A Configurable H.265-Compatible Motion Estimation Accelerator Architecture for Realtime 4K Video Encoding in 65 nm CMOS

Michael Braly, Aaron Stillmaker[a], and Bevan Baas
Department of Electrical and Computer Engineering
University of California
Davis, California
{mabraly, astillmaker, bbaas}@ucdavis.edu

***Abstract*—The design for a configurable motion estimation accelerator is presented and demonstrated as suitable for realtime digital 4K as well as H.265/HEVC. The design has two 4-KB frame memories necessary to hold the active and reference frames, designed using a standard cell memory technique, with line-based pixel write, and block-based pixel accesses. It computes a 16 pixel sum of absolute differences (SAD)s per cycle, in a 4×4 block, and is pipelined to take advantage of the high throughput block pixel memories. The architecture supports configurable search patterns and threshold-based early termination which allow for run-time tradeoffs to be made between pixel throughput and final quality of result. CMEACC is independently clocked and can operate up to 812 MHz at 1.3 V in 65 nm CMOS, achieving a throughput of 105 MPixel/sec for a single instance while consuming 0.933 pJ×sec/Pixel, and occupying approximately 1.04 mm^2 post place-and-route in 65 nm CMOS. While operating at 0.9 V, the presented design consumes 0.393 nJ/Pixel, which scales to 8.06 mW at 22.26 FPS in 720p.**

I. Introduction

As the number of pixels in video streams continues to increase and new video coding standards are introduced to cope with the increased compute requirements, new scalable hardware architectures are needed to perform these operations in real-time. The goal of digital video compression is to reduce the size of a video stream by identifying redundant information, removing it, and replacing it with a scheme to recreate that information during decompression.

There are two kinds of redundancies: *inter*-frame redundancy between frames in a video stream, and *intra*-frame redundancy within a single frame of a video stream. Stated another way: *inter*-frame redundancy describes repetition of data over *time* while *intra*-frame redundancy describes repetition of data over *space*. An object which is present throughout an entire sequence of frames would be an example of the kind of redundancy that *inter*-frame compression seeks to remove. A large section of blue sky taking up the top-half of a scene would be the sort of information redundancy that *intra*-frame compression would remove.

Redundancy is a qualitative description of an effect that humans see. The computer must be able to *quantify* the similarity between two sets of images. This quantification process generates a *figure of merit* which can be used to determine whether or not the two images are redundant enough to remove without significant loss of image quality. Two figures of merit are mean absolute error (MAE) and sum of absolute differences (SAD) [1]. These figures of merit are applied to pixel differences between the images. In the video coding standards that this work addresses (H.264 and H.265), the accepted figure of merit is SAD, an example of which is shown in Fig. 1.

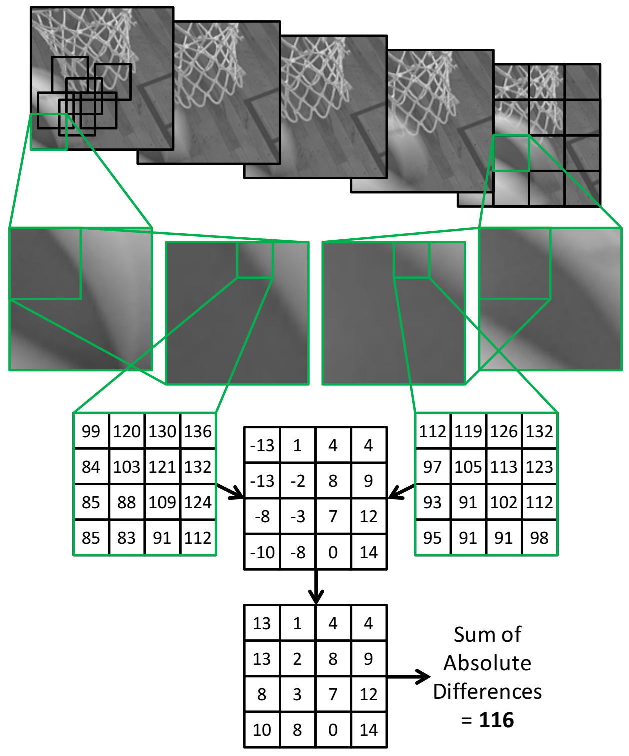

Fig. 1: Example of a sum of the absolute differences (SAD) computation, with the *current* frame on the right. Subslices of each frame are taken from 128x128, to 64x64, to 16x16 before computing the SAD of both 16x16 blocks.

IEEE promotes a standard for video coding referred to as H.264 [2], and published a new standard H.265, in 2013 [3]. These standards allow hardware designs for encoding and decoding video to be developed separately. The primary goal of the H.265 coding standard was to increase the compression efficiency of video streams by 50% without negatively impacting the overall video quality [4]. Initial analysis of the H.265 standard indicates that the standard meets that goal, with demonstrations on multiple video streams [5]. Each of these standards contain a set of tools to use to compress a video stream. For H.265, the various effects of each of these tools has been broken out into different levels, attempting to define a smooth tradeoff curve between computational complexity and final result quality [6].

[a]Now at the Department of Electrical and Computer Engineering, California State University, Fresno,

II. Previous Work

Significant work has been done in the motion estimation accelerator design space. Systolic array-based designs could initially handle the limited frame search areas of previous video coding standards, but as the effective search area has grown, up to a 128×128 pixel search area from the original 16×16 pixel search areas of past standards, systolic designs became more difficult to scale efficiently. Further analysis of available video streams have shown that 99.4% of the best block candidates are found in a 128×128 pixel search area [7].

A. Systolic Array Solutions

Systolic array implementations are motion estimation engines that make use of many parallel processing elements to generate the SADs for macroblocks as the image frame streams into the device. Lai and Chen introduced a 2D full-search block matching algorithm architecture, which achieved 100% hardware utilization in a tile-able architecture [8]. This architecture used a total of 256 PEs to process a 16×16 macroblock within a search area of $[-8, +7]$ in both the X and Y directions and was scalable to process the same macroblock across a search range of $[-16, +15]$ with 1024 PEs. Elagamel introduced an early termination mechanism in a systolic array that disabled PEs that were not going to produce a competitive matching candidate as well as the accumulation adders on the edge of the array, which saved 45% power over a normal array, by reducing the total number of comparisons by 50% [9]. Both of the previous designs could handle only fixed block sizes after implementation. Huang introduced a 2D systolic array implementation that was less efficient, with the PE array being only at 97% utilization, but capable of variable block size computations, chosen at run time, suitable for processing 720×480 video at 30 FPS [10]. This design also made use of a rectangular search range, with a larger search area in the horizontal direction $[-24, +23]$ than the vertical direction $[-16, +15]$. Deng expanded the search area of Huang to $[-32, +31]$ in both directions and scaled it up to handle 720×576 video at 30 FPS, at the cost of roughly double the total number of gates [11].

Chen et al. give an analysis of the cost of supporting variable block size motion estimation (VBSME) in systolic array style implementations, and propose an architecture suitable for 720p 30 FPS processing [12]. Their design makes use of pixel truncation, rounding to 5 bits for each pixel. The distortion from the loss of 3 LSB was about 0.1 db, while 4 LSB reduction costs 0.2 dB. Additionally, they make use of a prediction unit to choose which area of the search range their implementation will check, reducing the total area which needs to be searched, though rapid changes in direction will cause their prediction algorithm to miss.

B. Block Motion Estimation and Search Patterns

There are other motion estimation engines that use different architectures from 2D systolic arrays. These designs make use of search patterns, picking fewer points to sample using a strategy to trade PSNR loss for faster processing and significantly fewer points checked overall. Chun et al. modified a programmable DSP processor architecture to fetch and perform a subtract, absolute, add operation on 8 pixels at a time in the same cycle it fetches the next 8 pixels, resulting in a 20× speedup over a SISD architecture [13]. Since they were extending a programmable processor, their implementation could be extended to cover a wide range of search patterns, though they used it primarily with three step searches (TSS, typically Diamond-Diamond-Cross). Fatemi et al. experimented with using pixel truncation alongside bit-serial pipeline architecture to improve throughput further, while paying a similar cost to PSNR [14]. Their implementation looks similar to a 2D systolic array implementation, but its use of a bit-serial architecture instead of a bit-parallel one, distinguishes it.

Vanne et al. developed their own motion estimation implementation with design time configuration of search patterns and block access memory architectures [15]. This design can process 1080p video at 30 FPS while consuming 123 mW, and they demonstrated its robustness across five different search patterns. They also discussed, in detail, the math necessary to have separable memory addresses such that the pixel memory can be written in lines, but accessed in blocks. Their contribution was the primary starting point of our design.

Diamond search patterns have been built into fixed pattern motion estimators, where repeated repetitions of the diamond pattern can manage 1080p video frames at 55 FPS [16]. The number of points in a particular search pattern directly effects its computational complexity, but cross-based patterns miss diagonal movement. Purnachand looked into hexagonal patterns, recognizing that there are two types, called now HexA and HexB, which are biased in either the vertical or horizontal direction. Further work on search patterns have lead to back and forth hexagonal search patterns of type A and B, such as HexABA and HexBAB, which save 23% number of points checked versus the diamond patterns used in other accelerators [17].

Xiao et al. demonstrated a fully-featured H.264 compatible encoder on a 167-core asynchronous array of simple processors (AsAP) platform [18]. The design used a dedicated motion estimation accelerator by Landge et al. [19], along with 115 of the simple cores to implement a design suitable for 640×480, 21 FPS video encoding for 931 mW average power consumption. The design could also be scaled to the workload by managing the power supplies, from 95 inter FPS at 0.8 V to 478 inter FPS at 1.3 V in QCIF frames [20]. A better way of thinking of this is that the design could operate anywhere from 20% to 100% of its maximum throughput capacity by controlling the core voltage levels.

Kim and Sunwoo introduced an application specific processor that they called MESIP, which was capable of 720p, 50 FPS processing for 22.22 mW and a total of 8 KB of SRAM [21]. The MESIP required the development of its own software tools, but can leverage those tools to optimize data-reuse strategies. The execution unit of the MESIP resembles the 2D systolic arrays, but the memory management and search pattern functionality provided by its control unit removes it from the 2D systolic array class.

C. Standard Cell Memories

Meinerzhagen explored standard cell memories (SCM) in 65 nm, demonstrating memories with a 49.98% area penalty in trade for a 36.54% power reduction for the overall memory array [22]. Further investigation into how such memories stack up in the subthreshold domain, compared to SRAM macros, found that these SCMs were better than standard SRAM macros, but worse than full custom macros designed specifically for subthreshold operation [23]. This research, however, also surfaced the idea that these SCMs could be used in distributed memory blocks closely integrated with logic, and further, that these memories would work consistently with their accompanying logic. For a design that makes use of voltage dithering, low operating voltage, or other similar power control techniques, these memories would be very suitable. Meinerzhagen also demonstrated a 4-kb SCCM built with an automated compilation flow and demonstrated its reliability at subthreshold voltages [24].

III. Architecture

The configurable motion estimator accelerator (CMEACC), shown in Fig. 2 builds on Vanne's block-addressed memory [15] and search pattern encoding motion estimator and Meinerzhagen's SCM [23]. This is a natural extension, since the block-addressed memory architecture results in highly fragmented memory blocks which serve very particular parts of the datapath, as illustrated in Fig. 9, where the optimal placement of the reference frame memory, as dictated by the place-and-route tool, was distributed across the die. Additionally, those fragments are the correct size to outperform SRAM macros in terms of performance, without paying the full density penalty, as previously described by Meinerzhagen [23]. The use of SCMs also allows a power-conscious system on chip (SoC) which incorporates CMEACC to operate the entire block on a single low, near-threshold voltage. Our design is implemented as an accelerator for a SoC [25].

The accelerator can be conceptualized as a specialized micro-controller. It has its own instruction set, communicates with other blocks through input and output FIFOs, and has its own clock and sleep signals, which makes the design with respect to the other modules in the chip globally asynchronous locally synchronous (GALS). This encapsulation makes it straightforward to integrate as many accelerators as desired by the overall system designers of an SoC. These FIFOs do not limit the maximum throughput of CMEACC, as the block operating frequency of 812 MHz is sufficient even at 50% FIFO utilization to support the pixel transfers necessary for processing digital 4K at 60 FPS.

A top level block diagram of the entire accelerator is shown in Fig. 2. It's assumed that the input and output dual-clock FIFOs lead to separate modules with asynchronous clocks, but this is not architecturally necessary, and it is possible for the same module to act as both transmitter and receiver to CMEACC. This is made possible by the transmit and receive commands both being part of the same instruction set with non-overlapping opcodes. The device is capable of both full-search and pattern-search operations, by use of a pattern memory. Patterns are stored using the same encoding proposed by Vanne et al. [15]. This pattern memory is implemented using SCMs combined with a ROM, encoded with several different potential patterns. This lets a user pick between full-search, built-in patterns, or a programmable pattern depending upon user needs for throughput and overall search quality. Additionally, the user-defined and built-in patterns share the same pattern memory address space, so a user can define the first stage of a pattern and then use the built-in stages to finish.

The pixel datapath is a carry-save adder tree, pipelined for throughput, combined with a pixel rotation block from the active frame memory to deal with the offset introduced by the block memory addressing scheme. A pipeline diagram of the pixel datapath is shown in Fig. 3. The depth of the pipeline needed to be balanced against the nature of search patterns, where a number of candidate blocks are examined before a search-stage decision is made. If the datapath is pipelined too deeply, there are many wasted cycles, and the pipeline empties as the search-stage decision is made by the controller. An overall search controller manages the execution of the search and which candidate blocks are examined. An additional circuit checks to see whether all the necessary pixels for the block compare are in reference frame memory before executing the search; if they are not in memory, the block issues a memory request and stalls the pipeline until the pixels can be fetched.

A. Scalability

One of the advantages of building CMEACC so that its local working memory can contain an entire H.265-specified tile, is that multiple instances of CMEACC can can then scale smoothly to encoders which process tiles in parallel. Each image stream is divided into 256x256 tiles, and each tile can be processed separately. For an 832x480 stream, the partitioning fills 3 tiles completely, and 5 partial tiles. Since our simulations were run in series for each tile, with only one instance of CMEACC, the work can be sped up at least 3 times as 3 tiles can be kept at full utilization, while partial tiles have less utilization. Similarly, for a 1280x720 stream, there are 8 full tiles and 7 partial tiles, resulting in, at minimum, an 8x speedup. This additional silicon area is not free, especially in power and memory bandwidth terms, but if a system calls for maximum throughput, the architecture can be scaled to meet that throughput requirement.

IV. Controller Implementation

The control unit consists of the configuration registers, pattern memory, full-search address generator, pattern-memory address generator, out of bounds point checker, the controller FSM, and an instruction decoder, as shown in Figure 4. The instruction decoder samples the op-code bits of every input word and translates these into control signals for the controller FSM. In order to prevent random bits in the pixel transfers from being misinterpreted, all instruction decode signals pass through the controller FSM, where they are masked if the controller is not in an instruction-receiving state. Both address generators can generate the next inspection point for either a smart full-search or a pattern search run out of the pattern memory. The address out of bound checker, combined with the controller FSM handles pixel replacement for the reference frame memory.

The top FSM controller is not a monolithic FSM. Instead it is a series of hierarchical FSMs. Hierarchical finite state machines are a technique for managing the complexity of a controller with many separate states, but relatively ordered transitions [26]. These hierarchical FSMs are built so that there is no latency lost when traveling down the hierarchy,

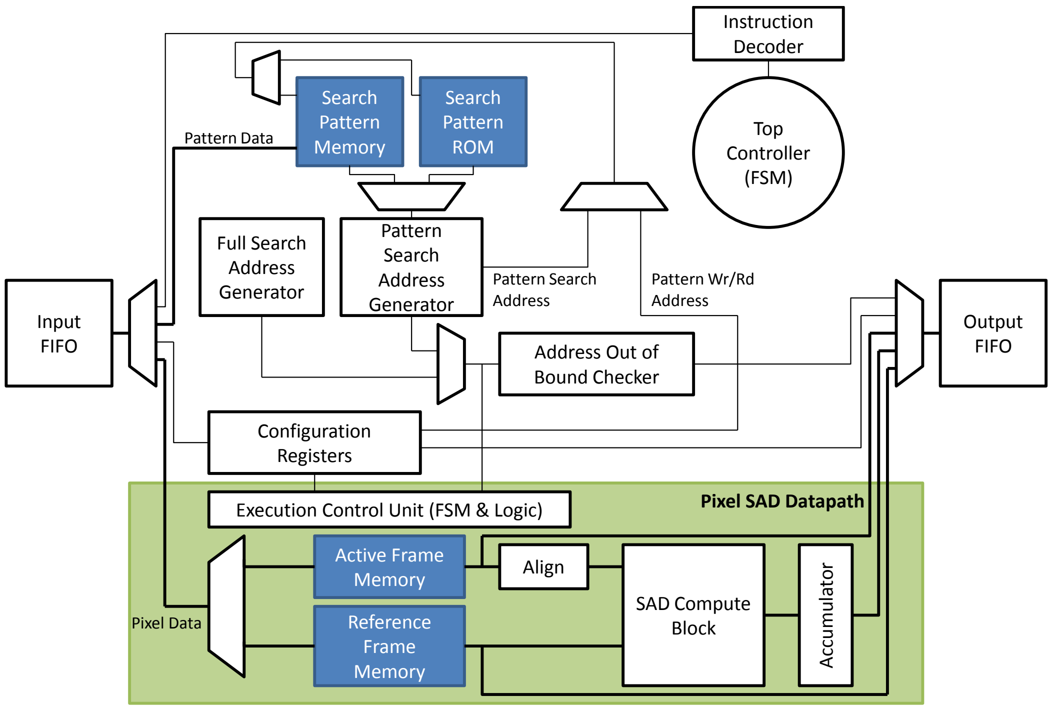

Fig. 2: Top level block diagram of the CMEACC design.

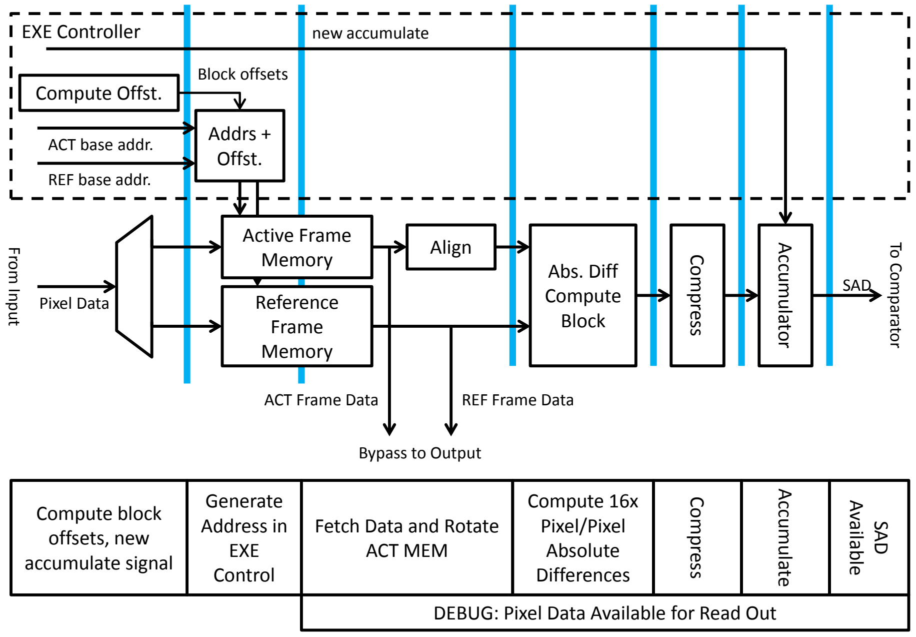

Fig. 3: Pipeline diagram of the pixel datapath of the CMEACC.

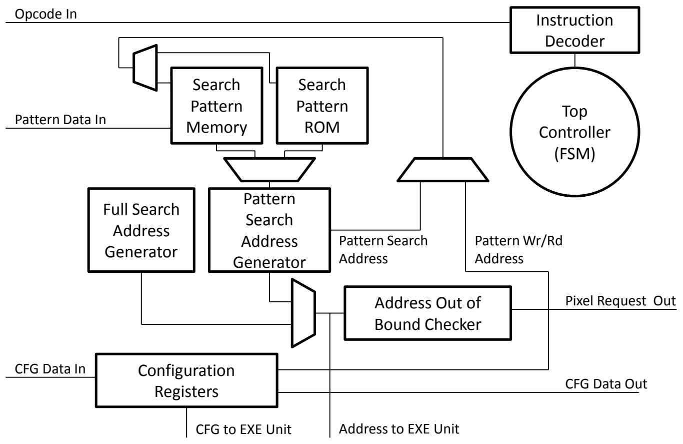

Fig. 4: A block diagram showing the control unit, including memory as well as data, address, and control lines.

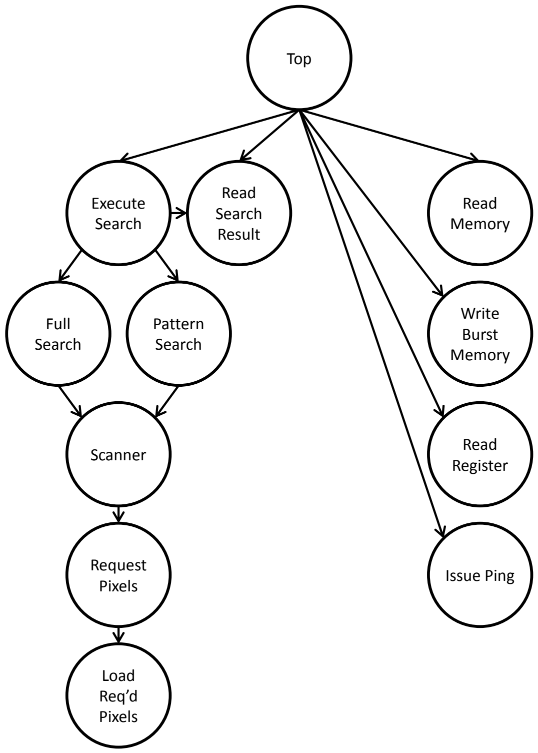

Fig. 5: FSM hierarchy of the top control unit of the CMEACC.

which requires careful handling of the idle states in each machine. This allows us to retain the full efficiency of a fully integrated top level FSM, without paying as much of the complexity price in terms of analysis and difficulties in correct implementation. The list of the component FSMs, and the relational hierarchy, is shown in Figure 5. Since both full search and pattern search make use of pixel replacement, the actual implementation of the execute search contains mux logic to arbitrate between which FSM has control of the scanner FSM. The state transition diagram is shown in Figure 6 with the hierarchical FSMs marked in dashed borders. The return to IDLE behavior adds latency to the rare register and pattern memory writes. Searches and their associated memory

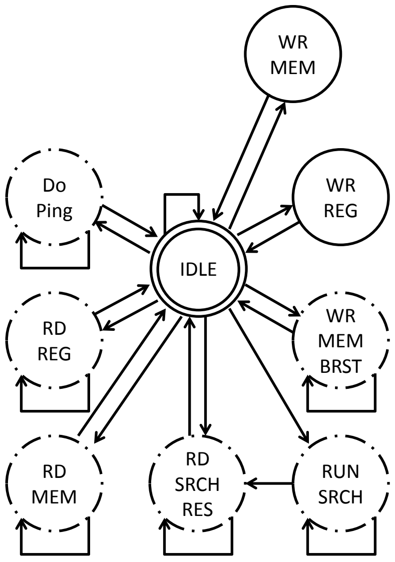

Fig. 6: State diagram of the top level controller. States which trigger other FSMs are given in dashed circles, and the reset state is shown with a double circle.

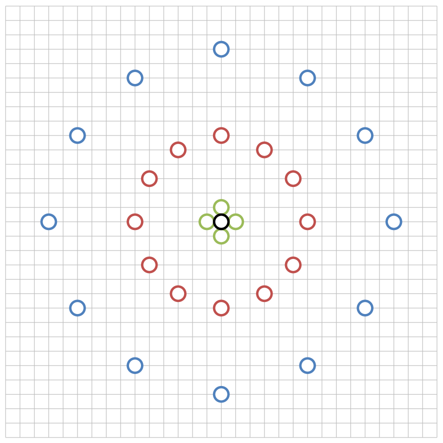

Fig. 7: 3-Stage, 12-point circular search pattern showing the three pixel search stages on an image.

operations are handled by a lower level state machine and are set up to be pipelined. The read out commands have their own state machines so that CMEACC can stall correctly if its output FIFO is full.

V. A 12-Point Circular Search Pattern

In the course of developing and testing CMEACC it became apparent that the current search pattern methodology could be extended to trade further compute for distortion. A cross pattern, for instance, captures motion in only the cardinal directions, while a diamond pattern captures motion in both the cardinal and diagonal directions. Hexagonal patterns capture motion, biased in either the horizontal or vertical direction depending upon the type of hexagon (type A or type B). All of these search patterns were developed in the context of H.264 and previous standards, where the maximum image size only went to 1080p. Movement in the cardinal

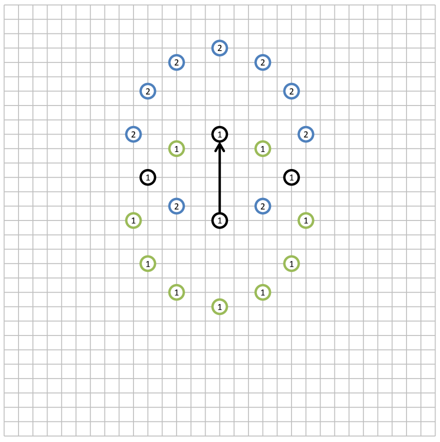

Fig. 8: Circular pattern type I reuse, showing how overlapping pixels can be reused from previous searches.

directions and the diagonals, then, would capture most of the movement possible in a particular frame. With larger image sizes, up to 4x the size of 1080p, motion within the image may fall within the areas missed by cardinal and diagonal motion vectors. At the same time, H.265 brings in additional motion vectors as possible candidates and with process shrink, the actual *computation* of a candidate SAD, once its relevant pixels have been brought into memory, is also relatively less expensive. Therefore, additional patterns which contain more search points (and require more compute), but cover more possible motion vectors, can become advantageous. A 12 point circular pattern, with a three-stage example shown in Figure 7, balances keeping the total number of points searched low, while still covering more possible motion directions. It also has the same overlapping characteristics of diamond, cross, and hexagonal patterns, where repeated searches at the same stage have overlapping check points which can be skipped, as shown in Figure 8. This reuse of 3 points is less than the reuse of the diamond pattern, which reuses either 3 or 5 points depending upon the movement type, comparable to hexagonal patterns which also reuse 3 points, and results in less distortion on average than the cross pattern, which reuses only 1 point. Table I gives a breakdown of points reuse in different patterns, excluding the center point of the pattern. As a percentage measure, the Circular pattern's per-stage pixel reuse is equivalent to the cross, while checking 3 times the total number of points results in less distortion.

TABLE I: Point reuse between stages in various search patterns.

Pattern	NumPts	Reuse	Reuse Pct.
Cross	4	1	25%
Diamond	8	3 or 5	38% - 50%
HexA	6	3	50%
HexB	6	3	50%
Circular	12	3	25%

VI. Results

The CMEACC architecture was synthesized using a low leakage 65 nm CMOS standard cell library, then placed and routed to a final design where it measured $1.04\,\mathrm{mm}^2$. A plot showing the resulting design is shown in Fig. 9. Results were collected at $1.3\,\mathrm{V}$ and $0.9\,\mathrm{V}$. The design was able to reach a maximum operating frequency of $812\,\mathrm{MHz}$ at $1.3\,\mathrm{V}$. Throughput for the design was modeled by replicating the design in Matlab, maintaining bit and cycle accuracy, running that model against various model video streams with a variety of characteristics and multiple search patterns, and then using those model runs to generate stimulus patterns to run against the device RTL. When simulated on the RTL, the total number of clock cycles spent, including transferring the necessary pixels into the CMEACC and configuring a search pattern, were collected. Final power and cycle period values were taken from place and route. Overall, in the video streams run, between 55.78% and 82.62% of the cycles were spent fetching or reading pixels from external memory, and the remaining 31.80% and 17.23% of the cycles were spent computing the SAD values, heavily dependent upon search pattern and video stream.

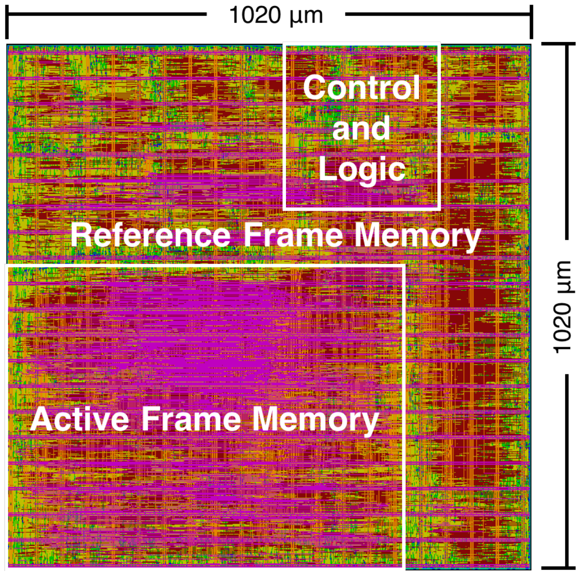

Fig. 9: A plot of the physical layout of the CMEACC which measures $1020\,\mu\mathrm{m}\times1020\,\mu\mathrm{m}$. The two types of memories, implemented with SCM, as well as the control and datapath logic are highlighted.

Comparisons against recent motion estimator hardware are shown in Table II. Note that a majority of the designs reported results from synthesis, which tends to be optimistic when compared to results from a full layout that have gone through the place and route step, as the CMEACC reported values have. Throughput was calculated as the total number of pixels processed, attained by multiplying the frame size by the FPS. As shown in the table, the CMEACC has the highest throughput at 105 MPixel/sec and lowest energy×time, at 0.933 pJ×sec/Pixel. At 0.9 V, CMEACC requires only 0.393 nJ/Pixel, which is believed to be the highest energy efficiency reported for a hardware motion estimator accelerator.

VII. Conclusion

We have designed and implemented a new, modular, motion estimation engine architecture, CMEACC suitable for

TABLE II: Comparisons of results with recent application specific hardware for motion estimation.

Work	Process (nm)	Voltage (V)	Alg.	Supported Block Sizes	Format	Die Area (mm^2)	Clock Freq. (MHz)	FPS	Power (mW)	Throughput (MPixel/sec)	Energy (nJ/Pixel)	Energy×Time (nJ×sec/Pixel)
Chun [13]	-	-	TSS	16×16	CIF	-	250*	24*	-	2.43*	-	-
Fatemi [14]	180	-	FS	4×4-16×16	CIF	-	440*	41*	-	4.16*	-	-
Vanne [15]	130	1.2	Prog.	4×4-16×16	1080p	-	200*	30*	59*	62.2*	0.948*	4740*
Landge [19]	65	1.3	FS	4×4-16×16	CIF	**0.672**	938	**210**	195	21.3	9.16	9770
Kim [21]	90	1.08	Prog.	4×4-16×16	720p	-	150*	50*	22.2*	46.1*	0.482*	3220*
CMEACC	65	1.3	Prog.	4×8-64×64	720p	1.04	812	114.4	79.8	**105**	0.757	**933**
CMEACC	65	0.9	Prog.	4×8-64×64	720p	1.04	158	22.26	**8.06**	20.5	**0.393**	2490

* These values were taken from synthesis.

use with modern video coding techniques, and with sufficient throughput to sustain real time 4K video streams. The device builds upon previous work on motion estimation hardware, while incorporating standard cell memories to implement the frame and pattern memories, pipelining the pixel datapath, and implementing a novel controller to handle memory access requests, pipeline control, and search pattern execution. It compares favorably in throughput and energy×time against previous works, while being more flexible in both block size and search pattern, which can both be configured at run time.

Acknowledgment

The authors gratefully acknowledge support from ST Microelectronics, C2S2 Grant 2047.002.014, NSF Grant 1018972 and 0903549 and CAREER Award 0546907, SRC GRC Grant 1598, 1971, and 2321 and CSR Grant 1659, and SEM.

References

[1] S. Vassiliadis *et al.*, "The sum-absolute-difference motion estimation accelerator," in *Euromicro Conference, 1998. Proceedings. 24th*, vol. 2, Aug 1998, pp. 559–566 vol.2.

[2] T. Wiegand *et al.*, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video Technology,*, vol. 13, no. 7, pp. 560–576, July 2003.

[3] J. Ohm and G. Sullivan, "High efficiency video coding: the next frontier in video compression [Standards in a Nutshell]," *Signal Processing Magazine, IEEE*, vol. 30, no. 1, pp. 152–158, Jan 2013.

[4] G. Sullivan *et al.*, "Overview of the high efficiency video coding (HEVC) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[5] H. Koumaras, M. Kourtis, and D. Martakos, "Benchmarking the encoding efficiency of H.265/HEVC and H.264/AVC," in *Future Network Mobile Summit (FutureNetw), 2012*, July 2012, pp. 1–7.

[6] P. Helle *et al.*, "A scalable video coding extension of HEVC," in *Data Compression Conference (DCC), 2013*, March 2013, pp. 201–210.

[7] M. Sinangil *et al.*, "Memory cost vs. coding efficiency trade-offs for hevc motion estimation engine," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, Sept 2012, pp. 1533–1536.

[8] Y.-K. Lai and L.-G. Chen, "A data-interlacing architecture with two-dimensional data-reuse for full-search block-matching algorithm," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 8, no. 2, pp. 124–127, Apr 1998.

[9] M. Elgamel, A. Shams, and M. Bayoumi, "A comparative analysis for low power motion estimation VLSI architectures," in *Signal Processing Systems, 2000. SiPS 2000. 2000 IEEE Workshop on*, 2000, pp. 149–158.

[10] Y.-W. Huang *et al.*, "Hardware architecture design for variable block size motion estimation in MPEG-4 AVC/JVT/ITU-T H.264," in *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, vol. 2, May 2003, pp. 796–799.

[11] L. Deng *et al.*, "An efficient hardware implementation for motion estimation of avc standard," *Consumer Electronics, IEEE Transactions on*, vol. 51, no. 4, pp. 1360–1366, Nov 2005.

[12] C.-Y. Chen *et al.*, "Analysis and architecture design of variable block-size motion estimation for H.264/AVC," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, no. 3, pp. 578–593, March 2006.

[13] Z. Chun *et al.*, "A DSP architecture for motion estimation accelerating," in *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*, Oct 2004, pp. 583–586.

[14] M. Fatemi, H. Ates, and R. Salleh, "A bit-serial sum of absolute difference accelerator for variable block size motion estimation of H.264," in *Innovative Technologies in Intelligent Systems and Industrial Applications, 2009. CITISIA 2009*, July 2009, pp. 1–4.

[15] J. Vanne *et al.*, "A configurable motion estimation architecture for block-matching algorithms," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 4, pp. 466–477, April 2009.

[16] G. Sanchez *et al.*, "High efficient motion estimation architecture with integrated motion compensation and fme support," in *Circuits and Systems (LASCAS), 2011 IEEE Second Latin American Symposium on*, Feb 2011, pp. 1–4.

[17] N. Purnachand, L. Alves, and A. Navarro, "Fast motion estimation algorithm for hevc," in *Consumer Electronics - Berlin (ICCE-Berlin), 2012 IEEE International Conference on*, Sept 2012, pp. 34–37.

[18] Z. Xiao, S. Le, and B. Baas, "A fine-grained parallel implementation of a H.264/AVC encoder on a 167-processor computational platform," in *Asilomar Conference on Signals, Systems and Computers*, Nov 2011, pp. 2067–2071.

[19] G. Landge, "A configurable motion estimation accelerator for video compression," Master's thesis, University of California, Davis, CA, USA, Dec. 2009, http://www.ece.ucdavis.edu/vcl/pubs/theses/2009-4.

[20] Z. Xiao, "Energy-efficient fine-grained many-core architecture for video and dsp applications," Ph.D. dissertation, University of California, Davis, CA, USA, Dec. 2012, http://www.ece.ucdavis.edu/vcl/pubs/theses/2012-4/.

[21] S. D. Kim and M. H. Sunwoo, "MESIP: A configurable and data reusable motion estimation specific instruction-set processor," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 10, pp. 1767–1780, Oct 2013.

[22] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, Aug 2010, pp. 129–132.

[23] P. Meinerzhagen *et al.*, "Benchmarking of standard-cell based memories in the sub- V_T domain in 65-nm CMOS technology," *Emerging and Selected Topics in Circuits and Systems, IEEE Journal on*, vol. 1, no. 2, pp. 173–182, June 2011.

[24] ——, "A 500 fW/bit 14 fJ/bit-access 4kb standard-cell based sub-VT memory in 65 nm CMOS," in *ESSCIRC (ESSCIRC), 2012 Proceedings of the*, Sept 2012, pp. 321–324.

[25] M. Braly, "A configurable H.265-compatible motion estimation accelerator architecture suitable for realtime 4K video encoding," Master's thesis, University of California, Davis, Davis, CA, USA, Dec. 2015, http://vcl.ece.ucdavis.edu/pubs/theses/2015-2.braly/.

[26] M. Keating, *The Simple Art of SoC Design: Closing the Gap Between RTL and ESL.* Springer Science & Business Media, 2011.

[27] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration, the VLSI Journal*, vol. 58, pp. 74–81, 2017, http://vcl.ece.ucdavis.edu/pubs/2017.02.VLSIintegration.TechScale/.