UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FÁBIO LUÍS LIVI RAMOS

Efficient High-Throughput and Power-Saving Hardware Architectural Design for the HEVC Entropy Encoder

Ph.D. Thesis

Thesis presented as partial requirement for the Ph.D. degree in Computer Science

Advisor: Prof. Dr. Sergio Bampi
Co-Advisor: Prof. Dr. Marcelo Schiavon Porto

Porto Alegre
2019

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ramos, Fábio Luís Livi

Efficient High-Throughput and Power-Saving Hardware Architectural Design the HEVC Entropy Encoder / Fábio Luís Livi Ramos. -- 2019.

187 f.:il.

Orientador: Sergio Bampi;

Co-orientador: Marcelo Schiavon Porto.

Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Instituto de Informática, Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2019.

1. HEVC. 2. CABAC 3.Binary Arithmetic Encoder 4. Real-time Video Processing 5. Power-saving Hardware Design. I. Bampi, Sergio, orient. II. Porto, Marcelo Schiavon, coorient. III. Título.

There are places I'll remember
All my life, though some have changed
Some forever, not for better
Some have gone, and some remain
All these places had their moments
With lovers and friends, I still can recall
Some are dead, and some are living
In my life, I've loved them all

John Lennon

Stop you're trying to bruise my mind
I can do it on my own
Stop you're trying to kill my time
It's been my death since I was born
I don't remember half the time
If I'm hiding or I'm lost
But I'm on my way, on my way

Chris Cornell

Toda vez que falta luz
Toda a vez que algo nos falta
O invisível nos salta aos olhos
Um salto no escuro da piscina

Humberto Gessinger

I can’t see, the end of me
My whole expanse, I cannot see
I formulated infinity
Stored deep inside me

Curt Kirkwood

AGRADECIMENTOS

Uma tese de doutorado, apesar de ser, a grosso modo, o trabalho de uma pessoa só, requer uma rede de apoio de pessoas amadas e queridas para dar suporte durante todo o processo. De alguma forma, todas essas pessoas são coautoras desse trabalho, seja no apoio emocional e carinho dados, seja efetivamente em contribuições técnicas para essa tese, seja em ambos.

Primeiramente, as duas pessoas mais importantes na minha vida, que são a minha esposa Betina e meu filho Pedro, os quais são o centro do meu universo, sem sombras de dúvidas. Minha esposa, que é, sem dúvida, minha melhor amiga e companheira, que estamos há mais de uma década juntos aprendendo juntos sobre a vida, e a quem eu amo muito! E meu filho, que fez com que minha percepção sobre o que é o amor e sobre o que é a vida mudar de uma forma como nunca antes, por quem eu tento ser um ser humano melhor, e a quem eu amo muito! Obrigado por todo o amor, carinho, ajuda e incentivo!! Sem vocês dois, nada disso teria sido possível e sem vocês dois também não teria valido a pena!

Meus pais, Gregório e Rosa, e minha irmã, Renata, são parte fundamental de quem eu sou. Os agradeço por terem me dado uma criação digna, respeito, amor, carinho, por sempre terem estado presentes, sempre terem me incentivado no que tange a questão do ensino, e também pelos (merecidos) puxões de orelha. Hoje percebo como sou privilegiado de ter tido uma família como a que tive, o que nem sempre é verdade no mundo que vivemos. Obrigado por tudo, amos vocês!

Meus sogros, Carmo e Leoni, e minha cunhada, Fernanda, que foram como uma segunda família para mim, igualmente sempre me apoiando em tudo desde e época do mestrado, sendo igualmente companheiros e fundamentais em quem eu sou hoje. São pessoas dignas, a quem devo meu reconhecimento e respeito. Obrigado por tudo!

Meu orientador, professor Sergio Bampi, pela sua orientação, apoio e respeito ao longo de todo o processo de doutoramento, além de me ensinar pontos muitos importantes que vão além da questão técnica da orientação. Cabe destacar o seu apoio incomensurável durante o começo do doutoramento onde, por questão burocráticas, quase tive que interromper o curso. Graças a sua ajuda (literalmente comprando briga com algumas pessoas para me apoiar) pude seguir no curso da forma como planejada. Meu reconhecimento e agradecimento ao prof. Bampi!

Meus co-orientadores Marcelo Porto (oficialmente) e Bruno Zatt (não oficialmente) por todo o apoio, aconselhamento e, por último, mas não menos, pela amizade. Os agradeço

profundamente por toda ajuda que me deram ao longo do doutoramento, os eventos durante minhas estadias em Pelotas, e pela parceria e amizade desde a época da graduação/mestrado. As suas orientações, sem dúvida, facilitaram as decisões sobre os caminhos a seguir ao longo do doutoramento! Muito obrigado, gurizada!!

A gurizada de “Imbé”, cuja a amizade vem desde a nossa época da graduação na Engenharia de Computação, especialmente: irmãos Zatt (Bruno e Gordo), Osvaldo, Kunz, Giancarlo, gêmeos (Artur e Guilherme), Nondillo e Jonas. Agradeço pela parceria e amizade. Uma pena não podermos nos ver mais tanto quanto antes, visto que cada um mora em uma parte do mundo, mas o elo e amizade continuam firmes e fortes!! Não vejo a hora de podermos nos reunir novamente em Imbé ou onde quer que seja!

A gurizada do CEITEC, onde trabalhei por quase cinco anos, e onde formamos um círculo forte de amizade e respeito: Wagston, Hervé, João, Fred, Ferrão, Rohde, Rubinei, Marcelo, Zé, Lauro, Garibotti, Cyrille e Jana. Fico no aguardo para nosso próximo churrasco de confraternização! Além disso, cabe um agradecimento ao Muru e ao Alain, que foram meus gerentes durante o período que lá estive, e foram líderes dignos e justos.

Também cabe meu agradecimento aos meus alunos da Unipampa, especialmente meus orientados, e mais especialmente a Camila, a Luana, e o Alessandro, cujos TCCs estiveram envolvidos no escopo dessa tese.

Um reconhecimento especial ao pessoal da época do colégio militar, da época do curso de Engenharia de Computação na UFRGS, da época da Datacom, e também da época do mestrado no Laboratório 215, assim como a alguns colegas professores da Unipampa. Muitos eu acabei perdendo o contato, mas ainda tenho carinho e reconhecimento a todos.

Devo também um agradecimento a algumas pessoas e entidades, seja por elas serem identitárias a meu respeito, seja por fazerem parte da minha vida de forma intrínseca: as bandas Soundgarden, Beatles, Engenheiros do Hawaii, Alice in Chains, Nirvana, Kyuss, Meat Puppets, Helmet, Raimundos, QOTSA, Pixies, Ween, Faith No More, Red Hot Chili Peppers, Led Zeppelin, Black Sabbath, Silverchair, Foo Fighters, Pearl Jam, Pantera, Rammstein, Offspring, Weezer, para citar as mais importantes. Aos artistas Van Gogh, Raul Seixas, Noel Guarany, Cenair Maicá, Júpiter Maçã, Bert Jansch, Jackson C. Frank, os comediantes do Chaves e do Hermes e Renato. Ao filme Clube da Luta e aos dois primeiros filmes do Predador e Alien. Às universidades UFRGS, Unipampa e UFpel. Às cidades de Porto Alegre, Santa Cruz do Sul, Imbé, São Borja, Canela e Bagé. Uma menção honrosa às bandas Temple of the Dog, Mad Season, Rage Against the Machine, Green Day, Legião Urbana, Titãs,

Queen, Creedence Clearwater Revival, Stone Temple Pilots, Red Fang, Stoned Jesus, Morphine, Ace of Base, Banzai e Slayer, e ao artista José Mendes e Michael Jackson.

Por fim, dentro da minha atual crença de vida, creio que uma coisa tão complexa como a nossa consciência não suma simplesmente ao morrer. Assim sendo, creio que as coisas podem ter um propósito e espero estar cumprindo, dentro do possível, ao meu. Agradeço por estar vivo, ter saúde, poder trabalhar, estudar, e viver com minha família e amigos momentos bons e ruins, como a vida tem que ser! Sou um privilegiado por tudo!

ABSTRACT

The advances in digital video processing, such as the new generation of videos resolutions, led to new challenges in order to transmit and storage the related data. In this scenario, real-time digital video processing is an important goal, which requires specific video-processing architectures to accomplish the demanded constraints. Moreover, tethered devices that transmit and receive video draw the attention to power-saving design for these architectures, due to the energy constraints of the battery-based context of these solutions. HEVC (High-Efficiency Video Coding) standard emerges as an alternative to cope with the mentioned situations involving digital video processing. In HEVC, only one entropy-encoding algorithm exists, which is the CABAC (Context-Adaptive Binary Arithmetic Coding). A single high-throughput instance of a hardware CABAC block is a desirable goal in order to save power, area, and coding efficiency. Therefore, the global goal of this research is configurable high-throughput power-efficient single-instance CABAC design, where high-throughput scheme along with power saving techniques are integrated, considering a compromise trade-off between both performance and power/energy dissipation, adapting the architecture according to it. This Thesis focused on the BAE (Binary Arithmetic Encoder) block, which is the processing bottleneck of the CABAC. As a first contribution, a low-power hardware BAE design is presented, where fine-grain insertion of power-saving reduction techniques into different proposed BAE designs, leading to power savings ranging from 10-40% for different BAEs architectural designs. Towards ultra-high throughput performance, the novel Multiple-Bypass Bins Scheme (MBBS) proposition happens within the context presented, where multiple values of a particular type of BAE data (i.e., bypass bins) are processed at the same time. The integration of the MBBS with prior-art techniques for the BAE blocks led to an increase of around 13% more bins/s compared to the highest prior-BAE design found in the literature. Additionally, an efficient BAE design with MBBS is proposed, achieving closely related throughput values compared to the highest performance of prior-art design, at the advantage of using smaller and easier-to-scale design. This latter design was used as the baseline of the final contribution of this Thesis, combining the power-saving approach and MBBS propositions: a configurable BAE design, which can configure itself to accomplish a better trade-off in terms of performance and energy dissipation through the video processing.

Keywords: HEVC. CABAC. Binary Arithmetic Encoder (BAE). Multiple-Bypass Bins Processing. Low-power CMOS Design. Configurable Design.

Design em Hardware Arquitetural Eficiente para Alta-Vazão e Economia de Potência para o Codificador de Entropia HEVC

RESUMO

Os avanços no processamento digital de vídeos, geraram novos desafios para transmitir e armazenar os dados relacionados. Nesse cenário, o processamento de vídeo em tempo real requer arquiteturas específicas para se alcançar as demandas relacionadas. Ademais, dispositivos móveis que transmitem e recebem vídeo necessitam de projetos visando eficiência energética, devido às restrições do uso de bateria nesse contexto. O padrão HEVC (*High-Efficiency Video Coding*) é uma alternativa para lidar com as situações apresentadas, onde apenas um algoritmo de codificação de entropia existe, que é o CABAC (*Context-Adaptive Binary Arithmetic Coding*). Uma única instância de um bloco em hardware do CABAC é desejável para economizar potência, área, e manter a eficiência de codificação. Portanto, o objetivo global dessa pesquisa é um projeto configurável de alta-vazão e eficiente energeticamente em uma única instância do bloco CABAC, onde técnicas para alta-vazão junto de técnicas para redução do consumo de potência são integradas, adaptando a arquitetura de acordo com isso. Essa tese focou no bloco BAE (*Binary Arithmetic Encoder*), pois esta etapa é o gargalo em termos de processamento do CABAC. Uma primeira contribuição é a inserção de técnicas em baixo nível para redução do consumo de potência em diferentes projetos do bloco BAE. O uso das técnicas escolhidas gerou economia de potência variando entre 10% a 40%. Em buscas de ultra-alta-performance, ocorreu a proposta para processamento de múltiplos bins bypass (MBBS), onde múltiplos valores de um tipo especifico de dados (i.e., bypass bins) são processados ao mesmo tempo. A integração do MBBS com técnicas da literatura para o BAE gerou um aumento de vazão na ordem de 13% quando comparado com o trabalho de maior vazão encontrado na literatura. Adicionalmente, uma alternativa eficiente do bloco BAE com MBBS é proposta, alcançando valores muito próximos quando comparada com a solução anterior com maior vazão da literatura, com a vantagem de um projeto menor e com maior escalabilidade. Essa última arquitetura foi utilizada como base para a contribuição final dessa tese, combinando as técnicas *low-power* e a proposta MBBS: um design BAE configurável, que consegue se modificar para alcançar um melhor balanceamento em termos de vazão e energia durante o processamento do vídeo.

Palavras-chave: HEVC. CABAC. Codificador Aritmético Binário. Processamento de Múltiplos Bins Bypass. Design CMOS de Baixo Consumo. Design Configurável.

FIGURE LIST

TABLE LIST

ABBREVIATIONS

AMP	Asymmetric Motion Partition
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
BAE	Binary Arithmetic Encoding
BC	Binarization Core
BPBS	Bypass Bin Splitting
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable Length Coding
CMBB	Complete Multiple Bypass Bins
CMOS	Complementary Metal-Oxide Semiconductor
COEFF1	coeff_abs_level_greater1_flag
COEFF2	coeff_abs_level_greater2_flag
CSBF	coded_sub_block_flag
CTU	Coding Tree Unit
CU	Coding Unit
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
EG	Exponential Golomb
FL	Fixed Length
FPGA	Field-Programmable Gate-Array
FSE	Full Syntax Element
FU	Full Unit
GoP	Group of Picture
GND	Ground
HD	High Definition
HEVC	High-Efficiency Video Coding
HM	HEVC Reference Model
HPC	Hybrid Path Coverage
JVET	Joint Video Exploration Team
LAST	last_sig_coeff_x, last_sig_coeff_y
LD	Low-Delay
LH	Look-ahead

LPS	Least Probable Symbol
LU	LPS Unit
LZD	Leading Zero Detector
MAE	Memory-Access Efficient Approach
MBBS	Multiple Bypass Bins Scheme
MRSET	Multiple Residual Syntax Element Treatment
MC	Motion Compensation
ME	Motion Estimation
MPS	Most Probable Symbol
MU	MPS Unit
MV	Motion Vector
NMOS	Negative Metal-Oxide Semiconductor
OB	Outstanding Bits
PDK	Project Design Kit
PIPO	Parallel-In Parallel-Out
PMOS	Positive Metal-Oxide Semiconductor
PN	Pre-renormalization
PRSET	Prior Residual Syntax Element Treatment
PU	Prediction Unit
PVT	Process Voltage Temperature
QP	Quantization Parameter
RA	Random Access
RGB	Red, Green, Blue
RMBB	Reduced Multiple-Bypass Bins
RQT	Residual Quad Tree
RTL	Register Transfer Level
PVT	Process, Voltage, Temperature
REM	coeff_abs_level_remaining
SAO	Sample Adaptive Offset
SE	Syntax Element
SIG	sig_coeff_flag
SIGN	coeff_sign_flag
SMP	Symmetric Motion Partition
SSE	Simplified Syntax Element

UHD	Ultra-High Definition
YCbCr	Luminance, Chrominance Blue, Chrominance Red
YUV	Luminance, Chrominance Component 1, Chrominance Component 2
TR	Truncate Rice
TU	Transform Unit
TU	Truncate Unary
U	Unary
VDD	Voltage
VVC	Versatile Video Coding
WPP	Wavefront Parallel Processing
WQXGA	Wide Quad Extended Graphics Array

SUMMARY

1 INTRODUCTION

Video coding is a field of high interest in the past few years, due to increasing demand for video processing, storage, and transmission. For instance, the limitation in traffic bandwidth causes the transmission of raw video (i.e., a given video sequence without any compression) to be prohibitive, especially considering real-time processing. Moreover, the amount of internet traffic share already occupied by video streaming providers, such as YouTube, Hulu, HBO, Netflix, and so on, has already reached 50% of the total data flowing through internet (SUMMERS, 2016), and tends to increase even more in the next years, reaching more than 82% of total internet share by 2022 (CISCO, 2014), (CISCO, 2017).

The scenario presented drives the research for efficient tools to compress, store, and transmit high-resolution video sequences. Therefore, video coding standards have been developed and updated for that purpose. Currently, the H.264/AVC (ITU-T, 2003) is the most used coding standard for many of the all-day devices one may use (e.g., smartphones, smart-TVs, etc.), even though its first version is dated from 2003 (HEADJACK, 2018).

The evolution of increasing video resolutions (e.g., bigger television screens with more pixels within), of increasing sample rates (i.e., the frequency that the frames of a video are updated throughout the video processing flow), and constraints for real-time processing, due to the higher amount of data to be processed, led to research for new methods to compress video. For instance, the 4K resolution has 3840 x 2160 pixels within each frame of video, whereas 8K resolution has 7680 x 4320 pixels. Considering also that sample rates currently may range from 24 up to 300 frames per second. For example, a single second of a raw 8K video at 120 frames per second may generate up to 15-Gigabits of data. The HEVC (High-Efficiency Video Coding) is the H.264/AVC successor, being able to achieve up to twice the compression capabilities of H.264/AVC, whereas keeping the same subjective visual quality when compared to its predecessor (SULLIVAN, 2012). The first version of the HEVC standard dates from 2013, whereas extensions were further developed (ITU-T, 2013).

The new generation of video coding standards has proved to be more efficient regarding data compression compared to the older standards. Nevertheless, real-time processing for video encoding is still a major bottleneck when a software solution is used. The alternative is to use specific silicon devices to achieve the required constraints, where either FPGA (Field-Programmable Gate-Array) or ASIC (Application Specific Integrated Circuit) are suitable for that purpose. Therefore, the usage of accelerators is mandatory to achieve that real-time processing goal. Moreover, battery-based devices (e.g., smartphones) drive a potential scenario where the application of these hardware blocks occurs. Thus, a low-power-driven design is another crucial step towards an energy-efficient silicon architecture.

The video coding standards are composed of macro operations, which are the focus of research to accomplish efficient hardware designs for dedicated architectures to execute these specific operations. For instance, one may cite for HEVC encoder: inter-prediction (motion estimation and compensation), intra-prediction, transforms, quantization, filters, and entropy coding.

The HEVC standard allows only one type of entropy coding algorithm, which is the CABAC (Context-Adaptive Binary Arithmetic Coding) (MARPE, 2003), whereas its predecessor, the already cited H.264/AVC, allowed two algorithms: the CABAC, and the CAVLC (Context-Adaptive Variable Length Coding) (ITU-T, 2003). The usage of CABAC offers coding improvements when compared to CAVLC, at the cost of increased computational cost (SZE, 2013). For instance, the challenge to process more than one bit of data at once in CABAC is more difficult, since the previous data bit will often create dependencies for the current datum processed. Many studies and researches in the past few years have pointed out possible solutions for high-throughput CABAC hardware designs: (CHEN, 2010), (LIU, 2011b), (FEI, 2011), (PENG, 2013), (ZHOU, 2013), (VIZZOTTO, 2015) and (ZHOU, 2015), as examples. Hence, the next sub-section presents the motivation and the problem definition of this Thesis.

1.1 Motivation and Problem Definition

Video coding (where coding has the same meaning of compression within the scope of video processing) is a mandatory part of the current scenario of video storage and transmission, considering the scope of real-time processing for increasingly higher videos resolutions and sample rates. The HEVC standard is one of the most recent alternatives to accomplish efficient video processing (ITU-T, 2013), offering some new features to increase higher parallelism processing, such as the use of Tiles and WPP (Wavefront Parallel Processing). The previously mentioned techniques divide a frame (which is a single image of a given video coding sequence) into partitions using different approaches, where an entropy encoder instance can process each partition. Hence, it increases the throughput of the video coded bitstream generation, at the cost of a decrease in coding gains (i.e., a higher amount of bits are generated by the coded video when compared to a sequence which was coded without the parallel-processing techniques mentioned). For instance, the usage of Tiles and WPP lead to a loss in coding efficiency, which may decrease, on average, 3.73 % and 1.07%, respectively (CHI, 2012).

Nevertheless, the use of the parallel processing techniques implies in multiple instances of the CABAC block, which undoubtedly leads to an increase in area and power dissipation for the encoder as a whole, considering the case where one is using an ASIC approach for HEVC encoder. The use of specific accelerator leads to a suitable option to reach the real-time constraints for high-resolution video sequences processing, when compared to the software option. Hence, a single CABAC instance seems to be a reasonable alternative, in case that single instance can achieve a given throughput requirement for a target real-time video resolution processing, whereas also keeping the coding efficiency at an optimal value (for instance, around 800 Mbits/s for real-time level 6.2 high tier, as defined by the HEVC standard).

One may ask about the contribution that the CABAC has within the HEVC encoder. In fact, the predictions (inter or intra) are the most time-consuming step of HEVC encoder, ranging from 42.5% up to 88.7% of total encoding execution time (WON, 2015). Nevertheless, the impact of CABAC in execution time is not negligible, being able to reach up to 11% of the total time contribution, with an average contribution of 4.7% (WON, 2015). Furthermore, CABAC behavior implies in heavily computational complexity, which affects, for instance, in its parallelization opportunities to increase throughput, or for efficient rate-estimation for mode decision (WON, 2015), becoming a potential bottleneck of the encoding flow, therefore. The reported data are another reason towards a high-performance hardware HEVC entropy encoder, as attested by the related CABAC designs cited before.

A power-saving CABAC approach is a desirable goal, considering that video encoding occurs within battery-based devices, such as smartphones, and they may be delivered in real-time for a live broadcast, for instance. HEVC seems to be more energy consuming than the previous generation of video encoding standards (MONTEIRO, 2015), where significant power dissipation increase is reported. Thus, since the HEVC encoder as a whole consumes more power, one may consider that it is an appropriate research focus to diminish the dissipated power in all encoding steps, including the entropy encoding.

Detailing some important literature efforts within the context presented before, in the work of (LIU, 2011b) we find one of the first alternatives to increase throughput for a CABAC hardware design, where renormalization steps required by the CABAC block runs in a single cycle. This work was done in the context of the previous H.264/AVC standard, but its architectural proposal is still valid for the CABAC of the HEVC standard since it affects the main sub-block of the entropy coding, which has not changed from one standard to the other: the Binary Arithmetic Encoder (BAE). Moreover, in one of the most recent and successful works concerning ultra-high-throughput results, the work of (ZHOU, 2015) presents many architectural novelties, gathering also prior-art proposals of (FEI, 2011) and (ZHOU, 2013), which will be further presented in this Thesis. Thus, the work of (ZHOU, 2015) is the up-to-date state-of-the-art in terms of CABAC design, which is able to reach the constraints for real-time 8K UHD (Ultra-High Definition) video resolution (CHEN, 2015).

The mentioned works point the interest of recent research for a high-throughput single-core CABAC hardware design, even with the parallel novelties the HEVC standard has proposed. Nevertheless, none of them focuses also on low-power design or low-power opportunities in their designs, which is, as one may notice, a relevant field of research for hardware video processing architectures. Moreover, since no recent work deals with the power dissipation issue, there is also no relevant data to use as a reference on where or how to seek for power reduction opportunities within the CABAC block in an efficient manner.

State-of-the-art CABAC design already surpasses the minimum requirement for the HEVC standard defined maximum constraint (i.e., the 6.2 Level at High tier). However, future real-time requirements can potentially be even more restrictive and far beyond the current ones, which will demand newer high-throughput approached for that reason. For instance, the intended successor of HEVC is already under development, named Versatile Video Coding (VVC) (BROSS, 2018). Power dissipation is crucial to take into account, considering the scope of battery-based encoding architectures. Considering that future scenarios may demand higher processing capabilities, which will consume more power due to the hardware increase to cope with these hypothetical scenarios, power-saving approaches seem necessary along with the performance-driven design.

Hence, some points may be drawn considering the global scope of this Thesis, which provide directions for the research described herein:

- Is it possible to increase the throughput of a CABAC hardware design beyond the current state-of-the-art for future real-time video processing constraints, especially considering a type of input data that suffers from fewer dependencies and thus is potentially more feasible to parallelize its processing? If possible, how is the implementation of this approach? What are the potential gains in terms of performance?
- Are there modules which may potentially be dissipating power which otherwise could be turned off? How to assure that these modules would, under

a real environment, save power effectively? Does this approach, in case possible, affects the throughput of the CABAC block?

- Is it possible to integrate into a configurable high-throughput low-power CABAC design, where, depending on the characteristics of a given video sequence being processing, an on-the-fly choice can be made considering the best throughput and energy required trade-off, turning on or off parts of the architecture? For instance, state-of-the-art CABAC designs focus on reaching throughput constraints for real-time 8K UHD video processing. However, considering that this same ultra-high-throughput architecture would process video sequences with smaller resolutions (e.g., Full-HD), would it not be overestimated for this resolution and therefore be dissipating additional power/energy? Moreover, for future real-time video constraints, is possible to have an energy vs. throughput CABAC design, which presents the best trade-off between the referred variables and still achieves the real-time requirements?

1.2 Main Contributions of the Thesis

The scope of this Thesis is the entropy-encoding step, according to the HEVC coding standard, which is the CABAC block. The reasons to focus on CABAC were presented previously (e.g., difficulties in the processing parallelization, being a potential bottleneck of the HEVC encoder flow; and the advantages of a single high-throughput CABAC instance design for power saving and coding efficiency). Efficient architectural for the critical part of the algorithm (i.e., the BAE block), targeting ultra-high-throughput and low-power design, are the primary goals, where the global objective is a configurable entropy encoder block, considering the best trade-off configuration between the two variables for the possible gamma of different video sequence characteristics: throughput and power.

For the desired objectives, recommended video sequences analysis (BOSSEN, 2013), running under the HEVC reference software HM (HM, 2016) was performed, in order to analyze and to validate novel techniques targeting increasing throughput for the CABAC block. Moreover, statistics were gathered for insertion of fine-grain low-power efficient techniques, based on the data assessed, reducing the power dissipation of designed BAE architectures. Furthermore, a novel scheme for parallel processing of bypass bins is derived (i.e., a type of CABAC data with less dependencies for processing), and its efficiency is measured in the BAE hardware design described as a gate-level netlist. Additionally, the integration of prior-art high-throughput approaches within a BAE block, along with the novel scheme for multiple-bypass bins processing of this Thesis, is carried out followed by simulations and hardware synthesis results. Finally, based on the multiple-bypass bins novel BAE approaches, an energy-throughput configurable BAE architecture is introduced for the first time. This approach can deliver possible current and future scenarios real-time throughput, and keep the energy consumption on an optimal level, when compared with the non-configurable BAE circuits, by adding a feature that isolate parts of the architecture and thus avoids extra power dissipation.

As a summary, the main contributions presented in this Thesis are as follows, and are depicted in Figure 1.1 in a macro point of view within a zoomed-in CABAC block diagram (one may notice that additional contributions also appear in Figure 1.1, which are shown in the Appendixes of this Thesis):

- **Statistical Analysis for Low-Power Opportunities (Chapter 4):** recommended video sequences were run in the HM software, and the generated statistics pointed out possible power-saving possibilities within a hardware BAE block.
- **Low-Power High-Throughput Binary Arithmetic Encoder Proposal (Chapter 4):** using the statistics gathered for low-power design, a Low-Power High-Throughput BAE architecture is presented, for which gate-level netlist power values are shown along with the power savings achieved by the use of the fine-grain low-power insertions herein.
- **Novel Multiple-Bypass Bin Processing Scheme - MBBS (Chapter 5):** a novel scheme is proposed, based on the behavior of CABAC variables that controls the processing during video encoding flow, to increase the throughput of hardware BAE design. The proposed approach takes into consideration the increasing amount of a specific data type of HEVC encoded video sequences (i.e., the bypass bins), which has smaller dependencies among them and, therefore, are more feasible to be processed in parallel.
- **Ultra-High-Throughput Binary Arithmetic Encoder Architecture using MBBS (Chapter 6):** combining prior-art techniques from literature along with the novel MBBS, a new BAE architecture is presented, having the highest throughput found in the literature, whereas the throughput results come from

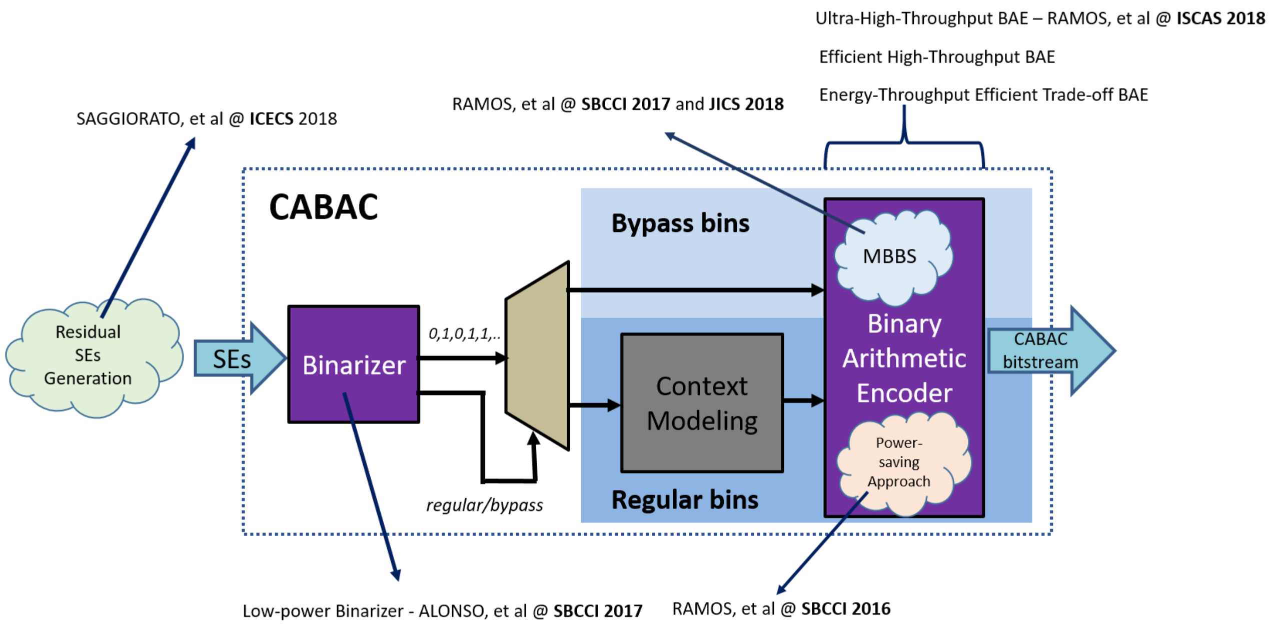
recommended video sequences using different parameters settings (BOSSEN, 2013).

- **Efficient High-Throughput Binary Arithmetic Encoder using MBBS (Chapter 6):** a second BAE architecture is proposed, presenting a faster time-to-market design, where the usage of MBBS achieves a similar throughput to prior-art BAE designs, at the advantage of less penalty in terms of area and potentially power.
- **Efficient Energy-Throughput Trade-off Binary Arithmetic Encoder Architecture and Methodology of Use (Chapter 7):** as the last novel contribution of this Thesis, the Efficient High-Throughput BAE presented in Chapter 6 is assessed, by analyzing smaller versions of the original design. In the end, together with the low-power approach of Chapter 4, a configurable BAE architecture is presented, using on-the-fly different cores, accomplishing potential energy efficiency for current and future real-time video processing scenarios, keeping the real-time throughput requirements.
- **Power-saving Binarization Design (Appendix A):** a power-saving Binarization block is presented, and the power results measured and compared with related literature works, attesting the power gains of the proposal.
- **Efficient Residual Syntax Elements Generation Architecture for High-Throughput CABAC Designs (Appendix C):** an approach to deliver the data that corresponds to the major input contributor of CABAC is presented, assuring sufficient input data rate and avoiding recent CABAC designs to starve of incoming symbols.

1.3 Outline

The text is organized as follows: **Chapter 2** presents the background concepts for video processing, video coding, the HEVC standard, information entropy and arithmetic coding basics, and power-consumption in CMOS circuits. **Chapter 3** gives an overall description of the CABAC algorithm, its main sub-steps, and variables, along with details of the most relevant reference CABAC works found in the literature. The statistical analysis to assess the opportunities for low-power insertions into a BAE design and the proposed BAE architecture using the low-power approach driven by the statistical analysis appear in **Chapter 4**, along with preliminary synthesis results. The novel Multiple-Bypass Bins Scheme definitions are presented in **Chapter 5**, along with the throughput gains achieved by the use of the technique into a baseline BAE design. **Chapter 6** presents the two new ultra-high-throughput proposed BAE architectures using the MBBS and other prior-art techniques, along with simulation and synthesis results, and comparisons with related relevant works. The efficient energy-throughput BAE design (gathering techniques and proposals from previous chapters of this Thesis), along with the methodology of use and the synthesis results related appear in **Chapter 7**. **Chapter 8** discusses the results obtained at this Thesis, along with future works within the same scope. **Appendix A** presents the low-power Binarization architecture, and **Appendix B** a different approach to explain the MBBS inception. **Appendix C** presents the residual Syntax Element architecture proposed within the context of high-throughput CABAC design, whereas **Appendix D** contains support data for the architecture proposed in Chapter 7.

Figure 1.1 – Main Thesis contributions


Ultra-High-Throughput BAE – RAMOS, et al @ ISCAS 2018
Efficient High-Throughput BAE
Energy-Throughput Efficient Trade-off BAE
RAMOS, et al @ SBCCI 2017 and JICS 2018
SAGGIORATO, et al @ ICECS 2018
CABAC
Bypass bins
MBBS
Binary Arithmetic Encoder
Residual SEs Generation
SEs
Binarizer
0,1,0,1,1,..
Context Modeling
CABAC bitstream
Power-saving Approach
regular/bypass
Regular bins
Low-power Binarizer - ALONSO, et al @ SBCCI 2017
RAMOS, et al @ SBCCI 2016

Source: the author.

2 VIDEO PROCESSING, ARITHMETIC CODING, AND POWER DISSIPATION CONCEPTS

This chapter will describe basic concepts related to video processing, such as how a video is composed, the color spaces that may be used. The HEVC standard, along with its basic concepts, also appears in this chapter. Furthermore, the depiction of the primordial explanation on the concept of entropy from communication theory, along with the basic understanding of general arithmetic coding, appears in this chapter. Additionally, the backbone elements regarding power consumption on CMOS circuits are presented, along with the techniques related to power saving on the architectural level of implementation.

2.1 Digital Video Basic Concepts

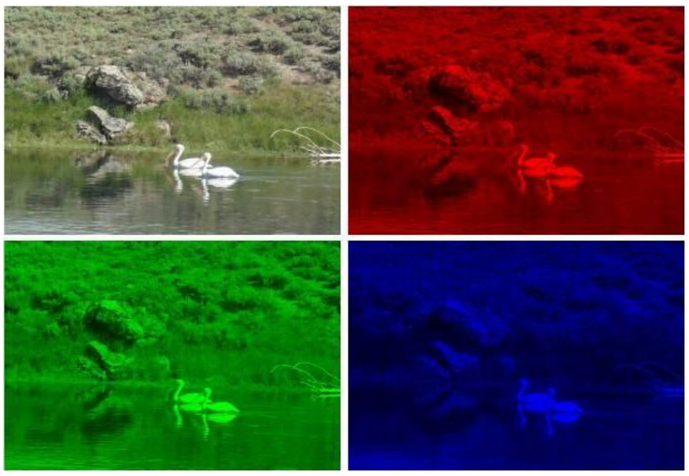
A digital video is a composition of several images that appear in sequence, which intends to emulate to the viewer the sensation that movement is occurring during those images presentation. The images are referred as frames, and a given sampling rate for those frames shall be achieved to transmit the mentioned sensation of movement to the viewer. The minimum sampling rate is 24 frames per second (RICHARDSON, 2010). Higher sampling rates (e.g. 30, 60, 120 frames per second) present smoother sensation during the transition between frames and are suitable options when the desired smoothness is a goal (RICHARDSON, 2010).

The basic primitive or cell of a frame is a pixel, which represents the color intensity that minimal point into a given picture space has. The more pixels a frame has, the more accurate compared to reality is that frame. The resolution of a video is the number of pixels we have into both horizontal and vertical axis. For example, 1920x1080 pixels (Full-HD), 2560x1600 pixels (WQXGA), and so on. A pixel is a composition of more than one aspect of color or brightness, which is called color space. For instance, a widely used color space is the RGB (Red Green Blue), where each pixel is composed of a value for the red, for the green, and for the blue color intensities (called color components). The composition of the values for the three colors mentioned forms a wide variety of color the human eye is able to recognize. Figure 2.1 presents an example of RBG image composition.

Another color space is the YCbCr (Luminance Chrominance Blue and Chrominance Red), also called YUV (MIANO, 1999). For that color space, the pixel composition is formed by a brightness/light (luminance) component (i.e., Y), and two color components (i.e., Cb and

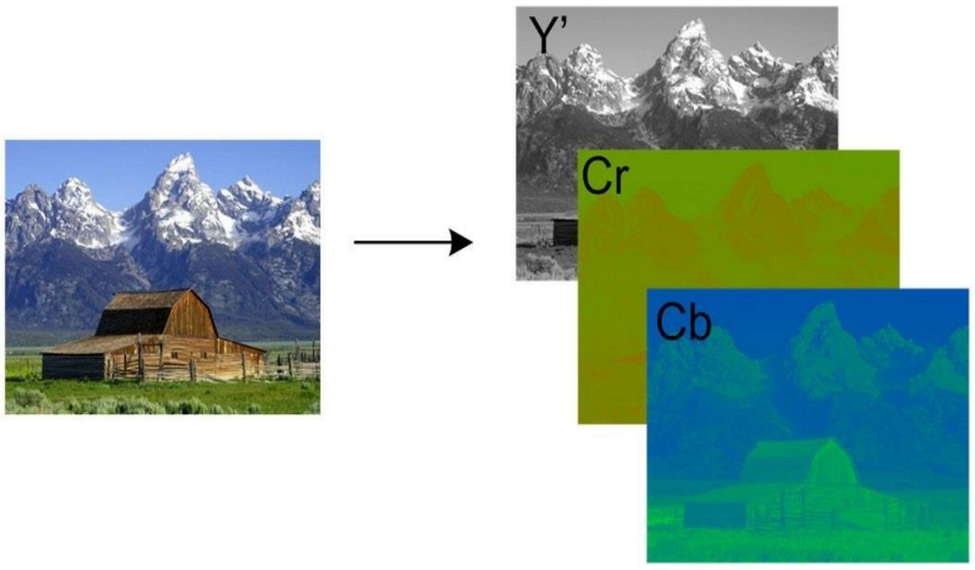
Cr, or U and V). The main difference between the YCbCr compared to RGB is the disassociation of the luminance and color components at YCbCr. Moreover, the fact that the human eye is more sensitive to the light than to color (RICHARDSON, 2010) leads to an interesting potential for subsampling of the different components, without prejudice to the subjective visual information a frame has. Figure 2.2 presents an example of YCbCr composition into an image.

Figure 2.1 – RGB image decomposition



Source: http://www.augustrs.com/fractal-camouflage/

Figure 2.2 – YCbCr image decomposition

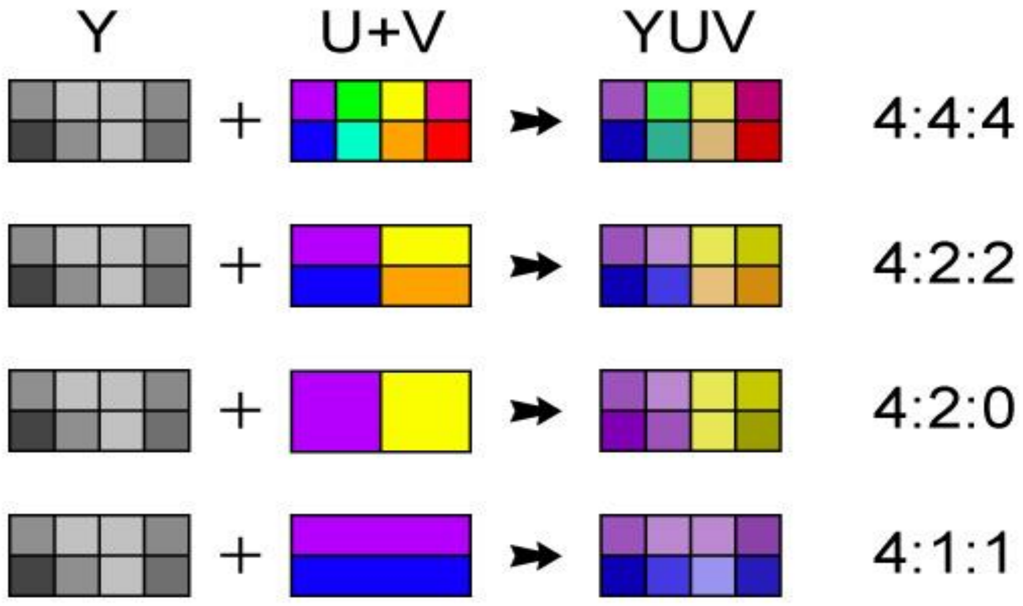


Source: https://hisour.com/pt/ycbcr-color-spaces-26075/

Each color component representation is as a binary value into an n-bits vector. The more bits this vector has, the wider the variety of colors an image can represent. For instance, 8-bit pixel representation leads to 256 possible colors possible, whereas 32-bit pixel representation leads to more than four billions colors. Nevertheless, the distribution of information among the components of given color space may vary. One may notice that, since the human eye is more sensitive to the luminance component compared to the color components of a YUV color space, the amount of information for the color components could be smaller than the luminance. For instance, for every four samples of the component Y, one may have a single sample for U and a single sample for V. This kind of processing is called sub-sampling (also known as pixel decimation). Below are presented some alternatives to sub-sampling, whereas Figure 2.3 presents the sub-sampling formats visually:

- **Sub-sampling 4:4:4**: For every four Y samples, also four U and four V samples exist (in other words, there is no sub-sampling). This pattern is used for high-fidelity applications since no loss will occur by using it.
- **Sub-sampling 4:2:2**: For every four horizontal Y samples, two U and two V samples exist (i.e., there is no sub-sampling in the vertical axis).
- **Sub-sampling 4:2:0**: For every four Y samples, there will be one U and one V samples. This sub-sampling pattern leads to 50% of data saving and is the most used when comes to video compressions (RICHARDSON, 2010).

Figure 2.3 – Sub-sampling examples



Source: http://www.ravepubs.com/chroma-subsampling/

2.2 Digital Video Encoding

Digital video encoding has the same meaning as video compression. The main reason for video encoding is the prohibitive amount of data a raw video generates, which leads to increasing demand for video storage. Moreover, in the context of video transmission through an internet channel, the bandwidth would be extremely limited for transmission of raw video, for instance. For example, Table 2.1 presents the estimated data amount required for storage and the required bandwidth for raw 10 minutes videos at different resolutions, considering 12-bits for each color component (MONTEIRO, 2017).

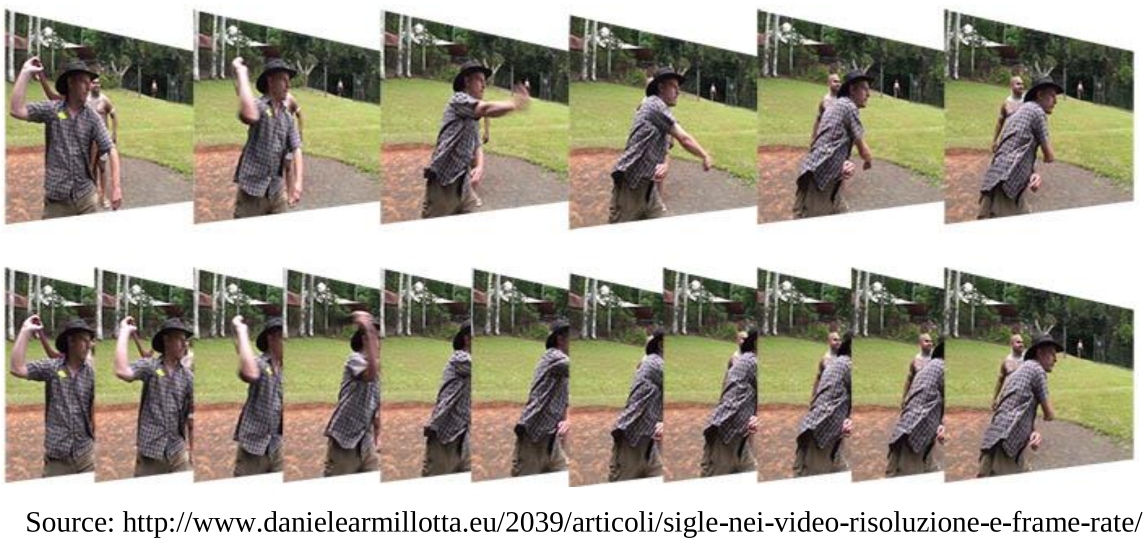
Table 2.1 – Examples of raw videos size and required bandwidth

Resolution	Size (GBytes)	Bandwidth (Mbytes/s)
834x480	10	17.13
1920x1080	52	88.98
2560x1600	103	175.78
3480x2160	189	322.58

Source: (MONTEIRO, 2017).

The basic leverage of video encoding is to explore redundancies in the image. For instance, Figure 2.4 shows a video sequence composed of a certain amount of frames. One may notice that, within a single frame, many pixels will have a very close (if not the same) color. Moreover, it is easy to observe by comparing the frames within the pointed sequence that some pixels are just “moving” between one frame to the other (possibly keeping the same color during this “movement,” or with a slight difference compared to the pixel from the previous frame). Therefore, there is no need to send the same or closely the same information related to pixels among different frames. One may sample the closely related value of a neighbor pixel from the current or a different frame and send this information just once. Hence, the redundancies categories among images within a video sequence are spatial, temporal, and entropic.

Figure 2.4 – Example of video sequence decomposition into frames



Source: http://www.danielearmillotta.eu/2039/articoli/sigle-nei-video-risoluzione-e-frame-rate/

2.2.1 Spatial Redundancy

Neighbor pixels within the same frame can have the same or a very similar color. Therefore, an encoding process could send the information of a chosen pixel inside that similarity group, and consider the color values of this chosen pixel as the base value. The information sent from the other pixel within the similarity set is the difference between their color values and the base value of the chosen pixel, named residues. Since the color difference tends to be minimal if a suitable group of pixels is selected, the difference values that are sent instead of the original pixel color values are smaller and, therefore, compression is accomplished. This type of redundancy is also known as intra-frame redundancy (GHANBARI, 2003). Usually, the intra-frame redundancy is the chosen to be applied when no other type of redundancy is available (for instance, at the very first frame of a video sequence), or in case random insertion points are required during the video encoding.

2.2.2 Temporal Redundancy

Pixels among different frames can simply “move” from one to another (i.e., from one frame to another, a given pixel is at another position within a different frame). Nevertheless, the color values of this pixel have just moved and can have changed slightly or have not changed at all. Hence, the same reasoning derived from the spatial redundancy can be applied, with one extra information: the encoding process has to discover the amount of movement that pixel (or group of pixels) has undergone, and the difference of values from the original pixel (or pixels), i.e., the residues. This temporal redundancy is also known as inter-frame redundancy (GHANBARI, 2003). The inter-frame redundancy is the major contributor for the data savings when comparing the original raw video to the final encoded sequence at the cost of being the bottleneck for video encoding performance (ZATT, 2012).

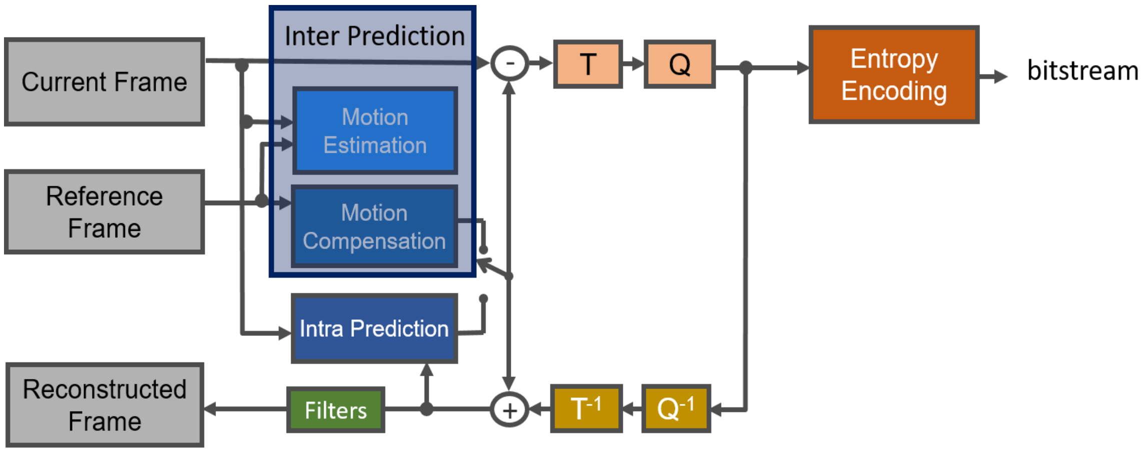
2.2.3 Entropic Redundancy

The derivation of this type of redundancy comes from statistical symbols distributions within given video sequences, instead of the contents of the images (BHASKARAN, 1997). For instance, the pixels movements among different frames related to the inter-frame redundancy have to be encoded using some binary representation. Furthermore, the statistics of whether there will be more zeros or ones for a given video sequence can be known prior to the encoding process and can be updated during the encoding. Hence, the symbols that are more probable to occur will have shorter bit representation, whereas symbols that are less probable to occur will have a longer bit representation, saving bits at the final bitstream generation, without data loss. This type of redundancy is the focus of this Thesis since it is the concept behind the CABAC entropy algorithm within the HEVC standard.

2.3 HEVC Video Coding Standard

The HEVC video coding standard is a relatively recent alternative for digital video compression (ITU-T, 2013). It is the successor of H.264/AVC (ITU-T, 2003), where it achieves twice the coding capabilities of its predecessor, keeping the same visual subjective quality (SULLIVAN, 2012). The inception of HEVC standard is the result of a new era of High Definition (HD) video resolutions, demand for video streaming through the internet, and urge for better compression capabilities. The HEVC is a hybrid-block based standard (RICHARDSON, 2002), where it is composed by some macro-operations, based on the redundancies concepts presented before. Figure 2.5 depicts the block diagram of the generic HEVC encoder, where its blocks are: (i) Intra-prediction and inter-prediction (Motion Estimation and Motion Compensation) (ii) Transforms; (iii) Quantization; (iv) Filters; and (v) Entropy Encoding, which will be further explained.

Figure 2.5 – HEVC block diagram



Source: the author, modified from H.264/AVC diagram in (AGOSTINI, 2007).

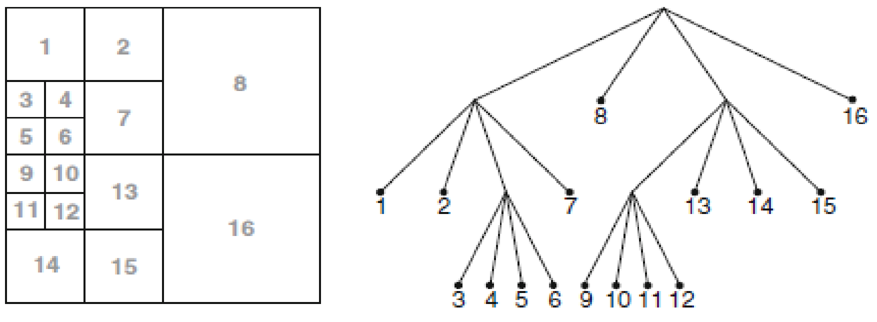
One may notice that the HEVC encoder has Inverse Transform and Quantization steps, and then the filtering steps occur. The requirement of the inverse steps comes from the necessity to keep the same reference frames (i.e., with the same pixels values) to be used for redundancy discoveries when different frames are used (i.e., inter-frame redundancy). The reason for that is simple: Quantization step inserts losses in the encoding process (i.e., some data after this step cannot be regenerated to their original value, as will be further explained). Hence, in case the currently processed frame is used as the reference for pixels of not-yet-processed frames in the decoder side, the same values have to be used at the encoder side, which implies the usage of the reconstructed frame after the losses insertion (i.e., after the Quantization step).

2.3.1 Coding Tree Units and Subdivisions

Frames are two-dimensional arrays of millions of pixels. Therefore, the HEVC splits the image into smaller blocks of pixels, and the encoding occurs at this level. The splitting starts at 64x64 group of pixels, whose name is Coding Tree Units (CTUs), but can also be groups of 32x32 and 16x16 pixels. The update comparing HEVC to the H.264/AVC is the inflation of the prior macro-blocks (block of 16x16 pixels) up to this new CTU abstraction maximum size. One of the reasons for the increase in the size of the CTU compared to the macro-blocks is the increase in the video resolutions of the new era of video processing (e.g., 4K, 8K) that did not exist at H.264/AVC development time. Thus, the 64x64 pixel structure is a more suitable approach for processing of this vast amount of pixels for Ultra High Definition (UHD) video resolutions (SZE, 2014).

The CTUs can be further split into smaller units, called Coding Units (CUs), which can be as large as the root CTU itself, or can be further split at the bottom level of 8x8 group of pixels, using the abstraction of a Coding Unit (CU) Tree structure. Hence, groups of pixels that present more similarities or correlations are processed together in order to achieve a more efficient redundancy finding process (which is called Prediction). Figure 2.6 depicts the possible values of CTUs and CUs divisions and sizes within the HEVC context and the correspondent CU Tree.

Figure 2.6 – Example of CUs division and CU tree structure

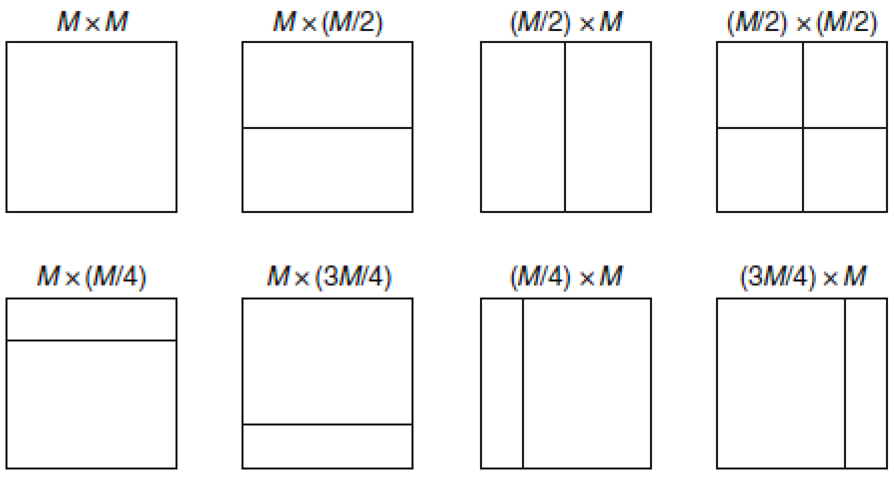


Source: (SZE, 2014).

2.3.2 Predictions

The Predictions are the process to find pixels that present more similarities with the currently processed set of pixels. The Prediction can either be within the same frame (i.e., intra-frame prediction) or among different frames (i.e., inter-frame prediction). The input to the prediction process is the Prediction Units (PUs) which are a group of pixels derived from the current processed CUs. Figure 2.7 illustrates the possible sizes for PUs in an inter-frame prediction process, which, as one may notice, could be Symmetric Motion Partitions (SMP) or Asymmetric Motion Partitions (AMP), where M is the dimension of a CU. The intra-prediction process allows only 2Mx2M and MxM divisions.

Figure 2.7 – PUs possible divisions



Source: (SZE, 2014).

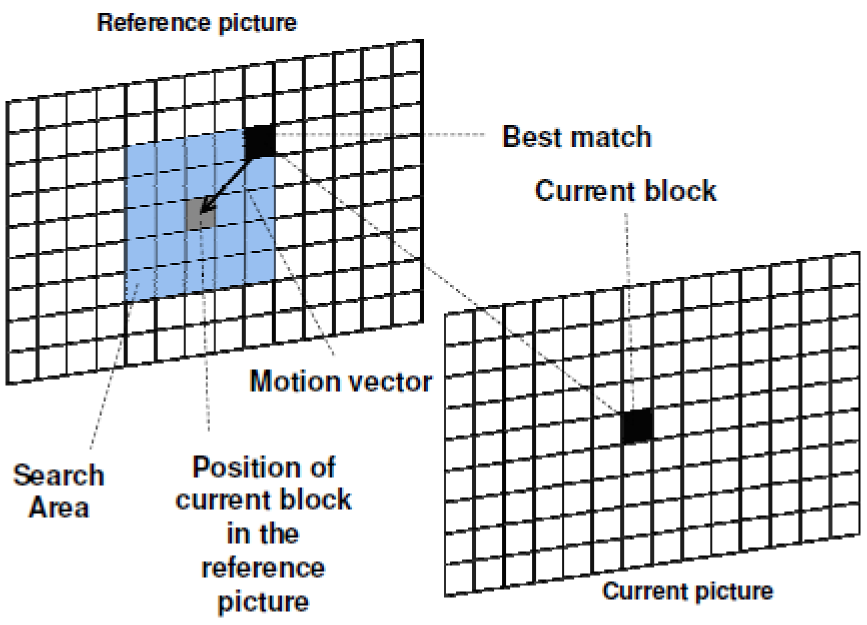
2.3.2.1 Intra-frame Prediction

The intra-frame prediction uses pixels located at the same frame the current encoding pixels are (i.e., intra-frame redundancy). HEVC updates the prediction modes; allowing 35 directions to find the most suitable pixels to be used as the reference for the current processed ones (i.e., to find the pixel base color value that has more similarities with the current pixel). The intra-frame prediction will be used at the first frame of a given video sequence, or for random point access during video encoding process. Nevertheless, an HEVC encoder could support only intra-frame prediction, for instance, since HEVC defines only how the decoder has to work (i.e., the HEVC decoder has to support all possible configurations any given HEVC encoder may allow) (ITU-T, 2013).

2.3.2.2 Motion Estimation

A pixel (or pixels) can change position across frames in a sequence. The inter-frame prediction is responsible for finding the more suitable groups of pixels in the different reference frame (or frames) that can be used as the reference value for the current group of pixels (i.e., the current PU). The process within the inter-frame prediction responsible for that is the Motion Estimation (ME). The ME tries to find the group of pixels in different frames where the difference of color values compared to the current PU is the minimal possible, if not zero. Therefore, the generation of a Motion Vector (MV) occurs, indicating the amount of motion (movement) the pixels at the other frames have dislocated compared to the current one (i.e., the change in the horizontal and vertical axis of the image). Figure 2.8 depicts the ME behavior. The ME is the most complex operation of HEVC encoder but is where most of the encoding gains occur (CHEN, 2007), (ZATT, 2011).

Figure 2.8 – ME behavior



Source: (PORTO, 2008),(DINIZ, 2015).

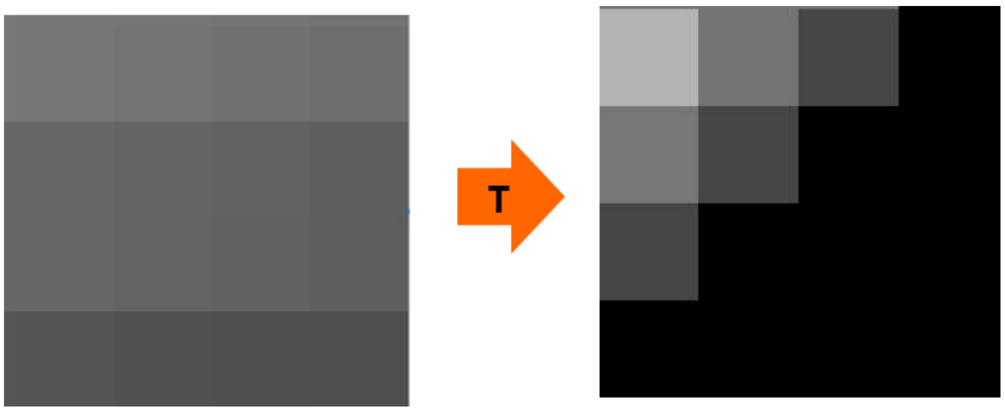
2.3.2.3 Motion Compensation

During the frame reconstruction, in case it will be used as the reference for future frames at both encoder and decoder sides, the current already encoded frame has to be stored with the losses inserted by the Quantization step. At this point, the encoder has two informations to reconstruct the current frame after the losses insertion: the MV generated at ME, to indicate how much the pixels of the current frame have moved compared to another previously processed frame; and the color difference values between the current and the previously referenced frame (i.e., the residues). The Motion Compensation (MC) uses this information, along with defined interpolation processes, thus generating the color values of the currently processed frame, with the mentioned Quantization losses.

2.3.3 Transforms

The Transform process consists to rearrange the residues in a fashion that only some few coefficients (i.e., the values after the Transform step) remain significant (i.e., remain with a value different from zero). Therefore, coding gains occur by representing the residues with fewer values compared to before the Transform process. The residues are the already mentioned difference value between the pixels of the current encoded frame with the predicted pixels from the own current frame (intra-prediction); or from a previously processed frame used (inter-prediction). These residues come from the CU, and are arranged in Transform Units (TUs), where the size of the TUs are 32x32, 16x16, 8x8, and 4x4 pixels, in which the abstraction of the Residual Quad-Tree (RQT) is used for the splitting of the TU. Figure 2.9 illustrates the Transform process in a 4x4 block of residues.

Figure 2.9 – Visual example of Transform process in a 4x4 block of residues



Source: the author.

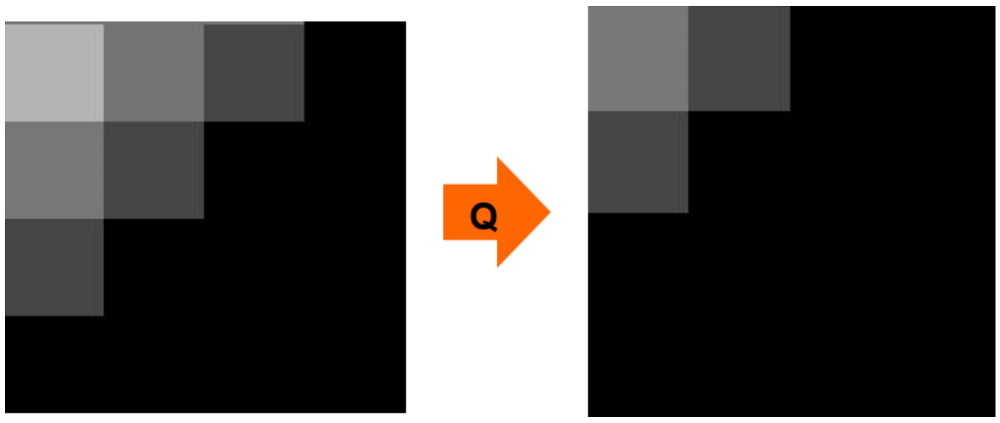
The Transform process is based in the Discrete Cosine Transform (DCT), using an integer approximation for a better feasibility of a simpler hardware implementation. The Discrete Sine Transform (DST) is also applied to a specific situation for Intra-prediction.

As one may notice, the Inverse Transform process also occurs. The reason already presented is the reconstruction necessity of the currently processed frame with the losses insertion by the Quantization step. Therefore, the Inverse Transform will reconstruct the coefficients values back to the residues. Since losses have occurred, these reconstructed residues are not the same as the ones used by the direct Transform process.

2.3.4 Quantization

The Quantization process has the same effect of an integer division on the coefficients generated by the Transform process. Hence, at the end of the Quantization step, all quantized coefficients will have a smaller value compared to the pre-quantized coefficients. Furthermore, the smaller coefficients tend to have value zero after this step, and that is the reason losses are insertion here since the zeroed quantized coefficients cannot be regenerated back to their original values. Figure 2.10 illustrates the Quantization step after the Transform operation has occurred.

Figure 2.10 – Visual example of Quantization process in a 4x4 block of residues



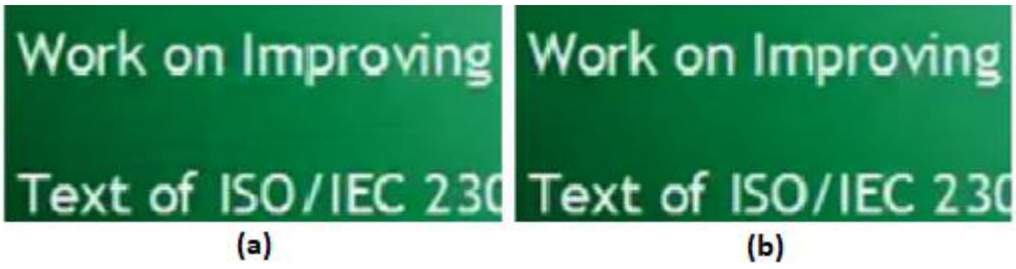
Source: the author.

The Inverse Quantization is the inverse process, as the name says, the quantized coefficients return to their values before the Quantization, but now with the losses inserted by direct Quantization step. The reason is to keep the same reference frame values in both HEVC encoder and decoder sides since the decoder will have only the frame with the inserted losses as possible reference frames.

2.3.5 Filters

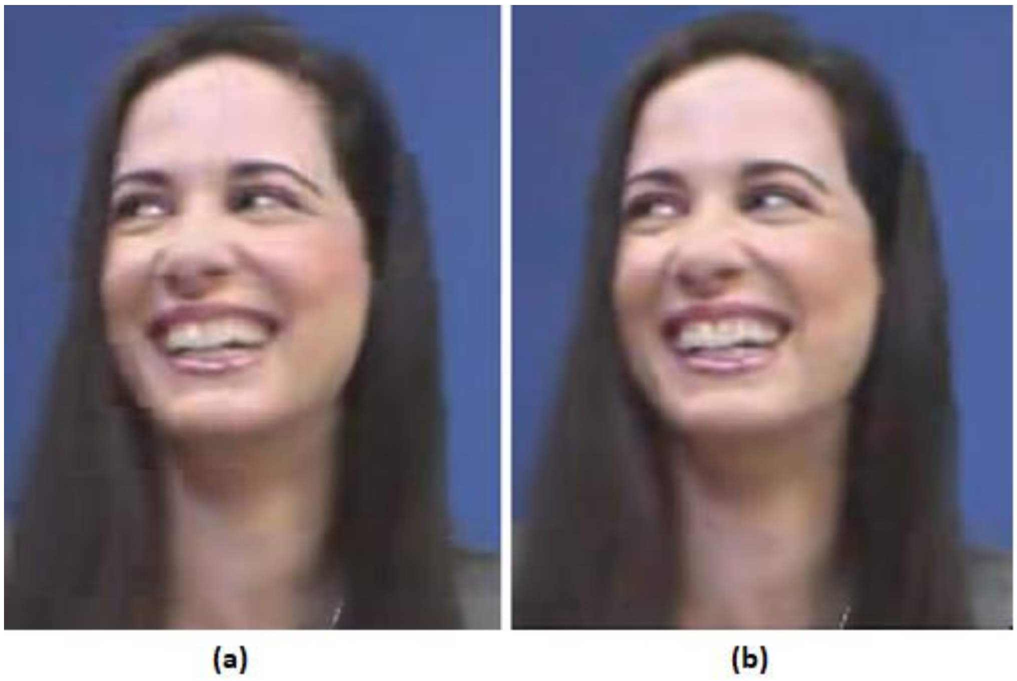
Filtering processes may be required at the reconstruction of the currently processed frame to be used as a future reference for Inter-prediction, due to insertion of block or ringing artifacts generated due to a high Quantization step applied. The usage of filter brings smoothness to parts of the image that otherwise would appear distorted after the Quantization. HEVC allows two options of Filters: Deblocking Filter, and Sample Adaptive Offset (SAO) Filter. Figure 2.11 illustrates an example of a reconstructed image (a) without and (b) with the use of SAO Filter, whereas Figure 2.12 depicts an example of a reconstructed image (a) without and (b) with the use of the Deblocking Filter.

Figure 2.11 – Example of SAO Filter application



Source: (SZE, 2014).

Figure 2.12 – Example of Deblocking Filter application



Source: (SZE, 2014).

2.3.6 Entropy Encoding

The Entropy Encoding process functionality is to generate the final encoded video bitstream, by considering that there are the most probable and the least probable symbols occurring throughout the encoding process. The most probable symbols or values will affect into the generation of fewer bits of the final bitstream compared to the least probable ones, by using an arithmetic encoding algorithm. In HEVC, only one type of Entropy Encoding is allowed, which is the CABAC (Context-Adaptive Binary Arithmetic Coding) (MARPE, 2003). As already said, CABAC is a binary arithmetic encoding algorithm, in which the occurrence probabilities of the processed symbols are updated according to the current incoming amount of these symbols (i.e., context-adaptive). CABAC input is all the data generated at the HEVC previous encoding steps, which receive the name of Syntax Elements (SEs). For instance, the MV, the Transform residues, and all previous encoding steps, will generate information about its contents in the form of different SEs, which later will feed the CABAC block.

The CABAC block in the context of HEVC encoding process is within the main scope of this Thesis. Therefore, Chapter 3 will explain in more details how the CABAC algorithm works. Nevertheless, the primitive elements of the entropy idea from communication theory and the arithmetic coding fundaments appear on the next section of this chapter.

2.4 Communication Theory and Arithmetic Coding Basic Concepts

2.4.1 Entropy Definition for the Communication Theory

As defined in the seminal work on the Mathematical Theory of Communication (SHANNON, 1948), let one consider a given set of n symbols that transport information. The symbols have different probabilities to occur throughout their transmission, which are known. Thus, as defined by Shannon, the entropy H in this scenario is given by (2.1), where p_i is the probability of each symbol to happen during the transmission.

$$H = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2.1}$$

The entropy H, which is stated as the amount of “choice” to encode the cited transmission, can be traduced to the number of average bits necessary (thus the $\log_2$ on the above equation) to transmit the related symbols. For instance, reproducing an example presented in (SHANNON, 1948), one may imagine an alphabet with only two symbols, where one has the probability p to occur, whereas the other q = 1 – p. The related entropy H for this example is as depicted in (2.2):

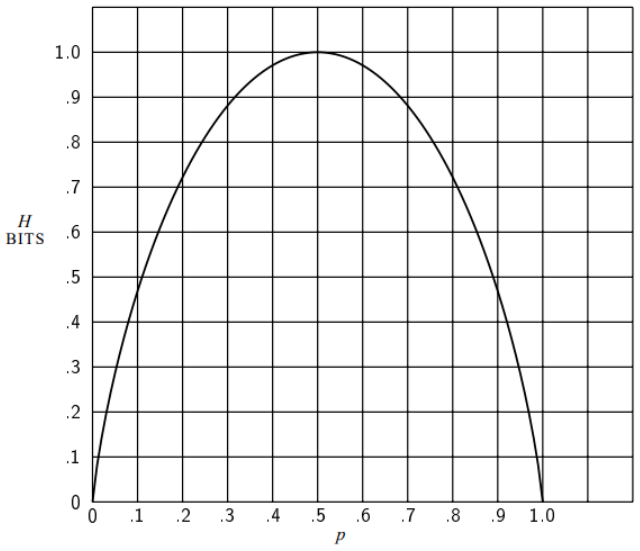
$$H = -(p \log_2 p + q \log_2 q) \tag{2.2}$$

The figure 2.13 shows the H as a function of p, in which the axis Y shows the amounts of bits (i.e. the entropy H) according to the variation of p. What one may conclude, expanding the results in (2.2) and figure 2.13, that H is zero when all but one p_i has value one. Moreover, the maximum H happens when the entire p_i are equal (i.e., all symbols have the same probability to occur). A wider variation on the symbols probabilities leads to a lower entropy value and thus a smaller average (SHANNON, 1948).

For the sake of understanding of the last concept presented, another example of (SHANNON, 1948) is used for that purpose. Let one consider a certain source, which has four letters to carry the information: A, B, C, and D. The occurrence probabilities of these symbols are respectively $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and $\frac{1}{8}$. The related entropy H to this example is as depicted in (2.3):

$$H = -(\tfrac{1}{2}\log_2 \tfrac{1}{2} + \tfrac{1}{4}\log_2 \tfrac{1}{4} + \tfrac{2}{8}\log_2 \tfrac{1}{8}) = 7/4 \text{ (i.e., 7/4 bits per symbol, in average)} \qquad (2.3)$$

Figure 2.13 – Function H of (2.2)



Source: (SHANNON, 1948).

The related binary codes to achieve the average number of bits per symbols may be as followed, respecting the method presented in (SHANNON, 1948) for that purpose): A = 0, B = 10, C = 110, and D = 111. The maximum entropy is accomplished when all the letters have the same probability to occur (i.e., ¼), in which H = 2. For that case, following the proposal in (SHANNON, 1948) the following codes apply to the related alphabet: A = 00, B = 01, C = 10, and D = 11.

2.4.2 Arithmetic Coding

Arithmetic Coding is a fashion to represent a piece of information in a compacted fashion. In contrast with the traditional Huffman algorithm (HUFFMAN, 1952), which gives a discrete code for every symbol of the information, Arithmetic Coding does not have this restriction. Instead, the whole message is codified together and, therefore, there is no guarantee that any individual symbols will have a specific code for it (MOFFAT, 1998). The basic principle is to maintain the individual probabilities for each symbols throughout the processing.

The fashion Arithmetic Coding works requires the occurrence probability of each symbol of the information alphabet. Let consider again that p_i is the probability occurrence of the i-th symbol of the alphabet; variables Range is the product of probabilities of these symbol, whereas Low is the smallest value compliant with the code of the already processed symbols up to a given moment. Range and Low are initialized with 1 and 0, respectively. For the next symbol to be encoded, which one may consider the j-th symbol of the alphabet, Range and Low shall be updated as depicted in (2.4) and (2.5), respectively:

$$Range = Range * p_j \qquad (2.4)$$

$$Low = Low + Range * \sum_{i=1}^{j-1} p_i \qquad (2.5)$$

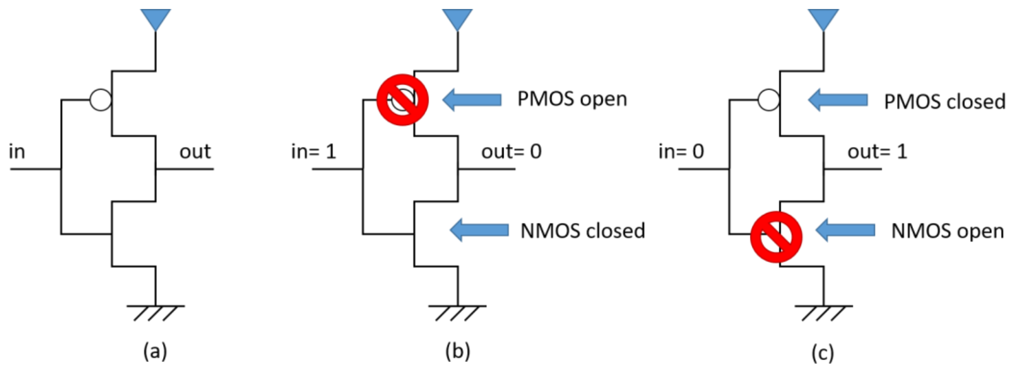
Therefore, at the end of the message, any binary value contained between the final *Low* and *Low* + *Range* will certainly represent the input information processed (MOFFAT, 1998). Since the final codified message will have, at least $-\log_2 Range$ and a maximum of $-\log_2 Range + 2$ bits, which respected the estimated entropy lower bound of (SHANNON, 1948), as already briefly explained before.

2.5 Power Consumption Concepts on CMOS Components

CMOS components are electric circuits, which are currently the main alternative to implement logic gates. CMOS is composed by an NMOS pull-up along with a PMOS pull-down. The pull-up has the complementary organization of the pull-down (i.e., complementary organization). Summarizing the behavior of a CMOS logic gate, when the pull-up is conducting current from the VDD to the output of the gate, the pull-down is open. When there is a path through the pull-down between the output of the gate and the GND, the pull-up is open (i.e., there is no path between the VDD and the output of the gate) (WESTE, 2011). For instance, Figure 2.14(a) depicts the primary logic gate, an inverter, whereas Figure 2.14(b) presents the behavior of the inverter when the input has value ‘1’; and Figure 2.14(c) when the input has the value ‘0’.

There are two main forms of power dissipation related to CMOS components: (i) the static power dissipation, (ii) and the dynamic power dissipation. One could made an analogy related to these two components, where the static consumption is related to the “consumption” a human would have when sleeping (i.e., the consumption a CMOS component has even when there is no switching activity). The dynamic consumption is related to the “consumption” a human would have when active (i.e., the consumption a CMOS component has when working, and therefore has switching activity). Considering the scope of this Thesis, the focus is the dynamic consumption and, therefore, will be further explained with more details.

Figure 2.14 – CMOS inverter organization and behavior



Source: the author.

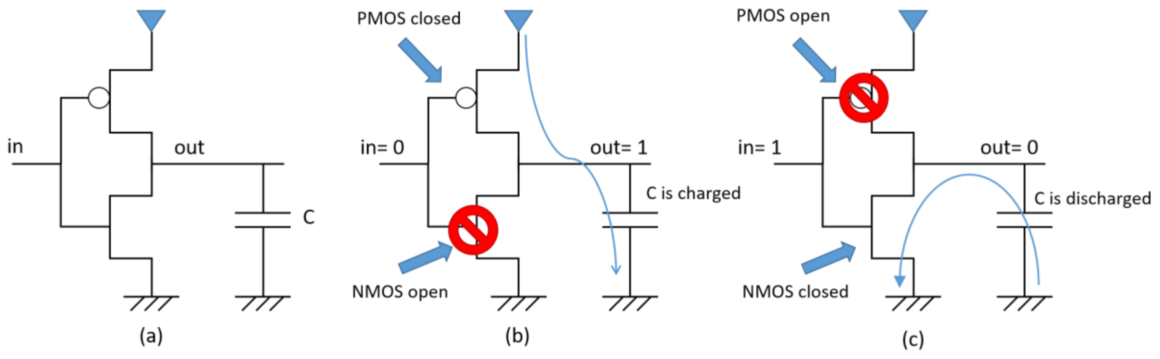
2.5.1 CMOS Dynamic Power Consumption

Dynamic power consumption in CMOS circuits occurs during the moments the circuit is doing what it is designed to do. Thus, the inputs of the logic gates will be changing its values actively (i.e., there is a switching activity for these logic gates). Furthermore, it is possible to divide the dynamic dissipation into two components: (i) switching consumption; and (ii) short-circuit consumption.

2.5.1.1 Switching Consumption

The switching consumption model is as presented in (2.6), where VDD is the voltage of the circuit; f is the clock frequency; C is the capacitance of the circuit; and α is the probability of switching the output of the circuit, when computed over a large number of clock cycles of active operation. Referring again to the example of the inverter, there is a capacitance attached to the output of the gate (any logic gate will have a capacitance, as well). Figure 2.15(a) shows the mentioned capacitance. When the input of the inverter has the value ‘0’, there will be a path between the VDD and the capacitance. Thus, the capacitor will be charged, as depicted in Figure 2.15(b). When the input of the inverter has the value ‘1’, a path between the capacitance and GND exists, and the capacitor is discharged, as presented in Figure 2.15(c). Therefore, during the charging of the capacitance, there will be dynamic power dissipation since it requires the inputs to be changing throughout the working of the circuit.

Figure 2.15 – CMOS switching consumption on inverter



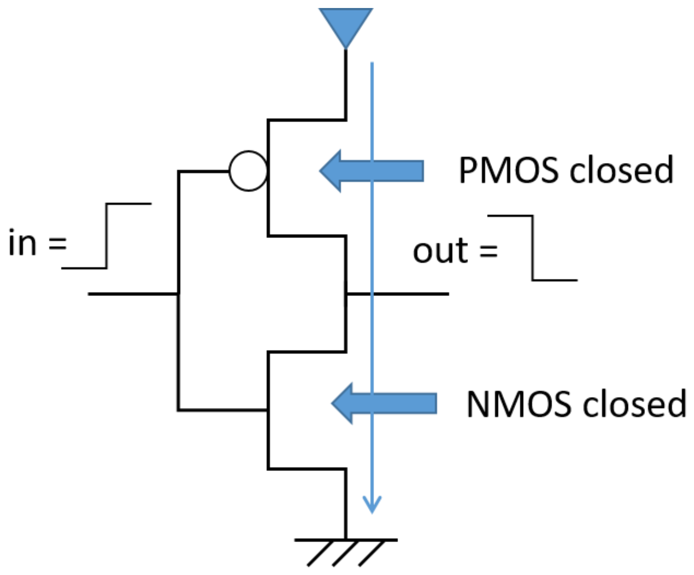
Source: the author.

$$P_{switching} = C * VDD^2 * f * \alpha \qquad (2.6)$$

2.5.1.2 Short-circuit Consumption

During the rise or fall of the inputs of a given logic gate, there will be a moment where both pull-up and pull-down of the gate will be conducting. Therefore, there will be a path between VDD and GND (i.e., a short-circuit). This current (and thus the power dissipation) is named short-circuit consumption and is modeled in (2.7), where f is the clock frequency, VDD is the voltage of the circuit, I_{short} the short-circuit current, and β the activity factor, which is analogous to α of (2.6). For instance, considering again the inverter, Figure 2.16 shows the visual representation of the short-circuit consumption.

Figure 2.16 – CMOS short-circuit consumption on inverter



Source: the author.

$$P_{short\text{-}circuit} = f * I_{short} * VDD * \beta \qquad (2.7)$$

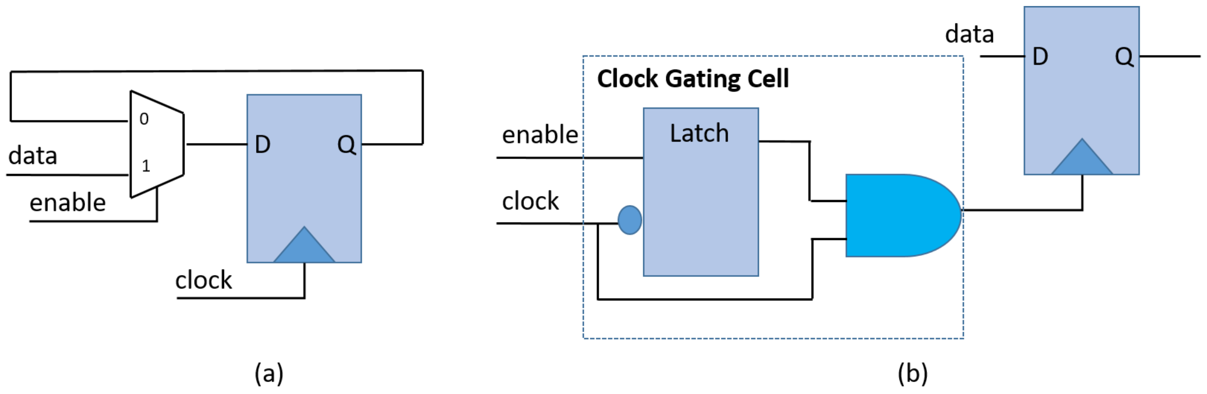
2.5.2 Architectural Techniques for Dynamic Consumption Reduction

Two techniques used at the architectural level (i.e., RTL) and related to this Thesis are further explained: Clock-gating and Operand Isolation.

2.5.2.1 Clock-gating

One may imagine a given Flip-flop (or n-bits register) that its behavior is as shown in Figure 2.17(a): in front of the register, there is a multiplexer, which controls when new data arrives at the register to be stored, controlled by an enable signal. When the ‘enable’ has the value ‘0’, the previously stored value is fed to the input of the register. Therefore, it is clear to observe that no switching consumption occurs, since the capacitance of the sequential logic does not change its current value. Nevertheless, since the clock signal is still switching, there will be a moment that it will cause a short-circuit consumption, in a moment where the Flip-flops are idle (i.e., are not storing any new value). A solution is to turn off (or gate) the clock during these moments, avoiding the short-circuit consumption. Figure 2.17(b) presents the Clock-gating technique (WU, 2000), modifying the original circuit presented in Figure 2.17(a), in order to turn off the clock and thus avoid the short-circuit consumption for the proper situations.

Figure 2.17 – Clock-gating technique

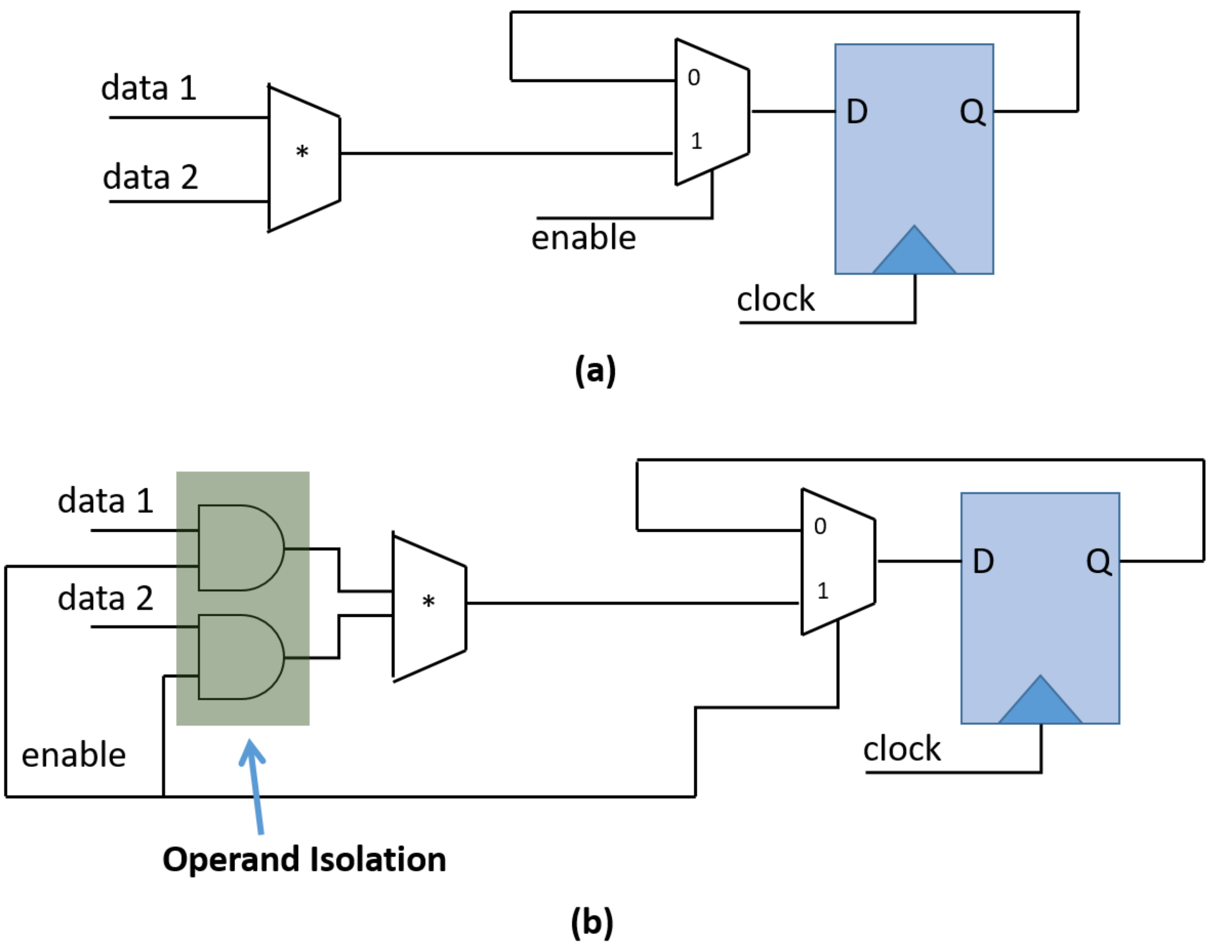


Source: the author.

2.5.2.2 Operand Isolation

For instance, one may consider a logic like the one presented in Figure 2.18(a), where the “*” represents a combinational multiplier logic. The output of the multiplier will be required only when the ‘enable’ of the register where it is attached has the value ‘1’. Thus, in case the inputs of the multiplier are switching during the moments where the enable has the value ‘0’, this will lead to unnecessary switching consumption at the multiplier (since those values will not be stored in the register). An option to avoid this consumption is the Operand Isolation technique (CORREALE, 1995), as depicted in Figure 2.18(b): an AND-gate is inserted into the inputs of the multiplier, where one input is the original value fed to the multiplier, and the other input is the 'enable' signal. Hence, in case the 'enable' remain with the value '0' for many cycles, no toggling will occur into the multiplier and, therefore, no switching consumption.

Figure 2.18 – Operand Isolation technique



Source: the author.

3 CABAC ALGORITHM

This chapter describes the central concepts of the CABAC algorithm: the operational blocks that compose the algorithm, the input data of the entropy coding on HEVC encoding, the variables used for the processing, and the fashion how they are updated/renormalized, according to the algorithm specification. In the end, related works with significant results regarding hardware implementation of CABAC block, as a whole or partially, are presented.

3.1 CABAC Concepts

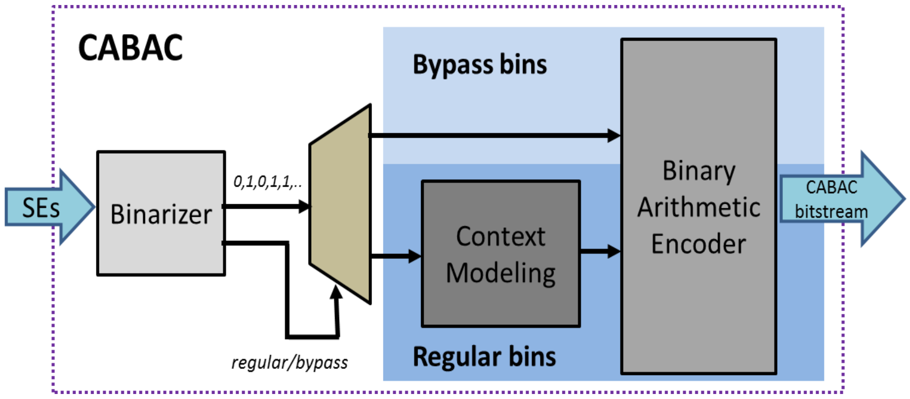
CABAC is an algorithm based in a recursive sub-interval division, which already appears as one of the entropy encoding options at the previous H.264/AVC, where the other option was the CAVLC (Context-Adaptive Variable Length Coding) along with Exponential-Golomb encoding (ITU-T, 2003). CABAC achieves more coding gains when compared to CAVLC + Exponential-Golomb, around 9-14% fewer bits generated at the final bitstream (SZE, 2013). The drawback is the higher computational complexity for its implementation, especially for parallelization of input data processing, since there are strict dependencies from a symbol to another one processed next (SZE, 2013). Nevertheless, the coding gains capability made CABAC the single choice for HEVC standard.

The input data of CABAC are the Syntax Elements (SEs), which come from all previous steps of encoding process (e.g., Inter or Intra-prediction, Transform, Quantization, etc.). Each SE has to be rearranged in a form that CABAC understands them since it only processes binary symbols. After that, these new binary values are categorized according to a given occurrence possibility, and the occurrences possibility is updated for the next incoming symbols of the same type. Finally, the recursive sub-interval-division step occurs, and the bitstream for the current symbol is appended to the other already processed ones. Thus, the mentioned steps are categorized into three macro-operations within CABAC, which are: (i) Binarization; (ii) Context-Modeling; and (iii) Binary Arithmetic Encoding (BAE). Figure 3.1 depicts the CABAC block diagram.

3.1.1 Binarization

CABAC algorithm requires the input data to be in binary representation and thus requires the called Binarization step. One may notices that all SEs are already in a binary representation, whether the HEVC encoding processes were implemented in hardware or software. Nevertheless, CABAC describes different types of binary representation than the simple binary-decimal conversion. Furthermore, those different binary representations allow better coding gains at the next steps of the entropy encoding. After the Binarization process, each bit of the resulting converted SE receives the name *bin*, and these bins are the input for Context Modeling and BAE. A low-power Binarization architecture proposal appears in Appendix A of this Thesis.

Figure 3.1 – CABAC block diagram



Source: (ALONSO, 2017).

In HEVC, there are five basic types of Binarization, which are: (i) Unary (U); (ii) Truncated Unary (TU); (iii) Fixed-Length (FL); (iv) Truncate-Rice (TR); and (v) Exponential-Golomb (EGk). Table 3.1 presents examples of binarization for the above-mentioned basic methods. There are also five custom methods, which are applied for certain SEs, which may be look-up tabled; or a merge of the basic types.

Table 3.1 – Examples of basic binarization types

SE Value	U	TU cMax=7	FL cMax=7	TR cMax=7 e cRiceParam=1	EGk k=0
0	0	0	000	00	1
1	10	10	001	01	010
2	110	110	010	100	011
3	1110	1110	011	101	00100
4	11110	11110	100	1100	00101
5	111110	111110	101	1101	00110
6	1111110	1111110	110	1110	00111
7	11111110	1111111	111	1111	0001000

Source: (ALONSO, 2017).

3.1.1.1 Unary and Truncated Unary

The Unary binarization consists in concatenate bins with value ‘1’, according to the decimal value of the SE binarized, and then adding a single bin ‘0’ to the end of the bin-code. For instance, looking at Table 3.1, one may notice that a SE with decimal value zero will have a bin code composed by a single ‘0’; whereas an SE with decimal value one is binarized to ‘10’; a SE with decimal value two is binarized to ‘110’; and so on.

The Truncated Unary is almost the same as the Unary binarization. The main difference is that there is a limit decimal value (i.e., the variable *cMax*), which restrains the maximum amounts of ‘1’ that may be appended one to the other for SEs that have a value above the limit, in which case will also not require the insertion of a ‘0’ at the end of the bin code. For example, looking again to Table 3.1, one may see that the *cMax* variable for the TU binarization is seven, which means that a given SE with decimal value equals to seven will have a bin code composed of seven ‘1’s concatenated, without a ‘0’ at the end of the code. Moreover, all SEs with values above the defined limit will have the bin code the same as the SE with decimal value seven (i.e., truncation will occur).

3.1.1.2 Fixed-Length

The Fixed-Length binarization scheme is the simple conversion from decimal to a binary representation, but defining a maximum decimal value that may be represented by this method. Table 3.1 presents an example, where the maximum decimal value that may be represented is seven (i.e., the bin code for all SEs will have three bins). This method is particularly useful when the converted SE is a flag (i.e., one-bit wide) and therefore the conversion in most cases will maintain the original value of the SE after the Binarization.

3.1.1.3 Truncated Rice

The Truncate Rice method is a composition of a prefix and a suffix (in case the suffix is necessary). The prefix derivation comes from the parameter *cMax, cRiceParam,* and the own SE original decimal value (here called *N*). A pre-calculation indicated by (3.1) generates the variable *prefixVal*, where “>>” means a shift-right operation. The final prefix value will be a Truncated Unary bin stream considering the TU code for the decimal value of the current SE divided by *cRiceParam* with a length of *prefixVal* + 1, when *prefixVal* value is less than the value of *cMax* divided by *cRiceParam*. Otherwise, the prefix will have a length of *cMax* divided by *cRiceParam*, where all bin has the value ‘1’.The suffix is a Fixed-Length binarization of the least significant bits of the SE value, where *cRiceParam* indicates the number of bits used for that purpose. When *cRiceParam* is equal to ‘0’, the Truncated Rice acts the same as Truncate Unary method. Table 3.1 shows an example of this binarization method when *cRiceParam* is ‘1’ and *cMax* is equal to seven.

$$prefixVal = N >> cRiceParam \tag{3.1}$$

3.1.1.4 Exponential-Golomb

The Exponential-Golomb method is also a composition of a prefix-suffix fashion. The parameter *k* indicates the initial amount of bits the suffix will have for the first values of the SEs processed, where the first 2^k codes will have *k* bits of a suffix. For instance, when *k* is ‘0’, the first 2^0 codes will have zero bits of suffix, then the next 2^1 will have one bit of suffix, then the next 2^2 will have two bits of suffix, and so on. Another example, when *k* is ‘1’, the first 2^1 codes will have one bit of suffix, then the next 2^2 will have two bits of suffix, the next 2^3 codes will have three bits of suffix, and so on. The suffix is a Fixed-Length representation, starting at the minimal value up to the maximum value possible to be represented by the current amount of bits a suffix has.

The prefix always has the number of suffix bits decremented by *k* for a given code. In other words, when *k* is equal to ‘0’, only the first code will not have the prefix, the next 2^1 codes will have one bit of prefix, the next 2^2 codes will have two bits of the prefix. For instance, when *k* is equal to ‘1’, the first 2^1 codes will have no prefix, the next 2^2 codes will have one bit of prefix, and the next 2^3 codes will have two bits of prefix, and so on. The prefix is a concatenation of ‘0’s for the number of bins mentioned for each generated code. Between the prefix and suffix, the insertion of a bin ‘1’ is required for all codes, whatever is the *k* value used.

3.1.1.5 Custom Methods

The custom methods are used for some specific SEs. There are five of them, named Custom1, Custom2, Custom3, Custom4, and Custom5. The first three custom methods are Look-up Tabled, with different input parameters and used for different SEs. In other words, their bin codes are already pre-calculated, and the value depends on the combination of the input variables for the proper SE that matches with the related code (i.e., no calculation required). The last two custom methods are combinations of Truncate Rice and Exponential-Golomb methods. The main difference is that Custom 4 utilizes both TR and EGk with fixed *cRiceParam* and *k* parameter, whereas Custom5 has the mentioned parameters updated throughout the processing flow of the associated SE for this binarization method.

3.1.2 Context Modeling

The binarization converts the original decimal values of the SEs into the representations already mentioned in the last section. Each bit of the converted SEs values is called a “bin”, as already defined. The bins have different properties, which lead to a categorization of them, as follows: (i) regular, (ii) bypass, (iii) terminate.

A regular bin also has two more categories: a regular bin could be the Most Probable Symbol (MPS) to occur, or the Least Probable Symbol (LPS). Since CABAC requires binary symbols, the algorithm only has one of the two above options for a given regular bin related to some SE. The MPS regular bin, as the name says, is more probable to occur during the processing of a binarized SE. Every time a MPS occurs, the occurrence probability of this bin to occur next is updated, along with the occurrence probability of the LPS bin related. On the other hand, when an LPS bin occurs, the occurrence probabilities for both MPS and LPS also have to be updated for the next bins to be processed. This update of probabilities according to the current bin classification is the context-adaptive part of CABAC, and the Context Modeling is the block responsible for it. The updates follow a Look-up Table fashion, where, according to whether the current bin is an LPS or an MPS, a variable called *State* indicates how and for which amount the probabilities have been updated, and the value of *State* will be required for the next step of the entropy encoding. The regular bins generated by a particular SE will have an initial *State* value and are updated starting from that initial value.

Bypass bins are considered to be equiprobable (i.e., whether the bin has a value ‘0’ or a value ‘1’, the probability for each one of them is always 50%). Therefore, this kind of bin does not undergo Context Modelling nor the *State* variable generation/update. The terminate bins, which occur very rarely, also do not have to undergo the Context Modeling processing.

3.1.3 Binary Arithmetic Encoder

3.1.3.1 Range and Low Variables

The Binary Arithmetic Encoder (BAE) is the final step of the CABAC block, being the bottleneck of the entropy encoding (ZHOU, 2015). The recursive sub-interval division occurs at this block, based into two main variables: *Range* and *Low*, following the Arithmetic Coding, as presented in the previous chapter. Since CABAC is a Binary Arithmetic Encoding algorithm, there will always be two single choices for any bin category, thus facilitating the Arithmetic Coding update of *Range* and *Low*, as shown in (2.4) and (2.5). Moreover, there is an asymptotical approximation to Shannon lower bound of the entropy by using the minimum precision bits to represent the lower bound of a sequence of symbols (MARPE, 2003). One may observe in the BAE, there is no need to accumulate probabilities from multiples symbols of an alphabet, since only two symbols are always the option for this scenario, and which.

Another important difference in CABAC is that the update of *Range*, instead of using a direct multiplication, utilizes pre-calculated and stored values, whose selection occur based on the Context Modeling flow throughout the processing (i.e., another Look-up Table fashion stores this pre-calculated values) (ITU-T, 2013).

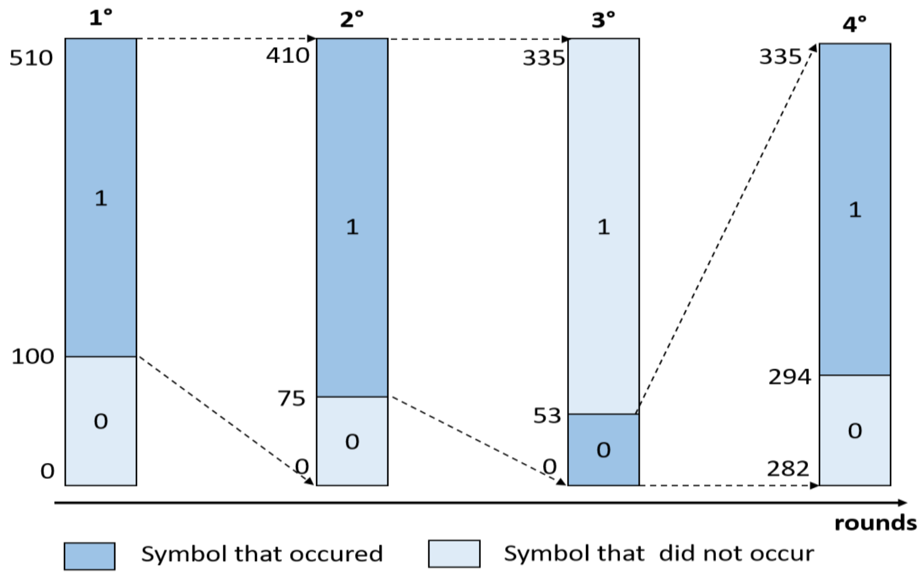
As already mentioned in section 2.4.2, the *Range* variable is the overall values to represent the probabilities that an MPS and an LPS bins have to occur at each moment of time. The *Low* value is the lower bound value to be used for the overall range of probabilities. The probability of an LPS bin to occur is represented within a smaller portion of the *Range* value, named *rLPS*, whereas the probability of a MPS to occur (named *rMPS*) is the value of the *Range* decreased by the *rLPS*. In case an MPS bin is the current one, the next *Range* value receives the *rMPS*, whereas the new *Low* value is kept the same as was before. Otherwise, in case an LPS bin is the current one, the new *Range* value receives the current *rLPS* value, and the *Low* is updated as being the previous *Low* value summed with the *rMPS*. The *rLPS* definition appears in (3.2) (where the p_{LPS} is the updated probability of the LPS bin to happen), but actually its derivation occurs directly by the *State* variable, using the mentioned look-up table fashion (i.e., multiplier-less), along with the two most significant bits of the current *Range* value (ITU-T, 2013). In (3.3) is observed the update behavior of the other referred variables.

$$rLPS \leftarrow Range * p_{LPS} \tag{3.2}$$

$$\begin{cases} rMPS \leftarrow Range - rLPS & \\ Range \leftarrow rMPS \text{ and } Low \leftarrow Low, & \text{for } bin = MPS \\ Range \leftarrow rLPS \text{ and } Low \leftarrow Low + rMPS, & \text{for } bin = LPS \end{cases} \tag{3.3}$$

Figure 3.2 shows an example of the behavior these variables have during the processing of some hypothetic regular bins. For instance, the *Range* value starts with value 510, whereas the *Low* value starts at zero. For the given example, a bin with value ‘1’ is considered the MPS, whereas a bin with value ‘0’ is the LPS. The *rMPS* has the value 410 (which is represented by the portion of the *Range* overall values between 100 and 510), whereas the *rLPS* has the value 100 (which is represented by the portion of the *Range* overall values between 0 and 100). At the first round, a MPS occurs; thus, the new *Range* is now represented by the value of the *rMPS*, starting at the prior *Low*, since it is not updated for a MPS bin. As one may see, since *rMPS* has the value 410, the *Range* for the second round is updated to 410, with the *Low* kept with value zero. The new *rMPS* and *rLPS* are updated according to *State* variable of the Context Modeling (which is implicit in Figure 3.2) and following (3.3), and now they are 335 and 75, respectively. Again, a MPS occur, following the same already mentioned protocol: *Range* now will represent values within the value of the *rMPS* (i.e., *Range* receives the value 335), and the *Low* value is not updated, maintaining the value zero. The *rMPS* and *rLPS* are updated again following the same behavior already presented, with the values 282 and 53, respectively. At the third round, a LPS occurs, and now the *Range* variable receives the current value of the *rLPS* variable (i.e., the overall probabilities are represented within the value represented by the *rLPS*). Since a LPS have occurred, the *Low* variable must be updated, being summed its original value with the *rMPS*. Therefore, the updated *Low* receives the value 282, and the updated *Range* is now represented by the values between 282 and 335, which 335 subtracted by 282 leads to 53 (the previous *rLPS* value, represented by the new lower and upper bounds of the *Range* variable). This process repeats for all regular bins as already summarized in (3.3).

Figure 3.2 – Example of Range and Low update



Source: (RAMOS, 2016).

Bypass bins do not update the *Range* variable but do update the *Low*. These bins are equiprobable (i.e., the occurrence percentage for either ‘0’ or ‘1’ is 50% throughout the whole entropy encoder flow). The updated *Low* is multiplied by two, but when the bypass bin has the value ‘1’, there is the need to add the current *Range* variable to the multiplied-by-two *Low* value (ITU-T, 2013). These conditions are summarized in (3.4).

$$\begin{cases} Range \leftarrow Range & \\ Low \leftarrow 2*Low, & for\ bin = \text{‘}0\text{’} \\ Low \leftarrow 2*Low + Range, & for\ bin = \text{‘}1\text{’} \end{cases} \tag{3.4}$$

3.1.3.2 Renormalization Process

Range and *Low* are 9-bits and 10-bit variables, respectively. Therefore, after the update, they may fall below a certain value, which requires a renormalization process (ITU-T, 2013). A new auxiliary variable called *Outstanding Bits* (OB) will assist the renormalization process, along with the bitstream generation (ITU-T, 2013). Each time the *Range* variable falls below the value 256 due to the current regular bin processed, it will be multiplied by two until it has reached at least the value 256. Each time this multiplication occurs, the *Low* variable is tested in this following order of priority:

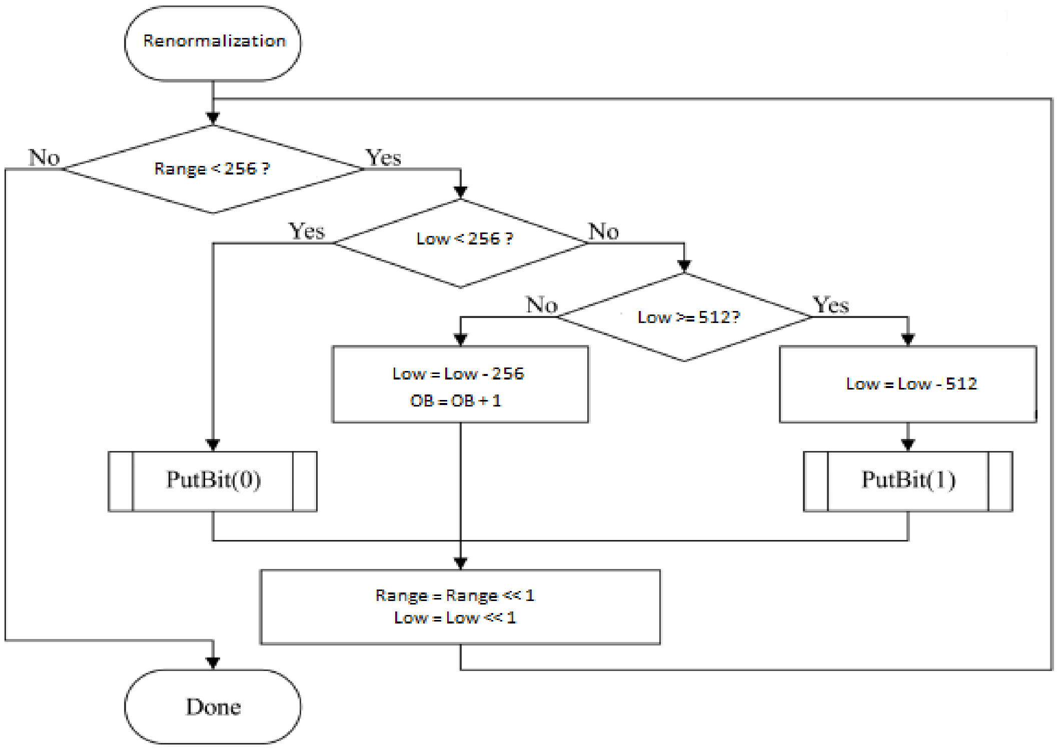
(i) ***Low* has a value below 256**: the OB variable is zeroed, and bitstream is generated (via *PutBit* function).

(ii) ***Low* has a value above or equal to 256, and below 512**: one is added to the current OB value, and the *Low* variable is decreased by 256.

(iii) **Low has a value equal to or above 512**: the OB variable receives the value zero, and the bitstream is generated (via *PutBit* function).

After any of the conditions (i), (ii), or (iii) occurred, the modified *Low* variable above is multiplied by two. These conditions are summarized in Figure 3.3 (ITU-T, 2013).

Figure 3.3 – Renormalization of Range and Low variable for regular bins



Source: the author, modified from (ITU-T, 2013).

Bypass bins do not update the *Range* variable, only *Low*. Hence, only *Low* shall be renormalized in case it is needed for bypass bins, following the conditions below:

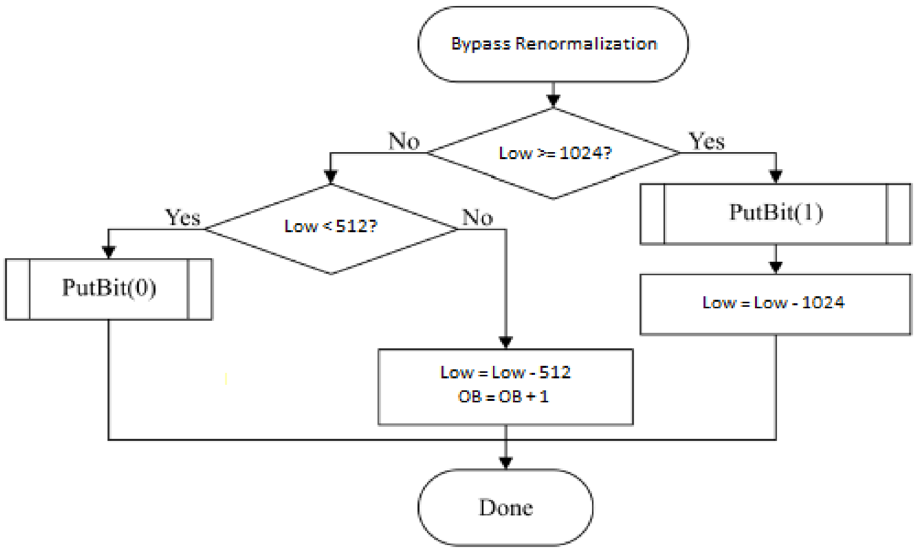
(i) ***Low* has a value above or equal to 1024**: the OB variable is zeroed, *Low* is decreased by 1024, and the bitstream is generated (via *PutBit* function).

(ii) ***Low* is below 1024 and above or equal to 512**: one is added to the current OB value, and the *Low* variable is decreased by 512.

(iii) **Low has a value below 512**: the OB variable receives the value zero, and bitstream is generated (via *PutBit* function).

Figure 3.4 summarizes the above-mentioned conditions.

Figure 3.4 – Low renormalization for bypass bins

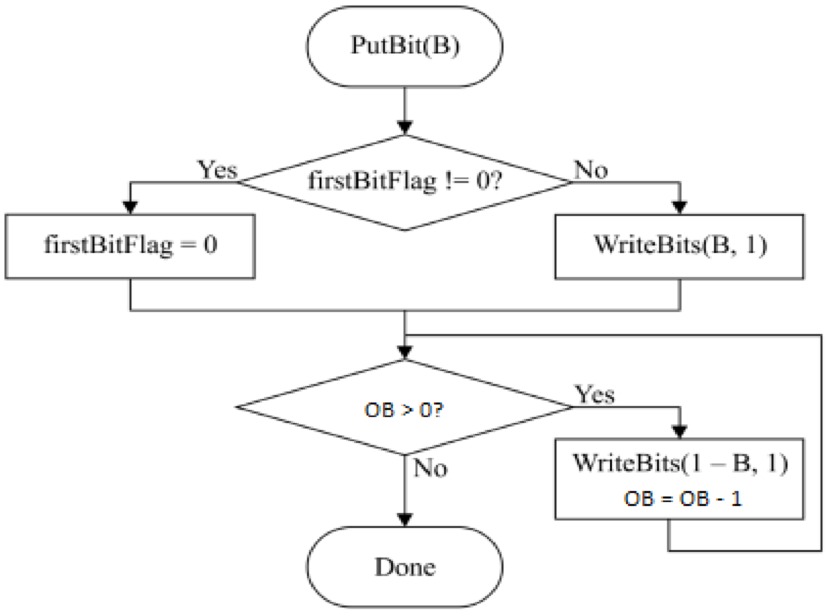


Source: the author, modified from (ITU-T, 2013).

3.1.3.2 Bitstream Generation

The entropy encoding bitstream generation is derived from the renormalization process mentioned and will require the use of the OB variable as well. When the clauses that generate bitstream during the renormalization process for both regular and bypass occur, the referred *PutBit* function on Figure 3.5 takes place, and is described as follows, in a simplified fashion, based on (ITU-T, 2013): the very first bit of the bitstream is inferred and not appended to the output. For all other cases, a bit of the value indicated by ‘0’ or ‘1’ (i.e., the *B* on the *PutBit* call) is generated. After that, a vector of n-bits with the opposite value of the first bit is generated (i.e., 1 – *B*, where *B* can be ‘0’ or ‘1’), where *n* has the same value of the OB variable generated at the renormalization step. After that, the OB variable restarts with value zero.

Figure 3.5 – Bitstream PutBit(B) generation function



Source: the author, modified from (ITU-T, 2013).

3.2 Related BAE Prior-art

There have been several CABAC research works in the past few years, related to the predecessor H.264/AVC, and currently focusing in the HEVC standard. One critical remark is the fact that the BAE block does not differ on the two standards, only Binarization and Context Modeling (e.g., there are different SE between the two standards; some probabilities were updated). Therefore, any prior-art improvement related to the BAE block within the H.264/AVC context is still valid for the BAE block for HEVC standard.

For example, the work of (CHEN, 2010) is one of the first to proposes pipeline structures to increase frequency in the BAE, whereas (KUO, 2006) a low-power approach for the CABAC block, both works on the H.264/AVC context. The work of (PENG, 2013) is one of the first to introduce a CABAC block for the HEVC context, whereas (VIZZOTTO, 2015) proposes an efficient area-throughput trade-off CABAC architecture.

For the purpose of this Thesis explanation, the most important related works (and which were used as baseline for this Thesis) with the remarkable features achieved by them are described in more specific details as follows. As important remark is the lack of power-saving consideration on most of the related recent entropy-encoding researches, even if some present power dissipation values.

3.2.1 Work of Liu et al.

The work of Liu et al. (LIU, 2011b) is the first to propose a BAE architecture for H.264/AVC able to do the renormalization process in just one clock cycle (considering a hardware architecture for the BAE block), for whatever value *Range* and *Low* may have after the update of these variables. For regular bins, the new renormalized *Range* value (named *R''* at the authors' work) is generated by left shifting the *Range* derived from equation (3.3) by the minimum amount *n* necessary for *R''* to be above 256. The renormalized *Low* (named *L'* by the authors) is derived following (3.5), where L^t is the temporary value by left shifting the new *Low* from (3.3) by *n*. For the derivation of OB variable of a regular bin, the authors first define two temporary variables, named ρ e γ, where ρ is equal to 9 – *n*, and γ is the bit index of the last zero in $L^t[8{:}\rho]$. In case no zero exist on the referred vector, γ is set to zero. Thus, if $L^t[9]$ is equal to one, OB is equal to the maximum value between 0 or γ – ρ. In case $L^t[9]$ is equal to zero, two conditions may apply: (i) if there is no zero bits in the vector $L^t[8{:}\rho]$, the new OB is equal to OB + *n*; (ii) otherwise, OB is equal to γ – ρ. Finally, the bitstream generation for regular bins follows the pseudo-code in Figure 3.6, where $\sim Low[9]^{OB}$ represents a negation of the value stored at the cited position on the *Low* value right after the update in (3.3), replicated OB times, and β is defined as '8' subtracted by the bigger value between γ and ρ. The ',' between brackets represents the concatenation of the values inside the brackets in a single vector. One may remark that the implementation proposed follows strictly what the standard demands, and is depicted in Figures 3.3 and 3.4.

$$\begin{cases} L'[8{:}0] \leftarrow L^t[8{:}0] & \\ L'[9] \leftarrow L^t[9] & \text{for } L^t[9{:}10-n] = 2^n - 1 \\ L'[9] \leftarrow 0 & \text{otherwise} \end{cases} \tag{3.5}$$

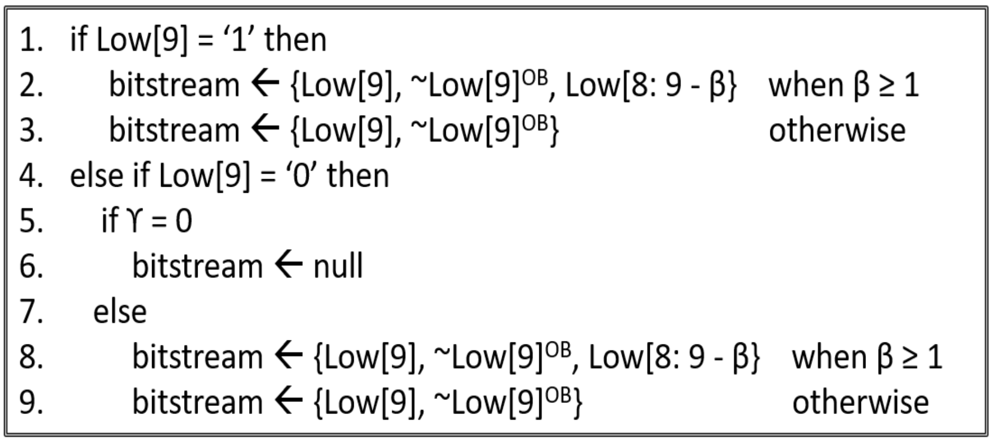
For bypass bins, the one-round renormalization of *Low*, the update of OB, and bitstream generation follows (3.6), (3.7), and (3.8), respectively, where now the L^t is the value of *Low* right after the update in (3.4). The ^{OB} and the ',' between brackets are used with the same meaning as already presented for the regular bins process. The developed full CABAC architecture can process 2-bins per clock cycle, and achieves a throughput of 634 Mbins/s, by synthesizing the architecture on a 90 nm CMOS technology. Moreover, the BAE consumes around 4.9 mW of power.

$$\begin{cases} L'[9{:}0] \leftarrow L^t[9{:}0] & \text{for } L^t[10] = 1 \\ L'[9] \leftarrow 0 \quad \text{and} \quad L'[8{:}0] \leftarrow Lt[8{:}0] & \text{otherwise} \end{cases} \tag{3.6}$$

$$\begin{cases} OB \leftarrow OB + 1 & \text{for } L^t[10:9] = 01 \\ OB \leftarrow 0 & \text{otherwise} \end{cases} \quad (3.7)$$

$$\begin{cases} bitstream \leftarrow \{1,0^{OB}\} & \text{for } L^t[10] = 1 \\ bitstream \leftarrow \{0,1^{OB}\} & \text{for } L^t[10:9] = 00 \\ bistream \leftarrow null & \text{otherwise} \end{cases} \quad (3.8)$$

Figure 3.6 – One-round bitstream generation pseudo-code for regular bins



```
1. if Low[9] = '1' then
2.     bitstream ← {Low[9], ~Low[9]$^{OB}$, Low[8: 9 - $\beta$}   when $\beta \geq 1$
3.     bitstream ← {Low[9], ~Low[9]$^{OB}$}                  otherwise
4. else if Low[9] = '0' then
5.     if $\Upsilon$ = 0
6.         bitstream ← null
7.     else
8.         bitstream ← {Low[9], ~Low[9]$^{OB}$, Low[8: 9 - $\beta$}   when $\beta \geq 1$
9.         bitstream ← {Low[9], ~Low[9]$^{OB}$}                  otherwise
```

Source: the author, based in (LIU, 2011b).

3.2.2 Work of Fei et al.

The work of Fei et al. (FEI, 2011) proposes a CABAC design for H.264/AVC, where four BAE cores are appended one to the other in a combinational fashion, making it a multi-bin processing design per clock cycle. A four-stage pipeline within each BAE core is applied, where the stages are related to the operation steps as follows:

1. **Generation of *rLPS*** – *rLPS* candidates are chosen (based only on *State* at this moment) before its use at *Range* update for regular bins;
2. ***Range* update**: as the name says, the Range variable is updated for regular bins, following (3.2);
3. ***Low* update**: the Low variable is updated for either regular or bypass bin, along with the OB variable;
4. **Bitstream generation**: the final bitstream is generated for each core, whenever the output bits are required to be delivered at a given BAE core at the current clock cycle.

As results, the architecture can process 4-bins per clock cycle. The synthesized architecture for 90 nm CMOS technology achieves 1,116 Mbins/s, running at 279 MHz, where the critical path is the *Range* update stage throughout the four appended cores.

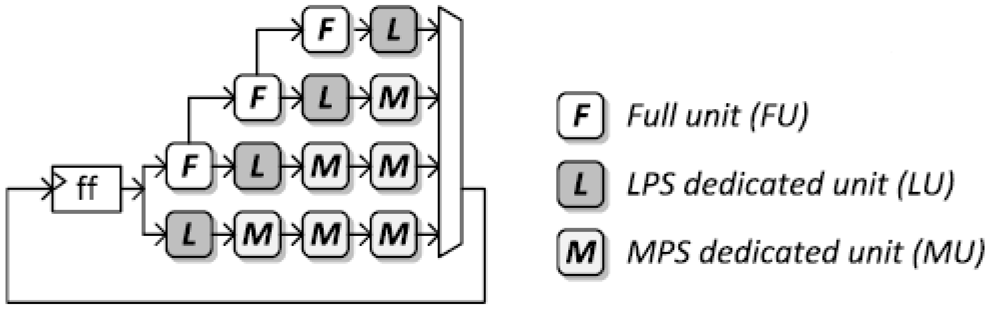
3.2.3 Work of J. Zhou et al.

The work of J. Zhou et al. (ZHOU, 2013) enhances the BAE architecture of (FEI, 2011), by proposing some high-throughput novel features to the design, as follows:

1. ***rLPS* pre-renormalization at the first pipeline stage (PN rLPS):** considering a four-stage pipeline architecture as presented by (FEI, 2011) the generation of *rLPS* candidates occurs at the first stage. Nevertheless, the renormalization of the *rLPS*, in case the current bin is a LPS (i.e., the *Range* variable will receive the *rLPS* as new value), occurs at the second stage, the *Range* update (the critical path of the prior design). Therefore, (ZHOU, 2013) proposes to transfer the *rLPS* renormalization from the second to the first pipeline stage thus decreasing the critical path of the BAE, i.e., the renormalization process, in order to be done in a single clock cycle, requires a Leading Zero Detector (LZD) block, which affects significantly the critical path of the block. By doing so, the four speculated alternatives that may be used as the *rLPS* for the current round are pre-renormalized before the decision of whether of them will effectively be used as the new *Range* value (if the current bin is an LPS). The decision of which of the four *rLPS* candidates will be used requires the two most significant bits of the current Range, which are available only at the second pipeline stage.
2. **Hybrid Path Coverage (HPC):** In case an LPS bin is the current one, the *Range* is the own *rLPS* value, not requiring any subtraction for this purpose, instead of MPS bins, which require the subtraction of the current *Range* value by the *rLPS* (3.3). Therefore, (ZHOU, 2013) proposes different BAE cores to process the different types of regular bins, named the Hybrid Path Coverage (HPC). Three different types of cores are proposed: Full Unit (FU), MPS Unit (MU), and LPS Unit (LU), where FU are able to process either an MPS or an LPS bin, LU only LPS bins, and MU only MPS bins. The Hybrid Path Coverage is shown in Figure 3.7. The following combinations of bins can be therefore processed at a given clock cycle: an LPS followed by three MPS; a

LPS or MPS followed by and LPS and two MPS; two regular bins of any type, followed by an LPS and a MPS; or three regular bins of any type, followed by an LPS. The advantage of using this technique is the removal of a subtractor in the critical path (i.e., *Range* update stage) of the work of (FEI, 2011) and thus decreasing the critical path in 13%. The drawback is the decrease in the number of bins per cycle the architecture is able to process since an LPS will not always be available to be processed at a given clock cycle (i.e., the bins-per-cycle throughput will be less than 4-bins/cycle).

Figure 3.7 – HPC technique depiction



Source: (ZHOU, 2013).

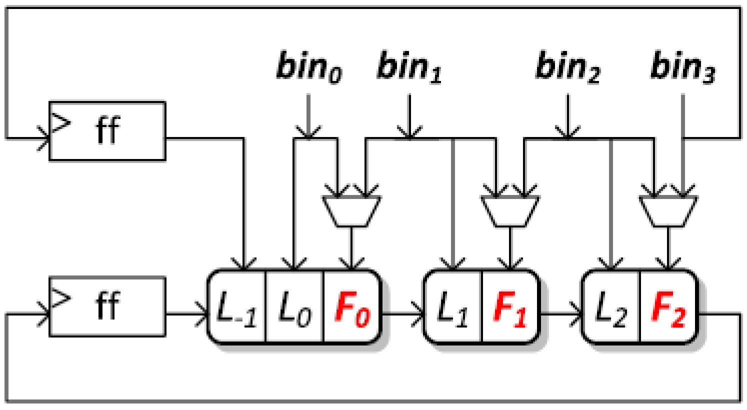
3. **Bypass Bin Splitting (BPBS):** Bypass bins do not require to update the *Range* variable. Hence, (ZHOU, 2013) proposed the splitting of the stream for regular and bypass bins right after the Binarization step of CABAC. PIPOs (Parallel-in Parallel-Out) structures are used to store the bins of different types, and the merge of streams occurs right before the *Low* update stage (i.e., third pipeline stage of the previous BAE design). The *Low* stage is only activated if five bins of any type are available at a given clock cycle during the merge process. Otherwise, the *Low* stage stalls until the required amount is reached. This feature increases the overall performance by 17%, leveraging the number of bins per cycle the BAE is able to process.

The synthesis results for the proposal presented at 65nm CMOS technology achieved around 4.40 bins per cycle using HEVC video sequences, and 1,769 Mbins/s running at a maximum achievable frequency of 402 MHz.

3.2.4 Work of D. Zhou et al.

The work of D. Zhou et al. (ZHOU, 2015) is a follow-up to the previous mentioned (ZHOU, 2013). Along with the features presents in (ZHOU, 2013), a novel proposition to decrease the critical path of the BAE architecture is proposed, in substitution of the HPC. This new feature, named Look-ahead rLPS (LH rLPS), proposes a different approach, where only two types of processing units exist for the BAE block: the LPS Unit (LU) and the Full Unit (FU), similarly as presented for the HPC. The difference is that only seven cascaded units will be available (named here L_{-1}, L_0, F_0, L_1, F_1, L_2, and F_2, where L_n correspond to LU, and F_n corresponds to FU), and at the beginning of the *Range* update stage, two LU are appended one another, as presented in Figure 3.8. The authors also proposed a way to speculate the possible *rLPS* candidates in these two cascades LU units at the first pipeline stage. Hence, a new portion of logic from the critical path (i.e., *Range* update stage) is transferred from the second to the first pipeline stage of the BAE cores.

Figure 3.8 – LH rLPS technique depiction



Source: (ZHOU, 2015).

The seven cores will have the following behavior: the F_n cores will always process a regular bin because they are suitable for LPS and MPS bins. Only one of the L_n cores will be active during a clock cycle, in case an LPS can be addressed to them in the correct incoming bins order. For instance, if the sequence of LPS, MPS, MPS and LPS bins is received, the cores L_0, F_0, F_1, and F_2 are assigned to process the bins in the respective order presented. In another example, if a sequence of bins is MPS, LPS, LPS, and MPS, the following cores are assigned respectively: F_0, L_1, F_1, and F_2. The L_{-1} core is only used if a sequence of MPS, MPS, MPS and LPS bin occurs. Hence, all FU cores will be used for the MPS bins, and the LPS bin will is stored to be processed in the next cycle by L_{-1} unit. The usage of the L_{-1} unit at a given clock cycle does not imply that the other LU cores cannot be used (i.e., they are suitable to be used for LPS bins at this situation, following the rules already presented).

The final proposed BAE architecture utilizes PN, LH rLPS, and BPBS, achieving an average of 4.37 bins per cycle of processing to related HEVC video sequences. The synthesized design for 90 nm CMOS technology achieves a frequency of 420 MHz, and throughput of 1,836 Mbins/s within the scope of HEVC standard, being the highest throughput so far found for the BAE (and therefore CABAC).

3.2.5 Summary of Related Works

Table 3.2 presents the summary of the related works. The variables presented are the proposed techniques at each work, the throughput of bins per clock cycle, the technology node used for synthesis, the maximum frequency, the amount of Mbins/s, the equivalent gates count for the whole CABAC, power consumption values (in case available), and the method for power estimation (if available). A critical reminder is the fact that none of the related works presents alternatives for power dissipation savings, even though the work of (LIU, 2011b) presents power values for tool-inferred stimuli.

Table 3.2 – Summary of related prior-art works

Design	**(LIU, 2011b)**	**(FEI, 2011)**	**(ZHOU, 2013)**	**(ZHOU, 2015)**
Proposed Techniques	One-round renormalization	4-stage pipeline BAE, four BAE cores	PN rLPS, HPC and BPBS	LH rLPS
#bins/cycle	2	4	4.4	4.37
CMOS Technology	90 nm	90 nm	65 nm	90 nm
Maximum Frequency	238 MHz	279 MHz	402 MHz	420 MHz
Mbins/s	634	1116	1769	1836
Gate Count	3.9 K	36.2 K	57.3 K	64.1 K
Power consumption	4.9 mW	-	-	-
Stimuli generated	User-defined	-	-	-

Source: the author.

4 LOW-POWER DESIGN FOR BAE BLOCK

This chapter will present a statistical analysis made in to verify any possible low-power opportunities within HEVC context of the BAE block. Moreover, the analysis corroborates one update proposed by the HEVC standard related to bypass bins, and the same behavior for bypass bin occur for regular bins. As a result, a proposition for a low-power BAE block version based on the work of (FEI, 2011) is presented, driven by the statistical analysis results.

4.1 Methodology for Low-power Design

Power consumption decrease is a sought-after goal related to real-time video processing architectures embedded into battery-based devices. Power saving opportunities can be derived by analyzing a given design and realizing that some parts of the architecture are required only in certain situations. The more these parts of the design are not required and kept switching, the more unnecessary power consumption the architecture is wasting. Therefore, the methodology for this part is fourfold:

(i) **Analysis of the block behavior**: considering the context of the BAE block, an analysis of the equations that command its flow occurs, and parts of the architecture that may be required for only certain situations are pointed.

(ii) **Statistical analysis for the block inputs:** after the step (i), a statistical analysis using random inputs (i.e., video sequences used for test purposes) are used to verify if the opportunities raised at the first step are valid. In other words, considering a hardware approach, a specific portion of the architecture is required to remain idle for some time in order to be worthy to avoid the switching activity into it.

(iii) **Low-power techniques insertion and measurement:** after steps (i) and (ii), and in case some potential successful opportunities are pointed out by the previous steps, suitable hardware power-saving techniques insertion occurs into the chosen BAE design.

(iv) **Synthesis and power results comparisons:** A synthesis occurs for chosen and available technology node, i.e., for the original design, and the design with the insertion of the low-power techniques. Power analysis results derived from real video sequences (or closely related sequences) are generated for both design versions. Therefore, the power results and possible gains are trustworthy when compared to purely tool-based input stimuli.

4.2 Analysis of the BAE Behavior

One may notice, looking back to (3.3) and (3.4) that some peculiar situations for the BAE block behavior happen. For instance, when a regular LPS bin occurs, it requires a sum of the current *Low* value with the *rMPS*, but that does not apply for the case an MPS is the current regular bin, where *Low* continues with its current value. Bypass bins do not update the *Range* variable. Thus, when the current bin is a bypass, the subtraction of *Range* by *rLPS* is also not required. Furthermore, if a regular bin of any type is the current processed one, the update of the *Low* variable for a bypass bin is unnecessary, as shown in (3.4). All of these mentioned situations may require the cited variables to be stored through the process. These storage devices within the BAE design are also necessary only when the proper bin type is occurring, and therefore the updated variables need to be saved. Nevertheless, all these scenarios will have to be implemented into a full BAE design, since both regular and bypass bins may occur seamlessly throughout the functionality.

4.3 Statistical Analysis of the BAE Inputs

Related recommended test video sequences of HEVC standards (BOSSEN, 2013) are run on the HM HEVC Reference Software (HM, 2016), in which a sampling procedure was made in order to raise the proper statistics for the power-saving opportunities. There are two central statistics for the goal desired: (i) the total percentage that each type of bin occupies within the sequences, and (ii) the average consecutive occurrence each type of bin undergoes (i.e., how many times in a row the same type of bin occurs before another type of bin starts to occur).

The first statistic is presented in (ZHOU, 2015) and reproduced in Table 4.1 below. One may notice that regular bins correspond to an average of 74.5% of total amount for the five sequences used, whereas bypass bins to 25.5 %. Regular MPS bins, as the name suggests, represent the majority of all bins (an average of 55.3% of total), whereas LPS bins to 19.2%. This result already points out that regular bins (especially MPS bins) are the vast majority. Therefore, parts of a BAE design required only for bypass bins have a potential of being turned off during regular bins processing. Nevertheless, the same reasoning applies to bypass bins, even if they are around ¼ of the total.

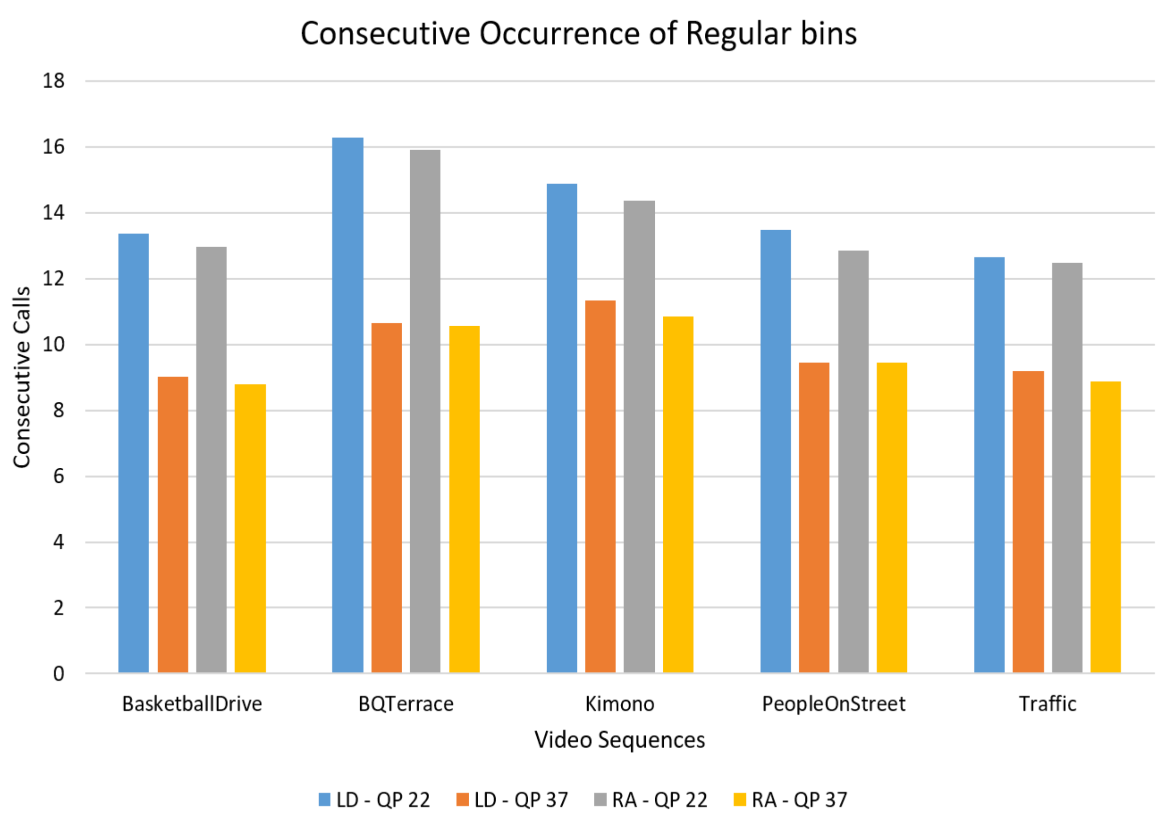
Table 4.1 – Proportion of bin types occurrence for test video sequences

Sequence	**P_{MPS}**	**P_{LPS}**	**P_{bypass}**
BasketballDrive	54.8%	18.1%	27.1%
Traffic	56.4%	21.1%	22.5%
PeopleOnStreet	50.9%	19.9%	29.2%
BQTerrace	61.6%	18.4¨%	20.0%
Kimono	52.7%	18.2%	29.1%
Average	55.3%	19.2%	25.6%

Source: (ZHOU, 2015).

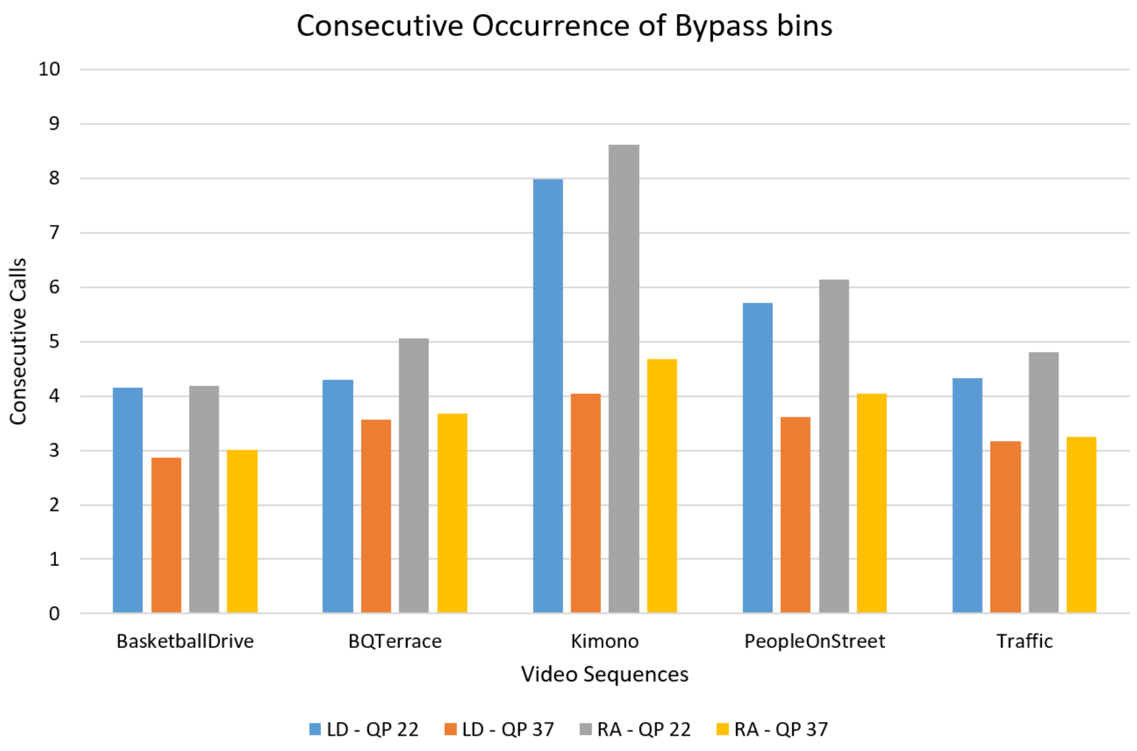
The same sequences used by (ZHOU, 2015) were run for the second statistic (i.e., occurrences in a row of the bins types), using the same recommended variation of configuration (Low Delay and Random Access) and the two upper and bottom values for Quantization Parameter (QP): 37 and 22. The results are presented in Figure 4.1, Figure 4.2, Figure 4.3, and Figure 4.4, respectively for regular bins (both LPS and MPS), bypass bins, only regular MPS bins, and only regular LPS bins.

Figure 4.1 – Consecutive occurrence of regular bins for test video sequences



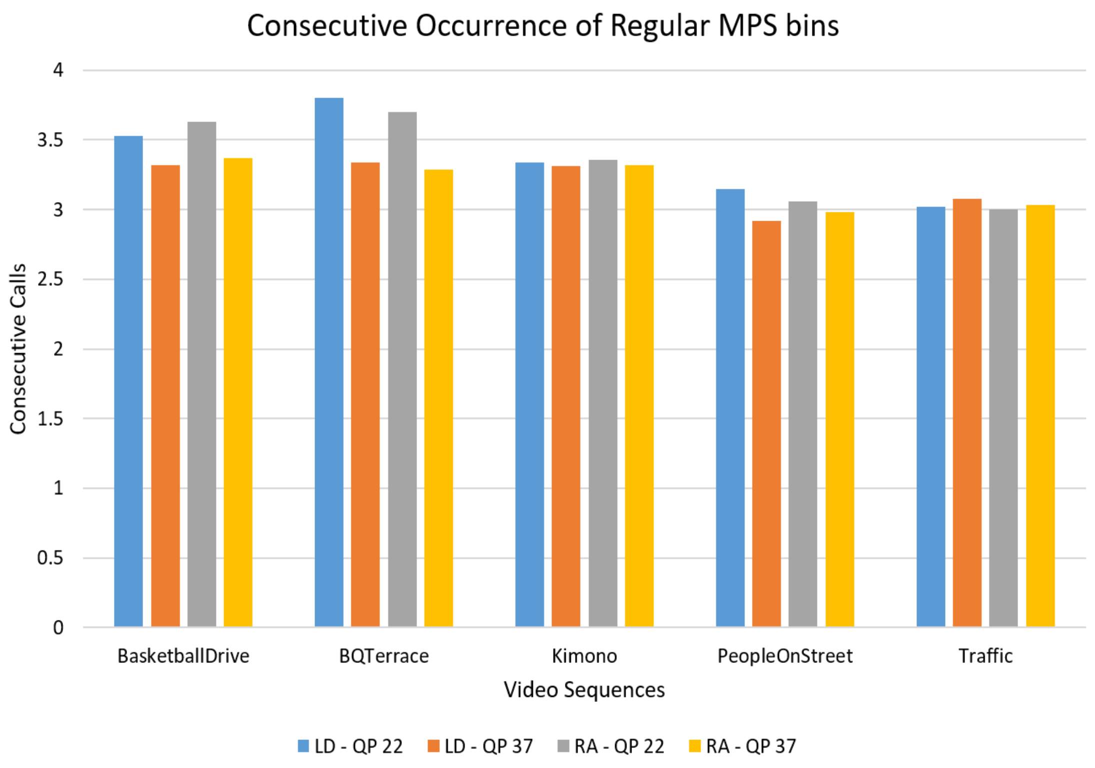
Source: (RAMOS, 2017).

Figure 4.2 – Consecutive occurrence of bypass bins for test video sequences



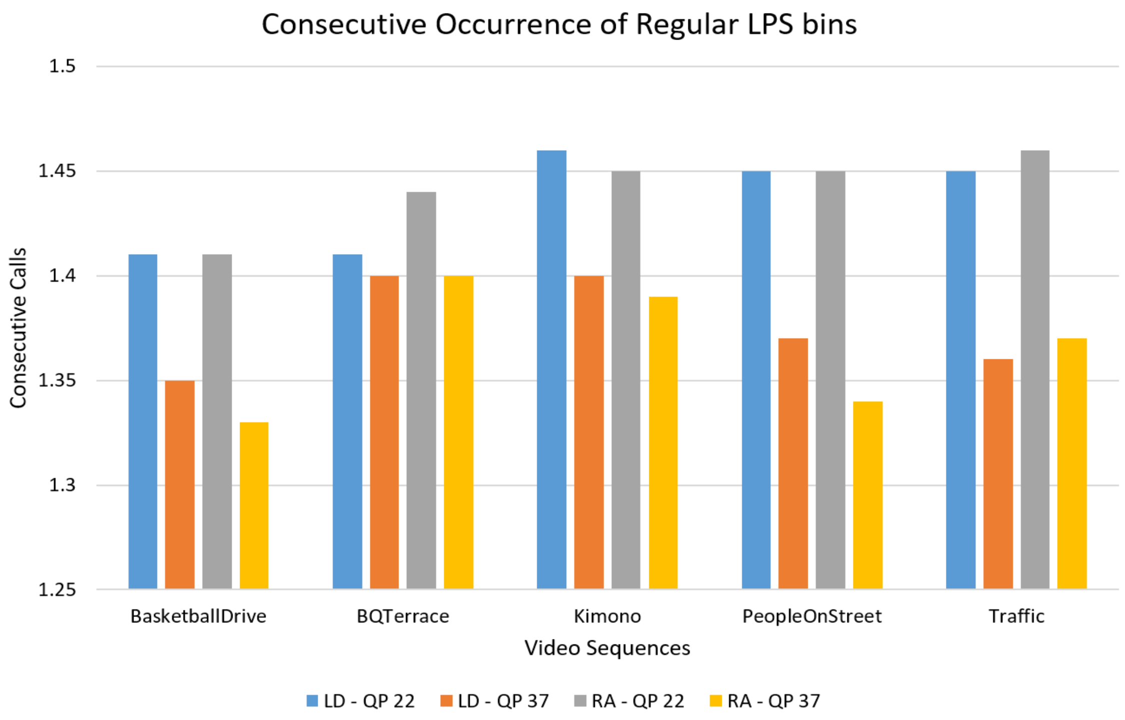
Source: (RAMOS, 2017).

Figure 4.3 – Consecutive occurrence of regular MPS bins for test video sequences



Source: (RAMOS, 2017).

Figure 4.4 – Consecutive occurrence of regular LPS bins for test video sequences



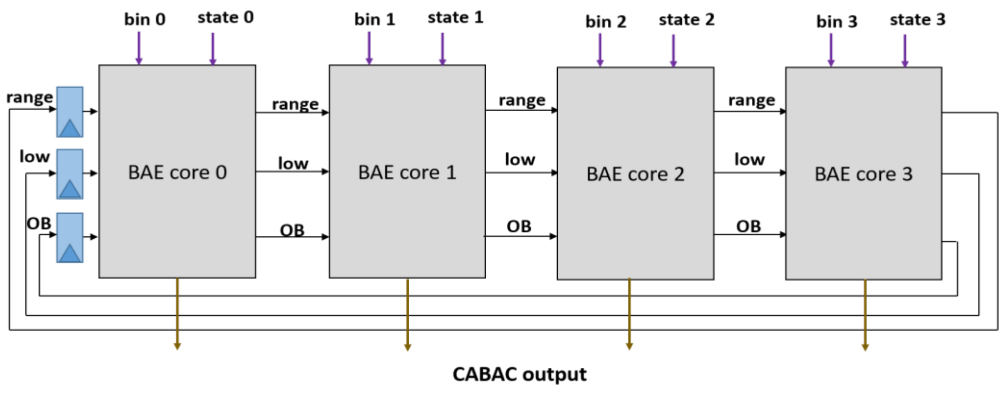
Source: (RAMOS, 2017).

Some conclusions are drawn from the analysis of the related figures. On average, regular bins happen 11.87 times in a row (i.e., when a regular bin occurs once, is it highly probable the next 10.87 bins are from the regular type). Bypass bins also occur grouped, but at a smaller rate (on average 4.55 times in a row). The bypass bins result corroborates an update of the HEVC standard that has proposed to bypass bins to occur grouped, in order to increase throughput, due to its smaller dependencies in processing (SZE, 2013). Within a sequence of regular bins of both types, the regular MPS bins tend to happen 3.27 times in a row, whereas regular LPS bins 1.4 times in a row. As one may notice, in case some parts of the architecture are required only for a particular bin type, during the bins burst of another type of bin, the parts required by the other kind could potentially be turned off thus saving power at that moment.

4.4 Power-saving BAE Architecture Design

Considering at first a baseline BAE architecture based on the solution presented at (FEI, 2011), where a four-BAE cores design is presented (Figure 4.5), each with a four-stage pipeline, as already mentioned: 1- rLPS pre-generation; 2- *Range* Update; 3-*Low* Update; 4-Bitstream generation. The only difference compared to the original work of (FEI, 2011) was the insertion of the PN rLPS technique proposed in (ZHOU, 2013) due to its minor effort for implementation. Each core can process a single bin, either a regular or a bypass bin.

Figure 4.5 – Four-BAE core structure



Source: (RAMOS, 2016).

The architecture requires adders/subtractors to achieve (3.3) and (3.4), and, as already raised on section 4.2, those combinational structures are required only for determined types of bins. Moreover, the baseline architecture has four times these components, since it is a four-BAE-core design. Therefore, the inputs of these components can be gated for the current non-required types of bins (which, as stated in section 4.3, tend to occur grouped, i.e., in a row). The chosen technique is the Operand Isolation, where an AND-gate is applied to each input of the adders/subtractors, where one input is the original value, and the other is an enable signal. When the enable is active, the original input will be fed to the gated logics. In case the enable is deactivated, zeroed inputs enter the gate logic, for several clock cycles (refer again to section 4.3). The two adders suitable for the Operand Isolation are the one required for the *Low* update of an LPS symbol (3.3), and the adder required for the *Low* update for a bypass bin (3.4). The subtractor required for the update of *Range* was chosen not to be gated, since, during a burst of bypass bins, the *Range* variable is not updated at all, which already guarantees that no switching activity is occurring at this component, without the use of the Operand Isolation technique.

Some registers are used to store and to posterior transmission between each pipeline stage, considering the 4-stage pipeline structure. One may notice that not all of these values require to be updated for each currently processed bin. Hence, a Clock Gating approach can be used. The Clock Gating technique utilizes an enable, which indicates where the clock of the selected registers can be turned off. Thus, no write occurs, nor switching activity generated due to the clock toggling into the registers. There are three potential macro-situations, which are described below:

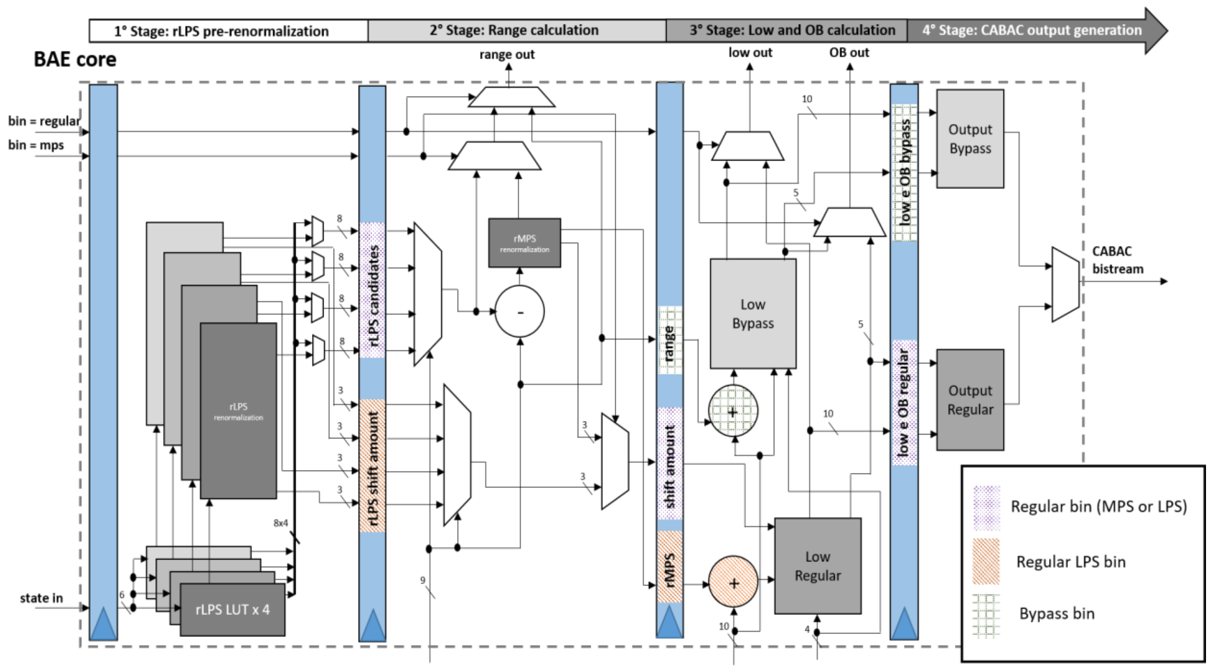
1. **Regular bins of any type (MPS or LPS):**
    a. At the present proposal, the PN rLPS technique is applied at the first pipeline stage. Therefore, four candidates for the rLPS value are generated, since the final decision of which is the correct to be used requires the two most-significant bits of Range variable (ITU-T, 2013), which will be available at the second pipeline stage. The registers for these values are 8-bits wide, performing 32 Flip-flops required only for regular bins.
    b. The other value is the shift-amount required by the *Low* update in case the *Range* was also renormalized (which is done through shift-left operation, i.e., multiplications by power of two). The same shift-amount applied to the *Range* variable shall be applied to the *Low* variable (refer to Figure 3.3). Therefore, the shift-amount shall be registered between the second to the third pipeline stage, to be used at the *Low* update stage. There is a 4-bits register required only when the current bin is a regular one.
    c. Furthermore, the updated *Low* and OB values are required for the bitstream generation at the fourth stage (refer to Figure 3.3 and Figure 3.5). Hence, 10-bit and 5-bit registers lie in the same situation presented for the aforementioned registers.
2. **Bypass bins:**
    a. As seen in (3.4), the *Range* variable is summed to the multiplied-by-two *Low* variable in case a bypass bin with value ‘1’ occurred. Therefore, the current *Range* value shall be transmitted through the pipeline register between the second and third stages, only when a bypass bin is the current one (which implies in 9-bits register for this situation).
    b. Furthermore, the updated *Low* and OB variables are also required for bitstream generation in case a bypass bin has occurred, leading to 10-bit and 5-bit registers that are gated for that reason.
3. **Regular LPS bin only:**
    a. In case the current bin is a regular LPS at the first pipeline stage, due to the use of the PN rLPS technique, there will be the generation of four shift-amount values for the already pre-renormalized *Range*

candidates. Only at *Range* stage the decision of which one of them is the correct will occur, according to the proper *rLPS* value chosen (i.e., both value, *rLPS* and rLPS-renormalization-shift-amount require the two most-significant bits of *Range* for the final decision). Therefore, four 4-bits values are required only for LPS bins (for MPS bins, the renormalization value is decided only at the second pipeline stage, and the *rLPS* shift-amount generated are unnecessary). Moreover, the already renormalized four *rLPS* candidate values are necessary to update the *Range* variable in case the current bin is an LPS (that is the main reason to remove the LPS renormalization from the second stage and thus decrease the critical path). Hence, more four 8-bits registers may be gated in any case besides a regular LPS bin.

b. The *Low* update for an LPS bin requires the *rMPS* as seen in (3.3). Therefore, the cited value has to be transmitted through the pipeline registers between the second and third stages (a 9-bit register).

The above-described situations, for the adders and pipeline registers, are depicted in Figure 4.6, where the full architecture of the proposed BAE-core design appears. One may remember that four of these cores are used for the whole design, in order to achieve 4-bin/cycle throughput, which leads to an increasing power saving potential by the use of the fine-grain low-power techniques insertion proposed.

Figure 4.6 – Single BAE core with the power saving techniques



Source: (RAMOS, 2016).

4.5 Synthesis Results and Comparisons

Three versions of the architecture were synthesized for Nangate 45nm CMOS PDK. The three versions are 4-cores BAE using only Clock Gating; 4-cores BAE using only Operand Isolation; and 4-cores BAE using both power savings techniques. Moreover, a baseline architecture, without the selected low-power techniques above mentioned, was also described and synthesized for the same mentioned scenario, for means of comparison. Power stimuli were provided, emulating the percentage of bins types presented in (ZHOU, 2015), in order to achieve more realistic power values. The power values are presented in Table 4.2, where the results are divided for the designed architectures, and also according each of the four cores (numbered from 0 to 3). As one may see, by using only Clock Gating, power savings ranging 10 to 16% are achieved, compared to the baseline design. Using only Operand Isolation, 12 to 28% of savings occurred. Finally, using Clock Gating and Operand Isolation at the same time, power savings ranging 25 to 40% were reached.

Table 4.2 – Power results of the power-saving techniques within four-BAE cores design

	Baseline	**Clock Gating**	**Operand Isolation**	**Clock Gating + Operand Isolation**
BAE 0	1.076 mW	0.971 mW	0.839 mW	0.675 mW
BAE 1	1.002 mW	0.887 mW	0.728 mW	0.604 mW
BAE 2	0.897 mw	0.756 mW	0.667 mW	0.545 mW
BAE 3	0.893 mW	0.793 mW	0.792 mW	0.670 mW
Power Saving from Baseline (Range)	-	**10%-16%**	**12%-28%**	**25%-40%**

Source: (RAMOS, 2016).

Table 4.3 provides comparisons with related works. The main conclusion drawn is that the proposed design can accomplish the constraints for 8K UHD real-time video processing, which is around 1-Gbin/s (CHEN, 2015), while having the smaller power consumption per Gbin processed (i.e., normalized power value for means of fair comparison with related works that present power consumption values). Moreover, for the designs that present power values, it is inferred that the results come from tool-inferred stimuli since the authors do not state what were the stimuli used for the values accomplished.

Table 4.3 – Comparisons with related works of the power-saving BAE design

Design	(LIU, 2011b)	(CHEN, 2010)	(ZHOU, 2015)	(KUO, 2006)	(FEI, 2011)	(PENG, 2013)	(VIZZOTTO, 2015)	**Our Work**
Clock Frequency (MHz)	238	222	420	-	279	357	380	**280**
# of bins/cycle	2	1~8	4.37	-	4	1.43	2.37	**4**
Mbins/s	634	1776	1836	200	1116	439	900	**1120**
CMOS Technology (nm)	90	130	90	180	90	130	130	**45**
Gate count (K)	3.9	14.7	7.98	-	8.22	24.9	31.1	**9.95**
Power Dissipation (mW)	4.9	-	-	20.7	-	-	-	**2.49**
mW/Gbin	7.72	-	-	103	-	-	-	**2.24**
Supports 8K videos	No	Yes	Yes	No	Yes	No	No	**Yes**

Source: (RAMOS, 2016).

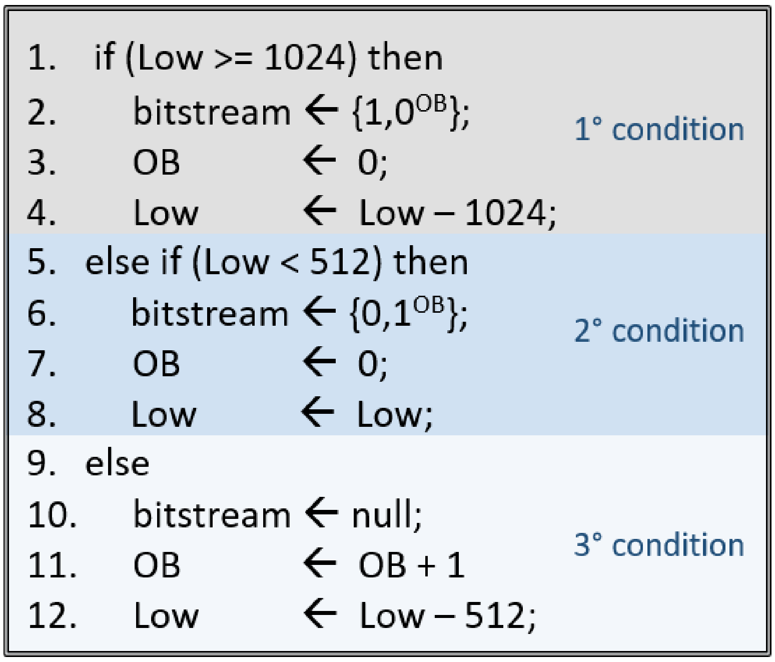
5 NOVEL MULTIPLE-BYPASS BINS PROCESSING FOR HEVC BAE

This chapter will present a novel proposal for multiple-bypass bins processing, named here Multiple-Bypass Bins Scheme (MBBS), where up to two bypass bins occurring in a row can be processed within a single BAE core, considering the baseline BAE design proposed at (FEI, 2011). Moreover, the proposed MBBS is applied to the baseline design, along with the low-power techniques already mentioned on the previous chapter. Simulation using real video sequences and synthesis results are provided, and comparisons with related BAE works are presented at the end.

5.1 Low Update and Renormalization for Bypass Bins

The *Low* variable has to be updated during a bypass bin processing, along with OB. The bitstream may or may not be generated for the current bypass bin. Figure 5.1 presents the pseudo-code for renormalization *Low*, OB update, and bitstream generation of a single bypass bin, based on Figure 3.4 and Figure 3.5, derived from the standard (ITU-T, 2013). The symbol $\{1,0^{OB}\}$ represents a concatenation of a value ‘1’, followed by the value ‘0’ *n* times, whereas $\{0,1^{OB}\}$ corresponds to the concatenation of the value ‘0’ followed by the value ‘1’ *n* times, i.e., is another way to represent the *PutBit* function. In both situations, *n* is the current value of the OB variable. When no bitstream is generated, it is pointed to as null in Figure 5.1.

Figure 5.1 – Pseudo-code for Low renormalization, OB update and bitstream generation for bypass bin



```
if (Low >= 1024) then
    bitstream ← {1,0^OB};
    OB        ← 0;
    Low       ← Low – 1024;
else if (Low < 512) then
    bitstream ← {0,1^OB};
    OB        ← 0;
    Low       ← Low;
else
    bitstream ← null;
    OB        ← OB + 1
    Low       ← Low – 512;
```

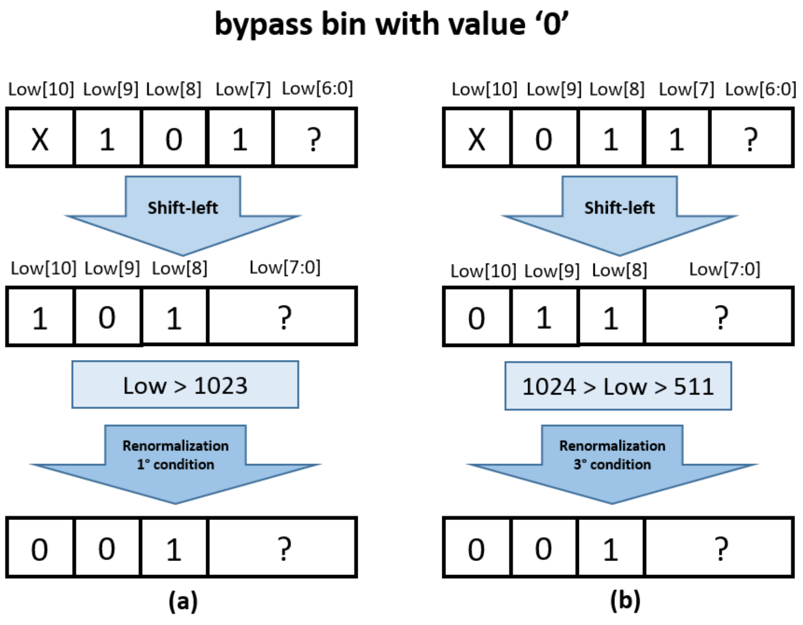
1° condition
2° condition
3° condition

Source: the author.

For the bypass renormalization flow, Figure 5.2, Figure 5.3, Figure 5.4, and Figure 5.5 present some possible behavior the variable *Low* may follow, based on (3.4) and Figure 3.4, where one may notice that the upper limit for the *Low* variable is 767 (MOFFAT, 1995). One important remark is that the *Range* value must always be above 256 after its renormalization (what implies that the ninth most significant bit of this variable has the value ‘1' when it is used for *Low* renormalization). One may see that the *Low* variable always starts with value zero (ITU-T, 2013), and is incremented with the *Range* variable, as shown in (3.4). Therefore, at the end of a given *Low* renormalization, we cannot have both *Low[9]* and *Low[8]* with the value ‘1’ (as already mentioned, the upper limit for *Low* is 767).

For instance, Figure 5.2(a), Figure 5.2(b), Figure 5.3(a), and Figure 5.3(b) start with different *Low* values, highlighting the four most-significant bits, and the posterior renormalization needed, considering only that a bypass bin with value ‘0’ has occurred. One may notice that a bypass bin with value ‘0’ implies that a simple shift-left is required at the variable, i.e., multiplication by two. For all the cases shown, after the update of the *Low*, the *Low[9]* is already zero or must be zeroed (i.e., a subtraction by 512), leading to the results presented, wherein all cases, we do not have both *Low[9]* and *Low[8]* with the value ‘1’.

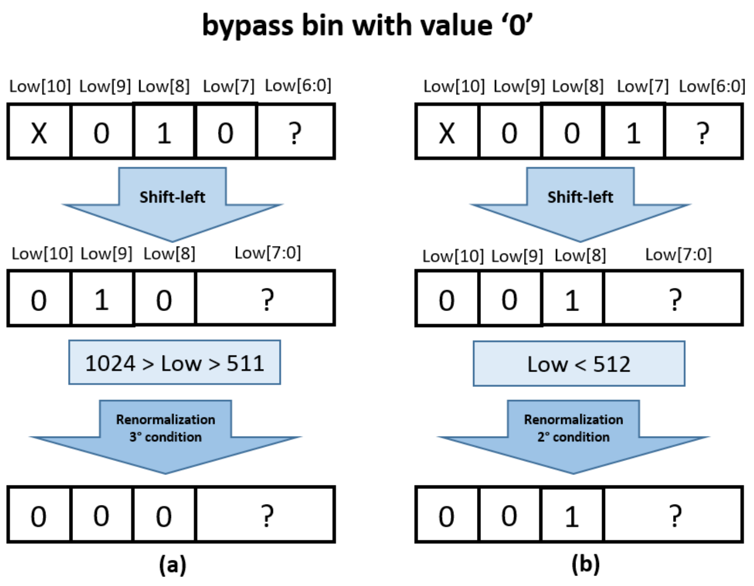
Figure 5.2 – Low renormalization possibilities when a bypass with value ‘0’ occurs – part 1


bypass bin with value ‘0’
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 1 0 1 ?
Shift-left
Low[10] Low[9] Low[8] Low[7:0]
1 0 1 ?
Low > 1023
Renormalization 1° condition
0 0 1 ?
(a)
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 1 1 ?
Shift-left
Low[10] Low[9] Low[8] Low[7:0]
0 1 1 ?
1024 > Low > 511
Renormalization 3° condition
0 0 1 ?
(b)

Source: (RAMOS, 2018b)

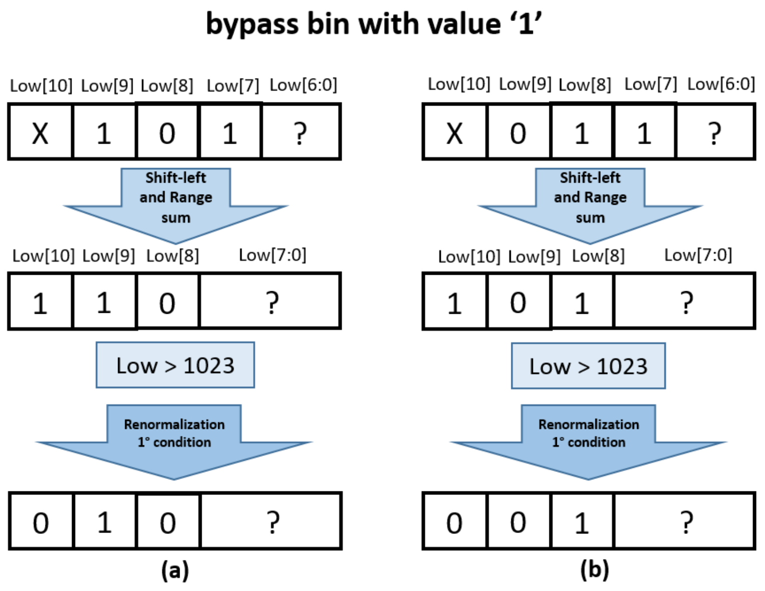
When the current bypass bin has the value ‘1’, Figure 5.4(a), Figure 5.4(b), Figure 5.5(a), and Figure 5.5(b) show the behavior of the *Low* variable. At first a shift-left is required (i.e., multiplication by two) and then a sum with the current renormalized *Range* (implying that the current *Range* value is at least 256, i.e., *Range[8]* has the value ‘1’). For all situations, the final renormalized *Low* would end with *Low[9]* equal to zero (either already having the value ‘0’, or needing to be zeroed). The exception is the condition presented in Figure 5.4(a), where the initial value may be the upper limit (i.e., *Low[9]* equal to ‘1’, *Low[8]* equal to ‘0’, and *Low[7]* equal to ‘1’), where the *Low[9]* must remain with value ‘1’, even after the renormalization process.

Figure 5.3 – Low renormalization possibilities when a bypass with value ‘0’ occurs – part 2


bypass bin with value ‘0’
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 1 0 ?
Shift-left
Low[10] Low[9] Low[8] Low[7:0]
0 1 0 ?
1024 > Low > 511
Renormalization 3° condition
0 0 0 ?
(a)
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 0 1 ?
Shift-left
Low[10] Low[9] Low[8] Low[7:0]
0 0 1 ?
Low < 512
Renormalization 2° condition
0 0 1 ?
(b)

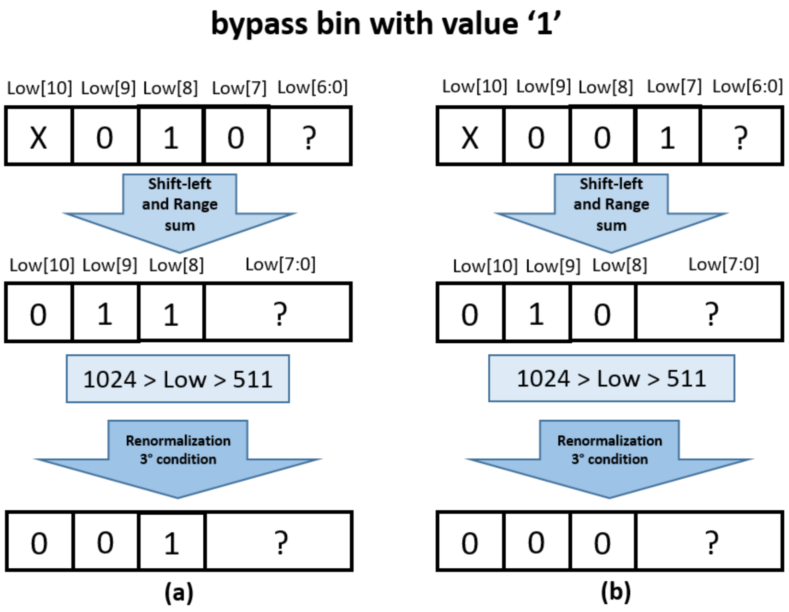
Source: (RAMOS, 2018b)

Figure 5.4 – Low renormalization possibilities when a bypass with value ‘1’ occurs – part 1


bypass bin with value ‘1’
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 1 0 1 ?
Shift-left and Range sum
Low[10] Low[9] Low[8] Low[7:0]
1 1 0 ?
Low > 1023
Renormalization 1° condition
0 1 0 ?
(a)
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 1 1 ?
Shift-left and Range sum
Low[10] Low[9] Low[8] Low[7:0]
1 0 1 ?
Low > 1023
Renormalization 1° condition
0 0 1 ?
(b)

Source: (RAMOS, 2018b)

Figure 5.5 – Low renormalization possibilities when a bypass with value ‘1’ occurs – part 2


bypass bin with value ‘1’
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 1 0 ?
Shift-left and Range sum
Low[10] Low[9] Low[8] Low[7:0]
0 1 1 ?
1024 > Low > 511
Renormalization 3° condition
0 0 1 ?
(a)
Low[10] Low[9] Low[8] Low[7] Low[6:0]
X 0 0 1 ?
Shift-left and Range sum
Low[10] Low[9] Low[8] Low[7:0]
0 1 0 ?
1024 > Low > 511
Renormalization 3° condition
0 0 0 ?
(b)

Source: (RAMOS, 2018b)

5.2 Multiple-Bypass Bins Scheme

One of the improvements of the HEVC standard compared to its predecessor is the increase in the total amount of bypass bins, since they have fewer data dependencies among them (SZE, 2013). Furthermore, as depicted in (3.4), they do not update the *Range* variable, which is the current critical path in the most recent BAE architectures (FEI, 2011), (ZHOU, 2015).

One further look at (3.4) shows that *Low* variable update (the only variable updated by bypass bins) may be rewritten in a general fashion as (5.1), for one up to *n* bypass bins. The updated *Low* variable for multiple bypass bins is called *Lb*, whereas *NumByp* corresponds to the total amount of bypass bins that are processed at a given round, and *ValuesByp* corresponds to the value of the bypass bins being processed in reverse order of incoming. In case a single bypass bin is processed, (5.1) is collapsed back to (3.4). Table 5.1 illustrate some examples to help clarify how the conclusion presented in (5.1) was achieved, where some arbitrary examples of incoming bypass bins with different values. In Table 5.1, the “L” stands for the *Low* variable, whereas “R” for the *Range* variable.

$$Lb \leftarrow (Low*2^{NumByp}) + (Range*ValuesByp) \qquad (5.1)$$

Table 5.1 – Examples of Low update for arbitrary bypass bins sequences

	Bypass bins sequence			
	1,0,1	1,1,1	0,1,1	1,0,0
Low for the 1st bin	2*L + R	2*L + R	2*L	2*L + R
Low for the 2nd bin	2*(2*L + R)	2*(2*L + R) + R	2*(2*L) + R	2*(2*L + R)
Low for the 3rd bin	2*(2*(2*L+ R) + R	2*(2*(2*L + R) + R) + R	2*(2*(2*L) + R) + R)	2*(2*(2*L + R))
Final Low expression	**8*L + 5*R**	**8*L + 7*R**	**8*L + 3*R**	**8*L + 4*R**

Source: the author.

The proposed Multiple Bypass Bin Scheme (MBBS) could process any number of bypass bins occurring at the same time, as a designer might want. Nevertheless, considering a hardware entropy encoder again, in case any number of bypass bins is processed in parallel, this may decrease the maximum frequency of the design. Therefore, the MBBS proposal limits up to two bypass bins processed at the same time, for the following reasons:

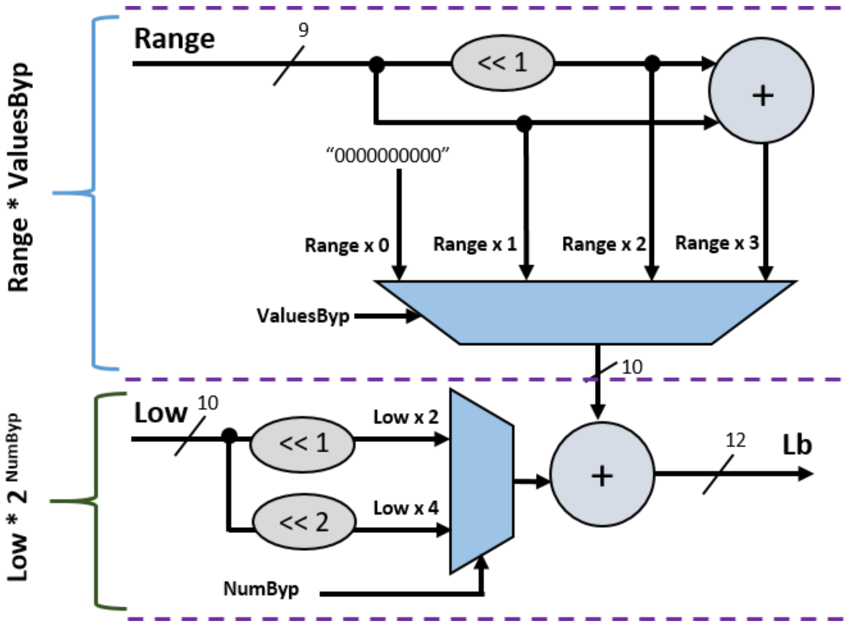
i. Only two adders and some shift-left logics are required for that approach, avoiding changing the critical path of designs without MBBS.
ii. Considering that bypass bins are around 25% (*p* in (5.2)) of the total of bins in HEVC (ZHOU, 2015), and using Amdahl Law as a guideline (AMDAHL, 1967), a potential overall maximum gain of about 14.28% (*St* in (5.2)) is expected by a factor of two speed-up (i.e., *Sp* in (5.2)) of bypass bins processing.

State-of-the-art BAE designs, as already presented, are multi-BAE cores approaches, where a single MBBS instance insertion may occur within each core. Thus, a BAE using MBBS would be able to process far more than two bypass bins.

$$St = \frac{1}{(1-p)+(\frac{p}{Sp})} \tag{5.2}$$

The kernel of MBBS implementation, which is (5.1), appears in Figure 5.6, in a multiplier-less fashion. Since, at most, two bypass bins are processed, the possible *ValuesByp* are 0, 1, 2 or 3, and the possible *NumByp* are either one or two (considering that at least one bypass bin will be processed at a given moment). Further explaining Figure 5.6 related to (5.1), the *Range* multiplication by zero is a vector of zeros; the *Range* multiplication by one is the original *Range* value; a shift-left of one position achieves the *Range* multiplication by two; and the *Range* multiplication by three is the *Range* multiplied by two added with the original *Range* value. The *Low* multiplication will be either by two or by four (power of two values) and thus are accomplished by a shift-left by one or by two positions, respectively. By doing so, the multiplications at (5.1) are achieved by the used of adders and shift logics. Moreover, as will be further explained, it avoids any increase in the critical path of the BAE logic for a multi-core BAE design.

Figure 5.6 – Structure for Low update of the MBBS



Source: (RAMOS, 2017).

As one may remember, the *Low* variable may undergo a renormalization, as already mentioned in Chapter 3, along with the update of the auxiliary variable OB, and the generation of the bitstream for the given bypass bins. Therefore, an alternative is required to implement the behavior depicted in Figure 3.4 and Figure 3.5 for one and two bypass bins, according to the amount being currently processed. In other words, the main objective is to find a form to analyze the *Lb* from (5.1) and verify in what situation presented in Figure 5.1 it shall fall, for each of the two bypass bins processed at the same time. Following the Arithmetic Coding fundaments in (MOFFAT, 1995), *Range* and *Low* must be bounded by the limits presented in (5.3), (5.4), and (5.5), considering that *Range* and *Low* are 9-bit and 10-bit variables, respectively (ITU-T, 2013). Thus, the *b* variable in the mentioned equations has to be 10 i.e., it is the number of bits necessary to represent the variables values (MOFFAT, 1995).

$$2^{b-2} < Range \leq 2^{b-1} \tag{5.3}$$

$$0 \leq Low < 2^{b} - 2^{b-2} \tag{5.4}$$

$$Range + Low \leq 2^{b} \tag{5.5}$$

Considering the limits presented, *Low* after any renormalization must be equal or below 767, which is a 10-bit vector of binary values represented by 1011111111_2. In other words, for any given situation, if the *Low[9]* is equal to '1', the *Low[8]* has to be '0' . The

Range is updated only for regular bins and must have a value of at least of 256, which means the *Range[8]* always has to be '1' – refer to (5.3).

For two bypass bins, the first thing to notice is that *Lb* is a 12-bit variable since it undergoes a multiplication by four (i.e., a shift-left by two positions), which will lead to two more positions on the original 10-bit *Low* variable. The nine less significant bits of *Low* will also remain the same (i.e., *Low[8:0]*). As already seen in Figure 5.2, Figure 5.3, Figure 5.4, and Figure 5.5, if we make any combination between two incoming bypass bins with any value in a row, the tenth bit (i.e., *Lb[9*]) will either have the value '0', or shall be zeroed. The exception occurs in case we have two bypass bin with'1' occurring in a row, and the *Low* initial value is above 640 (i.e., *Low[9:7]* is equal to '101'), requiring the 1st renormalization condition of Figure 5.1 twice. Furthermore, when a sequence of bypass bins with value '1' and '0' occurs, respectively, and the *Low* is again above 640, the 1st renormalization condition will be required twice. The difference is that, this time, *Lb[9]* will already have the value '0'. Thus, we can consider that, when the MBBS is processing two bypass bins, the final *Low[9]* always will be zeroed, except by the case that the 1st renormalization condition is required for both bypass bins processed, which is indicated by *Lb[11]* and *Lb[10]* having both the value '1'. The summary of MBBS *Low* renormalization is presented in (5.6), considering two bypass bin processing only.

$$\begin{cases} Low[8{:}0] \leftarrow Lb[8{:}0], & \\ Low[9] \leftarrow Lb[9], & if\ Lb[11{:}10] = 11 \\ Low[9] \leftarrow 0, & otherwise. \end{cases} \tag{5.6}$$

The OB variable also has to be updated, now considering that up to two bypass bins may be processed, which can be seen at (5.7). We have to look at the twelfth, eleventh, and tenth bit of the *Lb* variable to verify which of the situations presented at Figure 5.1 have occurred for the first and the second incoming bypass bins to be processed in parallel (i.e., *Lb [11]*, *Lb[10]*, and *Lb[9]*). All possible situations are depicted in Figure 5.7.

.

$$\begin{cases} OB \leftarrow 1, & if\ Lb[10{:}9] = 01 \\ OB \leftarrow OB + 2, & if\ Lb[11{:}9] = 011 \\ OB \leftarrow 0, & otherwise. \end{cases} \tag{5.7}$$

In case *Lb[10:9]* has the value '01', for the first incoming bypass bin, the value of the *Low* variable was either above 1023 or below 512, seen at Figure 5.7(a), meaning that the OB value has to be zeroed (1st or 2nd condition at Figure 5.1). After the shift, the final value of *Lb[10]* is zero (i.e., for the second bypass bin), meaning that the current updated *Lb* has a value equal to or greater than 512 (but still small than 1024, since the *Lb[10]* has the value ‘0’). Thus, a sum by one to the current OB value is necessary (3rd condition at Figure 5.1). Since it was zeroed for the first incoming bypass bin, the final updated OB value for this case is merely one.

When the twelfth, eleventh and tenth bit of the *Lb* have respectively the value ‘0’, ‘1’, and ‘1’ (i.e., *Lb[11:9]* = ‘011’), this means that the *Lb* should have had two values equal or greater 512, but smaller than 1024 for the first and second incoming bypass bins, depicted in Figure 5.7(b). The reason because it cannot have a third value ‘1’ at *Lb[11]* was presented during the explanations of Figure 5.2 and Figure 5.3. For the first incoming bypass bin, the value of the *Lb* is clearly between 1024 and 511, leading to the 3rd situation in Figure 5.1. That would require a renormalization (subtraction by the value 512, i.e., zeroing the *Lb[9]* if only a single bypass bin has occurred). After the shift, the next value would be again between 1024 and 511 (because the new *Lb[9]* bit is ‘1’, and the *Lb[10]* would have been zeroed, if done separately for the two bypass bins). The situation leads to adding one twice to the original value of OB (i.e., the original value of OB has to be added by the value two due to the 3rd condition of Figure 5.1 being occurring twice).

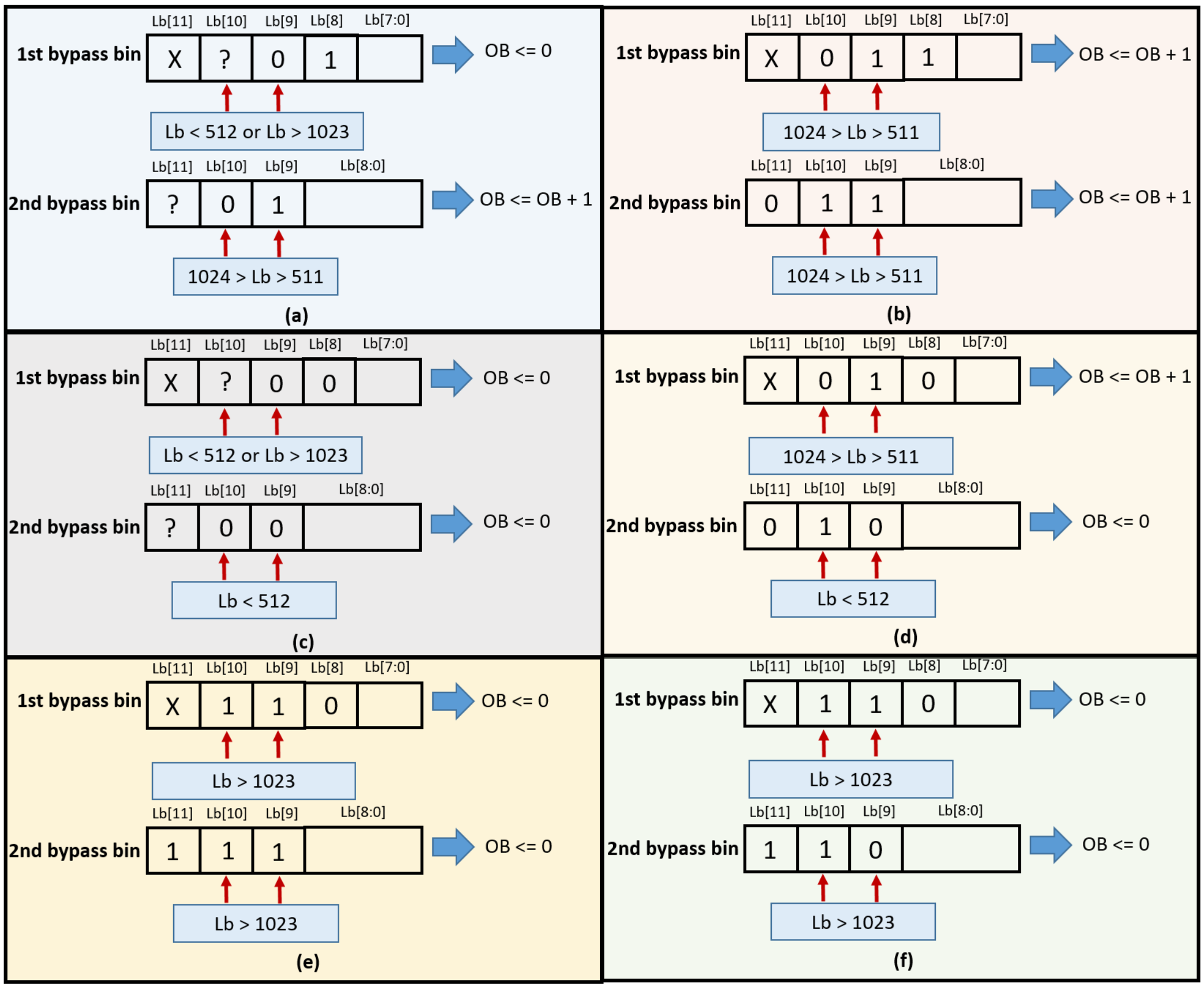
When both values of the eleventh and tenth bit are ‘0’, presented in Figure 5.7(c), it cannot be said whether, for the first bypass bin, the value of *Lb* was above 1023 or was smaller than 512 (the reading of *Lb[11]* is required to discover which condition occurred). Nevertheless, since the *Lb[9]* bit is ‘0’, and *Lb[10]* was also ‘0’, one can assure that the final *Lb* value is smaller than 512, which leads to OB to be updated with the value ‘0’ (2nd condition of Figure 5.1 for the second incoming bypass bin).

The OB variable also needs to be zeroed if *Lb[11:9]* has the value ‘010’, depicted in Figure 5.7(d). It is known that, for the first incoming bypass bin, *Lb* value is between 1024 and 511 (since the *Lb[10]* must be zero). Moreover, it is known, for sure, that its final value (i.e., for the second bypass bin) is below 512 (since *Lb[10]* should have been zeroed for the first bypass bin before the shift), thus leading to the 3rd condition of Figure 5.1, requiring the zeroing of OB. Nevertheless, due to the usage of MBBS, the eleventh bit of *Lb* (i.e., *Lb[10]*) that should have been, otherwise, zeroed for the first bypass bin, and it is not, exactly because MBBS does not renormalize the values during the intermediary step of the process (i.e., in between the processing of the two bypass bins).

In Figure 5.7(e) and Figure 5.7(f), we have the situations where the 1st renormalization condition was required twice, which is indicated by *Lb[11]* and *Lb[10]* having both the value ‘1’. Therefore, the OB variable shall be zeroed for both bypass bin, receiving the value zero for these situations.

The bitstream when two bypass bins at the current round will follow what is depicted in the pseudo-code at Figure 5.8. One has to take into account the *Lb[11]* (i.e., twelfth bit of *Lb)* because there is the need to differentiate whether the OB was zeroed (when zeroed) due to the first or to the second incoming bypass bin. The ~$Lb[11]^{OB}$ means the generation of a vector on the bitstream with the negation of *Lb[11]* replicated OB times.

Figure 5.7 – OB update for two consecutive bypass bins



Source: (RAMOS, 2018b)

When the tenth bit of *Lb* (i.e., *Lb[9]*) is zero, and *Lb[11]* is not equal to *Lb[10]*, one knows that the final renormalized *Low* will be below 512, for the same reasons presents for the OB update (refer to Figure 5.7(c) and Figure 5.7(d)). Thus, there will be bitstream generation at the current round (refer to 2nd condition at Figure 5.1). If *Lb[11:10]* value is

‘01’, the same situation presented in Figure 5.7(d) has occurred. For the first incoming bypass bin, no bitstream is generated; and for the second bypass bin, the bitstream will follow the 2nd condition of Figure 5.1, added by one extra bit due to the previous incoming bypass bin (i.e., OB was incremented by one at the first bypass bin). If *Lb[11:10]* = ‘10’, the situation presented at Figure 5.7(c) has occurred (but considering that *Lb[11]* has the value ‘1’) and it would have happened, at the beginning of the round, a *Low* value above than 1023. Therefore, the need to follow the bitstream behavior of the 1st condition at Figure 5.1, with the original OB value before update. After that, the *Low* variable would end with a value smaller than 512, leading to the 2nd condition of Figure 5.1. Nevertheless, since the OB was zeroed for the previous bin, a final bit ‘0’ is put into the bitstream. The lines 1 and 2 of the pseudo-code in Figure 5.8 contemplate both situations.

Considering again that *Lb[9]* is zero, both *Lb[11]* and *Lb[10]* may have the same value.. Therefore, this scenario would always lead to the situation where, at the first bypass bin, the *Lb* value was below 512, leading to the related 2nd condition of Figure 5.1 and zeroing the OB, generating a bitstream with the OB value before the update. When the next bypass bin occurred, the value of *Lb* was kept below 512. Thus, the same behavior of the previous bin has to be followed (but now the OB variable was already zeroed at the previous round, generating a single final bit ‘0’ at the bitstream). Furthermore, when *Lb[11]* and *Lb[10]* have both the value ‘1’, as already mentioned, the 1st renormalization conditions was needed twice (i.e., the *Low* was equal or above to 1024 for the two bypass bins), and the same bitstream behavior presented before is applied. These situations are the same presented at Figure 5.7(c), Figure 5.7(e), and Figure 5.7(f), and the lines 3 and 4 of Figure 5.8 are used for that purpose.

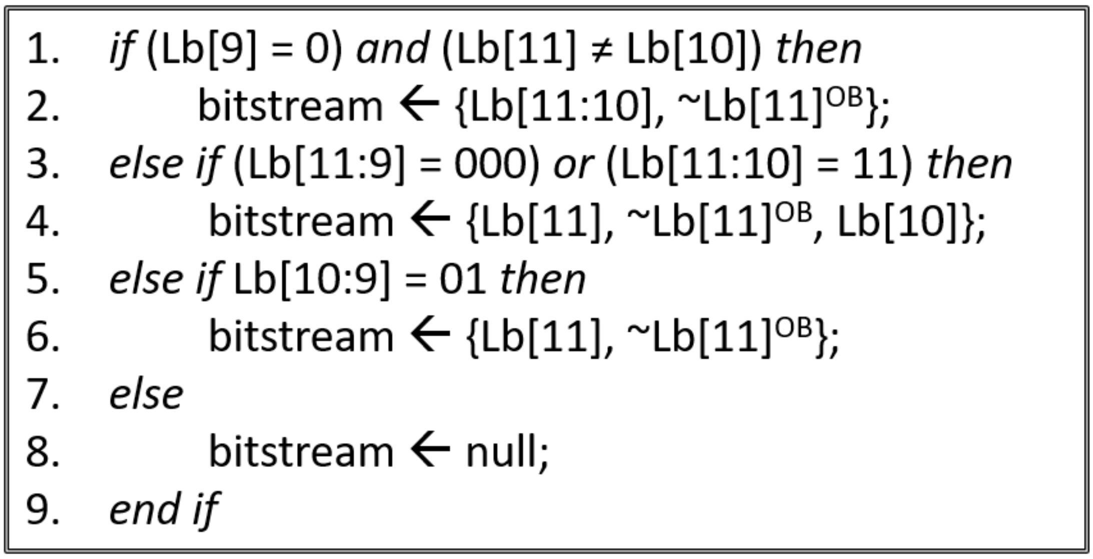
The other situation that leads to bitstream generation for two bypass bins happens when *Lb[10:9]* have the value ‘01’(refer to Figure 5.7(a)). The first value of *Lb* could be either above 1023 or below 512 for the first bypass bin, leading to either 1st or 2nd condition of Figure 5.1, generating the bitstream for that bin, or zeroing the OB. After that, for the second bypass bin, the value of *Lb* is between 1024 and 511 (3rd condition at Figure 5.1). Thus, no bitstream is generated for that bin, and the OB variable receives the value one, the first condition at (5.7). The lines 5 and 6 of Figure 5.8 contemplate that condition of bitstream generation

The remaining situation is when *Lb[9]* and *Lb[10]* have both the value ‘1’, and *Lb[11]* has the value ‘0’ (refer to Figure 5.7(b)). For that situation, it would happen, for each of the two incoming bypass bins, an *Lb* with a value between 511 and 1024 (3rd condition at Figure

5.1). Thus, no bitstream is generated at that round for the bypass bins; only the OB is updated, according to the second condition presented at (5.7). The lines 7 and 8 of Figure 5.8 shows how that situation occurs. As mean of further clarification, Appendix B of this Thesis presents a different analysis to derive the MBBS technique.

For all others situations, i.e., the updated of *Range*, *Low*, OB, and bitstream generation for regular bins; or for a single occurring bypass bin followed by a bin of another type, the behavior of the referred variables follows the same proposal of (LIU, 2011b), which, in a nutshell, corresponds to (3.6), (3.7), and (3.8).

Figure 5.8 – Pseudo-code for bitstream generation of MBBS for two bypass bins



```
1.  if (Lb[9] = 0) and (Lb[11] ≠ Lb[10]) then
2.          bitstream ← {Lb[11:10], ~Lb[11]$^{OB}$};
3.  else if (Lb[11:9] = 000) or (Lb[11:10] = 11) then
4.          bitstream ← {Lb[11], ~Lb[11]$^{OB}$, Lb[10]};
5.  else if Lb[10:9] = 01 then
6.          bitstream ← {Lb[11], ~Lb[11]$^{OB}$};
7.  else
8.          bitstream ← null;
9.  end if
```

Source: (RAMOS, 2018b)

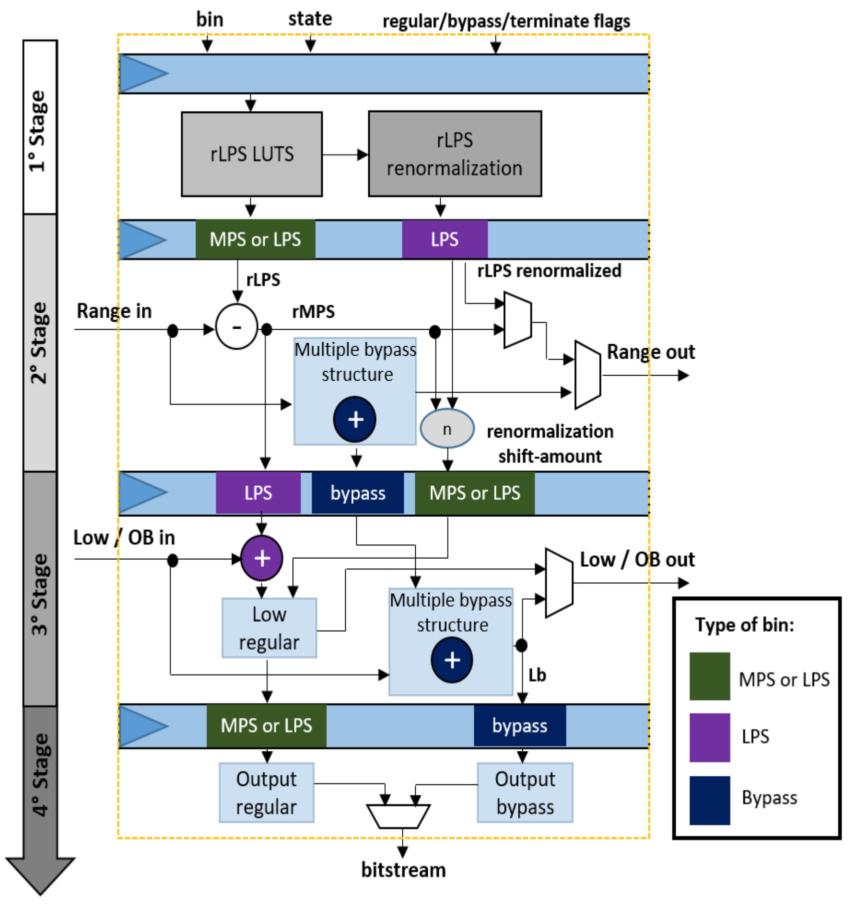
5.3 Results of MBBS and Power-saving Approach in BAE Design

The proposed MBBS was inserted in a four-core BAE design, derived from (FEI, 2011), and using the PN rLPS technique, due to its simplicity. The low-power approach, first proposed in Chapter 4, is applied to the new version of the BAE design, as is depicted in Figure 5.9. The main difference is that now two adders are used due to the insertion of MBBS, and are required only if one or two bypass bins occurred at the given cycle for that BAE core. Therefore, the Operand Isolation technique was applied to the mentioned adders. This version received the name LPBS-BAE.

Along with LPBP-BAE, two other versions of the BAE design are provided, for comparisons purposes: a baseline design the same as the work of (FEI, 2011), plus the PN rLPS, named BA-BAE; and the same design of LPBP-BAE, but without the power-saving techniques, named BP-BAE. The three BAE versions were described in VHDL and synthetized for ST 65 nm PDK, using Cadence RTL Compiler tool.

The first result is to verify if any degradation in frequency occurs in the case of the insertion of the MBBS and the low-power approach (i.e., the LPBP-BAE compared to BA-BAE). Therefore, the use of the work of (FEI, 2011) is a possible start, due its simplicity of design compared to others (ZHOU, 2013), (ZHOU, 2015), and to avoid biasing any results due to the proposed novel techniques by the other works. As a result, the maximum frequency achieved by BA-BAE was 268 MHz, whereas for LPBP-BAE was 264 MHz. Hence, the degradation in frequency is negligible (around 1.5%). That result was expected, since the use of the MBBS and the low-power techniques does not directly insert any logic into the critical path of (FEI, 2011), which is the *Range* update (i.e., the second stage). Nevertheless, since the *Range* variable is used for the *Low* update of MBBS, an increase into the capacitance of the critical path is possible, and that is the probable explanation for the minimal degradation in frequency.

Figure 5.9 – BAE core with MBBS and power-saving techniques



Source: (RAMOS, 2017).

The second result shown in Table 5.2 is related to the increase in throughput of bins per cycle comparing again BA-BAE against BP-BAE/LPBP-BAE (the bins per cycle of both architectures using MBBS is the same, whereas for BA-BAE is always 4-bins/cycle (FEI, 2011)). The bins/cycle will depend on the intrinsic behavior of a given video sequence for the versions with MBBS (i.e., the number of bypass bins and the consecutive occurrence of them will increase the throughput by different amounts). Hence, the only way to discover that value is through architecture simulation using recommended video sequences as stimuli. The same sequences, for the same configurations and same QPs values used at Chapter 4 were applied. The MBBS designs achieved an average of around 14.36% more bins/cycle compared to the BA-BAE, which corroborates the theoretical throughput gain as raised in (5.2). One may notice that the maximum frequency is almost the same for the versions with and without the MBBS. The video sequences with a higher amount of bypass bins as a whole (ZHOU, 2015) and with the larger amount of bypass bins consecutively occurring tend to have better results when compared to the others (i.e. Kimono and PeopleOnStreet sequences). One important remark is the fact that the throughput of bins/cycle depends only on the architecture description (i.e., RTL), and not on the synthesized version of the design.

Table 5.2 – Throughput increase related to baseline BAE by using MBBS

Video	**Configuration**		**Bins/Cycle – BA-BAE**	**Bins/ Cycle – BP-BAE and LPBP-BAE**	**Throughput Increase (%)**
BasketballDrive (1920x1080)	LD	22	4	4.52	**13.28**
		37		4.27	**7.02**
	RA	22		4.46	**11.78**
		37		4.34	**8.77**
BQTerrace (1920x1080)	LD	22		4.67	**17.04**
		37		4.31	**8.02**
	RA	22		4.65	**16.54**
		37		4.37	**9.52**
Kimono (1920x1080)	LD	22		4.94	**23.81**
		37		4.34	**8.77**
	RA	22		4.96	**24.31**
		37		4.45	**11.53**
PeopleOnStreet (2560x1600)	LD	22		4.82	**20.80**
		37		4.52	**13.28**
	RA	22		4.80	**20.30**
		37		4.54	**13.78**
Traffic (2560x1600)	LD	22		4.76	**19.30**
		37		4.38	**9.77**
	RA	22		4.73	**18.55**
		37		4.43	**11.03**
Average			**4**	**4.56**	**14.36**

Source: adapted from (RAMOS, 2017).

The third result is the power consumption comparison between the BP-BAE and LPBP-BAE. The same reasoning of the second analysis is applied: in order to achieve an accurate and realistic power analysis, real video sequences are required, since the parts of the architecture that shall be turned off depend on the statistics of a given video sequence. The same sequences, configurations, and QPs used by the bins/cycle throughput analysis were used for the power consumption comparison as stimuli, using the gate-level netlist of both versions. The results are presented in Table 5.3. The use of the low-power techniques accomplishes an average of 14.26% of power savings.

Table 5.3 – Power values and power savings of related BAE architectures

Video	Configuration		Power – BP-BAE (mW)	Power – LPBP-BAE (mW)	Power Savings (%)
BasketballDrive (1920x1080)	LD	22	18.82	16.41	**14.69**
		37	17.82	15.49	**15.04**
	RA	22	18.66	16.26	**14.76**
		37	17.66	15.37	**14.90**
BQTerrace (1920x1080)	LD	22	17.77	15.48	**14.79**
		37	18.24	15.94	**14.43**
	RA	22	17.79	15.60	**14.04**
		37	18.17	15.8	**14.56**
Kimono (1920x1080)	LD	22	16.39	14.30	**14.62**
		37	18.26	15.97	**14.34**
	RA	22	16.58	14.45	**14.74**
		37	18.29	16.02	**14.17**
PeopleOnStreet (2560x1600)	LD	22	18.21	15.82	**15.11**
		37	19.04	16.76	**13.60**
	RA	22	18.20	16.26	**11.93**
		37	19.09	16.83	**13.43**
Traffic (2560x1600)	LD	22	18.35	16.07	**14.19**
		37	19.08	16.73	**14.05**
	RA	22	18.71	16.41	**14.02**
		37	19.09	16.77	**13.83**
Average			**18.21**	**15.93**	**14.26**

Source: adapted from (RAMOS, 2017).

Finally, a comparison between LPBP-BAE and some literature related works is presented in Table 5.4. All works, besides (LIU, 2011b), can process 8K UHD video since they have a maximum throughput above 1-Gbin/s (CHEN, 2015). The proposed LPBP-BAE achieves the mentioned throughput constraint with the smaller clock frequency, thus showing the suitability of the MBBS for low-power design. Considering also low QP (i.e., only the QP with value 22), the minimum frequency is even smaller (one may notice that low QP refers to a higher video quality when compared to high QP sequences, i.e., less degradation in quality).

The maximum bin/s of LPBS-BAE is below the ones achieved by (ZHOU, 2013) and (ZHOU, 2015). Nevertheless, some works do not strictly cite any of the PVT conditions for the synthesis, or at least do not cite the process used, which is a reasonable explanation for the value accomplished by our work compared to them (our synthesis was made at the worst corner conditions: worst process, 125°C of temperature, and 0.95 V). Moreover, since the comparison between BA-BAE (i.e., the work of (FEI, 2011) with the PN rLPS) and the LPBP-BAE presented negligible frequency degradation, the use of the MBBS and the low-power approach cannot be considered as the explanation of the overall smaller clock frequency.

Regarding power consumption, only (LIU, 2011b) also presents power values, which are smaller than presented by LPBP-BAE. Nevertheless, (LIU, 2011b) did not use real sequences as stimuli and thus its results could be extremely optimistic, since one cannot attest which was the switching activity inferred by the authors, whereas the results of LPBP-BAE tend to be realistic, due to the usage real sequences stimuli.

Table 5.4 – Comparison with related BAE prior-arts works

Design	(LIU, 2011b)	(FEI, 2011)	(ZHOU, 2013)	(ZHOU, 2015)	**LPBP-BAE**	
					Low QP	**Average QP**
#bins/cycle	2	4	4.4	4.37	**4.73**	**4.56**
Minimum Frequency for 8K UHD	500 MHz	250 MHz	227 MHz	229 MHz	**211 MHz**	**219 MHz**
Maximum frequency	238 MHz	279 MHz	402 MHz	420 MHz	**264 MHz**	
CMOS Technology	90 nm	90 nm	65 nm	90 nm	**65 nm**	
Gate Count	3.9 K	8.22 K	57.3 K*	64.1 K*	**14.6 K**	
Power	4.9 mW	-	-	-	**15.93 mW**	
Power Stimuli	User-defined	-	-	-	**Real Sequences**	

* Whole CABAC results without memory

Source: (RAMOS, 2018b)

6 ARCHITECTURAL DESIGNS FOR HIGH-THROUGHPUT BAE USING MULTIPLE-BYPASS BINS SCHEME

This chapter presents two BAE architectural designs by using the novel MBBS presented in the previous chapter. The two solutions provide high-throughput bins per cycle and bins per second, being able to achieve the constraints for 6.2 High Tier real-time video processing, surpassing or with a close throughput compared to prior-art designs, with different advantages and drawbacks.

6.1 Ultra-High-Throughput BAE Architecture with MBBS

The first design consists of the usage of the MBBS technique along with the proposals presented in (ZHOU, 2015). Furthermore, a new pipeline scheme is presented, in order to keep the critical path with the same delay as the baseline design.

The BAE architecture, named here MB-BAE, is composed of an 8-stage pipeline structure. This new pipeline structure is required to avoid compromising the critical path of MB-BAE compared to prior-arts design, and will be further explained. The design utilizes PN rLPS, LH rLPS, BPBS and MBBS techniques, as following (one remark is the fact that BPBS split the bins stream of regular and bypass bins, hence the first pipeline stages do not process bypass bins):

(i) **Pre-selection and pre-renormalization of *rLPS***: the first pipeline stage corresponds to the pre-selection and pre-renormalization (i.e., PN rLPS), by utilizing the *State* variable. Since the choice of the *rLPS* also requires the first two bits of the *Range* variable (which are available at the second stage), four candidates are chosen and passed to the next pipeline stage. Along with the *rLPS* candidates, the already renormalized values are transmitted, and the shift-amount required for that renormalization (which is required for the *Low* update stage, in case an LPS is the current bin).

(ii) ***Range* update**: the second pipeline stage is used for the *Range* update, whether the current bin is an MPS or an LPS. At this point, the two most significant bits of the *Range* variable are available. Therefore, the proper LPS candidate is chosen for the update. In case the current bin is an LPS, the correct already renormalized value is used to update the *Range* variable, and the correspondent shift-amount is therefore transmitted to the next pipeline stage. The non-renormalized *rLPS* value is also used to generate the *rMPS*, which is required for the *Low* update of an LPS bin, according (3.3). Otherwise, if the current bin is a MPS, the non-renormalized correct *rLPS* candidate is chosen to generate the *rMPS*, which is the new *Range* value. One remark is the fact that *rMPS* may require renormalization of a single shift-left position occasionally (ZHOU, 2015). Since the *Low* variable is not updated for a MPS bin, there is no need to transmit the *rMPS* value for this type of regular bin.

(iii) **Regular PIPO write**: the third stage is the regular PIPO storage of the required values for the *Low* update of the current regular bin, or for the following bypass bin. The stored values are regular bin type (i.e., LPS or MPS); current updated *Range* (required if the next bin is a bypass with value '1'); current *rMPS* (required for the *Low* update if the current bin is an LPS); shift-amount used for *Range* renormalization (required for *Low* renormalization); next bin type (to indicate if the next bin is a regular or a bypass i.e., which PIPO to look at for the next bin to be read). Figure 6.1 depicted the organization of a single position of the regular PIPO structure.

(iv) **Sequence Merge**: as required for the BPBS technique, the bins stream for regular, and bypass symbols shall be merged right before the *Low* update stage. Therefore, at the fourth pipeline stage, that event happens. Both regular and bypass PIPOs indicates where is the next bin to be read. Moreover, this indication triggers the address update to be read in either the regular or the bypass PIPO. A sum by one is used for that purpose at both regular and bypass PIPO addresses update.

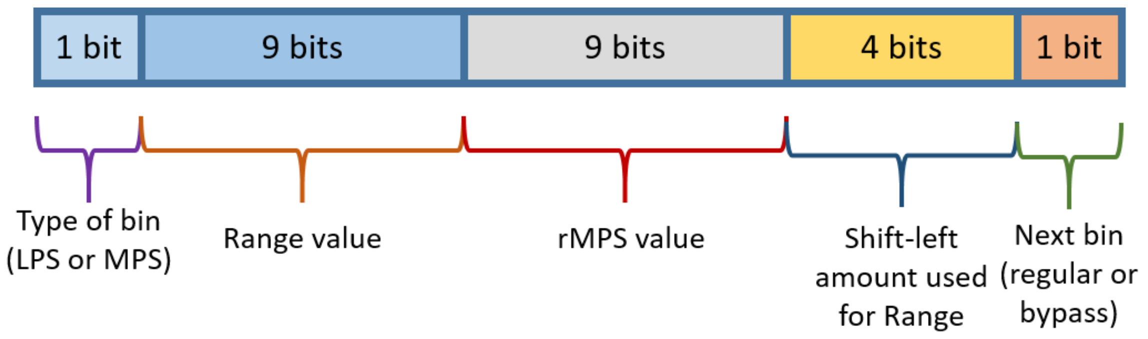
(v) ***Range* pre-calculation for MBBS**: as seen in the proposal for MBBS in (5.1), *Range* multiplication is required. The fifth stage uses the *Range* value stored at the regular PIPO and the amount of bypass bin indicated at the bypass PIPO to do this calculation at this stage. This operation occurs separately to avoid an increase in the path of the *Low* update stage (i.e., the next stage) and therefore the transfer of the original critical path from the *Range* update stage. In case the current bin is a regular, this stage is not required.

(vi) ***Low* update**: after the merge, the *Low* update takes place at the sixth pipeline stage. Considering the use of MBBS, up to two consecutive bypass bins may be processed at the same time for each core used at this stage, or a single regular bin.

(vii) **OB update**: the decision to split the OB update from the *Low* update stage is the same presented for the *Range* pre-calculation for MBBS. The OB update requires a sum for both regular and MBBS update of the referred variable, as seen in Figure 3.3 and (5.7). Therefore, an extra adder would be appended to the *Low* update logic if both *Low* and OB were processed together at the same pipeline stage, potentially causing the *Low* update stage to be the new critical path of the architecture.

(viii) **Bitstream generation**: the eighth and final pipeline stage generated the bitstream either for a regular or up to two bypass bins. The *Low* and OB updated variables control the bitstream generation and whether it is generated or not generated for a given(s) bin(s), following Figure 3.6 and Figure 5.8.

Figure 6.1 – Regular PIPO address organization

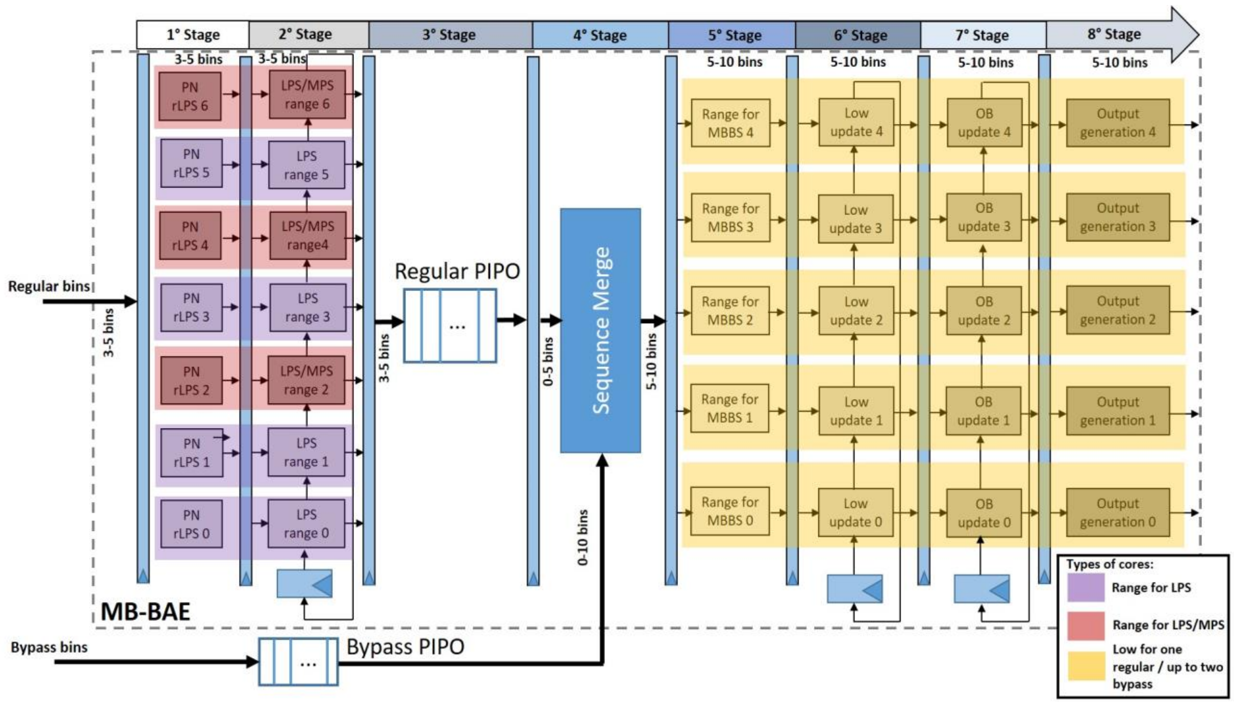


Source: the author.

The whole MB-BAE structure is presented in Figure 6.2, where along with the proposed pipeline structure; the cores for each stage are presented. The first two pipeline stages follow the proposal in (ZHOU, 2015), where there is a seven-core structure, which operate only for regular bins, as already presented in Chapter 3, using PN rLPS and LH rLPS techniques. The stages three and four are the same as presented above.

Stages five, six, seven and eight are composed of a five-cores structure, appended in a combinational fashion one-another. The main difference compared to (ZHOU, 2015) is the integration of the MBBS technique, which now increases the maximum number of bin per cycle that may be processed at a given cycle to up to ten bins (in case ten bypass bins occur in a row). Nevertheless, any combination of regular and bypass bins occurring in a cycle, where the bottom limit is five regular/bypass bins, and up to ten bypass bins (e.g., three regular bins followed by four bypass bins). As proposed in (ZHOU, 2015), only when at least five bins of any given type are available to be processed in the same cycle that the referred pipeline stages are activated. Otherwise, they stall until the minimum amount demanded are accomplished by the previous pipeline stages.

Figure 6.2 – MB-BAE architecture

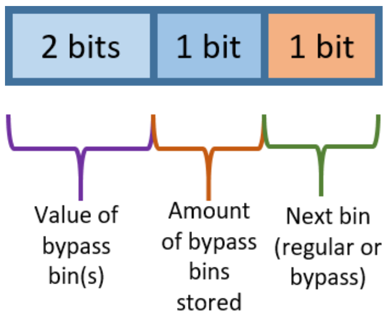


Source: (RAMOS, 2018a).

One may remark that, at the design of MB-BAE, the *Low* and OB update are split into two different stages (along with the *Range* update for MBBS). The *Low* update stage requires at least five cascaded adders, in case any combination of multiples LPS and bypass bins with value ‘1’ occurs, as referred in (3.3) and (3.4). The updated *Low* variable is required for the OB update, as referred in (5.7), where there will be a situation that the value ‘2’ is summed to the current OB variable. Moreover, the *Range* for MBBS also needs an adder, as seen in Figure 5.6. Therefore, in case all the three mentioned logics were allocated into the same pipeline stage, a new potential critical path would appear at the *Low* update stage, instead of the original critical path at *Range* update stage in (ZHOU, 2015). This explanation is the main reason for the new 8-stage pipeline structure proposed.

The bypass PIPO (i.e., the PIPO that store the bypass bins before the merging process) is necessary for the whole BAE structure when integrated to the full CABAC design, but is outside the MB-BAE design. Each address will contain the following information: two slots for up to two bypass bins values, a flag to indicate whether one or two bypass bins are stored at that address, and the flag to indicate if the next bin shall be retrieved from the regular or bypass PIPO. Figure 6.3 depicts that organization.

Figure 6.3 – Bypass PIPO address organization



Source: the author.

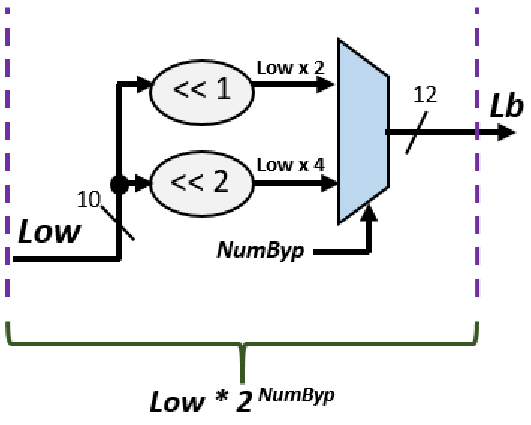
6.2 Efficient High-Throughput BAE Architecture with MBBS

A second proposal is to use MBBS in an efficient design. Efficient here has the meaning of a more straightforward design, or at least with less impact in area/power and time-to-market implementation effort when using as primitive parameter the baseline BAE design of (FEI, 2011), which are named here again as BA-BAE.

An alternative, but potentially efficient BAE architecture, is to gather MBBS along PN rLPS, and LH rLPS, due to the smoother modification these techniques require to modify the baseline BAE. By doing so, the intention is to use MBBS as an alternative to BPBS, and still achieve the same or a comparable throughput of bins/cycle, with negligible impact into the maximum frequency of the BAE proposed architecture. As one may remember, of BPBS implies in a more complex design, due to the insertion of extra structures (i.e., PIPOs), and the requirement to control these structures during the flow splitting and flow merge. Furthermore, the PIPOS will have an additional impact in the area (potentially in power dissipation) of the design. Moreover, as explained during MB-BAE inception, the usage of both BPBS and MBBS impacts in more pipeline stages (i.e., more logic) to avoid degradation into the maximum frequency of the design, along with bigger memories (i.e., PIPOs) to store the intermediary values. This situation may point that, in case a reduced-constraint driven BAE circuitry is desired, that BPBS and MBBS are not suitable to be used together, only if ultra-high-throughput is the variable of interest. Therefore, if instead, one utilizes an alternative technique, which had less impact on the baseline design and still achieves comparable throughput, this scenario would be advantageous in terms of area, power, and a time-to-market design, for instance.

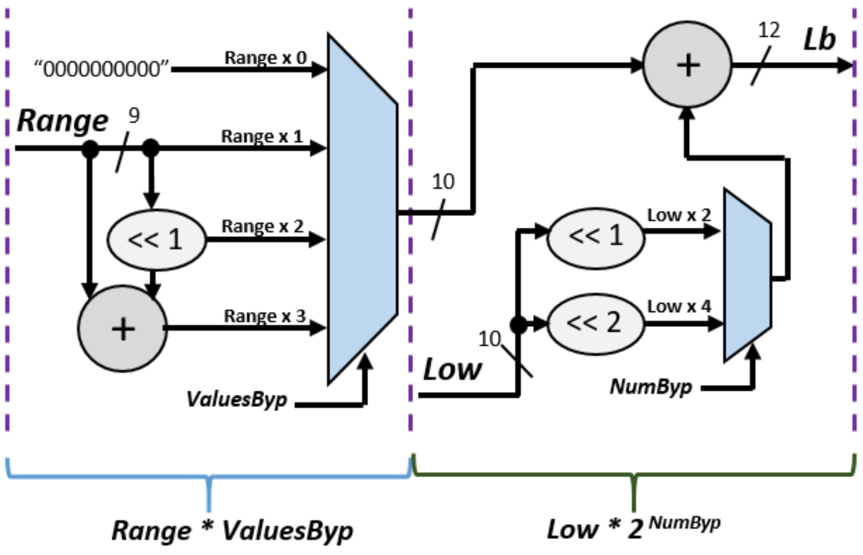
The first fact to notice is that bypass bins with value ‘0’ do not demand the *Range* multiplication for MBBS, as one may see at (5.1). Therefore, for two bypass bins in a row, being both with value ‘0’, the logic presented in Figure 6.4 is enough for the *Low* update, and which is named here Reduced Multiple Bypass Bins (RMBB) logic, in contrast to the Complete Multiple Bypass Bins (CMBB) logic, required in case any value combination of two bypass bins occurs. Figure 6.5 reproduces the CMBB logic, which is the original structure used for the MBBS presented at Chapter 5. The main conclusion one may derive from RMBB is the fact it has a lesser impact on increasing *Low* update since no adder is inserted. Thus, for a multiple-core logic, the use of RMBB is an option, as will be explained below.

Figure 6.4 – RMBB structure for MBBS for up to two bypass bins with value ‘0’



Source: the author.

Figure 6.5 – CMBB structure for MBBS for up to two bypass bins with any values



Source: (RAMOS, 2018a).

AltBAE composition is a five-stage pipeline structure, depicted in Figure 6.6., and the macro-structure is presented in Figure 6.7:

(i) **Pre-selection and pre-renormalization of *rLPS***: the first pipeline stage is used for the same purpose as the designs of MB-BAE and (ZHOU, 2015). The same seven core-structure is used.

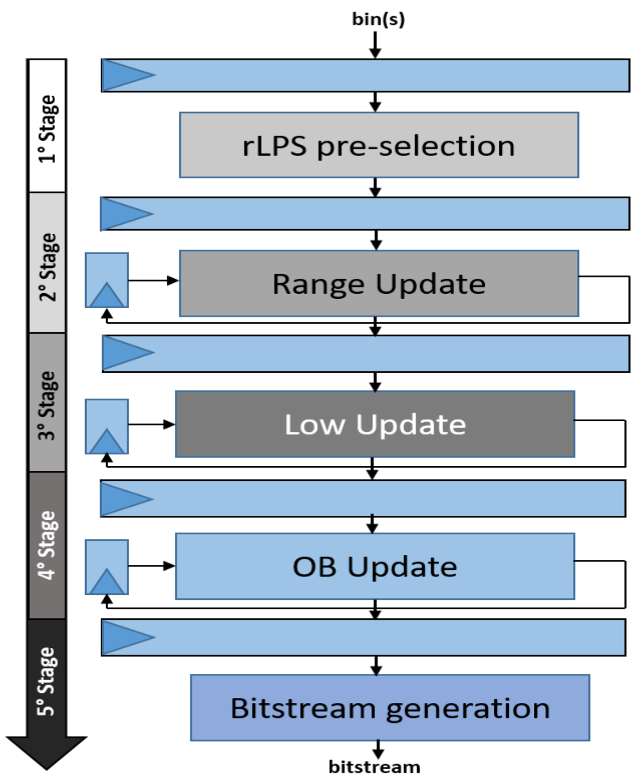
(ii) ***Range* update**: the second stage has the same purpose of the prior designs, but now has some different approaches, since the flows of regular and bypass bins are kept together, as will be further explained.

(iii) ***Low* update**: the third stage updates and renormalizes the *Low* variable. At this stage, either CMBB or RMBB are inserted.

(iv) **OB update**: the fourth stage updates the OB variable. The reason to split this logic apart the *Low* update stage is the same presented for the MB-BAE: to avoid an increase into the critical path of the architecture. For MBBS, the logic proposed in (5.7) is used.

(v) **Bitstream generation**: the fifth and final stage is used for the bitstream generation, the same fashion already presented for MB-BAE. Again, for MBBS, the logic proposed in Figure 5.8 is used.

Figure 6.6 – 5-stage pipeline organization of AltBAE

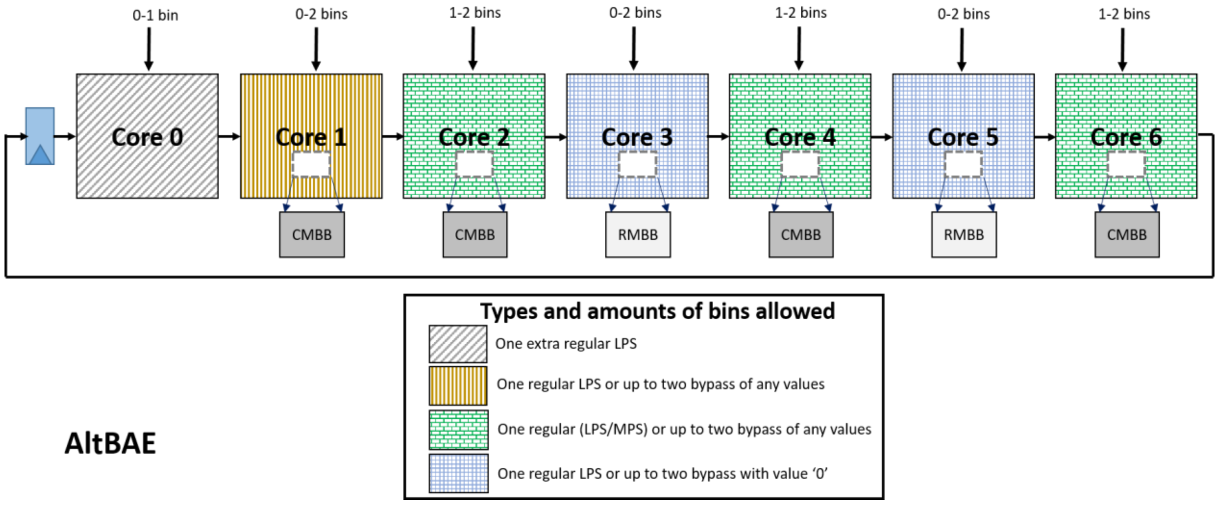


Source: the author.

The behavior of AltBAE is as follows: for core 0, neither CMBB nor RMBB are inserted (i.e., this core can process a single LPS in the condition presented in Chapter 3 for the LH rLPS). Core 1 can process a regular LPS bin, or up to two bypass bins of any values (i.e., CMBB is used within this core). Cores 2, 4, and 6 can process a regular bin of any type, or up to two bypass bins of any value (i.e., CMBB is used for those cores). Finally, core 3 and 5 can process a regular LPS bin, or up to two bypass bins with value ‘0’ (i.e., RMBB is used within those cores). One important remark is the fact that, at a given cycle, only one LPS bin can be processed by either (and exclusively) cores 1, or 3, or 5. For instance, if core 1 is able to process an LPS, core 3 and 5 cannot process a LPS at this cycle, and the same reasoning is used for the two others when they are able to process an LPS. Moreover, if core 1 is also used for bypass bins, cores 3 and 5 cannot be used for LPS bins. This behavior avoids the use of more than five serial adders at *Low* update stage and hence the transfer of the critical path to this stage.

Nevertheless, since the impact of RMBB at core 3 and 5 is much lesser than an adder, even if an LPS bin (or any bypass bins) are processed at core 1, in case one (or two) bypass bins, both with values ‘0’, are able to be processed at cores 3 and 5, they will be sent to these cores. This option has no potential prejudice to the maximum frequency of the architecture, and indeed increases the bins per cycle throughput, by providing the Core 3 and 5 to have an alternative to process bins in case Core 1 already has bins assigned to be processed at it, differently from the original proposal of LH rLPS.

Figure 6.7 – AltBAE architecture organization


0-1 bin
0-2 bins
1-2 bins
0-2 bins
1-2 bins
0-2 bins
1-2 bins
Core 0
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
CMBB
CMBB
RMBB
CMBB
RMBB
CMBB
Types and amounts of bins allowed
One extra regular LPS
One regular LPS or up to two bypass of any values
One regular (LPS/MPS) or up to two bypass of any values
One regular LPS or up to two bypass with value ‘0’
AltBAE

Source: the author.

6.3 Simulation, Synthesis Results, and Comparisons

The throughput of both MB-BAE and AltBAE were measured using the same methodology presented at Chapter 5: recommended test video sequences (BOSSEN, 2013) are applied to the design to accomplish the number of bins per cycle achieved by the designs. Five video sequences were used (the same used for the previous works presented in this Thesis), and using for each of them all possible combinations of the top and bottom QP (Quantization Parameter) recommended values (i.e., 22 and 37) and two recommended configurations (Low-Delay – LD and Random Access – RA). The improvement of the MBBS technique depends on the percentage of bypass bins occurring in a given sequence, and the amount of those bins occurring consecutively. The results are presented in Table 6.1 for MB-BAE, and in Table 6.2 for AltBAE. Along with the measured throughput, Table 6.1 also presents the throughput achieved by the work of (ZHOU, 2015) using BPBS (named here BS-BAE) and the increased percentage of bins/cycle of MB-BAE against BS-BAE; and Table 6.2 presents the work of (ZHOU, 2015) without the use of BPBS (named here PrelBAE). Table 6.2 also provides the percentage increase against the baseline BA-BAE of (FEI, 2011), which is constant (i.e., 4-bins/cycle) All designs make use of PN rLPS and LH rLPS, except BA-BAE.

Table 6.1 – Throughput comparisons of MB-BAE and BS-BAE

Videos	**Config.**		**Bins/cycle**		**Throughput Increase**
			BS-BAE (ZHOU, 2015)	**MB-BAE**	
BasketballDrive (1920x1080)	LD	22	4.37	**4.71**	**7.78 %**
		37	4.33	**4.66**	**7.62 %**
	RA	22	4.42	**4.85**	**9.73 %**
		37	4.37	**4.78**	**9.38 %**
BQTerrace (1920x1080)	LD	22	4.33	**4.33**	**0.00 %**
		37	4.33	**4.53**	**4.62 %**
	RA	22	4.43	**4.76**	**7.45 %**
		37	4.36	**4.69**	**7.57 %**
Kimono (1920x1080)	LD	22	4.43	**5.63**	**27.09 %**
		37	4.36	**4.76**	**9.17 %**
	RA	22	4.43	**5.83**	**31.60 %**
		37	4.36	**4.99**	**14.45 %**
PeopleOnStreet (2560x1600)	LD	22	4.38	**5.23**	**19.41 %**
		37	4.39	**5.14**	**17.08 %**
	RA	22	4.43	**5.46**	**23.25 %**
		37	4.44	**5.31**	**19.59 %**
Traffic (2560x1600)	LD	22	4.40	**4.79**	**8.86 %**
		37	4.35	**4.66**	**7.13 %**
	RA	22	4.36	**4.98**	**14.22 %**
		37	4.36	**4.76**	**9.17 %**
Average			4.37	**4.94**	**12.76 %**

Source: (RAMOS, 2018a).

Table 6.2 – Throughput comparisons of AltBAE, PrelBAE, and BA-BAE

Videos	Configuration		Bins/cycle		Throughput Increase	
			PrelBAE (ZHOU, 2015)	**AltBAE**	**Against PrelBAE**	**Against BA-BAE**
BasketballDrive (1920x1080)	LD	22	3.88	**4.18**	**7.73 %**	**4.76%**
		37	3.69	**4.14**	**12.20 %**	**3.76%**
	RA	22	3.86	**4.25**	**10.10 %**	**6.52 %**
		37	3.64	**4.19**	**15.11 %**	**5.01%**
BQTerrace (1920x1080)	LD	22	3.79	**3.95**	**4.22 %**	**-1.00%**
		37	3.77	**4.07**	**7.96 %**	**2.01%**
	RA	22	3.79	**4.23**	**11.61 %**	**6.02%**
		37	3.77	**4.16**	**10.34 %**	**4.26%**
Kimono (1920x1080)	LD	22	3.81	**4.66**	**22.31 %**	**16.79%**
		37	3.78	**4.20**	**11.11 %**	**5.26%**
	RA	22	3.80	**4.74**	**24.74 %**	**18.80%**
		37	3.79	**4.32**	**13.98 %**	**8.27%**
PeopleOnStreet (2560x1600)	LD	22	3.83	**4.46**	**16.45 %**	**11.78%**
		37	3.79	**4.39**	**15.83 %**	**10.03%**
	RA	22	3.89	**4.55**	**16.97 %**	**14.04%**
		37	3.83	**4.47**	**16.71 %**	**12.03%**
Traffic (2560x1600)	LD	22	3.83	**4.27**	**11.49 %**	**7.02%**
		37	3.82	**4.15**	**8.64 %**	**4.01%**
	RA	22	3.82	**4.37**	**14.40 %**	**9.52%**
		37	3.79	**4.20**	**10.82 %**	**5.26%**
Average			3.80	**4.30**	**13.14 %**	**7.71%**

Source: the author.

As one may notice, MB-BAE achieves the highest bin per cycle throughput along all works (an average of 4.94 bins/cycle), which perform around 12.76 % more bins per cycle when compared to BS-BAE (i.e., which had the highest bins per cycle throughput so far). AltBAE achieves around 4.30 bins/cycle, which is circa 13.14% and 7.71% higher than the bins/cycle of PrelBAE and BA-BAE, respectively. These two results, once more, corroborate the theoretical gain calculated prior to the MBBS inception, as stated in (5.2).

One remark is the fact that the bins/cycle throughput of both MB-BAE and AltBAE is higher for the sequences with QP 22 (i.e., higher quality videos). This result can be explained due to the increase in the transform residues SEs, which compose the majority of the SEs produced by any given video sequence (STANKOWSKI, 2014). These residual SEs tend to be zero in case a higher QP value is used and therefore are ignored by the entropy coding. The final value of the residues after decreasing them by a baseline value, which is the SE *coeff_abs_level_remaining*, which generates only bypass bins (ITU-T, 2013). The lower QP value makes these elements to remain with a non-zero value, tending to increase the amount of the bypass bins.

Moreover, some sequences achieved a considerable higher throughput when compared to the others (i.e., Kimono and PeopleOnStreet). The reason is the higher amount of bypass bins compared to regular bins, which is stated in (ZHOU, 2015) and was reproduced in Table 4.1. Nevertheless, for almost all sequences, QP values, and configurations, MB-BAE has the best bins/cycle throughput, whereas AltBAE has results closely related to BS-BAE.

A synthesis was also performed for both MB-BAE and AltBAE, using ST 65 nm CMOS PDK and the Cadence Encounter RTL Compiler Suite. Table 6.3 depicts the synthesis results and the comparison with the relevant related works. MB-BAE has the highest bins per second throughput among all BAE designs, showing that the usage of MBBS along the prior-art techniques for high-throughput BAE design is worthy and valid. The critical path for MB-BAE is the second pipeline stage (i.e., *Range* update stage), which is the same critical path of the prior-art related works. Therefore, since the *Range* update stage is the same of the architecture in (ZHOU, 2015), we could infer that the maximum frequency achieved by MB-BAE is the same as BS-BAE, in case a 90 nm synthesis was done. Furthermore, a normalized bins/s throughput for 90 nm is inferred, which would be around 2,075 Mbins/s, around 13% above the value accomplished by BS-BAE. The major drawback of MB-BAE is the higher impact in area due to the PIPOs utilization, since it utilizes BPBS along with MBBS, which requires new pipeline stages and organizations on the cited PIPOS, which are mandatory for the splitting of the regular and bypass streams for a whole CABAC design. As will be further presented, it also influences in higher power consumption.

AltBAE critical path is also the *Range* update stage, which corroborates the results of MB-BAE in which the MBBS techniques does not increase the original critical path of the baseline architecture, as long as the division into more pipeline stages proposed is followed. Therefore, the same reasoning used for the comparison of MB-BAE and BS-BAE is applied: the expected maximum frequency for AltBAE is the same or very close to the value accomplished by PrelBAE if a 90 nm technology were used, since no significant logic is inserted into the *Range* update stage due to the use of the MBBS. The normalized bins/s throughput of AltBAE is 1,767 Mbins/s, which is only 3.8% below the values achieved by BS-BAE, at the major advantage that AltBAE does not utilize the BPBS technique, which, as already discussed, infers a higher area and more complex project. Moreover, in case one consider only low QP sequences (i.e., sequences with QP 22), the bins/cycle and bins/second of AltBAE for 90 nm would be 4.36 and 1,792 Mbins/s, only 2.4% below PrelBAE throughput. This result presents AltBAE as a valid alternative for a high-throughput and efficient real-time processing, with potential faster time-to-market CABAC design. Furthermore, the expectation is that AltBAE is also more efficient in terms of power consumption, exactly due the avoidance of BPBS employment.

Most of the detailed designs do not present area values for the BAE block only. Thus, a fair comparison is not possible for this variable. Nevertheless, as one may see, both MB-BAE and AltBAE are considerably beneath the gate-count results for the whole CABAC design presented by the prior-art designs.

Table 6.3 – Comparisons of MB-BAE and AltBAE against prior-arts

Design		**PrelBAE (ZHOU, 2015)**	**BS-BAE (ZHOU, 2015)**	**MB-BAE**		**AltBAE**	
#bins/cycle (avg)		3.8	4.37	**4.94**		**4.30**	
Maximum Frequency (MHZ)		411	420	**537**	**420”**	**527**	**411”**
#Mbins/s		1116	1836	**2653**	**2075”**	**2266**	**1767”**
CMOS Technology (nm)		90	90	**65**	**90”**	**65**	**90”**
Gate Count (K)		n. a.	64.1*	**33**		**20.2**	
Features	**BPBS**	no	yes	**yes**		**no**	
	MBBS	no	no	**yes**		**yes**	

* Whole CABAC results without memory
” Normalized values for 90 nm

Source: the author.

7 TOWARDS HIGH-THROUGHPUT AND ENERGY-EFFICIENCY TRADE-OFF BAE DESIGN

This chapter presents a proposal for BAE architecture on the HEVC context, following the two architectural BAEs proposals described in the last chapter. The two versions are compared in terms of power dissipation, and the AltBAE is selected as the best choice considering the trade-off between high-throughput and energy-efficiency, at first glance. Furthermore, alternative lower-performance AltBAE versions are analyzed using real video sequences stimuli in order to verify throughput and power dissipation, and therefore point out suitable choice considering the performance-energy balance. Additionally, the power saving techniques used in the first proposal of low-power entropy encoding design application occurs into the different AltBAE versions, achieving even smaller power dissipation. Finally, a variable-throughput BAE design proposal occurs, named thereafter ET-BAE which, based on the previous analysis, allows different configuration one may follow to achieve the best trade-off between the performance needed for a given video constraint and the power dissipation

7.1 Preliminary Power-Analysis Comparison between High-Throughput BAE Architectures with MBBS

The two previously presented BAE circuits, namely MB-BAE and AltBAE, are the starting point for this analysis. One may conclude that MB-BAE is, up to this date, the highest-performance BAE design available in the literature, whereas AltBAE reaches comparable throughput to the previous highest-throughput BAE proposal (ZHOU, 2015), with the advantage of not using the memory-consuming and design structural changing BPBS technique.

As a preliminary estimative, MB-BAE seems to be less energy-efficient when compared to AltBAE precisely by the requirement of the PIPO structure and more pipeline stages to avoid frequency degradation (i.e., more registers are necessary). An initial power analysis using the default and a defined switching activity on both designs indicates that expected behavior, and can be verified in Table 7.1 (the first parameter refers to the probability of a net being high, whereas the second is the toggle count per toggle rate unit of a net). One may see that, when comparing the bins per second of AltBAE against MB-BAE solely (since the maximum frequency difference is almost negligible – refer to Table 6.3), the degradation in throughput is smaller (i.e., 13.20 % less bins per cycle) than the power dissipation on both designs (i.e., 21.19% to 27.05% less power dissipation of AltBAE compared to MB-BAE). It is important to notice that these throughput values were derived by using, additionally to the high-quality video sequences presented on the last chapter, other smaller resolution recommended video sequences (BOSSEN, 2013), and which detailed values are found in Appendix D of this Thesis.

Table 7.1 – Throughput and estimated power dissipation comparison between AltBAE and MB-BAE

			Switching activity parameter			
	Throughput		**Default (0.5 , 0.02)**		**Defined (0.5 , 0.2)**	
Design	bins/cycle	Degradation	Power Dissipation	Savings	Power Dissipation	Savings
MB-BAE	5	-	4.620 mW	-	39.947 mW	-
AltBAE	4.34	13.2 %	3.641 mW	21.19 %	29.141 mW	27.05%

Source: the author.

Regarding the upper bound constraints for the levels defined by the HEVC standard (ITU-T, 2013), as can be seen in Table 7.2, both MB-BAE and AltBAE surpass by far the real-time necessary bins per second requirement for the highest Level (i.e., 6.2) at High tier – refer to Table 6.3. The bit-to-bins conversion is roughly, a multiplication by 4/3 of the required Mbits/s (CHEN, 2015). The constraint of Table 7.2 does not differentiate the frames per second a sequence has, or the QP used, but only the maximum number of bins per second each sequence categorized will produce, at most. A design has to process in one second or less the given requirement to be able of real-time capabilities on the related Level. Therefore, the indication is towards the selection of AltBAE as the starting design for the purpose already stated in this chapter.

Table 7.2 – Throughput requirement for different levels of the HEVC standard

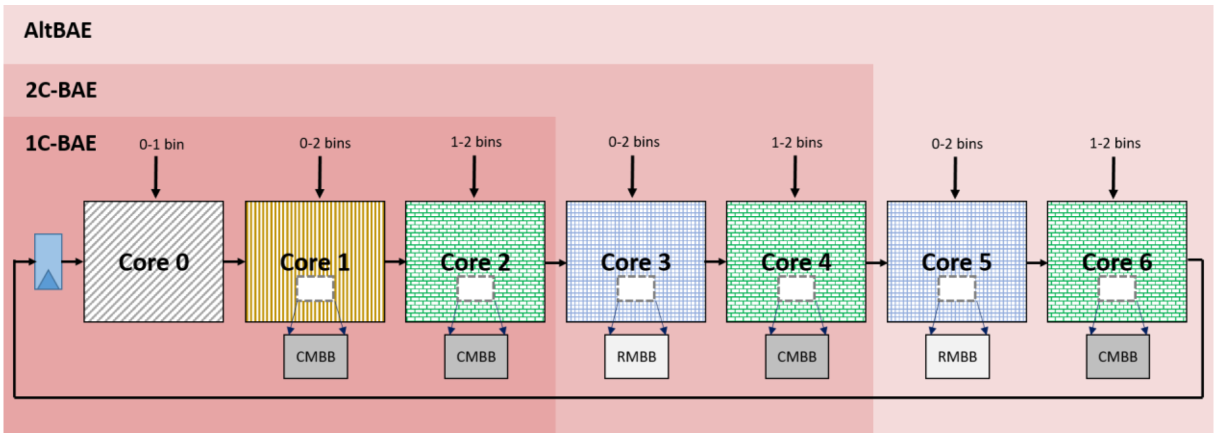
		Level Requirements												
Tier		1	2	2.1	3	3.1	4	4.1	5	5.1	5.2	6	6.1	6.2
Main	Mbit/s	0.128	1.5	3	6	10	12	20	25	40	60	60	120	240
	Mbins/s	0.17	2	4	8	13	16	27	33	53	80	80	160	320
High	Mbit/s	-	-	-	-	-	30	50	100	160	240	240	480	800
	Mbins/s	-	-	-	-	-	40	67	133	213	320	320	640	1,067

Source: the author, adapted from (ITU-T, 2013)

7.2 AltBAE Different Configurations Proposal: Throughput and Power-Consumption Analysis

A further look at Figure 6.7, which depicts the AltBAE full proposal, and it is possible to notice the granularity formed by the usage of the similar Cores in the cited design. In other words, smaller AltBAE versions derivation are possible from the full proposition, as can be seen in Figure 7.1 below: an AltBAE version comprised of cores 0 up to 4, named 2C-BAE and an even smaller version composed of core 0 up to 2, named 1C-BAE. The original AltBAE keeps its original name here.

Figure 7.1 – AltBAE granularity, and the internal 2C-BAE and 1C-BAE.



Source: the author.

The first analysis is how much throughput degradation the 2C-BAE and 1C-BAE architectures would suffer compared to the AltBAE. The same recommended video sequences used for the assessment shown in Table 7.1 were used (and which are entirely depicted in Appendix D of this Thesis). As a summary, 2C-BAE accomplishes around 3-bins per cycle, whereas 1C-BAE 1.6-bins per cycle of throughput. Along with the simulation, synthesis results are provided by using the CMOS ST 65 nm PDK, which appear in Table 7.3. Firstly, the 2C-BAE has a slight improvement in frequency, which is expected due to the remove of one adder from the critical path but is negligible compared to the bins per cycle degradation (i.e., around 6% of frequency improvement against around 30% of bins per cycle throughput degradation). 1C-BAE has the same frequency of 2C-BAE, due to the critical path still having the same number of adders of the 2C-BAE (i.e., the critical path has changed from *Range* to *Low* stage, in this case). The area results show the decrease according to the removal of cores in the smaller architectures, as expected.

The second analysis was to use real stimuli (i.e., recommended video sequences) to obtain accurate power values for each of the AltBAE versions. For that purpose, one sequence of each class is used (BOSSEN, 2013): Traffic, Kimono, PartyScene, BlowingBubbles, Johnny, and China Speed, respectively from classes A, B, C, D, E, and F. For each sequence, both Low-Delay (LD) and Random-Access (RA), and QPs 22 and 37, were applied. The simulation clock was 250 MHz, which is, considering the original AltBAE architecture, the frequency in which the already cited 1-Gbin/s constraint is achievable. The complete results (power values and power savings compared to the preceding version, i.e., 2C-BAE to AltBAE and 1C-BAE to 2C-BAE) appear in Table 7.4, 7.5, 7.6, 7.7, 7.8, and 7.9, respectively for each of the cited sequences and for each version of the AltBAE proposal.

As average power dissipation values, AltBAE consumes 28.086 mW, 2C-BAE consumes 12.981 mW, whereas 1C-BAE consumes 7.154 mW. Table 7.10 summarizes these values, along with the average percentage saving in power from one version to the next one. Additionally, to help guide the analysis towards the best trade-off scenario, the throughput percentage degradation also appears in Table 7.10, based on the simulation values previously presented and reproduced in the same Table. Moreover, Figure 7.2 plots the power dissipation (axis Y) against the throughput increase (axis X) of each version, to illustrate in more details the advantages/drawbacks of each design for the two variables. It is important to notice that we are considering a scenario where all the architectures are running on 250 MHz (i.e., the same clock frequency). The NA symbol indicates that there is no throughput degradation nor power saving (which will occur from a bigger version when compared to the smaller one).

Table 7.3 – Simulation and synthesis results for AltBAE, 2C-BAE, and 1C-BAE

Design	**Maximum Frequency (MHz)**	**Gate Count (K)**	**Throughput**	
			Bins/cycle	**MBins/s**
AltBAE	527	20.20	4.34	2,287
2C-BAE	559	14.65	3	1,677
1C-BAE	559	9.14	1.6	894

Source: the author.

Table 7.4 – Power dissipation results for the Traffic sequence on AltBAE versions

	Traffic							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	28.538	29.011	29.121	29.073	-	-	-	-
2C-BAE	13.613	13.505	13.719	13.438	53.31	54.67	53.97	54.95
1C-BAE	7.360	7.132	7.438	7.217	48.80	48.63	49.12	45.41

Source: the author.

Table 7.5 – Power dissipation results for the Kimono sequence on AltBAE versions

	Kimono							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	25.634	26.591	25.852	28.125	-	-	-	-
2C-BAE	12.026	13.071	12.041	13.083	53.09	50.84	53.42	53.48
1C-BAE	6.454	7.028	6.433	6.959	46.33	46.23	46.57	46.81

Source: the author.

Table 7.6 – Power dissipation results for the PartyScene sequence on AltBAE versions

	PartyScene							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	27.628	26.905	27.761	29.349	-	-	-	-
2C-BAE	12.728	13.728	12.712	13.268	53.93	48.98	54.21	54.79
1C-BAE	6.518	6.979	6.616	7.390	48.79	49.16	47.95	44.30

Source: the author.

Table 7.7 – Power dissipation results for the BlowingBubbles sequence on AltBAE versions

	BlowingBubbles							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	29.013	28.203	29.048	29.665	-	-	-	-
2C-BAE	13.369	12.123	13.523	12.858	53.92	57.02	53.45	56.66
1C-BAE	7.201	7.417	7.389	7.425	46.14	38.82	45.36	42.25

Source: the author.

Table 7.8 – Power dissipation results for the Johnny sequence on AltBAE versions

	Johnny							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	28.880	26.698	27.304	27.995	-	-	-	-
2C-BAE	13.509	10.627	12.194	13.055	53.22	60.2	55.34	53.37
1C-BAE	7.227	6.621	7.196	7.029	46.50	37.70	40.99	46.16

Source: the author.

Table 7.9 – Power dissipation results for the ChinaSpeed sequence on AltBAE versions

	ChinaSpeed							
	Power Dissipation (mW)				**Power saving (%)**			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
AltBAE	28.551	27.745	27.943	29.444	-	-	-	-
2C-BAE	12.002	13.874	13.497	13.976	57.96	49.99	51.70	52.53
1C-BAE	7.489	7.773	7.558	7.845	37.60	43.97	44.00	43.87

Source: the author.

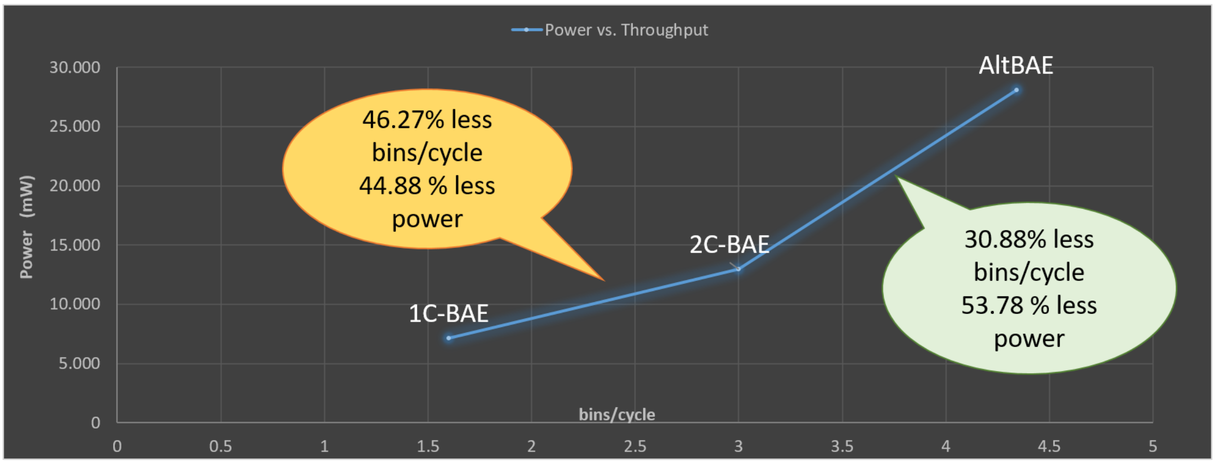
The first remarkable outcome of Table 7.10 and Figure 7.2 is that the degradation in throughput comparing AltBAE against 2C-BAE (30.88%) is smaller than the power savings comparing both versions (53.78 %). This output indicates that there seems to be room for energy efficiency in case it is possible to switch somehow between both versions, as will be further presented. In fact, the same reasoning can be made comparing AltBAE against 1C-BAE (i.e., 68% in performance degradation and 74.52% of power savings). However, the same could not be said when putting 2C-BAE against 1C-BAE (i.e., 46.27% in throughput prejudice and 44.88% of power savings). The AltBAE vs. 1C-BAE comparison values are derived from the values already presented in Table 7.4 up to Table 7.9, and are presented just for the sake of clarity of the text.

Table 7.10 – Comparison between power saving and throughput degradation among AltBAE versions

	General Performance		**Vs. AltBAE**		**Vs. 2C-BAE**	
Design	Throughput (bins/cycle)	Power Consumption avg. (mW)	Throughput Degradation (%)	Power Saving (%)	Throughput Degradation (%)	Power Saving (%)
AltBAE	4.34	28.086	-	-	NA	NA
2C-BAE	3	12.981	30.88	53.78	-	-
1C-BAE	1.6	7.154	68	74.52	46.27	44.88

Source: the author.

Figure 7.2 – Power-throughput curves comparison among AltBAE, 2C-BAE, and 1C-BAE



Source: the author.

For instance, Table 7.11 and 7.12 presents the estimated processing time each of the proposals would take to perform one second of video considering the bins per second Level requirements shown in Table 7.2 (considering again each design is running at 250 MHz), from Level 6.2 down to Level 4 on High and Main tier, respectively. One may remember that the constraints of Table 7.2 do not discriminate the frames per second nor the QP values used, but only the amount of bins per second target for the given Level. Any processing time below one second attests that the given design can accomplish real-time performance for the referred Level. For High tier, the constrains definition happens down to Level 4 as the bottom limit, whereas below Level 4 for the Main tier, the requirements become so low that the capabilities of the architectures are extremely overestimated and, therefore, the values are omitted. For the High tier scenario, it is clear that real-time 6.2 level is not supported by the 2C-BAE running at 250MHz, whereas the 1C-BAE does not support the Level 6.2 and 6.1 at the same clock frequency (highlighted in red on the respective Table). For the Main tier, all BAE solutions achieve the requirements for real-time processing (refer to Table 7.12).

By multiplying the values on both Table 7.11 and 7.12 by the average power values in Table 7.10, the estimated energy consumption (in mJ) for the one-second real-time amount of bins on each scenario is obtained for the High and Main tier, and whose values appear in table 7.13 and 7.14, respectively.

Table 7.11 –Estimated time values for standard HEVC levels High tier constraints

	Time (s) to process the related Mbins amount on Levels for High tier videos							
Design	**6.2**	**6.1**	**6**	**5.2**	**5.1**	**5**	**4.1**	**4**
AltBAE	0.98	0.59	0.29	0.29	0.20	0.12	0.06	0.04
2C- BAE	1.42	0.85	0.43	0.43	0.28	0.18	0.09	0.05
1C-BAE	2.67	1.60	0.8	0.8	0.53	0.33	0.17	0.1

Source: the author.

Table 7.12 – Estimated time values for standard HEVC levels Main tier constraints

	Time (s) to process the related Mbins amount on Levels for Main tier videos							
Design	**6.2**	**6.1**	**6**	**5.2**	**5.1**	**5**	**4.1**	**4**
AltBAE	0.295	0.147	0.074	0.074	0.049	0.030	0.025	0.015
2C-BAE	0.427	0.213	0.107	0.107	0.071	0.044	0.036	0.021
1C-BAE	0.800	0.400	0.200	0.200	0.133	0.083	0.068	0.040

Source: the author.

One may conclude, by observing the values of Table 7.13 and 7.14, that AltBAE has the highest energy consumption of all. For the proposed scenario, considering that all Levels have real-time support for every architecture (except level 6.2 High tier for 2C-BAE and level 6.2 and 6.1 for High tier for 1C-BAE), the smaller versions of AltBAE seems to be more suitable for an energy efficient demand. Furthermore, the 2C-BAE seems to have the best trade-off efficiency, since 1C-BAE, even if it dissipates less power, consumes more energy due to the inferior balance between power dissipation and throughput degradation (refer to Table 7.10 and the related discussion). In fact, for current video resolutions and capabilities situations, like the ones defined by the HEVC standard, as already stated in Table 7.2, the 2C-BAE solely is already sufficient to provide the real-time requirements with enough margin. Nevertheless, future requirements with more demanding processing capabilities would require a different approach, which will be presented next. At first, a further improvement in terms of power dissipation happens on all AltBAE versions.

Table 7.13 –Estimated energy values for standard HEVC levels High tier constraints

	Estimated energy (mJ) to the related Mbins amount on Levels for High tier videos							
Design	**6.2**	**6.1**	**6**	**5.2**	**5.1**	**5**	**4.1**	**4**
AltBAE	27.61	16.56	8.28	8.28	5.51	3.44	1.73	1.04
2C-BAE	18.47	11.08	5.54	5.54	3.69	2.30	1.16	0.69
1C-BAE	19.07	11.44	5.72	5.72	3.81	2.38	1.20	0.72

Source: the author.

Table 7.14 –Estimated energy values for standard HEVC levels Main tier constraints

	Estimated energy (mJ) to the related Mbins amount on Levels for Main tier videos							
Design	**6.2**	**6.1**	**6**	**5.2**	**5.1**	**5**	**4.1**	**4**
AltBAE	8.28	4.14	2.07	2.07	1.37	0.85	0.70	0.41
2C-BAE	5.54	2.77	1.38	1.38	0.92	0.57	0.47	0.28
1C-BAE	5.72	2.86	1.43	1.43	0.95	0.59	0.48	0.29

Source: the author.

7.3 Power Saving techniques within the Scope of the Energy-Efficient High-Throughput BAE

As already presented in Chapter 4 and 5, two power saving techniques were integrated into preliminary BAE versions, showing potential improvement in terms of power dissipation. Thus, the same approach is also applied to the AltBAE, 2C-BAE, and 1C-BAE, in order to measure any power savings when compared to the original values already presents. The same logics already depicted in Figures 4.6 and 5.9 are isolated through Operand Isolation, whereas the same pipeline registers have the clock turned off due to the Clock Gating technique. Synthesis results show the frequency and area results for the same technology PDK used throughout this Thesis (i.e., ST 65 nm). For the sake of clarity, the AltBAE versions received the name LP-AltBAE, LP-2C-BAE, and LP-1C-BAE. Table 7.15 shows a negligible frequency and area degradation compared to the original version (less than 2%).

Following the synthesis, the same power analysis was performed for the power saving versions, at the same clock frequency (i.e., 250 MHz), for the same video sequences, configuration, and QP values as already shown in Tables 7.4 to 7.9. The new power values occur in Tables 7.16, 7.17, 7.18, 7.19, 7.20, and 7.21, following the same video sequence order of the original versions. The difference in the new Tables is that the power savings are now considering the original analogous versions without the low-power techniques, instead of the immediately preceding AltBAE version (i.e., the power gains of LP-AltBAE against the original AltBAE, the power gains of LP-2C-BAE against the original 2C-BAE, and the power gains of LP-1C-BAE against the original 1C-BAE).

Table 7.15 – Synthesis results of the power-saving AltBAE versions

Design	Throughput (bins/cycle)	Maximum Frequency (MHz)	Gate Count (K)
LP-AltBAE	4.34	525	20.76
LP-2C-BAE	3	558	14.89
LP-1C-BAE	1.6	558	9.18

Source: the author.

Table 7.16 – Power dissipation results for the Traffic sequence on power-saving AltBAE versions

	Traffic							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	23.585	23.940	24.157	23.953	17.36	17.48	17.05	17.61
LP-2C-BAE	11.011	10.851	11.120	10.792	19.11	19.65	18.94	19.69
LP-1C-BAE	5.728	5.491	5.776	5.567	22.17	23.01	22.34	22.86

Source: the author.

Table 7.17 – Power dissipation results for the Kimono sequence on power-saving AltBAE versions

	Kimono							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	20.992	23.260	21.147	23.236	18.11	12.53	18.2	17.38
LP-2C-BAE	9.528	10.492	9.557	9.800	20.77	19.73	20.63	25.09
LP-1C-BAE	4.878	5.390	4.863	5.350	24.42	23.31	24.41	23.12

Source: the author.

Table 7.18 – Power dissipation results for the PartyScene sequence on power-saving AltBAE versions

	PartyScene							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	22.806	23.384	22.918	24.333	17.45	13.09	17.45	17.09
LP-2C-BAE	10.204	11.107	10.196	10.598	19.83	19.09	19.79	20.12
LP-1C-BAE	4.944	5.688	5.044	5.762	24.15	18.5	23.76	22.03

Source: the author.

Table 7.19 – Power dissipation results for the BlowingBubble sequence on power-saving AltBAE versions

	BlowingBubbles							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	24.016	24.140	24.122	25.261	17.22	14.41	16.96	14.85
LP-2C-BAE	10.769	10.319	10.937	10.786	19.45	14.88	19.12	16.11
LP-1C-BAE	5.543	5.765	5.705	5.466	23.02	22.27	22.79	26.38

Source: the author.

Table 7.20 – Power dissipation results for the Johnny sequence on power-saving AltBAE versions

	Johnny							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	23.875	21.911	23.977	23.062	17.33	17.93	12.19	17.62
LP-2C-BAE	10.880	8.313	10.552	10.530	19.46	21.77	13.46	19.34
LP-1C-BAE	5.579	5.175	5.577	4.662	22.8	21.84	22.5	33.67

Source: the author.

Table 7.21 – Power dissipation results for the ChinaSpeed sequence on power-saving AltBAE versions

	ChinaSpeed							
	Power Consumption (mW)				Power saving (%)			
Design	LD 22	LD 37	RA 22	RA 37	LD 22	LD 37	RA 22	RA 37
LP-AltBAE	23.578	24.426	23.606	24.337	17.42	11.96	15.52	17.34
LP-2C-BAE	10.330	11.228	10.956	11.305	13.93	19.07	18.83	19.11
LP-1C-BAE	5.838	6.116	5.939	6.166	22.05	21.32	21.42	21.4

Source: the author.

As a summary, the low-power techniques led to an average of 16.33% of power gains for the LP-AltBAE version, 19.06% for the LP-2C-BAE, and 23.11% for the LP-1C-BAE, with average values of 23.50 mW, 10.50 mW, and 5.5 mW, respectively, as presented in Table 7.22. Nevertheless, the LP-1C-BAE still consumes less power proportionally to the bins per cycle degradation compared to the LP-2C-BAE, even with the slight improvement in power dissipation. Therefore, the same reasoning as made in the last section can be applied, and in terms of energy dissipation, the LP-1C-BAE is still more inefficient when compared to the LP-2C-BAE. This last information is depicted in Table 7.23 and Figure 7.3.

As a conclusion, a balance between LP-AltBAE and LP-2C-BAE still seems to be the best suitable choice when one wants a more efficient balance between performance and throughput. This remark will drive the architectural proposition and methodology to be presented next when considering future high demanding real-time video processing scenarios for the entropy-encoding step, which would also require energy savings (e.g., battery-based devices). Moreover, current real-time video processing solutions, running at lower clock frequencies, could also receive the advantages of using the high-throughput and energy-efficient BAE circuitry, as will be further discussed.

Table 7.22 – Summary of the power-saving AltBAE versions and comparison with original designs

Design	Power Consumption of LP designs avg. (mW)	Power Consumption of original design avg. (mW)	Power Savings avg. (%)
LP-AltBAE	23.501	28.086	16.33
LP-2C-BAE	10.507	12.981	19.06
LP-1C-BAE	5.501	7.154	23.11

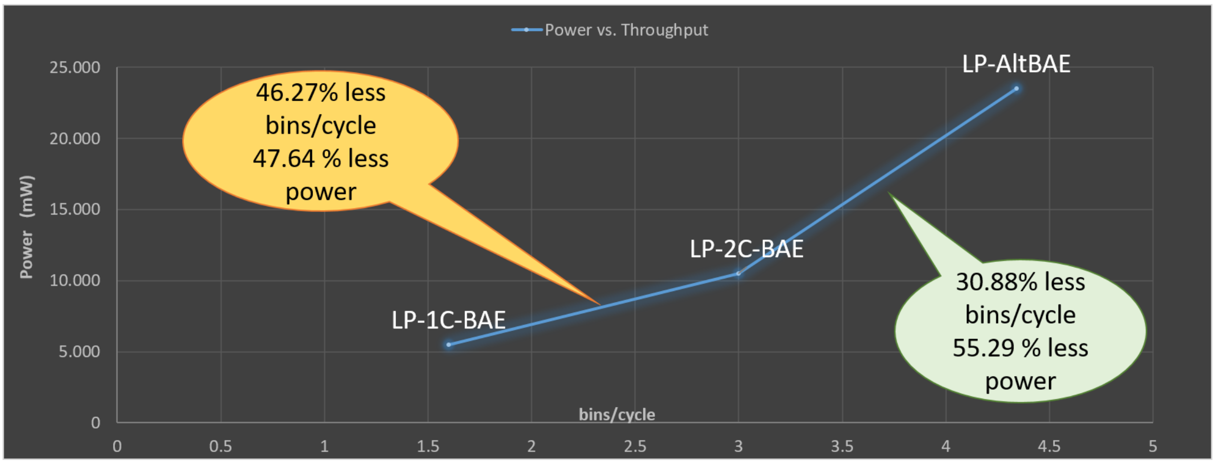
Source: the author.

Table 7.23 – Comparison of power saving AltBAE versions against each other

	Vs. LP-AltBAE		Vs. LP-2C-BAE	
Design	Throughput Degradation (%)	Power Saving (%)	Throughput Degradation (%)	Power Saving (%)
LP-AltBAE	-	-	NA	NA
LP-2C-BAE	30.88	55.29	-	-
LP-1C-BAE	68	76.59	46.27	47.64

Source: the author.

Figure 7.3 – Power-throughput curves comparison among LP-AltBAE, LP-2C-BAE, and LP-1C-BAE



Source: the author.

7.4 Performance and Energy Trade-off BAE Architecture

Based on the results aforementioned, a proposal of parallel usage of LP-AltBAE and LP-2C-BAE seems the best solution to balance high-throughput and energy-efficiency BAE design for future high-demanding real-time video processing (for both throughput and energy). Thus, possible scenarios of the parallel usage are presented, along with methodologies to achieve them. Furthermore, on-the-fly requirement changes (e.g., a video encoder solution running low on battery) are also discussed to support the usage of trade-off BAE proposal. Finally, a configurable architecture, gathering LP-AltBAE and LP-2C-BAE proposal is shown, named ET-BAE, which accomplishes the functionality and results desired in a single component.

7.4.1 Proposal of Usage Methodology

The maximum frequency of LP-AltBAE is 525 MHz for the ST 65 nm node, as stated before. Thus, Table 7.24 presents the bin per second throughput, and estimated power dissipation on this scenario, for both LP-AltBAE and LP-2C-BAE. The throughput of bins/s is simply the multiplication of the bins per cycle times the clock frequency. The power values are estimated based on the linear increase factor of the frequency when compared to the average values derived from the netlist simulation presented (i.e., a multiplication factor of 2.104), as presented in (2.6) and (2.7), for both switching and short-circuit consumption.

On Table 7.25, one may observe potential future real-time bins per second constraints. For instance, one could imagine the increase in resolutions (e.g., 16K pixels), the increase in frames per second (e.g., above 300), the usage of high-fidelity sampling (e.g., 4:4:4) and increase in bit width for each component in the used color space (e.g., 16 or more bits). This hypothetical but probable scenario may be real in the near future, such as that, at the moment of this Thesis inception, the upcoming Versatile Video Coding (VVC) (BROSS, 2018) standard is under development as the successor of the HEVC, to face a new era of video processing capabilities.

Table 7.25 also presents the time required for both LP-AltBAE and LP-2C-BAE to process in each hypothetical scenario, along with the estimated energy dissipated, by taking into account the average power results presented in Table 7.24 for 525 MHz of clock frequency. The first conclusion is that LP-2C-BAE does not accomplish the real-time constraint for any of the hypothetical future cases (values in red on Table 7.25). Nevertheless, its energy efficiency is still considerably above LP-AltBAE. Below the bottom requirement presented in Table 7.25 (e.g., 1,600 Mbins/s), the LP-2C-BAE is enough to achieve the real-time requirement and still consume less energy. Based on the results depicted, a potential proposition is to have both architectures working in parallel, but each of them running on a certain amount of time during the video processing.

Table 7.24 – Estimated throughput and power consumption at 525 MHz

	Estimated @ 525 MHz		
Design	Bins/cycle	Mbins/s	Power (mW)
LP-AltBAE	4.34	2278.5	49.45
LP-2C-BAE	3	1575	22.11

Source: the author.

Table 7.26 and the related Figure 7.4 will be used to illustrate that proposal, with the weighted throughput and energy dissipation based on the portion of time each architecture is employed. Furthermore, each architecture is used by a certain percentage of the processing time in seven proposed different scenarios, in which letters are used for the sake of clarity on the mentioned Table. The last column shows the energy saving, considering the proposed configuration against the sole usage of LP-AltBAE on the maximum reachable scenario for each configuration (e.g., scenario 'a' can accomplish the 2,208 Mbins/s throughput, whereas scenario 'b' the 2,102 Mbins/s performance, scenario 'c' the 2,023 Mbins/s requirement, etc).

a. 90% of LP-AltBAE and 10% of LP-2C-BAE;
b. 75% of LP-AltBAE and 25% of LP-2C-BAE;
c. 66% of LP-AltBAE and 33% of LP-2C-BAE;
d. 50% of LP-AltBAE and 50% of LP-2C-BAE;
e. 33% of LP-AltBAE and 66% of LP-2C-BAE;
f. 25% of LP-AltBAE and 75% of LP-2C-BAE;
g. 10% of LP-AltBAE and 90% of LP-2C-BAE.

Table 7.25 – Estimated time and energy dissipation on hypothetical future real-time scenarios

		Future real-time scenarios Mbins/s constraints						
Variable	**Design**	**2,200**	**2,100**	**2,000**	**1,900**	**1,800**	**1,700**	**1,600**
Time (s)	LP-AltBAE	0.96	0.92	0.87	0.83	0.78	0.74	0.70
	LP-2C-BAE	1.39	1.33	1.26	1.20	1.14	1.07	1.01
Energy (mJ)	LP-AltBAE	47.74	45.57	43.40	41.23	39.06	36.89	34.72
	LP-2C-BAE	30.88	29.47	28.07	26.67	25.26	23.86	22.46

Source: the author.

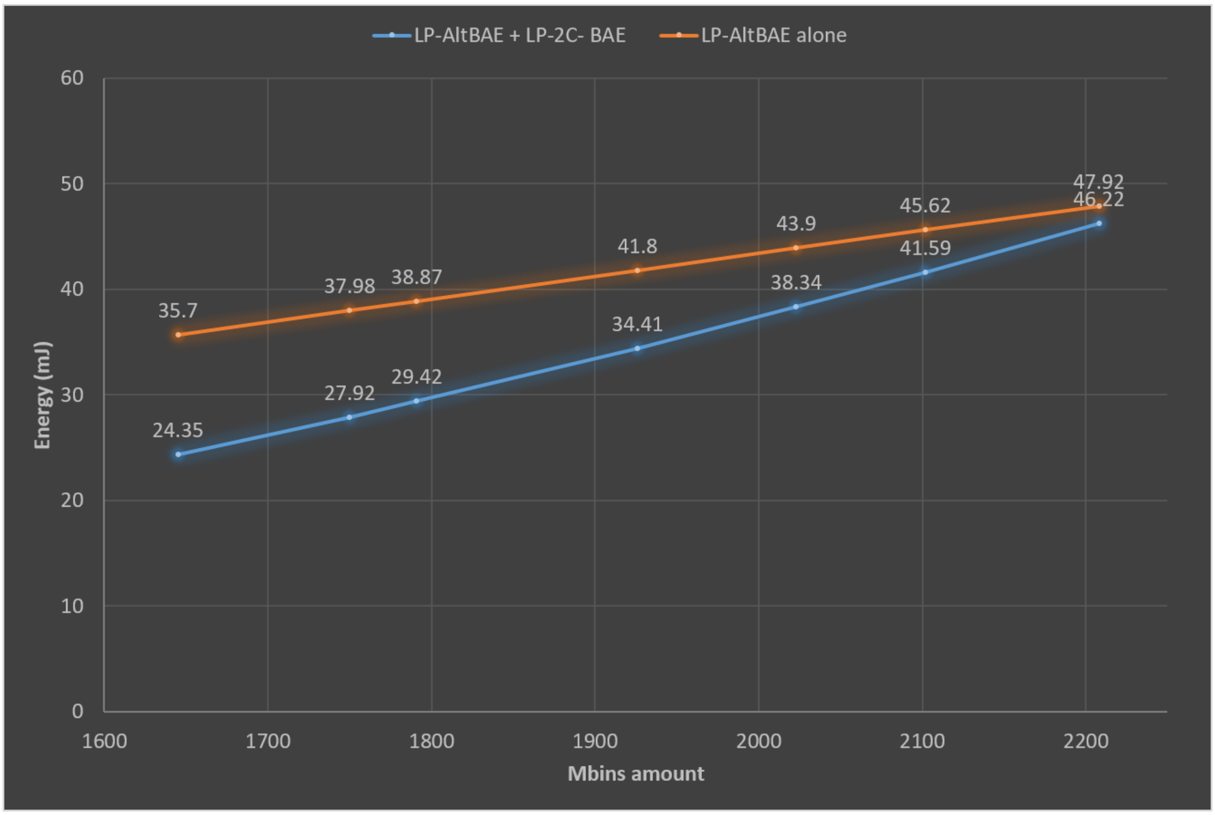
As a conclusion, it is noticeable that all scenarios consume less energy that the sole use of LP-AltBAE for the same maximum amount of Mbins achievable for each scenario in one second of processing (i.e., LP-AltBAE processes the bitrate accomplished by each scenario in less than one second – refer to Table 7.25 for means of clarity) . Moreover, the less demanding is the scenario, the less energy is consumed by the cases where LP-2C-BAE increases its operating percentage time.

Table 7.26 – Estimated energy savings using energy-efficient approach against LP-AltBAE

Scenarios	Maximum Mbins/s	Energy for the scenario (mJ)	Energy of LP-AltBAE sole (mJ)	Energy saving (%)
a	2,208.15	46.22	47.92	3.53
b	2,102.62	41.59	45.62	8.83
c	2,023.56	38.34	43.90	12.66
d	1,926.75	34.41	41.80	17.66
e	1,791.40	29.42	38.87	24.31
f	1,750.87	27.92	37.98	26.49
g	1,645.35	24.35	35.70	31.79

Source: the author.

Figure 7.4 – Energy comparison between the different proposed scenarios against LP-AltBAE

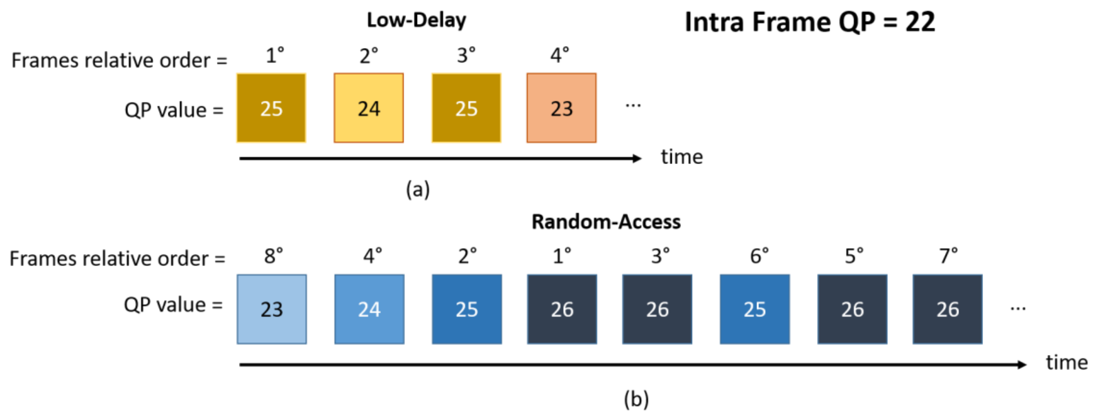


Source: the author.

Additionally, a potential methodology to perform the percentages presented on the scenarios ‘a’ to ‘g’ during the processing can account the GoP (Group-of-Pictures) structure proposed by the HEVC standard in recommended video sequences (BOSSEN, 2013). For instance, considering the base QP value (e.g., 22 or 37 in the analysis presented), this value, in fact, will be used only on intra-prediction frames (i.e., the first on the sequence and the next ones defined by the intra frame period – the amount of frames between intra-prediction frames). The next frames, which will be inter-predicted, will use different QP values. For LD and RA configurations, the following values will be as illustrated in Figure 7.5(a) and 7.5(b) for QP 22, and Figure 7.6(a) and Figure 7.6(b) for QP 37, respectively. Moreover, the decrease from one QP value to the next one can lead to more than 50% of bits (and therefore bins) generated by the frames (examples derived from HM simulation for some video sequence can be found in Appendix D of this Thesis).

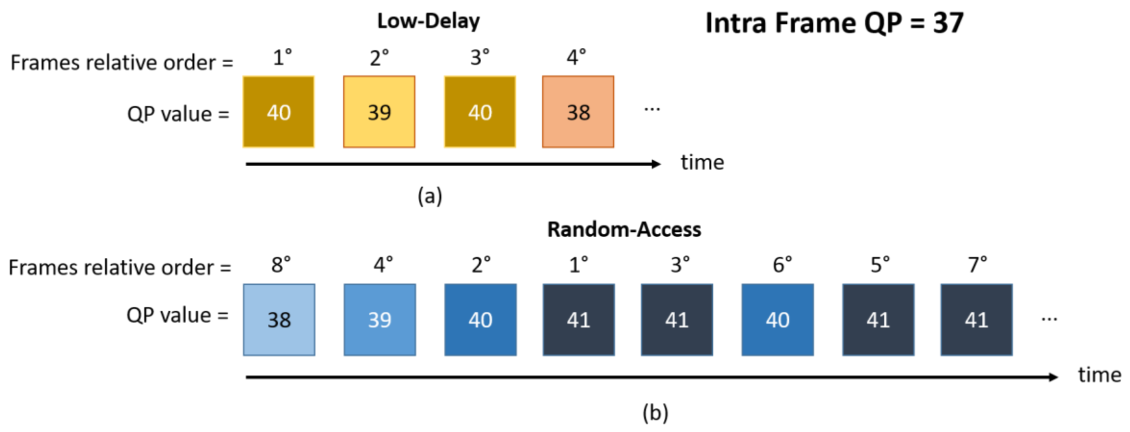
Table 7.27 and Table 7.28 present the occurrence of each QP-valued frame, according to the organization presented in Figure 7.5 and Figure 7.6, for initial QPs 22 and 37, respectively. Additionally, the bitrate contribution percentage a single of these frames has on the whole GoP bitstream,, the total bitrate contribution all the frames of the same QP value has within the GoP (i.e., the multiplication of the occurrence within a GoP times the contribution percentage of a single frame), and the average total contribution. As analyzing the data on Appendix D, within the same video sequence, the bitrate contribution for each type of frame does not vary much from one GoP to the other (i.e., the percentages for the first GoP tend to continue for all upcoming GoPs).

Figure 7.5 – GoP structure for recommended test video sequences – Intra Frame QP 22



Source: the author.

Figure 7.6 – GoP structure for recommended test video sequences – Intra Frame QP 37



Source: the author.

As a conclusion, one may notice that the frames contributions are heavily dependent on the QP associated. For example, in RA sequences, even if the QP 26 and QP 41 frames occur four times within a single GoP, the total bitstream contribution of these frames can be as low as 4% of the total, and as high as 24%. The QP 23 and QP 38 frames vary from 31% to 76% in the same scenarios, although occurring just once within a GoP. Therefore, a valid methodology could prefer to use LP-2C-BAE for higher QP values, where for smaller ones the LP-AltBAE, but trying to match the timing contribution these frames have within a GoP, for example.

The intra-prediction frames percentage depends on the intra-frame period, as already stated, and thus have an even smaller occurrence percentage along the video sequence. Nevertheless, due to the higher demand the lowest QP and the intra-prediction frame required, in the proposed heuristic, this type of pictures will always utilize the LP-AltBAE (e.g., sometimes, the intra-prediction frame can generate as much bitstream as nine times the next whole GoP).

Table 7.27 – Occurrence percentage for each value QP within a GoP – Intra Frame QP 22

LD – Intra frame QP 22					**RA – Intra frame QP 22**				
QP	**# within GoP**	**Bitstream Contribution (%)**			**QP**	**# within GoP**	**Bistream Contribution (%)**		
		Single frame	**All frames**	**Avg.**			**Single Frame**	**All frames**	**Avg.**
23	x1	36 - 61	36 - 61	49	23	x1	31 - 62	31 - 62	47
24	x1	19 - 25	19 - 25	22	24	x1	17 - 20	17 - 20	19
25	x2	8 - 19	16 - 38	27	25	x2	9 - 13	18 - 26	22
-	-	-	-	-	26	x4	1 - 6	4 - 24	14

Source: the author.

Table 7.28 – Occurrence percentage for each value QP within a GoP – Intra Frame QP 37

LD – Intra frame QP 37					**RA – Intra frame QP 37**				
QP	**# within GoP**	**Bitstream Contribution (%)**			**QP**	**# within GoP**	**Bitstream Contribution (%)**		
		Single frame	**All frames**	**Avg.**			**Single frame**	**All frames**	**Avg.**
38	x1	54 - 68	54 - 68	61	38	x1	48 - 76	48 - 76	62
39	x1	12 - 18	12 - 18	15	39	x1	10 - 19	10 - 19	15
40	x2	9 - 14	18 - 28	23	40	x2	4 - 10	8 - 20	14
-	-	-	-		41	x4	1 - 3	4 - 12	8

Source: the author.

Analyzing the data on Table 7.27 and Table 7.28, the discrepancy in terms of bitstream contribution of the frames among different video sequences, whereas the average values seem closely related for the same QP-valued frames for both LD and RA configurations. From this conclusion, there are three possible methodologies to be followed for the optimal usage of LP-AltBAE and LP-2C-BAE, namely: a static, a pseudo-dynamic, and a dynamic.

7.4.1.1 Static Methodology

The static corresponds to consider the average values pre-calculated for as many sequences as possible (as done on Table 7.27 and 7.28) and consider that statistics for any other sequence to be processed. The distribution to which frame either LP-AltBAE or LP-2C-BAE will be used is as follows:

1. As already stated, intra-prediction frames will always make use of LP-AltBAE for their processing.
2. Address the maximum frames within the GoP to the LP-2C-BAE, trying to match the percentage of time processing suggested by each scenario, starting by the highest QP valued first (i.e., the less demanding ones first).
3. Always try to address the maximum amount of frames in a row to a given architecture, in order to maximize the period the other circuit will remain idle (improving power efficiency, therefore).
4. Finally, all remaining frames within the GoP are addressed to the LP-AltBAE circuitry.

Table 7.29 – Example of static methodology for QP 22 sequences

Scenarios	GoP LD	GoP RA
a	LP-2C-BAE for a single QP 25 frames at every two GoPs.	LP-2C-BAE for the two last QP 26 frame of every GoP.
b	LP-2C-BAE for all QP 25 frames at every GoP.	LP-2C-BAE for the all QP 26 frame of every GoP and a single QP 25 frame ate every two GoPs.
c	LP-2C-BAE for all QP 25 frames in a GoP and for the QP 24 frame at every two GoPs.	LP-2C-BAE for all QP 26 and QP 25 frames of every GoP.
d	LP-2C-BAE for all QP 24 and 25 frames.	LP-2C-BAE all QP 26, QP 25, and QP 24 frames of every GoP.
e	LP-AltBAE for a single QP 23 frame at every two GoP.	LP-AltBAE for a single QP 23 frame at every two GoP.
f	LP-AltBAE for a single QP 23 frame at every three GoP.	LP-AltBAE for a single QP 23 frame at every three GoP.
g	LP-AltBAE for a single QP 23 frame at every four GoPs.	LP-AltBAE for a single QP 23 frame at every four GoPs.

Source: the author.

Table 7.29 shows an example of the static methodology, considering QP 22 as the intra-frame value. For instance, observing Table 7.29, one example to help illustrate the static heuristic is scenario ‘b’. QP 25 frames represent roughly 25% of bitstream contribution on LD configuration. Thus, these frames are suitable to utilize LP-2C-BAE, leaving all others to LP-AltBAE. As another illustrative example, one may observe the scenario ‘d’ for both LD and RA configurations. On LD configuration, the QP 23 frames represent, roughly, half of the total bitrate contribution. Therefore, the LP-AltBAE is used for all the QP 23 frames, whereas all the other QP-valued frames will make use of LP-2C-BAE.

7.4.1.2 Pseudo-dynamic Methodology

As one may conclude, the static heuristic has the disadvantage of not take into account the actual video sequence being processing to make the division of LP-AltBAE and LP-2C-BAE. Hence, a pseudo-dynamic methodology has to assume that a profiling tool will provide these contributions information for the first GoP and, based on this result, implies the upcoming division usage of LP-AltBAE and LP-2C-BAE. For the first GoP, the pure static methodology is the only option available. The starting point is also the average percentage values of the static approach. After the first GoP, the pseudo-dynamic approach may consider the following:

1. At every decrease of roughly 10-15% percentage contribution on the lowest inter-predicted QP-valued frames (e.g., QP 23) have compared to the average value of initial percentage, one may expect the proportion increase for the highest QP-valued frames of the GoP. Therefore, the new distribution of frames between LP-AltBAE and LP-2C-BAE is as in Table 7.30 for initial intra-frame QP equal to 22.
2. At every increase of roughly 10-15% percentage contribution on the lowest inter-predicted QP-valued frames have compared to the average value of initial percentage, one may expect the proportional decrease of the highest QP valued frames of the GoP. Hence, the new distribution of frames between LP-AltBAE and LP-2C-BAE is as in Table 7.31 for initial intra-frame QP equal to 22.
3. In case none of the above situations occurred, keep the initial distribution, as depicted in Table 7.29.

What is remarkable is that, since the GoP distribution rarely chances (considering the data on Appendix D), the pseudo-dynamic methodology could be employed only once after the first GoP, or at every arbitrary amount of GoPs to verify any drop in quality of the video, for instance (e.g., QP drops due to loss in bandwidth). The drawback of the pseudo-dynamic approach is that it still relies on the static distribution of LP-AltBAE and LP-2C-BAE among the frames, which may not be the optimal scenario to achieve the energy efficiency as shown in Table 7.26.

Table 7.30 – Pseudo-dynamic approach for QP 23 drop in bitstream contribution

Scenarios	GoP LD	GoP RA
a	LP-2C-BAE for a single QP 25 frames at every three GoPs.	LP-2C-BAE for the last QP 26 frame of every GoP.
b	LP-2C-BAE for a single QP 25 frames at every two GoPs.	LP-2C-BAE for the two last QP 26 frames of every GoP.
c	LP-2C-BAE for a single QP 25 frame at every GoP.	LP-2C-BAE for the all QP 26 frames of every GoP and a single QP 25 frame ate every two GoPs
d	LP-2C-BAE for all QP 25 frames and for the QP 24 frame at every two GoPs.	LP-2C-BAE for all QP 26 and QP 25 frames of every GoP.
e	LP-2C-BAE for all QP 24 and 25 frames.	LP-2C-BAE all QP 26, QP 25, and QP 24 frames of every GoP.
f	LP-AltBAE for a single QP 23 frame at every two GoPs.	LP-AltBAE for a single QP 23 frame at every two GoP.
g	LP-AltBAE for a single QP 23 frame at every three GoPs.	LP-AltBAE for a single QP 23 frame at every three GoPs.

Source: the author.

Table 7.31 – Pseudo-dynamic approach for QP 23 increase in bitstream contribution

Scenarios	GoP LD	GoP RA
a	LP-2C-BAE for a single QP 25 frames at every GoP.	LP-2C-BAE for all the QP 26 frame and a single QP 25 frame at every GoP
b	LP-2C-BAE for all QP 25 frames at every GoP and for the QP 24 frame at every two GoPs.	LP-2C-BAE for all the QP 26 and QP 25 frames of every GoP.
c	LP-2C-BAE for all QP 24 and 25 frames.	LP-2C-BAE all QP 26, QP 25, and QP 24 frames of every GoP.
d	LP-AltBAE for a single QP 23 frame at every two GoPs.	LP-AltBAE for a single QP 23 frame at every two GoPs.
e	LP-AltBAE for a single QP 23 frame at every three GoPs.	LP-AltBAE for a single QP 23 frame at every three GoPs.
f	LP-AltBAE for a single QP 23 frame at every four GoPs.	LP-AltBAE for a single QP 23 frame at every four GoPs.
g	LP-AltBAE for a single QP 23 frame at every five GoPs.	LP-AltBAE for a single QP 23 frame at every five GoPs.

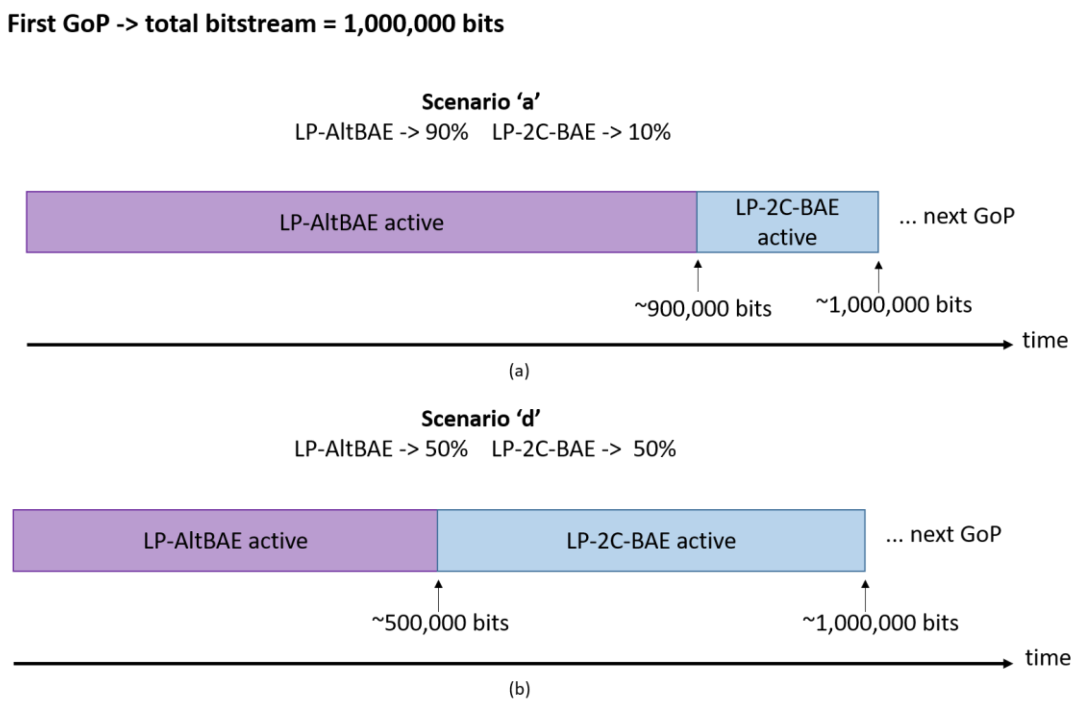
Source: the author.

7.4.1.3 Dynamic Methodology

Therefore, the last approach is the so-called dynamic one, where the profiling tool, GoP after GoP, store the bitstream generated amount for the GoP, and then calculate, based on the precise percentage of use either LP-AltBAE or LP-2C-BAE has to accomplish the scenarios of Table 7.26 for the next GoP, for instance. On this approach, there is no need to differentiate between the frames QPs, but solely on the total percentage of bits already produced to decide whether to change from one architecture to the other. Figure 7.7(a) and Figure 7.7(b) helps illustrate this approach for scenarios ‘a’ and ‘d’, respectively. Though being the most efficient in terms of energy efficiency, this last heuristic also requires more support from the encoder profiling tool, since it has to receive the information from when one GoP ends and the next one starts. Nevertheless, since the GoP bitrate does not seem to change significantly among the processing (refer to Appendix D), the calculation of the amount of bitstream to be processed by one or the other architecture can be done just once (e.g., for the first GoP).

Table 7.32 summarizes the three proposed approaches estimated characteristics. At this moment, this Thesis proposes to guide possibilities to drive the inception of the efficient use of the power-saving BAE designs. It is outside the scope of this Thesis to fully develop or analyze how they would behave along with the two BAE circuitry on the proposed scenarios of Table 7.26.

Figure 7.7 – Example of dynamic methodology for energy-efficient BAE design



Source: the author.

Table 7.32 – Summary of the methodologies for the energy-efficient BAE approach

Methodology	**# uses during the processing**	**Profiling support**	**Implementation effort**	**Compliance with Energy-efficient BAE**
Static	None	Not required	Low	Low
Pseudo-dynamic	At least once	Low	Medium	Medium
Dynamic	At least once	High	High	High (optimal)

Source: the author.

Besides the scenarios presented before, which are valid on limit situations (i.e., the future high-demanding real-time video processing requirements), where the usage of both LP-AltBAE and LP-2C-BAE lead to a better energy efficiency, whereas keeping the performance on an optimal value, other possibilities of switching between each architecture are also plausible in current real-time video processing situations. For instance, looking again to the tables in Appendix D, it is easily noticeable that between one QP frame to another with a QP added by one compared to the first, the bits (and therefore bins) demand drops considerably (sometimes around 50%). Hence, in a scenario that the quality of the video drops considerably, for instance, due to a sudden degradation in the bandwidth to transmit the video, it is possible to switch from LP-AltBAE to LP-2C-BAE to save energy.

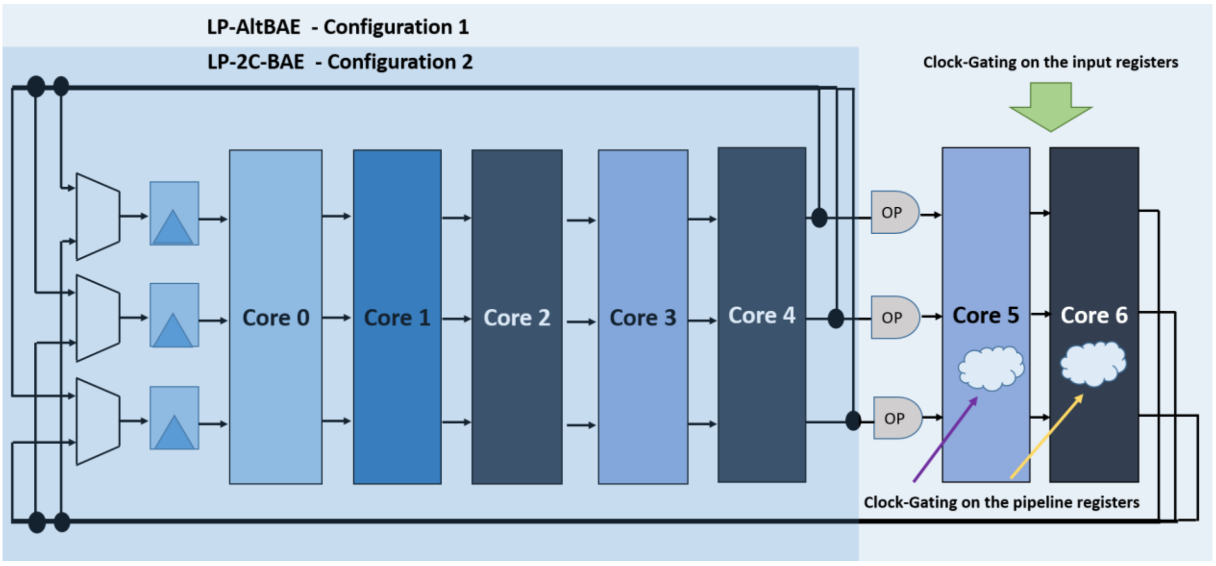
Furthermore, one may imagine a device, which encodes video, using a battery as a source of energy. Supposing the battery is running low, the user may want to choose to enter the device in battery-saving mode. In other words, by switching to an operational mode that will consume less energy, the proposed LP-AltBAE/LP-2C-BAE is a potential and highly suitable choice for the entropy-encoding block of this video encoding solution.

7.4.2 ET-BAE Architecture

Finally, considering the ASIC scope of this Thesis, both architectures LP-AltBAE and LP-2C-BAE have to co-exist in order to achieve the desired methodology for high-throughput and energy-efficiency. In fact, as already presented in Figure 7.1, the LP-2C-BAE is contained within the LP-AltBAE circuit. Thus, there is no need to have the two designs existing separately one another, but simply turning off the components of LP-AltBAE when the LP-2C-BAE is intended for usage (actually, having the two components completely separated would much possibly ruin the energy efficiency here proposed, along with a considerable area penalty, therefore). The point it to, whenever the LP-2C-BAE configuration is necessary, to thoroughly or almost completely turn-off the non-required circuitry from LP-AltBAE and, hence, possibly to almost zero dynamic power dissipation for this part of the whole component. The fashion to achieve that goal can be done by the usage of the already mentioned and used techniques: Operand Isolation and Clock Gating.

Figure 7.8 presents the final architectural proposal, named ET-BAE, where Configuration 1 is analogous to the LP-AltBAE design, whereas Configuration 2 to the LP-2C-BAE. One may notice that the registers of the *Range*, *Low*, and OB variables receive a multiplexer in front of them. The reason is to select the point where the current configuration shall end and store this value. One may seem these values may come from either the end of the whole component (i.e., LP-AltBAE) or from the boundaries between the two configurations (i.e., LP-2C-BAE). The second approach is to add a Clock Gating cell to every register on the input of core 5 and core 6. By doing so, when this part of the architecture is not required (i.e., LP-2C-BAE configuration), the entrances will have their values with the same value for many clock cycle, avoiding any switching activity for combinational logic of the first pipeline stage, and also short-circuit consumption due to the Clock-Gating. Furthermore, all the other pipeline registers also receive the same low-power technique, thus avoiding switching, and short-circuit on the registers, and any switching activity in the combinational logic within each pipeline stage, in the same fashion as proposed for the first pipeline stage (refer to Figure 6.6 for more details).

Figure 7.8 – ET-BAE architecture and configurations


LP-AltBAE - Configuration 1
LP-2C-BAE - Configuration 2
Clock-Gating on the input registers
OP
OP
OP
Core 0
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Clock-Gating on the pipeline registers

Source: the author.

Finally, Operand Isolation (e.g., an AND cell for every input of a given combinational logic) is applied to the inputs of the *Range*, *Low*, and OB right in front the entrance to the core 5 in Figure 7.8 (i.e., the AND symbol with OP inside). Therefore, any toggling generated on the previous cores does not affect any combinational logic up to this point (one may remember that *Range*, *Low*, and OB requires adders/subtractors and multiplexers to perform their update).

Table 7.33 presents the synthesis results of the ET-BAE, where one may notice the degradation in maximum frequency and area are according to the expected, and almost negligible (3.43% and 2.22%, respectively) when compared to the LP-AltBAE design solely. The reason for the frequency degradation is due to more logic insertion (i.e., the Operand

Isolation cells and new multiplexers), which occurs in the critical path of the original LP-AltBAE design (i.e., the *Range* update), which also influence into the area increase. Moreover, the new Clock-Gating cells are the other factor of area increase.

Table 7.33 – Synthesis results for ET-BAE

Design	Frequency		Area	
	Maximum	Degradation	Maximum	Degradation
ET-BAE	507 MHz	3.43 %	21.22 Kgates	2.22%

Source: the author.

Tables 7.34, 7.35, 7.36, 7.37, 7.38, and 7.39 presents the power results for both configurations of ET-BAE running the same video sequences used for LP-AltBAE and LP-2C-BAE analysis, at 250 MHz of clock frequency. As one may observe, there is an increase in terms of power when compared to the original solutions running separately, i.e., ET-BAE Configuration 1 against LP-AltBAE, and ET-BAE Configuration 2 against LP-2C-BAE (roughly ranging 10% to 14%, as can be seen in the mentioned tables). This result was also expected, since there is more circuitry which will be switching, and that did not exist in the previous designs separately.

On average, and as summarized in Table 7.40, for the same video sequences as for LP-AltBAE and LP-2C-BAE, ET-BAE on Configuration 1 consumes 26.178 mW, whereas on Configuration 2, it consumes 11.907 mW, respectively with a power degradation of 10.22% and 11.75% comparing with the analogous power dissipation of LP-AltBAE and LP-2C-BAE.

Table 7.34 – ET-BAE power dissipation results for the Traffic sequence

	Traffic				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	26.289	26.386	26.931	26.843	10.16
2	12.305	12.141	12.427	12.085	10.59

Source: the author.

Table 7.35 – ET-BAE power dissipation results for the Kimono sequence

	Kimono				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	23.182	25.983	23.311	25.800	9.79
2	10.689	11.749	10.729	11.757	12.28

Source: the author.

Table 7.36 – ET-BAE power dissipation results for the PartyScene sequence

	PartyScene				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	25.198	25.791	25.791	26.819	9.81
2	11.365	12.420	11.354	11.842	10.37

Source: the author.

Table 7.37 – ET-BAE power dissipation results for the BlowingBubbles sequence

	BlowingBubbles				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	26.753	26.522	27.292	28.258	10.36
2	12.013	12.893	12.180	12.860	14.16

Source: the author.

Table 7.38 – ET-BAE power dissipation results for the Johnny sequence

	Johnny				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	26.804	24.353	26.470	25.817	10.26
2	12.171	9.433	12.144	11.729	11.45

Source: the author.

Table 7.39 – ET-BAE power dissipation results for the ChinaSpeed sequence

	ChinaSpeed				
	Power Consumption (mW)				Power Degradation
Configuration	LD 22	LD 37	RA 22	RA 37	Average (%)
1	26.346	27.504	26.204	27.616	10.87
2	12.071	12.563	12.204	12.634	11.45

Source: the author.

Table 7.40 – Average power dissipation results for ET-BAE and power degradation

Configuration	Power Consumption for ET-BAE avg. (mW)	Power Consumption of LP-BAE designs avg. (mW)	Power Degradation avg. (%)
1	26.178	23.501	10.22
2	11.907	10.507	11.75

Source: the author.

Nevertheless, the advantage of ET-BAE against the separated usage of LP-AltBAE and LP-2C-BAE is confirmed: it is possible to switch between one configuration to the other during the operation, and thus reduce the energy required in a factor that is below the throughput decrease between each configuration (refer to Table 7.23) for most of the scenarios proposed. Table 7.41 and Figure 7.9 illustrate that reasoning, in a fashion as presented in Table 7.26: for the proposed scenarios, the necessary energy for ET-BAE with each configuration running for a certain percentage of time against the sole use of LP-AltBAE. One remark is that all power values (and thus the energy required) were updated now considering a linear increase factor of 2.028 from the values on Table 7.22 for LP-AltBAE (i.e., 250 MHz to 507 MHz – the maximum frequency ET-BAE runs), and from the average values each ET-BAE configuration consumes at 250 MHz, presented in Table 7.40.

What one may notice is that, due to the power increase of ET-BAE, the two first scenarios require more energy than the analogous ones by merely using LP-AltBAE, which is expected, due to the average power increase of 11.39% and 13.32%, respectively for Configuration 1 and Configuration 2 of ET-BAE, compared with the separated circuits. Nevertheless, all other scenarios, where the percentage of ET-BAE Configuration 2 starts to increase (starting on scenario ‘c’), ET-BAE requires less energy than the sole use of LP-AltBAE, even if LP-AltBAE requires less time to process the same amount of bins during one second of ET-BAE processing. Therefore, for the majority of cases, ET-BAE is still more efficient energetically than LP-AltBAE.

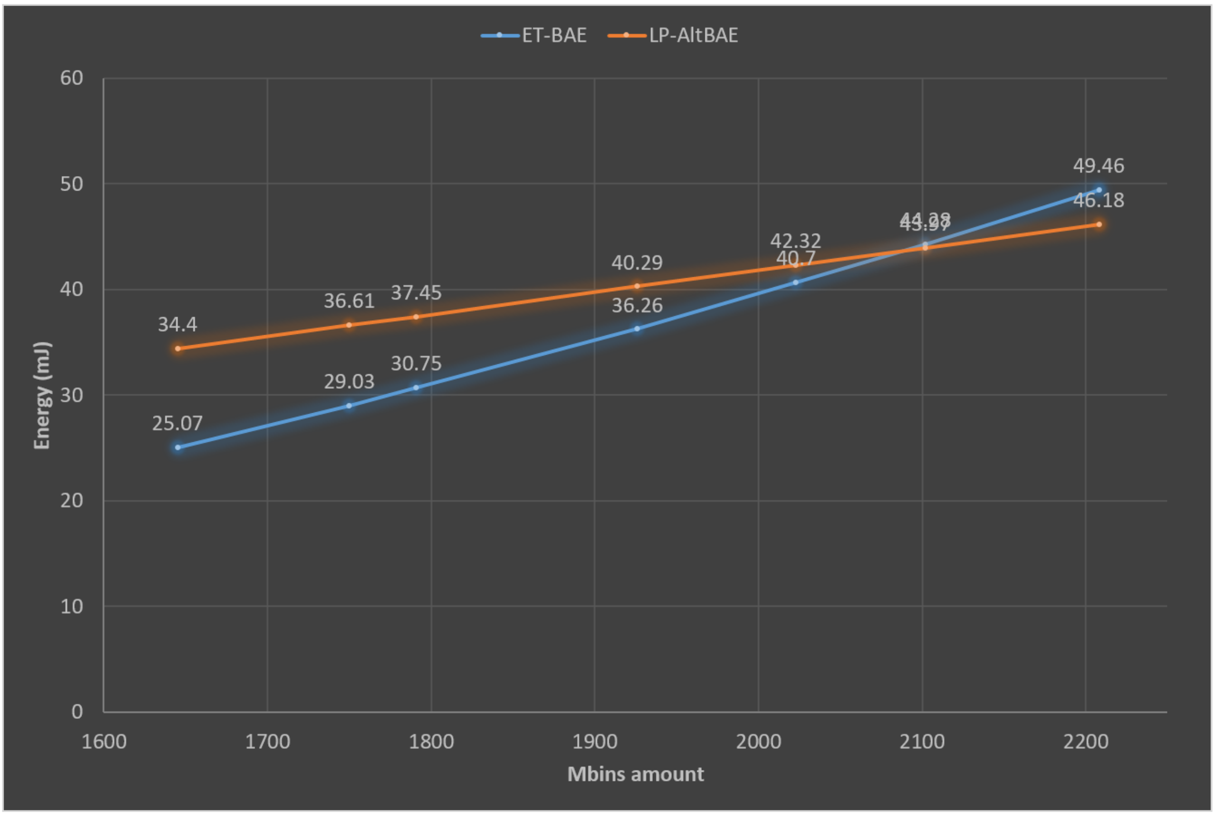
Table 7.41 – ET-BAE estimated energy savings against LP-AltBAE

Scenarios	Maximum Mbins/s	Energy for the scenario (mJ)	Energy of LP-AltBAE sole (mJ)	Energy saving (%)
a	2132.442	49.46	46.18	-7.11
b	2030.535	44.28	43.97	-0.70
c	1954.181	40.70	42.32	3.84
d	1860.69	36.26	40.29	10.00
e	1729.985	30.75	37.45	17.96
f	1690.845	29.03	36.61	20.69
g	1588.938	25.07	34.40	27.11

Source: the author.

Furthermore, in theory, only a design with both LP-AltBAE and LP-2C-BAE as separate components would be more efficient in terms of energy requirement than ET-BAE at the penalty of 73% more area compared to ET-BAE (i.e., the sum of area resources of LP-AltBAE and LP-2C-BAE separately). Finally, in the CMOS technology used for the synthesis (65 nm) the static power consumption is negligible (less than 1% of the total, as reported), what is not true if a smaller node is used, where the static component becomes considerable (SEMICONDUCTORS, 2013). Thus, the extra area would significantly influence the leakage (i.e., static power consumption).

Figure 7.9 – Energy comparison of ET-BAE against LP-AltBAE



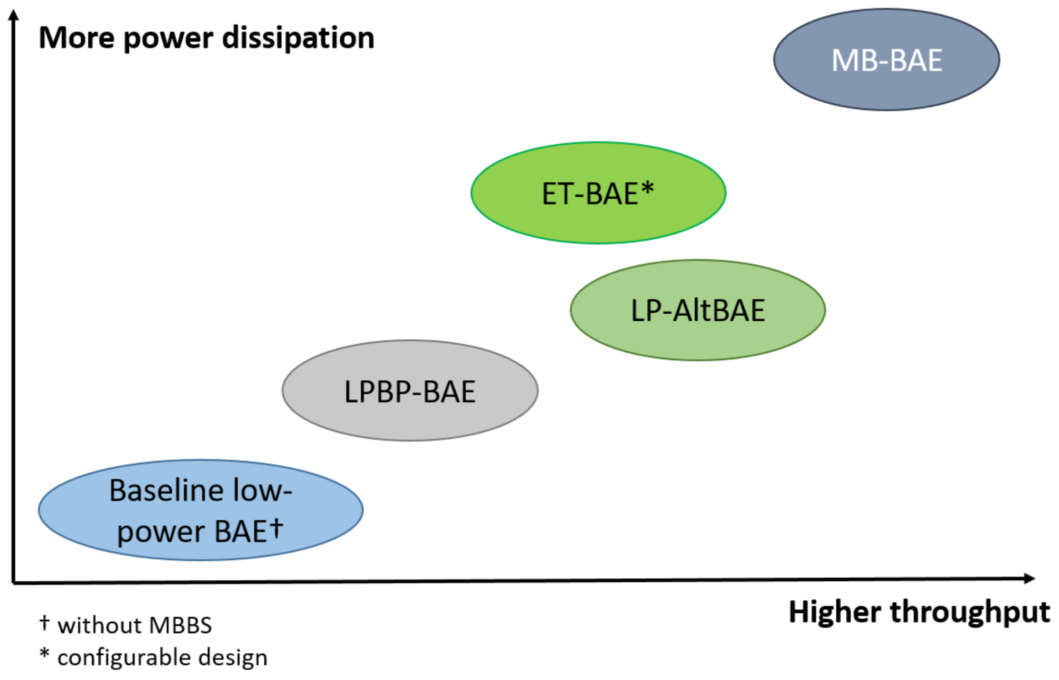
Source: the author.

7.5 Overall Comparisons of BAE Architectural Proposals of the Thesis

The main advantage of using ET-BAE is to have, in a single circuit (saving area, therefore), the possibility to choose, during execution, the most suitable configuration given the throughput and energy target required. Figure 7.10 depicts each hardware proposal developed by the author of this Thesis, showing the relative values for power dissipation and throughput for each design, along the axis Y and axis X, respectively. Table 7.42 summarizes and compares the characteristics of all proposed BAE designs developed in this Thesis. As a conclusion, for ultra-high demanding throughput target, the MB-BAE (described in Chapter 6) is the proper choice, at the cost of higher power dissipation and area. For throughput requirements which are still high, but lower that the one achieved by the MB-BAE, the LP-AltBAE is suitable, as it achieves a performance which is already above the real-time requirements of current HEVC defined levels, with the advantage of less power dissipation and area, compared to MB-BAE. For current HEVC maximum defined constraints, at the advantage of the smaller area and power values, the baseline low-power BAE (described in

Chapter 4), and LPBP-BAE (described in Chapter 5) are the most appropriate choices. Finally, the ET-BAE is the best option to accomplish a configurable architecture in which the proper configuration can be set according to the performance constraints required by the user application and the battery status at any given moment, while keeping the energy at a compromise level for most of the estimated high-throughput scenarios. Furthermore, ET-BAE does not suffer the huge area penalty of using LP-AltBAE and LP-2C-BAE as separate components.

Figure 7.10 – All proposed BAE designs characteristics for power dissipation and throughput



Source: the author.

Table 7.42 – Proposed BAE designs comparison

Design	Baseline low-power BAE	LPBP-BAE	MB-BAE	LP-AltBAE	LP-AltBAE + LP- 2C-BAE	ET-BAE – Configuration 1
#bins/cycle (avg)	4	4.56	5	4.34	4.34	**4.34**
Maximum Frequency (MHz)	280	264	537	527	527	**507**
#Mbins/s	1120	1204	2685	2287	2287	**2200**
CMOS Technology (nm)	45	65	65	65	65	**65**
Gate Count (K)	9.95	14.6	33	20.76	35.65	**21.22**
Power (mW) @ 250 MHz	2.49	15.93	39.947*	23.501	23.501†	**26.178**

* Tool-inferred results
† Based on LP-AltBAE sole dissipation

Source: the author.

As final remark, the ET-BAE architecture herein proposed is not only suitable for upcoming real-time video requirements with restricted energy scenario (i.e., battery-supplied devices), but also on current video processing scenarios with dynamic change conditions (e.g., QP increase during processing, battery level running low), in which the other proposed BAE architectures (e.g., MB-BAE and LP-AltBAE) do not offer any configuration feature.

8 CONCLUSION AND FUTURE WORKS

This Thesis has focused on dedicated hardware design for high-performance and energy-efficiency entropy encoder block of the HEVC standard. The main reasons highlighted for the research interest on this block were the difficulties in order to parallelize the input data processing for this block, especially considering UHD resolutions real-time video processing, and which accounts for a not negligible percentage of the operation time a given HEVC encoder may have. Moreover, the advantages concerning a high-throughput single-CABAC instance hardware block, such as a smaller area and power dissipation along with better coding efficiency, justify the intended goal of the research. Finally, a new era of even higher resolutions and the existence of more battery-based video processing devices seem imminent, in which the novel schemes in this Thesis may find a potential application. All the main innovations developed in this Thesis were on the Binary Arithmetic Encoder (BAE) block of CABAC, the bottleneck in terms of processing of the entropy-encoding algorithm.

A low-power BAE architecture was presented, based on a statistical analysis of the BAE inputs to corroborate the presented choices for power-saving techniques insertion. The use of these techniques led to power savings ranging from 10% to 40%, running on the gate-level netlist of the proposed architecture. Negligible degradation of the maximum frequency compared to a baseline BAE design occurred, showing that this approach is suitable for high-throughput BAE designs as well.

A novel architectural proposal for multiple-bypass bins processing, named MBBS, is presented and evaluated using a baseline BAE architecture, leading to improvement in bins/cycle throughput of around 14.36%, without any harm to the original maximum frequency achieved by the baseline BAE architecture, corroborating the theoretical gain previously calculated. Moreover, the same power-saving approach previously mentioned was applied, along with the MBBS. Power consumption results of the gate-level netlist using test video sequences as stimuli presented savings around 14.26%.

Additionally, two new architectural solutions for ultra-high-throughput BAE block, named MB-BAE and AltBAE were presented, wherein both the novel MBBS is inserted, with and without the usage of the BPBS technique, respectively. The throughput results for MB-BAE are around 13% above the related prior highest throughput architecture found in the literature, proving the efficiency of the use of MBBS for an ultra-high-throughput CABAC solution, and once again proving the theoretical gains expected. The result accomplished by the MB-BAE architecture point out a possible design to be used for future high-demanding real-time scenarios, such as potentially new video standards will face (e.g., VVC), and which intend to utilize BAE as entropy encoder kernel. AltBAE has presented closely related throughput results when compared to the highest throughput solutions of the literature, with the advantage of not using the BPBS to achieve its results. Thus, being an efficient solution in terms of less area, power, and time-to-market implementation.

Finally, the proposal of a configurable high-throughput and energy-efficient BAE occurs, named ET-BAE, whose intended application is for both current and future video processing situations. The ET-BAE solution is based on the efficient high-throughput proposal that utilizes the novel MBBS (i.e., AltBAE), which presented less power dissipation when compared to MB-BAE, and without proportional throughput degradation, pointing it as the best choice. Three versions of AltBAE (the original one, 2C-BAE, and 1C-BAE) are analyzed in terms of power dissipation using real video sequences, and as result, AltBAE and 2C-BAE present the best results in terms of throughput and energy-efficient. The application of the same power-saving techniques used in the first low-power BAE approach happened on AltBAE and 2C-BAE (named LP-AltBAE and LP-2C-BAE), presenting significant power gains compared to the original versions. Finally, a configurable ASIC architecture (i.e., ET-BAE) targeting future high-demanding real-time video scenarios, or even current battery-saving scenarios, is proposed, gathering LP-AltBAE and LP-2C-BAE in a single circuitry, and the related proposed methodologies of use, indicating a throughput-energy efficient trade-off alternative BAE design.

Along with the main contributions of this Thesis, additional results appear in Appendixes A and C. Architectural designs for the Binarization block were proposed (Appendix A), and an efficient architectural solution of the residual SEs generation is incepted (the major contributors of bins for CABAC), in order to avoid the starvation of the entropy encoding, due to the throughput improvements presented in this Thesis and also found in the recent literature (Appendix C).

As future works, a complete CABAC hardware solution, combining the MB-BAE and the ET-BAE with Binarization and Context Modeling blocks is the goal. Moreover, considering the upcoming of new standards (e.g., VVC – Versatile Video Coding), which intends to utilize CABAC, an entropy-encoding following the novelties here presented is also intended as future work within the context of these video coding standards. Another future work is to assess the proposed energy-efficient methodologies in a real CABAC block and to verify the real performance each one can actually deliver.

8.1 Publications by the Author

The publications by the author related to the theme of this Thesis, during the time span of the developed research, are listed below:

8.1.1 Journals

RAMOS, F. L. L.; SAGGIORATO, A. V.; ZATT, B.; PORTO, M.; BAMPI, S. Residual Syntax Elements Analysis and Design targeting High-Throughput HEVC CABAC. Submitted to the IEEE Transactions on Circuits and Systems I – Regular Papers (TCAS-I) – **Under third round of review.**

RAMOS, F. L. L.; ZATT, B.; PORTO, M.; BAMPI, S. Novel multiple bypass bin scheme and low-power approach for HEVC CABAC binary arithmetic encoder. Journal of Integrated Circuits and Systems (JICS), v. 13, n. 3, p. 1-11, Dec. 2018.

8.1.2 Conferences

RAMOS, F. L. L.; ZATT, B.; PORTO, M.; BAMPI, S. High-throughput binary arithmetic encoder using multiple-bypass bins processing for HEVC CABAC. Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy: IEEE, p. 1-5, 2018.

SAGGIORATO, A. V.; **RAMOS, F. L. L.**; ZATT, B.; PORTO, M.; BAMPI, S. HEVC residual syntax elements generation architecture for high-throughput CABAC design. Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS), Bordeaux, France: IEEE, p. 193-196, 2018 – **Best Student Paper Award.**

RAMOS, F. L. L.; ZATT, B.; PORTO, M. S.; BAMPI, S. Novel multiple bypass bins scheme for low-power UHD video processing HEVC binary arithmetic encoder architecture. Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI), Fortaleza, Brazil: ACM, p. 47-52, 2017.

ALONSO, C. M.; **RAMOS, F. L. L**.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power HEVC binarizer architecture for the CABAC block targeting UHD video processing. Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI), Fortaleza, Brazil: ACM, p. 30-35, 2017.

BONATTO, L. V. M.; **RAMOS, F. L. L**.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power multi-size HEVC DCT architecture proposal for QFHD video processing. Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI), Fortaleza, Brazil: ACM, p. 41-46, 2017.

RAMOS, F. L. L.; GOEBEL, J.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power hardware design for the HEVC binary arithmetic encoder targeting 8K videos. Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI), Belo Horizonte, Brazil: IEEE, p. 1-6, 2016.

REFERENCES

AGOSTINI, L. V. **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas a Compressão de Vídeo Segundo o Padrão H.264/AVC**. 2007. 172f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2007.

ALONSO, C. M.; RAMOS, F. L. L.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power HEVC binarizer architecture for the CABAC block targeting UHD video processing. **Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)**, Fortaleza, Brazil: ACM, p. 30-35, 2017.

AMDAHL, G. M. Validity of the single processor approach to achieving large scale computing capabilities. **Proceedings of the Sprint Joint Computer Conference (AFIPS)**, Atlantic City, NJ, USA: ACM, p. 483-485, 1967.

BHASKARAN, V.; KONSTANTINIDES, K. **Image and Video Compression Standards: Algorithms and Architectures**. s.l. : Kluwer Academic Publishers, 1997.

BOSSEN, F. **Common Test Conditions and Software Reference Configurations (JCTVC-L1100)**, 2013.

BROSS, B. CHEN, J.; LIU, S. Versatile Video Coding (Draft 3), document JVET-LI001. **Proceeding of 12th JVET meeting**, Oct. 2018.

CHEN, T. C.; CHEN, Y. H.; TSAI, S. F.; CHIEN, S. Y.; CHEN, L. G. Fast algorithm and architecture design of low-power integer motion estimation for H.264/AVC. **IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 17, n. 5, p. 568-577, Apr. 2007.

CHEN, J. W.; WU, L. C.; LIU, P. S.; LIN, Y. L. A high-throughput fully hardwired CABAC encoder for QFHD H.264/AVC main profile video**. IEEE Transactions on Consumers Electronics (TCE)**, v. 56, n. 4, p. 2529-2536, Nov. 2010.

CHEN, Y. H.; SZE, V. A deeply pipelined CABAC decoder for HEVC supporting level 6.2 high-tier applications**. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 25, n. 5, p. 856-868, May 2015.

CHI, C. C., MESA, M. A., JUURLINK, B., CLARE, G., HENRY, F., PATEUX, S., SCHIERL, T. Parallel Scalability and Efficiency of HEVC Parallelization Approaches. **IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1827-1838, Oct. 2012.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology, 2013-2018**. White paper, Jun. 2014.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology, 2017-2022**. White Paper, Sep. 2017.

CORREALE, A. Overview of the power minimization techniques in the IBM PowerPC 4xx embedded controllers. **Proceedings of the International Symposium on Low Power Electronics Design (ISLPED)**, Dana Point, CA, USA: ACM, p. 75-80, 1995.

DINIZ, C. M**. Dedicated and Reconfigurable Hardware Accelerators for High Efficiency Video Coding Standard**. 2015. 141f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2015.

FEI, W.; ZHOU, D.; GOTO, S. 1 Gbin/s CABAC encoder for H.264/AVC. **Proceedings of the European Signal Processing Conference (EUSIPCO)**, Barcelona, Spain: IEEE, p. 1524-1528, 2011.

GHANBARI, M. **Standard Codecs: Image Compression to Advanced Video Coding**. s.l. : Institution Electrical Engineers, 2003.

HEADJACK. HEVC, VP9 and the Future of Video Codecs. Available: https://headjack.io/blog/hevc-vp9-vp10-dalaa-thor-netvc-future-video-codecs/. Access: Jul. 2018.

HEVC Test Model (HM) version 16.6. Available: https: //hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-16.6. Access: Jul. 2016.

ITU-T and ISO/IEC, **Advanced video coding for generic audiovisual services, ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC)**, 2003.

ITU-T and ISO/IEC, **High Efficiency Video Coding, ITU-T Recommendation H.265 and ISO/IEC 23008-2**, 2013.

KRSTIC, M.; GRASS, E.; FAN, X. Asynchronous GALS design overview and perspectives. **Proceedings of the New Generation of CAS (NGCAS)**, Genova, Italy: IEEE, p. 85-88, 2017.

KUO, C. C.; LEI, S. F. Design of low-power architecture for CABAC encoder in H.264. **Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems (APCCAS).** Singapore, Singapore: IEEE, p. 243-246, 2006.

LIU, Y.; SONG, T.; SHIMAMOTO, T. High performance binarizer for H.264/AVC CABAC. **Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE)**. Wuhan, China: IEEE, p. 2237-2240, 2011a.

LIU, Z.; WANG, D. One-round renormalization based 2-bin/cycle H.264/AVC CABAC encoder. **Proceedings of the IEEE International Conference on Image Processing (ICIP).** Brussels, Belgium: IEEE, p. 369-372, 2011b.

MARPE, D.; SCHWARZ, H.; WIEGAND, T. Context-based binary arithmetic coding in the H.264/AVC video compression standard**. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 13, n. 7, p. 620-636, Jul. 2003.

MARTINS, A. L. M.; ROSA, V.; BAMPI, S. A low-cost hardware architecture binarizer design for the H.264/AVC CABAC entropy coding. **Proceedings of the 17th IEEE**

International Conference on Electronics, Circuits and Systems (ICECS). Athens, Greece: IEEE, p. 392-395, 2010.

MIANO, J. **Compressed Image File Formats: Jpeg, Png, Gif, Xbm, Bmp.** Boston: ACM Press, 1999.

MOFFAT, A.; NEAL, R. M.; WITTEN, I. H. Arithmetic coding revisited. **Proceedings of the IEEE Data Compression Conference (DCC)**, Snowbird, UT, USA: IEEE, p. 202-211, 1995.

MONTEIRO, E.; GRELLERT, M.; BAMPI, S.; ZATT, B. Rate-distortion and energy performance of HEVC and H.264/AVC encoders: a comparative analysis. **Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)**. Lisbon, Portugal: IEEE, p. 1278-1281, 2015.

MONTEIRO, E, R. **Caracterização Energética da Codificação de Vídeo de Alta Eficiência (HEVC) em Processador de Propósito Geral**. 2017. 144 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2017.

PENG, B.; DING, D.; ZHU, X.; YU, L. A hardware CABAC encoder for HEVC. **Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS).** Beijing, China: IEEE, p. 1372-1375, 2013.

PORTO, R. **Desenvolvimento Arquitetural para Estimação de Movimento de Blocos de Tamanhos Variáveis Segundo o Padrão H.264/AVC de Compressão de Vídeo Digital, 2008.** 96f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2008.

RICHARDSON, I. **Video Codec Design: Developing Image and Video Compression Systems**. Chichester: John Wiley and Sons, 2002.

RICHARDSON, I. **The H.264/AVC Advanced Video Compression**. 2nd Edition. Chichester: John Wiley and Sons, 2010.

RAMOS, F. L. L.; GOEBEL, J.; ZATT, B.; PORTO, M.; BAMPI, S. Low-power hardware design for the HEVC binary arithmetic encoder targeting 8K videos. **Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)**, Belo Horizonte, Brazil: IEEE, p. 1-6, 2016.

RAMOS, F. L. L.; ZATT, B.; PORTO, M. S.; BAMPI, S. Novel multiple bypass bins scheme for low-power UHD video processing HEVC binary arithmetic encoder architecture. **Proceedings of the Symposium on Integrated Circuits and Systems Design (SBCCI)**, Fortaleza, Brazil: ACM, p. 47-52, 2017.

RAMOS, F. L. L.; ZATT, B.; PORTO, M.; BAMPI, S. High-throughput binary arithmetic encoder using multiple-bypass bins processing for HEVC CABAC. **Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)**, Florence, Italy: IEEE, p. 1-5, 2018a.

RAMOS, F. L. L.; ZATT, B.; PORTO, M.; BAMPI, S. Novel multiple bypass bin scheme and low-power approach for HEVC CABAC binary arithmetic encoder. **Journal of Integrated Circuits and Systems (JICS)**, v. 13, n. 3, p. 1-11, Dec. 2018b.

SAGGIORATO, A. P. **Arquitetura Eficiente para a Geração dos Elementos Sintáticos advindos dos Resíduos de Transformada, segundo o Padrão HEVC.** 2017. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Computação) – Unipampa, Bagé, 2017.

SAGGIORATO, A. P.; RAMOS, F. L. L.; ZATT, B.; PORTO, BAMPI, S. HEVC residual syntax elements generation architecture for high-throughput CABAC design. **Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS)**, Bordeaux, France: IEEE, p. 193-196, 2018.

SEMICONDUCTORS INDUSTRY ASSOCIATION. **2013 International Technology Roadmap for Semiconductors (ITRS).** White paper, Aug. 2013

SHANNON, C. E. A Mathematical Theory of Communication. **The Bell System Technical Journal**, v. 27, n. 3, p. 379-423, Jul. 1948.

STANKOWSKI, J.; KARWOSKI, D.; GRAJEK, T.; WEGNER, K.; SIAST, J.; KLIMASZEWSKI, K.; STANKIEWICZ, O.; DOMÁNSKI, M. Bitrate distribution of syntax elements in the HEVC encoded video. **Proceedings of the International Conference on Signals and Electronics Systems (ICSES)**, Poznan, Poland: IEEE, p. 1-4, 2014.

SULLIVAN, G. J.; OHM, J. R.; HAN, W. J.; WIEGAND, T. Overview of the High Efficiency Video Coding (HEVC) standard**. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 22, n. 12, p. 1649-1668, Dec. 2012.

SUMMERS, J.; BREACHT, T.; EAGER, D; GUTARIN, A. Charactering the workload of a Netflix streaming video service. **Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)**. Providence, RI, USA: IEEE, p. 1-12, 2016.

SZE, V.; BUDAGAVI, M. A comparison of CABAC throughput for HEVC/H.265 vs. AVC/H.264. **Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS).** Taipei, Taiwan: IEEE, p. 165-170, 2013.

SZE, V.; BUDAGAVI, M.; SULLIVAN, G. J. **High Efficiency Video Coding (HEVC): Algorithms and Architecture**. s.l.: Springer, 2014.

VIZZOTTO, B.; MAZUI, V.; BAMPI, S. Area efficient and high throughput CABAC encoder architecture for HEVC. **Proceedings of the IEEE International Conference on Electronics, Circuits and Systems (ICECS).** Cairo, Egypt: IEEE, p. 572-575, 2015.

WESTE, N. H. E.; HARRIS, D. M. **CMOS VLSI Design: A Circuits and Systems Perspective**. s.l: Pearson, 2011.

WON, K.; JEON, B. Complexity-efficient rate estimation for mode decision on the HEVC encoder. **IEEE Transactions on Broadcasting**, v. 61, n. 3, p. 425-435, Sep. 2015.

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

ZATT, B.; SHAFIQUE, M.; BAMPI, S.; HENKEL, J. A multi-level dynamic complexity reduction scheme for multiview video coding. **Proceedings of the IEEE International Conference on Image Processing (ICIP)**. Brussels, Belgium: IEEE, p. 749-752, 2011.

ZATT, B. **Energy-efficient Algorithm and Architectures for Multiview Video Coding.** 2012. 236f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre, 2012.

ZHOU, D.; ZHOU, J.; FEI, W.; GOTO, S. Ultra-high-throughput VLSI architecture of H.265/HEVC CABAC encoder for UHDTV applications**. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)**, v. 25, n. 3, p. 497-507, Mar. 2015.

ZHOU, J.; ZHOU, D.; FEI, W.; GOTO, S. A high-performance CABAC encoder architecture for HEVC and H.264/AVC**. Proceedings of the IEEE International Conference on Image Processing (ICIP)**. Melbourne, VIC, Australia: IEEE, p. 1568-1572, 2013.

APPENDIX A <LOW-POWER BINARIZATION ARCHITECTURE>

This Appendix presents the Binarization block made in the context of this Thesis, first presented in SBCCI 2017 (Symposium on Integrated Circuits and Systems Design) (ALONSO, 2017). At first, an statistical analysis to assess the most used types of binarization is presented, using recommended video sequences. The results drives the inception of the low-power Binarization architecture. Finally, simulation and synthesis results, along comparisons with related works, are presented.

A.1 Statistical Analysis of Binarization Methods Occurrence Rate.

The occurrence rate of SEs and thus the type of binarization required for them may vary according the given parameters for a given video sequence (e.g. Quantization Parameter (QP), resolution of the video, configuration used, etc). A statistical analysis of recommended test sequences is an alternative in order to assess the data to verify any suitable low-power alternatives for a possible Binarizer architecture.

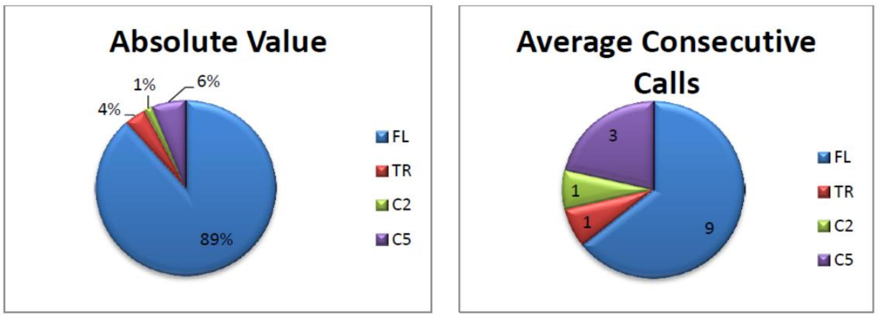
Two test video sequences were used (BOSSEN, 2013) with different video resolutions: PeopleOnStreet (2560x1600) and BasketballDrive (1920x1080) respectively named as WQXGA and 1080 HD resolutions (ITU-T, 2013). For each sequence, four rounds of processing were made with two different configurations, LowDelay (LD) and RandomAccess (RA); and two QPs, 22 and 37, which are the lower and upper limit values defined by (BOSSEN, 2013). The total eight rounds were run over the HEVC Software Reference 16.6 (HM) (HM, 2016) for the initial frames of the sequence (due to limitations in the data processing available resources), and as results we obtained the average total occurrence percentage of the occurring binarization methods; and the average consecutive calls for those same occurring binarization methods.

The average results (i.e. average results for the four rounds) for the PeopleOnStreet analyses are presented in Figure A.1. As was expected, not all Binarization methods were called for the initial frames, but the results are significant: 89% of the total amount of binarizations called corresponds to the FL method, whereas the others summed make a total of 11%. Moreover, Figure A.1 also shows the total consecutive calls for the aforementioned methods: as one may see, when a FL method is called at first, the tendency to be called again is highly probable for at least nine times after that first one. For the C5 method, it tends to be called consecutively for three times.

Depicted in Figure A.2 are the average results for the BasketballDrive sequence. The assessment showed values close to the ones presented for the previous sequence: a total occurrence percentage of 93% for the FL method, and 7% for all others; and an average FL consecutive calls of 11.25 times, while for the C5 an average of 2.75 consecutive calls.

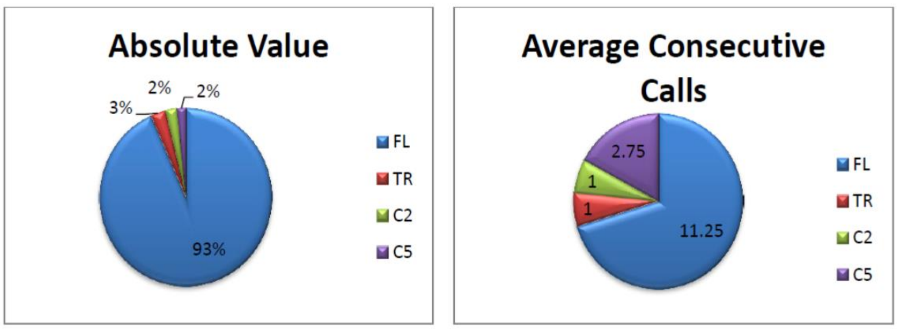
The results are expected, since the majority of SEs undergoes the FL binarization method (ITU-T, 2013). The significant occurrence of the C5 method was also expected, since it is used by a relevant SE generated by transform residues, which comprise 60%-90% of total data generated by a given video sequence according the QP variation (STANKOWSKI, 2014). Nevertheless, a very interesting result is the average consecutive calls, especially for the FL method: as one may remember, this method does not re-encode the SE, but simply limits its original binary representation. All other methods do infer a more complex logic (and this will imply in an interesting opportunity for low-power alternative at the design level to be presented next).

Figure A.1 – Total average percentage and average consecutive calls for the PeopleOnStreet sequence



Source: (ALONSO, 2017).

Figure A.2 – Total average percentage and average consecutive calls for the BasketballDrive sequence



Source: (ALONSO, 2017).

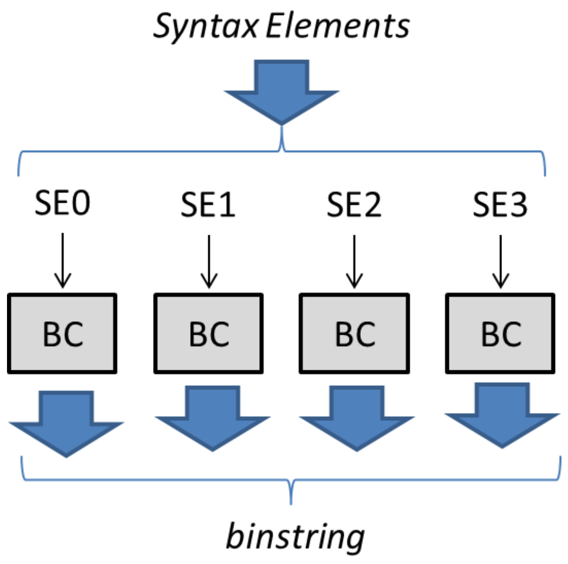
A.2 Low-Power Binarization Proposal

The work of (ZHOU, 2015) proposes high-throughput CABAC architecture, with many state-of-the-art improvements, especially for the critical CABAC block, which is the BAE. Nevertheless, in order to achieve the maximum average throughput of BAE (around 4.36 bins/cycle), the Binarizer block was overestimated and thus avoiding it to become the bottleneck of the architecture. The proposed CABAC achieves a throughput higher than 1-Gbin/s, fulfilling the requirements for 8K UHD videos at 6.2 high tier of the HEVC standard (CHEN, 2015).

The Binarizer block of (ZHOU, 2015) is composed of two FSE (Full Syntax Element) and two SSE (Simplified Syntax Element). The differences between them is that FSE is able to process all SEs types with all possible sizes, while SSE is able to process only some types of SEs, limited to two bits of size.

We decided to use the same overestimated Binarizer proposal of (ZHOU, 2015), with a slightly different change: we simply used four FSE cores, which we now named as Binarization Core (BC), as can be seen in Figure A.3, able to process all types of SEs with all possible sizes. The decision was made because we analyzed that the restriction of using the SSE would have a minor impact in power consumption on the proposal we intended, since power is our focus on this work. Hence, any four SEs could be inserted to the Binarizer architecture at any given cycle of the processing.

Figure A.3 – Four cores binarizer architecture

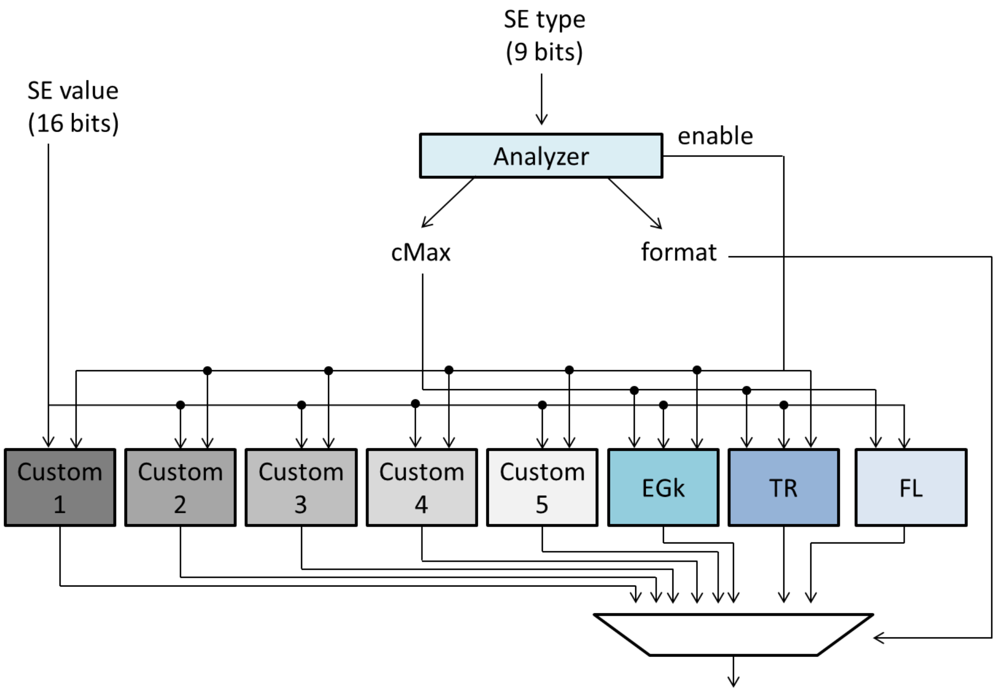


Source: (ALONSO, 2017).

A single BC is shown in Figure A.4, based on architecture of (ZHOU, 2015), where we have an Analyzer block, which indicates what is the type of the current processed SE, and what is the respective binarization method to be used (via the *format* signal). Next to it, we have the logic for the eight effectively used methods (the U and TU methods are used internally to some of the other methods (ITU-T, 2013)). As one may notice, at the original Binarizer of (ZHOU, 2015), when a SE is binarized, it is actually processed by all the available methods at once, and a multiplexer logic selects the correct binarization at the end, via the *format* signal, as already cited (i.e there were no *enable* signal). Thus, all eight logics are active for the four BCs cores, even when we only need one of them for each of the cores at once to fulfill the functionality.

Based on the arguments of previous paragraph and the results presented at previous section, we decided to apply an AND-based Operand Isolation (CORREALE, 1995) directly to the inputs of the binarization methods on each of the BCs cores, as shown in Figure A.5. An AND-logic is placed right before each bit of all inputs, and the *enable* indicates whether the original input value or a zero value shall be sent to the logic.

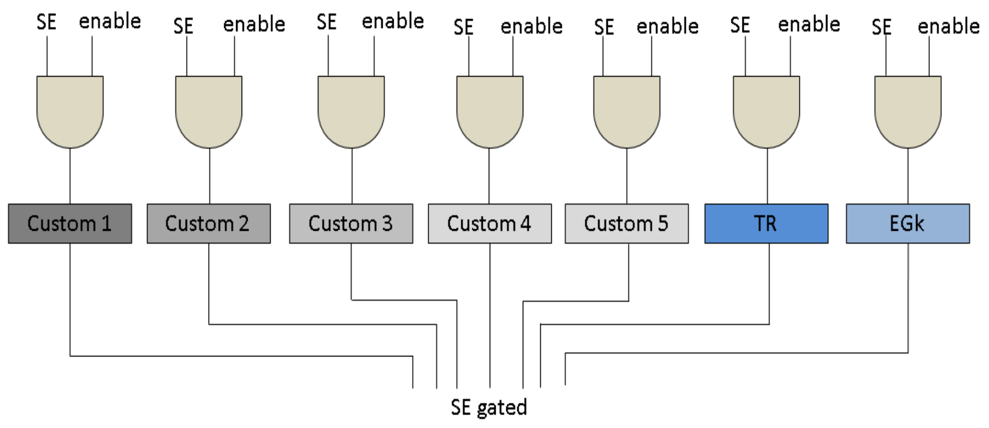
Figure A.4 – Binarizer core architecture



Source: (ALONSO, 2017).

The *format* signal still chooses among the correct binarization methods to be used at a given moment. The difference is that, at the proposed design, only one core will effectively be switching (i.e. only the correct logic will have the input being correctly fed to it, while the others will have a zeroed vector input). Considering the amount of consecutive calls that a single method may undergo, as presented at previous section, potentially we will keep the same zeroed vector input for all other logics for many cycles and thus generating no switching activities for the not required binarization logics. The only exception, as can be seen in Figure A.4 and Figure A.5, is FL method. The reason is simple: the method is extremely simple, as said before (it does not re-encoder the SE); and it is the most and more consecutively called method of all (refer to section A.1) and the use of the low-power technique would only increase power consumption instead of decrease.

Figure A.5 – Operand Isolation insertion on binarization methods



Source: (ALONSO, 2017).

A.3 Simulation and Synthesis Results

Four Binarizer architecture versions were made and described in VHDL: a baseline single-BC version without the Operand Isolation technique, named SBA-BIN; and other with the low-power insertion presented at previous section, named SLP-BIN. A baseline four-BC without the low-power approach was also made, named FBA-BIN, as long as a four-BC with the Operand Isolation applied to all its cores, called FLP-BIN. All versions were synthesized for the 65nm ST PDK for the worst corner conditions (worst process, 0.95 V and 125° C) using Cadence RTL Compiler.

Table A.1 presents the synthesis results. As one may notice, both single version (SBA-BIN and SLP-BIN) have almost four times less gates than the full version (FBA-BIN and FLP-BIN), as expected, since the full architectures process four SEs/cycle. As expected, the low-power versions for both the single and full cases have more gates than the baseline version of them (around 3.24 % and 1.54%, respectively), since the AND logics for the

Operand Isolations are added to the low-power versions. Nevertheless, minor impact in area was noticed. The more curious result is the increase in maximum frequency for the low-power architectures. The opposite was expected, since ANDs were inserted in the possible critical path of the designs. One possible explanation is the choice of the synthesis tool to utilize a suitable cell of the PDK which alone would do more logic in a single cell, while for the baseline versions the tool could not do the same type of optimization, leading to a smaller clock frequency result.

The gate-level netlist simulations were done for the same test sequences presented for the statistical analysis of section 3, and for the same variation of parameters (configuration and QPs). The simulation used 500 MHz as clock frequency and that value was chosen due to its proximity to the maximum frequency achieved by the CABAC architecture presented in (ZHOU, 2015), which was 420 MHz. This way, extremely high frequency and thus pessimistic value for power was used to verify the power behavior at these conditions (knowing that the Binarizer is not the performance bottleneck of CABAC as whole and that frequency is highly above the value the Binarizer would face in real life).

Table A.1 – Proposed binarizer synthesis results

Version	Clock Frequency	Gate Count
SBA-BIN	850 MHz	3.08 Kgates
SLP-BIN	858 MHz	3.18 Kgates
FBA-BIN	826 MHz	11.67 Kgates
FLP-BIN	834 MHz	11.85 Kgates

Source: (ALONSO, 2017).

Table A.2 shows the results for the single-BC versions. Power savings ranging 12% to 41% where accomplished when we compare the power values of SBA-BIN to SLP-BIN. Higher QP showed better power improvements, and the possible explanation is that, due to the increase in QP, SE transform residues tend to appear less (STANKOWSKI, 2014), which make use of binarization methods besides FL less frequent. Hence, at SBA-BIN version, the non-FL methods were always active, even when they were not required.

Table A.3 shows the results for the four-BC versions, as long with the bins throughput for the FLP-BIN. An equivalent behavior as the presented in Table A.2 is shown: power savings ranging 21% to 37% are achieved, and better results are reported for higher QPs. The total average power consumption of 1.87 mW and power savings of 22% are presented for FLP-BIN. A throughput of average 8.34 bins/cycle is shown, and considering the maximum frequency achieved by FLP-BIN of 834 MHz, the design achieves around 7 Gbin/s, proving that the low-power four-BC architecture is able to reach the requirements for 8K UHD videos at 6.2 high tier (more than 1 Gbin/s (CHEN, 2015)).

Table A.2 – Power results for the single-core binarizer architectures

Videos	Config.	QP	SBA-BIN	SLP-BIN	Power Saving
BasketballDrive	LD	22	0.596 mW	0.486 mW	17%
		37	0.640 mW	0.454 mW	41%
	RA	22	0.518 mW	0.450 mW	15%
		37	0.639 mW	0.452 mW	41%
PeopleOn Street	LD	22	0.524 mW	0.469 mW	12%
		37	0.610 mW	0.476 mW	28%
	RA	22	0.522 mW	0.468 mW	12%
		37	0.606 mW	0.475 mW	28%
Average			**0.581 mW**	**0.466 mW**	**20%**

Source: (ALONSO, 2017).

Table A.3 – Power and throughput results for the four-core binarizer architectures

Videos	Config.	QP	FBA-BIN	FLP-BIN	Power Saving	Bins/ Cycle
BasketballDrive	LD	22	2.46 mW	1.97 mW	25%	7.09
		37	2.38 mW	1.74 mW	37%	10.81
	RA	22	2.34 mW	1.90 mW	23%	6.70
		37	2.37 mW	1.73 mW	37%	10.77
PeopleOn Street	LD	22	2.38 mW	1.96 mW	21%	6.56
		37	2.51 mW	1.86 mW	35%	9.27
	RA	22	2.35 mW	1.94 mW	21%	6.45
		37	2.50 mW	1.88 mW	33%	9.05
Average			**2.41 mW**	**1.87 mW**	**22%**	**8.34**

Source: (ALONSO, 2017).

Table A.4 shows the comparisons with related works which were focused on the Binarizer or that we could extract relevant information from the Binarizer block. None of them presented power results for the sole Binarizer (nor for the CABAC in some cases), which attests the novelty of our proposed low-power design. One important remark is the use of gate-level simulation to obtain average power consumption by utilizing real video sequence as stimuli. This methodology supplies a more accurate power value than simply using RTL simulation, or tool-inferred switching activity.

The gate count and maximum frequency and throughput results of (ZHOU, 2015) are related to the CABAC as a whole and thus there would not be a fair comparison between it and FLP-BIN. Nevertheless, since the design of FLP-BIN is based on the Binarizer of

(ZHOU, 2015), it is predicable that the values for frequency and area would be very close one another. Compared to (MARTINS, 2010) and (LIU, 2011a), the proposed design achieves higher clock frequency and throughput, and also less equivalent gates when compared to (LIU, 2011a).

Table A.4 – Comparison with related works

Design	**(MARTINS, 2010)**	**(LIU, 2011a)**	**(ZHOU, 2015)**	**FLP-BIN**
Video Standard	H.264	H.264	HEVC	**HEVC**
Technology	180 nm	90 nm	90 nm	**65 nm**
Frequency	370 MHz	250 MHz	420 MHz*	**834 MHz**
Gate count	1.8 Kgates	19.3 Kgates	64.1 Kgates*	**11.85 Kgates**
Throughput	1 SE/cycle	1-2 SE/cycle	2 SE/cycle + 2-bits SE/cycle	**4 SE/cycle**
	0.42 bins/cycle	4.11 bins/cycle	4.36 bins/cycle*	**8.34 bins/cycle**
Power Consumption	-	-	-	**1.87 mW**

* Whole CABAC values

Source: (ALONSO, 2017).

APPENDIX B <ALTERNATIVE APPROACH TO EXPLAIN MBBS INCEPTION>

This appendix purpose is to give a different explanation on how to derive the MBBS technique proposed in Chapter 5 of this Thesis.

For any two bypass bins occurring in sequence, there are nine possible outcomes, which are related to renormalization conditions presented in Figure 5.1: the 1st condition followed by the 1st, the 2nd, or the 3rd condition. The 2nd condition followed by the 1st, 2nd, or 3rd condition. Finally, the 3rd condition followed by the 1st, by the 2nd, or by the 3rd condition.

In order to illustrate, and to facilitate the understanding, significant examples are presented in Figure B.1, Figure B.2, Figure B.3, Figure B.4, and Figure B.5. All the nine renormalization conditions two consecutive bypass bin may undergo are represented in the examples. Therefore, one may observe how the interest variables behave. The mentioned figures organization is as follows for all of them:

i. *Low* and *Range* initial values appear at the top.

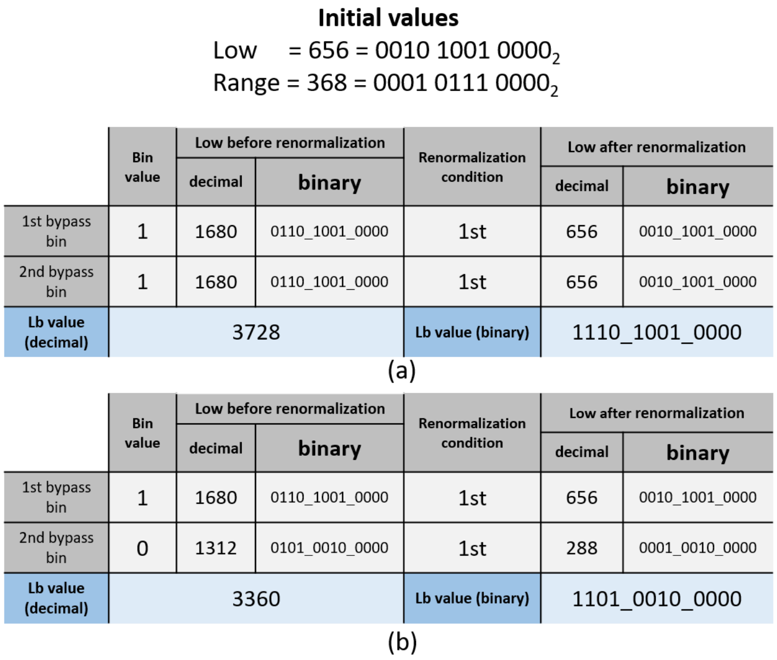
ii. *Low* undergoes the behavior according to what condition related to (3.4) and Figure 5.1 they fall, for each bypass bin separately (second and third lines in each figure). The *Low* values before and after renormalization for each bypass bin are shown on the left and right of each image, respectively.

iii. At the bottom, the *Lb* value for the two bypass bins by using MBBS, in decimal and binary (notice that *Lb* must be 12-bits wide for two bypass bins).

In Figure B.1, *Low* starts below its upper-bound value (i.e., 656) and *Range* has the value 368. Therefore, the conditions in (5.4) and (5.5) are respected. In Figure B.1(a), two bypass bins with value ‘1’ happen. For the first, the 1st renormalization condition happens, as also for the second bypass bin. The MBBS approach would lead to an *Lb* with both *Lb[11]* and *Lb[10]* having value ‘1’, and comparing the *Lb* value with the final renormalized *Low* after the second bypass bin, the *Lb[9]* shall remain with its value unaltered. In Figure B.1(b), starting with the same initial *Low* and *Range*, but now, at first, a bypass bin with value ‘1’ occurs, followed by a bypass bin with value ‘0’. Therefore, for both bypass bins, the required renormalization condition is the 1st. By using MBBS, the *Lb[11]* and *Lb[10]* also have the value ‘1’, and the *Lb[9]* must maintain the value ‘0’ when compared to the *Low* value after the second occurring bypass bin.

In Figure B.2, the *Low* and *Range* initial values are 510, which respects (5.3), (5.4), and (5.5). In Figure B.2(a), at first a bypass bin with value ‘0’ occurs, followed by a bypass bin with value ‘1’. This situation presented leads to, at first, the 3rd renormalization condition, followed by the 1st renormalization condition. By using MBBS, *Lb[9]* would have value ‘0’, with *Lb[11]* and *Lb[10]* having different values. *Lb[9]* would remain unchanged with ‘0’ as the final renormalized *Low*. Figure B.2(b) shows the update of *Low* when a bypass with value ‘1’ and with value ‘0’ occur, respectively. At first, for the first bin, the 1st renormalization condition is required. The next bin will require the 3rd renormalization condition. One may notice that *Lb[10]* and *Lb[9]* for the MBBS approach would have the values ‘0’ and ‘1’, respectively. Furthermore, the *Lb[9]* should be zeroed to be according to the correct *Low* value after the renormalizations.

Figure B.1 – The first example of MBBS description



Initial values

Low = 656 = 0010 1001 0000$_2$

Range = 368 = 0001 0111 0000$_2$

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	1	1680	0110_1001_0000	1st	656	0010_1001_0000
2nd bypass bin	1	1680	0110_1001_0000	1st	656	0010_1001_0000
Lb value (decimal)	3728			**Lb value (binary)**	1110_1001_0000	

(a)

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	1	1680	0110_1001_0000	1st	656	0010_1001_0000
2nd bypass bin	0	1312	0101_0010_0000	1st	288	0001_0010_0000
Lb value (decimal)	3360			**Lb value (binary)**	1101_0010_0000	

(b)

Source: the author.

Figure B.3 presents *Low* and *Range* with initial values 382 and 510, respectively. In Figure B.3(a), a bypass bin ‘0’ is followed by a bypass bin ‘1’. This situation leads to the 3rd renormalization condition occurring twice for the bypass bins. The *Lb* value would have the *Lb[9]* with value ‘1’, requiring to be zeroed to achieve the correct final *Low* value, whereas *Lb[11]* and *Lb[10]* would have respectively ‘0’ and ‘1’. In Figure B.3(b), a bypass bin with value ‘1’ followed by a bypass bin with value ‘0’ happen. Thus, this situation requires the 1st, followed by the 2nd renormalization condition. By using MBBS, the *Lb[9]* has the value ‘0’, whereas *Lb[11]* and *Lb[10]* have respectively ‘1’ and ‘0’. *Lb[9]* shall remain with the value zero for the correct renormalization.

Figure B.2 – The second example of MBBS description

Initial values

Low = 510 = 0001 1111 1110_2
Range = 510 = 0001 1111 1110_2

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	0	1020	0011_1111_1100	3rd	508	0001_1111_1100
2nd bypass bin	1	1526	0101_1111_0110	1st	502	0001_1111_0110
Lb value (decimal)	2550			**Lb value (binary)**	1001_1111_0110	

(a)

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	1	1530	0101_1111_1010	1st	506	0001_1111_1010
2nd bypass bin	0	1012	0011_1111_0100	3rd	500	0001_1111_0100
Lb value (decimal)	3060			**Lb value (binary)**	1011_1111_0100	

(b)

Source: the author.

Figure B,4 depicts the situation where the *Low* and *Range* variables have respectively as initial values 254 and 510. In case two bypass bins with value ‘0’ occur in a row, Figure B.4(a) shows that this would lead to the 2nd and 3rd renormalization conditions to happen in the mentioned sequence. The MBBS approach leads to *Lb[10]* and *Lb[9]* to have the values ‘0’ and ‘1’, respectively, where the *Lb[9]* shall be zeroed to keep the *Low* with the correct value. In Figure B.4(b), one may notice that a bypass bin ‘0’ followed by a bypass bin ‘1’ occur. Therefore, the renormalization conditions needed are the 2nd and the 1st, in this order of occurrence. The usage of MBBS directly leads to a situation already presented for other examples, where the *Lb[9]* has value ‘0’, whereas *Lb[11]* and *Lb[10]* have different values (i.e., ‘0’ and ‘1’, in that order). *Lb[9]* has to remain with the value zero for the correct renormalized *Low* value.

Finally, in Figure B.5, *Low* and *Range* begin with values 126 and 510, respectively. In case two bypass bins with value ‘0’ occur in a row, the 2nd renormalization condition is required once for each bin, as depicted in Figure B.5(a). The MBBS approach would lead to having *Lb[9]* with value ‘0’, and *Lb[11]* and *Lb[10]* with the same value (i.e., with value ‘0’). The *Lb[9]* shall be kept with the same value to be according to the correct *Low* after both renormalizations. Figure B.5(b) presents what happens when a bypass bin with value ‘1’ and ‘0’ occur in a row, leading to the requirement of the 3rd and 2nd renormalization conditions to occur, in that order. By using MBBS, the *Lb[9]* would have value ‘0’, whereas *Lb[11]* and

Lb[10] respectively ‘0’ and ‘1’ (i.e., different values). Again, *Lb[9]* shall remain with the value ‘0’ to be correct for the next occurring bin.

Figure B.3 – The third example of MBBS description

Initial values

Low = 382 = 0001 0111 1110$_2$

Range = 510 = 0001 1111 1110$_2$

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	0	764	0010_1111_1100	3rd	252	0000_1111_1100
2nd bypass bin	1	1014	0011_1111_0110	3rd	502	0001_1111_0110
Lb value (decimal)	2038			**Lb value (binary)**	0111_1111_0110	

(a)

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	1	1274	0100_1111_1010	1st	250	0000_1111_1010
2nd bypass bin	0	500	0001_1111_0100	2nd	500	0001_1111_0100
Lb value (decimal)	2548			**Lb value (binary)**	1001_1111_0100	

(b)

Source: the author.

Figure B.4 – The fourth example of MBBS description

Initial values

Low = 254 = 0000 1111 1110$_2$

Range = 510 = 0001 1111 1110$_2$

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	0	508	0001_1111_1100	2nd	508	0001_1111_1100
2nd bypass bin	0	1016	0011_1111_1000	3rd	504	0001_1111_1000
Lb value (decimal)	1016			**Lb value (binary)**	0011_1111_1000	

(a)

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	0	508	0001_1111_1100	2nd	508	0001_1111_1100
2nd bypass bin	1	1526	0101_1111_0110	1st	502	0001_1111_0110
Lb value (decimal)	1526			**Lb value (binary)**	0101_1111_0110	

(b)

Source: the author.

Figure B.5 – The fifth example of MBBS description

Initial values

Low = 126 = 0000 0111 1110_2

Range = 510 = 0001 1111 1110_2

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	0	252	0000_1111_1100	2nd	252	0000_1111_1100
2nd bypass bin	0	504	0001_1111_1000	2nd	504	0001_1111_1000
Lb value (decimal)	504			Lb value (binary)	0001_1111_1000	

(a)

	Bin value	Low before renormalization		Renormalization condition	Low after renormalization	
		decimal	binary		decimal	binary
1st bypass bin	1	762	0010_1111_1010	3rd	250	0000_1111_1010
2nd bypass bin	0	250	0000_1111_1010	2nd	500	0001_1111_0100
Lb value (decimal)	1524			Lb value (binary)	0101_1111_0100	

(b)

Source: the author.

Hence, there is a pattern based on the values of *Lb[11]*, *Lb[10]*, and *Lb[9]* to deliberate the final renormalized *Low*, the updated OB, and the bitstream generated for the MBBS. The pseudo-codes in Figure B.6, Figure B.7, and Figure B,8 presents the MBBS update for *Low*, OB, and bitstream generation, respectively. The *bp* variable indicates when the MBBS processes one or two bypass bins. The description of each pseudo-code appears below:

i. If *Lb[11]* and *Lb[10]* have value '1' and two bypass bins undergo the processing (refer to Figure 5.1), or if *Lb[10]* has value '1' and only one bypass bin is being processed (LIU, 2011b), *Lb[9]* shall keep the current value (lines 2-3 in Figure B.6). Otherwise, the *Lb[9]* shall be zeroed (lines 4-5 in Figure B.6). The other bits are a copy of *Lb* remaining bits (line 1 in Figure B.6).

ii. For two bypass bins (line 7 in Figure B.7), the OB could have three different values. (i) value '0' (i.e., 1st, 2nd, or 3rd renormalization condition followed by the 1st or 2nd renormalization condition – lines 12-13 in Figure B.7 – refer to Figure B.1, Figure B.2(a), Figure B.3(b), Figure B.4(b), and Figure B.5). (ii) The original OB value added by two (i.e., 3rd renormalization condition occurring twice – lines 10-11 in Figure B.7 – refer to Figure B.3(a)); (iii) value 1 (i.e., 1st or 2nd renormalization condition followed by the 3rd renormalization condition – lines 8-9 in Figure B.7 –

refer to Figure B.2(b), and Figure B.4(a)). For one bypass bin, OB can be incremented by one (line 2-3 in Figure B.7), or zeroed (lines 4-5 in Figure B.7) (LIU, 2011b).

iii. The symbols $\sim Lb[11]^{OB}$ and $\sim Lb[10]^{OB}$ in Figure B.8 mean the negation of the referred *Lb* bit, replicated OB times, whereas the brackets indicate a concatenation of the values. The symbol *null* means no bitstream generation. (i) When *Lb[9]* has the value ‘0’, and *Lb[11]* and *Lb[10]* are different, that indicates that at least for the second bypass bin, the 1st or the 2nd renormalization conditions have occurred. Any renormalization conditions may have occurred for the first bypass bin, except the as for the second bin (refer to Figure B.2(a), Figure B.3(b), Figure B.4(b), and Figure B.5(b)). Hence, this will lead to the 1st bitstream generation behavior (lines 2-3 in Figure B.8). (ii) When *Lb[11]*, *Lb[10]*, and *Lb[9]* have all the value ‘0’; or if *Lb[11]* and *Lb[10]* have both value ‘1’, that implies that the 1st or the 2nd renormalization condition have occurred twice (refer to Figure B.1, and Figure B.5(a)), leading to the 2nd bitstream generation behavior (lines 4-5 in Figure B.8). (iii) If *Lb[10:9]* have the value ‘01’, this is the case that the second bypass bin required the 3rd renormalization condition, whereas the first bypass bin required either the 1st or the 2nd condition (refer to Figure B.2(b), and Figure B.4(a)), leading to the 3rd bitstream generation condition (lines 6-7 in Figure B.8). (iv) Finally, all other *Lb[11]*, *Lb[10]*, and *Lb[9]* combinations of values, the 3rd renormalization condition has happened for the two bypass bins (refer to Figure B.3(a)), and therefore no bitstream is generated at this situation (lines 8-9 in Figure B.8). For a single bypass bin, either *Lb[10]* has the value ‘1’, or *Lb[10]* and *Lb[9]* have both the value ‘0’, and therefore the 4th bitstream generation condition occurs (LIU, 2011b) (lines 12-13 in Figure B.8). For any other situation for a single bypass bin, no bitstream is generated (lines 14-15 in Figure B.8).

Figure B.6 – Low renormalization for MBBS

```
1. Low[8:0] ← Lb[8:0];
2. if (bp=2 and Lb[11:10] = '11') or (bp =1 and Lb[10] = '1') then
3.     Low[9] ← Lb[9];
4. else
5.     Low[9] ← 0;
```

Source: the author.

Figure B.7 – OB update for MBBS

```
if bp = 1 then
    if Lb[10:9] = 01 then
        OB $\leftarrow$ OB + 1;
    else
        OB $\leftarrow$ 0;
    end if
else if bp = 2 then
    if Lb[10:9] = ‘01’ then
        OB $\leftarrow$ 1;
    else if Lb[11:9] = ‘011’ then
        OB $\leftarrow$ OB + 2;
    else
        OB $\leftarrow$ 0;
    end if
else
    OB $\leftarrow$ OB;
end if
```

Source: the author.

Figure B.8 – Bitstream generation for MBBS

```
if bp = 2 then
    if (Lb[9] = ‘0’ and Lb[11] $\neq$ Lb[10]) then
        bitstream $\leftarrow$ {Lb[11:10], $\sim$Lb[11]$^{OB}$};
    else if (Lb[11:9] = ‘000’) or (Lb[11:10] = ‘11’) then
        bitstream $\leftarrow$ {Lb[11], $\sim$Lb[11]$^{OB}$, Lb[10]};
    else if Lb[10:9] = ‘01’ then
        bitstream $\leftarrow$ {Lb[11], $\sim$Lb[11]$^{OB}$};
    else
        bitstream $\leftarrow$ null;
    end if
else if bp = 1 then
    if (Lb[10] = ‘1’) or (Lb[10:9] = ‘00’) then
        bitstream $\leftarrow$ {Lb[10], $\sim$Lb[10]$^{OB}$};
    else
        bitstream $\leftarrow$ null;
    end if
else
    bitstream $\leftarrow$ null;
end if
```

Source: the author.

APPENDIX C <HEVC RESIDUAL SYNTAX ELEMENTS GENERATION FOR HIGH-THROUGHPUT CABAC DESIGNS>

This appendix purpose is to give a brief explanation on the graduate work of Alessandro Via Piana Saggiorato, advised by the author of this Thesis, in which the context is strictly related to this Thesis research. The original and complete work can be found in Portuguese in (SAGGIORATO, 2017), and the subsequent publication in the IEEE International Conference on Electronics, Circuits, and Systems (ICECS) 2018 (SAGGIORATO, 2018) which was awarded as the Best Student Paper of the conference.

C.1 Motivation

This Thesis has proposed high-throughput architectures and approaches for the Binary Arithmetic Encoder (BAE) block, the critical one from CABAC. Moreover, other alternatives on the literatures also achieve significant performance in terms of bins per cycle and bins per second throughput, in order to accomplish the constraints for real-time processing of UHD videos. Nevertheless, no consideration is taken related to the input rate generation of the Syntax Elements (SEs) that the CABAC block consume. Therefore, there is no guarantee that the improvements in terms of performance of these state-of-the-art CABAC architectures will succeeded effectively, in case the input fed to the entropy encoding does not match a rate that allows these designs a full-throttle processing. The research here presented analyses which are the SEs that are the majority among them all, along with the contribution in bins generation. Statistical and performance analysis using the HEVC HM reference software is presented, leading to the namely Multiple Residual Syntax Element Generation Treatment (MRSET). Finally, an architectural approach for the MRSET, dealing with the dependencies each residual SEs has among them during production. Five versions of the MRSET are presented, along with an architectural fashion to deal with some preliminary resisual SEs, named Prior Residual Syntax Element Generation Treatment (PRSET). Finally, synthesis results and comparisons with related recent CABAC works drives the discussion to whether MRSET version is the best suitable choice of use.

C.2 HEVC Syntax Elements Analysis

An analysis of the HEVC SEs is presented, based on statistical and analytical data from the HEVC reference software. The initial software version of the MRSET proposal appear in the end, with execution time comparisons against the original HM version.

C.2.1 Major SEs Contributors for CABAC processing

As already cited throughout this Thesis, the input data of the CABAC block are the SEs, which come from all preliminary stages on the HEVC processing flow. One question that comes is which are the SE category that contributes the most to the input data of CABAC step.

As stated in (STANKOWSKI, 2014), transformed data from luminance, followed by transformed data from chrominance, represent the vast majority of SEs, for any range of QP values for 1920x1080 video sequences. Table C.1 presents a summary of these results, categorized by the four most frequent category of SEs (i.e., transformed data for luma, transformed data for chroma, Intra/Inter Prediction, and Merge Index), and discriminated within the different types of frames in a Group of Pictures (GoP). For each type of frame, the percentages are presented for the minimum and maximum recommended Quantization Parameter (QP) values for analysis (BOSSEN, 2013). For the sake of clarity the I-type frames referred to Intra-prediction, whereas the B-type to Inter-prediciton (ITU-T, 2013).

Table C.1 – Main syntax elements proportion among different types of HEVC frames

	Frame Type									
	I		**B0**		**B1**		**B2**		**B3**	
SE / QP	**22**	**37**	**22**	**37**	**22**	**37**	**22**	**37**	**22**	**37**
Transform Luma	66%	58%	67%	56%	69%	62%	69%	65%	75%	60%
Transform Chroma	17%	14%	15%	7%	13%	-	13%	-	6%	-
Intra/Inter Prediction	8%	12%	7%	17%	7%	24%	8%	19%	9%	16%
Merge Index	-	-	-	-	-	-	-	8%	-	10%

Source: the author, adapted from (STANKOWSKI, 2014).

As one may conclude by looking to Table C.1, the transformed luma corresponds to 56% - 75% of the total SEs, whereas transformed chroma to 6% - 17% of total SEs. Table C.2, adapted from (SZE, 2014) shows the bins percentage of the major contributors for bins generation, according the level of hierarchy of HEVC standard, using the Commom Test Condition (i.e., AI – All-Intra; LD-B – Low-Delay B; LD-P – Low-Delay P; and RA –

Random Access), along with the namely wort-case defined by the authors (i.e., maximum number of bins per 16x16 CTU). The TU correspond to the transformed data already mentioned, and it is clear that at the Transform level that the bins generation has the maximum impact (i.e., ranging from 63.7% to 94% of total bins contributions). The elements from the Transform step are also called residual SEs, which will be explained in the sequence. The conclusion is that data coming from the Transform step are the primary candidate for assessment, considering the scope presented.

Table C.2 – Major bins contributors among HEVC data hierarchy

	Common Test Conditions				
Hierarchy Level	**AI**	**LD-P**	**LD-B**	**RA**	**Worst case**
CTU/CU bins	5.4%	15.8%	16.7%	11.7%	1.4%
PU bins	9.2 %	20.6%	19.5%	18.8%	5.0%
TU bins	85.4%	63.7%	63.8%	69.4%	94.0%

Source: (SZE, 2014).

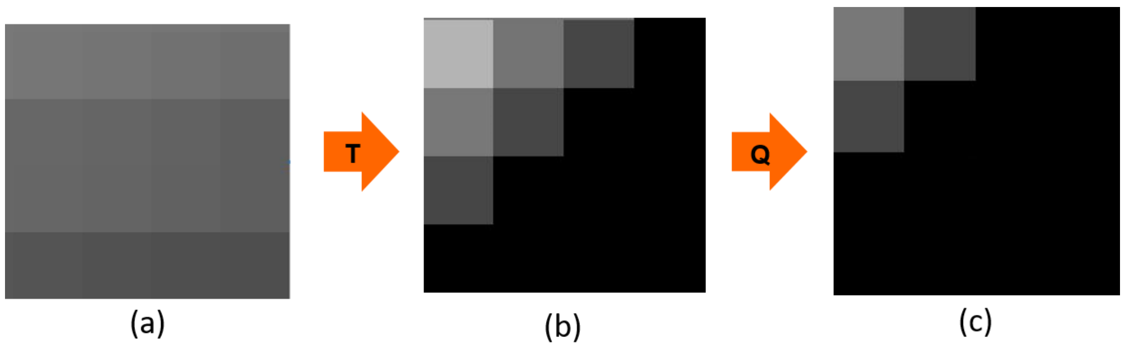
C.2.2 Residual SEs Processing

The input of the Transform step are the so-called residue from the prior predictions. During the prediction process, current groups of pixels are compared with already processed pixels in the flow. The already processed group of pixels used for comparison are chosen by the prediction stage, in a fashion that they have the most similar value to the current processed ones. Therefore, by decremented the current pixels by the chosen predicted pixels, the residual value tends to be the minimal possible. This difference is the definition of a residue on the HEVC context. The pixels undergo prediction by using pixel from the same frame they are (i.e., Intra-prediction), or from another frame (Inter-prediction).

A reorganization of the residues occurs in the Transform step, in a fashion similar to what is presented in Figure C.1. In Figure C.1(a), an example group of residues is presented., whereas in Figure C.1(b) the same residues after transformation appear. The blackest is the transformed residue the smaller is its value after Transformation. After transformation, the residue may undergo Quantization, which corresponds to an integer division of the transformed residues. Therefore, the higher the QP, the more losses the encoding process suffer, and the smaller will be the residues after this process, along with decrease quality of the encoded video sequence. Figure C.1(c) presents the Quantization for the same group of pixels already mentioned. Transformed residues (also called coefficients) that have an absolute value different from zero are called significant coefficients/transformed residues, whereas the ones that have exactly zero as value are named non-significant coefficients/transformed residues.

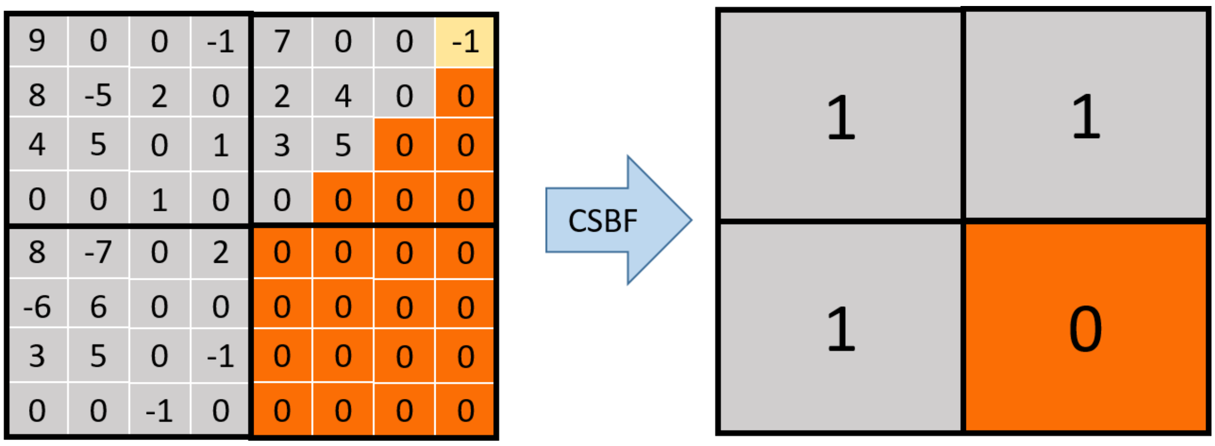
From this point, the residual SEs generation takes place. One may notice that this generation is a mandatory step of the HEVC standard, which, therefore, implies that this logic is present in every HEVC encoder solution (hardware or software).The granularity of this processing is in 4x4 block of coefficients. Even if a TU can be as higher as 32x32 residues, it has to suffer a division to output only 4x4 blocks for the residual SEs fabrication. Thus, the first residual SE is the *coded_sub_block_flag* **(CSBF),** which indicates, within the TU, which 4x4 blocks actually has at least one significant coefficients, or if a given 4x4 block has all zero coefficients. For instance, Figure C.2 shows an example of CSBF signaling for an 8x8 TU, where 4x4 blocks with all zeroed transformed residues have a CSBF with value ‘0’, whereas the ones with at least one significant coefficient receive a CSBF with value ‘1’ .

Figure C.1 – Generic 4x4 group of residues undergoing Transform and Quantization



Source: the author.

Figure C.2 – Example of CSBF signaling



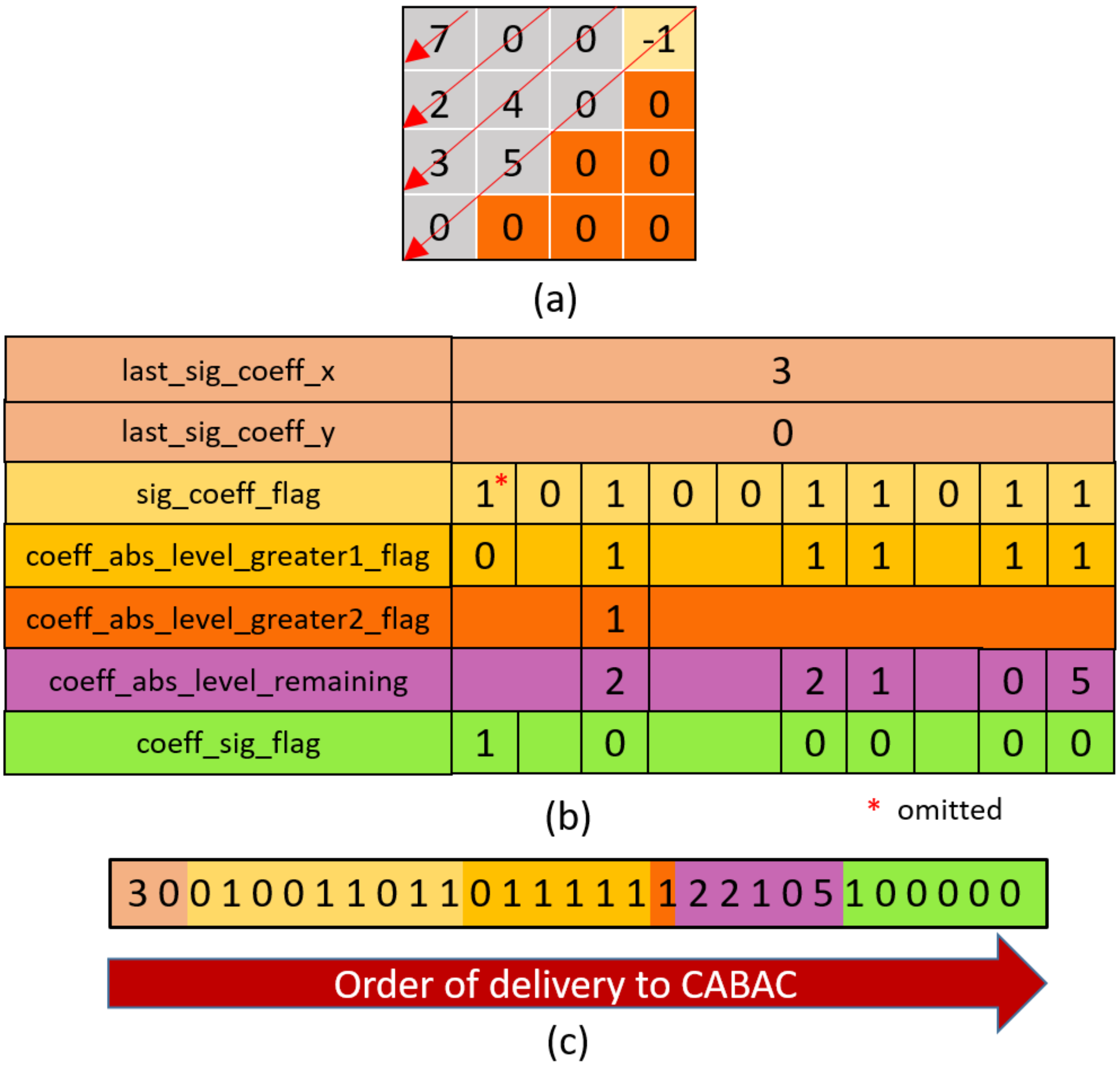
9	0	0	-1	7	0	0	-1
8	-5	2	0	2	4	0	0
4	5	0	1	3	5	0	0
0	0	1	0	0	0	0	0
8	-7	0	2	0	0	0	0
-6	6	0	0	0	0	0	0
3	5	0	-1	0	0	0	0
0	0	-1	0	0	0	0	0

1	1
1	0

Source: the author.

After the CSBF generation, the other residual SEs production can happen, within the 4x4 blocks with CSBF with value ‘1’. Figure C.3 shows an example, which will guide the explanation of the other SEs. In Figure C.3(a), a generic 4x4 block is presented, considering that it is the first one to be processed in a given TU of size above 4x4. The arrows shows the coefficients order of processing. Figure C.3(b) shows the values of the residual SEs for each of the transformed residues, whereas Figure C.3(c) presents the mandatory SE order of delivery to the entropy encoding. The explanation of each residual SE comes as follows:

Figure C.3 – Example of residual SEs generation for generic 4x4 block of coefficients



last_sig_coeff_x	3									
last_sig_coeff_y	0									
sig_coeff_flag	1*	0	1	0	0	1	1	0	1	1
coeff_abs_level_greater1_flag	0		1			1	1		1	1
coeff_abs_level_greater2_flag			1							
coeff_abs_level_remaining			2			2	1		0	5
coeff_sig_flag	1		0			0	0		0	0

Source: (SAGGIORATO, 2018).

- *last_sig_coeff_x* and *last_sig_coeff_y* **(LAST)**: this SE represents the position for the axis x and y of the last significant coefficient within the TU. Respecting the reading order of the HEVC standard (ITU-T, 2013), within the first 4x4 block with CSBF equal to ‘1’, the first significant coefficient also has to be found. For instance, considering the example in Figure C.2, the top-right 4x4 block will contain the LAST. Inside that block, the reading order of the Figure C.3(a) is followed, starting by the zero transformed residues at the bottom-right, until the first significant coefficient is found (i.e., the '-1' value on the Figure C.3(a)). Hence, the relative position considering the axis x and y is signaled as the LAST for this TU.

- *sig_coeff_flag* **(SIG)**: this SE indicates whether the coefficient is significant yes or no, considering only the coefficients that occur before the LAST.
- *coeff_abs_level_greater1_flag* **(COEFF1)**: for every transformed residue which is also significant, up to a certain number (maximum of eight of this SE per 4x4 block), if its absolute value is greater than one, yes or no.
- *coeff_abs_level_greater2_flag* **(COEFF2)**: for a single transformed residue for every 4x4 block within the TU, in case its absolute value is greater than one, if it is also greater than two.
- *coeff_sign_flag* **(SIGN)**: for every significant coefficient, if it positive of negative.
- *coeff_abs_level_remaining* **(REM)**: all significant transformed residues shall undergo a decrement by a certain base value, depending if they have a valid associated COEFF1 or/and COEFF2 to them. The result of this decrement is the associated REM.

C.2.3 HEVC Reference Software Analysis

The HEVC HM Reference Software (HM, 2016) can provide the first clues towards data to support an efficient generation of the residual SEs. For that purpose, the usage of recommended test video sequences (BOSSEN, 2013) in order to gather statistics involving the execution time, number of significant coefficients, and amount of residual SEs generated, appears in Table C.3, for the Low-Delay configuration, using different QP values. At this moment, the QPs used are different from the ones recommended by (BOSSEN, 2013), because of the willingness to observe the results in a wider range of QPs, when compared to the ones recommended. The chosen video sequences are from HD resolutions, which will illustrate the amount of data derived from residual coding in the scenario where the high-throughput CABAC designs are most suitable. That is main reason to limit that preliminary assessment to the sequences selected.

As a conclusion, the lower is the QP value, the higher the execution time, the higher the amount of significant coefficients, and, as a consequence, the higher the number of residual SEs. Another remark is that higher quality video (i.e., sequences with low QP value) tend to have remarkably more residual SEs than lower quality sequences. This argument reinforces the necessity of efficient ways to produce this SEs, because of the increasing demand for high quality video definition.

Observations of the HM operational part show that it generate the residual SEs in the following fashion: every residual SE of a given type is produced at once, and then all other residual SEs of the next required type, and so on. Using an approximate depiction, it would look similar to what appears in Figure A.4, considering an RTL approach, and using the same 4x4 block that is in Figure C.3. What one may notice is that this fashion is compliant with the demanded input data fed of recent CABAC works. For instance, the first SEs generated are the SIG, which are flags. Every residual SE whose name ends with “flag” will produce a single bin during the Binarization process of CABAC. Therefore, considering the example in Figure C.4, nine bins are produced (the first SE of this type is ignored), and this value is enough to supply data to a given entropy encoding architecture for around the first two clock cycles. On the second cycle of residual SEs fabrication, more six bins are produced (i.e., COEFF1 is also a flag). Therefore, there is a tendency to fulfill the requirements of input data for the entropy encoder.

Table C.3 – Videos characterization for residual SEs generation

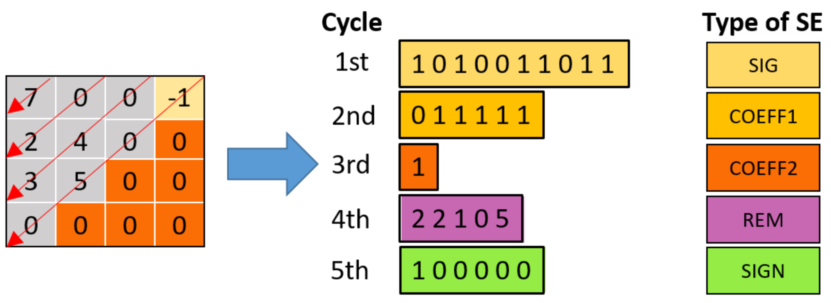
Videos	QP	Execution Time (s)	Significant Coefficients	Residual SEs
PeopleOnStreet	1	2954.8	$3.89x10^9$	$1.10x10^{10}$
	30	580.6	$1.03x10^8$	$3.34x10^8$
	50	103.1	$2.49x10^6$	$9.43x10^6$
BastekballDrive	1	1435.7	$2.15x10^9$	$5.96x10^9$
	30	662.4	$9.28x10^7$	$3.90x10^8$
	50	199.8	$5.47x10^6$	$3.02x10^7$
BQTerrace	1	1522.3	$1.99x10^9$	$5.70x10^9$
	30	81.7	$2.56x10^7$	$9.03x10^7$
	50	16.7	$6.44x10^5$	$2.85x10^6$
Kimono	1	1513.8	$2.06x10^9$	$5.75x10^9$
	30	303.8	$1.95x10^7$	$6.58x10^7$
	50	22.33	$2.97x10^5$	$1.14x10^6$

Source: the author.

The drawback of the HM original method is that, for every new type of residual SEs fabrication, the same memory positions where the coefficients are stored have to accessed over and over again for every new type of SEs, which, regarding a hardware approach, would certainly lead to waste of power dissipation and penalty in performance.

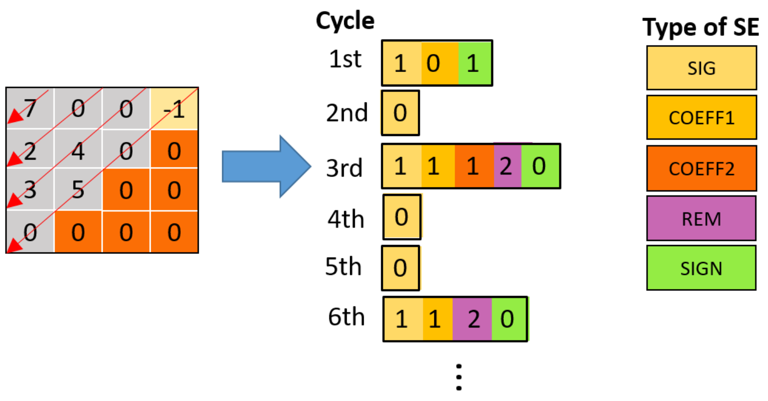
Another way to process the residual SEs is depicted in Figure C.5, named here as Memory-Access Efficient Approach (MAE), where basically every memory position of the stored coefficients is accessed just once per 4x4 block, and all associated residual SEs to that position (i.e., to that coefficient) is generated per clock cycle. The advantage is the removal of degradation in performance and power consumption of the multiple accesses to the same memory positions. Nevertheless, the MAE fashion leads to the problem of starving any recent entropy encoder architecture that is supplied by it. For instance, looking back to Figure C.5, for the first transformed residue visited, all the related residual SEs to it are generated. However, a single SIG can then be sent to the entropy encoder at that moment, which will generate a single bin, as already explained. Only at the second cycle of residual SEs generation, that a new SIG is produced and sent to the CABAC. As one may see, a single bin per clock cycle is no enough to supply the recent CABAC designs to achieve their potential maximum throughput. Thus, they would starve of incoming data and any improvement focusing a hardware entropy encoding design would be fruitless, for a MAE approach.

Figure C.4 – Residual SEs generation for the original HM approach



Source: the author.

Figure C.5 – Residual SEs generation for the MAE approach

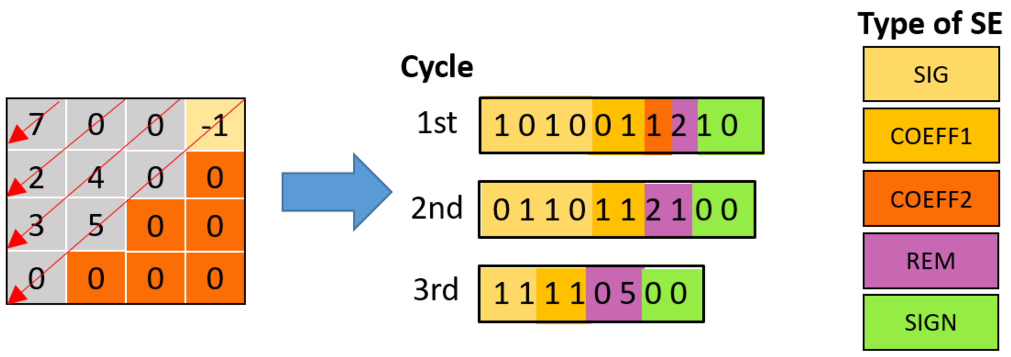


Source: the author.

The efficient fashion to generate the residual SEs is, therefore, to merge the advantages of both HM original method and ME approach, named here as the Multiple Residual Syntax Element Treatment (MRSET), and which functionality is depicted in Figure C.6. In the MRSET, multiple memory positions are visited at the same time and thus, multiple coefficients (following what is originally proposed in the HM). For these visited transformed residues, all related residual SEs are generated. As shown in Figure C.6, the MRSET approach for that example considerers that four coefficients are accessed at once, and the related SEs are fabricated. As one may see, at the first clock cycle, at least four SIGs are generated, which is compliant which the average bin per cycle throughput of recent CABAC designs. At the second clock cycle, more four SIGs occurred and this situation provides sufficient bins for the entropy encoder. Finally, at the third cycle, all residual SEs are produced, and therefore enough input data is guaranteed until the end of the processing of the example 4x4 block.

It is important to notice that, any of the three cited approaches to generate the residual SEs may exist within a given HEVC encoder solution (since this step is part of the standard). Thus, that drives the reason to choose the most efficient fashion to deliver the data to the entropy-encoder block.

Figure C.6 – Residual SEs generation for the MRSET approach



Source: the author.

C.3 Preliminary MRSET Software Version Analysis

Potential gains of the MRSET can be provided by modifying the original HM fashion in a way that would look like the MRSET approach. A software version would not behave exactly like a hardware design, but can point out preliminary gains estimative. For that reason, the original HM was modified, eliminating the exclusive loops for every SEs type, and substituting them by a single loop, which generates all the residual SEs at once. This

MRSET version was embedded in the HM version 16.1 (HM, 2016), and run against the original unmodified HM. A machine with an Intel i5® running on 3.1 GHz and 8-GB of RAM was used for that analysis. The same video sequences were used as the ones presented in Table C.3, by the same reasons: the high-throughput CABAC solutions targets high demanding constrains, which one may see associated exactly with higher resolution video sequences

The comparison between the original HM against the modified version was performed for ten thousand, one hundred thousand, one million and ten millions blocks of transformed residues, using again as QPs value 1, 30, and 50, for some recommended video sequences. These blocks amounts were chosen due to memory limitations of the machine available to process the analysis. Again, the QPs used are not the ones recommended in (BOSSEN, 2013) to verify a broader difference for the results in a wider variation of QP values.

Table C.4 depicts the execution times (in seconds) for both version (the original HM on the left, and the modified version with MRSET on the left). For the MRSET version, all the results which appear in black are the same as the original version, all the results in green are better than the HM method, whereas all the results in red are worse than the unmodified HM. As one may seem, the MRSET software preliminary version has better execution time for the majority of the cases assessed. The bigger the amount of blocks, the better tend to be the results, as shown in Table C.4. For the sake of explanation, the higher the QP, the more time the video sequences tend to provide 4x4 blocks with at least one significant coefficient, and that is the reason higher values QP sequences last longer to execute.

Table C.4 – Execution time for original HM method against modified HM with MRSET

		Execution Time (s)							
		Original HM				**Modified HM with MRSET**			
Videos	**blocks** / **QP**	$10x10^3$	$100x10^3$	$1x10^6$	$10x10^6$	$10x10^3$	$100x10^3$	$1x10^6$	$10x10^6$
PeopleOnStreet 2560x1600	1	**0.18**	**1.84**	**18.27**	**183.17**	**0.17**	**1.78**	**17.83**	**181.33**
	30	**0.55**	**3.43**	**29.71**	**403.75**	**0.55**	**3.38**	**29.85**	**399.35**
	50	**1.27**	**8.23**	**85.56**	**774.98**	**1.22**	**8.21**	**84.98**	**772.42**
BasketballDrive 1920x1080	1	**0.18**	**1.74**	**17.79**	**177.89**	**0.17**	**1.67**	**16.85**	**173.92**
	30	**0.68**	**4.22**	**38.97**	**253.91**	**0.67**	**4.18**	**37.17**	**249.94**
	50	**1.29**	**11.51**	**56.71**	**377.62**	**1.28**	**11.52**	**55.57**	**371.58**
BQTerrace 1920x1080	1	**0.17**	**1.82**	**17.21**	**179.51**	**0.17**	**1.76**	**16.91**	**170.75**
	30	**0.33**	**3.17**	**26.83**	**341.48**	**0.33**	**3.16**	**26.27**	**329.71**
	50	**0.76**	**7.22**	**81.41**	**965.61**	**0.75**	**7.15**	**81.95**	**957.49**
Kimono 1920x1080	1	**0.18**	**1.87**	**18.22**	**182.68**	**0.17**	**1.81**	**17.56**	**176.41**
	30	**0.36**	**2.81**	**31.17**	**490.25**	**0.35**	**2.76**	**30.56**	**473.96**
	50	**0.73**	**10.11**	**100.72**	**1087.15**	**0.73**	**10.13**	**99.24**	**1061.46**

Source: the author.

As one may estimate, the tendency is for an even increasing amount of blocks, the MRSET software performance would possibly improve. For example, the Kimono sequence on QP 30 has a slight improvement of 0.01s for ten thousand blocks, 0.05s for one hundred thousand blocks, 0.61 s for one million blocks, and 16.29s for ten million blocks (a behavior that seems to replicate for the other sequences). Another remark is that the improvements presented represent only the very beginning of each video sequence (refer to the data on Table C.3 for comparison), and thus the estimative is to have better execution times the further the sequence runs (which was not possibly in this analysis due to the machinery resources restriction). However, as a valid conclusion, the results point out the validity and potential of the MRSET, before moving to a hardware approach of the proposal.

C.4 MRSET Hardware Proposal

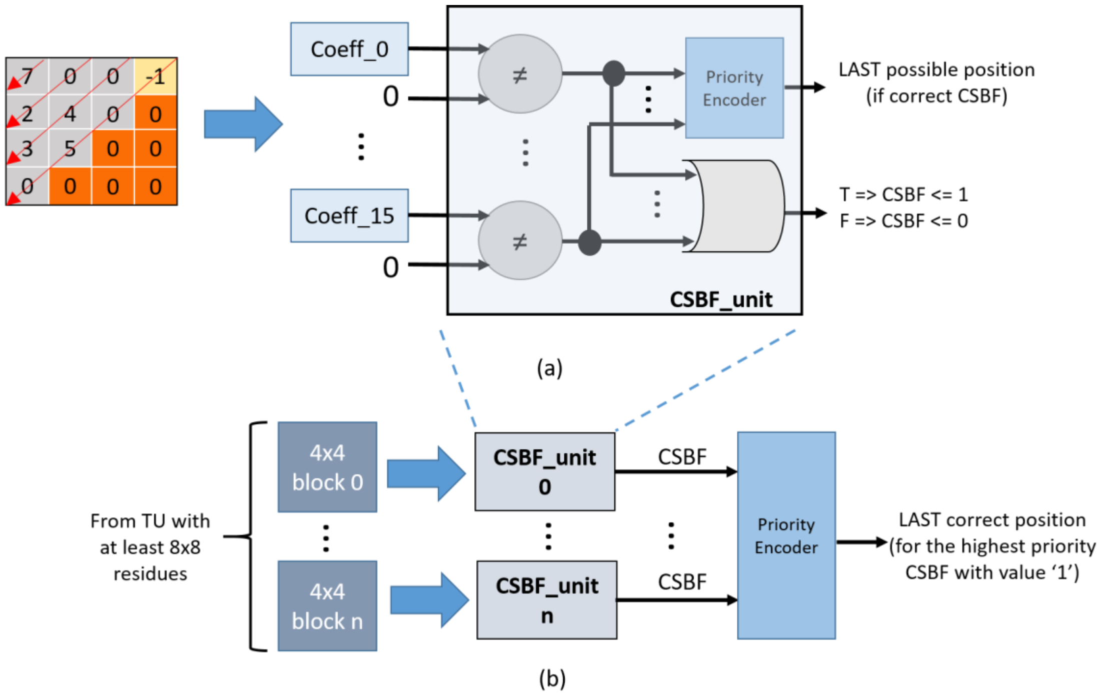
C.4.1 Prior Residual Syntax Elements Treatment (PRSET)

The rest of the residual SEs only occur for 4x4 blocks of transformed residues whose associated CSBF indicates that, at least, one significant coefficient has a non-zero value (i.e., CSBF has the value ‘1’). Moreover, only before the LAST that all other residual SEs can start to be produced. Thus, these two SEs have to be discovered prior to the others. There is also dependencies related to the Context Modeling of CABAC that required the CSBF and LAST values before the other residual SEs (ITU-T, 2013).

An efficient proposal in terms of performance to fabricate these two cited SEs is to use sixteen parallel comparator-wit-not-zero logic to every position of a given 4x4 block within the TU, and at the end group the results of the comparators in an OR logic, as shown in Figure C.7(a). Therefore, in one clock cycle, this circuitry is able to discover whether there is at least one significant coefficient within that block (i.e., the CSBF receives the value ‘1’) or all coefficients have value ‘0’ (i.e., the CSBF receives the value ‘0’). This architecture has to replicated up to the maximum number of 4x4 blocks a TU can support (the maximum TU processes 32x32 residues). Additionally, a priority encoder is appended to the comparators logic, to indicate the potential LAST (i.e., the priority indication comes from the required read order of the residues defined by the HEVC standard. The first ones have higher priority) (ITU-T, 2013), as depicted in Figure C.7(a) again. This block of logic receives the name *CSBF_unit*.

Every 4x4 block signaled with CSBF equal to ‘1’ will produce a potential LAST. Nevertheless, the highest priority block (i.e., again, the priority is defined by the reading order of this blocks, the read first has the highest priority, and so on) is where the correct LAST is. Hence, a new priority encoder is added to every *CSBF_unit*, and the LAST is also discovered, as depicted in Figure C.7(b). This whole circuit receives the name Prior Residual Syntax Element Treatment (PRSET). One may notice that, since the entire PRSET is combinational, all this process performs in a single clock cycle and is expected to not degrade the performance generation of the upcoming residual SEs.

Figure C.7 – Prior residual syntax element treatment logic



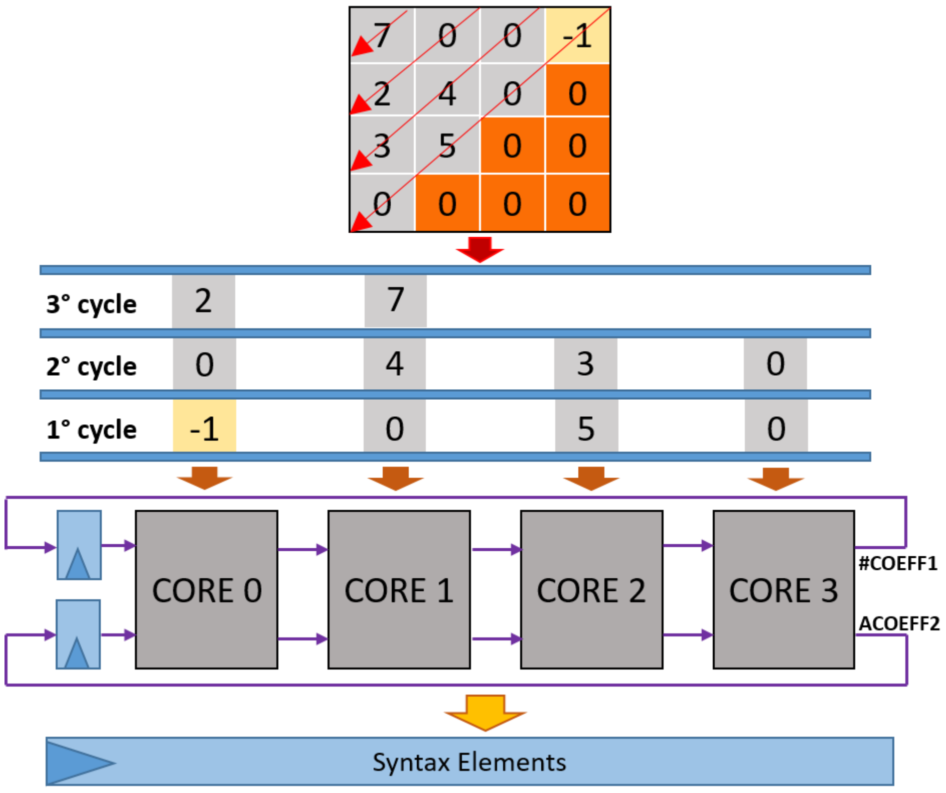
Source: the author.

C.4.2 MRSET Hardware Approach

Up to this point, the hardware proposition of the Multiple Residual Syntax Element Treatment (MRSET) can start its processing. As the name suggests, the MRSET is conceived as a multi-core hardware structure, each core able to receive and process a single coefficient, and to produce all the related residual SEs to this element. For instance, Figure C.8 shows an MRSET version with four cores, and how the timing delivery of the example 4x4 block would proceed for it. The registers #COEFF1 and ACOEFF2 are used for the storage of the amount of these elements already processed, and whose detailed explanation will come next.

One may conclude that, the more cores used, the more residual SEs fabrication occur at the same time. The trade-off between the number of cores and degradation in maximum frequency will drive the discussion to which is the best number of cores related to that matter in the next section.

Figure C.8 – Four cores MRSET behavior



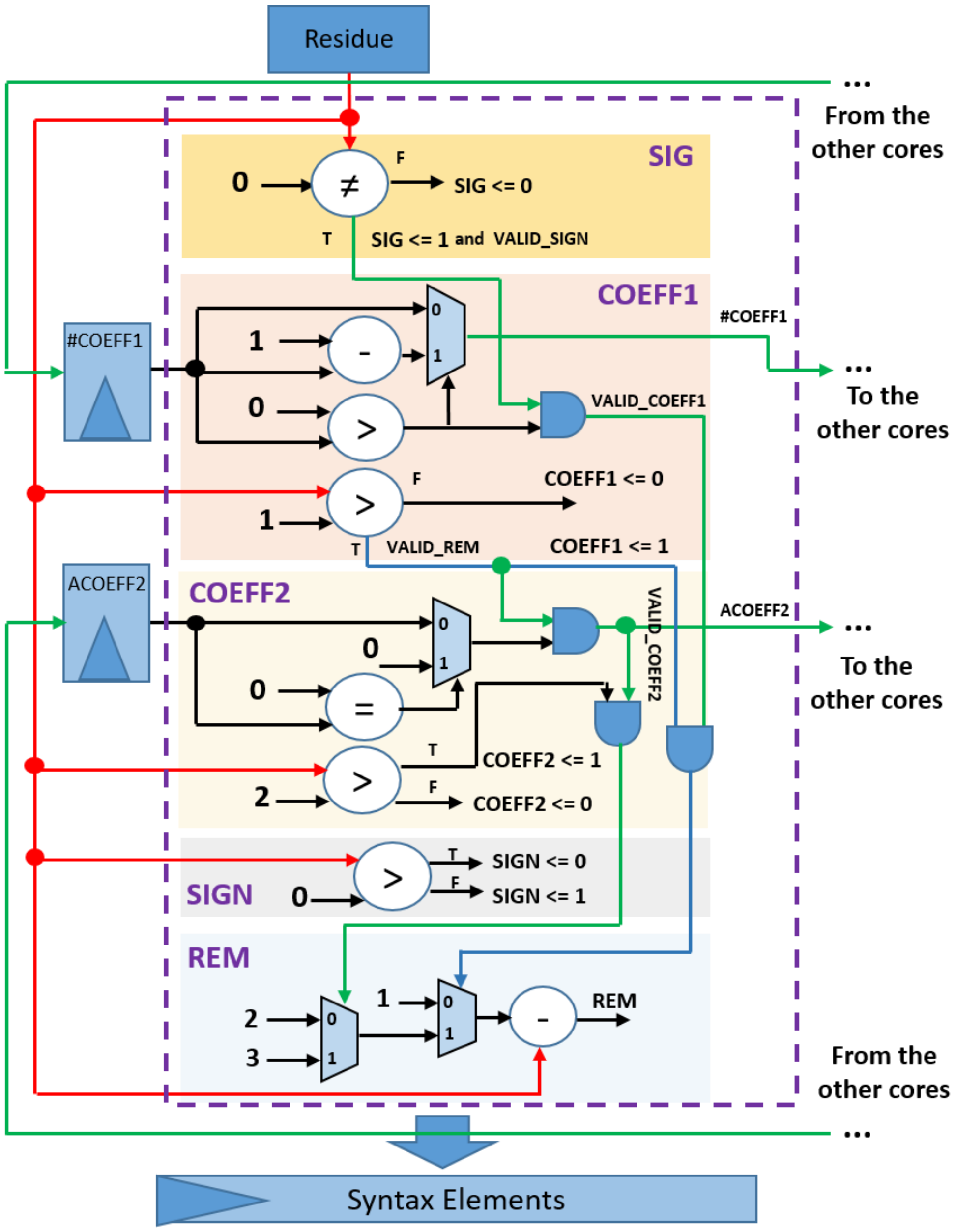
Source: (SAGGIORATO, 2018).

The crucial point in a multi-core MRSET is the ability to deal with the dependencies during the generation of the SEs. One may notice, based on the example on Figure C.3, that the last generated residual SEs may require information from the first generated. Moreover, some of these elements are of limited amount (i.e., the already cited #COEFF1 and ACOEFF2 are used for that purpose, respectively for the COEFF1 and COEFF2 elements). Table C.5 depicts the details related to the dependencies and limit amounts treatment among the residual SEs. The namely dependencies intra-cores are related to constraints within the same core, whereas the dependencies inter-cores are related to constraints coming from other cores. The valid condition is a combination of the intra and inter dependencies, which are used to validate the referred SE. The maximum amount per 4x4 block indicates the maximum allowed number of the given residual SE. Figure C.9 is provided to guide the understanding of the MRSET internal functionality, in which the arrows traveling inside a core relate to the intra-core dependencies, whereas the arrows traveling outside the core associate to the inter-core dependencies. Each residual SEs clear behavior discrimination comes as follows:

- **SIG**: sixteen of these elements can occur per 4x4 block. They can only be generated after the LAST position. Significant coefficients receive the value ‘1’, whereas zeroed coefficients receive the value ‘0’.
- **COEFF1**: a maximum of eight of these elements are permitted per 4x4 block. In case this limit is not yet reached, the other condition to fabricate COEFF1 for the given transformed residue is to have a SIG with value ‘1’ associated to the same coefficient. The #COEFF1 register starts with the value eight, and every time a valid COEFF1 is produced within a MRSET core, its value is decremented, and this value travels combinationally to the next appended core. At the time the #COEFF1 signal reaches zero, no more valid COEFF1 is generated for the upcoming transformed residues. At the end of the cycle, the update value of #COEFF1 is store in the referred register. The valid COEFF1 signaled as ‘1’ indicates that the processed coefficient has an absolute value above one, whereas signaled as ‘0’ indicates that the absolute value is exactly one.
- **COEFF2**: only one of this element processing happens per 4x4 block. The ACOEFF2 starts with value ‘1’ to indicate that COEFF2 has not yet happened, whereas the value ‘0’ indicates the opposite, and this signal travel in a combinational fashion among the MRSET cores. At the end of the cycle, the updated signal is stored in the ACOEFF2 register. The first valid COEFF1 produces this element. If the current coefficient absolute value is also greater than two, the COEFF2 receive the value ‘1’, or receives ‘0’, otherwise.
- **SIGN**: sixteen of these elements can happen per 4x4 block. It signals if the coefficient is positive or negative, whereas the dependency is related to the SIG element of the same transformed residue (i.e., in case SIG has the value ‘1’, a SIGN will be produced). SIGN with value ‘1’ indicates a negative coefficient, whereas with value ‘0’ a positive one.
- **REM**: sixteen of this element may exist per 4x4 block. It will be produced every time the respective transformed residue has an absolute value greater than one, but not necessarily that a COEFF1 is associated to this residue (the associated SIG has to have value ‘1’, at least). Nevertheless, the REM value

depends whether the COEFF1 and COEFF2, only COEFF1, or no valid COEFF1 exist for the processed coefficient. For instance, when both COEFF1 and COEFF2 with value ‘1’ exist for a given transformed residue, the REM is the absolute value of the residue decremented by three. If only COEFF1 with value ‘1’ exist for the respective coefficient, the REM it its respective absolute value decremented by two. In case no valid COEFF1 can be associated to the current coefficient, but it has an absolute value greater than one, the REM is this absolute value decremented by one.

Figure C.9 – Internal structure of a MRSET single core



Source: (SAGGIORATO, 2018).

Table C.5 – MRSET dependencies and limits amount

Syntax Element	SIG	COEFF1	COEFF2	SIGN	REM

Value	0, if residue = 0
	1, otherwise.
		1, if residue > 1
		0, otherwise.
			1, if residue > 2.
			0, otherwise.
				0 for positive residue,
				1 for negative residue.
					residue – 3, if COEFF2 and COEFF1 valid;
					residue – 2, if only COEFF1 valid;
					residue – 1, otherwise

Dependency intra-core	-	SIG = 1	COEFF1 = 1	SIG = 1	SIG = 1 and |residue| > 1
Dependency inter-core	-	#COEFF1 > 0	ACOEFF = 1	-	-
Valid condition	-	SIG = 1 and #COEFF1 > 0	COEFF1 = 1 and ACOEFF = 1	SIG = 1	SIG = 1 and |residue| > 1
Maximum amount per 4x4 block	16	8	1	16	16

Source: (SAGGIORATO, 2018).

C.4.3 Power-saving MRSET Approach

Power dissipation is a key variable when one considers embedded hardware design on battery-based devices, such as smartphones, for instance. Video coding is suitable of usage on this context. Thus, a power-saving approach of the MRSET circuit is welcome. There are two interested points that may be feasible for low-power insertions: the registers used for the residual SEs storage, and the subtractor required for the REM generation (refer to the REM block on Figure C.9).

The registers are only active when they have to store a new value of the correspondent residual SE i.e., when the valid condition of the respective element is true. For example, at a given cycle, only non-significant coefficients occurs for a given version of the MRSET. Therefore, except for the registers that have to store the SIG, no other will memorize any new value. In that case, these registers are suitable for a Clock-Gating approach (WU, 2000), where the clock of these registers is fed to them only when they require saving new SE values. The same reasoning occurs to the #COEFF1 and ACOEFF2, in case the valid condition for both COEFF1 and COEFF2 has not happen for any of the MRSET cores in a given cycle. The application of Clock-Gating technique avoids unnecessary short-circuit dynamic power consumption, considering the context presented.

The other approach is to add Operand Isolation (CORREALE, 1995) cells (i.e., and AND gate in front of every input of the logic) to the already mentioned subtractor of each MRSET core, since they are only required when valid REM happens. Hence, this approach avoids switching dynamic power consumption when no valid REM generation occurs.

A power–saving approach PRSET was decided not to be done, because this logic operation lasts for only one clock cycle for every TU size. After that, it is estimated that the memories that feed the PRSET inputs would wait, keeping the same values on input of the mentioned circuit, until MRSET process its results. Thus, no dynamic power consumption is expected, since no switching happens within the circuitry, due to the halted inputs (remembering that PRSET is a purely combinational circuit).

C.5 ASIC Synthesis Results and Discussion

The related CABAC main variables appear in Table C.6, and will guide the discussion related to the synthesis results presented in this section. The interested variables are the maximum frequency achieved by these entropy encoder designs, the average number of bins per cycle, the technology used for the synthesis, and the area resources. The PRSET and MRSET have to, therefore (i) to have at least the same maximum frequency of the related CABAC architectures, and (ii) to be able to supply an average bins per cycle compliant with these state-of-the-art CABAC circuits. The problem of different clock frequencies is the need to add measures concerning that matter, like asynchronous FIFO to manage the two clock domains, for instance (KRSTIC, 2017). Regarding the average bins per cycle, the goal is exactly what is the primary objective of this research: to avoid the transfer of the bottleneck from CABAC to the SEs generation. Hence, PRSET and MRSET must be compliant with the average bins per cycle throughput of the related designs.

Table C.6 – Related HEVC entropy encoder characteristics

Design	(FEI, 2011)	(ZHOU, 2013)	(ZHOU, 2015)	(RAMOS, 2018a)
Max. Frequency	279 MHz	402 MHz	420 MHz	537 MHz
Bins/cycle	4	4.4	4.37	4.94
Technology	90 nm	65 nm	90 nm	65 nm
Gates Count	36.24 K	57.3 K †	64.1 K †	33K ‡

† Whole CABAC result without memory
‡Only BAE result

Source: (SAGGIORATO, 2018).

The more cores one appends to another in the MRSET architecture, the more bins per cycle the logic will be able to delivery, but at the cost of degradation in frequency, due to the increased critical path. Thus, five MRSET version were described in VHDL for the assessment wanted. Each version with a different number of cores, respectively one, two, four, eight, and sixteen cores, and named based on the number of cores: 1-MRSET, 2-MRSET, 4-MRSET, 8-MRSET, and 16-MRSET. The upper limit is sixteen cores, because there will be never more than sixteen transformed residues to be processed at once, per 4x4 block.

These five versions underwent a synthesis for ST 65nm CMOS using the Cadence® RTL Compiler tool. The maximum frequency for each one of them, along with the gates count, power, estimated energy results, and power-area ratio (stated as P-A Ratio) appear in Table C.7. Moreover, an optimistic estimative of maximum processed residual SEs per second is also provided in Table C.7. The first thing to notice is, considering that the first residual SE after the LAST is the SIG, which is a flag, that the 1-MRSET and 2-MRSET can deliver one and two bins per cycle for the beginning of residual SEs processing, which is below the average number of all related high-throughput entropy-encoder architectures. However, starting at the 4-MRSET up to the 16-MRSET, these MRSET designs can supply at least around the average of the related bins per cycle throughput, or even more. Furthermore, the design in (RAMOS, 2018a) can process multiple bypass bins in parallel, which makes valuable to have more SIGN and REM elements at once, since they generate bypass bins, for instance.

Table C.7 – Synthesis results for the MRSET designs

MRSET version	Max. Frequency	Gates Count	Power	Estimated energy†	Throughput	P-A Ratio
1-MRSET	1.63 GHz	1.12 K	3.788 mW	37.18 pJ	1.63 GR/s	3.32
2-MRSET	1.06 GHz	2.02 K	6.742 mW	50.88 pJ	2.12 GR/s	3.34
4-MRSET	668 MHz	3.67 K	11.52 mW	69.00 pJ	2.67 GR/s	3.14
8-MRSET	376 MHz	6.89 K	21.21 mW	113.35 pJ	3.00 GR/s	3.09
16-MRSET	196 MHz	13.56 K	41.46 mW	211.55 pJ	3.13 GR/s	3.06

GR/s = Giga-transformed residues per second
† Considering one worst-case 4x4 block running on maximum frequency reachable

Source: the author.

The second outcome from Table C.7 is that only 1-MRSET, 2-MRSET, and 4-MRSET accomplish the maximum frequency of all related CABAC designs. From this perspective, and considering what was presented on last paragraph, it seems that the 4-MRSET is the best trade-off choice among the MRSET designs (i.e., it achieves around the average bins per cycle throughput and maximum frequency of the referred CABAC circuits). In fact, the 8-MRSET and 16-MRSET have a potentially higher transformed residues processing per second than the others MRSETs. Nevertheless, this throughput is extremely optimistic (i.e., it is considered that, the entire time, the maximum number possible of coefficients is available for these two MRSET logics, which is highly improbable). Moreover, as already discussed, the usage of different clock domains (since the 8-MRSET and 16-MRSET do not achieve the maximum frequency of the state-of-the-art entropy encoding designs) is a drawback that requires extra measures to deal with, or it requires to slow the clock frequency of the whole MRSET plus CABAC logic, which is not an efficient fashion.

The power results of Table C.7, derived from the synthesis tool, increase as the MRSET architecture has more cores, as would be expected. One may observe that the power-area ratio vary from 3.06 to 3.34, a difference of around 8%, and thus are relatively close (i.e., seems to be a linear correlation between the increase in area resources and the power dissipated, therefore). None of the high-throughput related works of Table C.6 presents power values, but comparing to the results of the low-power BAE of (RAMOS, 2017) (i.e., 15.93 mW), the 4-MRSET corresponds to 72% of the power dissipated by the cited BAE design, which states the importance of the power-saving MRSET approaches, whose results will be further presented. One remark is that the results refer to the total power dissipated, whereas the majority contributor component is the dynamic power consumption (around 99% of total), whereas static power consumption is almost negligible.

The estimated worst-case energy per one 4x4 block of coefficients (i.e., all the sixteen transformed residues of all 4x4 blocks will have to be processed), running on the maximum frequency reachable by each architectures, points the efficiency of the lesser-cores MRSET versions against the bigger ones. Even if the smaller MRSET circuits demands more clock cycles to process the 4x4 block, they requires less time (due to higher maximum frequency), and consume less power during this time.

Regarding the area and focusing especially on the 4-MRSET design (the best trade-off choice, as conclude before), one may notice that, when compared to the area results of the related CABAC (or BAE) architectures, the 4-MRSET design occupies roughly around 10% of (FEI, 2011), whereas for the other designs this area percentage is even smaller. For instance, compared to (ZHOU, 2013) and (ZHOU, 2015), the percentage would be below 10%, since in the results presented in Table C.6 the memory area is omitted (and which account for significant amount of resources). The 4-MRSET represents around 11% of the resources from (RAMOS, 2018a), but since this last related work only presents BAE results, for a whole CABAC architecture using the proposal of (RAMOS, 2018a), the 4-MRSET would occupy an even smaller percentage of area when compared to the cited entropy encoder solution.

Table C.8 presents the power dissipation results, along with the area, maximum frequency, estimated energy, power-area ratio (P-A Ratio), and power gains of the power-saving approach of MRSET versions (stated in the Table C.8 as LP-MRSET). The values were obtained using the same synthesis scenario as presented for the original versions of the

MRSET. As one may conclude, by utilizing the low-power techniques, power consumption decreases by a range 26.58% to 30.38%. The degradation in area and clock frequency (around 2% for both variables) by utilizing these techniques are negligible when compared with the gains in terms of power consumption (that is the reason to omit the estimated throughput as presented in Table C.7, since they would be closely the same).

Table C.8 – Power results for the LP-MRSET designs

LP-MRSET version	Max. Frequency	Gates Count	Power	Estimated energy†	P-A Ratio	Power saving
1-MRSET	1.61 GHz	1.17 K	2.658 mW	26.41 pJ	2.27	29.82 %
2-MRSET	1.05 GHz	2.08K	4.734 mW	36.06 pJ	2.28	29.78 %
4-MRSET	652 MHz	3.75 K	8.460 mW	51.90 pJ	2.26	26.58 %
8-MRSET	369 MHz	6.97 K	15.47 mW	83.86 pJ	2.22	27.39 %
16-MRSET	191 MHz	13.82 K	28.86 mW	151.13 pJ	2.09	30.38 %

† Considering one worst-case 4x4 block running on maximum frequency reachable

Source: the author.

The energy values follows the same behavior as presented in Table C.7, with the lesser-cores versions being more efficient, but now requiring less energy to process the same worst-case 4x4 block, Moreover, the power-area ratio decreases on an average of 30.3% as compared to the original versions, but keeping the same apparent linear correlation. As a conclusion, the application of Clock-Gating and Operand Isolation techniques to the MRSET seems valid, since the frequency-area degradation is far below the power dissipation savings. As a remark, the static power consumption is negligible, as were for the power values of the original MRSET architectures.

The PRSET design was also synthesized for the same PDK, using the same synthesis tool. The intention was to set different frequencies constraints to verify the impact in area for each one of them, as depicted in Table C.9. Furthermore, total power, and power-area (P-A) ratio also appear in Table C.9. As can be noticed, the PRSET is able to accomplish the maximum frequency of any MRSET version, and thus any of the related CABAC architectures. The higher is the desired frequency, the higher is the penalty in terms of gates count. However, considering an average frequency constraint, for instance 666 MHz (close to the frequency accomplished by 4-MRSET), the area penalty drops considerably compared with the higher frequency PRSET versions. Since the 4-MRSET was defined as the best choice, there seems to be no need to have a PRSET running above the mentioned frequency, and which achieves a compromise area value.

Moreover, the power results are far higher for the highest-frequency goal synthesis than for the smaller-frequencies goal versions. As one may notice, the P-A ratio does not seem to be linear, as was for the MRSET architecture. This result corroborates the decision to use a PRSET version with a maximum frequency close to the value achieved by the 4-MRSET design. The best choice PRSET (i.e., for the 666 MHz clock constraint) occupies an area around 19% against (FEI, 2011), and down to roughly 11% compared to (ZHOU, 2013), (ZHOU, 2015) considering only the non-memory area resources, as stated in Table C.6. Against the BAE results of (RAMOS, 2018a), the chosen PRSET design is around 21% of the related area (but remembering that this latter work does not preset a whole CABAC result).

Table C.9 – Synthesis results for the PRSET

Target Frequency	Gates Count	Power	P-A Ratio
1.65 GHz	32.9 K	75.75 mW	2.30
1.00 GHz	15.9 K	22.18 mW	1.39
666 MHz	6.86 K	8.349 mW	1.21
500 MHz	6.45 K	6.176 mW	0.95

Source: the author.

Finally, for the best of our knowledge, there was not found up to the moment no other work that deals with the scenario here presents and hence no comparisons are possible. Nevertheless, this reinforces the novelty of the proposal, especially considering the high-throughput CABAC designs on an even higher demanding future for UHD video processing and transmission.

APPENDIX D <SUPPORT DATA FOR EFFICIENT ENERGY-THROUGHPUT TRADE-OFF BAE ARCHITECTURE – ET-BAE >

This appendix shows the support data used during the inception and design of the ET-BAE proposal on Chapter 7, which appear on this Appendix by means of organization and readability of the text. The results from Table D.1, Table D.2, Table D.3, Table D.4, Table D.5, and Table D.6 are real simulation values, using the mentioned recommended video sequences for MB-BAE and AltBAE designs, and decreasing the amount of cores for each subsequent design.

For instance, MB-BAE 2-Cores is MB-BAE full design – refer to Figure 6.2 - minus PN rLPs 6, LPS/MPS 6, PN rLPS 5, LPS 5, Range for MBBS 4, Low update 4, OB update 4, and Output generation 4. MB-BAE 1-Core is MB 2-Cores minus PN rLPs 4, LPS/MPS 4, PN rLPS 3, and LPS 3, Range for MBBS 3, Low update 3, OB update 3, and Output generation 3. AltBAE designs are the same as decribed in Chapter 7.

The values on Table D.7, Table D.8, Table D.9, Table D.10, Table D.11, and Table D.12 are the amount of bits generated by each recommended video sequence in the HM, for each frame within the sequence, for LD and RA configuration and initial QP 22 and 37, for the first initial second of video.

Table D.1 – Throughput for MB-BAE full architecture

MB-BAE - Full							
Sequence	**Config.**	**QP**	**Bins/cycle**	**Sequence**	**Config.**	**QP**	**Bins/cycle**
Traffic	LD	22	4.79	BlowingBubbles	LD	22	4.98
		37	4.66			37	4.96
	RA	22	4.98		RA	22	5.25
		37	4.76			37	5.06
PeopleOnStreet	LD	22	5.23	BasketballPass	LD	22	4.94
		37	5.14			37	4.73
	RA	22	5.46		RA	22	5.28
		37	5.31			37	4.9
BasketballDrive	LD	22	4.71	Johnny	LD	22	4.35
		37	4.66			37	4.19
	RA	22	4.85		RA	22	4.46
		37	4.78			37	4.26
Kimono	LD	22	5.63	FourPeople	LD	22	4.59
		37	4.76			37	4.36
	RA	22	5.83		RA	22	4.76
		37	4.99			37	4.48
RaceHorses	LD	22	5.07	ChinaSpeed	LD	22	5.34

Sequence	Config.	QP	Bins/cycle	Sequence	Config.	QP	Bins/cycle
		37	4.94			37	4.82
	RA	22	5.29		RA	22	5.45
		37	5.11			37	4.94
PartyScene	LD	22	4.99	SlideEditing	LD	22	6.21
		37	4.89			37	5.07
	RA	22	5.26		RA	22	6.32
		37	5.02			37	5.16
Average							5.00

Source: the author.

Table D.2 – Throughput for MB-BAE 2-Cores architecture

MB-BAE – 2Cores							
Sequence	Config.	QP	Bins/cycle	Sequence	Config.	QP	Bins/cycle
Traffic	LD	22	3.31	BlowingBubbles	LD	22	3.26
		37	3.19			37	3.28
	RA	22	3.44		RA	22	3.48
		37	3.26			37	3.33
PeopleOnStreet	LD	22	3.61	BasketballPass	LD	22	3.59
		37	3.54			37	3.41
	RA	22	3.78		RA	22	3.78
		37	3.65			37	3.54
BasketballDrive	LD	22	3.22	Johnny	LD	22	2.96
		37	3.19			37	2.84
	RA	22	3.32		RA	22	3.04
		37	3.26			37	2.89
Kimono	LD	22	3.9	FourPeople	LD	22	3.14
		37	3.26			37	2.97
	RA	22	4.04		RA	22	3.26
		37	3.41			37	3.04
RaceHorses	LD	22	3.5	ChinaSpeed	LD	22	3.72
		37	3.4			37	3.3
	RA	22	3.68		RA	22	3.8
		37	3.51			37	3.38
PartyScene	LD	22	3.45	SlideEditing	LD	22	4.52
		37	3.37			37	3.48
	RA	22	3.65		RA	22	4.61
		37	3.45			37	3.55
Average							3.45

Source: the author.

Table D.3 – Throughput for MB-BAE 1-Core architecture

MB-BAE – 1Core							
Sequence	Config.	QP	Bins/cycle	Sequence	Config.	QP	Bins/cycle
Traffic	LD	22	1.73	BlowingBubbles	LD	22	1.69
		37	1.65			37	1.7

	RA	22	1.8		RA	22	1.81
		37	1.68			37	1.73
PeopleOnStreet	LD	22	1.88	BasketballPass	LD	22	1.88
		37	1.84			37	1.77
	RA	22	1.97		RA	22	1.98
		37	1.89			37	1.84
BasketballDrive	LD	22	1.66	Johnny	LD	22	1.52
		37	1.64			37	1.45
	RA	22	1.71		RA	22	1.56
		37	1.68			37	1.47
Kimono	LD	22	2.03	FourPeople	LD	22	1.62
		37	1.68			37	1.52
	RA	22	2.11		RA	22	1.68
		37	1.76			37	1.56
RaceHorses	LD	22	1.82	ChinaSpeed	LD	22	1.94
		37	1.77			37	1.7
	RA	22	1.92		RA	22	1.98
		37	1.82			37	1.75
PartyScene	LD	22	1.79	SlideEditing	LD	22	2.37
		37	1.75			37	1.8
	RA	22	1.9		RA	22	2.43
		37	1.79			37	1.84
Average							**1.79**

Source: the author.

Table D.4 – Throughput for AltBAE full architecture

AltBAE - Full							
Sequence	**Config.**	**QP**	**Bins/cycle**	**Sequence**	**Config.**	**QP**	**Bins/cycle**
Traffic	LD	22	4.27	BlowingBubbles	LD	22	4.36
		37	4.15			37	4.33
	RA	22	4.37		RA	22	4.5
		37	4.2			37	4.37
PeopleOnStreet	LD	22	4.46	BasketballPass	LD	22	4.35
		37	4.39			37	4.2
	RA	22	4.55		RA	22	4.52
		37	4.47			37	4.27
BasketballDrive	LD	22	4.18	Johnny	LD	22	3.97
		37	4.14			37	3.84
	RA	22	4.25		RA	22	4.04
		37	4.19			37	3.88
Kimono	LD	22	4.66	FourPeople	LD	22	4.12
		37	4.2			37	3.96
	RA	22	4.74		RA	22	4.21
		37	4.32			37	4.02
RaceHorses	LD	22	4.41	ChinaSpeed	LD	22	4.55

		37	4.3			37	4.24
	RA	22	4.54		RA	22	4.6
		37	4.38			37	4.31
PartyScene	LD	22	4.37	SlideEditing	LD	22	5.01
		37	4.29			37	4.4
	RA	22	4.51		RA	22	5.06
		37	4.35			37	4.45
Average							**4.34**

Source: the author.

Table D.5 – Throughput for 2C-BAE architecture

2C-BAE							
Sequence	**Config.**	**QP**	**Bins/cycle**	**Sequence**	**Config.**	**QP**	**Bins/cycle**
Traffic	LD	22	2.96	BlowingBubbles	LD	22	3.02
		37	2.86			37	3
	RA	22	2.9		RA	22	3.13
		37	3.03			37	3.03
PeopleOnStreet	LD	22	3.09	BasketballPass	LD	22	3.02
		37	3.04			37	2.9
	RA	22	3.18		RA	22	3.14
		37	3.1			37	2.95
BasketballDrive	LD	22	2.88	Johnny	LD	22	2.72
		37	2.85			37	2.62
	RA	22	2.93		RA	22	2.77
		37	2.89			37	2.64
Kimono	LD	22	3.23	FourPeople	LD	22	2.83
		37	2.9			37	2.71
	RA	22	3.29		RA	22	2.9
		37	2.97			37	2.75
RaceHorses	LD	22	3.06	ChinaSpeed	LD	22	3.16
		37	2.98			37	2.93
	RA	22	3.15		RA	22	2.97
		37	3.03			37	3.19
PartyScene	LD	22	3.02	SlideEditing	LD	22	3.48
		37	2.97			37	3.03
	RA	22	3.13		RA	22	3.52
		37	3.02			37	3.08
Average							**3.00**

Source: the author.

Table D.6 – Throughput for 1C-BAE architecture

1C-BAE							
Sequence	**Config.**	**QP**	**Bins/cycle**	**Sequence**	**Config.**	**QP**	**Bins/cycle**
Traffic	LD	22	1.58	BlowingBubbles	LD	22	1.61
		37	1.53			37	1.61

	RA	22	1.63		RA	22	1.68
		37	1.55			37	1.63
PeopleOnStreet	LD	22	1.66	BasketballPass	LD	22	1.61
		37	1.64			37	1.55
	RA	22	1.71		RA	22	1.69
		37	1.67			37	1.58
BasketballDrive	LD	22	1.53	Johnny	LD	22	1.43
		37	1.52			37	1.43
	RA	22	1.56		RA	22	1.46
		37	1.54			37	1.39
Kimono	LD	22	1.73	FourPeople	LD	22	1.5
		37	1.54			37	1.43
	RA	22	1.77		RA	22	1.54
		37	1.58			37	1.47
RaceHorses	LD	22	1.63	ChinaSpeed	LD	22	1.7
		37	1.6			37	1.55
	RA	22	1.69		RA	22	1.71
		37	1.63			37	1.58
PartyScene	LD	22	1.61	SlideEditing	LD	22	1.88
		37	1.6			37	1.61
	RA	22	1.67		RA	22	1.9
		37	1.62			37	1.63
Average							**1.60**

Source: the author.

Table D.7 – Traffic sequence first second bitrate distribution

Traffic - LD -22			Traffic - LD -37			Traffic - RA -22			Traffic - RA -37		
TID	QP	Bits	TID	QP	Bits	TID	QP	Bits	TID	QP	Bits
0	22	3549016	0	37	655648	0	22	3801400	0	37	715432
1	25	259792	1	40	8824	8	23	1689896	8	38	143760
2	24	389488	2	39	14976	4	24	571016	4	39	21688
3	25	259528	3	40	11272	2	25	279280	2	40	10280
4	23	1056848	4	38	70072	1	26	75880	1	41	3480
5	25	214744	5	40	7952	3	26	64744	3	41	2688
6	24	406608	6	39	13760	5	25	300680	5	40	11368
7	25	231104	7	40	11992	6	26	62928	6	41	2592
8	23	1019160	8	38	83400	7	26	66320	7	41	2904
9	25	220744	9	40	8360	16	23	1514136	16	38	145800
10	24	393056	10	39	13944	12	24	580640	12	39	21528
11	25	240152	11	40	10832	10	25	281368	10	40	9744
12	23	975256	12	38	84432	9	26	68264	9	41	2704
13	25	220176	13	40	8296	11	26	60760	11	41	2688
14	24	396176	14	39	13736	14	25	281968	14	40	11104
15	25	229040	15	40	10808	13	26	60248	13	41	2856
16	23	952744	16	38	85248	15	26	64008	15	41	2464

17	25	219144	17	40	7632	24	23	1475448	24	38	138616
18	24	373072	18	39	12264	20	24	488680	20	39	16208
19	25	222968	19	40	10816	18	25	274232	18	40	9504
20	23	891816	20	38	79504	17	26	64320	17	41	2312
21	25	209320	21	40	7048	19	26	54576	19	41	2304
22	24	363408	22	39	11784	22	25	265608	22	40	8592
23	25	221272	23	40	9576	21	26	54760	21	41	2336
24	23	881776	24	38	80464	23	26	56480	23	41	2368
25	25	194776	25	40	6304	28	24	677192	28	39	28744
26	24	340784	26	39	11176	26	25	249864	26	40	7608
27	25	230720	27	40	9464	25	26	56808	25	41	2312
28	23	871696	28	38	77720	27	26	58096	27	41	2288
29	25	201264	29	40	7464	29	26	95048	29	41	5056

Source: the author.

Table D.8 – Kimono sequence first second bitrate distribution

Kimono - LD -22			Kimono - LD -37			Kimono - RA -22			Kimono - RA -37		
TID	QP	Bits	TID	QP	Bits	TID	QP	Bits	TID	QP	Bits
0	22	773544	0	37	164280	0	22	836976	0	37	175312
1	25	211248	1	40	14192	8	23	667848	8	38	123088
2	24	301832	2	39	24176	4	24	410112	4	39	42072
3	25	233624	3	40	16744	2	25	263008	2	40	21080
4	23	457992	4	38	83912	1	26	100128	1	41	4464
5	25	213176	5	40	13016	3	26	98720	3	41	4768
6	24	280840	6	39	22712	6	25	253736	6	40	19096
7	25	218944	7	40	15904	7	26	97656	7	41	4216
8	23	419480	8	38	81736	5	26	92704	5	41	3728
9	25	200784	9	40	10904	16	23	698152	16	38	124944
10	24	269912	10	39	21824	12	24	403072	12	39	38760
11	25	222000	11	40	16216	10	25	245472	10	40	18168
12	23	463536	12	38	85176	9	26	90304	9	41	3760
13	25	213600	13	40	11144	11	26	93136	11	41	3784
14	24	286776	14	39	22024	14	25	249960	14	40	19152
15	25	220880	15	40	14904	13	26	90696	13	41	3368
16	23	449624	16	38	83448	15	26	90432	15	41	3224
17	25	201472	17	40	10416	20	24	434320	20	39	46848
18	24	272824	18	39	21152	18	25	243680	18	40	18080
19	25	211824	19	40	14136	17	26	92048	17	41	3704
20	23	421472	20	38	79432	19	26	85480	19	41	2928
21	25	194240	21	40	10152	22	25	293600	22	40	26840
22	24	255984	22	39	18552	21	26	83608	21	41	2736
23	25	186104	23	40	11696	23	26	122352	23	41	4744

Source: the author.

Table D.9 – PartyScene sequence first second bitrate distribution

PartyScene - LD -22		
TID	QP	Bits
0	22	1046848
1	25	128
2	24	144
3	25	157616
4	23	406336
5	25	162360
6	24	246880
7	25	173576
8	23	405640
9	25	158208
10	24	239032
11	25	166872
12	23	398936
13	25	155072
14	24	234792
15	25	176696
16	23	413960
17	25	163024
18	24	238056
19	25	175296
20	23	397952
21	25	153696
22	24	228824
23	25	165776
24	23	383280
25	25	148240
26	24	222120
27	25	159928
28	23	377392
29	25	145576
30	24	215896
31	25	147208
32	23	358216
33	25	124488
34	24	193592
35	25	121768
36	23	326504
37	25	102952
38	24	158528
39	25	100744
40	23	301872
41	25	88360

PartyScene - LD -37		
TID	QP	Bits
0	37	230120
1	40	120
2	39	128
3	40	7032
4	38	44592
5	40	6784
6	39	12320
7	40	8512
8	38	57328
9	40	6528
10	39	12392
11	40	7832
12	38	56976
13	40	6040
14	39	12032
15	40	8360
16	38	57824
17	40	6920
18	39	11904
19	40	8136
20	38	54952
21	40	6544
22	39	11888
23	40	8760
24	38	53512
25	40	6824
26	39	11872
27	40	8784
28	38	51640
29	40	6744
30	39	10712
31	40	8432
32	38	45752
33	40	6008
34	39	8944
35	40	5912
36	38	36736
37	40	4864
38	39	6528
39	40	4872
40	38	33272
41	40	3952

PartyScene - RA -22		
TID	QP	Bits
0	22	1091408
8	23	568176
4	24	284560
2	25	168
1	26	96
3	26	73368
6	25	200032
5	26	73528
7	26	72560
16	23	576776
12	24	302752
10	25	195368
9	26	71664
11	26	68560
14	25	197496
13	26	70856
15	26	72040
24	23	521776
20	24	303864
18	25	202232
17	26	77184
19	26	74208
22	25	203016
21	26	71408
23	26	69344
32	23	491288
28	24	273032
26	25	191000
25	26	69584
27	26	68952
30	25	181888
29	26	67536
31	26	61192
40	23	442296
36	24	206440
34	25	154896
33	26	55304
35	26	44624
38	25	133616
37	26	35800
39	26	32664
48	22	1004776

PartyScene - RA -37		
TID	QP	Bits
0	37	255320
8	38	79272
4	39	18128
2	40	96
1	41	88
3	41	2424
6	40	11776
5	41	1904
7	41	2064
16	38	87392
12	39	21472
10	40	11296
9	41	2008
11	41	1776
14	40	11096
13	41	1840
15	41	2016
24	38	76272
20	39	22104
18	40	12144
17	41	1912
19	41	1736
22	40	11400
21	41	1848
23	41	1888
32	38	67528
28	39	20856
26	40	12024
25	41	2200
27	41	2096
30	40	10736
29	41	1752
31	41	1920
40	38	54144
36	39	11984
34	40	8752
33	41	1776
35	41	1744
38	40	6416
37	41	1568
39	41	1624
48	37	229552

42	24	146680	42	39	6392	44	24	166072	44	39	11136
43	25	96424	43	40	5016	42	25	112544	42	40	6320
44	23	288664	44	38	30416	41	26	31664	41	41	1816
45	25	87616	45	40	4464	43	26	32872	43	41	1608
46	24	154344	46	39	8488	46	25	124280	46	40	8088
47	25	112624	47	40	6456	45	26	38712	45	41	1864
48	22	963776	48	37	208304	47	26	45480	47	41	2280
49	25	111240	49	40	6648	49	26	69512	49	41	3768

Source: the author.

Table D.10 – BlowingBubbles sequence first second bitrate distribution

BlowingBubbles - LD -22			BlowingBubbles - LD -37			BlowingBubbles - RA -22			BlowingBubbles - RA -37		
TID	QP	Bits	TID	QP	Bits	TID	QP	Bits	TID	QP	Bits
0	22	195384	0	37	34784	0	22	207504	0	37	39528
1	25	17080	1	40	1088	8	23	112648	8	38	12472
2	24	32176	2	39	1368	4	24	42248	4	39	1976
3	25	20584	3	40	1024	2	25	23352	2	40	1592
4	23	71136	4	38	7544	1	26	5688	1	41	304
5	25	15032	5	40	736	3	26	5240	3	41	232
6	24	29760	6	39	1048	6	25	19888	6	40	984
7	25	16008	7	40	904	5	26	4016	5	41	176
8	23	66440	8	38	6256	7	26	4976	7	41	296
9	25	15848	9	40	560	16	23	113176	16	38	12824
10	24	30288	10	39	936	12	24	44064	12	39	2184
11	25	20816	11	40	920	10	25	21312	10	40	848
12	23	67912	12	38	6408	9	26	5000	9	41	256
13	25	18984	13	40	640	11	26	5456	11	41	224
14	24	33776	14	39	1288	14	25	22520	14	40	1008
15	25	22232	15	40	880	13	26	6344	13	41	296
16	23	72608	16	38	7712	15	26	5680	15	41	368
17	25	16984	17	40	768	24	23	117584	24	38	13944
18	24	33360	18	39	1176	20	24	52800	20	39	2528
19	25	22160	19	40	992	18	25	24880	18	40	928
20	23	74952	20	38	8224	17	26	5648	17	41	240
21	25	17552	21	40	648	19	26	6712	19	41	200
22	24	32552	22	39	1000	22	25	26824	22	40	1288
23	25	22992	23	40	824	21	26	5864	21	41	240
24	23	78912	24	38	8736	23	26	7536	23	41	344
25	25	23368	25	40	1280	32	23	123808	32	38	17248
26	24	41472	26	39	2328	28	24	64512	28	39	4520
27	25	33888	27	40	2128	26	25	36496	26	40	2472
28	23	84288	28	38	11088	25	26	11520	25	41	696
29	25	36272	29	40	2288	27	26	15704	27	41	1088
30	24	52272	30	39	3456	30	25	43504	30	40	3232
31	25	37304	31	40	2560	29	26	17472	29	41	1136

32	23	87376	32	38	11768	31	26	17920	31	41	1160
33	25	32984	33	40	2768	40	23	126400	40	38	18232
34	24	47104	34	39	3584	36	24	64928	36	39	6080
35	25	36712	35	40	3392	34	25	42480	34	40	3680
36	23	84008	36	38	11384	33	26	17680	33	41	1552
37	25	39176	37	40	3480	35	26	19312	35	41	1856
38	24	53560	38	39	4544	38	25	43336	38	40	4464
39	25	37080	39	40	3560	37	26	20336	37	41	2264
40	23	87040	40	38	13520	39	26	17728	39	41	1992
41	25	33024	41	40	2856	48	22	192848	48	37	34992
42	24	47728	42	39	3648	44	24	60088	44	39	5024
43	25	35376	43	40	2768	42	25	39560	42	40	3776
44	23	84704	44	38	12512	41	26	16312	41	41	1752
45	25	30656	45	40	2248	43	26	16536	43	41	1560
46	24	43952	46	39	2896	46	25	34160	46	40	2848
47	25	29656	47	40	2064	45	26	15088	45	41	1184
48	22	181928	48	37	31360	47	26	11728	47	41	976
49	25	23856	49	40	1312	49	26	14472	49	41	1056

Source: the author.

Table D.11 – Johnny sequence first second bitrate distribution

Johnny - LD -22			Johnny - LD -37			Johnny - RA -22			Johnny - RA -37		
TID	QP	Bits	TID	QP	Bits	TID	QP	Bits	TID	QP	Bits
0	22	337688	0	37	63064	0	22	366056	0	37	69992
1	25	15048	1	40	560	8	23	147272	8	38	8304
2	24	30376	2	39	840	4	24	40528	4	39	1080
3	25	13200	3	40	776	2	25	20696	2	40	488
4	23	93768	4	38	4544	1	26	1920	1	41	112
5	25	11504	5	40	496	3	26	1568	3	41	128
6	24	23576	6	39	760	5	25	20720	5	40	496
7	25	12112	7	40	944	6	26	1400	6	41	152
8	23	84328	8	38	5200	7	26	1464	7	41	144
9	25	11504	9	40	456	16	23	142744	16	38	9160
10	24	29912	10	39	952	12	24	41160	12	39	1344
11	25	14384	11	40	912	10	25	26880	10	40	496
12	23	89280	12	38	5696	9	26	2216	9	41	176
13	25	14288	13	40	512	11	26	2008	11	41	176
14	24	31264	14	39	984	14	25	24696	14	40	592
15	25	12408	15	40	888	13	26	1608	13	41	200
16	23	78448	16	38	4568	15	26	2128	15	41	224
17	25	13320	17	40	488	24	23	134864	24	38	8896
18	24	27760	18	39	1024	20	24	35336	20	39	1216
19	25	11008	19	40	592	18	25	21752	18	40	600
20	23	76504	20	38	5144	17	26	1880	17	41	200
21	25	11096	21	40	384	19	26	1256	19	41	184

22	24	25088	22	39	696	22	25	18984	22	40	584
23	25	10296	23	40	672	21	26	1592	21	41	160
24	23	79616	24	38	5384	23	26	1608	23	41	144
25	25	11832	25	40	480	32	23	114504	32	38	7544
26	24	25816	26	39	800	28	24	29616	28	39	1136
27	25	12072	27	40	624	26	25	21608	26	40	616
28	23	67968	28	38	4552	25	26	2496	25	41	192
29	25	11160	29	40	536	27	26	2072	27	41	168
30	24	18800	30	39	456	30	25	16376	30	40	512
31	25	9832	31	40	536	29	26	1936	29	41	184
32	23	64168	32	38	4672	31	26	1504	31	41	160
33	25	11968	33	40	392	40	23	130632	40	38	9240
34	24	23712	34	39	744	36	24	32960	36	39	1392
35	25	10984	35	40	712	34	25	21224	34	40	560
36	23	65584	36	38	4192	33	26	2560	33	41	184
37	25	13192	37	40	472	35	26	2048	35	41	208
38	24	30032	38	39	1040	38	25	23584	38	40	680
39	25	13496	39	40	848	37	26	2056	37	41	200
40	23	77096	40	38	5384	39	26	2352	39	41	176
41	25	16632	41	40	472	48	23	138000	48	38	10640
42	24	35448	42	39	840	44	24	42504	44	39	1528
43	25	14384	43	40	864	42	25	27648	42	40	632
44	23	76888	44	38	6120	41	26	2360	41	41	192
45	25	21040	45	40	640	43	26	1664	43	41	200
46	24	38888	46	39	1248	46	25	30440	46	40	752
47	25	13712	47	40	1080	45	26	1856	45	41	208
48	23	75960	48	38	5368	47	26	1984	47	41	232
49	25	15840	49	40	648	56	23	140080	56	38	9008
50	24	31392	50	39	1104	52	24	40264	52	39	1816
51	25	13760	51	40	1016	50	25	22512	50	40	968
52	23	79448	52	38	5224	49	26	2648	49	41	256
53	25	13936	53	40	616	51	26	2280	51	41	248
54	24	27416	54	39	1136	54	25	17728	54	40	704
55	25	13664	55	40	784	53	26	1744	53	41	280
56	23	81512	56	38	4976	55	26	1944	55	41	272
57	25	13888	57	40	696	58	25	28016	58	40	1480
58	24	26992	58	39	1128	57	26	2800	57	41	280
59	25	14280	59	40	968	59	26	5504	59	41	528

Source: the author.

Table D.12 – ChinaSpeed sequence first second bitrate distribution

ChinaSpeed - LD -22			ChinaSpeed - LD -37			ChinaSpeed- RA -22			ChinaSpeed - RA -37		
TID	QP	Bits	TID	QP	Bits	TID	QP	Bits	TID	QP	Bits
0	22	743176	0	37	227208	0	22	775688	0	37	244112
1	25	126448	1	40	12336	8	23	360952	8	38	71992

2	24	157312	2	39	15648	4	24	206272	4	39	23952
3	25	121504	3	40	12152	2	25	147472	2	40	15032
4	23	236944	4	38	46936	1	26	72720	1	41	5352
5	25	119880	5	40	11528	3	26	75904	3	41	5168
6	24	159912	6	39	15448	5	25	148304	5	40	15032
7	25	132760	7	40	12768	6	26	74464	6	41	5336
8	23	224408	8	38	46624	7	26	76328	7	41	5872
9	25	128352	9	40	13072	16	23	298296	16	38	64984
10	24	151904	10	39	16408	12	24	190728	12	39	23096
11	25	118592	11	40	12752	10	25	146072	10	40	16584
12	23	212832	12	38	45096	9	26	79704	9	41	6440
13	25	120584	13	40	13128	11	26	73872	11	41	5464
14	24	139904	14	39	14160	14	25	142384	14	40	15736
15	25	119544	15	40	12768	13	26	69536	13	41	5784
16	23	218848	16	38	45152	15	26	72464	15	41	5880
17	25	114552	17	40	12880	24	23	304616	24	38	62336
18	24	147696	18	39	16512	20	24	200248	20	39	24480
19	25	114264	19	40	11840	18	25	137272	18	40	16256
20	23	225528	20	38	46000	17	26	71536	17	41	5928
21	25	123056	21	40	12592	19	26	69784	19	41	6368
22	24	140328	22	39	13800	22	25	143680	22	40	15144
23	25	125656	23	40	13104	21	26	68432	21	41	5656
24	23	224904	24	38	45184	23	26	72248	23	41	6120
25	25	122224	25	40	13224	28	24	215568	28	39	27608
26	24	160944	26	39	15992	26	25	149952	26	40	16008
27	25	125880	27	40	12576	25	26	69456	25	41	6792
28	23	218408	28	38	43912	27	26	66440	27	41	5336
29	25	126904	29	40	13024	29	26	86816	29	41	7856

Source: the author.