Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18513-4

A power-efficient approximate approach to improve the computational complexity of coding tools in versatile video coding

Sina Shah Oveisi[1] · Hoda Roodaki[1,3] · Morteza Rezaalipour[2] · Masoud Dehyadegari[1,3]

Received: 19 July 2022 / Revised: 6 September 2023 / Accepted: 29 January 2024

Abstract
Approximate computing is a technique for optimising algorithms while taking into account both application quality of service and computational complexity. Video encoding is a computationally difficult operation, and the complexity is growing as new compression standards, such as Versatile Video Coding (VVC), emerge. Because it uses several new coding methods, the VVC standard delivers superior video compression. However, the human perceptual constraint presents a good chance for a new alternative form of computation, such as approximation computing, which allows for some acceptable level of inaccuracy in the output. In this paper, first, the VVC coding tools are analysed to determine the most computationally complex functions of various coding tools as the best candidates to perform approximate computing. Because addition is a frequent operation employed in all reletaed functions of various coding tools and no one would be feasible without an efficient adder circuit, correct approximate computing is done to the adder circuits of these chosen candidate functions. First, the LOA approximation adder is employed, which achieves 9.67x and 5x power and area reductions, respectively, as compared to the original VVC as the baseline, with no discernible influence on subjective quality. The AxMAP framework is then examined to build more effective approximate adders, which lowered the area and power consumption by 14.88% and 9.11%, respectively, over the LOA technique.

Keywords Approximate computing · VVC standard · Coding tools · LOA · AxMAP

1 Introduction

Versatile video coding (VVC) is the leading video coding standard by the Joint Collaborative Team on Video Coding (JCT-VC) forming the Motion Picture Experts Group (MPEG) and the Video Coding Expert Group (VCEG) [1]. VVC delivers a coding efficiency boost of

✉ Hoda Roodaki
hroodaki@kntu.ac.ir

Extended author information available on the last page of the article

roughly 50% while keeping the same visual quality as prior standards, such as High Efficiency Video Coding (HEVC) [1]. Better quality and greater resolutions are becoming more important as video technologies advance. As a result, the new video codec is highly intriguing in terms of improving the compression efficiency and quality of the predecessor standards. Superior compression efficiency is obtained by the use of novel coding methods that necessitate sophisticated motion and mode searches during encoding. Such complicated encoders are appropriate for applications where offline encoding is available. Complex encoders, on the other hand, provide major issues in maintaining the appropriate throughput and quality in applications demanding realtime video encoding, such as videoconferencing [2]. During the rate-distortion optimization phase, each enabled coding tool is considered one by one for each Coding Tree Unit (CTU). Then, for that CTU's encoding procedure, the coding tools that improve coding efficiency are chosen.

While these improvements improve coding efficiency, they significantly increase computational complexity, in terms of area and power consumption. This challenge is addressed using approximation computation in this situation. It is a potential method for designing hardware elements for efficient video encoding. Approximate computing sacrifices precision for the power, area, and latency required by modern applications such as machine learning and video/image processing [3]. Approximate computing evolved as a useful tool for data processing with lower area and power consumption. Video applications can benefit from the human eye's tolerance for compute error in order to reach high power-efficient architectures. If the degree of computing inaccuracy is kept to a minimum, the result exhibits negligible observable distortion. Of course, the inaccuracy produced by approximate arithmetic cannot be more than a particular threshold.

The benefits of approximation computing are used in this research to lower the computational complexity of the VVC video encoder. The computational complexity of several VVC encoder coding techniques is studied for this purpose in order to determine the most computationally complex ones. Then, as a common operation utilized in most coding tools, addition is chosen as a contender to perform approximate computing. Finally, two distinct frameworks, Lower-part-OR Adder (LOA) [4] and AxMAP [5], are used to create several types of approximate adders applied to the VVC coding tools' adder circuits. The experimental results reveal that these techniques outperform the original VVC encoder in terms of overall power consumption and area. The remainder of the paper is structured as follows. Section 2 goes on the history of the VVC standard and related coding tools. Section 3 discusses the relevant works. The proposed approach is presented in Section 4. The experimental results are presented in Section 5, and the paper is concluded in Section 6.

2 Background

2.1 VVC Encoding Process Overview

In VVC, a frame is divided into CTUs for encoding. Then, according to the quadtree plus a nested multi-type tree partitioning [1], each CTU is divided into certain Coding Units (CU) of varying depths. The largest CU size is 64×64, and the smallest is 8×8. A larger CU can considerably enhance the coding efficiency of the smooth zone, whereas a tiny CU can be used to forecast the complex image more accurately in places with more features. Each CU is divided into one or more Prediction Units (PU) during the prediction phase, and the CU is divided into one or more Transform Units (TU) [1] during the transformation process.

2.1.1 AFFINE coding tool

Traditional video coding's translational motion mode is too simplistic to describe complex motions in natural videos, such as rotation and zooming. As seen in Fig. 1, high-order motion models such as AFFINE are proposed to characterise these complex motions [6].

2.1.2 Decoder-side Motion Vector Refinement (DMVR)

In the bi-prediction motion estimation procedure, this coding tool is used to optimise the extracted motion vectors in order to minimise the bitrate. Because of the use of a B-picture in the VVC standard, the bi-prediction blocks in a B-picture allow us to employ an arbitrary collection of reference images in forward and backward directions, which are termed list1 and list0, respectively [7]. The motion vectors of list0 and list1 are then used to select two prediction blocks for each coding block. This is used with a prediction signal to give a more efficient compression than uni-prediction [7]. The DMVR tool attempts to refine the extracted motion vectors by combining the two prediction blocks extracted from the initial MV0 of list0 and MV1 of list1, respectively.

2.1.3 Bi-Directional Optical Flow (BIO)

A better prediction for the current block could be obtained via bi-prediction. Even with this approach, however, certain motions may persist in some areas of the block because the size of these regions may be smaller than the least permissible prediction block size. As a result, it is preferable to have some method of compensating for each pixel's motion. The motion vector signalling should then be done only for the full block to reduce the total bitrate. BIO enables the refinement of pixel-like motion without the need to transmit additional signals to the decoder. The BIO algorithm may correct for the pixel's minor displacement without the need for extra partitioning. As a result, the ability to perform pixel-like correction without the use of additional syntax will considerably boost the coding efficiency of B-pictures [8].

2.2 Rate-distortion optimization process

In VVC, the rate-distortion optimization process is utilised to select the appropriate coding tools for each CTU in the encoding process based on the rate and distortion cost [9]. The

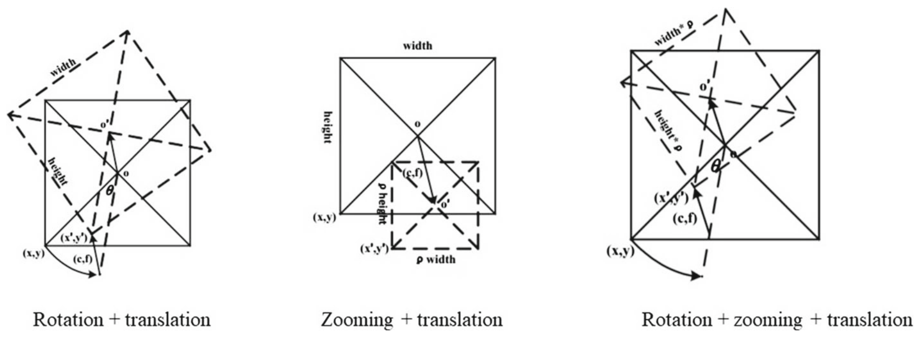


Fig. 1 Examples of the affine model [6]

bitrate cost R and the distortion cost D are merged in the following equation [10] to form a single cost function J.

$$J = D + \lambda \times R \tag{1}$$

The RDO method seeks a coding tool that reduces the overall cost J.

To achieve great compression efficiency, typical VVC encoders use rate-distortion optimization to select the optimum coding tools and parameters from a large number of candidates. However, calculating the rate-distortion cost for each coding tool and parameter is computationally complex. As a result, the significant computational complexity of the RDO process should be managed. To control the computational complexity in this paper, approximation computing is used. The following parts go into greater detail.

3 Related works

In this section, we will look at the most recent approaches to approximate computation in video coding applications. To accelerate the HEVC encoder, approximation logic operators are applied to a sum of absolute differences (SAD) kernel in [1]. An adder-tree approximate logic is implemented at the gate level in this method, and the influence of approximate computing on coding efficiency and power dissipation is calculated. In [11], several strategies such as fractional motion estimation are examined, as well as the computational complexity of these coding tools. To tackle the computational complexity, the fractional motion estimation defines 90 different filters. Then, based on the estimate of the original filter coefficients, an approximate solution for fractional motion estimation interpolation filters is proposed. The hardware acceleration for approximation computing in [11] can accomplish realtime interpolation for Ultra-high-definition (UHD) 8K movies at 30 frames per second. According to the method proposed in [12], the Sum of Absolute Transformed Differences (SATD) is a distortion metric based on the Hadamard Transform that is used in video encoders to refine the motion estimation process in order to find the best block of pixels to use as a prediction for each block to be encoded. As a result, in [12], an approximation SATD hardware accelerator that avoids columns of adders/subtractors of the 8×8 Hadamard Transform is presented. [13] proposes an approximation Versatile Video Coding (VVC) fractional interpolation filter to greatly minimise the computational complexity of this filter. The method proposed in [14] attempts to build a power-efficient SAD architecture based on the use of Lower-Part-OR Adders (LOA) [4] for existing video encoders. The architecture described in [14] attempts to process all of the supported block sizes of modern video encoders. The results demonstrate that this strategy can save a significant amount of power and space. Motion estimation is one of the most computationally demanding modules in the video encoder, according to [15], and an approximate adder to expedite this process is described. The method provided in [16] provides a hardware design for the Fractional Motion Estimation's sub-pixel interpolation unit. This method employs an approximate computation methodology to reduce the number of taps in each filter, hence drastically reducing memory usage. [17] proposes an approximation fractional interpolation filter for Versatile Video Coding (VVC) that decreases the computational cost of VVC fractional interpolation filters while sacrificing very little video quality. In [18], a reconfigurable adder and subtractor block is created, which is then used in the MPEG encoder's motion estimation and discrete cosine transform modules. The degree of hardware approximation is dynamically modified in this method dependent on the input video.

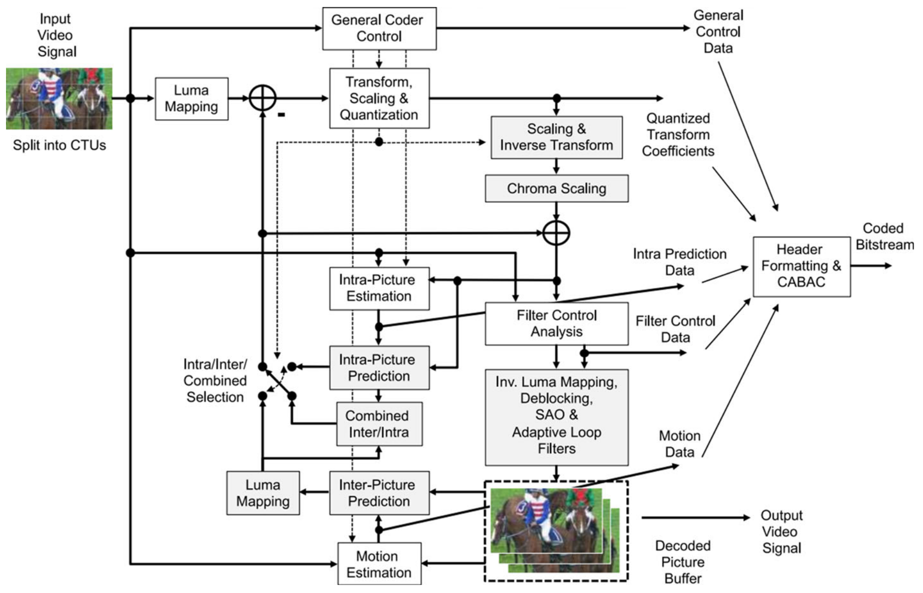


Fig. 2 Typical VVC encoder [1]

The method provided in this study has an advantage over earlier ones in that it takes into account the coding tools employed in the rate-distortion optimization process, which impose considerable computational cost on the encoder. As a result, it is a good option for implementing approximation computing. In comparison to state-of-the-art methods that take only a few specific units into account, such as the interpolation filter or the transformation unit, the suggested method in this research can yield a more significant improvement in power consumption and area.

4 Proposed method

VVC was created to give a significant improvement in video compression, a quest that is aided by a plethora of new or improved coding tools. In this study, approximation computing is used in conjunction with several coding strategies to reduce computational complexity in terms of power consumption and are. The functional diagram of a typical VVC encoder is shown in Fig. 2.

4.1 Select the candidate functions for approximate computing

Various test sequences with varied resolutions were chosen to evaluate the correct candidate functions of different coding tools for applying approximate computing. The features of these sequences are summarised in Table 1.

To extract the result, the VTM reference software version 6.3 [20] is utilized. First, the "encoder randomaccess" configuration file is used to activate each coding tool. The test sequences are then encoded using one of four distinct QP values: 22, 27, 32, and 37. To extract the list of encoding functions, the Intel VTune Profiler [21] is used. Finally, the per-

Table 1 The properties of test sequence [19]

Sequences	Resolution	Frame rate
Basketball Drill	832 x 480	50
Basketball Drive	1920 x 1080	50
Basketball Pass	416 x 240	50
Blowing Bubbles	416 x 240	50
BQMall	832 x 480	60
BQTerrace	1920 x 1080	60
Cactus	1920 x 1080	50
Four People	1280 x 720	60

centage of time necessary to perform the functions of each coding tool to total encoding time is calculated, as shown in Tables 2, 3 and 4. As mentioned in section 2, the XAffineMotionEstimation, XPredAffineInterSearch, and XPredAffineBlk functions are connected to the inter prediction and motion estimation processes in the AFFINE coding tool. The XCheckRDCostAffineMerge2Nx2N function is used in the AFFINE Merge mode to pick the best candidate MV. The XCheckRDCostInterIMV function determines the current pixel precision, whereas XGetAffineTemplateCost computes the rate-distortion value of the current AFFINE candidate. SIMD (Single Instruction Multiple Data) vertical and horizontal filters are related to interpolation filters for intra prediction and are utilized in the SimdFilter, FilterVer, and FilterHor functions. Finally, the BIO coding tools ApplyBiOptFlow and XSubPuBio sunctions are utilized to provide sample-wise motion refinement on top of block-wise motion compensation. As illustrated, the XPredAffineBlk, XPredAffineInterSearch, and AffineMotionEstimation functions consume the majority of processing time. As a result, these functions are ideal candidates for approximate computing implementation.

In addition to the processing time, we propose using the number of calls made to each function by the others as a metric to pick a function as a suitable candidate for implementing approximate computing. Figure 3 depicts the link between functions in terms of how they

Table 2 The percentage of the required time to execute the main functions of AFFINE coding tool to the total encoding time

Sequences	AFFINE					
	Encoding Functions					
	XGetAffine Template Cost	XCheck RDCost InterIMV	XCheck RDCostAffine Merge2Nx2N	XPred Affine Blk	XPred Affine InterSearch	XAffine Motion Estimation
Basketball Drive	21.48	17.51	6.19	31.86	31.08	8.81
Basketball Drill	19.82	16.95	5.64	28.74	28.71	7.36
Basketball Pass	21.42	17.40	4.86	28.08	28.43	5.78
Blowing Bubbles	21.06	17.77	5.56	29.46	29.24	6.82
BQMall	14.60	12.88	6.13	21.67	21.35	5.56
BQTerrace	12.96	10.09	7.69	25.58	23.60	9.18
Cactus	19.77	14.53	6.17	29.42	28.49	7.39
Four People	7.45	13.54	5.95	28.94	28.10	19.27
Average	7.29	15.08	6.02	27.96	27.37	18.79

Table 3 The percentage of the required time to execute the main functions of DMVR coding tool to the total encoding time

Sequences	DMVR		
	Encoding Functions		
	FilterHor	FilterVer	SimdFilter
Basketball Drive	11.60	14.02	15.45
Basketball Drill	11.93	13.04	14.29
Basketball Pass	10.39	11.37	12.93
Blowing Bubbles	11.74	13.37	15.04
BQMall	9.10	9.94	11.69
BQTerrace	11.69	13.00	15.52
Cactus	11.49	12.89	15.06
Four People	9.82	11.47	13.00
Average	10.97	12.38	14.12

are referred to. Because the functions on the graph's leaves are more frequently called by the other functions, they may be better candidates for implementing approximate calculations. Some functions, such as XCheckBestAffineMVP and addAffineMVPCandUnscaled, are not usually used by the other functions. As a result, it is not worthwhile to implement approximate computing in these functions. The use of approximation computing in these functions will merely raise the complexity of the system while not improving the area or power consumption appreciably. The FilterVer and FilterHor functions are likewise suitable candidates for consideration in approximate computing, according to the metric provided in Fig. 3. Finally, the following functions were chosen as viable candidates for the implementation of approximation computations based on the aforementioned criteria, XAffineMotionEstimation, XPredAffineInterSearch, XPredAffineBlk, XCheckRDCostAffineMerge2Nx2N, ApplyBiOptFlow, XSubPuBio, and SimdFilter.

In the following stage, we investigated two different ways for approximate computing implementation in order to reduce the complexity of the candidate functions of each encoding tool in VVC. Since the most common operation in coding tools is addition and subsequent additions can be used to construct other mathematical operations like multiplication and subtraction, the addition operation is used to implement approximate computing.

Table 4 The percentage of the required time to execute the main functions of BIO coding tool to the total encoding time

Sequences	BIO	
	Encoding Functions	
	XSubPuBio	ApplyBiOptFlow
Basketball Drive	6.17	7.40
Basketball Drill	5.45	6.19
Basketball Pass	5.61	6.38
Blowing Bubbles	5.23	5.78
BQMall	7.40	8.60
BQTerrace	9.18	11.90
Cactus	8.05	9.84
Four People	10.96	14.80
Average	8.86	7.25

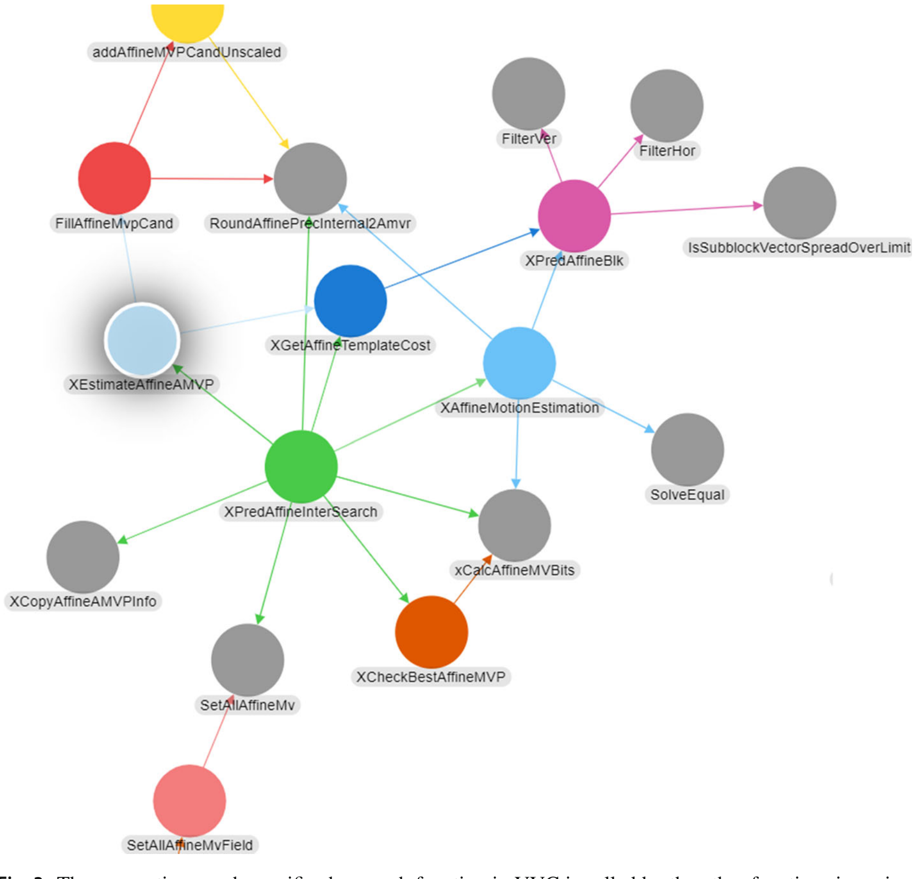


Fig. 3 The connection graph specifies how each function in VVC is called by the other functions in various coding tools

4.2 First Approach: Lower-part-OR Adder (LOA)

The Lower-part-OR Adder (LOA) [4], an ultra-low power adder for soft-computing applications, is utilized in the first technique. According to Fig. 4, an n-bit LOA consists of a k-bit exact adder for the result's most-significant k bits (MSB), and an approximate sub-adder for the n-k least significant bits (LSB). The result's LSB bits are calculated using n two-input OR gates, each with a pair of input bits. Finally, an extra two-input AND gate [4] is used to provide the carry input for the MSB portion. The two's complement approach is used to obtain negative values in signed addition.

This approximation adder is utilized in the first recommended strategy to implement the approximate computing for the candidate functions chosen in the previous subsection. The additional operations in these routines are substituted with the implemented code for approximation LOA adder for this purpose. This would lower both the space and the power consumption. Because the approximate adder introduces some mistake into the output results when compared to the precise adder, compression efficiency is reduced. As a result, we should only employ approximation computing when we have a lot of computer power. In such instances, the amount of power consumption improvement will take precedence over the

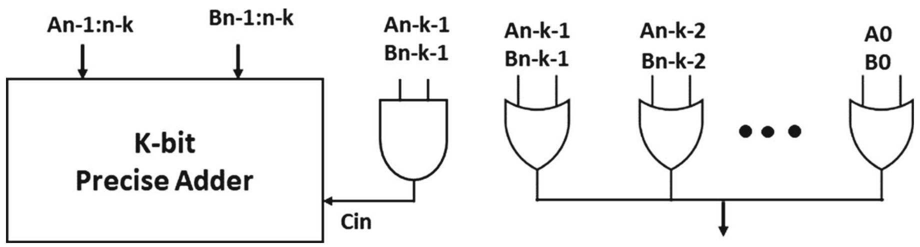


Fig. 4 Lower-Part-OR Adder structure [4]

drop in compression efficiency. To apply this method on the tools chosen in Section 4.1, the parts of their functions where the addition operation is employed in big numbers, should be extracted. As a result, the flowchart of the requested functions is analysed, and the appropriate sections to execute approximation computing are selected accordingly. Figure 5 illustrates the flowchart of the "xPredAffineBlk" function, which was extracted using the Code to Flowchart software [22]. This function has a nested loop, as shown in the flowchart, and the number of times the inner and outer loops are executed is equal to the length and breadth of the frames, respectively, to obtain the prediction vectors block by block. As a result, this nested loop is the most computationally costly section of the function and is an excellent choice for approximate computing. Similarly, other suitable elements for implementing approximation computation in various functions of various coding tools are extracted and implemented using the LOA.

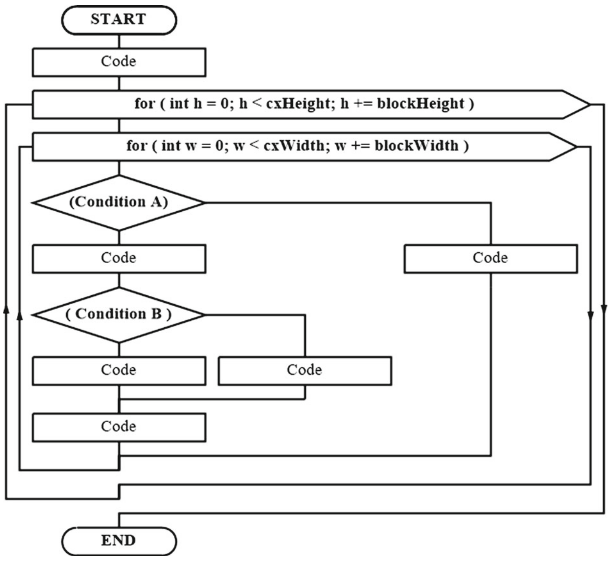

START
Code
for (int h = 0; h < cxHeight; h += blockHeight)
for (int w = 0; w < cxWidth; w += blockWidth)
(Condition A)
Code
Code
(Condition B)
Code
Code
Code
END

Fig. 5 The flowchart of “xPredAffineBlk” function

4.3 Second approach: AxMAP

The second approach employs AxMAP [5] as a framework for generating efficient application-specific approximate adders. As the designers' circuit budget restrictions, the AxMAP tool takes some input parameters as power, area, and delay. Using the random search technique, AxMAP explores the design space and generates approximate adders. The selected design satisfying the restrictions is regarded a legitimate output in each iteration. The recommended approach for using the AxMAP framework is shown in Fig. 6, with six basic building blocks and a basic library containing all possible single-bit approximate adders.

The first stage involves a random search to select the L single-bit samples from the basic library. In the second stage, delay estimation is performed. The longest carry chain of the adder is compared to the input delay constraint in this phase. The adder object is transferred to the next stage if the longest path's latency is less than or equal to the input delay restriction. These processes are repeated until the fourth stage for power and area estimation. This is accomplished by determining the type and number of gates utilized in the adder object. If the power or area exceeds the input restrictions, AxMAP returns to the first stage to try again. Finally, a Verilog HDL Code for the adder object is constructed using a gate-level circuit description. For a more accurate estimation of circuit properties, the produced Verilog file is given to the Design Compiler for synthesis. If the created adder meets all of the input circuit restrictions in this phase, it is a legitimate output adder. The AxMAP framework is implemented in MATLAB using the following input parameters:

- NumOfDesiredDesigns: The number of generated approximate adders.
- Adder Length: The bit-width of adders to be generated by AxMAP.
- Delay Constraint: The maximum length of the carry chain is given as a natural number for the adders generated by AxMAP.
- MED constraints: The maximum amount of error allowed for a specific application is specified as a real number.
- Power and area constraints: The maximum amount of power consumption and area is determined as two real numbers.

This framework is used to create a variety of approximate adders for candidate functions for approximate computing. Then, as indicated in Section 4.1, the appropriate palaces to employ approximate adders are extracted. The implementation of accurate computing is then substituted with the AxMAP framework's implementation of approximation adders.

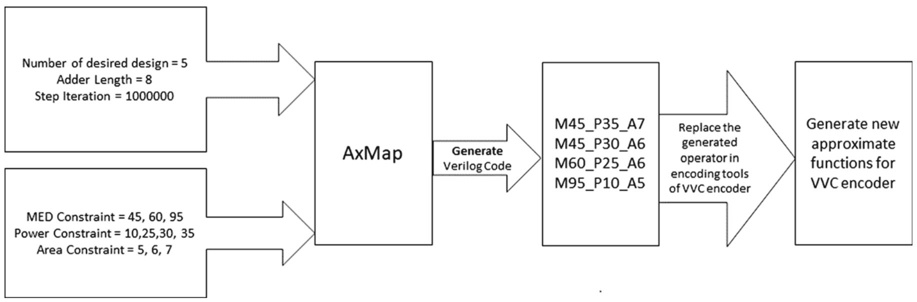

Number of desired design = 5
Adder Length = 8
Step Iteration = 1000000
MED Constraint = 45, 60, 95
Power Constraint = 10,25,30, 35
Area Constraint = 5, 6, 7
AxMap
Generate Verilog Code
M45_P35_A7
M45_P30_A6
M60_P25_A6
M95_P10_A5
Replace the generated operator in encoding tools of VVC encoder
Generate new approximate functions for VVC encoder

Fig. 6 The proposed approach to use AxMAP framework to produce approximate adders [5]

Table 5 The BD-bitrate degradation of each encoding function using the approximate computing over the baseline method for the Blowingbubble test sequence

Encoding functions	coding tool	BD-rate degradation(%)
XAffineMotionEstimation	AFFINE	0
XPredAffineInterSearch	AFFINE	0
XPredAffineBlk	AFFINE	0.59
XCheckRDCostAffineMerge2Nx2N	AFFINE	0
ApplyBiOptFlow	BIO	3.15
XSubPuBio	BIO	0.36
Average		0.68

5 Evaluation

We develop precise tests to evaluate the area, power, and performance potentials of applying approximation computing to various functions of the VVC encoder.

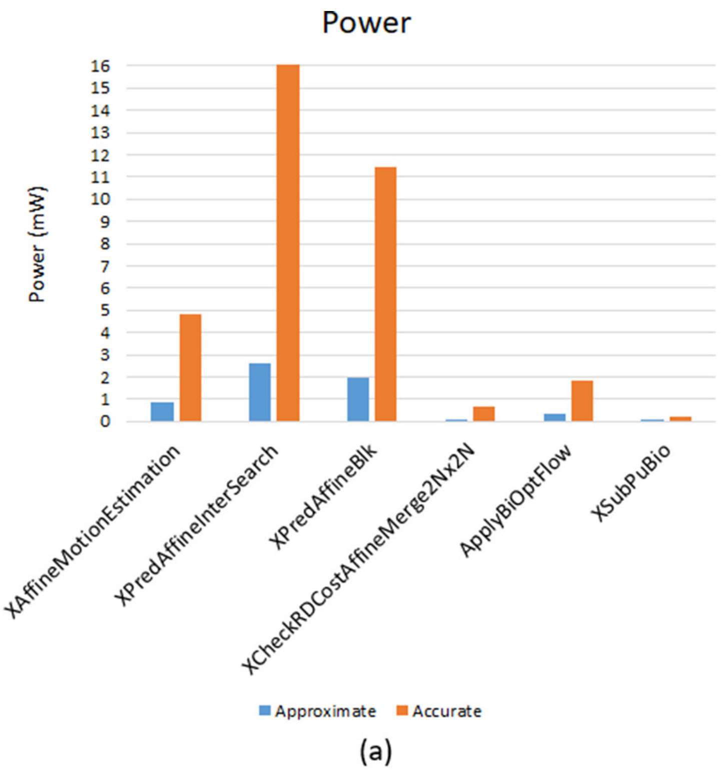
5.1 Experimental setup

The video sequences in Table 1 are used to demonstrate the performance of our suggested technique. We encoded the test sequences with four distinct QP values, 22, 27, 32, and 37, using the VTM reference software version 6.3. The coding efficiency of the test sequences in which approximation computing is used to accomplish various encoder tasks is then extracted. The findings are compared using the Bjntegaard-Delta bitrate (BD-bitrate) metric to the original VTM reference software results as the baseline. Furthermore, the area and power consumtion of approximate and accurate encoders are considered as a measure of computational complexity.

5.2 First approach: Lower-part-OR Adder (LOA)

We ran several tests to see how the first technique performed in terms of area, power, and coding performance. The LOA adder is implemented and used in the candidate functions of AFFINE, DMVR, and BIO coding tools for this purpose, as shown in Table 5 for the Blowingbubble test sequence. while compared to the baseline approach, the results of this table demonstrate a modest performance degradation in compression efficiency while utilising LOA adder.

The results of improving area and power using LOA method for various candidate fucnctions of each coding tool are presented in Fig. 7. The VHDL version of the encoding routines is constructed to analyse the area and power, and these metrics are generated using the design _vision tool [23]. Using approximation computation, the results demonstrate a significant improvement in area and power.



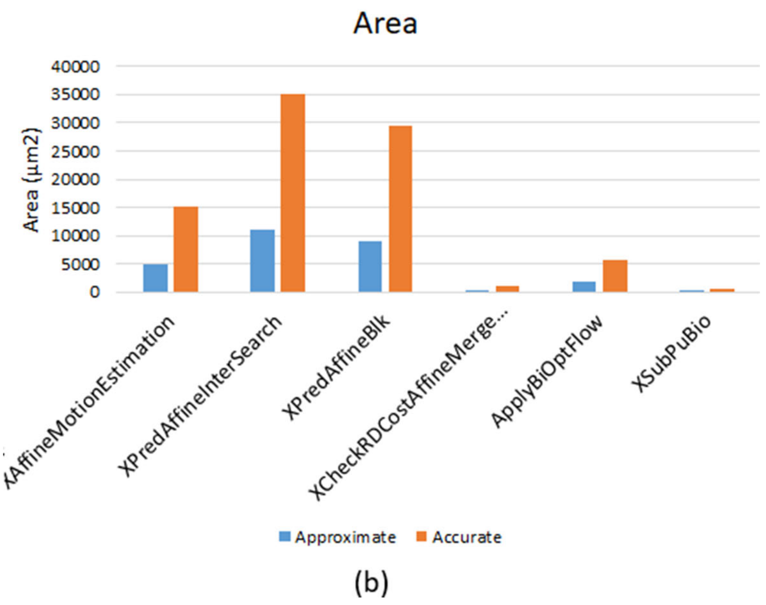
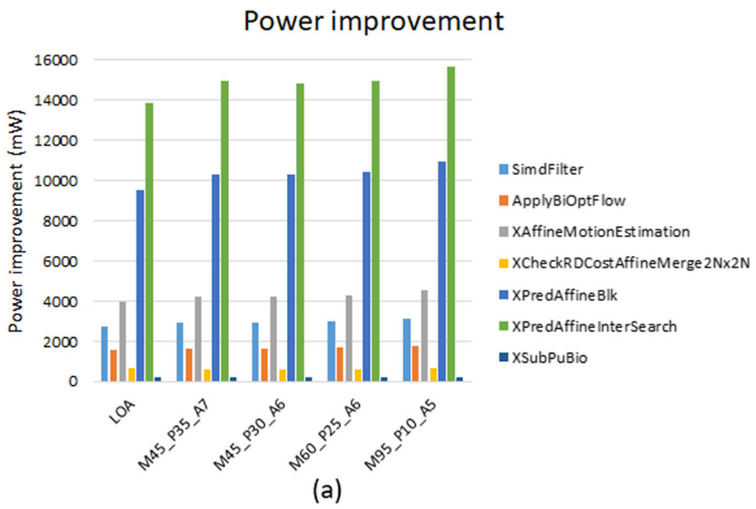


Fig. 7 The results of (a) power and (b) area improvement of applying approximate computing to candidate functions compared to the baseline method

Table 6 The various parameters of the AxMAP framework that is used to generate approximate adder circuits

NumofDesiredDesign	5
Adder Length	8
Power constraints	10, 25, 30, 35, 40
Area constraints	5, 6, 7, 8
MED constraints	45, 50, 60, 95



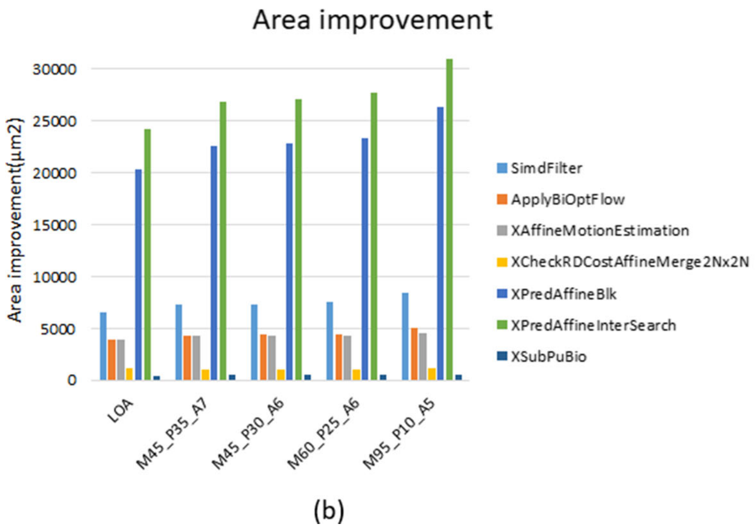


Fig. 8 The results of (a) power and (b) area improvement of applying approximate computing using AxMAP framework for various functions compared to the LOA method

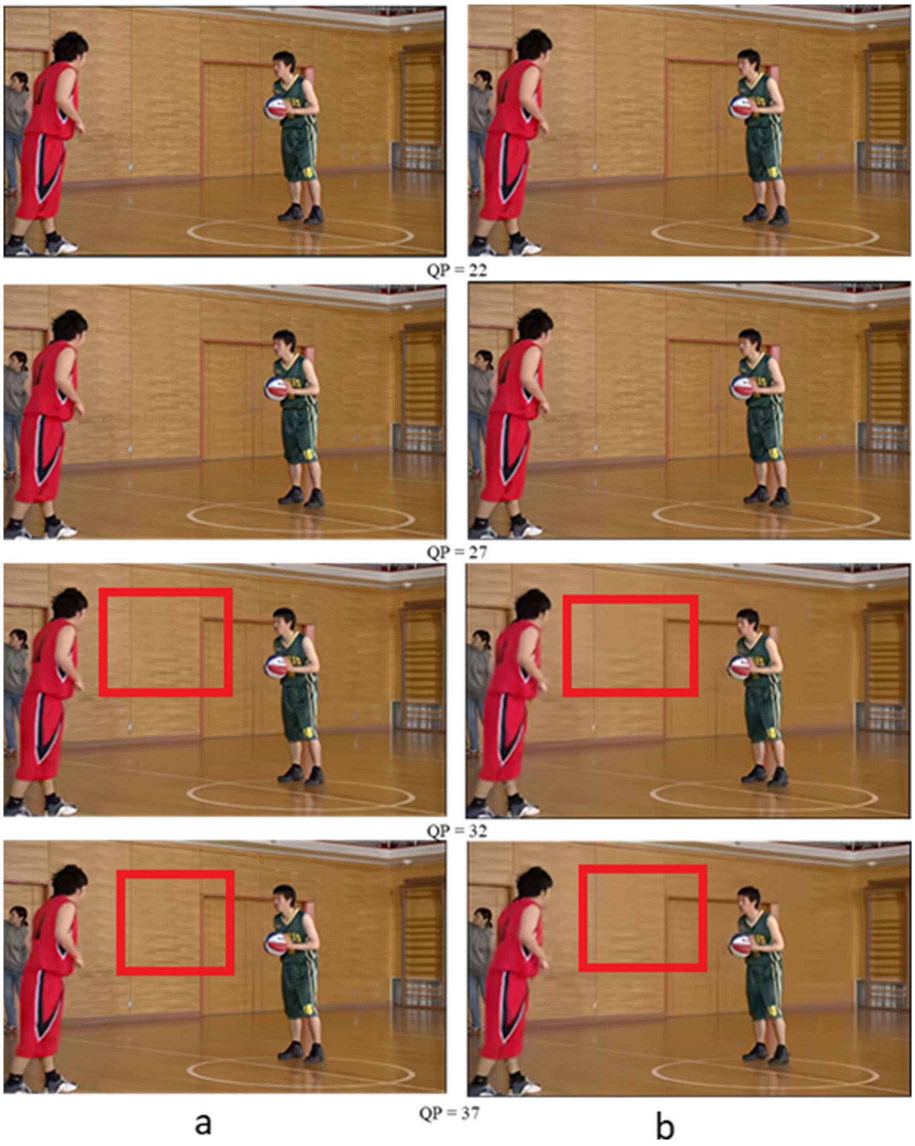


Fig. 9 The comparison of the subjective quality of the encoded video using the VVC encoder with (a) accurate computing and (b) approximate computing using LOA

5.3 Second approach: AxMAP

The various parameters in these trials are established as shown in Table 6. A wide range of factors are used to produce thorough results. For example, for power constraints, the numbers 10, 25, 30, 35, and 40 are evaluated.

Table 7 The amount of power reduction and compression performance in [24] and [25], as well as our proposed methodology over accurate VVC standard as the baseline for the functions related to filter operations

	The method proposed in [24]	The method proposed in [25] (First Approach)	The method proposed in [25] (Second Approach)	Our proposed method
Total power reduction	39.84%	56.82%	59.06%	82%
BD-rate	0.03%	1.6%	2.01%	3.81%

Using these parameters, we generated five different approximate adders: M50_P40_A8, M45_P35_A7, M45_P30_A6, M60_P25_A6, and M60_P25_A6. The numbers in these names represent the inaccuracy, power consumption, and area limits specified in the design process using the AxMAP framework, respectively. The produced approximation adders were then applied to candidate functions for approximate computation in the DMVR, BIO, and AFFINE coding tools. The results of improving area and power utilising the various created approximate adders are presented in Fig. 8. The results reveal that using approximate computing to VVC coding tools reduces power and area by 9.67x and 5x, respectively, as compared to the original VTM reference programme as the baseline. Furthermore, the AxMAP framework-generated adders outperform the LOA technique in terms of area and power consumption by 14.88% and 9.11%, respectively.

To demonstrate that using approximate computing does not significantly change the subjective quality of the decoded videos, the decoded videos obtained from the accurate computation in the VVC encoder are compared with the decoded videos obtained from the approximate computation, as shown in Fig. 9. Only the features of the frames, such as vertical lines on walls, are warped at low bit rates due to approximation computations.

Finally, we compared our findings to the methodologies described in [24] and [25]. According to [24], the fractional interpolation hardware implemented for VVC employs 15 fractional interpolation filters in parallel and has a higher computational cost. As a result, this work implements an approximate VVC fractional interpolation hardware that takes up less space and consumes less power than VVC fractional interpolation hardware. Two ways are provided in [25] to improve the power consumption of fractional interpolation filters in VVC. In the first approach, an approximate fractional interpolation filter for VVC and its baseline hardware (BF) is provided, which uses adders and shifters to implement multiplications with filter coefficients. The parallel processing is then employed to increase the performance of the suggested BF in the second way. Table 7 compares the power consumption for our proposed strategy employing the AxMAP framework's approximate adders with the method presented in [24] and [25]. Because the approaches of these two reference papers only focused on the approximate implementation of filters, only the outcomes of the functions related to the filters were considered for comparison to fairly compare the results of our proposed method with the reference ones.

6 Conclusion

The use of approximation computing to minimize the computational complexity of the VVC video coding standard is proposed in this study. The study begins by analyzing the computational complexity of the new video coding tools introduced in the standard in order to identify the most computationally complex functions of various coding tools as the best candidates for approximation computing implementation. The approximate computation is then applied to the adder circuits of these candidate functions using the LOA approximate adder. Because addition is the most commonly used operation in coding tools and consecutive additions can be utilized to build additional mathematical operations such as multiplication and subtraction. Our evaluation results show that utilizing the proposed approach, we can achieve 9.67x and 5x power and area reductions compared to the original VVC as the baseline, with no discernible impact on subjective quality. The AxMAP framework is then examined to build more efficient approximate adders in order to improve area and power consumption over the LOA technique by 14.88% and 9.11%, respectively.

Data Availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflicts of interest There is no conflict of interests regarding the publication of this manuscript.

References

1. Bross B, Wang YK, Ye Y, Liu S, Chen J, Sullivan GJ, Ohm JR (2021) Overview of the versatile video coding (vvc) standard and its applications. IEEE Trans Circuits Syst Video Technol 31(10):3736–3764
2. S. K, D. M, G. P (2013) A gpu based real-time video compression method for video conferencing. In: 18th International conference on digital signal processing (DSP), pp 1–6
3. Paim G, Rocha LMG, Amrouch H, da Costa EAC, Bampi S, Henkel J (2020) A cross-layer gate-level-to-application co-simulation for design space exploration of approximate circuits in hevc video encoders. IEEE Trans Circuits Syst Video Technol 30(10):3814–3828
4. Tajasob S, Rezaalipour M, Dehyadegari M (2019) Designing energy-efficient imprecise adders with multi-bit approximation. Microelectronics J 89:41–45
5. Rezaalipour M, Rezaalipour M, Dehyadegari M, Bojnordi MN (2020) Axmap: Making approximate adders aware of input patterns. IEEE Trans Comput 69(6):868–882
6. Li L, Li H, Liu D, Yang H, Lin S, Chen H, Wu F (2017) An efficient four-parameter affine motion model for video coding. IEEE Trans Circuits Syst Video Technol 28:1934–1948
7. Chen X, An J, Zheng J (2016) Decoder-side motion vector refinement based on bilateral template matching. In: Joint video exploration team of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, 4th Meeting
8. Aramvith S, Sun M-T (2009) The essential guide to video processing (second edition)
9. E. RI (2003) The h.264 avanced video compression standard 2nd edition
10. Li X, Amon P, Hutter A, Kaup A (2009) Lagrange multiplier selection for rate-distortion optimization in svc. Picture Coding Symposium (PCS)
11. Domanski R, Kolodziejski W, Correa G, Porto M, Zatt B, Agostini L (2021) Low-power and high-throughput approximated architecture for av1 fme interpolation. In: 2021 IEEE International symposium on circuits and systems (ISCAS)
12. Stigger MF, Lima VHS, Soares LB, Diniz CM, Bampi S (2020) Approximate satd hardware accelerator using the 8 $\times$ 8 hadamard transform. In: 2020 IEEE 11th Latin american symposium on circuits and systems (LASCAS)
13. Azgin H, Kalali E, Hamzaoglu I (2020) An approximate versatile video coding fractional interpolation hardware. In: 2020 IEEE International conference on consumer electronics (ICCE)
14. Porto R, Agostini L, Zatt B, Roma N, Porto M (2019) Power-efficient approximate sad architecture with loa imprecise adders. In: 2019 IEEE 10th Latin american symposium on circuits and systems (LASCAS)
15. Ahmad W, Ayrancioglu B, Hamzaoglu I (2020) Comparison of approximate circuits for h.264 and hevc motion estimation. In: 2020 23rd Euromicro conference on digital system design (DSD)
16. da Silva R, Siqueira I, Grellert M (2019) Approximate interpolation filters for the fractional motion estimation in hevc encoders and their vlsi design. In: 32nd Symposium on integrated circuits and systems design (SBCCI)

17. Mahdavi H, Azgin H, Hamzaoglu I (2021) Approximate versatile video coding fractional interpolation filters and their hardware implementations. IEEE Trans Consum Electron 67(3):186–194
18. Jayakodi J, Sagadevan K (2016) Arithmetic unit based reconfigurable approximation technique for video encoding. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 5(4)
19. Bossen F, Boyce J, Li X, Seregin V, Sühring K (2020) Vtm common test conditions and software reference configurations for sdr video. Joint Video Experts Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29, 20th Meeting, by teleconference
20. Vvc. https://jvet.hhi.fraunhofer.de/trac/vvc/browser/vtm
21. Intel vtune profiler. https://software.intel.com/content/www/us/en/develop/documentation/vtunehelp/top.html
22. Code flow. https://code2flow.com
23. Design vision tool. https://www.mics.ece.vt.edu/ICDesign/Tutorials/Synopsys/Verilog2.html
24. Azgin H, Kalali E, Hamzaoglu I (2020) n approximate versatile video coding fractional interpolation hardware. In: IEEE International conference on consumer electronics (ICCE)
25. Mahdavi H, Azgin H, Hamzaoglu I (2021) Approximate versatile video coding fractional interpolation filters and their hardware implementations. IEEE Trans Consum Electron 67(3):186–194

Authors and Affiliations

Sina Shah Oveisi[1] · Hoda Roodaki[1,3] · Morteza Rezaalipour[2] · Masoud Dehyadegari[1,3]

Sina Shah Oveisi
shahoveisysina@email.kntu.ac.ir

Morteza Rezaalipour
morteza.rezaalipour@usi.ch

Masoud Dehyadegari
dehyadegari@kntu.ac.ir

[1] Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran

[2] Universitá della Svizzera italiana, Lugano, Switzerland

[3] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran