Embracing Approximate Computing for Energy-Efficient Motion Estimation in High Efficiency Video Coding

Walaa El-Harouni, Semeen Rehman[3], Bharath Srinivas Prabakaran[3], Akash Kumar[3],
Rehan Hafiz[2], Muhammad Shafique[1]

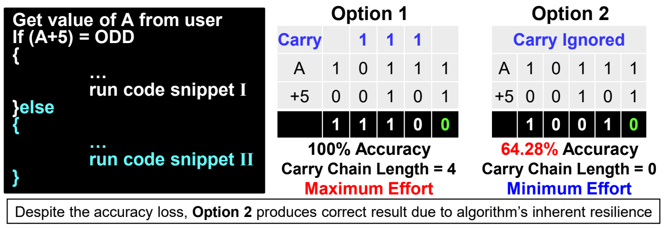
[1]Institute of Computer Engineering, Vienna University of Technology (TU Wien), Austria
[2]National University of Sciences and Technology, Islamabad, Pakistan
[3]Chair for Processor Design, TU Dresden, Germany
Corresponding Authors: muhammad.shafique@tuwien.ac.at, semeen.rehman@tu-dresden.de

***Abstract*— Approximate Computing is an emerging paradigm for developing highly energy-efficient computing systems. It leverages the inherent resilience of applications to trade output quality with energy efficiency. In this paper, we present a novel approximate architecture for energy-efficient motion estimation (ME) in high efficiency video coding (HEVC). We synthesized our designs for both ASIC and FPGA design flows. ModelSim gate-level simulations are used for functional and timing verification. We comprehensively analyze the impact of heterogeneous approximation modes on the power/energy-quality tradeoffs for various video sequences. To facilitate reproducible results for comparisons and further research and development, the RTL and behavioral models of approximate SAD architectures and constituting approximate modules are made available at https://sourceforge.net/projects/lpaclib/.**

***Keywords*— Approximate Computing, Hardware Accelerator, Motion Estimation, HEVC, Video Coding, Energy Efficiency.**

I. Introduction and Motivation

Approximate computing has recently gained a lot of interest by the industrial (Intel [1], IBM [2], Microsoft [3]) and academic research groups [4][5] as a new paradigm for developing highly energy-efficient computing systems. Approximate computing relaxes the strict Boolean/Numeric equivalence of underlying computing hardware, and trades the computation accuracy loss to obtain significant area, power/energy, and performance efficiency. The key is to leverage the inherent error resilience properties of target applications, i.e., their ability to tolerate approximation errors and yet produce useful output (of acceptable quality to users) [1]-[5]. Therefore, approximate computing is highly amenable to image/video processing applications, which are widely proliferated in entertainment, consumer, automotive, security, and communication industries. Their inherent resilience comes from the fact that camera sensors already provide noisy input data with high correlation, and the application output is graded by different users based on their visual perception and psychological factors [1]-[5]. Moreover, various algorithms have inherent error masking data/control paths [6], as also shown in Fig 1 and Section I.B.



```
Get value of A from user
If (A+5) = ODD
{
        ...
        run code snippet I
}else
{
        ...
        run code snippet II
}
```

Despite the accuracy loss, **Option 2** produces correct result due to algorithm's inherent resilience

Fig 1: An example showing algorithm's resilience to approximation errors.

Resilience Example: Fig 1 presents an *abstract example* showing algorithmic resilience for a code snippet, and additions using *accurate* and *approximate* adders. In this example, the approximation is done by ignoring the output carry of each 1-bit Full Adder (FA). It is noteworthy that although the accuracy loss is about 35%, the outcome of the algorithm is still correct because the result of both accurate and approximate adders is an Even number, and the code snippet II is executed. Option 2 however provides much lower power and energy consumption by eliminating the carry generation and propagation logic. Section IV provides the circuits and corresponding power-accuracy tradeoffs for different approximate adder designs.

Target Application and the Associated Energy Problem: We target the Motion Estimation (ME) for High-Efficiency Video Coding (HEVC, [7]). It is the latest video coding standard that provides 1.6x – 2x improved coding efficiency compared to the H.264/AVC standard [7] at the cost of >40% more computation effort and higher energy consumption [8][9]. This is mainly due to the increased ME effort and mode decision space considering the HEVC's novel Coding Tree Unit (CTU) structure. According to the studies of [8][10], multi-mode ME for different block sizes requires about 80% of the total energy consumption of HEVC encoder. Similar observations were also made by our recent experiments in Fig 2. It shows the energy distribution for "BasketballDrive" Full-HD (1920×1080) sequence using 5 reference frames and CTU size of 64x64. Note, the energy of Inter-prediction is more than 80%, which is mostly taken by the ME process for different block modes (see Section III for complexity analysis).

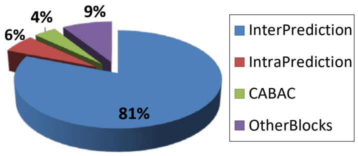


Fig 2: Energy distribution [%] of the HEVC encoder for the "BasketballDrive" sequence.

Therefore, reducing the energy-consumption of the ME process is one of the major research challenges for realizing energy-efficient HEVC systems.

In this paper, we aim at embracing the emerging trend of approximate computing to develop energy-efficient ME architectures.

Our motivational case study in Section I.B illustrates the available potential of approximate computing and inherent resilience of the motion estimation process, which can be leveraged for energy reduction without significant quality loss. Before that, we present the experimental setup for better understanding of the results.

A. Experimental Setup and Tool Flow (Fig 3)

The RTL (in VHDL code) of accurate and different approximate adders and SAD variants are synthesized using: (1) ASIC design flow with Synopsys Design Compiler, 45 nm technology, *WCCOM* (Worst-Case Commercial) operating conditions, *Wire_load_model* set to segmented, and the area optimization option enabled; and (2) FPGA design flow with Xilinx ISE 14.7 for VIRTEX 7 XC7VX330T FPGA device. The generated netlist is verified using gate-level simulations and detailed area, power/energy, latency estimation is performed, e.g., using ModelSim to obtain VCD (Value Change Dump) and SAIF

(Switching Activity Interchange Format) files, which are then used for power estimation using PrimeTime in the ASIC flow. For elementary approximate designs, error analysis is performed using number of error cases, maximum error magnitude, and occurrences of maximum error cases. For full-application evaluation, we developed the equivalent behavior models of these approximate accelerators (in C, C++, and MATLAB), and integrated into an open-source optimized HEVC implementation called *x265* [11]. The quality evaluation, in terms of motion vector difference (MVD), SAD value, bit rate, and video quality (PSNR), is performed for various test video sequences for different block sizes.

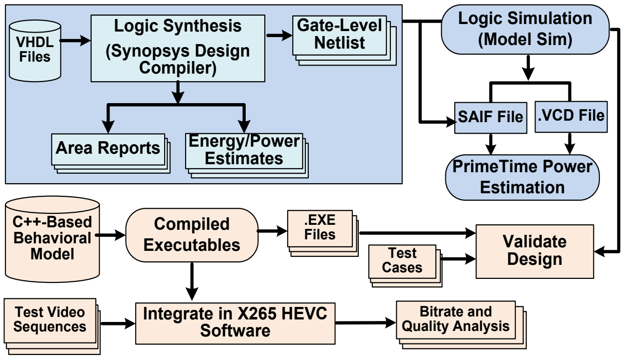

VHDL Files
Logic Synthesis (Synopsys Design Compiler)
Gate-Level Netlist
Logic Simulation (Model Sim)
SAIF File
.VCD File
PrimeTime Power Estimation
Area Reports
Energy/Power Estimates
C++-Based Behavioral Model
Compiled Executables
.EXE Files
Test Cases
Validate Design
Test Video Sequences
Integrate in X265 HEVC Software
Bitrate and Quality Analysis

Fig 3: Our experimental flow showing the hardware and software tools.

B. Motivational Case Study: Resilience Analysis of ME

The main processing kernel of motion estimation (ME) is *SAD* (Sum of Absolute Difference) computation for different candidate blocks; see ME overview in Section III. Finding the best matching candidate block is primarily a minimization problem, i.e. finding the candidate block with the minimum SAD value out of *N* candidates. Let us consider Fig 4(a) that shows four example SAD values computed using accurate and approximate SAD accelerators, which are composed of approximate adders and subtractors (see Section IV for designs). Note, approximate adders are also used for approximate subtractors using 2's complement. Throughout this paper, only approximate adders are discussed.

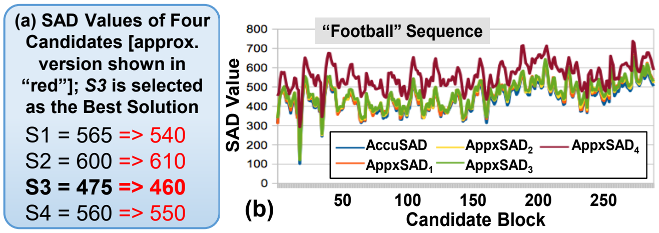


Fig 4: Illustrating the inherent resilience of the ME process (a) An abstract example illustrating that the selection of minimum value is unaffected due to approximations; (b) Error surface plots for accurate and four different approximate SAD architectures [4].

Although the approximate values have errors in the absolute magnitude, it is noteworthy that the minimum solution is always the case "S3". This phenomenon can be best understood by analyzing the error surface plot of Fig 4(b) [4], which shows the SAD values for different candidates using accurate and four different approximate designs of SAD accelerators.

Note that the complete error surface plots in Fig 4(b) are shifted for different approximate designs, and follow almost similar minimization trends for local and global minimas. Therefore, when finding the best match, although the individual SAD values for different candidates have errors due to approximations, the candidate with minimum SAD value remains the same in most of the cases. The trend is very similar for different approximate variants of the SAD accelerator, which can be realized by using different types of approximate arithmetic modules and different number of LSBs to be approximated (see design details in Section IV). This shows the inherent resilience of the ME process that can be leveraged to achieve significant energy savings through (relaxed or aggressive) approximations.

The error in the decision process of ME can only happen if two candidates have very close SAD values, and the impact of approximations is different due to different input values. This can be understood from Fig 4(a): if S3 would have not been a potential candidate in the matching process, then S4 would have been selected in the accurate case, while S1 would have been selected in the approximate case. However, there is another interesting point to note: since the SAD values of these candidates are close, even a wrong decision would not lead to a significant degradation of output quality because the resulting residual to encode after the inter-prediction will also be in close range.

C. Our Novel Contributions and Open-Source Library

In this paper, we demonstrate how the emerging trend of approximate computing can be leveraged for energy-efficient motion estimation in HEVC. Besides presenting the energy distribution of HEVC (Section 1), resilience analysis of ME (Section I.B) and computational complexity analysis of HEVC ME (Section III), *we make the following further novel contributions*:

1) An approximate architecture for energy-efficient motion estimation (Section V) that employs *different SAD accelerators with accurate and heterogeneous approximation modes* for different block sizes. It provides different tradeoff points in terms of energy consumption, area, resulting bit rate, and output video quality, and allows user to select an appropriate variant depending upon their requirements, e.g., in terms of bit rate and required energy savings.

2) Tradeoff analysis of heterogeneous approximate SAD variants (Section VI) for energy, area, power, quality and bit rate for different video sequences. These designs are synthesized and validated using ASIC and FPGA design flows. These accelerators are then integrated into the HEVC motion estimation for further analysis in terms of bit rate, video quality, and motion vector difference.

An open-source library of approximate SAD accelerators and constituting modules (i.e. approximate multipliers and adders) is provided at https://sourceforge.net/projects/lpaclib/ [17]. It contains the RTL (in form of VHDL codes) and behavioral models (in form of C-/C++ and MATLAB codes). This library facilitates reproducible results for comparisons and further research/development of energy-efficient video coding systems based on approximate computing.

II. RELATED WORK

Approximate Computing: Comprehensive surveys on approximate computing can be found in [4][12]. A majority of the work is done on developing elementary approximate arithmetic blocks like approximate adders [13][14][15] and approximate multipliers [16] [17][23], error correction in high-performance approximate accelerators [24], approximate cache [26], and some works have been done at the programming language [3] and application-level approximations [25]. There are two types of approximate adder designs: (1) *Low-latency approximate adders* (like ACA and GeAr [15]) that break the carry chain to achieve high performance design. However, these designs employ overlapping sub-adders to approximately predict the carry, which incur high area and power, and therefore only beneficial for high performance, high-power designs. (2) *Low-power 1-bit approximate full-adders* (like IMPACT [13][14]) that approximate the circuit logic of single-bit full adders through different circuit decimation and simplification techniques. Similar concept for multipliers is proposed in [16][23]. Hardware-level approximate computing research has primarily targeted circuit and elementary arithmetic blocks, and has not explored the design of low-power approximate architecture and approximate accelerators, especially for video coding, and in particular for the HEVC.

Fast and Energy-Efficient Motion Estimation in HEVC: Much of the energy-efficient ME works exist for H.264, which primarily employ techniques like early search termination and SAD decimation [10][18], modification of SAD formula [19], and voltage over-scaling [19][20]. Many of these techniques provide fixed low-power solution without run-time adaptability, or employ standard low-power techniques and suffer from significant quality loss, but do not exploit the potential of heterogeneous approximation modes. Besides computation, memory energy reduction during ME can also be achieved through search window design, exploitation of data reuse, and power-gating of memory blocks [8].

There has been some early works from error-tolerant computing domain [19][20] where voltage over-scaling has been employed in ME architectures to save power at the cost of voltage-induced timing errors. Similar concepts have also been applied to fault-tolerant JPEG2000 [21], and frame buffer memory of H.264 decoder [22], but not motion estimation. This is somewhat analogous to approximate computing, except the source of errors, which is incorrect functionality (i.e. functional errors) in approximate hardware. This feature requires completely different design principles and architectural concepts, as explored in this paper. The work in [19] performs input sub-sampling for power reduction, but incurs *severe* PSNR (Peak Signal to Noise Ratio) losses for videos containing high texture and motion content. *We do not incur significant PSNR losses by exploiting heterogeneous approximate modes.* The work in [20] performs error-inducing voltage-scaling for the less-significant computations. The above works employ their concepts to a three-step search based motion estimator of an old-generation of codec to analyze the error tolerance. However, they have two main limitations: (1) They do not exploit the potential of emerging approximate arithmetic blocks (like adders). (2) The three-step search gets trapped into local minima, therefore the best accurate answer is already highly erroneous compared to any good adaptive ME algorithm. Therefore, the analysis of voltage-overscaling induced errors is not representative in terms of realistic benefits vs. error rates.

In short, state-of-the-art has not yet systematically explored the potential of approximate accelerator-based architecture for energy reduction in motion estimation (also not in HEVC) and the energy-quality tradeoff analysis for heterogeneous approximation modes. This paper makes the first attempt to bridge this gap, and facilitates further research towards this growing field through *first open-source contributions in approximate accelerators for video coding.*

Note: our proposed concepts and techniques are orthogonal to most of the related works based on SAD decimation, search window re-sizing, and memory energy reduction, and thereby can be employed in conjunction with those techniques.

III. Background Knowledge and Computational Complexity Analysis of HEVC Motion Estimation

Coding Structure of HEVC: The key processing block in HEVC is called *Coding-Tree Block or Unit* (CTB, CTU), which defines a flexible variable-sized block structure for coding. Each CTU is partitioned into multiple *Coding Units* (CU) of sizes 64x64, 32x32, down to 4x4 pixels. The search for the best CU mode, that provides the best coding efficiency in terms of coded video quality and bitrate, is done recursively in a tree structure, starting from the Largest CU (LCU). Fig 5 depicts an example of CTU partitioning. For each CU, the motion estimation is performed for (multiple) reference frames.

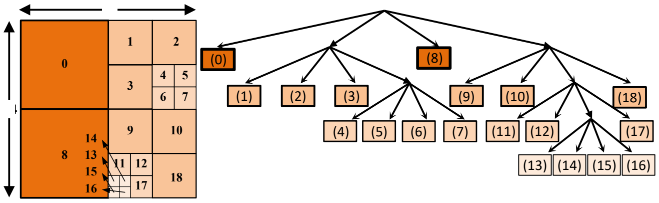


Fig 5: The organization of the coding-tree unit/block in HEVC.

Overview of Motion Estimation (ME): For each CU in the CTU of the *current* video frame (C), the motion estimator finds out its best match by comparing it with different selected candidate blocks (R), of same size as of CU, in one or multiple *reference* video frames (i.e. previously encoded and reconstructed frames). The selection of candidate depends upon the ME algorithm, while the search is performed in a restricted *search window.* The most widely used matching metric is *Sum of Absolute Differences* (SAD, Eq. 1). CU_{height} and CU_{width} are the height and width of a given CU in number of pixels.

$$SAD = \sum_{y=0}^{CUheight} \sum_{x=0}^{CUwidth} \left| C(x,y) - R(x,y) \right| \qquad (1)$$

The candidate with the minimum SAD is given as the *best match*, and its distance from the current CU is given as the *motion vector*. The selection of best CU for final encoding is done through the rate-distortion optimization process. Since ME can be performed for each possible CU inside a CTU, it results in a very high energy cost as discussed in Section I. Fig 6 illustrates the ME process and the generic data path of the SAD accelerator.

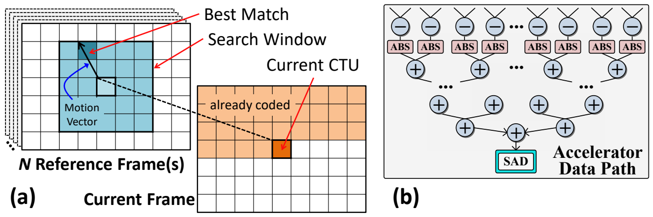


Fig 6: (a) Motion estimation/search process; (b) SAD accelerators data path.

Complexity Analysis of ME in HEVC: The energy consumption is proportional to the computational requirements of the ME, which directly depend upon the number of operations in one SAD, and the number of computed SADs for candidate blocks. While the first factor is fixed, the latter depends upon the search algorithm and mode decision. A full search ME (FME) exhaustively evaluates all candidate blocks within in the search window. For instance, for each CU size, for a search window of size 32x32, there are 1024 candidates for SAD computation (4096 for 64x64, and 16384 for 128x128 search window). Typically, in energy-efficient real-world implementations, fast adaptive search (AME) algorithms like *TZ*, *EPZS*, or *UMHexagonS* are used. A good fast ME algorithm will cut down the search complexity (i.e. number of candidates for SAD evaluation) of full search by about 80%-90% [10]. However, in general, the computation complexity of ME increases with the number of CU sizes. Table I illustrates the number of CTUs, CUs, and candidates evaluated per frame using FME and AME, for different video resolutions.

Note, one block SAD (for a 32x32 CU size) requires 1024 subtractions for difference computation and 1023 additions for the adder tree (see data path in Fig 6), i.e. about 2K arithmetic operations per CU. Therefore, even using AME, the compute effort for one frame of HD1080p and CIF videos will require more than 1.8 T and 5.4 B operations, respectively. For a frame rate of 30fps, this corresponds to more than 50 T and 160 B operations for HD1080p and CIF videos, respectively. Such a high number of arithmetic operations in SAD computations illustrates a high potential of energy reduction through using approximate computing modules, besides providing parallel SAD arrays for high throughput.

Table I: Complexity Analysis for Different Video Resolutions.
(AME, with 90% complexity reduction compared to FME)

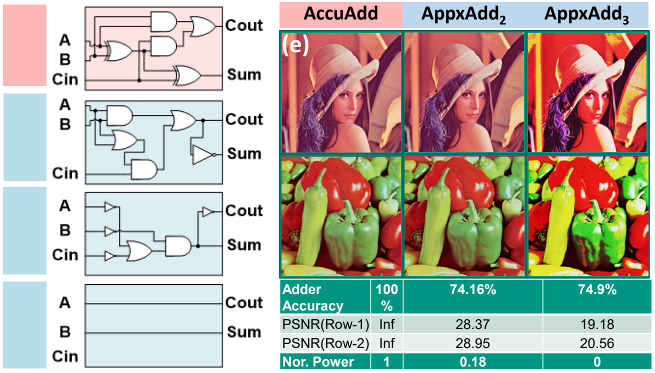
Resolution	**QCIF**	**CIF**	**480p**	**HD720p**	**HD1080p**
Width × Height	176 x 144	352 x 288	640 x 480	1280 x 720	1920 x 1080
LCU Size	16	16	32	32	64
#CTUs in 1 frame	99	396	300	900	510
#CUs in 1 CTU	65	65	273	273	1105
Search Window	32x32	32x32	64x64	64x64	128x128
#Cand. FME	1024	1024	4096	4096	16384
#Cand. FME/frame	**6.6M**	**26.4M**	**335.4M**	**1B**	**9.2B**
#Cand. AME [10%]	102	102	410	410	1639
#Cand. FME/frame	**0.6M**	**2.64M**	**33.5M**	**0.1B**	**0.92B**

IV. Approximate Adders: Design and Analysis

A. Designing Approximate Adders

In this paper, we deploy the approximate 1-bit full adders (FA) of IMPACT designs [13][14], and the multi-bit approximate adders from the open-source library of [23]. In the following, we provide only the necessary background information required to understand the novel contributions of this paper.

Approximate 1-Bit Full Adders (FA): Fig.7 shows the circuit diagrams of accurate adder (*AccuAdd*) and 3 approximate adders (*AppxAdd$_1$*, *AppxAdd$_2$*, and *AppxAdd$_3$*) based on the designs of [13][14]. The VHDL codes of these designs can be found in our open-source library of [17]. The area, latency, power, and error results are depicted in Table II, which were observed exactly the same as reported in the [23], thus also validating the reproducibility. Note, *AppxAdd$_3$* offers the best area, power, and latency results, because it is simply a short-circuit logic, and does not incur any switching power. However, it also has the highest error rate. *AppxAdd$_1$* and *AppxAdd$_2$* are reasonable tradeoff points w.r.t. power and error rate.



Adder Accuracy	100 %	74.16%	74.9%
PSNR(Row-1)	Inf	28.37	19.18
PSNR(Row-2)	Inf	28.95	20.56
Nor. Power	1	0.18	0

Fig.7: (a) Accurate 1-bit FA; (b, c, d) Approximate 1-bit FA of [13][14]; (e) Power-quality impact of approximate adders for low-pass image filtering.

Table II Characterization of Approximate 1-Bit Full-Adders.
(Our results came out same as reported in the open-source library of [23])

	Area [GE]	Latency [ns]	Power [nW]	Number of Error Cases	Max Error Magnitude	Occ. of Max Error
AccuAdd	4.41	0.12	1130	0	0	0
AppxAdd$_1$	1.94	0.07	294	2	1	2
AppxAdd$_2$	1.59	0.05	198	3	1	3
AppxAdd$_3$	0	0.00	0	4	1	4

Approximate Multi-Bit Adders: For building the multi-bit adders, we follow the design method depicted in Fig.8 [23], where approximations are only done for the LSBs to avoid high error magnitude. For an N-bit adder, only k-bits are approximated using one of the three types of approximate 1-bit FAs (FA_{apx}) as shown in Fig.7, while for N-k bits accurate 1-bit FA (FA_{acc}) are used. To avoid design complexity, each approximate multi-bit adder variant has only one type of approximate 1-bit FA for all the k-approximated bits.

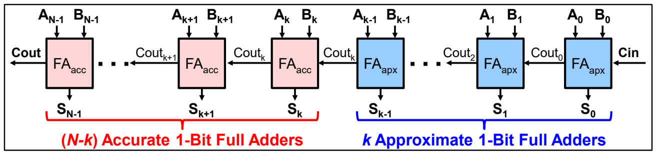


Fig.8: Multi-Bit Adder Chain using Accurate or Approximate 1-Bit Adders.

B. Power-Quality Analysis for Approximate Adders

To validate our implementations and to analyze the power-quality impact of these multi-bit approximate adders, we deployed accurate and two approximate 8-bit adders in a low-pass image filtering applications. Fig.7(e) shows the results for power (normalized to that of the accurate version), adder accuracy (% loss compared to accurate version), and output quality in terms of subjective quality (images) and objective quality (PSNR – Peak Signal to Noise Ratio, compared to the output of accurate adder-based filtering). It is interesting to see that the *AppxAdd$_2$*, though having reduced PSNR, still produces similar subjective quality as of the accurate design. However, the subjective quality of the *AppxAdd$_3$* is degraded yet recognizable, but provides a very high power reduction.

V. Approximate Architecture for Motion Estimation in HEVC

A. Approximate ME Architecture

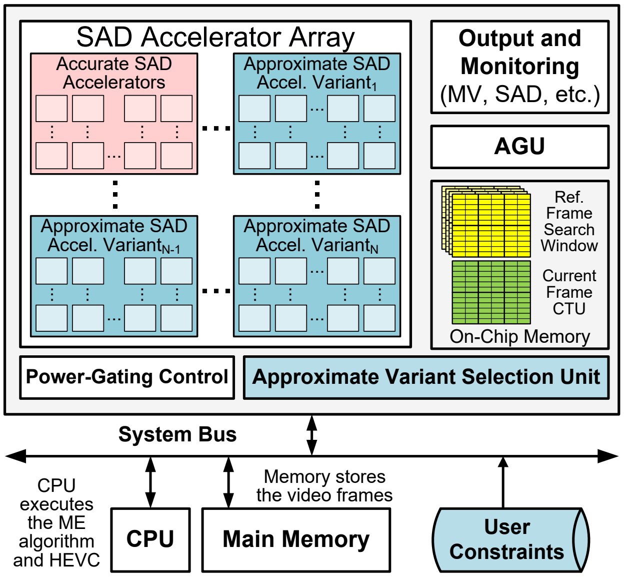

SAD Accelerator Array
Accurate SAD Accelerators
Approximate SAD Accel. Variant1
Approximate SAD Accel. VariantN-1
Approximate SAD Accel. VariantN
Output and Monitoring (MV, SAD, etc.)
AGU
Ref. Frame Search Window
Current Frame CTU
On-Chip Memory
Power-Gating Control
Approximate Variant Selection Unit
System Bus
CPU executes the ME algorithm and HEVC
CPU
Memory stores the video frames
Main Memory
User Constraints

Fig 9: Hardware architecture of our approximate motion estimator.

Fig 9 illustrates our approximate ME architecture for HEVC. It contains an array of SAD accelerators organized in form of heterogeneous approximate SAD tiles. Each tile contains multiple instances of a particular type of SAD variant. The approximate variant selection unit contains a look up table (LUT) filled with entries like power/energy and quality obtained from the design time analysis for heterogeneous approximate SAD variants for different block sizes. Depending upon the user requirements in terms of tolerable error and required energy reductions, tiles of appropriate approximate SAD variants are powered-on, while keeping all unrequired tiles in power-gated states. The HEVC and motion estimation algorithm execute on the general-purpose processor core. The candidate vectors and pointer addresses are forwarded to the Address Generation Unit (AGU), which generates the memory address to fetch the data from on-chip memories storing the current CTU data and the search windows from reference frame(s). In case the required data is not in the on-chip memories, it is fetched from the main memory which stores the complete current and reference frames. The data from on-chip memories is forwarded to the SAD accelerators for computing the matching cost. The monitoring unit is responsible for maintaining the intermediate SAD and motion vector (MV) values, such that, the motion estimator can make fast search decisions, and can determine the best match, i.e. the one with the minimum SAD value.

B. Heterogeneous Approximate Variants of SAD Accelerator

We developed different SAD units of sizes 8x1, 8x8, 16x16, and 32x32. The elementary accelerator is 8x1, which is then re-used to build bigger SAD blocks. For instance, we constructed 8x8 SAD unit out of one 8x1 SAD units that runs for 8 cycles. Alternatively, 8 such units can also be placed in parallel to obtain a single cycle implementation with more area cost. A 32x32 SAD unit was built using four 8x1 units and runs for 32 cycles.

Accurate and heterogeneous approximate variants of SAD accelerators (numbered as *"V"*) are built using accurate or three approximate adders, and choosing 2, 4, or 6 LSBs for approximation. The area, power, and energy results for 8x8 and 32x32 SADs using the ASIC design flow are shown in Table III.

Table III Area, Power, and Energy Results for Heterogeneous Approximate SAD Variants: (a) 8x8 SAD; (b) 32x32 SAD

V	**8x8 SAD**	*Approx LSBs*	*Area [GE]*	*Power [µW]*	*Energy [pJ]*	**32x32 SAD**	*Approx LSBs*	*Area [GE]*	*Power [µW]*	*Energy [pJ]*
0	*AccuAdd*	0	1383	190.4	4.90	*AccuAdd*	0	4038	567.1	14.61
2	*AppxAdd$_1$*	2	1336	176.4	4.54	*AppxAdd$_1$*	2	3847	506.7	13.05
5	*AppxAdd$_1$*	4	1257	154.9	3.99	*AppxAdd$_1$*	4	3531	411.8	10.60
7	*AppxAdd$_1$*	6	1178	133.7	3.44	*AppxAdd$_1$*	6	3215	318.8	8.21
3	*AppxAdd$_2$*	2	1323	175.0	4.51	*AppxAdd$_2$*	2	3794	503.0	12.96
6	*AppxAdd$_2$*	4	1233	151.3	3.90	*AppxAdd$_2$*	4	3433	401.0	10.33
8	*AppxAdd$_2$*	6	1143	128.2	3.30	*AppxAdd$_2$*	6	3071	302.0	7.78
1	*AppxAdd$_3$*	2	1280	173.5	4.47	*AppxAdd$_3$*	2	3626	495.3	12.76
4	*AppxAdd$_3$*	4	1100	142.4	3.67	*AppxAdd$_3$*	4	2887	363.0	9.35
9	*AppxAdd$_3$*	6	912	112.7	2.90	*AppxAdd$_3$*	6	2119	237.6	6.12

We additionally evaluated the efficacy of designs using the FPGA design flow. The area (in terms of LUTs and Slices), power and latency (critical clock delay and circuit's delay after place-and-route) results are shown in Fig 10 for 8x8 and 32x32 SAD accelerators, using three approximate adders (*AppxAdd$_1$*, *AppxAdd$_2$*, and *AppxAdd$_3$*) with 6 LSBs approximated. It is noteworthy that S3 has the lowest power and latency, and therefore is the most aggressive approximate variant, while S1 is the worst w.r.t. power and S2 in the middle.

VI. Results and Discussion

Area, latency, and power/energy results have already been presented in the previous sections. Now, we will discuss the following three major results, analyzing the impact of our approximations on the:

1) output of ME in terms of MV difference and SAD,
2) bit-rate and PSNR of *x265* reconstructed videos, and
3) energy consumption of ME vs. MV difference.

Due to long encoding delays and memory constraints, we present results mostly for CIF sequences, though our approach is equally valid for HD sequences, as we will show for one experiment.

A. Impact of Approximation on the outcome of ME

Approximation impact can be studied in form of (1) change in the SAD values (as also demonstrated in Section I.B); and (2) change in the motion vector in form of motion vector difference (MVD, as defined in Eq.2).

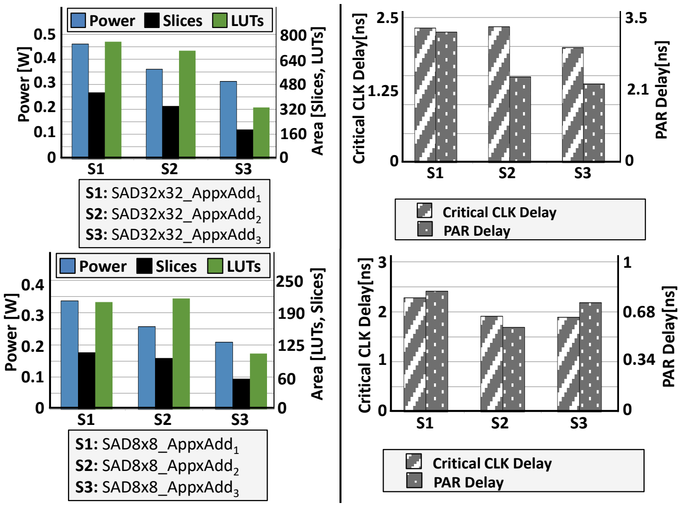


Fig 10: FPGA synthesis results: (a, c) Comparing power and area (in terms of slices and LUTs) of approximate SAD variants of 32x32 and 8x8 sizes, respectively. (b, d) Comparing latency (critical clock delay and post place-and-route delay) of approximate SAD variants.

$$sqrt[(MV.X_{accurate} - MV.X_{approx})^2 + (MV.Y_{accurate} - MV.Y_{approx})^2] \qquad (2)$$

Fig 11 shows the average MVD for different approximate variants (as defined in Table III) for different video sequences, considering 16x16 SAD accelerators and a search range of 32. The resulting non-zero differences are averaged to obtain single bar for each case as shown in Fig 11. Note, increasing the number of approximate LSBs does not necessarily decrease accuracy. This is the case, for example, for "Bus" sequence when comparing *AppxAdd$_1$* with 2-, 4-, and 6-bits approximation (i.e. variants V=2, 5, and 7). There is no general rule for which adder performing generally better than another adder, i.e. when arranging the approximate variants ascendingly according to their corresponding MVDs will not result in the same order. Table IV illustrates the variant order w.r.t. increasing MVD values.

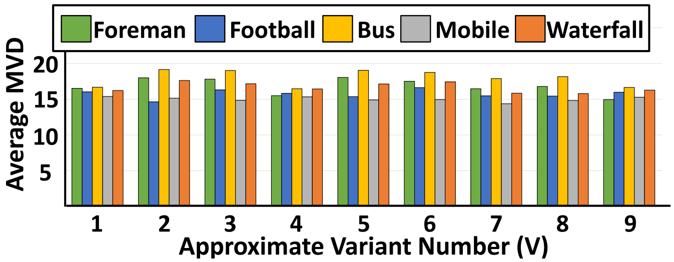


Fig 11: Average motion vector difference (MVD) for different video sequences at different approximation levels (see level description in Table III).

Table IV Variant Order w.r.t. Increasing MVD for Different Videos

Video Sequence	Variant order according to increasing MVDs
Foreman	[9, 4, 7, 1, 8, 6, 3, 2, 5]
Football	[2, 5, 8, 7, 4, 9, 1, 3, 6]
Bus	[4, 9, 1, 7, 8, 6, 3, 5, 2]
Mobile	[7, 8, 3, 5, 6, 2, 9, 4, 1]
Waterfall	[8, 7, 1, 9, 4, 5, 3, 6, 2]

Fig 12 shows the error surface plot (in terms of SAD values) for different approximate variants and for two HD720p video sequences, for a search window of 65x65. It is noteworthy that in several HD video sequences, the error surface degradation is not significant (row 1), and overall for all sequences, error surface degradation still follows the same minimization trend, illustrating the high resilience of ME, and low impact of our approximations on output quality degradation.

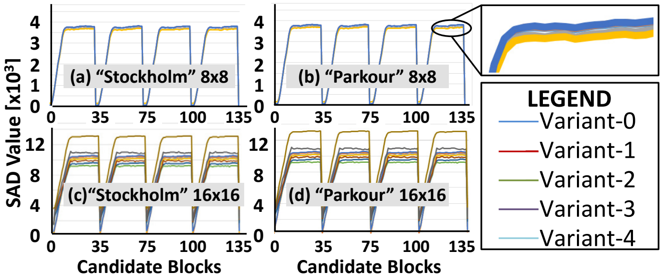


Fig 12: Error surface plots for accurate and different approximate SAD variants for HD720p video sequnces for 8x8 SAD and 16x16 SAD.

B. Bit-Rate & Quality Analysis of x265 Reconstructed Video

Fig 13 analyzes the impact of heterogeneous approximate SAD variants on the coding efficiency, in terms of bit rate, output stream size, and PSNR video quality for the reconstructed video when executing the full x265 video encoder flow. We notice that approximating 6-bits of the adders in the SAD accelerator results in very high increase in the bit-rate (which may be unacceptable), while approximating 2- and 4-bits results in a marginal bit-rate increase, which shows that they are good tradeoff options w.r.t. quality and power/energy reduction. It is noteworthy that the video quality in terms of PSNR stays within the range of 33.01 and 33.58, i.e. only a minimal PSNR degradation of 0.57 dB is incurred. Since the net coding efficiency is determined by both bit rate and PSNR, Fig 13 shows that our approximations maintain the video quality in terms of PSNR, but lead to bit rate increase due to high prediction error in the motion estimation process in case the best matches are different for accurate and approximate versions.

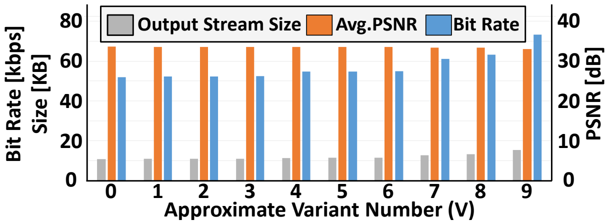


Fig 13: Comparing the bit-rate, output stream size, and PSNR quality for different approximate SAD variants for "Foreman" sequence.

C. Energy Consumption of Motion Estimation

Table V shows the average energy vs. average MVD results for selected approximate SAD variants for the "Waterfall" sequence. *We selected this sequence as it presents a counter-example.* From this table, we can conclude that, for the same adder type, increasing the number of approximate bits will increase the average MVD indicating quality degradation. However, an increase of average energy is also noted. For instance, Variant-7 (with 6 approximate bits) consumes as much energy as the accurate variant, which is due to more candidate evaluations, as the stopping criteria of the ME is not achieved due to the high texture content and camera panning motion in this sequence. That is, it may happen that for certain video contents, not all approximate variants bring energy benefit for the ME process. In this case, Variant-3 is the best in terms of energy vs. quality tradeoff.

Table V: Energy consumption vs. MVD Analysis for ME for "Waterfall".

Variant	0	2	3	5	7
Avg. MVD	0.307	0.330	0.328	0.371	0.462
Avg. Energy [mJ]	1.24	1.15	1.14	1.24	1.24

VII. Conclusion

We presented an approximate computing architecture for HEVC motion estimation. Our architecture employs various tiles of accurate and heterogeneous approximate variants of the SAD kernel, which enables a wide-range of energy-quality tradeoffs for the user. We analyzed the inherent resilience of the motion estimation process, performed detailed characterization of heterogeneous approximate SAD variants in terms of power/energy, area, and latency. We synthesized our designs for both ASIC and FPGA design flows and integrated into a real-world HEVC applications (x265). We also made the RTL and behavioral implementations of our various designs open-source at https://sourceforge.net/projects/lpaclib/, which enable reproducible comparisons and further research and development. Embracing approximate computing unleashes new avenues for energy-efficient embedded multimedia systems employing highly complex HEVC codecs, which will particularly be beneficial for battery-constrained mobile devices.

References

[1] A. K. Mishra, R. Barik, S. Paul, *"iACT: A Software-Hardware Framework for Understanding the Scope of Approximate Computing"*, Workshop on Approximate Computing Across the System Stack (**WACAS**), 2014.

[2] R. Nair, *"Big data needs approximate computing: technical perspective"*, **ACM Communications**, 58(1): 104, 2015.

[3] H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, *"Architecture support for disciplined approximate programming"*, International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**), 2012.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, J. Henkel, *"Cross-Layer Approximate Computing: From Logic to Architectures"*, IEEE/ACM **DAC**, 2016.

[5] V. Chippa, S. Chakradhar, K. Roy, A. Raghunathan, *"Analysis and characterization of inherent application resilience for approximate computing"*, **DAC**, 2013.

[6] M. Shafique, S. Rehman, P. V. Aceituno, J. Henkel, "Exploiting Program-Level Masking and Error Propagation for Constrained Reliability Optimization", **DAC**, 2013.

[7] B. Bross, W. J. Han, J. R. Ohm, G. J. Sullivan, T. Wiegand, "High Efficiency Video Coding (HEVC) text specification draft 7", May 2012.

[8] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, J. Henkel, "Energy-Efficient Architecture for Advanced Video Memory", IEEE **ICCAD**, pp. 132-139, 2014.

[9] M. Shafique, J. Henkel "Low Power Design of the Next-Generation High Efficiency Video Coding", ASP-DAC, pp. 274-281, 2014.

[10] M. Shafique, L. Bauer, J. Henkel, "enBudget: A Run-Time Adaptive Predictive Energy-Budgeting Scheme for Energy-Aware Motion Estimation in H.264/MPEG-4 AVC Video Encoder", IEEE/ACM **DATE**, pp. 1725-1730, 2010.

[11] X265: http://www.codeforge.com/article/364279.

[12] S. Mittal, *"A Survey of Techniques for Approximate Computing"*, ACM Computing Surveys (**CSUR**), vol. 48 no. 4, article 62, May 2016.

[13] V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan, *"IMPACT: IMPrecise adders for low-power approximate computing"*, IEEE **ISLPED**, pp. 409 – 414, 2011.

[14] V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, *"Low-Power Digital Signal Processing Using Approximate Adders"*, IEEE **TCAD** 32(1): 124-137, 2013.

[15] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, *"A Low Latency Generic Accuracy Configurable Adder"*, IEEE/ACM **DAC**, 2015.

[16] P. Kulkarni, P. Gupta, M. Ercegovac, *"Trading Accuracy for Power with an Underdesigned Multiplier Architecture"*, **VLSI Design**, pp. 346 – 351, 2011.

[17] Open-Source Library of Low-Power Approximate Computing Modules: https://sourceforge.net/projects/lpaclib/.

[18] C. Kim, et al., "Complexity scalable motion estimation for H.264/AVC", Proc. SPIE, Visual Comm. Image Proc., vol. 6077, pp. 109-20, Jan 2006.

[19] G. V. Varatkar and N. R. Shanbhag, "Energy-efficient motion estimation using error-tolerance", IEEE **ISLPED**, pp. 113-118, 2006.

[20] D. Mohapatra, G. Karakonstantis, K. Roy, "Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator", IEEE **ISLPED**, pp. 195-200, 2009.

[21] M. A. Makhzan, A. Khajeh, A. Eltawil, F. J. Kurdahi, "A low power JPEG2000 encoder with iterative and fault tolerant error concealment", IEEE **TVLSI**, vol. 17, no. 6, pp. 827-837, 2009.

[22] A. K. Djahromi, A. Eltawil, F. J. Kurdahi, "Exploiting fault tolerance towards power efficient wireless multimedia applications", IEEE **CCNC**, pp. 400-404, 2007.

[23] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, J. Henkel, "Architectural-Space Exploration of Approximate Multipliers", IEEE/ACM International Conference on Computer-Aided Design (**ICCAD**), 2016.

[24] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, "An Area-Efficient Consolidated Configurable Error Correction for Approximate Hardware Accelerators", ACM/IEEE Design Automation Conference (**DAC**), 2016.

[25] D. Palomino, M. Shafique, A. Susin, J. Henkel, "Thermal Optimization using Adaptive Approximate Computing for Video Coding", IEEE/ACM **DATE**, 2016.

[26] F. Sampaio, M. Shafique, B. Zatt, S. Bampi, J. Henkel, "Approximation-Aware Multi-Level Cells STT-RAM Cache Architecture", IEEE International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (**CASES**), 2015.