A 93 TOPS/Watt Near-Memory Reconfigurable SAD Accelerator for HEVC/AV1/JEM Encoding

Jainaveen Sundaram[1], Srivatsa Rangachar Srinivasa[1], Dileep Kurian[1], Indranil Chakraborty[2], Sirisha Kale[1], Nilesh Jain[1], Tanay Karnik[1], Ravi Iyer[1] and Anuradha Srinivasan[1]

[1]Intel Labs, 2111 NE 25th Ave, Hillsboro OR 97124 USA

[2]Purdue University, West Lafayette, IN 479006 USA

***Abstract*— Motion Estimation (ME) is a major bottleneck of a Video encoding pipeline. This paper presents a low power near memory Sum of Absolute Difference (SAD) accelerator for ME. The accelerator is composed of 64 modular SAD Processing Elements (PEs) on a Reconfigurable fabric, offering maximal parallelism to support traditional and futuristic Rate-Distortion-Optimization (RDO) schemes consistent with HEVC/AV1/JEM. The accelerator offers up-to 55% speedup over State-of-art accelerators and a 7x speedup when compared to a 12 core Intel Xeon E5 processor. Our solution achieves 93 TOPS/Watt running at 500MHz frequency, capable of processing real-time 4K 30fps video. Synthesized in 22nm process, the accelerator occupies 0.08mm^2 and consumes 5.46mW dynamic power.**

Keywords*— Near Memory Compute, Motion Estimation, Accelerator

I. Introduction

Innovations in the field of AI, IoTs and smart automobiles have led to billions of connected devices expected to generate up to 40 trillion gigabytes of internet traffic by 2022, with nearly 80% of it being video data [1]. Video coding standards like HEVC/AV1/JEM are introduced to guide the encoder improve overall video quality for a given bitrate [2]. Block Matching Algorithms (BMA) is employed with Sum of Absolute Differences (SAD) is used where the Coding Unit (CU) kernels are compared against multiple candidates from the search area to calculate a resultant Motion Vector (MV). This process is repeated for every single CU in the frame, iterated across multiple CU sizes to obtain an optimal configuration used to encode a frame, in a process known as Motion Estimation (ME).

This "try all and select the best" approach, with each pass entailing repeated data access results in high computational complexity and limits use of encoders in real-time applications. ME is the most memory intensive task of the encoder, with a significant 39.6% of encoder energy and 43% of encoding cycles expended [3].

To address the ME bottleneck, on the Hardware front, solutions include traditional CPU/GPU based approaches [4], Accelerator based methods [3,5-9,10]. At the system level, most of the recent work focusses on optimizing the RDO portion to arrive at optimal CU split configuration. The goal is to predict CU partitioning split for the frame, to process minimal number of CU's. Spatial/Temporal Correlation of CU's [12] and machine learning techniques such as SVM/ CNN's [13] have been used successfully to predict CU splits, with a 30-50% decrease coding time with negligible impact on quality.

In line with the present research direction, we present a Near Memory Compute (NMC) accelerator architecture achieving 93 TOPS/Watt and capable of processing up-to 4K30FPS in real time. For traditional encoding mode, smaller PE's are combined to form bigger CU's achieving state-of-art (SOTA) performance. For modern encoding schemes with cleverer RDO controls, the fabric parallelizes PE's to work on the same CU to achieve up-to 55% speedup over SOTA. Per our knowledge, this work is the first SAD Accelerator to cater to newer RDO schemes and adopt NMC. Main contributions of this paper are:

1. We present an energy efficient NMC solution which accelerates Motion Vector (MV) computation for traditional and futuristic RDO schemes.
2. A novel reconfigurable fabric along with PE clustering strategy is proposed, offering 7X speedup over CPUs and capable of meeting real time 4K 30FPS video encode
3. We provide a qualitative design description backed by the quantitative power and performance analysis and achieve 93 TOPS/Watt running at 500MHz

II. Background And Related Work

A. Video Encoding Pipeline

In a typical Video encoder, every frame is categorized into I/P/B frames by the encoder, with the luma component of the I-frame coded spatially. P-frames are split into smaller Coding Units (CU's) using a flexible quad-tree partitioning as shown in Fig. 2 [2]. The choice of CU size used to encode is calculated by computing MV across all possible CU sizes before picking the optimal point, using the following equation:

$$J = B + \lambda \cdot D$$

Here, B specifies bit cost considered for the chosen CU size, lambda is the Lagrange multiplier and D is the distortion metric. The CU split with the least overall cost is chosen by the encoder to be transformed, quantized and then entropy coded.

B. Related Works

Approaches for efficient parallelization of SAD Kernels between CPU/GPU cores have been proposed in [4], with all of them suffering high data movement cost. ME is a memory

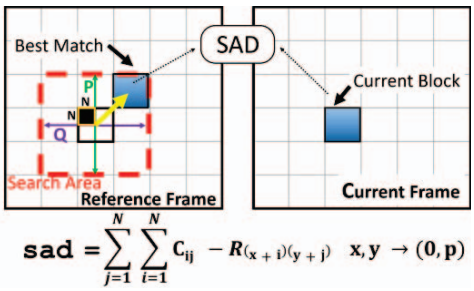

Fig. 2: Depiction of the SAD implementation with reference and current frame for finding the similarity metric

bound task with every 4K frame entailing about 85MB data movement [3]. Accelerator based solution has been popular, with first wave of Systolic arrays [8] replaced by basic 2D Vector Streaming architectures [6,7] which fell out of favor to Partial SAD Propagation schemes [5]. Vector PEs connected by adders provides a low latency and efficient data path for various CU sizes. Succeeding work in the architecture space involved smaller tweaks like extending the Propagate Partial SAD to multiple CU sizes [9].

More recently, a volume of research has been focused on reducing the number of CU's calculated for the RDO engine to decide on frame level CU split. [11, 12] explored spatial and temporal correlation are used to make CU splits, resulting in a 40% reduction in coding time with negligible PSNR/Bit-rate impact. Content aware algorithms and machine learning techniques [13] have also been successfully deployed to predict CU splits to optimize RDO selection, saving up-to 70% coding time.

With the developments in the RDO space, design goals for a SAD Accelerator is clear – (a) When operating with a traditional RDO scheme, the Accelerator must be fast to provide MV's for all CU configurations (b) When operating with a modern RDO scheme, the Accelerator needs to only process specific CU sizes, but efficiently. Our work is an NMC SAD Accelerator, comprised of 64 4x4 SAD PEs. A novel reconfigurable fabric bundles PEs according to RDO mode chosen, delivering state of art performance in situation (a) and offering up-to 55% speedup for (b).

III. Proposed Design

A. System Overview

Fig. 3 shows the high-level system design of a video encoding capable hardware. BMA portion of an encoding task is offloaded to the Accelerator sub-system by general-purpose core/application specific encoder. We designed an SRAM memory of 8MB with a 32B/cycle bandwidth to operate near accelerator. CU Kernel being scanned, and search area are populated into the memory before computation is triggered. Upon receiving the offload notification and memory pointers, the Accelerator first loads the CU to be searched (from the current frame) and then scans the search area (reference frame) to calculate best SAD. Post completion, the result is written back to the memory.

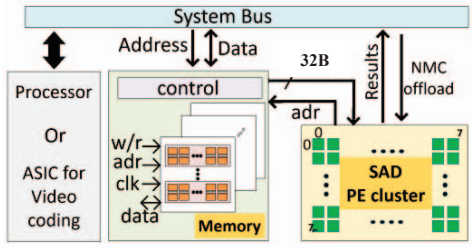

Fig. 3: Video coding system where NMC solution is used for SAD computation while processor performs rest of the video coding tasks.

B. Accelerator Design and Data flow

As shown in Fig. 4, the accelerator is made up of four major elements:

1. ***The Controller***, which interfaces with rest of the system and orchestrates data flow for the SAD computation.
2. ***Data-buffer*** which holds and broadcasts data to all PE's.
3. ***Compute Grid (CG)***, made of 8x8 mesh of PE's, interspersed with adders and comparators. Data flow within the compute grid is reconfigurable based on CU size and RDO requirement.
4. *A **final Comparator*** which completes the filtration of results from the grid; to yield the best SAD.

Working of the SAD Accelerator can be visualized in three stages: (1) The Set Stage – where CU mode decision is passed and controller which configures the compute grid (2) The Load Phase - Where the CU Kernel being searched is loaded (3) The Compute Phase – Where search area is brought in (32B/cycle), and broadcast to all the PE's for processing.

Incoming 32B data is split into chunks of 4B busses which are interleaved and broadcast between rows of PE within the CG. The Compute Grid is made of 64 PE's, organized in 8 rows of 8 PE's. Based on the fabric setting, each PE choses 4B of the 8B data it has access to (4 from bus above, 4 from bus below). Over 4 cycles, each PE calculates a 4x4 SAD, with bigger CU sizes formed by passing generated PSADs to other PE or Adders. Henceforth, a cluster PE's to forming a bigger processing unit is refer to as a "Logical Core".

Output of the Logical Core is sent along the nearest comparator chain to compare and filter out results from other Logical cores. The final comparator filters out results from individual Row comparators, to report out the minimum SAD found by all Logical cores with its position.

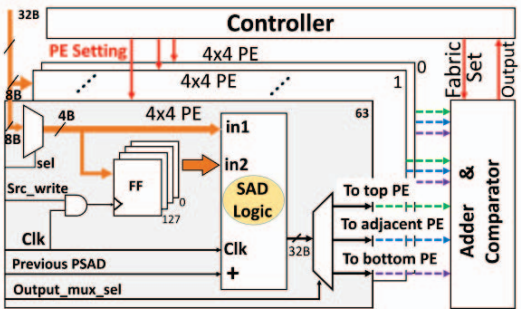

Fig. 4: NMC design for SAD with 64 PE clusters. Controller helps in offloading the task from processor and retrieving the output.

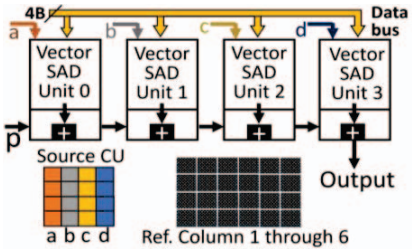

Fig. 5: SAD logic in the PE with vector compute capability.

C. Compute Grid

The design is composed of multiple 4x4 PE's, built of 4 4B column-wise partial SAD calculators pipelined to get the full SAD as shown in Fig. 5. The SAD Logic within the PE has 3 inputs: 4B data from the input mux, a 16B port carrying the loaded 4x4 CU and a 32B port "P" carrying partial SAD from preceding stage. During load state, 4x4 CU chunk (column data in the SRAM) is loaded into the PE after which the load flops are clock gated.

Fabric settings are appropriated before triggering the compute phase where columns from the search area are brought in every cycle. Each Vector SAD unit computes Partial SAD on 4B data broadcasted in with the 4B CU column. This result is added to the incoming partial SAD before passed to the next element in line. This aggregation of Partial SAD's with subsequent column engines mathematically works to building a bigger SAD Kernel on the column direction. The PE's in the grid work together in one of two possible ways:

1. All CU Mode*:* For a 32x32 CU, all of them work together to produce one 32x32 min SAD Kernel, with individual adders and comparators also generating smaller 16x16/8x8 min SAD's

2. Specific CU Mode**:** For a specific CU (smaller than32x32), PE's are aggregated differently to maximize number of Logical cores working in parallel to produce one min SAD. The role of the Reconfigurable fabric is to seamlessly alter data flow, to enable these two modes of operation.

D. Reconfigureable Fabric

The Reconfigurable Fabric lets our work attain SOTA performance with traditional RDO schemes while having an option to accelerate computation even further when paired with

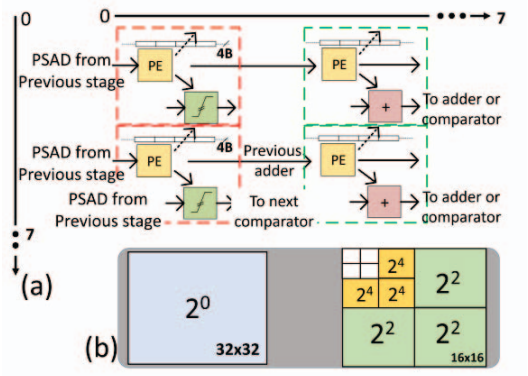

Fig. 6: (a) PE grid layout with Each PEs in the row alternating between adder and comparator connections. (b) Clustering of PE grid into logical cores based on CU mode.

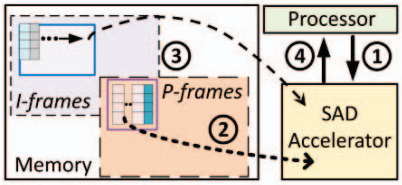

Fig. 7: Data flow sequence during MV compute. Each phase will occur only once and without back-forth data movement.

newer RDO approaches. The fabric configures (a) The 4B data (of 8B input) each PE works with (b) PE clustering for any mode to ensure maximal parallelism.

For a PE, the first part is addressed with a MUX at the input (shown in Fig. 4) which chooses data being fed in. The second part is answered by an output de-mux which lets every PE pass its output to adjacent PE, an adder or a comparator. Similarly, every adder has a choice to pass the result to another adder (to aggregate a bigger CU) or to a comparator (after reaching the required CU size). When following a traditional RDO approach, the fabric configures the compute grid to a data flow like [5] – where every PE picks 4B data from the top bus and is daisy chained to aggregate and compare SADs.

For modern RDO schemes where the CU size to be searched is known in advance, the fabric organizes all the available PE's for maximum parallelism. Required CU size is broken into smaller 4x4 blocks and mapped onto the PE grid to form multiple parallel logical cores. Each logical cores of a row work with a different 4B of input data to generate a min SAD. This flexible interconnect allows efficient HW utilization across all RDO modes, offering massive parallelism at reduced data movement cost.

IV. Analysis And Results

The proposed accelerator along with a memory module was implemented with Verilog-HDL and synthesized with Synopsys Design Compiler using Intel 22nm CMOS technology.

A. NMC Acceelrator Design Results

The proposed NMC Accelerator was synthesized at 500Mhz clock frequency residing next to an 8MB memory. The unit 4x4 PE tile within the accelerator occupies 3652um^2 area and consumes 59.7uW dynamic power for an operating voltage of 0.9V. With 230K gates, the Accelerator occupies a total area 0.0834mm^2 and consumes 5.46mW dynamic power under peak performance (1TOPS with all PEs working simultaneously).

TABLE I COMPARISON AGAINST OTHER ACCELERATORS

Architecture	*Latency (cycles)*	*Cycles/CU*	*Interface Width (B/cycle)*	*Reference Buffer (pixels)*
Zhang et al. [7]	N	$(P+1)^2 + N$	P+N	$N^2 + P*N$
Huang et al. [6]	N	$(P+1)^2+N-1$	2N	2N
Chen et al. [5]	N	$(P+1)^2 + N -1$	N+1	N(N+1)
Proposed Work	N	$P*(P-N+1) + N$	N	N

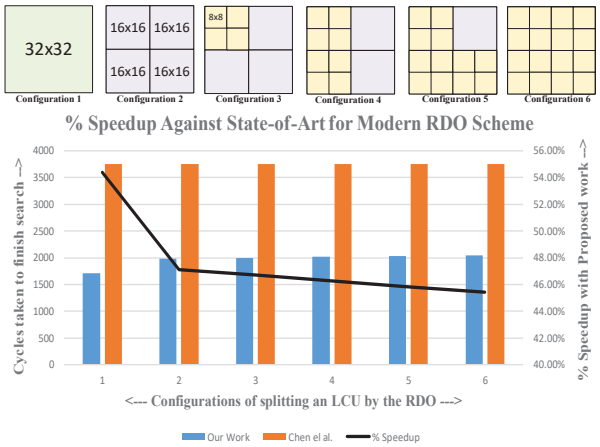

Fig. 8: Speedup against State-Of-Art when paired with recent advances in RDO Scheme

B. Performance Analysis

a) Comparison against Accelerators for traditional RDO Schemes: Table I summarizes comparison of various SAD processing architectures, against a similar traditional RDO workload - for an NxN CU searched in a PxP grid.

Our design achieves a better Cycles/CU with a lower reference buffer size, with similar Interface width and SAD Latency when compared against other SOTA work.

b) Evaluation with modern RDO Schemes: With recent RDO Algorithms, proposed reconfigurable fabric clusters PE's to maximize parallelism for any given CU mode. We picked a 60x60 search area and an LCU split shown in Fig 8. The proposed accelerator accelerates LCU processing by upto 55% when compared to [5]. Real-time processing was tested by clocking the system at 500Mhz to calculate clock cycle budget to process an LCU. As seen in Table II , across configurations, the accelerator finished computations within the budget.

TABLE II REAL TIME PROCESSING CAPABLITIES

Video Resolution (W x L x fps)				3840p	2160p	30fps
# of 32x32 LCU to process/second				243,000		
System Clock Frequency				500Mhz		
Cycles Available to process an LCU				2057 cycles		
LCU Configuration	1	2	3	4	5	6
Cycles to finish Search	1712	1984	2000	2016	2032	2048
As % of Cycles Alloted	83.2%	96.4%	97.2%	97.9%	98.6%	99.5%

c) Performance Comparison against CPU's: Similar comparison was run against a 12-core Intel Xeon E5 processor with AVX extensions running SVT Algorithm at 3Ghz. As shown in Table III, the Accelerator achieves a 3.6X-7X speedup over a multicore CPU for the same workload, with significantly lesser data movement.

TABLE III COMPARISON AGAINST CPU

Mode	*Search Area (PxP)*	*# cycles to complete search*		*Speedup*
		Accelerator (0.5Ghz)	*Xeon E5 (3Ghz)*	
8x8	128x128	384	16107	7X
32x32	36x36	144	3139	3.6X

V. Conclusion

Video data accounts for nearly 80% of the internet traffic with ME being a major bottlenecks for encoding. Our NMC solution for obtaining MV with a novel reconfigurable fabric ensures maximal parallelism, offering 7X speedup over CPUs, capable of meeting real time 4K 30FPS video encode and achieving up to 93 TOPS/W at 500MHz.

References

[1] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Sullivan, Gary J et al., "Overview of the high efficiency video coding (HEVC) standard." IEEE Transactions on circuits and systems for video technology 22: 1649-1668.

[3] Amirali Boroumand et al.2018. Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks. SIGPLAN Not. 53, 316–331. DOI:https://doi.org/10.1145/3296957.3173177

[4] C. Lee et al., "Multi-Pass and Frame Parallel Algorithms of Motion Estimation in H.264/AVC for Generic GPU," 2007 IEEE International Conference on Multimedia and Expo, Beijing, 2007, pp. 1603-1606.

[5] Ching-Yeh Chen et al. "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 53, no. 3, pp. 578-593, March 2006, doi: 10.1109/TCSI.2005.858488.

[6] Yu-Wen Huang et al. "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03., Bangkok, 2003, pp. II-II, doi: 10.1109/ISCAS.2003.1206094.

[7] Li Zhang and Wen Guo, "Improved FFSBM algorithm and its VLSI architecture for variable block size motion estimation of H.264," 2005 International Symposium on Intelligent Signal Processing and Communication Systems, Hong Kong, China, 2005, pp. 445-448, doi: 10.1109/ISPACS.2005.1595442.

[8] T. Komarek and P. Pirsch, "Array architectures for block matching algorithms," in IEEE Transactions on Circuits and Systems, vol. 36, no. 10, pp. 1301-1308, Oct. 1989, doi: 10.1109/31.44346.

[9] Z. Liu, S. Goto and T. Ikenaga, "Optimization of Propagate Partial SAD and SAD tree motion estimation hardwired engine for H.264," 2008 IEEE International Conference on Computer Design, Lake Tahoe, CA, 2008, pp. 328-333, doi: 10.1109/ICCD.2008.4751881.

[10] W. El-Harouni et al. "Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, Lausanne, 2017, pp. 1384-1389, doi: 10.23919/DATE.2017.7927209.

[11] L. Shen et al., "An Effective CU Size Decision Method for HEVC Encoders," in *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 465-470, Feb. 2013.

[12] Z. Pan et al., "Early MERGE Mode Decision Based on Motion Estimation and Hierarchical Depth Correlation for HEVC," in *IEEE Transactions on Broadcasting*, vol. 60, no. 2, pp. 405-412, June 2014, doi: 10.1109/TBC.2014.2321682.

[13] X. Liu et al. "An Adaptive CU Size Decision Algorithm for HEVC Intra Prediction Based on Complexity Classification Using Machine Learning," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 1, pp. 144-155, Jan. 2019, doi: 10.1109/TCSVT.2017.2777903.