Approximate Subtractor Operator for Low-Power Video Coding Hardware Accelerators

Rafael Ferreira*, Mateus Leme*, Marcel Corrêa*, Luciano Agostini*, Cláudio Diniz†, Bruno Zatt*
*Graduate Program on Computer Science, Federal University of Pelotas (UFPel), Pelotas, Brazil
†Graduate Program on Electronic Engineering and Computing, Catholic University of Pelotas (UCPel), Pelotas, Brazil

***Abstract*—Video coding requires a massive computational effort leading to large power dissipation and energy consumption. Thus, energy efficiency becomes a significant concern especially under limited energy resources such as in mobile devices. Approximate computing is a promising technique to improve energy efficiency. Therefore, this work presents a new approximate subtractor operator to be used in video coding hardware accelerators. The proposed subtractor reduces power of a Sum of Absolute Differences (SAD) hardware accelerator on approximately 10.39% (on average of different videos) when compared to the use of the subtractor from the synthesis tool. It also presents a power reduction of 18.13% when compared to state-of-the-art approximate adder.**

***Index Terms*—Video Coding; Hardware Architecture; Subtractor; Approximate; Imprecise; SAD; HEVC.**

I. Introduction

The High Efficiency Video Coding (HEVC) standard [1] was developed to meet the demands for high-resolution video services. Compared to H.264/AVC (Advanced Video Coding) standard [2], HEVC delivers encoded video with similar quality at nearly half the bit rate. This benefit was reached due to the new coding tools and partitioning structures of HEVC [3] that increase by up to 50% the encoder complexity. This additional complexity translates into high power and energy of HEVC encoders, which is a severe issue for energy-constrained devices. Developing dedicated low power hardware architectures for video coding modules is crucial for achieving power-/energy-efficient solutions.

A significant source of video encoding complexity comes from the calculation of Sum of Absolute Differences (SAD). The calculation of SAD and other distortion metrics accounts for about 40% of the total execution time of the reference HEVC video encoder [4]. SAD is the most used metric to determine the best match between two video blocks in the Integer Motion Estimation (IME) process. The SAD calculation is given by the sum of absolute sample-by-sample differences between the block to be encoded and a reference block (from a set of candidate blocks). SAD calculation is shown in Eq. (1), in which O is the original block, R is the reference block, i and j are the sample positions, and m and n are the dimensions of the block in samples.

$$SAD = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |O_{i,j} - R_{i,j}| \quad (1)$$

Approximate computing [5] is a promising approach to reduce computational effort and energy consumption while reducing application quality/precision. This approach has been applied to applications that are resilient to quality loss, e.g., video coding already apply lossy compression algorithms. One of the approximate computing techniques is to design approximate arithmetic operators that are prone to errors but dissipate less power than precise operators [6].

This work proposes a new approximate subtractor operator suitable to be applied to SAD architectures. SAD architectures employ subtractors to calculate the sample-by-sample difference of the video blocks. The proposed approximate subtractor operator is implemented in hardware description language and evaluated inside an 8-input sample SAD hardware architecture considering distinct levels of imprecision (from 1 to 4 bits). Through the HEVC reference software and the model implementation of the proposed approximate subtractor, it was possible to perform the quality loss evaluation.

Before moving to the main contribution of this work (Section III), Section II reviews related work. Section IV shows the results and discussion. Section V concludes the work.

II. Related Work

Previous works have proposed SAD architectures using approximate adders. Porto *et al.* [7] evaluated the use of 6 approximate adders to replace the subtraction operations in the SAD computation of HEVC Test Model (HM) encoder software [8]. Lower-Part-OR Adder (LOA) [9] with 3 precise and 5 imprecise bits achieved the best results in this analysis and also in a multi-variable evaluation of the 6 adders considering Bjontegaard Delta Rate (BD-Rate) increase, delay, area, power, and power delay product. LOAs with 3 precise and 5 imprecise bits were applied to the subtractors of the SAD architecture with 16-input pixel block. Paltrinieri *et al.* [10] have also evaluated 5 approximate adders into 3 levels of the SAD architecture. This work has also concluded that LOA is one of the most power-efficient approximate adders. A comparative review on approximate adders [6] has also highlighted the power efficiency of LOA. The works [7], [10] have not considered the use of approximate subtractors.

Approximate subtractors are also proposed in previous works, mainly applied as part of approximate dividers. Chen *et al.* [11] proposed three subtractor circuits employing pass transistor logic. Reddy *et al.* [12] proposed three subtractor circuits, but they used conventional CMOS logic. The works in [11], [12] have not evaluated the use of these subtractors in SAD architectures. Therefore, there is a promising research

978-1-7281-0996-1/19/$31.00 ©2019 IEEE

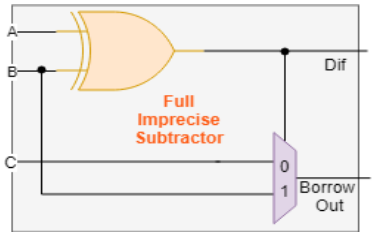

Fig. 1. Proposed 1-bit approximate subtractor (Apps).

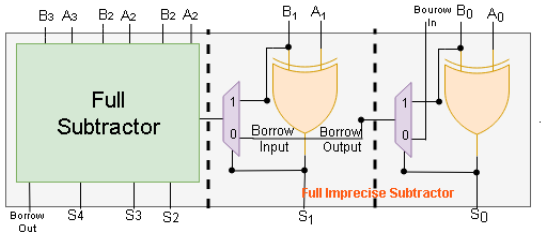

Fig. 2. An example of proposed approximate subtractor of 4-bit operands with 2 imprecise bits.

space related to approximate subtractors, specially when considering energy-constrained video processing systems.

III. Proposed Approximate Subtractor and SAD Accelerator

Fig. 1 shows the diagram of the proposed 1-bit approximate subtractor. It is composed of one XOR gate, that produces output S with the difference from input operands A and B, and a multiplexer that generates the borrow output from the input operand B and the input C (borrow-in). Throughout the paper the proposed subtractor will be identified by **AppS** (Approximate Subtractor). Fig. 2 depicts an example of the proposed approximate subtractor when considering 4-bit input operands. Similar to approximate full adders (as reviewed in [6]), the proposed subtractor is composed of two parts: a precise part, that generates the most significant bits of the subtraction, and the imprecise part, that generates the least significant bits. The example in Fig. 2 shows an approximate subtractor with 2 imprecise bits. The imprecise part consists of two imprecise subtractors. The inputs of the least significant bit, with inputs A0, B0, and borrow-in, generate a difference bit and borrow-out that is connected to the borrow-in of the second subtractor with inputs A1 and B1. The precise part is composed of a precise subtractor with the borrow-out output obtained from the second approximate subtractor.

Table I shows the truth table of precise and imprecise subtractors. Considering the 8 combinations for the 1-bit subtraction, the proposed approximate subtractor presents error in difference bit in 4 cases. However, the borrow out is correct in all cases. Hence, the path that generates the borrow signal is precise. This behavior is desirable as it limits the error propagation in the borrow chain.

We have compared the percentage of correct operations of our subtractor to LOA [9] and other 6 approximate state-of-the-art subtractors [11], [12]. The subtraction operation using

TABLE I
Subtractor Truth Table

			Precise		Imprecise	
A	B	Borrow In	Diference	Borrow Out	Diference	Borrow Out
0	0	0	0	0	0 ✓	0 ✓
0	0	1	1	1	0 ×	1 ✓
0	1	0	1	1	1 ✓	1 ✓
0	1	1	0	1	1×	1 ✓
1	0	0	1	0	1 ✓	0 ✓
1	0	1	0	0	1 ×	0 ✓
1	1	0	0	0	0✓	0 ✓
1	1	1	1	1	0×	1 ✓

LOA needs a two's complement operation in the second input. An exhaustive subtraction test was conducted considering 8-bit input operands A and B resulting in a total of 65,536 subtractions. Four levels of imprecision were evaluated, i.e. using 1 to 4 imprecise least significant bits. This evaluation was done by in-house C++ simulator using bitwise library. Table II shows the results of this evaluation. Our proposed subtractor with 1 imprecise bit produces correct outputs in 100% of the cases. That happens because errors only appear when the borrow-in signal is equal to one and such case does not happen for the first bit (see Table II). Considering all levels of imprecision our subtractor has the higher percentage of correct operations compared to other approximate operators. Our subtractor produces a correct output in 65.62% of the cases considering an average of the 4 levels of imprecision. Our subtractor has a more regular result than other operators: each increase in level of imprecision increases error by approximately 13.33%. Hence, the behavior of the subtractor can be predicted depending on the level of imprecision

The proposed subtractor is implemented into a SAD accelerator. Fig. 3 presents the diagram of the SAD accelerator used in this work. It is composed of eight subtractors to calculate the difference between eight original and eight reference samples (O and R). Eight absolute operators receive the differences as input and deliver the absolute values to an adder tree that accumulates them to generate the final SAD value. The subtractors (focus of this work) are highlighted in Fig. 3.

IV. Results and Discussion

A. Coding Efficiency Results

Before moving to the power results of the SAD accelerator using our proposed approximate subtractor, we show the im-

TABLE II
Correct operations (%)

Operator	Level of imprecision				Avg.
	1	2	3	4	
AppS	100	75	56.25	41.18	65.62
LOA [9]	75	56.25	42.18	31.64	49.21
AXSC1 [11]	75	56.25	42.18	31.64	49.21
AXSC2 [11]	50	25	12.5	6.25	18.75
AXSC3 [11]	50	25	12.5	6.25	18.75
AXS1 [12]	50	31.25	18.75	11.32	25.00
AXS2 [12]	50	25	12.5	6.25	18.75
AXS3 [12]	75	56.25	42.18	31.64	49.21

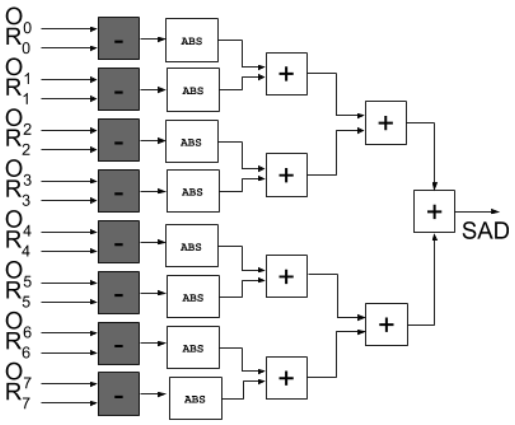

Fig. 3. Diagram of the SAD accelerator used in this work

pact on coding efficiency (BD-Rate) of video encoding when applying the proposed approximation. The experiments were performed on the HM 16.9 [8] with video sequences recommended by the Common Test Conditions (CTCs) [13]. A total of 60 frames of the following video sequences were encoded using intra main configuration: Beauty (3840x2160), Steam Locomotive Train (2560x1600), BQTerrace (1920x1080), and Four People (1280x720). Each SAD function (for all block sizes) was modified by replacing the precise subtraction operation by a software version of the approximate operators AppS (our proposed subtractor) and LOA.

Table III shows the BD-Rate increase (in %) of the approximate operators (AppS and LOA) considering 4 imprecision levels. We can note a low coding efficiency variantion for Beauty, BQTerrace, and Steam Locomotive videos with up to 2 imprecise bits. When 3 and 4 imprecise bits are considered, larger variations are observed. However, the BD-Rate is negative for some cases, which means coding efficiency improvements. The Four People sequence shows a distinct behavior. For 1 and 2 imprecise bits the coding efficiency is significantly improved whereas for 3 and 4 imprecise bits a severe coding efficiency degradation is observed for both AppS and LOA. On average, our proposed subtractor presents about 1.1% worse coding efficiency than LOA.

TABLE III
BD-RATE (Y CHANNEL)

Operator	Sequence	1 Bit	2 Bit	3 Bit	4 Bit	Avg
LOA [9]	Beauty	1.555	2.957	-3.971	-2.965	**-2.965**
	BqTerrace	-0.684	-0.541	-0.266	-3.063	**-1.139**
	Steam Locomotive	-0.684	0.864	0.957	0.533	**0.418**
	Four People	0.824	-2.651	-1.938	-1.547	**-1.326**
	Avg	**0.252**	**0.3145**	**-5.218**	**-1.304**	**-1.253**
AppS	Beauty	0.532	0.881	1.228	1.600	**1.060**
	BqTerrace	-0.295	0.277	-0.414	-0.048	**-0.120**
	Steam Locomotive	0.824	0.561	-2.388	-4.514	**-1.379**
	Four People	-2.279	-0.886	0.050	1.464	**-0.413**
	Avg.	**-0.305**	**0.833**	**-0.762**	**-0.749**	**-0.213**

B. Synthesis Results

Approximate operators were described in Verilog and synthesized to CMOS 65 nm TSMC library using Cadence RTL Compiler. Table IV shows the power comparison of our subtractor compared to other 8-bit-width approximate operators considering the 4 levels of imprecision (1 to 4 bits). The target frequency for all operators is set to 666.66 MHz and the circuits were stimulated exhaustively with 65,536 inputs.

Compared with LOA, the proposed subtractor has a small increase in average power dissipation (Table IV), while achieving a significant increase in correct operations. Compared to other approximate subtractors, the power dissipation is substantially increased, but since percentage of the correct operations is very low, it will be less useful for video coding applications.

TABLE IV
CELL, AREA (μm^2) AND POWER (μW) RESULTS OF APPROXIMATE SUBTRACTORS

Operator		Level of inaccuracy 1	2	3	4	Avg.
AppS	**Power**	888	876	868	810	860
	Cell	72	72	68	69	70
	Area	379	376	376	355	371
LOA [9]	**Power**	1089	881	769	663	850
	Cell	84	77	73	63	74
	Area	456	390	349	320	378
AXSC1 [11]	**Power**	868	868	840	864	860
	Cell	68	70	68	71	69
	Area	376	372	363	366	369
AXSC2 [11]	**Power**	846	758	739	626	742
	Cell	66	66	64	54	62
	Area	363	341	333	315	338
AXSC3 [11]	**Power**	807	776	644	657	721
	Cell	63	65	53	56	59
	Area	352	339	300	301	323
AXS1 [12]	**Power**	844	868	666	510	722
	Cell	66	70	60	50	61
	Area	362	372	308	263	326
AXS2 [12]	**Power**	752	617	589	510	617
	Cell	68	55	54	50	56
	Area	341	299	291	263	298
AXS3 [12]	**Power**	807	708	639	550	676
	Cell	63	64	59	55	60
	Area	352	317	293	267	307

Nine versions of the SAD architecture were described in Verilog. All architectures versions were synthesized to 65 nm library with 166.66 MHz target frequency. The first version of SAD architecture uses standard arithmetic operators from the synthesis tool. The other versions were implemented by modifying all subtractors of the SAD architecture by our approximate subtractor and LOA [9], considering four imprecision levels (1 to 4 imprecise bits, increasing the imprecise level from the least to the most significant bits). LOA works as subtractor because one of its inputs uses a two's complement operator. The power estimation consists of first extracting from HM the inputs from SAD functions. A 10,000-input vector was used to stimulate the architecture in Verilog generating a Value Change Dump (VCD) file that represents switching activity. One VCD file was generated for each architecture version and each video sequence. Design files and the VCD file are delivered to the Cadence RTL Compiler to obtain power reports for each architecture and each video.

Table V presents power results detailing the leakage and

dynamic power breakdown. The last two columns present the power reduction of our subtractor when compared to the precise operator from synthesis tool and LOA [9], respectively. Considering all evaluated precisions, our proposed approximate subtractor presents an average power reduction of 18.13% when compared to LOA and 10.39% when compared to the tool operator. Power reduction was observed for the most of the cases reaching up to 41% and 22% when compared with LOA and the tool operator, respectively. Power increase was observed for 1-bit imprecision when compared to the tool operator. When compared to LOA, power increase was observed for only a few cases using 3-bit (BQTerrace) and 4-bit (BQTerrace and Steam Locomotive) imprecision.

TABLE V
POWER RESULTS (μW) OF SAD ACCELERATOR USING APPROXIMATE OPERATORS

Video	Version	Leakage	Dynamic	Total	Tool	LOA [9]
Beauty 3840x2160	Tool	1.767	611.224	612.991		
	LOA01 [9]	3.207	869.609	872.816	-15.07	-40.35
	AppS01	1.944	518.671	520.614		
	LOA02 [9]	2.899	766.584	769.483	-15.91	-33.01
	AppS02	1.887	513.556	515.443		
	LOA03 [9]	2.745	657.411	660.156	-16.41	-22.38
	AppS03	2.012	510.402	512.415		
	LOA04 [9]	2.494	585.805	588.299	-16.89	-13.40
	AppS04	2.212	507.257	509.469		
SteamLocomotiveTrain 2560x1600	Tool	1.772	501.411	503.183		
	LOA01 [9]	3.243	690.036	693.279	-2.23	-29.04
	AppS01	1.953	490.006	491.958		
	LOA02 [9]	2.923	591.981	594.903	-3.87	-18.69
	AppS02	1.897	481.81	483.708		
	LOA03 [9]	2.765	493.05	495.815	-3.63	-2.20
	AppS03	2.025	482.891	484.917		
	LOA04 [9]	2.514	430.231	432.745	-5.44	9.96
	AppS04	2.218	473.617	475.835		
BQTerrace 1920x1080	Tool	1.761	265.539	267.3		
	LOA01 [9]	3.216	311.29	314.506	0.16	-14.87
	AppS01	1.986	265.754	267.74		
	LOA02 [9]	2.907	288.537	291.444	-1.27	-9.45
	AppS02	1.914	261.993	263.906		
	LOA03 [9]	2.742	258.883	261.625	-1.29	0.85
	AppS03	2.07	261.783	263.852		
	LOA04 [9]	2.488	240.071	242.559	-2.53	7.42
	AppS04	2.236	258.313	260.549		
Johnny 1280x720	Tool	1.779	441.299	443.078		
	LOA01 [9]	3.174	595.064	598.238	-20.91	-41.42
	AppS01	1.951	348.482	350.433		
	LOA02 [9]	2.866	538.073	540.939	-22.22	-36.29
	AppS02	1.900	342.732	344.632		
	LOA03 [9]	2.722	484.895	487.617	-20.89	-28.11
	AppS03	2.041	348.49	350.53		
	LOA04 [9]	2.478	446.883	449.361	-17.91	-19.06
	AppS04	2.233	361.471	363.704		
				Avg.	-10.39	-18.13

TABLE VI
CELL AND AREA (μm^2) RESULTS OF SAD ACCELERATOR USING APPROXIMATE OPERATORS

Version	Cells	Tool	LOA [9]	Area	Tool	LOA [9]
Tool	608	23.19	-38.61	3152	6.03	-27.72
LOA01 [9]	1220			4624		
AppS01	749			3342		
LOA02 [9]	1170	27.63	-33.68	4372	6.44	-23.26
AppS02	776			3355		
LOA03 [9]	1121	21.55	-34.08	4268	5.52	-22.07
AppS03	739			3326		
LOA04 [9]	1062	28.45	-16.46	4065	9.84	-14.83
AppS04	781			3462		
	Avg.	**25.21**	**-33.20**	**Avg.**	**6.96**	**-21.97**

Table VI details the number of cells and area. Observe that our subtractor leads to area increase when compared to the tool operator. It is expected since the tool operator uses optimized cells from the library. In turn, LOA [9] spends significantly more area when compared to the other solutions. This is due to the need for an additional two's complement operator. Overall, the proposed subtractor leads to important power reduction associated to a competitive coding efficiency when compared to the state-of-the-art LOA [9].

It is difficult to conduct a fair comparison of the results for SAD architectures using our approximate subtractor with the ones presented in [7], [10]. The related works employ different architectures, with different number of input samples, frequencies, and CMOS technologies.

V. CONCLUSION

This paper presents a novel approximate subtractor operator and its application to a SAD hardware architecture. Results in terms of coding efficiency, hardware area, and power dissipation demonstrate that our proposed subtractor is a promising solution for energy-efficient video processing systems. When applied to the SAD architecture, our subtractor enabled 10.39% power reduction compared to the subtractor generated by the synthesis tool and 18.13% power reduction compared to LOA. The coding efficiency is similar to that observed when using LOA.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, and also by CNPq and FAPERGS Brazilian research support agencies.

REFERENCES

[1] ITU-T and ISO/IEC, "High efficiency video coding," *ITU-T Recommendation H.265 and ISO/IEC 23008-2*, 2013.
[2] ITU-T and ISO/IEC JTC 1, "Advanced video coding for generic audiovisual services," *ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4 AVC)*, 2011.
[3] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.
[4] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, Dec 2012.
[5] Sparsh Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 62, 2016.
[6] Honglan Jiang, Jie Han, and Fabrizio Lombardi, "A comparative review and evaluation of approximate adders," in *GLSVLSI*, New York, NY, USA, 2015, GLSVLSI '15, pp. 343–348, ACM.
[7] Roger Porto, Luciano Agostini, Bruno Zatt, Nuno Roma, and Marcelo Porto, "Power-efficient approximate SAD architecture with LOA imprecise adders," in *LASCAS*. IEEE, 2019, pp. 65–68.
[8] "HEVC Test Model (HM) v. 16.9," http://HEVC.hhi.fraunhofer.de.
[9] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, April 2010.
[10] Alberto Paltrinieri, Riccardo Peloso, Guido Masera, Muhammad Shafique, and Maurizio Martina, "On the effect of approximate-computing in motion estimation," *Journal of Low Power Electronics*, vol. 15, no. 1, pp. 40–50, 2019.
[11] Linbin Chen, Jie Han, Weiqiang Liu, and Fabrizio Lombardi, "Design of approximate unsigned integer non-restoring divider for inexact computing," in *GLSVLSI*. ACM, 2015, pp. 51–56.
[12] K Manikantta Reddy, MH Vasantha, YB Nithin Kumar, and Devesh Dwivedi, "Design of approximate dividers for error tolerant applications," in *MWSCAS*. IEEE, 2018, pp. 496–499.
[13] F. Bossen, "Common test conditions and software configurations," in *JCT-VC document no. L1100*, Jan 2013.