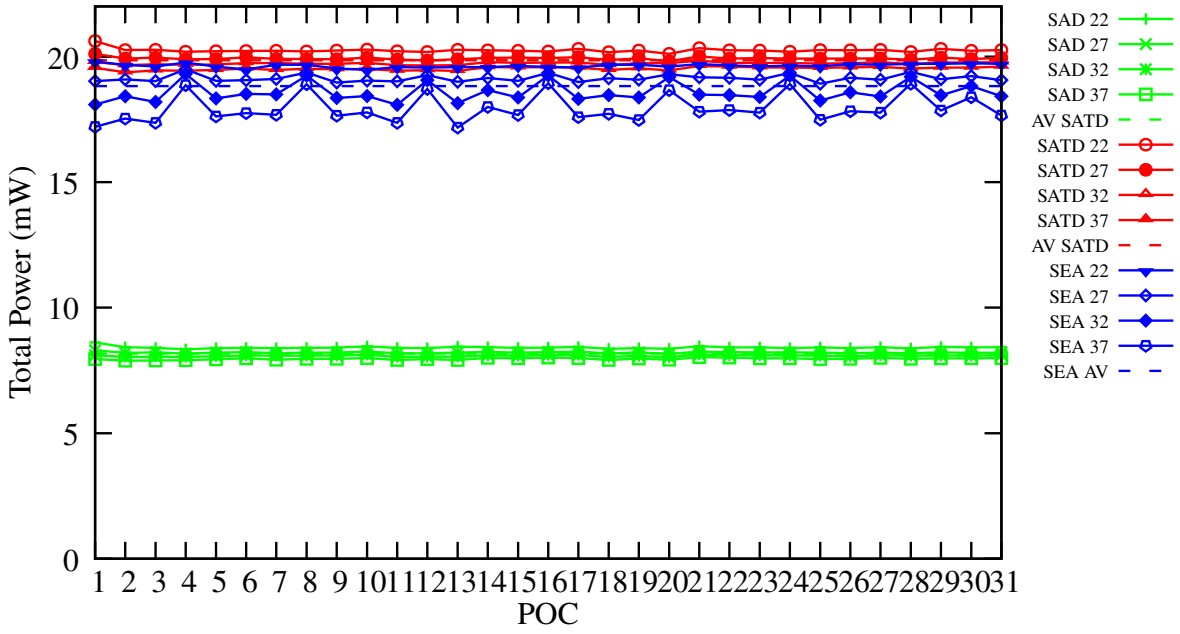
There is no results for the first frame (POC= 0) since it is an I-frame. Comparing to

Figure 79 – (a) shows the overheads for “flat” syntheses, while (b) shows the overheads for syntheses that preserved hierarchies.



Source: the author.

Figure 80 – Power results for each frame of “BQTerrace” sequence.



Source: the author.

the results from Tables 19, 20 and 21, it is possible to notice that the simulations unveil a larger power consumption than the initially reported by the synthesis tool, for all three architectures. Although using SATD instead of the SATD still requires around 2× more power, there was a increase in the difference: without simulation was 1.9×, increasing to 2.3× after the simulation (for this content). Moreover, Figure 80 shows evidences that the use of SEA-FME is able to successful reduce the power consumption of a FME architecture that adopts SATD.

Notice that the results of SEA-FME have the largest deal of variation among the architectures considering the POC. This is because despites we determine a QP value in the HM

configuration, the QP that is actually used depends on the frame position within the GOP. For instance, the configuration with QP 37 uses the following sequence of QPs (POC): 34 (0), 46 (1), 45 (2), 46 (3), 44 (4), 46 (5), 45 (6), 46 (7), 41 (8), 46 (9), 45 (10), and so on. Therefore, it is clear the impact that the QP has in the savings provided by our SEA-FME. This is because a larger QP increases the λ, which in turn weights heavier on the rate term of the adopted simplified RDO and thus increases also the effectiveness of the elimination criteria. Such result is thus reaffirming those presented in Section 4.2.

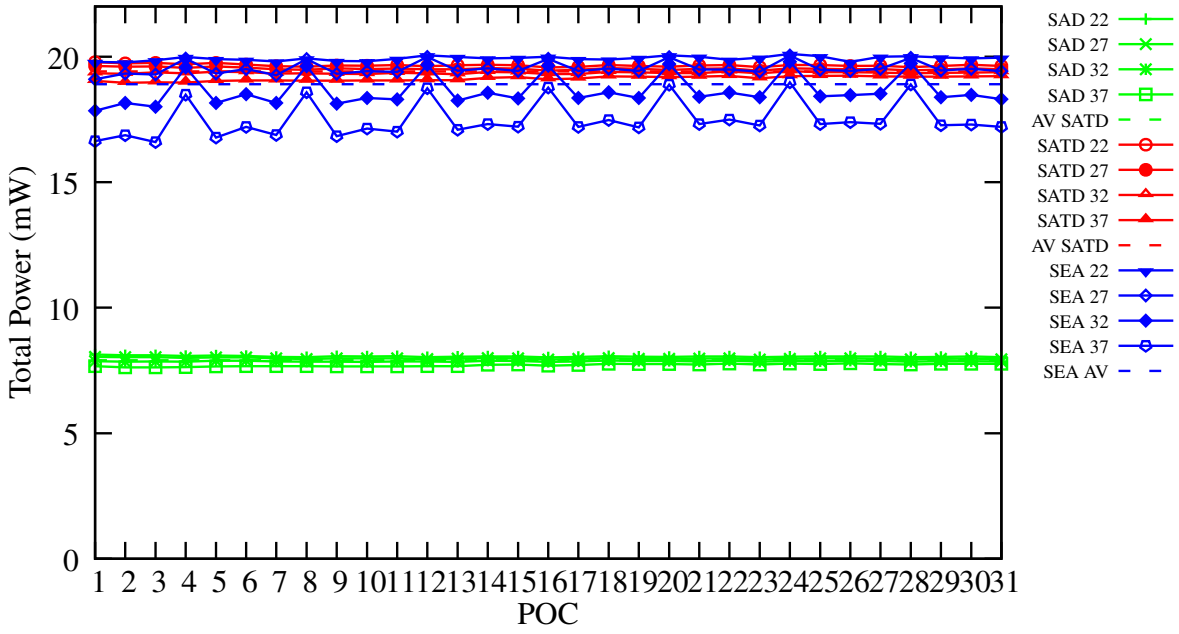
Figure 81 shows simulation results for other two sequences from the CTC (BOSSEN, 2012), PeopleOnStreet and Jhonny. However, since we use the same synthesized netlist and testbench (including the same clock period, 10ns) for simulation that we used for "BQTerrace", the actual results could be a little bit different. Nevertheless, these results illustrate effect that video content has on the eliminations, as expected by the results from Section 4.2, and thus on energy. While "PeopleOnStreet" is a very dynamic sequence with a lot of movement, "Jhonny" has much less movement and large homogeneous areas. On the one hand, both these sequences exhibit higher benefits (lower power) than "BQTerrace" for the higher QPs. On the other hand, they all show worst results for QP 22. In fact, "Jhonny" is clearly taking no benefit of the Elimination module for such QP. Nevertheless, it is the sequence that shows the highest energy for QP 37: using SEA-FME used for this sequence is up to 25% more energy-efficient the SATD-FME (this can be seen in Figure 86).

The power results are a balance of the savings achieved by eliminations, i.e., the reduction in switching activity, and the overheads that are present even when no eliminations occur. As pointed in Chapter 4, the higher the eliminations (effectiveness), the more expressive are the benefits brought by adopting these techniques.

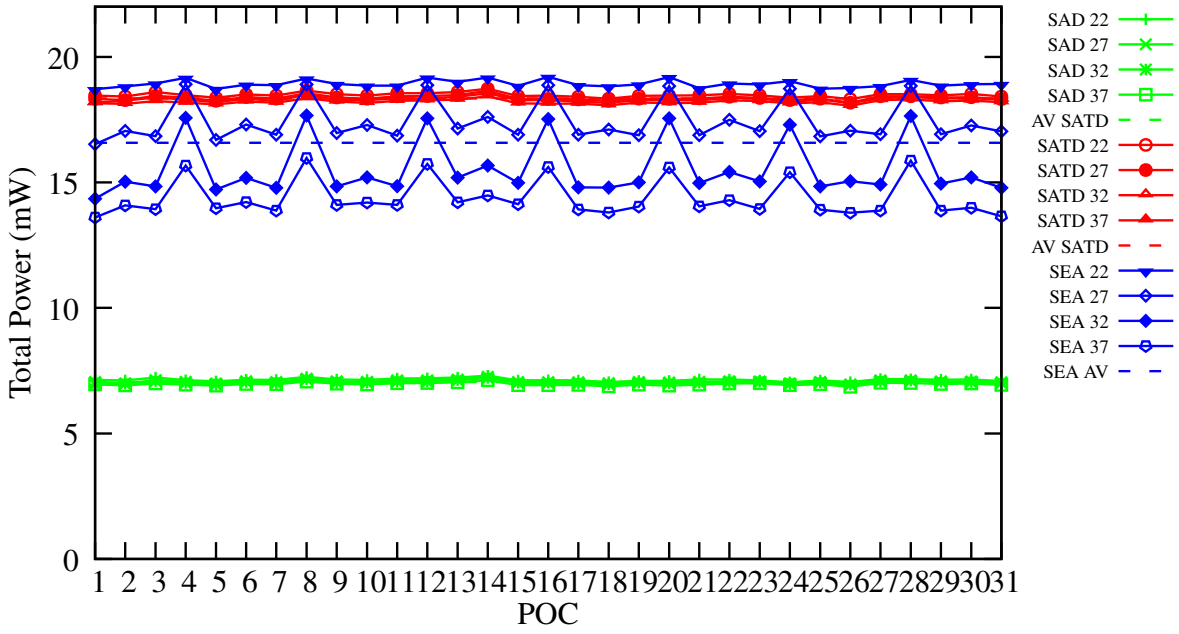
Figure 82 shows the total power that is spent during each clock cycle, highlighting one FME execution where eliminations occur. Such execution is the first one simulated, as can be seen in the time axis. Considering the SAD-FME architecture, it is possible to identify two distinct power plateaus, one higher than the other. The first one (higher power plateau) begins right after the horizontal interpolation begins, while the second one (lower power plateau) begins when the horizontal interpolation finishes. Indeed, one may notice that the higher plateaus happen during the cycles when `en_sr`= 1. Thus, the initial cycles consume most of the power for the SAD-FME, because the loading of integer pixels in the integer buffer, their interpolation and subsequent loading in the horizontal buffer. The peak in power in the lower plateau is due to the change in the direction of original block buffer.

Both SATD-based architectures have a power consumption similar to the SAD-FME during the initial cycles. However, in the SATD case the power increases in a larger proportion when the distortions of the first candidates begin to be computed (`en_0_5`). In such case, the SATD-based architectures consume about 50% more power than the SAD-FME. After the horizontal pel interpolations are done, the SAD architecture experiences a sudden drop in power, given the low complexity operations of the SAD and the buffers being now accessed by addresses (reduced switching activity in the buffers). Notice that the second time the SAD-FME initializes

Figure 81 – Power results for each frame of (a) “PeopleOnStreet” and (b) “Jhonny” sequences. Figures 85 and 86 shows the relative savings of the SEA-FME with respect to the SATD-FME for (a) and (b), respectively.



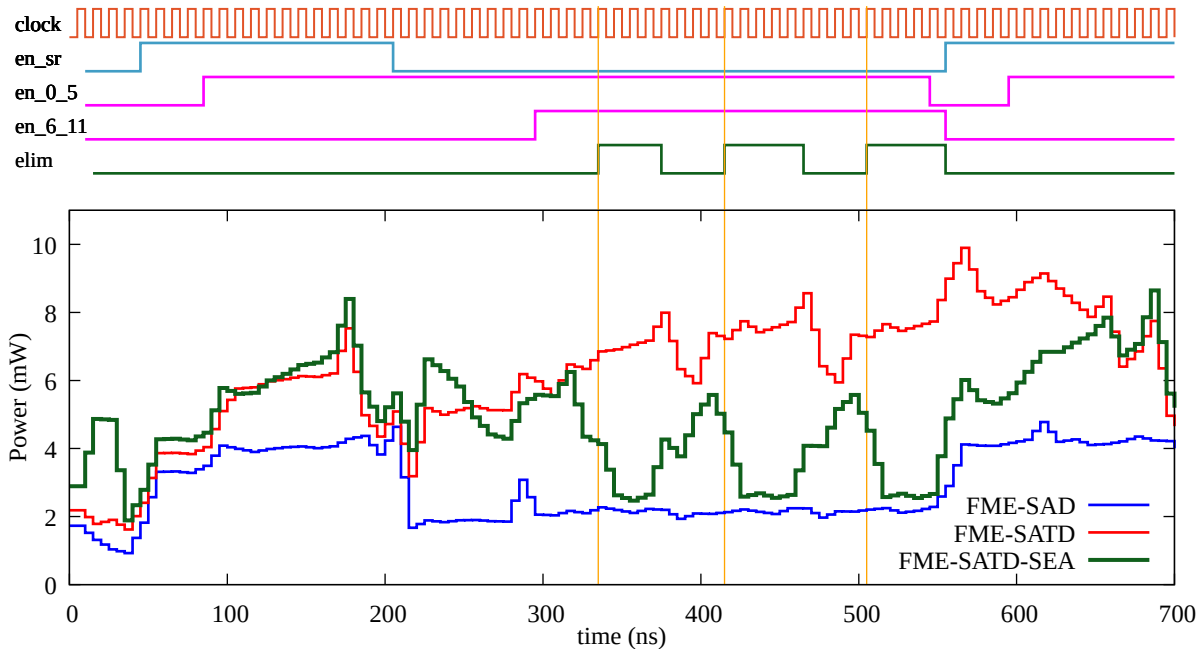
(a)



(b)
Source: the author.

a FME (near the second time `en_sr` is enabled), the higher power plateau is lower than the fist higher power plateau. Considering that the first is starting with reset buffers, its switching activity may be larger than the activity within the second plateau. After all, the differences between interpolated pixels can be small. In fact, for this same reason the sampling of FMEs may overestimate the power consumption.

Figure 82 – Power at each cycle of the three architectures depicting eliminations. The three vertical lines that span the signals and power graphs are showing the three times all the SATDs for the twelve SATD architectures were eliminated. This clearly demonstrates the potential for saving energy by means of SEA.



Source: the author.

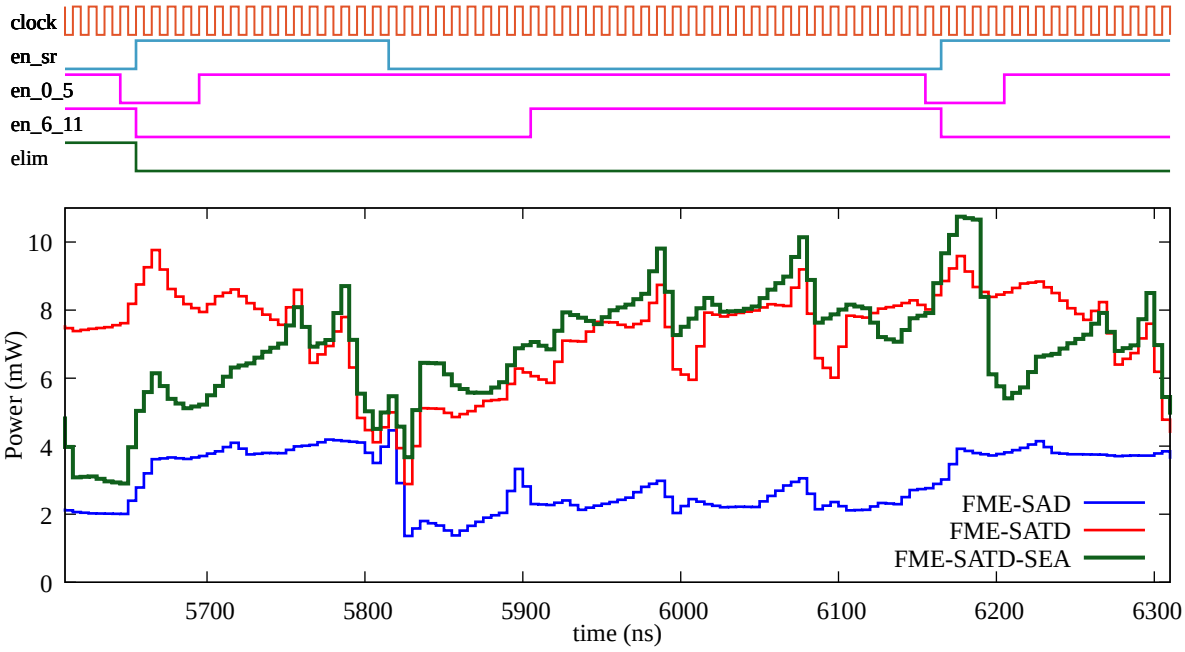
Regarding the SATD-FME architecture, there is no distinction of power plateaus after finishing horizontal interpolation. Such is because that, although the integer and horizontal buffers reduced their switching activity, the TBs of SATD modules are constantly switching, alternating directions. This also explains the highest power peaks near the begin of the second FME shown in Figure 82. In such a case the TBs of the SATD modules are still switching to finish the fist FME in parallel to the initial interpolation cycles of the second FME, that enables the Integer and Horizontal Pel TB modules. Also, it is possible to notice the large increase in power after enabling the second row of SATD architectures (`en_6_11`).

The best case, in terms of reducing power consumption, is when all 12 parallel SATD calculations are eliminated. When this happens, not only the TBs of all SATD architectures are halted, but also the interpolation. To avoid disrupting the regular dataflow, the power reduction strategy adopted in these cases is clock gating. Of course, the sooner all parallel SATDs are eliminated, the larger are the power savings. Moreover, considering the SEA-FME architecture, when eliminations occur, the power consumption drops to almost being the same as in SAD-FME. This is how the elimination algorithms, proposed in Sections 4.2 and 4.3, can provide savings after all, in a highly parallel hardware architecture.

Yet, as we presented in those sections, if the elimination ratio is too small, the overhead of computing the elimination criteria may decrease the savings. Finally, Figure 83 shows that, in fact, the overheads cause the architecture to consume more power if no elimination occurs. Thus, the effectiveness must be high enough to ensure that the overall savings must be higher

than the overheads. For instance, the power consumed for executing the FME on “BQTerrace” (Figure 80) and QP 37 shows that there are more FMEs in each frame with power profiles similar to Figure 82 than similar to Figure 83.

Figure 83 – Power at each cycle of the three architectures depicting no eliminations. This shows the overheads of the Elimination module that takes its toll when no eliminations occur.



Source: the author.

8.5 CHAPTER CLOSING REMARKS

In this chapter, we proposed a new SATD-based and SEA-capable FME hardware architecture. To abide for the data flow of FME, we proposed three novel elimination criteria, trying also to avoid redundant transform operations. Two of them are evaluated in the first cycles the interpolated data are ready, and thus they have the largest potential to save operations. The architecture was devised as to avoid computing other criteria if the current block is already eliminated. The third criterion serves as a “backup” way to eliminate the current candidate, if none of the the other two were successful.

However, the main limitation of the current SEA-FME architecture is its overhead in terms of power for the cases where almost no eliminations occur, even though the architecture was carefully tailored to reduce the number of operations in the Elimination module to a minimum. Therefore, although the area overhead may be tolerable if it brings enough power/energy consumption benefits, there is still room for further improvements regarding the power overheads. One way to achieve that is by understanding better what are the cases when eliminations are more probable to happen, and only enable the Elimination module for those cases. By

doing so, it would be possible to benefit the savings from eliminations, without the extra cost of computing the criteria even if they would not eliminate candidates.

Nevertheless, in this chapter we have demonstrated that the elimination of impossible candidates can be applied for the SATD, even on a highly parallel architecture as the one for HEVC FME that we proposed. Although such parallelism limits the effectiveness of the techniques from Chapter 4, our detailed power analysis shows that when eliminations occur on most of the J Tree, the savings are quite expressive: almost to the point where the architecture would consume as little power as the SAD-based FME architecture (at least for the remaining cycles for that group of 12 SATDs). In some of the evaluated cases, the present SEA-FME architecture saves up to 25% of total power and energy with a remarkable and valuable added benefit: without reducing coding-efficiency.

9 CONCLUSIONS

There is no doubt that portable devices, in special the smartphones, changed the way of life by the beginning of 21st century. Video coding is one major application embedded in these devices. Therefore, there is no doubt either that video applications changed every day's life. We communicate through video; we learn; we often record and share special moments. Not surprisingly, astonishing 75% (CISCO, 2018) of today's Internet traffic is video. The foreseeable future holds even more video with more features: we on the verge of immersing in such a media. Hence, we can only expect to have even more data to compress, and to compress it ever more efficiently.

As presented in Chapter 1, video compression involves several trade-offs, from the rate-distortion (coding efficiency) to complexity and energy. Paradoxically, most methods to reduce the computational complexity end up by decreasing the coding efficiency. Also, Moore's law is no longer sufficient to compensate for the snowballing complexity increase and thus it cannot be expected anymore to be the sole enabler of those new applications.

Video encoders embedded in smartphones must provide excellent trade-offs, given that the user satisfaction depends on both the camera quality and battery duration. Moreover, increased energy consumption shortens batteries lifetime, resulting in electronic waste. Therefore, dedicated hardware architectures for video coding are "a must have" feature for these devices. After all, saving energy while maintaining coding efficiency provides benefits for consumers, manufacturers, and the environment as well.

The main contribution of this thesis is the confirmation of our hypothesis that **it is possible to improve the energy efficiency of video encoding without reducing coding efficiency by relying upon mathematical properties of SATD and prediction tools**. For that intent, we proposed and evaluated novel techniques to reduce the cost (in terms of number of operations, energy, and area) of adopting the SATD instead of the SAD in one the most expensive tools of video coding, the FME, that executes two main steps: interpolation and BMA. To avoid reducing coding efficiency, the techniques presented exploit mathematical properties of SATD when used during BMA, without compromising coding efficiency.

The complexities of the SATD itself and SATD-based FME were laid down in Chapter 2. In Chapter 3, we provided a detailed coding efficiency analysis, showing the benefits of using the SATD and also demonstrating different approaches to FME, foreseeing its hardware implementation. Thus, the increase in coding efficiency worths the overhead added from using SATD during inter prediction and this answers our first research question (RQ-1). Moreover, in such chapter we also demonstrate the coding efficiency brought by using the FME, showing that it is a key tool of the encoding, answering the second research question (RQ-2).

To answer the third one (RQ-3), we extended three techniques from the context of SAD to be used with the SATD. In other words, these techniques were adapted for SATD, given rise to new techniques that we described and analyzed, showing that they can be simple yet effective. We presented mathematical proof that the proposed techniques in their most simple form can

save operations while keeping the same coding efficiency. Therefore, we could overcome the limitations imposed by the transform part while adapting those techniques.

The remaining of the thesis was dedicated to answer the final research question (RQ-4). For that purpose, we had to design hardware architectures for SATD and also evaluate them in the context of a broader encoding tool, such as the FME. Our design space exploration led to novel SATD and FME hardware architectures, revealing that some mathematical properties of SATD could be further exploited by adopting appropriate hardware organizations. Such was the case, for instance, of reusing transform operations that allowed us to also reorganize internal buffers of the architecture, representing about 1/3 of area and power reduction. Moreover, as a byproduct of designing the FME for both SAD and SATD distortion metrics, we provided means to reliably compare them in terms of area and energy efficiency.

The SATD-FME hardware design was then adapted, and we show that the elimination techniques proposed in Chapter 4 can be adopted in parallel hardware implementations, in spite that their effectiveness may be reduced. This is because the parallelism jeopardizes the effectiveness of the proposed techniques. Even so, up to 25% of energy was saved by adopting the proposed techniques within the FME architecture. Furthermore, in some cases the increased energy efficiency came from avoiding filter operations in the first step of FME, showing the importance of integrating interpolation with BMA for the FME.

Thus, the answer to the last research question (RQ-4) is that the proposed techniques bring benefits in terms of energy efficiency, even though they may be limited by the parallelism. Also, while the proposed techniques alone are not sufficient to allow for energy-efficient complete video encoder hardware designs, they certainly help to improve energy efficiency. Even if sometimes the benefits in terms of energy efficiency may be small, these techniques are worthy to be employed. After all, every pico Joule of energy saved is valuable and even when combined with other lossy techniques, the techniques we proposed will not cause further coding efficiency degradations. As a matter of fact, any energy saving technique that does not impose coding efficiency losses deserves high consideration.

We also show that the power estimates must consider the switching activity to be more accurate. Moreover, to ensure correct and fair comparisons with related works, it is fundamental to use exactly the same synthesis configuration (method involving tools, commands and standard-cell library). This poses as an implicit limitation of most works that propose new hardware designs. As the synthesis tools are commercial and thus one may not have them all to abide to other works methods, the ideal case would be for the authors of scientific papers describing hardware architectures to provide their RTL description along with the used testbench. This would allow more fair and reliable comparisons, as well as it would serve as a cross-check on the correctness of descriptions. To such extent, we have published our descriptions in public repositories[83], thus allowing for reproducibility and fair comparisons.

Finally, it is worthy to sustain that the proposed FME design strategy can be used to

[83] All related materials not listed throughout the text are available in: `https://gitlab.com/ismaelseidel/thesis-exploiting-satd-properties`.

design FME architectures for future video coding standards. Furthermore, each one of the four proposed architectures for the HEVC FME may be adopted “as is” for these standards. This is because they use the same filters adopted in HEVC for their 1/2 and 1/4 fractional sample interpolation[84]. The drawback is a possible loss in coding efficiency. Yet, such an analysis should be carried out as a future work.

9.1 FUTURE WORKS

As future works it should be analyzed how much the combination with other algorithms to reduce complexity can affect the capabilities of this proposal to also reduce complexity, i.e., how other algorithms may reduce the effectiveness and thus the savings obtained from adoption of the proposed elimination strategies. Also, it is possible to adaptively change the adopted strategy to only enable the proposed techniques only in cases where they can improve the energy efficiency.

Other related works could provide additional results, such as the analysis the coding efficiency loss of using only 10-bits for the Horizontal Pel TB. Also, one could analyze the use of increased MV fractional accuracy as defined in future standards, such as VVC and EVC, providing insights on its coding efficiency. A complete analysis of the SEA savings for each of the CTC sequences, considering the elimination criteria adopted in the FME architecture could help to identify other oportunities to reduce the power overhead of the cases with low elimnation ratio. Such analysis could take into account the parallelism and delay to update the best cost, as a way to measure the elimination ratio of the architecture without its simulation.

[84] See Note 39.

REFERENCES

A. LUNDGREN, Cynthia et al. Lithium-Ion Batteries and Materials. In: SPRINGER Handbook of Electrochemical Energy. [S.l.: s.n.], Jan. 2017. p. 449–494. DOI: 10.1007/978-3-662-46657-5_15.

ABDELAZIM, Abdelrahman; VARLEY, Martin; AIT-BOUDAOUD, Djamel. EFFECT OF THE HADAMARD TRANSFORM ON MOTION ESTIMATION OF DIFFERENT LAYERS IN VIDEO CODING. In: ISPRS Commission V Mid-Term Symposium. [S.l.: s.n.], June 2010.

ABREU, Brunno; GRELLERT, Mateus, et al. Exploring Motion Vector Cost with Partial Distortion Elimination in Sum of Absolute Differences for HEVC Integer Motion Estimation. In: IEEE International New Circuits and Systems Conference. Munich, Germany: [s.n.], Apr. 2019.

ABREU, Brunno; SANTANA, Gustavo, et al. Exploiting Partial Distortion Elimination in the Sum of Absolute Differences for Energy-Efficient HEVC Integer Motion Estimation. In: PROC. of the 31st Symp. on Intg. Circuits and Syst. Design. Bento Golçalves, Brazil: [s.n.], 2018. (SBCCI '18).

AFONSO, V. et al. Memory-aware and high-throughput hardware design for the HEVC fractional motion estimation. In: SBCCI'15. [S.l.: s.n.], Aug. 2015. p. 1–6.

AFONSO, Vladimir et al. Hardware Implementation for the HEVC Fractional Motion Estimation Targeting Real-Time and Low-Energy. **Journal of Integrated Circuits and Systems**, v. 11, n. 2, p. 106–120, 2016.

AGOSTINI, L. V.; SILVA, I. S.; BAMPI, S. Pipelined fast 2D DCT architecture for JPEG image compression. In: SYMPOSIUM on Integrated Circuits and Systems Design. [S.l.: s.n.], 2001. p. 226–231. DOI: 10.1109/SBCCI.2001.953032.

AGOSTINI, L.; PORTO, R., et al. High throughput multitransform and multiparallelism IP for H.264/AVC video compression standard. In: ISCAS'06. [S.l.: s.n.], May 2006. p. 5418–5422. DOI: 10.1109/ISCAS.2006.1693859.

AHMED, N.; NATARAJAN, T.; RAO, K. R. Discrete Cosine Transform. **IEEE Transactions on Computers**, v. C-23, n. 1, p. 90–93, Jan. 1974.

AITKEN, Rob. **Moore's Law Ending? No Problem**. Electronic Engineering Times. 2019. Available from: <https://www.eetimes.com/author.asp?section_id=36&doc_id=1334474>. Visited on: 9 July 2019.

AKRAMULLAH, Shahriar. Video Quality Metrics. In: DIGITAL Video Concepts, Methods, and Metrics: Quality, Compression, Performance, and Power Trade-off Analysis. Berkeley, CA: Apress, 2014. p. 101–160. ISBN 978-1-4302-6713-3. DOI: 10.1007/978-1-4302-6713-3_4. Available from: <https://doi.org/10.1007/978-1-4302-6713-3_4>.

AMARAL, Lívia et al. An Energy Consumption Analysis Of Motion Estimation Algorithms Using Data Reuse In Video Coding Systems. In: PROCEEDINGS of the 2014 Symposium on Integrated Circuits and Systems Desing (SBCCI). [S.l.]: ACM, Sept. 2014.

ANTON, H.; RORRES, C. **Elementary Linear Algebra: Applications Version**. [S.l.]: John Wiley & Sons, 2010. ISBN 9780470432051.

BAES, Kurt et al. **Future of batteries**: Winner takes all? [S.l.], May 2018. Available from: <https://www.adlittle.com/sites/default/files/viewpoints/adl_future_of_batteries-min.pdf>. Visited on: 8 July 2019.

BARNEA, Daniel I.; SILVERMAN, H.F. A Class of Algorithms for Fast Digital Image Registration. **IEEE Transactions on Computers**, v. C-21, n. 2, p. 179–186, Fevereiro 1972.

BENNETT, Hugh. **Hugh's News**. [S.l.: s.n.], 2011. Available from: <http://www.hughsnews.ca/faqs/authoritative-blu-ray-disc-bd-faq/4-physical-logical-and-application-specifications4.6>. Visited on: 15 Aug. 2011.

BJØNTEGAARD, Gisle. **Calculation of average PSNR differences between RD-curves**. Austin, Texas, USA, Apr. 2001.

_____. **Improvements of the BD-PSNR model**. Berlin, Germany, July 2008.

BLASI, S.G. et al. Adaptive precision motion estimation for HEVC coding. In: PICTURE Coding Symposium (PCS), 2015. [S.l.: s.n.], May 2015. p. 144–148.

BOLAÑOS-JOJOA, J. D.; ESPINOSA-DURAN, J. M.; VELASCO-MEDINA, J. Efficient hardware design of Forward and Inverse Walsh-Hadamard transform. In: STSIVA'14. [S.l.: s.n.], Sept. 2014. p. 1–5. DOI: 10.1109/STSIVA.2014.7010174.

BONATTO, L. V. M. et al. Low-power multi-size HEVC DCT architecture proposal for QFHD video processing. In: SBCCI'17. Fortaleza, Brazil: IEEE, Aug. 2017. p. 41–46.

BONOTTO, Bruno et al. A Named-Pipe Library for Hardware Simulation. In: SIM'18. Curitiba, Brazil: SBC, May 2018.

BORKAR, Shekhar; CHIEN, Andrew A. The Future of Microprocessors. **Commun. ACM**, ACM, New York, NY, USA, v. 54, n. 5, p. 67–77, May 2011. ISSN 0001-0782. DOI: 10.1145/1941487.1941507. Available from: <http://doi.acm.org/10.1145/1941487.1941507>.

BOSSEN, F. et al. HEVC Complexity and Implementation Analysis. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1685–1696, Dezembro 2012.

BOSSEN, Frank. **Common test conditions and software reference configurations**. Shanghai, Oct. 2012.

BOVIK, Alan C. (Ed.). **Handbook of Image and Video Processing**. Second. [S.l.]: Elsevier Academic Press, 2005. ISBN 0121197921.

BRÄSCHER, A. Beims; SEIDEL, I.; GÜNTZEL, J. L. Improving the energy efficiency of a low-area SATD hardware architecture using fine grain PDE. In: SBCCI'17. [S.l.: s.n.], Aug. 2017. p. 155–161.

BRÄSCHER, André Beims; SEIDEL, Ismael; GÜNTZEL, José Luís. Improving the Energy Efficiency of a Low-Area SATD Hardware Architecture Using Fine Grain PDE. In: PROCEEDINGS of the 30th Symposium on Integrated Circuits and Systems Design (SBCCI). Fortaleza, Brazil: IEEE, Sept. 2017. DOI: 10.1145/3109984.3110009.

BROSS, Benjamin et al. Inter-Picture Prediction in HEVC. In: **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. Ed. by V. Sze, Madhukar Budagavi and Gary J. Sullivan. Switzerland: Springer International Publishing, 2014. p. 113–140. (Integrated Circuits and Systems). ISBN 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4.

BRUNIG, M.; NIEHSEN, W. Fast full-search block matching. **Circuits and Systems for Video Technology, IEEE Transactions on**, v. 11, n. 2, p. 241–247, Fevereiro 2001.

CAI, J.; PAN, W. D. Fast exhaustive-search motion estimation based on accelerated multilevel successive elimination algorithm with multiple passes. In: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. [S.l.: s.n.], Mar. 2010. p. 1190–1193. DOI: 10.1109/ICASSP.2010.5495384.

CANCELLIER, Luiz Henrique De Lorenzi. **Algoritmo de Eliminações Sucessivas em Níveis baseado na Soma das Diferenças Transformadas Absolutas**. 2016. Trabalho de Conclusão de Curso – UFSC.

CANCELLIER, Luiz Henrique; BRÄSCHER, André Beims, et al. Energy-Efficient Hadamard-Based SATD Architectures. In: SBCCI'14. New York, NY, USA: ACM, 2014. 36:1–36:6.

CANCELLIER, Luiz Henrique; SEIDEL, Ismael, et al. Exploring Optimized Hadamard Methods to Design Energy-Efficient SATD Architectures. **Journal of Integrated Circuits and Syst.**, v. 10, n. 2, p. 113–122, Aug. 2015.

CE ZHU; WEI-SONG QI; WEE SER. A new successive elimination algorithm for fast block matching in motion estimation. In: 2004 IEEE International Symposium on Circuits and Systems (ISCAS). [S.l.: s.n.], May 2004. v. 3, p. iii–733. DOI: 10.1109/ISCAS.2004.1328851.

CHAKRABARTI, I. et al. **Motion Estimation for Video Coding: Efficient Algorithms and Architectures**. [S.l.]: Springer International Publishing, 2015. (Studies in Computational Intelligence).

CHAN, Yui-Lam; SIU, Wan-Chi. Search strategy for partial distortion elimination in motion estimation. **Electronics Letters**, v. 38, n. 23, p. 1427–1428, Nov. 2002.

CHEN, C. et al. Optimized Transcoding for Large Scale Adaptive Streaming Using Playback Statistics. In: 2018 25th IEEE International Conference on Image Processing (ICIP). [S.l.: s.n.], Oct. 2018. p. 3269–3273. DOI: 10.1109/ICIP.2018.8451307.

CHEN, Ching-Yeh et al. Level C+ data reuse scheme for motion estimation with corresponding coding orders. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 16, n. 4, p. 553–558, Apr. 2006.

CHEN, K. et al. Efficient SIMD optimization of HEVC encoder over X86 processors. In: APSIPA ASC'12. [S.l.: s.n.], Dec. 2012. p. 1–4.

CHIANG, C. et al. Adaptive interpolation filter scheme in AV1. In: 2017 IEEE International Conference on Image Processing (ICIP). [S.l.: s.n.], Sept. 2017. p. 934–937. DOI: 10.1109/ICIP.2017.8296418.

CHIARIGLIONE, Leonardo. **A vision made real Past, present and future of MPEG**. Edition of 2019/05/20. [S.l.: s.n.], 2019. Available from: <http://leonardo.chiariglione.org/wp-content/uploads/2019/05/A-vision-made-real-Past-present-future-of-MPEG.pdf>.

CHIU, Man-Yau; SIU, Wan-Chi. New results on exhaustive search algorithm for motion estimation using adaptive partial distortion search and successive elimination algorithm. In: CIRCUITS and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on. [S.l.: s.n.], Maio 2006. 4 pp.–3981.

CHOU, Philip A. **Holograms are the Next Video**. Amsterdam, Netherlands: [s.n.], 2018. Keynote.

CISCO. **VNI Complete Forecast Highlights: Global - 2022 Forecast Highlights**. Ed. by Cisco Systems. [S.l.], 2018.

CLEVELAND, C.J.; MORRIS, C.G. **Dictionary of Energy: Expanded Edition**. [S.l.]: Elsevier Science, 2009. ISBN 9780080965178.

COBAN, M.Z.; MERSEREAU, R.M. A fast exhaustive search algorithm for rate-constrained motion estimation. v. 7, n. 5, p. 769–773, May 1998.

COCKSHOTT, P.; RENFREW, K. **SIMD Programming Manual for Linux and Windows**. London, United Kingdom: Springer-Verlag London, 2013. (Springer Professional Computing). ISBN 9781447138624. DOI: 10.1007/978-1-4471-3862-4.

COLWELL, R. The chip design game at the end of Moore's law. In: 2013 IEEE Hot Chips 25 Symposium (HCS). [S.l.: s.n.], Aug. 2013. p. 1–16. DOI: 10.1109/HOTCHIPS.2013.7478302.

CONSULT, Morning. **National Tracking Poll - Crosstabulation Results**. [S.l.: s.n.], Nov. 2018. Survey. Available from: <https://morningconsult.com/wp-content/uploads/2018/11/181114_crosstabs_BRANDS_Adults_v1_AP.pdf>. Visited on: 15 June 2019.

CONTI, M. et al. The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis. **IEEE Communications Surveys Tutorials**, v. 20, n. 4, p. 2658–2713, Fourthquarter 2018. ISSN 1553-877X. DOI: 10.1109/COMST.2018.2843533.

CORREA, G. et al. Performance and Computational Complexity Assessment of High-Efficiency Video Encoders. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1899–1909, Dezembro 2012.

CORRÊA, G. et al. **Complexity-Aware High Efficiency Video Coding**. [S.l.]: Springer International Publishing, 2015. ISBN 9783319257785. DOI: 10.1007/978-3-319-25778-5.

CUBITT, S. **The Practice of Light: A Genealogy of Visual Technologies from Prints to Pixels**. [S.l.]: MIT Press, 2014. (Leonardo (MIT Press)). ISBN 9780262027656.

CULLEN, C.G. **Matrices and Linear Transformations: Second Edition**. Second. New York, USA: Dover Publications, 2012. (Dover Books on Mathematics). ISBN 9780486663289.

CUNNINGHAM, D. **A Logical Introduction to Proof**. [S.l.]: Springer New York, 2012. ISBN 9781461436300.

DEGUCHI, Yuichiro; TREE, John. **Bypass using sum of absolute transformed differences value (SATD) in a video coding process**. [S.l.]: Google Patents, Jan. 2001. US Patent App. 29/126,679.

DELAGI, G. Harnessing technology to advance the next-generation mobile user-experience. In: SOLID-STATE Circuits Conf. Digest of Technical Papers (ISSCC), 2010 IEEE International. [S.l.: s.n.], Fevereiro 2010. p. 18–24.

DENNARD, R. H. et al. Design of ion-implanted MOSFET's with very small physical dimensions. **IEEE Journal of Solid-State Circuits**, v. 9, n. 5, p. 256–268, Oct. 1974. ISSN 0018-9200. DOI: 10.1109/JSSC.1974.1050511.

DIAMANTARAS, K. I.; STRINTZIS, M. G. Optimal transform coding in the presence of quantization noise. **IEEE Transactions on Image Processing**, v. 8, n. 11, p. 1508–1515, Nov. 1999. ISSN 1057-7149. DOI: 10.1109/83.799879.

DINIZ, C. M. et al. High-throughput interpolation hardware architecture with coarse-grained reconfigurable datapaths for HEVC. In: ICIP'13. [S.l.: s.n.], Sept. 2013. p. 2091–2095.

DINIZ, Claudio M et al. A reconfigurable hardware architecture for fractional pixel interpolation in high efficiency video coding. IEEE, v. 34, n. 2, p. 238–251, 2015.

DIVYA, K.C.; ØSTERGAARD, Jacob. Battery energy storage technology for power systems: An overview. **Electric Power Systems Research**, v. 79, n. 4, p. 511–520, 2009. ISSN 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2008.09.017. Available from: <http://www.sciencedirect.com/science/article/pii/S0378779608002642>.

DOMAŃSKI, Marek. **Approximate video bitrate estimation for television services**. Warsaw, Poland, 2015.

DONG, L.; PAN, Z. Fast motion estimation algorithm using multilevel distortion search in Walsh–Hadamard domain. **IET Image Processing**, v. 11, n. 1, p. 22–30, 2017. DOI: 10.1049/iet-ipr.2016.0453.

EADLINE, Douglas. **May's Law and Parallel Software**. 2011. Available from: <http://www.linux-mag.com/id/8422/>. Visited on: Mar. 2017.

FRÖJDH, PER; NORKIN, ANDREY; SJÖBERG, RICKARD. Next generation video compression. **ERICSSON REVIEW**, 2013.

FU, T. et al. Fast CU Partitioning Algorithm for H.266/VVC Intra-Frame Coding. In: 2019 IEEE International Conference on Multimedia and Expo (ICME). [S.l.: s.n.], July 2019. p. 55–60. DOI: 10.1109/ICME.2019.00018.

FULLER, S. H.; MILLETT, L. I. Computing Performance: Game Over or Next Level? **Computer**, v. 44, n. 1, p. 31–38, Jan. 2011. ISSN 0018-9162. DOI: 10.1109/MC.2011.15.

GAO, X.Q.; DUANMU, C.J.; ZOU, C.R. A multilevel successive elimination algorithm for block matching motion estimation. **IEEE Transactions on Image Processing**, v. 9, n. 3, p. 501–504, 2000.

GEADAH, Y. A.; CORINTHIOS, M. J. G. Natural, Dyadic, and Sequency Order Algorithms and Processors for the Walsh-Hadamard Transform. **IEEE Trans. Comput.**, IEEE Computer Society, Washington, DC, USA, v. 26, n. 5, p. 435–442, May 1977. ISSN 0018-9340. DOI: 10.1109/TC.1977.1674860. Available from: <https://doi.org/10.1109/TC.1977.1674860>.

GHANBARI, Mohammed. **Standard Codecs: Image compression to advanced video coding**. London, UK: Iet, 2003. v. 49.

GOEBEL, J. et al. An HEVC multi-size DCT hardware with constant throughput and supporting heterogeneous CUs. In: ISCAS'16. [S.l.: s.n.], May 2016. p. 2202–2205. DOI: 10.1109/ISCAS.2016.7539019.

GOLUB, G.H.; VAN LOAN, C.F. **Matrix Computations**. [S.l.]: Johns Hopkins University Press, 1996. (Johns Hopkins Studies in the Mathematical Sciences). ISBN 9780801854149.

GONÇALVES, P. et al. Octagonal-Axis Raster Pattern for Improved Test Zone Search Motion Estimation. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). [S.l.: s.n.], Apr. 2018. p. 1763–1767. DOI: 10.1109/ICASSP.2018.8462580.

GONZÁLEZ-DE-SUSO, J. L. et al. Improved Method to Select the Lagrange Multiplier for Rate-Distortion Based Motion Estimation in Video Coding. v. 24, n. 3, p. 452–464, Mar. 2014. ISSN 1051-8215. DOI: 10.1109/TCSVT.2013.2276857.

GOUGH, Brian. **GNU Scientific Library Reference Manual - Third Edition**. 3rd. [S.l.]: Network Theory Ltd., 2009. ISBN 0954612078, 9780954612078.

GROIS, D. et al. Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders. In: PCS'13. [S.l.: s.n.], Dec. 2013. p. 394–397. DOI: 10.1109/PCS.2013.6737766.

GROUP, Ultra Video. **Kvazaar**. [S.l.: s.n.], 2016. Available from: <https://github.com/ultravideo/kvazaar.wiki.git>. Visited on: Oct. 2016.

GSMA. **The Mobile Economy 2019**. London, United Kingdom, Feb. 2019. Available from: <https://www.gsmaintelligence.com/research/?file=b9a6e6202ee1d5f787cfebb95d3639c5&download>. Visited on: July 2019.

GUO, Z. et al. An optimized MC interpolation architecture for HEVC. In: ICASSP'12. [S.l.: s.n.], Mar. 2012. p. 1117–1120. DOI: 10.1109/ICASSP.2012.6288083.

HABIBI, A. Comparison of nth-Order DPCM Encoder With Linear Transformations and Block Quantization Techniques. **IEEE Transactions on Communication Technology**, v. 19, n. 6, p. 948–956, Dec. 1971. ISSN 0018-9332. DOI: 10.1109/TCOM.1971.1090776.

EL-HADEDY, M. et al. Performance and area efficient transpose memory architecture for high throughput adaptive signal processing systems. In: NASA/ESA AHS'10. [S.l.: s.n.], June 2010. p. 113–120. DOI: 10.1109/AHS.2010.5546272.

HASHIMOTO, R. et al. VLSI Architecture of 1.264 Block Size Decision based on Rate-Distortion Optimization. In: 2006 International Symposium on Intelligent Signal Processing and Communications. [S.l.: s.n.], Dec. 2006. p. 618–621. DOI: 10.1109/ISPACS.2006.364732.

HASIB, Abdullah Al. **Energy Efficient Computing on Multi-core Processors**: Vectorization and Compression Techniques. May 2018. PhD thesis – Norwegian University of Science and Technology, Trondheim, Norway. ISBN 978-82-326-3077-6.

HE, G. et al. High-Throughput Power-Efficient VLSI Architecture of Fractional Motion Estimation for Ultra-HD HEVC Video Encoding. v. 23, n. 12, p. 3138–3142, Dec. 2015. ISSN 1063-8210. DOI: 10.1109/TVLSI.2014.2386897.

_____. High-Throughput Power-Efficient VLSI Architecture of fractional motion estimation for ultra-HD HEVC Video Encoding. v. 23, n. 12, p. 3138–3142, Dec. 2015. ISSN 1063-8210. DOI: 10.1109/TVLSI.2014.2386897.

HORN, R.A.; JOHNSON, C.R. **Matrix Analysis**. Second. New York, USA: Cambridge University Press, 2012. ISBN 9781139788885.

HU, L. et al. A hardware-friendly hierarchical HEVC motion estimation algorithm for UHD applications. In: ISCAS'17. [S.l.: s.n.], May 2017. p. 1–4. DOI: 10.1109/ISCAS.2017.8050322.

IHRKE, I.; RESTREPO, J.; MIGNARD-DEBISE, L. Principles of Light Field Imaging: Briefly revisiting 25 years of research. **IEEE Signal Processing Magazine**, v. 33, n. 5, p. 59–69, Sept. 2016. ISSN 1053-5888. DOI: 10.1109/MSP.2016.2582220.

ISO/IEC. **ISO/IEC 10918-1:1994: Information technology – Digital compression and coding of continuous-tone still images: Requirements and guidelines**. [S.l.], Feb. 1994. p. 182.

_____. **Requirements for a Future Video Coding Standard v1**. Warsaw, Poland, June 2015.

ITU. **Recommendation T.81: Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines**. [S.l.], 1992.

ITU-T. **H.264 Corrigendum 1**. [S.l.: s.n.], Jan. 2009.

_____. **Parameter values for the HDTV standards for production and international programme exchange**. Genebra, Maio 2008.

_____. **Parameter values for ultra-high definition television systems for production and international programme exchange**. Genebra, Agosto 2012.

_____. **Recommendation ITU-T H.265: High efficiency video coding**. Genebra, 2013.

_____. **Recommendation ITU-T P.910: Subjective video quality assessment methods for multimedia applications**. [S.l.], 2008.

_____. **Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios**. Genebra, Mar. 2011.

_____. **User requirements for objective perceptual video quality measurements in digital cable television**. [S.l.], May 2000. Available from: <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-J.143-200005-I!!PDF-E&type=items>.

JCT-VC. **HEVC Test Model**. [S.l.: s.n.], 2013. Available from: <http://hevc.hhi.fraunhofer.de/>. Visited on: Sept. 2015.

JERVAN, G. et al. An Approach to System-level Design for Test. In: REORDA, M.S.; PENG, Z.; VIOLANTE, M. (Eds.). **System-level Test and Validation of Hardware/Software Systems**. [S.l.]: Springer, 2005. (Advanced microelectronics). chap. 8, p. 121–149. ISBN 9781852338992.

JOU, F.D. **Method for fast SATD estimation**. [S.l.], Aug. 2007. US Patent App. 11/360,552. Available from: <https://www.google.com/patents/US20070198622>.

KALALI, E.; HAMZAOGLU, I. A low energy HEVC sub-pixel interpolation hardware. In: ICIP'14. [S.l.: s.n.], Oct. 2014. p. 1218–1222.

KALVA, H. The H.264 Video Coding Standard. **IEEE MultiMedia**, IEEE Computer Society, Los Alamitos, CA, USA, v. 13, n. 04, p. 86–90, Oct. 2006. ISSN 1070-986X. DOI: 10.1109/MMUL.2006.93.

KALVA, H.; FURHT, B. Complexity Estimation of the H.264 Coded Video Bitstreams. **The Computer Journal**, OUP, v. 48, n. 5, p. 504–513, Jan. 2005. ISSN 0010-4620. DOI: 10.1093/comjnl/bxh112.

KAMACI, N.; ALTUNBASAK, Y. Performance comparison of the emerging H.264 video coding standard with the existing standards. In: PROCEEDINGS of the 2003 International Conference on Multimedia and Expo, 2003. ICME '03. [S.l.]: IEEE, 2003. v. 1, i-345–8 vol.1.

KAMAT, R.K. et al. **Harnessing VLSI System Design with EDA Tools**. [S.l.]: Springer Netherlands, 2011. ISBN 9789400718647.

KARCZEWICZ, Marta; ALSHINA, Elena. **JVET AHG report: Tool evaluation (AHG1)**. Macao, China, Oct. 2017.

KARWOWSKI, Damian et al. 20 Years of Progress in Video Compression – from MPEG-1 to MPEG-H HEVC. General View on the Path of Video Coding Development. In: _____. **Image Processing and Communications Challenges 8**. Cham: Springer International Publishing, 2017. p. 3–15. ISBN 978-3-319-47274-4.

KIM, J.; PARK, T. A novel VLSI architecture for full-search variable block-size motion estimation. v. 55, n. 2, p. 728–733, May 2009.

KIM, T. S.; RHEE, C. E., et al. Fast Integer Motion Estimation with Bottom-up Motion Vector Prediction for an HEVC Encoder. **IEEE Trans. on Circuits and Syst. for Video Technol.**, p. 1–24, Oct. 2017. ISSN 1051-8215. DOI: 10.1109/TCSVT.2017.2759245.

KRISHNAN, Rathish. **Method of reducing computations in intra-prediction and mode decision processes in a digital video encoder**. [S.l.]: Google Patents, Apr. 2011. US Patent 7,929,608.

KUHN, Peter M. Fast MPEG-4 Motion Estimation: Processor Based and Flexible VLSI Implementations. **The Journal of VLSI Signal Processing**, Springer Netherlands, v. 23, p. 67–92, 1 1999.

KUMAR, Rajender; KUMAR, Krishan; PANDIT, Amit Kant. Performance and Complexity Analysis of Motion Estimation Using Multiple Constraints in Video Compression. In: _____. **ICICCT 2019 – System Reliability, Quality Control, Safety, Maintenance and Management**. Singapore: Springer Singapore, 2019. p. 698–706.

LAHTI, S. et al. Are We There Yet? A Study on the State of High-level Synthesis. **IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.**, p. 1–1, 2018. ISSN 0278-0070. DOI: 10.1109/TCAD.2018.2834439.

LAINEMA, J. et al. Intra Coding of the HEVC Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1792–1801, Dezembro 2012.

LARBIER, Pierre. Using 10-bit AVC/H.264 Encoding with 4:2:2 for Broadcast Contribution. In: available from: <http://extranet.ateme.com/download.php?file=1114>. Visited on: 12 June 2011.

LEE, Do Kyung; PARK, Miso; JEONG, Jechang. Fast intra coding by using RD cost candidate elimination for high efficiency video coding. English. In: _____. **World Congress on Engineering and Computer Science, WCECS 2014**. [S.l.]: Newswood Limited, Jan. 2014. v. 1, p. 512–515. ISBN 9789881925275.

LEMMETTI, A. et al. AVX2-optimized Kvazaar HEVC intra encoder. In: ICIP. [S.l.: s.n.], Sept. 2016.

LI, W.; SALARI, E. Successive elimination algorithm for motion estimation. **IEEE Transactions on Image Processing**, v. 4, n. 1, p. 105–107, Jan. 1995.

LI, Xiang; WIEN, M.; OHM, J.-R. Rate-complexity-distortion evaluation for hybrid video coding. In: MULTIMEDIA and Expo (ICME), 2010 IEEE International Conference on. [S.l.: s.n.], July 2010. p. 685–690.

LIAO, Weihang; CHEN, Zhenzhong. A fast CU partition and mode decision algorithm for HEVC intra coding. **Signal Processing: Image Communication**, v. 67, p. 140–148, 2018. ISSN 0923-5965. DOI: https://doi.org/10.1016/j.image.2018.06.003. Available from: <http://www.sciencedirect.com/science/article/pii/S0923596518305897>.

LIAO, Yabin; SODANO, Henry A. Model of a single mode energy harvester and properties for optimal power generation. **Smart Materials and Structures**, IOP Publishing, v. 17, n. 6, p. 1–14, Nov. 2008. DOI: 10.1088/0964-1726/17/6/065026.

LIDDLE, Andrew. **An Introduction to Modern Cosmology**. 3. ed. [S.l.]: Wiley, 2015. ISBN 9781118502099.

LIN, Yao-Chung et al. **Geo-popularity assisted optimized transcoding for large scale adaptive streaming**. v. 10752. San Diego, California, United States: [s.n.], 2018. DOI: 10.1117/12.2322191. Available from: <https://doi.org/10.1117/12.2322191>.

LIU, S.; WEI, S.; LAI, S. Fast Optimal Motion Estimation Based on Gradient-Based Adaptive Multilevel Successive Elimination. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 18, n. 2, p. 263–267, Feb. 2008. DOI: 10.1109/TCSVT.2007.913973.

LIU, Tsu-Ming; LEE, Chen-Yi. Design of an H.264/AVC Decoder with Memory Hierarchy and Line-Pixel-Lookahead. English. **Journal of Signal Processing Systems**, Springer US, v. 50, n. 1, p. 69–80, 2008.

LIU, Z.; ZHOU, J., et al. Register Length Analysis and VLSI Optimization of VBS Hadamard transform in H.264/AVC. **IEEE Trans. Circuits Syst. Video Technol.**, v. 21, n. 5, p. 601–610, May 2011.

LOPES, Alba Sandyra Bezerra; SILVA, Ivan Saraiva; AGOSTINI, Luciano Volcan. A Memory Hierarchy Model Based on Data Reuse for Full-search Motion Estimation on High-definition Digital Videos. **International Journal of Reconfigurable Computing**, Hindawi Publishing Corp., New York, NY, United States, v. 2012, 2:2–2:2, Jan. 2012. ISSN 1687-7195.

MAHMOOD, Arif. **Computation Elimination Algorithms for Correlation Based Fast Template Matching**. 2011. PhD thesis – Lahore University of Management Sciences.

MALVAR, H.S. et al. Low-complexity transform and quantization in H.264/AVC. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 13, n. 7, p. 598–603, July 2003.

MERIKOSKI, Jorma Kaarlo. On the trace and the sum of elements of a matrix. **Linear Algebra and its Applications**, v. 60, p. 177–185, 1984. ISSN 0024-3795. DOI: http://dx.doi.org/10.1016/0024-3795(84)90078-8. Available from: <http://www.sciencedirect.com/science/article/pii/0024379584900788>.

MEURER, Aaron et al. SymPy: symbolic computing in Python. **PeerJ Computer Science**, v. 3, e103, Jan. 2017. ISSN 2376-5992. DOI: 10.7717/peerj-cs.103.

MISSEMER, Antoine. William Stanley Jevons' The Coal Question (1865), beyond the rebound effect. **Ecological Economics**, v. 82, p. 97–103, 2012. ISSN 0921-8009. DOI: https://doi.org/10.1016/j.ecolecon.2012.07.010. Available from: <http://www.sciencedirect.com/science/article/pii/S0921800912002741>.

MOHAMED, K.S. **IP Cores Design from Specifications to Production: Modeling, Verification, Optimization, and Protection**. [S.l.]: Springer International Publishing, 2015. (Analog Circuits and Signal Processing). ISBN 9783319220352.

MONTAÑA, R.C. **Portable Moving Images: A Media History of Storage Formats**. [S.l.]: De Gruyter, 2017. ISBN 9783110553925.

MONTEIRO, M.; SEIDEL, I.; GÜNTZEL, J. L. On the calculation reuse in Hadamard-based SATD. In: LASCAS'18. Puerto Vallarta, Mexico: IEEE, Feb. 2018. p. 1–4. DOI: 10.1109/LASCAS.2018.8399925.

MONTEIRO, Marcio. **Arquitetura energeticamente eficiente para a FME com SAD combinada com uma nova abordagem para o cálculo de resíduos**. 2019. Dissertação de mestrado – UFSC.

MOONS, B.; VERHELST, M. An Energy-Efficient Precision-Scalable ConvNet Processor in 40-nm CMOS. **IEEE Journal of Solid-State Circuits**, v. 52, n. 4, p. 903–914, Apr. 2017. ISSN 0018-9200. DOI: 10.1109/JSSC.2016.2636225.

MOORE, Gordon E et al. Cramming more components onto integrated circuits. **Electronics**, McGraw-Hill, New York, NY, USA, v. 38, n. 8, p. 114–, Apr. 1965.

_____. Progress in digital integrated electronics. In: ELECTRON Devices Meeting. [S.l.: s.n.], 1975. v. 21, p. 11–13.

NING, T. H. A perspective on the theory of MOSFET scaling and its impact. **IEEE Solid-State Circuits Society Newsletter**, v. 12, n. 1, p. 27–30, Winter 2007. ISSN 1098-4232. DOI: 10.1109/N-SSC.2007.4785538.

OHM, J. et al. Comparison of the Coding Efficiency of Video Coding Standards – Including High Efficiency Video Coding (HEVC). **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1669–1684, Dezembro 2012.

OHM, Jens-Rainer; SULLIVAN, Gary J. **Versatile Video Coding – towards the next generation of video compression**. [S.l.: s.n.], 2018. Invited Talk. Available from: <http://www.pcs2018.com/uploads/1/1/0/0/110091239/pcs2018_vvc_overview_06_26.pdf>. Visited on: June 2019.

ORTEGA, A.; RAMCHANDRAN, K. Rate-distortion methods for image and video compression. v. 15, n. 6, p. 23–50, Nov. 1998.

ORTEGA, Antonio. Video coding: Predictions are hard to make, especially about the future. In: CITESEER. PROC. Image Media Processing Symposium. [S.l.: s.n.], 2007.

OSTERMANN, J. et al. Video coding with H.264/AVC: tools, performance, and complexity. **IEEE Circuits and Systems Magazine**, v. 4, n. 1, p. 7–28, 2004. ISSN 1531-636X. DOI: 10.1109/MCAS.2004.1286980.

PARADISO, J. A.; STARNER, T. Energy scavenging for mobile and wireless electronics. **IEEE Pervasive Computing**, v. 4, n. 1, p. 18–27, Jan. 2005. ISSN 1536-1268. DOI: 10.1109/MPRV.2005.9.

PASTUSZAK, Grzegorz; TROCHIMIUK, Maciej. Architecture design of the high-throughput compensator and interpolator for the H.265/HEVC encoder. **Journal of Real-Time Image Processing**, v. 11, n. 4, p. 663–673, Apr. 2016.

PATTERSON, D.A.; HENNESSY, J.L. **Computer Organization and Design RISC-V Edition: The Hardware Software Interface**. [S.l.]: Elsevier Science, 2017. (The Morgan Kaufmann Series in Computer Architecture and Design). ISBN 9780128122761.

PATTERSON, Jason Robert Carey. **Video Encoding Settings for H.264 Excellence**. Surfers Paradise, Australia, Abril 2012. Available from: <http://www.lighterra.com/papers/videoencodingh264/>. Visited on: June 2015.

PATTERSON, Nick; RICHTER, Daniel J, et al. Genetic evidence for complex speciation of humans and chimpanzees. **Nature**, Nature Publishing Group, v. 441, n. 7097, p. 1103–1108, 2006. DOI: 10.1038/nature04789.

PEREIRA, Fabio et al. H.264 8x8 Inverse Transform Architecture Optimization. In: PROCEEDINGS of the 24th Edition of the Great Lakes Symposium on VLSI. Houston, Texas, USA: ACM, 2014. (GLSVLSI '14), p. 83–84.

PIAO, Yinji; MIN, Junghye; CHEN, Jianle. **Encoder improvement of unified intra prediction**. Guangzhou, CN, Oct. 2010.

PILLOT, Christophe. **The rechargeable battery market 2017–2025**. [S.l.: s.n.], May 2018. Slides. Available from: <https://www.rechargebatteries.org/wp-content/uploads/2019/05/Avicenne_The-Rechargeable-Battery-Market-2017-2025.pdf>. Visited on: July 2019.

PLACKE, Tobias et al. Lithium ion, lithium metal, and alternative rechargeable battery technologies: the odyssey for high energy density. **Journal of Solid State Electrochemistry**, v. 21, n. 7, p. 1939–1964, July 2017. ISSN 1433-0768. DOI: 10.1007/s10008-017-3610-7.

POLIMENI, J.M. **The Jevons Paradox and the Myth of Resource Efficiency Improvements**. [S.l.]: Taylor & Francis, 2012. (Earthscan research edition). ISBN 9781849773102.

PORTO, M.S. et al. Design space exploration on the H.264 4x4 Hadamard transform. In: NORCHIP'05. [S.l.: s.n.], Nov. 2005. p. 188–191.

POYNTON, Charles. **Digital Video and HD: Algorithms and Interfaces**. Second. [S.l.]: Elsevier Science, 2012. (The Morgan Kaufmann Series in Computer Graphics). ISBN 9780123919267.

QUALCOMM. **Qualcomm Snapdragon 835 mobile platform**. [S.l.: s.n.], 2016. Available from: <https://www.qualcomm.com/media/documents/files/snapdragon-835-mobile-platform-product-brief.pdf>. Visited on: Oct. 2019.

RABAEY, Jan. **Low Power Design Essentials**. 1. ed. [S.l.]: Springer US, 2009. (Integrated Circuits and Systems).

REDI, Judith A. et al. Digital Video Image Quality and Perceptual Coding. In: ed. by H.R. Wu and K.R. Rao. [S.l.]: CRC Press, Nov. 2005. How Passive Image Viewers Became Active Multimedia Users: New Trends and Recent Advances in Subjective Assessment of Quality of Experience, p. 72.

REORDA, M.S.; PENG, Z.; VIOLANTE, M. Test Generation: A Heuristic Approach. In: _____. **System-level Test and Validation of Hardware/Software Systems**. Ed. by M.S. Reorda, Z. Peng and M. Violante. [S.l.]: Springer, 2005. (Advanced microelectronics). chap. 4, p. 47–65. ISBN 9781852338992.

REQUIREMENTS. **Requirements for a New Video Coding Standard**. Macau, CN, Oct. 2018.

RICHARDSON, Iain E. **H. 264 and MPEG-4 video compression: video coding for next-generation multimedia**. [S.l.]: John Wiley & Sons Inc, 2003.

_____. **The H.264 Advanced Video Compression Standard, Second Edition**. [S.l.]: John Wiley & Sons Ltd, 2010.

_____. **Video codec design: developing image and video compression systems**. Chichester, England: John Wiley and Sons, 2002. ISBN 0-470-84837-5.

RITHE, R.; CHENG, C. C.; CHANDRAKASAN, A. P. Quad Full-HD Transform Engine for Dual-Standard Low-Power Video Coding. **IEEE J. Solid-State Circuits**, v. 47, n. 11, p. 2724–2736, Nov. 2012. ISSN 0018-9200. DOI: 10.1109/JSSC.2012.2211694.

ROSEWARNE, C. et al. **High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description**: Update 5. San Diego, USA, Feb. 2016.

ROSEWARNE, C. et al. **High Efficiency Video Coding (HEVC) Test Model 16 (HM 16) Improved Encoder Description**: Update 9. Torino, Italy, 2017.

S. LIVRAMENTO, Vinícius dos et al. Evaluating the Impact of Architectural Decisions on the Energy Efficiency of FDCT/IDCT Configurable IP Cores. **Journal of Integrated Circuits and Systems**, v. 7, n. 1, p. 23–36, Mar. 2012.

SAMSUNG. **Exynos 7880**: Versatile talents for superlative mobile experience. [S.l.: s.n.], 2016. Available from: <https://www.samsung.com/semiconductor/global.semi.static/minisite/exynos/file/solution/MobileProcessor-7-Series-7880.pdf>. Visited on: Oct. 2019.

_____. **Mobile Processor**: Exynos 990. [S.l.: s.n.], 2019. Available from: <https://www.samsung.com/semiconductor/global.semi.static/minisite/exynos/file/solution/MobileProcessor-990.pdf>. Visited on: 16 Dec. 2019.

SAPATNEKAR, Sachin. Retiming. In: TIMING. [S.l.]: Springer, 2004. p. 306.

SAPONARA, Sergio et al. The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis. In: PROCEEDING of the IEEE Packet Video Workshop (PV'03. [S.l.: s.n.], 2003. p. 98–109.

SCHAFER, R.; SIKORA, T. Digital video coding standards and their role in video communications. **Proc. IEEE**, v. 83, n. 6, p. 907–924, June 1995. ISSN 0018-9219. DOI: 10.1109/5.387092.

SCHECHTER, E. **Handbook of Analysis and Its Foundations**. [S.l.]: Elsevier Science, 1996.

SCHWARZ, Heiko; MARPE, Detlev; WIEGAND, Thomas. **Hierarchical B pictures**. Warsaw, Poland, 2005.

SCHWARZ, S.; PREDA, M., et al. Emerging MPEG Standards for Point Cloud Compression. **IEEE Journal on Emerging and Selected Topics in Circuits and Systems**, IEEE, v. 9, n. 1, p. 133–148, Mar. 2019. ISSN 2156-3357. DOI: 10.1109/JETCAS.2018.2885981.

SEIDEL, I.; CANCELLIER, L. H., et al. Rate-constrained successive elimination of Hadamard-based SATDs. In: 2016 IEEE International Conference on Image Processing (ICIP). [S.l.: s.n.], Sept. 2016. p. 2395–2399. DOI: 10.1109/ICIP.2016.7532788.

SEIDEL, I.; GÜNTZEL, J. L.; AGOSTINI, L. Coarse grain partial distortion elimination for Hadamard ME in HEVC. In: ICECS'16. [S.l.: s.n.], Dec. 2016. p. 704–707. DOI: 10.1109/ICECS.2016.7841299.

SEIDEL, I.; MONTEIRO, M., et al. Energy-Efficient Hadamard-Based SATD Hardware Architectures Through Calculation Reuse. **IEEE Transactions on Circuits and Systems I: Regular Papers**, v. 66, n. 6, p. 2102–2115, June 2019. DOI: 10.1109/TCSI.2019.2900004.

SEIDEL, Ismael; BRÄSCHER, André Beims; GÜNTZEL, José Luís. Combining Pel Decimation with Partial Distortion Elimination to Increase SAD Energy Efficiency. In: PROCEEDINGS of the 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS). Salvador, Brazil: IEEE, 2015.

SEIDEL, Ismael; BRÄSCHER, André Beims; GÜNTZEL, José Luís; AGOSTINI, Luciano Volcan. Energy-Efficient SATD for Beyond HEVC. In: 2016 IEEE International Symposium on Circuits and Systems. Montreal, Canada: IEEE, May 2016.

SEIDEL, Ismael; FILHO, Vanio Rodrigues, et al. Coding- and Energy-Efficient FME Hardware Design. In: 2018 IEEE International Symposium on Circuits and Systems. Firenze, Italy: IEEE, 2018. DOI: 10.1109/ISCAS.2018.8351114.

SERGE PARNOVSKY, Aleksei Parnowski. **How the Universe Works: Introduction to Modern Cosmology**. [S.l.]: World Scientific Publishing, 2018. ISBN 9789813234949.

SHARMAN, Karl; SUEHRING, Karsten. **Common test conditions**. Geneva, Jan. 2017.

SHI, Yun Q.; SUN, Huifang. **Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards**. second. [S.l.]: CRC Press/Taylor & Francis, 2008. (Image Processing Series). ISBN 9780849334917.

SILVEIRA, B. et al. SATD hardware architecture based on 8x8 Hadamard Transform for HEVC encoder. In: ICECS. [S.l.: s.n.], Dec. 2015.

SILVEIRA, B.; ABREU, B., et al. Using adder and subtractor compressors to sum of absolute transformed differences architecture for low-power video encoding. In: ICECS'17. [S.l.: s.n.], Dec. 2017. p. 409–493. DOI: 10.1109/ICECS.2017.8292076.

SILVEIRA, B.; FERREIRA, R., et al. Low power SATD architecture employing multiple sizes Hadamard transforms and adder compressors. In: NEWCAS'17. [S.l.: s.n.], June 2017. p. 277–280. DOI: 10.1109/NEWCAS.2017.8010159.

SILVEIRA, B.; PAIM, G., et al. Power-Efficient Sum of Absolute Differences Hardware Architecture Using Adder Compressors for Integer Motion Estimation Design. **IEEE Trans. Circuits Syst. I, Reg. Papers**, v. 64, n. 12, p. 3126–3137, Dec. 2017. ISSN 1549-8328. DOI: 10.1109/TCSI.2017.2728802.

SILVEIRA, D.; PORTO, M.; BAMPI, S. Performance and energy consumption analysis of the X265 video encoder. In: 2017 25th European Signal Processing Conference (EUSIPCO). [S.l.: s.n.], Aug. 2017. p. 1519–1523. DOI: 10.23919/EUSIPCO.2017.8081463.

SINANGIL, M.E. et al. Cost and Coding Efficient Motion Estimation Design Considerations for High Efficiency Video Coding (HEVC) Standard. **IEEE Journal of Selected Topics in Signal Processing**, v. 7, n. 6, p. 1017–1028, Dezembro 2013.

SINGH, K.; AHAMED, S. R. Low Power Motion Estimation Algorithm and Architecture of HEVC/H.265 for Consumer Applications. **IEEE Transactions on Consumer Electronics**, p. 1–1, 2018. ISSN 0098-3063. DOI: 10.1109/TCE.2018.2867823.

SIPSER, M. **Introduction to the Theory of Computation**. Third. [S.l.]: Cengage Learning, 2012. p. 1–504. ISBN 9781285401065.

SMITH, G.V.; PARR, R.L. **Intellectual Property: Licensing and Joint Venture Profit Strategies**. [S.l.]: Wiley, 2004. (Intellectual property series). ISBN 9780471669937.

SOARES, L. B. et al. A novel pruned-based algorithm for energy-efficient SATD operation in the HEVC coding. In: SBCCI'16. [S.l.: s.n.], Aug. 2016. p. 1–6.

SOLE, Joel et al. **Performance comparison of video coding standards: an adaptive streaming perspective**. Netflix Technology Blog. Dec. 2018. Available from: <https://medium.com/netflix-techblog/performance-comparison-of-video-coding-standards-an-adaptive-streaming-perspective-d45d0183ca95>. Visited on: June 2019.

SONG, Y. et al. Enhanced Strict Multilevel Successive Elimination Algorithm for Fast Motion Estimation. In: 2007 IEEE International Symposium on Circuits and Systems. [S.l.: s.n.], May 2007. p. 3659–3662. DOI: 10.1109/ISCAS.2007.378546.

_____. Lossy Strict Multilevel Successive Elimination Algorithm for Fast Motion Estimation. In: 2006 International Symposium on Intelligent Signal Processing and Communications. [S.l.: s.n.], Dec. 2006. p. 431–434. DOI: 10.1109/ISPACS.2006.364691.

STANKOWSKI, J. et al. Rate-distortion optimized quantization in HEVC: Performance limitations. In: 2015 Picture Coding Symposium (PCS). [S.l.: s.n.], May 2015. p. 85–89. DOI: 10.1109/PCS.2015.7170052.

STRAUBEL, Jeffrey Brian. **Energy Storage, EV's and the Grid**. [S.l.: s.n.], 2015. Slides. Available from: <https://www.eia.gov/conference/2015/pdf/presentations/straubel.pdf>. Visited on: July 2019.

SULLIVAN, G.J.; OHM, J., et al. Overview of the High Efficiency Video Coding (HEVC) Standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649–1668, Dezembro 2012.

SULLIVAN, G.J.; WIEGAND, T. Rate-distortion optimization for video compression. **Signal Processing Magazine, IEEE**, v. 15, n. 6, p. 74–90, Nov. 1998.

SULLIVAN, Gary J. Introduction. In: SZE, V.; BUDAGAVI, Madhukar; SULLIVAN, Gary J. (Eds.). **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. Switzerland: Springer International Publishing, 2014. (Integrated Circuits and Systems). chap. 1, p. 1–12. ISBN 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4.

_____. **Overview of International Video Coding Standards (preceding H.264/AVC)**. [S.l.: s.n.], 2005. Workshop. Available from: <https://www.itu.int/ITU-T/worksem/vica/docs/presentations/S0_P2_Sullivan.pdf>. Visited on: June 2019.

SULLIVAN, Gary; OHM, Jens-Rainer. **Meeting Report of the 12th meeting of the Joint Video Experts Team (JVET), Macao, CN, 3–12 October 2018**. Macau, CN, Oct. 2018.

SYNOPSYS. **Synopsys Design Compiler, V. L-2016.03-SP1.** [S.l.], 2016.

_____. **Synopsys Design Compiler, Version M-2016.SP4.** [S.l.: s.n.], 2015.

TABATABAI, Ali et al. Compression Performance Analysis in HEVC. In: SZE, V.; BUDAGAVI, Madhukar; SULLIVAN, Gary J. (Eds.). **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. Switzerland: Springer International Publishing, 2014. (Integrated Circuits and Systems). chap. 9, p. 275–302. ISBN 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4.

TAE GYOUNG AHN; YONG HO MOON; JAE HO KIM. Fast full-search motion estimation based on multilevel successive elimination algorithm. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 14, n. 11, p. 1265–1269, Nov. 2004. DOI: 10.1109/TCSVT.2004.835146.

TANG, M. et al. Hadamard Transform Based Optimized HEVC Video Coding. **IEEE Trans. on Circuits and Syst. for Video Technol.**, p. 1–1, 2018. ISSN 1051-8215. DOI: 10.1109/TCSVT.2018.2810324.

TAYLOR, B. **No More Suffering Fools**. [S.l.]: Lulu.com, 2014. ISBN 9781304624796.

THANGAVELU, Madhu; CHAU, Alain. Surrogate Astronaut Robotic Avatars: Co-Robotics for Safe, Economic Space Operations. In: AIAA SPACE 2013 CONFERENCE AND EXPOSITION, p. 1–35. ISBN 978-1-62410-239-4. DOI: 10.2514/6.2013-5394.

TROCHIMIUK, Maciej. Simplifications in inter-frame prediction in the H.265/HEVC encoder. **Proc. SPIE**, v. 9662, 96621z-96621z–9, 2015.

TROCHIMIUK, Maciej; ABRAMOWSKI, Andrzej. Hardware-oriented simplifications of the prediction algorithms in the H.265/HEVC encoder. **Proc. SPIE**, v. 9290, 92902u-92902u–12, 2014.

TRUDEAU, L.; COULOMBE, S.; DESROSIERS, C. Cost-Based Search Ordering for Rate-Constrained Motion Estimation Applied to HEVC. **IEEE Transactions on Broadcasting**, v. 64, n. 4, p. 922–932, Dec. 2018. DOI: 10.1109/TBC.2018.2847444.

_____. Rate distortion-based motion estimation search ordering for rate-constrained successive elimination algorithms. In: 2014 IEEE International Conference on Image Processing (ICIP). [S.l.: s.n.], Oct. 2014. p. 3175–3179. DOI: 10.1109/ICIP.2014.7025642.

_____. Sub-partition reuse for fast optimal motion estimation in HEVC successive elimination algorithms. In: 2016 IEEE International Conference on Image Processing (ICIP). [S.l.: s.n.], Sept. 2016. p. 2003–2007. DOI: 10.1109/ICIP.2016.7532709.

TRUDEAU, Luc; COULOMBE, Stephane; DESROSIERS, Christian. An adaptive search ordering for rate-constrained successive elimination algorithms. In: IEEE. IMAGE Processing (ICIP), 2015 IEEE International Conference on. [S.l.: s.n.], 2015. p. 207–211.

TSAI, Sung-Fang; TSAI, Cheng-Han; CHEN, Liang-Gee. Encoder Hardware Architecture for HEVC. In: SZE, V.; BUDAGAVI, Madhukar; SULLIVAN, Gary J. (Eds.). **High Efficiency Video Coding (HEVC): Algorithms and Architectures**. Switzerland: Springer International Publishing, 2014. (Integrated Circuits and Systems). chap. 11, p. 343–375. ISBN 978-3-319-06894-7. DOI: 10.1007/978-3-319-06895-4.

TSMC. **TSMC STANDARD CELL Library TCBN45GSBWPTC**. Taiwan, 2011.

TUMEO, A. et al. A Pipelined Fast 2D-DCT Accelerator for FPGA-based SoCs. In: ISVLSI'07. [S.l.: s.n.], Mar. 2007. p. 331–336. DOI: 10.1109/ISVLSI.2007.13.

USC. **The USC-SIPI Image Database**. Volume 3: Miscellaneous. California, USA: University of Southern California (USC). Available from: <http://sipi.usc.edu/database/database.php?volume=misc>. Visited on: 11 Sept. 2019.

VANNE, J. et al. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. **IEEE Journal of Selected Topics in Signal Processing**, v. 22, n. 12, p. 1885–1898, Dezembro 2012.

VANNERSON, Eric; LIPPINCOTT, Louis. **Transpose buffering for video processing**. [S.l.], Aug. 2007. US Patent App. US20070047655A1. Available from: <https://www.google.com/patents/US20070047655>.

WALLACE, G. K. The JPEG still picture compression standard. **IEEE Transactions on Consumer Electronics**, v. 38, n. 1, p. xviii–xxxiv, Feb. 1992. ISSN 0098-3063. DOI: 10.1109/30.125072.

WALTER, FábioLeandro; DINIZ, CláudioMachado; BAMPI, Sergio. Synthesis and comparison of low-power high-throughput architectures for SAD calculation. English. **Analog Integrated Circuits and Signal Processing**, Springer US, v. 73, n. 3, p. 873–884, 2012.

WANG, Yilin; INGUVA, Sasi; ADSUMILLI, Balu. YouTube UGC Dataset for Video Compression Research. **CoRR**, abs/1904.06457, 2019. arXiv: 1904.06457. Available from: <http://arxiv.org/abs/1904.06457>.

WANG, Z.; BOVIK, A. C.; LU, L. Why is image quality assessment so difficult? In: 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing. [S.l.: s.n.], May 2002. v. 4, p. iv-3313-iv–3316. DOI: 10.1109/ICASSP.2002.5745362.

WANG, Z.; BOVIK, A.C. **Modern Image Quality Assessment**. USA: Morgan & Claypool Publishers, 2006. (Synthesis lectures on image, video and multimedia processing). ISBN 9781598290226.

WANG, Zhou; BOVIK, Alan C.; SHEIKH, Hamid R. Digital Video Image Quality and Perceptual Coding. In: ed. by H.R. Wu and K.R. Rao. [S.l.]: CRC Press, Nov. 2005. Structural Similarity Based Image Quality Assessment.

WHITTINGHAM, M. Stanley. Ultimate Limits to Intercalation Reactions for Lithium Batteries. **Chemical Reviews**, v. 114, n. 23, p. 11414–11443, 2014. DOI: 10.1021/cr5003003.

WIEGAND, T.; SULLIVAN, G. J., et al. Overview of the H.264/AVC video coding standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 13, n. 7, p. 560–576, July 2003. ISSN 1051-8215. DOI: 10.1109/TCSVT.2003.815165.

WIEGAND, Thomas; SCHWARZ, Heiko. Source Coding: Part I of Fundamentals of Source and Video Coding. **Foundations and Trends in Signal Processing**, v. 4, p. 1–224, Jan. 2010. DOI: 10.1561/2000000010.

WIEN, M. **High Efficiency Video Coding - Coding Tools and Specification.** [S.l.: s.n.], 2013. Tutorial.

_____. **High Efficiency Video Coding: Coding Tools and Specification**. [S.l.]: Springer Berlin Heidelberg, 2014. (Signals and Communication Technology). ISBN 9783662442760.

_____. Inter Prediction. In: HIGH Efficiency Video Coding: Coding Tools and Specification. [S.l.]: Springer Berlin Heidelberg, 2014. p. 179–204. (Signals and Communication Technology). ISBN 9783662442760.

_____. Residual Coding. In: HIGH Efficiency Video Coding: Coding Tools and Specification. [S.l.]: Springer Berlin Heidelberg, 2014. p. 205–227. (Signals and Communication Technology). ISBN 9783662442760.

_____. **Versatile Video Coding – Video Compression beyond HEVC: Coding Tools for SDR and 360° Video**. [S.l.: s.n.], May 2018. Talk.

_____. Video Coding Fundamentals. In: HIGH Efficiency Video Coding: Coding Tools and Specification. [S.l.]: Springer Berlin Heidelberg, 2014. p. 22–71. (Signals and Communication Technology). ISBN 9783662442760.

WINKLER, Stefan. In: **Digital Video Image Quality and Perceptual Coding**. Ed. by H.R. Wu and K.R. Rao. [S.l.]: CRC Press, Nov. 2005. Perceptual Video Quality Metrics – A Review, p. 155–179.

_____. **Digital video quality: vision models and metrics**. [S.l.]: John Wiley and Sons, 2005. ISBN 9780470024041.

WIRTH, N. A plea for lean software. **Computer**, v. 28, n. 2, p. 64–68, Feb. 1995. ISSN 0018-9162. DOI: 10.1109/2.348001.

WU, G.; MASIA, B., et al. Light Field Image Processing: An Overview. **IEEE Journal of Selected Topics in Signal Processing**, v. 11, n. 7, p. 926–954, Oct. 2017. ISSN 1932-4553. DOI: 10.1109/JSTSP.2017.2747126.

WU, H.R.; RAO, K.R. **Digital Video Image Quality and Perceptual Coding**. [S.l.]: CRC Press, Nov. 2005.

XU, Weizhi; YIN, Shouyi, et al. High-performance motion estimation for image sensors with video compression. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 15, n. 8, p. 20752–20778, 2015.

XU, Xiaozhong; LIU, Shan. Recent advances in video coding beyond the HEVC standard. **APSIPA Transactions on Signal and Information Processing**, Cambridge University Press, v. 8, p. 1–10, 2019. DOI: 10.1017/ATSIP.2019.11.

YAMORI, K.; TANAKA, Y. Relation between willingness to pay and guaranteed minimum bandwidth in multiple-priority service. In: APCC/MDMC '04. The 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceeding. [S.l.: s.n.], Aug. 2004. v. 1, 113–117 vol.1. DOI: 10.1109/APCC.2004.1391663.

YANG, Shengqi; WOLF, W.; VIJAYKRISHNAN, N. Power and performance analysis of motion estimation based on hardware and software realizations. **IEEE Transactions on Computers**, v. 54, n. 6, p. 714–726, June 2005. ISSN 0018-9340.

YEVGEN VORONENKO, Carnegie Mellon University. **Spiral Software/Hardware Generator for DSP Algorithm**. [S.l.: s.n.], 2017. Available from: <http://spiral.ece.cmu.edu/mcm/gen.html>. Visited on: Oct. 2017.

ZATT, Bruno et al. Run-time Adaptive Energy-aware Motion and Disparity Estimation in Multiview Video Coding. In: PROCEEDINGS of the 48th Design Automation Conference. New York, NY, USA: ACM, 2011. (DAC '11), p. 1026–1031. ISBN 978-1-4503-0636-2.

ZHAI, Guangtao. Digital Video Image Quality and Perceptual Coding. In: ed. by H.R. Wu and K.R. Rao. [S.l.]: CRC Press, Nov. 2005. Recent Advances in Image Quality Assessment, p. 73–97.

ZHANG, F.; BULL, D. R. Rate-distortion Optimization Using Adaptive Lagrange Multipliers, p. 1–1, 2018. ISSN 1051-8215. DOI: 10.1109/TCSVT.2018.2873837.

ZHANG, J. et al. Context Adaptive Lagrange Multiplier (CALM) for Rate-Distortion Optimal Motion Estimation in Video Coding. v. 20, n. 6, p. 820–828, June 2010. ISSN 1051-8215. DOI: 10.1109/TCSVT.2010.2045915.

ZHANG, Y. et al. SIMD acceleration for HEVC encoding on DSP. In: APSIPA ASC'17. [S.l.: s.n.], Dec. 2017. p. 1719–1725. DOI: 10.1109/APSIPA.2017.8282310.

ZHAO, L.; ZHANG, L., et al. Fast mode decision algorithm for intra prediction in HEVC. In: VISUAL Comm. and Image Proc. (VCIP), 2011 IEEE. [S.l.: s.n.], Nov. 2011. p. 1–4. DOI: 10.1109/VCIP.2011.6115979.

ZHAO, L.; ZHAO, X., et al. Wide Angular Intra Prediction for Versatile Video Coding. In: 2019 Data Compression Conference (DCC). Snowbird, USA: [s.n.], Mar. 2019. p. 53–62. DOI: 10.1109/DCC.2019.00013.

ZHIHAI HE et al. Power-rate-distortion analysis for wireless video communication under energy constraints. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 15, n. 5, p. 645–658, May 2005. ISSN 1051-8215. DOI: 10.1109/TCSVT.2005.846433.

ZHU, Ce; XIONG, Bing. Transform-Exempted Calculation of Sum of Absolute Hadamard Transformed Differences. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 19, n. 8, p. 1183–1188, 2009.

Appendix

APPENDIX A – NOTATION

A matrix $\mathbf{A}$ with $m \times n$ elements is denoted as $\mathbf{A}_{m\times n}$. Then, $\mathbf{A}_{i,j}$ is an element of $\mathbf{A}_{m\times n}$ such that $1 \leq i \leq m$ and $1 \leq j \leq n$. The element $\mathbf{A}_{i,j}$ is said to have row index i and column index j (CULLEN, 2012, p. 11), and thus $\mathbf{A}_{m\times n}$ have m rows and n columns. Additionally, if $\mathbf{A}_{i,j}$ is a scalar element, it can also be denoted as $a_{i,j}$. Sometimes we need to index only specific rows or columns from a matrix, and for that we adopt the colon notation (GOLUB; VAN LOAN, 1996).

Definition 1 (Colon Notation (GOLUB; VAN LOAN, 1996))
By such notation, $\mathbf{A}_{k,:}$ means the k^{th} row from $\mathbf{A}_{m\times n}$, i.e.:

$$\mathbf{A}_{k,:} = \begin{bmatrix} a_{k,1} & \dots & a_{k,n} \end{bmatrix} \tag{A.1}$$

Similarly, $\mathbf{A}_{:,k}$ is the k^{th} column from $\mathbf{A}_{m\times n}$:

$$\mathbf{A}_{:,k} = \begin{bmatrix} a_{1,k} & \dots & a_{m,k} \end{bmatrix}^T \tag{A.2}$$

We also often use partitioned matrices. Partitioning a matrix $\mathbf{A}$ means decomposing such matrix into submatrices in a way that each element $\mathbf{A}_{i,j}$ is in one and only one of these submatrices (HORN; JOHNSON, 2012, p. 17). For instance, a block within the original frame $\mathbf{F}^{\text{ori}}$ is a partition of such frame. Such a partition is baptized as original block ($\mathbf{B}^{\text{ori}}$).

To allow for indexing the partitions of a matrix, we have defined the following notation. A matrix $\mathbf{A}$ with $m \times n$ elements and $p \times q$ partitions with $m' \times n'$ elements each, where $p = m/m'$ and $q = n/n'$, is denoted as $\mathbf{A}_{m\times n;m'\times n'}$. Thus, $\mathbf{A}_{i,j}$ is still an element of $\mathbf{A}_{m\times n}$ such that $1 \leq i \leq m$ and $1 \leq j \leq n$. Similarly, $\mathbf{A}_{;i,j}$ is a partition of $\mathbf{A}_{;m'\times n'}$ such that $1 \leq i \leq p$ and $1 \leq j \leq q$. By such notation, a $(\mathbf{A}_{;i,j})_{i',j'}$ is the element in the i'^{th} row and j'^{th} column from within the partition $\mathbf{A}_{;i,j}$ such that $1 \leq i' \leq m'$ and $1 \leq j' \leq n'$.

Also, we can denote $\mathbf{A}_{m\times n}$ as $\mathbf{A}_m$ if $m = n$ and $\mathbf{A}_{;m'}$ if $m' = n'$. For instance, let us define an 8×8 matrix $\mathbf{A}$ divided in 2×2 partitions with 4×4 elements each as:

$$\mathbf{A}_{8;4} = \left[\begin{array}{ccc|ccc} \mathbf{A}_{1,1} & \dots & \mathbf{A}_{1,4} & \mathbf{A}_{1,5} & \dots & \mathbf{A}_{1,8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{4,1} & \dots & \mathbf{A}_{4,4} & \mathbf{A}_{4,5} & \dots & \mathbf{A}_{4,8} \\ \hline \mathbf{A}_{5,1} & \dots & \mathbf{A}_{5,4} & \mathbf{A}_{5,5} & \dots & \mathbf{A}_{5,8} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{8,1} & \dots & \mathbf{A}_{8,4} & \mathbf{A}_{8,5} & \dots & \mathbf{A}_{8,8} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{A}_{;1,1} & \mathbf{A}_{;1,2} \\ \hline \mathbf{A}_{;2,1} & \mathbf{A}_{;2,2} \end{array}\right] \tag{A.3}$$

We also need to define two operations involving partitioned matrices.

Definition 2 (Multiplication of matrices and partitioned matrices)

Be $\mathbf{C}_{m\times n;m'\times n'} = \mathbf{B}_{p\times q}\times\mathbf{A}_{m\times n;m'\times n'}$*, then:*

$$\underbrace{\mathbf{C}_{;i,j}}_{\substack{matrix\\partition}} = \sum_{o=1}^{q} \underbrace{\mathbf{B}_{i,o}\times\mathbf{A}_{;o,j}}_{\substack{scalar\text{-}matrix\\multiplication}} \tag{A.4}$$

Notice that the sizes of the partitioned matrix $\mathbf{A}$ *must be such that the operations in Equation A.4 can be performed. In this case, the number of partitions in each direction in* $\mathbf{A}$ *must be the same as the number of elements in each respective direction in* $\mathbf{B}$*. When such condition is satisfied, the matrices* $\mathbf{A}$ *and* $\mathbf{B}$ *are said to be compatible for such operation.*

Similarly for $\mathbf{C}'_{m\times n;m'\times n'} = \mathbf{A}_{m\times n;m'\times n'}\times\mathbf{B}_{p\times q}$*:*

$$\mathbf{C}'_{;i,j} = \sum_{o=1}^{p} \mathbf{A}_{;i,o}\times\mathbf{B}_{o,j} \tag{A.5}$$

By considering each partition of $\mathbf{A}$ *as an element, both operations mimic the standard matrix multiplication. In fact, they also share some common properties. These multiplications are distributive over matrix addition.*

A function over a matrix may return a scalar or another matrix. In the former case, it is denoted as $f(\mathbf{A})$. In the latter, as $\mathbf{F}(\mathbf{A}_{l\times m})$ and results in a matrix with size $l\times m$. Moreover, the elements of such matrix at position i,j can be denoted as $(\mathbf{F}(\mathbf{A}_{l\times m}))_{i,j}$ or simply $\mathbf{F}(\mathbf{A}_{l\times m})_{i,j}$. We also denote a matrix function over matrix partitions $\mathbf{F}\left(\mathbf{A}_{l\times m;l'\times m'}\right)$, such that $\forall i,j|1 \le i \le l/l'$ and $1 \le j \le m/m'$:

$$\left(\mathbf{F}\left(\mathbf{A}_{l\times m;l'\times m'}\right)\right)_{i,j} = \mathbf{F}\left(\mathbf{A}_{;i,j}\right) \tag{A.6}$$

This means that each $l'\times m'$ sized partition of the $l\times m$ sized resulting matrix of the function $\mathbf{F}\left(\mathbf{A}_{l\times m;l'\times m'}\right)$ is the result of applying the function $\mathbf{F}(\ldots)$ over each co-located $l'\times m'$ sized partition of $\mathbf{A}$. For instance, let $\mathbf{B}_{4;2}$:

$$\begin{aligned}\mathbf{F}\left(\mathbf{B}_{4;2}\right) &= \mathbf{F}\left(\left[\begin{array}{c|c}\mathbf{B}_{;2,1} & \mathbf{B}_{;2,2}\\\hline \mathbf{B}_{;3,1} & \mathbf{B}_{;3,2}\end{array}\right]\right) = \mathbf{F}\left(\left[\begin{array}{cc|cc} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4}\\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4}\\\hline b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4}\\ b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4}\end{array}\right]\right) \\ &= \left[\begin{array}{c|c}\mathbf{F}\left(\mathbf{B}_{;2,1}\right) & \mathbf{F}\left(\mathbf{B}_{;2,2}\right)\\\hline \mathbf{F}\left(\mathbf{B}_{;3,1}\right) & \mathbf{F}\left(\mathbf{B}_{;3,2}\right)\end{array}\right] = \left[\begin{array}{c|c}\mathbf{F}\left(\left[\begin{array}{cc} b_{1,1} & b_{1,2}\\ b_{2,1} & b_{2,2}\end{array}\right]\right) & \mathbf{F}\left(\left[\begin{array}{cc} b_{1,3} & b_{1,4}\\ b_{2,3} & b_{2,4}\end{array}\right]\right)\\\hline \mathbf{F}\left(\left[\begin{array}{cc} b_{3,1} & b_{3,2}\\ b_{4,1} & b_{4,2}\end{array}\right]\right) & \mathbf{F}\left(\left[\begin{array}{cc} b_{3,3} & b_{3,4}\\ b_{4,3} & b_{4,4}\end{array}\right]\right)\end{array}\right]\end{aligned} \tag{A.7}$$

(applying Eq.A.6)

Distortion metrics often need a sum of elements in a matrix to provide a single scalar result. Thus, there are two common ways to sum up the elements. They are presented in Equations A.8 and A.10.

Definition 3 (Sum of absolute elements from a matrix)

$$sa(\mathbf{A}_{m\times n}) = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{i,j}| \tag{A.8}$$

$$\mathbf{SA}\left(\mathbf{A}_{;m'\times n'}\right) = \mathbf{C}_{m'\times n'} \quad | \quad \mathbf{C}_{i,j} = sa\left(\mathbf{A}_{;i,j}\right) \tag{A.9}$$

Definition 4 (Sum of all elements from a matrix (MERIKOSKI, 1984))
Be $\mathbf{A}_{l\times m}$*:*

$$s(\mathbf{A}) = \sum_{i=1}^{m}\sum_{j=1}^{n}a_{i,j} \tag{A.10}$$

The following two properties of Definition 4 are useful in other chapters:

Property 4 (Additivity (MERIKOSKI, 1984, p.178))
$\forall\mathbf{A}_{l\times m}$ *and* $\mathbf{B}_{l\times m}$*:*

$$s(\mathbf{A}+\mathbf{B}) = s(\mathbf{A}) + s(\mathbf{B}) \tag{A.11}$$

Property 5 (Homogeneity of degree 1 (MERIKOSKI, 1984, p.178))
$\forall\mathbf{A}_{l\times m}$ *and* $\forall c \in \mathbb{R}$

$$s(c\times\mathbf{A}) = c\times s(\mathbf{A}) \tag{A.12}$$

APPENDIX B – A BRIEF OVERVIEW ON VIDEO QUALITY ASSESSMENT

This apendix is intended to complement this thesis for the readers that are not familiar with quality assessment. Therefore, we provide a general overview of quality assessment, including a introduction to the Bjøntegaard Delta (BD) model.

The decoded video quality can be assessed either subjectively or objectively. According to Wu and Rao (2005, p. 126), "Subjective assessment[85] is the most reliable way to determine actual image quality, and cannot be replaced with objective testing." Several factors may account for a viewer's experience of "quality" when watching a video, including, among others (WINKLER, 2005a, p. 156):

- The fidelity of the reproduction;
- Display type and properties;
- Viewing conditions; and even
- The accompanying soundtrack.

The fidelity of the reproduction is a crucial factor, as impairments provoke dissatisfaction to the user and may decrease their willingness to pay (YAMORI; TANAKA, 2004) and use some multimedia services or devices (REDI et al., 2005, p. 31). Even if one needs only to assess fidelity subjectively, the other factors must be accounted for during the assessment. Thus, even though being reliable, subjective quality assessment is too slow and expensive for most real world applications (WANG; BOVIK, 2006, p. vii; WIEN, 2014d, p. 64), such as the evaluation of video coding tools.

On the other hand, the majority of objective quality metrics are oblivious to most of those listed factors, besides the reproduction fidelity. Moreover, objective metrics have the advantage of being reproducible by independent evaluators, given the same test set (WIEN, 2014d, p. 65). Objective quality models can be classified as follows (ITU-T, 2000, p. 4; WINKLER, 2005a, p. 158; AKRAMULLAH, 2014, p. 121):

- Full Reference (FR): compares the reconstructed frame directly with the original "raw" one (see Figure 10). The original one is assumed pristine, i.e., to have perfect quality (WANG; BOVIK; SHEIKH, 2005, p. 225). Such an assumption is one of the weaknesses of this class (WANG; BOVIK; LU, 2002, p. 3314). In general, the quality metrics in this category perform pixel-wise evaluation;
- Reduced Reference (RR): are based on features from both original and processed sequences. Those features are what are directly compared. Yet, to derive the features, the original sequence is still required;

[85] I.e., the quality is evaluated by human subjects (WANG; BOVIK, 2006, p. vii), thus it depends, among other factors, on the HVS.

- No Reference (NR): only have the reconstructed video information. They often need to make assumptions about the content and the distortions (AKRAMULLAH, 2014, p. 123).

Among these, the FR metrics are computed similarly to the distortion metrics[86] used in BMA. Indeed, FR quality metrics are often defined over the error[87] matrix $\mathbf{E}$, as in the SSE presented in Equation 2.9. However, instead of being the difference between blocks (Equation 2.8), in this case, $\mathbf{E}$ contains the difference between the frames (both with h vertical pixels and w horizontal pixels) as:

$$\mathbf{E}_{h\times w} = {\mathbf{F}^{\text{ori}}}_{h\times w} - {\mathbf{F}^{\text{rec}}}_{h\times w} \tag{B.1}$$

One problem measuring frame distortion using the sum of errors, or even the sum of squared errors, is that it depends on the frame size $h \times w$. One solution to avoid that is to compute the MSE defined in Equation B.2, given that $e_{i,j}$ is an element of $\mathbf{E}_{h\times w}$.

$$m(\mathbf{SE}) = \frac{su(\mathbf{SE})}{width \times height} = \frac{\sum_{i=1}^{width}\sum_{j=1}^{height}(\mathbf{SE}_{i,j})}{width \times height} \tag{B.2}$$

Essentially, MSE measures the energy difference between $\mathbf{F}^{\text{ori}}$ and $\mathbf{F}^{\text{rec}}$ (ZHAI, 2005, p. 79). One problem with MSE is that it depends strongly on the image intensity scaling. In contrast, the so-called PSNR avoids this problem by scaling the MSE according to the image dynamic-range, i.e., the maximum amplitude of the original signal which depends on the number of bpp/channel (WIEN, 2014d, p. 65). The PSNR (in dB) is obtained after Equation B.3, where b is the color depth (in bpp) of the evaluated channel:

$$psnr\left(\mathbf{F}^{\text{ori}},\mathbf{F}^{\text{rec}}\right) = 10 \times log_{10}\left(\frac{2^b - 1}{m(\mathbf{SE})}\right) \tag{B.3}$$

Notice that while MSE is a distortion metric, the PSNR is, in fact, a quality metric (WINKLER, 2005b, p. 55, 144) and thus the higher the $psnr\left(\mathbf{F}^{\text{ori}},\mathbf{F}^{\text{rec}}\right)$, the lower is the error $\mathbf{E}$ (also-called noise) and hence, the better is the reconstructed image ($\mathbf{F}^{\text{rec}}$) quality (since $\lim_{m(\mathbf{SE})\to 0}\text{PSNR} = +\infty$). However, according to Larbier (2011, p. 2), when comparing the quality figures of two different video samples, their PSNR have little to no value. Even so, two PSNR measurements for the same video sample evidence quite well the potential of two CODECs (LARBIER, 2011, p. 2). Therefore, considering the PSNR as a quality metric, a given CODEC is said to be more efficient than another when it minimizes the bitrate whereas maximizing the PSNR.

Both the MSE and PSNR are widely known and have been ubiquitously used as a criterion in visual signal communication systems (ZHAI, 2005, p. 78–79; WINKLER, 2005b, p. 54; WIEN, 2014d, p. 65), despite being little correlated to the HVS (WINKLER, 2005b, p. 54, 150). In fact, during the development of most video encoder tools, the most used and

[86] A distortion metric can be interpreted as a inverse of a quality metric. Therefore, the larger is the distortion, smaller is the quality, and vice versa.

[87] See Note 24.

accepted quality metric is the PSNR. As the PSNR is computed frame-by-frame, its average is taken as a quality measure for the whole sequence (WIEN, 2014d, p. 65).

B.1 THE BJØNTEGAARD DELTA (BD) MODEL

Given that the quantization generates losses according to the quantization level, i.e., with the used QP, the CODEC efficiency must be measured within a quantization interval. In general, one may compare two CODECs by analyzing their RD curves (WIEN, 2014d, p. 65). However, such a visual approach is not practical for evaluating a large number of CODECs, nor even the efficiency of each tool in a given CODEC.

That is why Bjøntegaard (2001) proposed a method for comparing the average RD curves, known as Bjøntegaard Delta (BD) model. Such a model is based on the RD curve interpolation using third-degree polynomials, as presented in Equation B.4, adapted from (LI; WIEN; OHM, 2010). In Equation B.4, $distortion^{\text{cod}}(r)$ is the PSNR after a codification (cod) as a function of the resulting bitrate (r). Each a_i is a coefficient obtained after the interpolation of four RD pairs, each obtained using one out of four QPs.

$$distortion^{\text{cod}}(r) = a_0 \times (log_{10} r)^3 + a_1 \times (log_{10} r)^2 + a_2 \times log_{10} r + a_3 \tag{B.4}$$

One may notice that the bitrate is considered on a logarithmic scale to compensate for the logarithmic scale of the PSNR (Equation B.3). The bitrate variation (Δr) is obtained as shown in Equation B.5, considering a pair of interpolated RD curves, the reference curve (baseline) and the test curve (test):

$$\Delta r = \frac{\int_{r^{\text{low}}}^{r^{\text{high}}} (distortion^{\text{baseline}}(r) - distortion^{\text{test}}(r))\text{dr}}{r^{\text{high}} - r^{\text{low}}} \tag{B.5}$$

The r^{low} and r^{high} parameters are, respectively, the minimum and maximum bitrate used to ensure that the entire interval of the obtained integral is in both interpolated curves, limited by their QPs, thus avoiding extrapolations of the curves beyond the limiting points (BJØNTEGAARD, 2008). The way (direction) by which such parameters are obtained defines the computed metric, which can be either the BD-PSNR or the BD-Rate. In the latter case, another transformation is required to represent the value on a linear scale again. While BD-PSNR represents the expected PSNR variation for a given bitrate, the BD-Rate represents the average relative variation (in %) of the bitrate for a given PSNR. Since both BD-Rate and BD-PSNR relate together **Rate** and **Distortion** (quality), they are a means to evaluate **coding efficiency**.

Thus, this work employs the BD-Rate as a metric to evaluate coding efficiency in this thesis, as presented in the next section. A negative BD-Rate means compression gains, i.e., the evaluated encoder/configuration reduces the bit-rate for the same quality in PSNR with respect to the baseline encoder/configuration. A positive value thus means compression losses, i.e., an average bit-rate increase for the same PSNR quality.

APPENDIX C – FME ARCHITECTURE RESULTS

Table 22 – Synthesis results for SAD (hierarchy, clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	86302.28	86364.02	88867.67	96400.30
Dynamic P. (μW)	1392.70	2821.90	5500.00	11767.10
Static P. (μW)	1344.80	1345.40	1481.00	1814.80
Total P. (μW)	2737.50	4167.30	6981.00	13581.90
Energy (μJ)	2.79	2.13	1.78	1.73

Source: the author

Table 23 – Synthesis results for SAD (hierarchy, no clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	98646.40	99676.05	101204.38	109897.73
Dynamic P. (μW)	2960.50	5825.10	11571.20	23714.10
Static P. (μW)	1442.50	1507.10	1577.90	1977.40
Total P. (μW)	4403.00	7332.20	13149.10	25691.50
Energy (μJ)	4.49	3.74	3.35	3.28

Source: the author

Table 24 – Synthesis results for SAD (flat, clock-gating)

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	79937.95	79980.64	83031.83	93998.44
Dynamic P. (μW)	25631.70	51062.00	101955.50	204193.00
Static P. (μW)	1271.30	1277.00	1435.80	1743.60
Total P. (μW)	26903.00	52339.00	103391.30	205936.60
Energy (μJ)	27.44	26.69	26.36	26.26

Source: the author

Table 25 – Synthesis results for SAD (flat, no clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	96090.19	96088.08	97917.34	109489.71
Dynamic P. (μW)	2802.40	5426.50	10777.00	21971.90
Static P. (μW)	1435.90	1434.90	1523.60	1837.20
Total P. (μW)	4238.30	6861.40	12300.60	23809.10
Energy (μJ)	4.32	3.50	3.14	3.04

Source: the author

Table 26 – Synthesis results for SATD (hierarchy, clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	168425.48	168512.62	171093.71	184858.20
Dynamic P. (μW)	3188.80	6171.70	12388.40	24690.80
Static P. (μW)	2298.60	2298.70	2712.10	3224.30
Total P. (μW)	5487.40	8470.40	15100.50	27915.10
Energy (μJ)	5.60	4.32	3.85	3.56

Source: the author

Table 27 – Synthesis results for SATD (hierarchy, no clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	189210.70	189869.90	191918.96	206761.09
Dynamic P. (μW)	5796.10	11245.20	22377.20	43939.30
Static P. (μW)	2439.60	2557.10	2871.60	3427.80
Total P. (μW)	8235.70	13802.30	25248.80	47367.10
Energy (μJ)	8.40	7.04	6.44	6.04

Source: the author

Table 28 – Synthesis results for SATD (flat, clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	158010.30	158066.39	162899.04	176950.89
Dynamic P. (μW)	27751.40	55096.10	109846.50	220103.10
Static P. (μW)	2516.20	2515.90	2685.70	3187.70
Total P. (μW)	30267.60	57612.00	112532.20	223290.80
Energy (μJ)	30.87	29.38	28.70	28.47

Source: the author

Table 29 – Synthesis results for SATD (flat, no clock-gating)

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	183577.36	183547.90	187335.21	199020.30
Dynamic P. (μW)	5457.30	10594.90	20904.00	42619.70
Static P. (μW)	2584.00	2583.30	2798.20	3214.80
Total P. (μW)	8041.30	13178.20	23702.20	45834.50
Energy (μJ)	8.20	6.72	6.04	5.84

Source: the author

Table 30 – Synthesis results for SEA (hierarchy, clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	198026.28	198154.88	201118.93	217917.85
Dynamic P. (μW)	3319.50	6433.30	12751.40	25602.50
Static P. (μW)	2691.50	2698.50	3138.00	3787.00
Total P. (μW)	6011.00	9131.80	15889.40	29389.50
Energy (μJ)	6.13	4.66	4.05	3.75

Source: the author

Table 31 – Synthesis results for SEA (hierarchy, no clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	221806.95	222465.10	224967.68	244131.96
Dynamic P. (μW)	6308.50	12485.20	24031.10	49441.50
Static P. (μW)	2844.10	2957.50	3292.90	3984.50
Total P. (μW)	9152.60	15442.70	27324.00	53426.00
Energy (μJ)	9.34	7.88	6.97	6.81

Source: the author

Table 32 – Synthesis results for SEA (flat, clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	185306.43	185227.05	190644.83	203609.34
Dynamic P. (μW)	27500.50	54584.00	108980.60	217913.20
Static P. (μW)	2955.90	2950.90	3117.70	3559.50
Total P. (μW)	30456.40	57534.90	112098.30	221472.70
Energy (μJ)	31.07	29.34	28.59	28.24

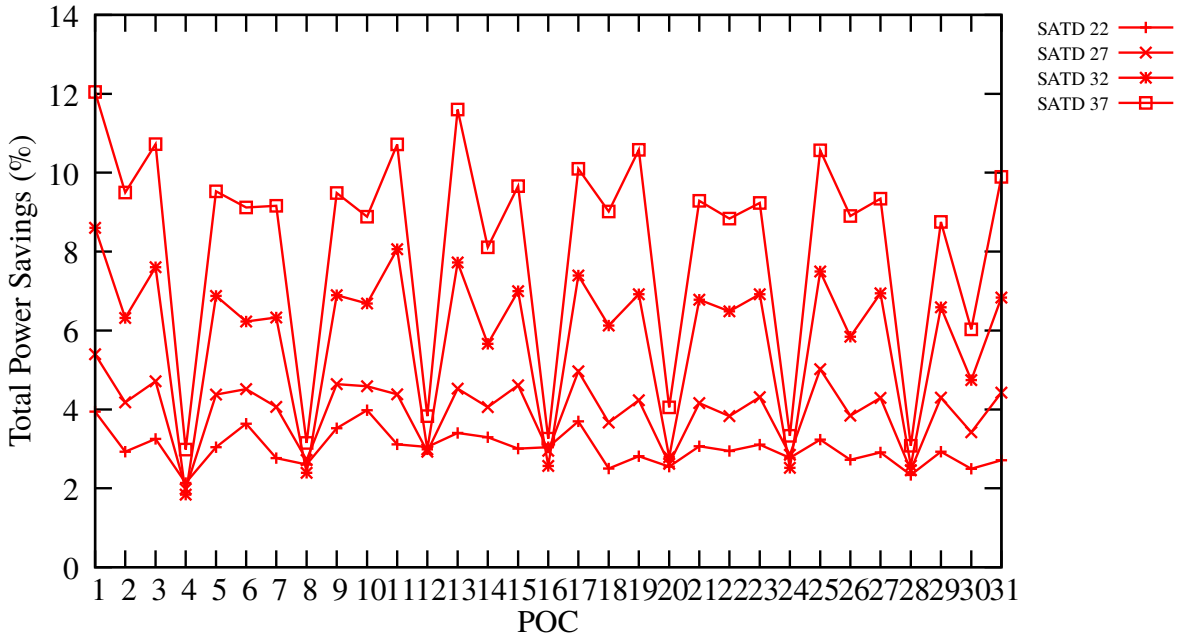
Source: the author

Table 33 – Synthesis results for SEA (flat, no clock-gating).

Target	1080@30fps	1080@60fps	2160@30fps	2160@60fps
Period (ns)	20	10	5	2.5
Area (μm^2)	216124.57	216390.06	218176.46	236913.84
Dynamic P. (μW)	6139.90	11907.60	23559.70	48299.60
Static P. (μW)	3168.40	3172.30	3212.10	3860.60
Total P. (μW)	9308.30	15079.90	26771.80	52160.20
Energy (μJ)	9.49	7.69	6.83	6.65

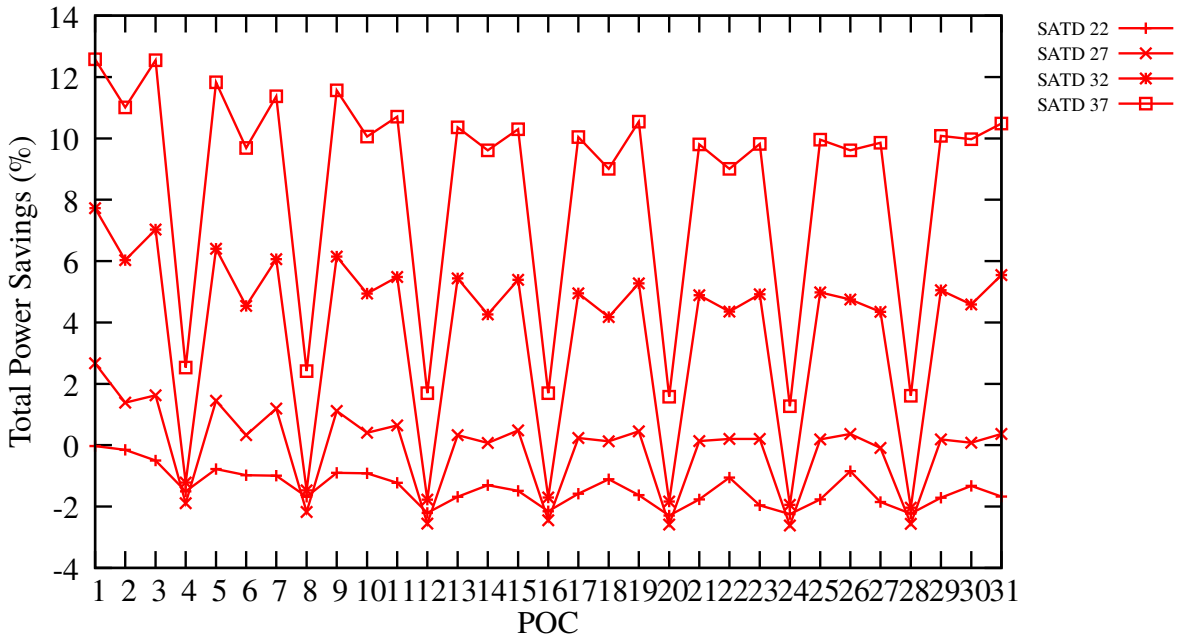
Source: the author

Figure 84 – Total power savings of SEA-FME architecture simulated with “BQTerrace” sequence.



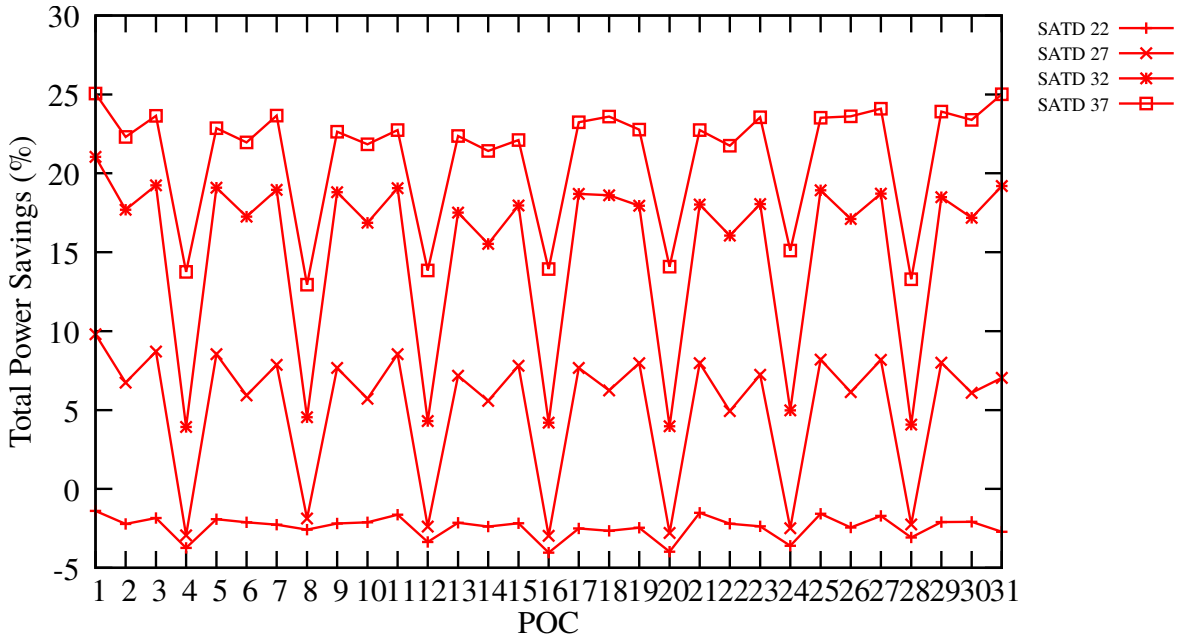
Source: the author.

Figure 85 – Total power savings of SEA-FME architecture simulated with “PeopleOnStreet” sequence.



Source: the author.

Figure 86 – Total power savings of SEA-FME architecture simulated with “Jhonny” sequence.



Source: the author.

APPENDIX D – LIST OF PUBLICATIONS

D.1 WORKS AS FIRST AUTHOR

1. **ALOG – Springer Science & Business Media Analog Integrated Circuits and Signal Processing**;
    - **Title**: Towards Optimal Use of Pel Decimation to Trade off Quality for Energy;
    - **Authors**: Ismael Seidel, André Beims Bräscher, Marcio Monteiro, José Luís Güntzel;
    - **Qualis CC**: P-B1;
    - **Date**: October, 2015;
2. **PATMOS 2015 – 25^{th} International Workshop on Power And Timing Modeling, Optimization and Simulation**;
    - **Title**: Combining Pel Decimation with Partial Distortion Elimination to Increase SAD Energy Efficiency;
    - **Authors**: Ismael Seidel, André Beims Bräscher, José Luís Güntzel;
    - **Qualis CC**: E-B3;
    - **Date**: September, 2015;
3. **LASCAS 2016 – 7^{th} IEEE Latin American Symposium on Circuits and Systems**;
    - **Title**: Squarer Exploration for Energy-Efficient Sum of Squared Differences;
    - **Authors**: Ismael Seidel, Marcio Monteiro, José Luís Güntzel, Luciano Volcan Agostini;
    - **Qualis CC**: E-B2;
    - **Date**: February, 2015;
4. **ISCAS 2016 – 2016 IEEE International Symposium on Circuits and Systems**;
    - **Title**: Energy-Efficient SATD for Beyond HEVC;
    - **Authors**: Ismael Seidel, André Beims Bräscher, José Luís Güntzel, Luciano Volcan Agostini;
    - **Qualis CC**: E-A1;
    - **Date**: May, 2016;
5. **ICIP 2016 – 2016 IEEE International Conference on Image Processing**;

- **Title**: Rate-constrained Successive Elimination of Hadamard-based SATDs;
- **Authors**: Ismael Seidel, Luiz Henrique Cancellier, José Luís Güntzel, Luciano Agostini
- **Qualis CC**: E-A1;
- **Date**: September, 2016;

6. **ICECS 2016 – 23rd IEEE International Conference on Electronics Circuits and Systems**;

   - **Title**: Coarse Grain Partial Distortion Elimination for Hadamard ME in HEVC;
   - **Authors**: Ismael Seidel, José Luís Güntzel, Luciano Volcan Agostini;
   - **Qualis CC**: E-B1;
   - **Date**: December, 2016;

7. **ISCAS 2018 – 23rd IEEE International Conference on Electronics Circuits and Systems**;

   - **Title**: Coding- and energy-efficient FME hardware design;
   - **Authors**: Ismael Seidel, Vânio Rodriguez Filho, Luciano Volcan Agostini, José Luís Güntzel;
   - **Qualis CC**: E-A1;
   - **Date**: May, 2018;

8. **IEEE TCAS-I – IEEE Transactions on Circuits and Systems I: Regular Papers** ;

   - **Title**: Energy-efficient Hadamard-based SATD Hardware Architectures Through Calculation Reuse;
   - **Authors**: Ismael Seidel, Marcio Monteiro, Bruno Bonotto, Luciano Agostini, José Luís Güntzel;
   - **Qualis CC**: P-A1;
   - **Date**: June, 2019;

D.2 OTHER WORKS – AS CO-AUTHOR

1. **JICS – Journal of Integrated Circuits and Systems**;

   - **Title**: Exploring Optimized Hadamard Methods to Design Energy-Efficient SATD Architectures;
   - **Authors**: Luiz Henrique Cancellier, **Ismael Seidel**, André Beims Bräscher, José Luís Güntzel, Luciano Volcan Agostini;
   - **Qualis CC**: P-B4;

- **Date**: August, 2015;

2. **36o Concurso de Trabalhos de Iniciação Científica (CTIC)**;

   - **Title**: Algoritmo de Eliminações Sucessivas em Níveis baseado na Soma das Diferenças Transformadas Absolutas;
   - **Authors**: Luiz Henrique Cancellier, **Ismael Seidel**, José Luís Güntzel;
   - **Date**: July, 2017;
   - **Note**: Classified among the 10 best works in the Contest of Scientific Initiation (Computer Science);

3. **SBCCI 2017 – 30rd Symposium on Integrated Circuits and Systems Design**;

   - **Title**: Improving the energy efficiency of a low-area SATD hardware architecture using fine grain PDE;
   - **Authors**: André Beims Bräscher, **Ismael Seidel**, José Luís Güntzel;
   - **Qualis CC**: E-B2;
   - **Date**: August, 2017;

4. **SBCCI 2017 – 30rd Symposium on Integrated Circuits and Systems Design**;

   - **Title**: Block Matching Hardware Architecture for SATD-based Successive Elimination;
   - **Authors**: Luiz Henrique Cancellier, **Ismael Seidel**, José Luís Güntzel;
   - **Qualis CC**: E-B2;
   - **Date**: August, 2017;

5. **LASCAS 2018 – 9th IEEE Latin American Symposium on Circuits and Systems**;

   - **Title**: On HEVC Robustness to Integer Motion Estimation Pruning;
   - **Authors**: Marcio Monteiro, **Ismael Seidel**, José Luís Güntzel;
   - **Qualis CC**: E-B2;
   - **Date**: February, 2018;

6. **SIM 2018 – 33rd South Symposium on Microelectronics**;

   - **Title**: A Named-Pipe Library for Hardware Simulation;
   - **Authors**: Bruno Bonotto, **Ismael Seidel**, Luiz Henrique Cancellier, Marcio Monteiro, José Luís Güntzel;
   - **Date**: Maio, 2018;

- **Note**: Best Paper Award no SIM2018, categoria "With na undergraduate student as first author", SBC (Sociedade Brasileira de Computação) e SBMicro (Sociedade Brasileira de Microeletrônica);

7. **ICECS 2018 – 25rd IEEE International Conference on Electronics Circuits and Systems**;

   - **Title**: On HEVC Robustness to Integer Motion Estimation Pruning;
   - **Authors**: Luiz Henrique Cancellier, **Ismael Seidel**, José Luís Güntzel;
   - **Qualis CC**: E-B1;
   - **Date**: December, 2018;