

# A Fully Integrated Energy-Efficient H.265/HEVC Decoder With eDRAM for Wearable Devices

Mehul Tikekar, *Member, IEEE*, Vivienne Sze, *Senior Member, IEEE*,  
and Anantha P. Chandrakasan, *Fellow, IEEE*

**Abstract**—This paper proposes a fully integrated H.265/high efficiency video coding (HEVC) video decoder that supports real-time video playback within the 50-mW power budget of wearable devices, such as smart watches and virtual reality (VR) headsets. Specifically, this paper focuses on reducing data movement to and from off-chip memory as it dominates energy consumption, consuming 2.8–6 times more energy than processing in most video decoders. Embedded dynamic random access memory (eDRAM) is used for main memory, and several techniques are proposed to reduce the power consumption of the eDRAM itself: 1) lossless compression is used to store reference frames in two times fewer eDRAM macros, reducing refresh power by 33%; 2) eDRAM macros are powered up on-demand to further reduce refresh power by 33%; and 3) syntax elements are distributed to four decoder cores in a partially compressed form to reduce decoupling buffer power by four times. These approaches reduce eDRAM power by two times in a fully integrated H.265/HEVC decoder with the lowest reported system power. The test chip containing 10.5 MB of eDRAM requires no external memory and consumes 24.9–30.6 mW for decoding 1920 × 1080 video at 24–50 frames/s.

**Index Terms**—Embedded dynamic random access memory (eDRAM), H.265/high efficiency video coding (HEVC), reference frame compression (RFC), video coding, wearable devices.

## I. INTRODUCTION

WEARABLE devices are gaining popularity in the form of smart watches and virtual reality (VR) headsets. These devices have stringent power and footprint constraints that limit their ability to support video playback. For instance, the power budget can be as low as 50 mW [1], which makes it extremely challenging to perform video coding, especially for state-of-the-art video coding standards, such as H.265/high efficiency video coding (HEVC) [2], [3]. Most of the previous work on hardware video decoders has focused on improving the energy efficiency of the decoder chip [4]–[6]. However, from a system perspective, the energy required for video decoding is dominated by memory access to off-chip dynamic random access memory (DRAM). Some of the more recent works [7], [8] have demonstrated techniques, such as caching, reference frame compression (RFC), and DRAM-aware memory mapping, to reduce off-chip memory power. However, even with these techniques, memory accesses require

2.8–6 times as much energy as the video decoder, thus remaining the main obstacle in the path of reducing energy consumption.

In this paper, we propose techniques to effectively use embedded DRAM (eDRAM) in a fully integrated H.265/HEVC decoder achieving two times energy saving in eDRAM itself and six times energy savings compared with a decoder system using off-chip double data rate 3 memory [9]. While the eDRAM reduces system power and physical footprint relative to the traditional DRAM, it requires more frequent refresh due to two factors. First, the DRAM uses larger bit-cell capacitance and access transistors with lower leakage than the eDRAM. Hence, eDRAM bit-cells need more frequent refresh (on the order of microseconds) compared with the DRAM (on the order of milliseconds) [10]. Second, for typical HEVC decoding workloads, tens of eDRAM macros are needed to meet the storage requirement, but just one to two macros are sufficient to meet the bandwidth requirement. All the macros need to be refreshed to retain stored data even though only one to two macros are active. As a result of these two factors, the number of refresh commands sent to the eDRAM macros exceeds the number of read and write commands, and the energy cost of using the eDRAM is dominated by refresh energy. Accordingly, in this paper, we focus on techniques to reduce eDRAM refresh energy by exploiting redundancy in the data and the low latency and energy/access of eDRAM. This paper makes the following contributions.

- 1) *Reduce Energy per Memory*: By using eDRAM to eliminate off-chip memory.
- 2) *Reduce Refresh Power of eDRAM*: By reducing the number of enabled eDRAM macros using RFC and an on-demand power-up scheme.
- 3) *Reduce the Number of Memory Accesses*: By using partial compression in the decoupling buffer, and RFC and caching for the frame buffer.

This paper is organized as follows. Section II gives an overview of the system architecture describing its use of parallel decoder cores and eDRAM macros. Section III focuses on techniques to reduce the number of enabled eDRAM macros to reduce refresh power, whereas Section IV discusses techniques to reduce the number of eDRAM accesses to reduce dynamic power. Finally, Section V presents measured results from a test chip.

## II. SYSTEM ARCHITECTURE

Fig. 1 shows the system block diagram of the H.265/HEVC video decoder chip. The decoding process is separated into two

Manuscript received January 15, 2018; revised April 21, 2018; accepted May 5, 2018. Date of publication June 6, 2018; date of current version July 20, 2018. This paper was approved by Associate Editor Edith Beigné. This work was supported in part by NSF and in part by the MIT Center for Integrated Circuits and Systems. (Corresponding author: Vivienne Sze.)

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: [sze@mit.edu](mailto:sze@mit.edu)).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JSSC.2018.2837124

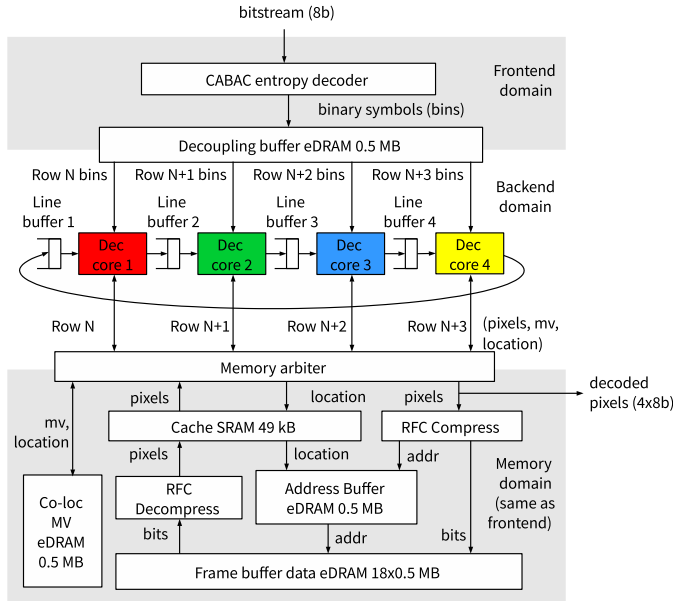


Fig. 1. High-level architecture of hardware HEVC decoder showing separate clock domains and four parallel decoder cores.

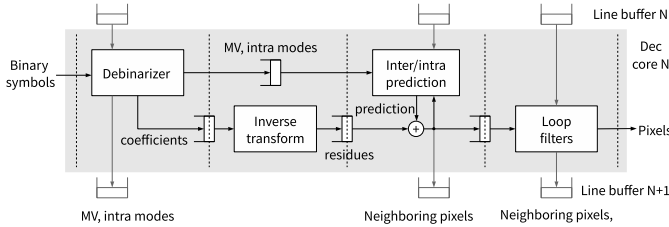


Fig. 2. Pipelining of decoder core showing the four processing elements: debinarizer, inverse transform, inter-prediction and intra-prediction, and loop filters. Pipeline FIFOs between the processing elements and line-buffer connections between decoder cores are also shown.

clock domains: 1) *Frontend and memory domain* contains the context adaptive binary arithmetic coding (CABAC) entropy decoder [11] to generate the binary symbols (bins) of the syntax elements [e.g., motion vectors (MVs), prediction modes, and transform coefficients] from the compressed bitstream, and frame buffer plus associated logic to store the reference frames for motion compensation and 2) *Backend domain* contains the decoder cores that generate the pixels from the bins of the syntax elements. To balance the throughput, the frontend operates at a higher clock frequency than the backend as the bit-level decoding is serial, whereas the pixel-level decoding can be parallelized using multiple cores.

#### A. Fine Grained Pipelining of Decoder Core

Each decoder core consists of modules, such as debinarizer, inverse transform [12], inter- and intra-prediction, and loop filters, as shown in Fig. 2. These modules are connected to each other in a pipelined fashion with FIFOs. In previous work, these pipeline FIFOs dominated the on-chip static random access memory (SRAM) cache [7]; this was due to the fact that the processing elements operated on a coding tree unit (CTU) granularity. To reduce the SRAM size, the processing elements in this chip were designed to operate on a transform unit (TU) granularity. Table I shows the difference in pipeline block size between this chip and that in [7].

TABLE I  
PIPELINE BLOCK SIZE FOR [7] AND CURRENT CHIP. PIPELINE BLOCK SIZE IN CURRENT CHIP IS EQUAL TO LARGEST TU SIZE

Coding Tree Unit (CTU)	Pipeline block in [7]	Pipeline block in this work	Reduction in FIFO element size
$16 \times 16$	$64 \times 16$	$16 \times 16$	$4 \times$
$32 \times 32$	$64 \times 32$	$32 \times 32$	$2 \times$
$64 \times 64$	$64 \times 64$	$32 \times 32$	$4 \times$

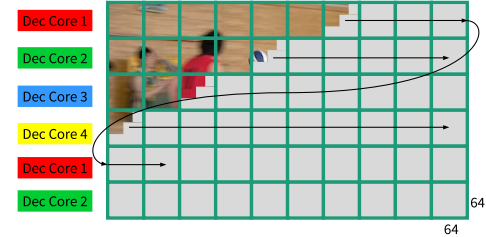


Fig. 3. Parallel processing of video frame by four *Dec* Cores.

#### B. Parallel Decoder Cores

The chip uses four decoder cores (*Dec* cores 1–4 in Fig. 1), where each *Dec* core processes a row of CTUs as shown in Fig. 3. Using multiple decoder cores in parallel enables voltage and frequency scaling, although this test chip demonstrates frequency scaling only. In video decoding, each CTU has dependencies on its top and left neighbors due to intra-prediction and in-loop filtering. In previous work, where a single decoder core is used to decode the frame in horizontal raster-scan order, the data dependencies between CTU rows were managed by a single line buffer [7]. However, when multiple decoders are used, using a single line buffer requires tracking the dependencies for all decoder cores as follows.

- 1) *Dec* core  $N + 1$  must read a pixel from the line buffer only after *Dec* core  $N$  has written it.
- 2) *Dec* core  $N + 1$  must not read stale pixels from older CTU rows.
- 3) *Dec* core  $N$  must write new pixels to the line buffer only after *Dec* core  $N + 1$  is done with the old pixels in the same location.

This chip uses a scalable multicore design by using FIFOs between *Dec* cores instead of a common shared memory for line buffers as shown in Fig. 4. The *Dec* core is designed to have a regular access pattern on the line buffer irrespective of coding unit (CU), prediction unit (PU), or TU partitioning within a CTU. This allows the shared line buffer to be replaced by FIFOs between adjacent cores. The FIFO empty/full logic guarantees that the three conditions for dependence tracking are met. This design is easily scalable with a number of decoder cores as the arbitration logic for tracking dependencies is effectively decentralized.

The depth of the individual line buffers FIFOs can be designed to achieve appropriate scheduling of the decoder cores. Previous work [13] uses unequal FIFO depths as shown in Fig. 5(b). In this paper, four equally size FIFOs with a maximum depth of 8 are used between each row, which allows for small variations in processing times for CTUs to be averaged out. This results in a regular design and also provides

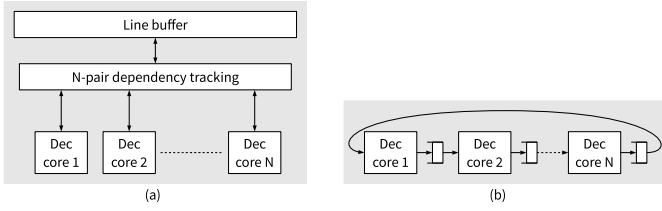


Fig. 4. (a) Shared line buffer requires complex dependence tracking for all pairs of adjacent decoders. (b) Designing the decoder cores to work with FIFO-based line buffers results in a more scalable design.

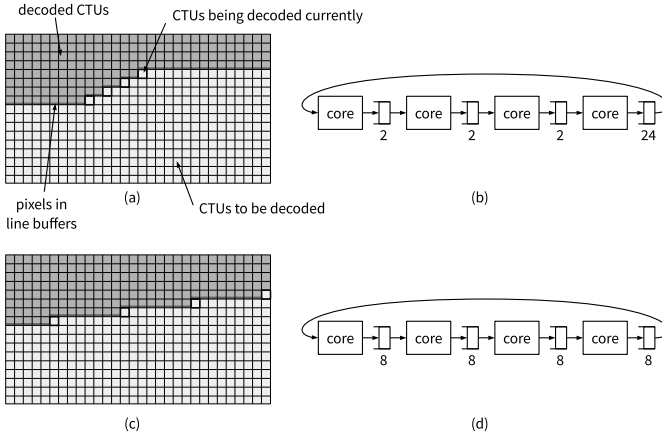


Fig. 5. Relative sizes of line-buffer FIFOs affect processing schedule of CTUs. (a) and (b) Approach taken in [13]. (c) and (d) Approach taken in this paper.

a slight improvement in throughput (about 5% on a test set of video sequences).

### C. Embedded DRAM as Main Memory

Despite various techniques to reduce off-chip memory power [7], [8], it continues to dominate system power consumption. Most of the power consumption comes from driving the large off-chip I/O capacitance. To address this, this work uses eDRAM as main memory. The chip contains a total of 21 eDRAM macros, each 0.5 MB in size. The macros are allocated as follows.

- 1) *Frame Buffer*: Eighteen eDRAM macros to store the two previous and one current frame (six macros per frame).
- 2) *Address Buffer*: One eDRAM macro to store the lookup table to translate pixel location in frames to byte address in the frame buffer. This is necessary to support lossless RFC as discussed in Section III-A.
- 3) *Colocated MV Buffer*: One eDRAM macro to store the colocated MVs from previous frames to support MV parsing for the current frame.
- 4) *Decoupling Buffer*: One eDRAM macro to distribute data from the entropy decoder in the frontend to the four parallel decoder cores in the backend.

One of the major drawbacks of using eDRAM is that it requires more frequent refreshes compared with DRAM. Section III will discuss techniques to address the eDRAM refresh power in order to reduce the refresh power of the system. While the eDRAM consumes less energy per access

compared with the DRAM, it still consumes more energy than standard on-chip SRAM. Section IV will discuss methods to reduce the number of accesses to these eDRAM macros in order to reduce the dynamic energy of the system.

### III. REDUCING REFRESH POWER OF eDRAM

In the hierarchy of memory technologies, the eDRAM stands between SRAM and DRAM in terms of energy/access and density. For example, in the 28-nm technology, the eDRAM has 321 times lower energy/access than the DRAM and 2.85 times higher density than the SRAM [14]. The key challenge with the eDRAM is that their bit cells require frequent refresh to retain data. Due to higher leakage and smaller bit-cell capacitance of the eDRAM process technology as compared with the DRAM process, refresh requests need to be made more frequently, and so refresh power is more significant on the eDRAM than the DRAM. The SRAM does not need to be refreshed as the data are actively maintained by cross-coupled inverters. Furthermore, for video decoding application, the instantaneous read–write bandwidth requirement can be satisfied by two to three eDRAM macros in the frame buffer, whereas the rest of the macros remain in self-refresh to retain data. As a result, the eDRAM energy is dominated by refresh power (80% of total eDRAM power and 40% of total chip power<sup>1</sup>).

Accordingly, we focus on reducing refresh power as a means to reduce the energy consumption of the eDRAM. The eDRAM macros can operate in three main modes.

- 1) *Active Mode*: Frequent read/write requests and regular refresh requests are sent to the macro.
- 2) *Self-Refresh Mode*: No read/write request is sent, but regular refresh requests are sent to retain stored data.
- 3) *Deep Power-Down Mode*: The macro is power-gated and all data are lost.

The deep power-down mode is the only mode in which refresh is disabled. Accordingly, we propose techniques to keep as many macros as possible in the deep power-down mode to minimize refresh power.

#### A. Reference Frame Compression

RFC is a popular technique used on video decoders with DRAM main memory to reduce off-chip bandwidth [15]–[17]. Data compression for memory savings has also been explored for processor caches [18]–[20] but for specific applications like video coding, we can use prior knowledge of structure in the data to design better compression methods.

To maintain compliance with the video coding standard, the RFC must use lossless compression which makes the compressed data variable length. In previous work [15], [16], the compressed data are stored at fixed offsets as shown in Fig. 6(a) in order to maintain random accessibility of data. The fixed offsets correspond to the maximum size of the

<sup>1</sup>The refresh power reduction techniques proposed in this paper become even more critical as more frames are stored on-chip. If the number of reference frames is increased from 2 to 16, as allowed by the HEVC standard, the refresh power would grow to 95% of total eDRAM power and increase the total chip by 110%.

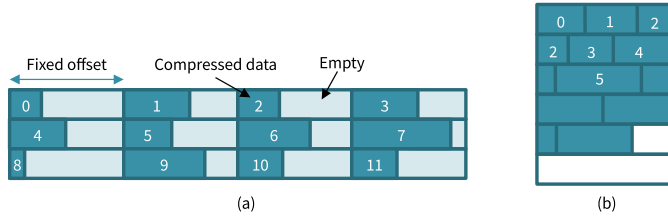


Fig. 6. (a) Traditional RFC using variable-length lossless compression stores compressed data at fixed offsets to maintain random accessibility at the cost of higher memory size and refresh power. (b) This paper stores compressed data in packed format to reduce eDRAM size and refresh power.

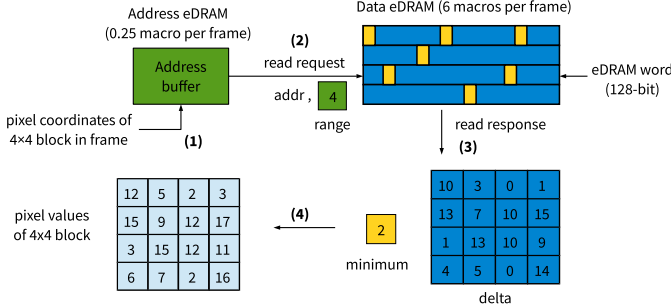


Fig. 7. Complete process of reading a  $4 \times 4$  block.

compressed data (which is greater than or equal to the size of uncompressed data due to lossless compression). The size of the compressed data to be read is stored in a separate on-chip buffer, which is read before the off-chip data are read. In this way, bandwidth reduction is achieved, without storage size reduction. Hence, this method is not useful for reducing eDRAM refresh power.

In this paper, we store the compressed data in the memory in a fully packed form as shown in Fig. 6(b). As a result, the starting byte address must also be stored along with the size of the compressed data in a separate address buffer. Reading the pixel values at a given location in the image involves the following steps (Fig. 7).

- 1) Convert pixel coordinates into an index of the address buffer.
- 2) Read starting byte address and compressed size from the address buffer. Use this to index into the data buffer.
- 3) Read compressed data from the data buffer.
- 4) Decompress the data.

1) *Lightweight RFC Algorithm*: A lightweight compression technique is applied on  $4 \times 4$  pixel blocks. Each  $4 \times 4$  block is compressed to three elements.

- 1)  $M$ : Minimum of the 16 pixel values (8 bit: 0-255).
- 2)  $R$  (*Log Range*): The number of bits required to represent the delta above the minimum (4 bit: 0-8).
- 3)  $D$ : Delta above  $M$  for 16 pixels using  $R$  bits (16R bits).

Fig. 8 shows an example of the proposed RFC algorithm. The compressed size is  $12 + 16R$  bits, for 128 bits of uncompressed data. In the special case when  $R = 8$ , no bit reduction is achieved by this algorithm. In this case, the minimum-delta representation is not used, and the pixels are stored as their original 8-bit values. The  $R$  value, which determines the size of the compressed data, is stored separately in the address

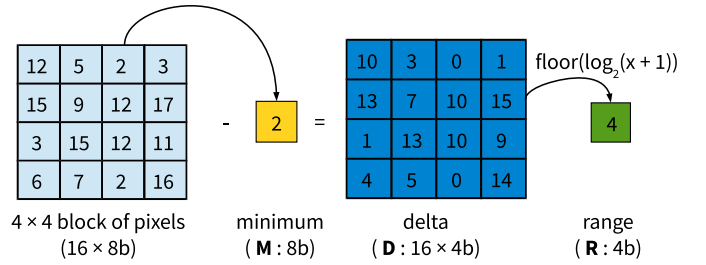


Fig. 8. Example of lossless compression of  $4 \times 4$  block.

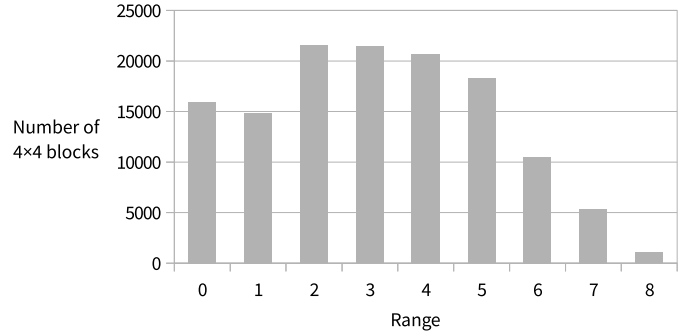


Fig. 9. Histogram of  $4 \times 4$  blocks according to their  $R$  values for one  $1920 \times 1080$  luma frame.

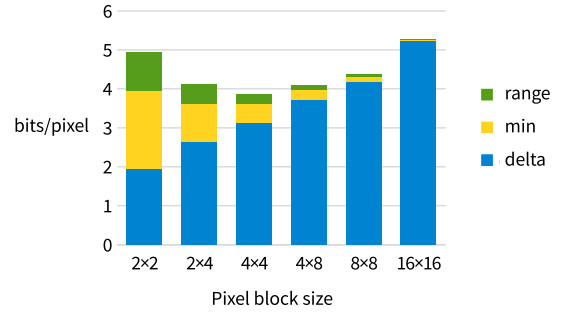


Fig. 10. Average bits/pixel achieved using various block sizes from  $2 \times 2$  to  $16 \times 16$  on one  $1920 \times 1080$  frame shows the impact of pixel block size on compression ratio.

buffer and only  $M$  and  $D$  (or original values for  $R = 8$ ) are stored in the data buffer. The data block is always byte-aligned and its size in bytes is

$$\text{byte-size}(R) = \begin{cases} 1 + 2R, & \text{for } R < 8 \\ 2R, & \text{for } R = 8 \end{cases}$$

Fig. 9 shows the distribution of  $R$  values for  $4 \times 4$  blocks in a  $1920 \times 1080$  frame. Compression is achieved, since the pixels in a  $4 \times 4$  block are typically correlated and  $R$  is around 3 to 4 (as opposed to 8 for uncompressed data).

The block size affects the compression efficiency as shown in Fig. 10. For larger block sizes, there is more variation in the pixels resulting in more bits required to represent the delta; however, the fixed overhead of the range and minimum are reduced per pixel due to amortization. For smaller block sizes, there is less variation in pixels resulting in fewer bits required to represent the delta; however, the fixed overhead



TABLE II  
COMPARISON OF LIGHTWEIGHT RFC ALGORITHM WITH  
A STATE-OF-THE-ART ALGORITHM

Compression method	minimum-delta	intra-prediction + DPCM + coding [15]
Data savings	50%	60%
Logic Area	8 kgates	80 kgates
Throughput	32 pixel/cycle	32 pixel/cycle

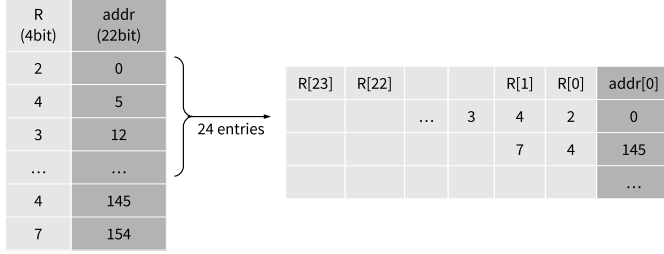


Fig. 11. Reducing the size of address buffer.

is large per pixel. A block size of  $4 \times 4$  provides the best tradeoff with the lowest bits/pixel.

The lightweight lossless RFC algorithm achieves a compression of 1.2–5 times over 384 video sequences in the HEVC common test conditions [21]. The compression depends on factors such as level of detail in the video and quantization level. Video content with small spatial variation (background sky, for example) and heavily compressed video with high quantization achieve better RFC compression. The average compression is 50%.

Implementing this algorithm in hardware takes up a total of 8 kgates of logic area for compression and decompression at a throughput of one  $4 \times 4$  block per cycle at 100 MHz. Table II shows a comparison of the lightweight RFC algorithm with a state-of-the-art algorithm used for DRAM-based video decoders [15]. We can see that the lightweight algorithm achieves a good cost–performance tradeoff. Overall, the RFC has <1% power and area overhead but saves total power by 16%.

2) *Compact Address Format for Address Buffer Size and Energy Savings:* The data buffer is addressed by a 22-bit address, so the entry in the address buffer is 26-bit (22-bit address + 4-bit  $R$ ). For 128-bit uncompressed data, this overhead is 20%, requiring five eDRAM macros for the address. To reduce the address buffer size and refresh power, the address entries are stored in a compact format as shown in Fig. 11.

Twenty four consecutive address entries are packed in a 128-bit eDRAM word by storing the starting address of the first entry and 24  $R$  values. Since the compressed data byte-size is a function of  $R$ , the starting addresses for the 2nd to 24th entries can be computed from all the  $R$  values as shown in Fig. 12. This method reduces address storage for 24 addresses from 624 bits ( $24 \times 26$ ) to 128 bits which enables the address buffer to fit in a single 0.5-MB eDRAM macro.

The loop filter is designed to output pixels in the units of  $4 \times 4$  blocks composed of 16 luma and 8 chroma blocks as

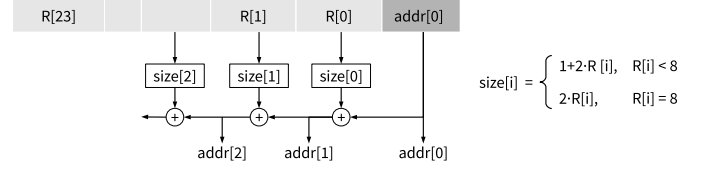


Fig. 12. Computing starting addresses of all  $4 \times 4$  blocks from compact address entry.

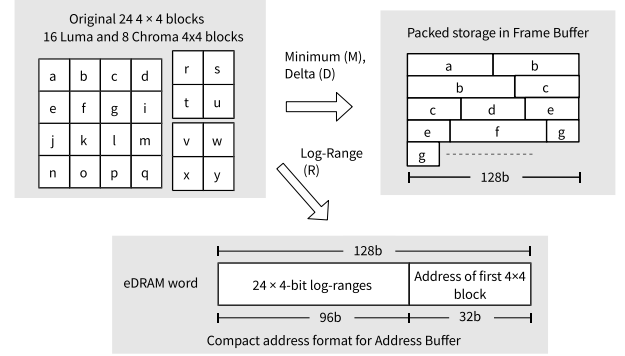


Fig. 13. Storing 24  $4 \times 4$  blocks in consecutive positions in frame buffer eDRAM and address buffer eDRAM.

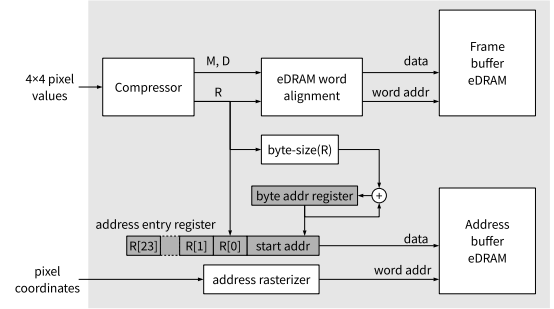


Fig. 14. Write circuit for RFC showing data compressor and bookkeeping circuits for address lookup.

shown in Fig. 13. When writing pixels to the frame buffer, the compressed data (minimum and deltas) are written to the frame eDRAM after aligning to 128-bit eDRAM word. The starting address of the first  $4 \times 4$  block in the group of 24 blocks is saved to an address entry register along with the 24  $R$  values. When all the 24 blocks are written to the frame eDRAM, the address entry register is written to the address eDRAM at an index computed from the pixel coordinates of the first  $4 \times 4$  block. This process is shown in Fig. 14. The corresponding read process is shown in Fig. 15.

### B. On-Demand eDRAM Power-Up

Due to data-dependent compression of RFC, the number of eDRAM macros needed to store a frame cannot be known *a priori*. The maximum number of macros is needed when no compression is achieved ( $R = 8$  for all  $4 \times 4$  blocks). Frames with a resolution of  $1920 \times 1080$  pixels require six macros. In the best case ( $R = 0$  for all blocks), the compressed frame needs less than one macro. In a simple scheme, a maximum

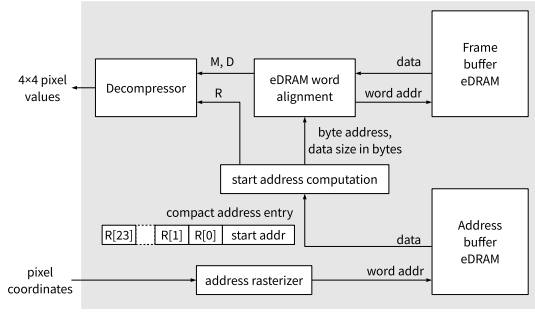


Fig. 15. Read circuit for RFC showing address lookup from compact address format, start address computation, and data decompressor.

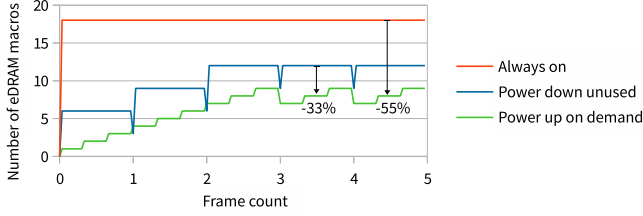


Fig. 16. Number of eDRAM macros powered up over the course of decoding the first five frames.

number of macros needed for a frame are powered up at the start of decoding a new frame. When the frame is fully decoded, we can determine how many macros were actually used and place the rest in deep power-down mode.

To further reduce the number of macros used, we propose an on-demand power-up scheme. At the start of decoding a new frame, one eDRAM macro is powered up for writing. When the storage utilization of the bank reaches a predetermined threshold, a new bank is powered up. The threshold is designed to take into account the eDRAM startup time. This on-demand scheme for powering up macros reduces eDRAM refresh power by 33% over the simple scheme and 55% over keeping all macros powered up always. Fig. 16 shows the number of eDRAM macros powered up over time for each scheme.

#### IV. REDUCING NUMBER OF ACCESSES TO eDRAM

While the eDRAM consumes less energy per access than the DRAM, it still consumes more energy than on-chip SRAM. In this section, we discuss methods to reduce access to these eDRAM macros in order to reduce the dynamic energy of the system.

##### A. Decoupling Buffer

As explained in Section II, the chip uses separate clock domains for frontend and backend processing. The frontend consists of the entropy decoder and the backend consists of four pixel decoder cores (Dec cores 1–4). The frontend and backend frequencies can be configured to balance throughputs; nominally frontend operates at four times the clock frequency as the backend. The decoder cores process four consecutive rows of CTUs in the frame as shown in Fig. 3.

To address the mismatch in the processing order of entropy decoder and the four Dec cores, a decoupling buffer is needed between them. To keep all four Dec cores running, the buffer

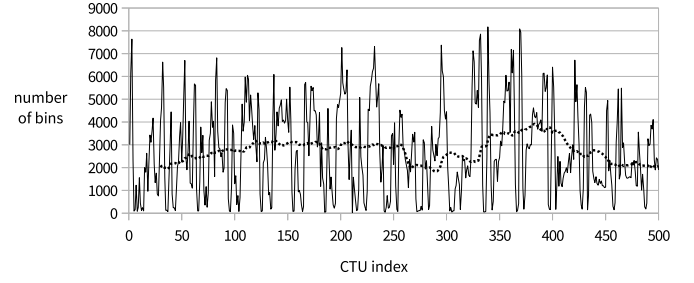


Fig. 17. Variation in workload of entropy decoder over CTUs as measured by number of binary symbols. Dotted line shows moving average over 30 CTUs.

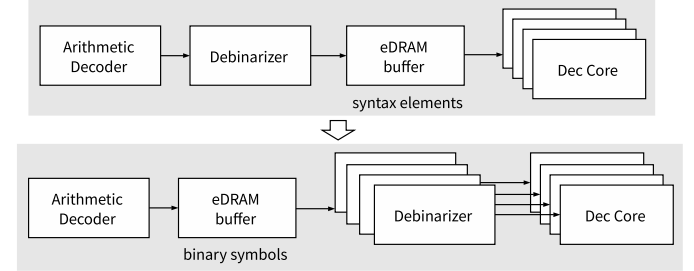
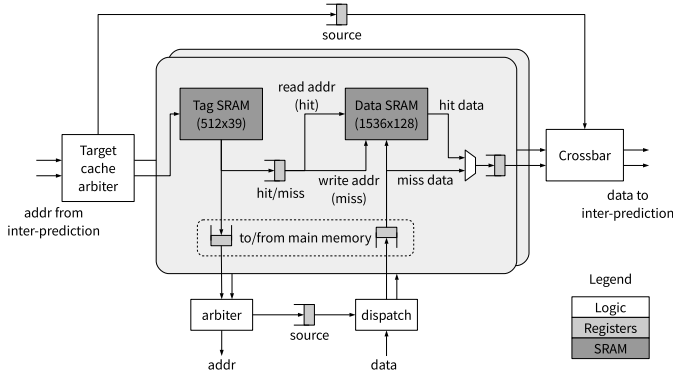


Fig. 18. Decoupling buffer between CABAC and Dec cores that stores binary symbols instead of syntax elements.

needs to store entropy decoder output of four rows of CTUs. An additional four rows of buffer is needed to allow the entropy decoder to write its output.

The decoupling buffer is useful even when multiple cores are not used, i.e., with one entropy decoder and one Dec core. The bit-level throughput of entropy decoder varies widely due to varying levels of quantization of transform coefficients. Comparatively, the Dec cores have a more regular pixel throughput. With a decoupling buffer capable of storing multiple CTUs of entropy decoder output, the throughput variation can be averaged out. Fig. 17 shows the variation in the workload of the entropy decoder as measured by the number of binary symbols (bins) per CTU for one intra-frame in a  $1920 \times 1080$  video sequence (ParkScene encoded at  $QP = 27$ ). The workload varies considerably from as low as 30 bins per CTU to as high as 8000 bins per CTU. The dotted line in Fig. 17 shows the workload when averaged over one row of CTUs (30 CTUs) in a moving average. We observe that the variation is much more tolerable (1800–4000 bins per CTU).

If syntax elements are stored in the decoupling buffer, most of the buffer space is taken up by transform coefficients (16 bit/pixel) as the other syntax elements, such as MVs and intra-modes, are signaled at PU granularity. The buffer size for storing eight rows of  $64 \times 64$  CTUs in a  $1920 \times 1080$  frame is 3 MB, or 6 eDRAM macros. To reduce this size, the entropy decoder is split into CABAC that outputs binary symbols (0s and 1s) and a Debinarizer that parses the stream of binary symbols for syntax elements as shown in Fig. 18. The binary symbols are very compact representations of the syntax elements using a variety of lossless coding techniques (e.g., unary, truncated unary, and  $k$ th-order ExpGolomb [22]). The Debinarizer is moved into each Dec core so that the decoupling buffer can store the binary symbols instead of syntax elements.

Fig. 19. Architecture of 2 $\times$  parallel cache.

This reduces bandwidth to the decoupling buffer by 66 times and its power by 4 times. Debinarizers are needed in each Dec core to decode the bins to syntax elements, which add an overhead of 1-mW power (4% of chip power), 102 kgates of logic area (10% of chip logic area), and 16 kB of SRAM (10% of chip SRAM bits).

### B. Frame Buffer

To reduce access to the frame buffer, this paper uses a common cache for all inter-prediction requests coming from all four decoder cores. Fig. 19 shows the architecture of the cache. The cache is placed in the memory-clock domain which runs at four times the frequency of the decoder cores in the backend-clock domain to ensure sufficient cache throughput.

Two parallel caches are used for increased throughput. The caches store mutually exclusive regions in the main memory such that adjacent 4 $\times$  4 pixel blocks always go to separate caches. Two 4 $\times$  4 addresses are requested by inter-prediction per cycle. If their target caches are different, the target cache arbiter dispatches both addresses to their respective caches in parallel. If the target caches are the same, the addresses are dispatched sequentially.

The tag file in each cache is made of a 20-kb SRAM along with a hit/miss determination logic. The tag file is three-way set associative with an FIFO update method. The cache line in the data SRAM is 128 b corresponding to a 4 $\times$  4 pixel block. A total of 3072 cache lines are stored in the two parallel caches.

Bypass FIFOs are used to reduce the hit latency, which ranges from 3 cycles to a maximum of 14 cycles with the eDRAM memory. However, the inter-prediction modules see at most a four-cycle latency on their read requests, as they run at four times lower clock frequency. Overall, this three-way set-associative, 2 times parallel cache reduces the frame buffer bandwidth by 2.1 times while using 12.7 kgates of logic and 54 kB of SRAM.

## V. TEST CHIP RESULTS

The test chip shown in Fig. 20 was fabricated in 40-nm CMOS process [9]. The chip was tested using four 1920 $\times$  1080 video sequences (Kimono, ParkScene, Cactus, and BQTerrace) that were encoded with two reference frames

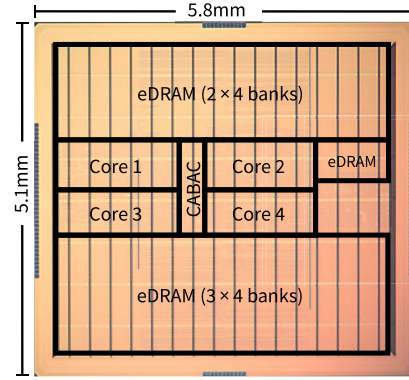


Fig. 20. Die micrograph of HEVC decoder test chip.

TABLE III  
SUMMARY OF CHIP SPECIFICATIONS

Technology	TSMC 40nm LP
Supply voltage	Core: 0.8 - 1.1V, eDRAM: 1.1V, IO: 2.5V
Video standard	H.265/HEVC (Main profile with 2 reference frames)
Chip size	5.8mm $\times$ 5.1mm
Core size	5.1mm $\times$ 4.3mm
Gate count	1122 kgates (NAND2 logic area only)
On-chip SRAM	162.75 kB
On-chip eDRAM	21 $\times$ 0.5 MB
Max resolution	1920 $\times$ 1080
Max throughput	47.9 MPixel/s
Power at 1.1V	30.6 mW (I frame) and 24.9 mW (B frame)
Min throughput	18.9 MPixel/s
Power at 0.8V	9.5 mW

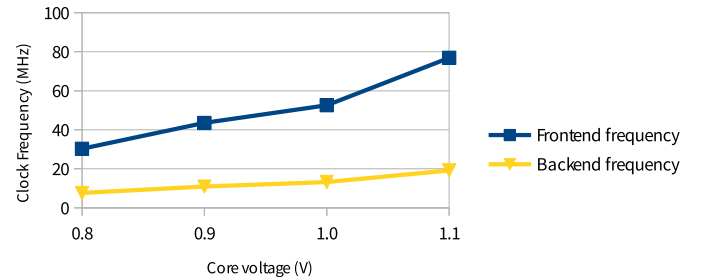


Fig. 21. Measured voltage-frequency performance plot for test chip.

across various quantization levels (22, 27, 32, and 37). The test chip can operate from 0.8 to 1.1 V (eDRAM fixed at 1.1 V) with the frontend/memory-clock domain frequency ranging from 30.3 to 76.9 MHz and a backend domain frequency ranging from 7.6 to 19.2 MHz.

At maximum frequency, the chip can decode 1920 $\times$  1080 video at 24–50 frames/s depending on encoding parameters. The chip consumes 30.6 mW (0.35 nJ/pixel) for I frames (only intra-prediction is used), and 24.9 mW (0.77 nJ/pixel) for B frames (both intra-prediction and inter-prediction is used). Table III summarizes the chip specifications.

Fig. 21 shows the voltage-frequency plot for the test chip. The frontend frequency is kept at four times the backend frequency at all voltages. On the lowest voltage setting (0.8 V), the frequency is sufficient to decode 640 $\times$  480 at 60 frames/s while consuming only 9.5 mW of power.

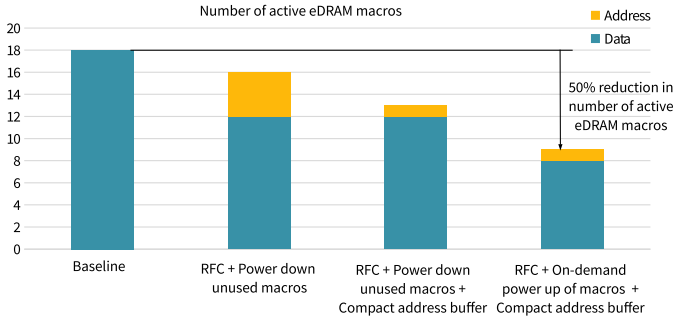


Fig. 22. Reduction in the number of active eDRAM macros through the use of RFC, compact address buffer format, and on-demand power-up of macros.

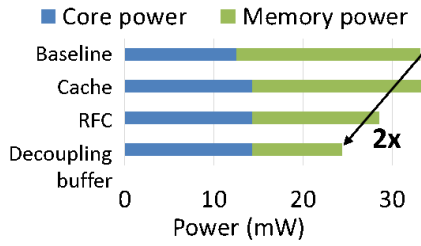


Fig. 23. Reduction in total chip power using proposed techniques. Two times savings in eDRAM power is achieved.

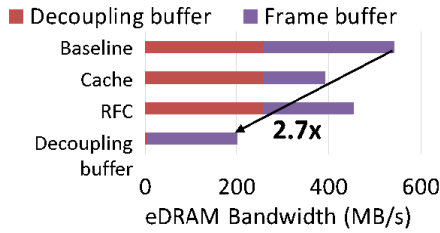


Fig. 24. eDRAM bandwidth is reduced by 2.7 times using a combination of RFC, cache, and partial compression of syntax elements.

Figs. 22 and 23 show the reduction in the number of active eDRAM macros and resulting power reduction achieved using RFC, compact address buffer format, and on-demand power-up of macros. Together, the number of powered down macros reduces refresh power by 50%.

Fig. 24 shows the reduction in eDRAM bandwidth due to caching for the reference frame and using partial compression format. RFC leads to a slight increase in eDRAM bandwidth due to the use of the address buffer. The overall eDRAM bandwidth is reduced by 2.7 times.

#### A. Energy and Area Breakdown

Fig. 25 shows the breakdown of energy consumed for decoding  $1920 \times 1080$  frames. The power for eDRAM macros in the frame buffer accounts for only 33% of the total power after applying the eDRAM optimizations in Sections III and IV. The portion marked memory controller in the breakdown contains the cache, RFC circuits, and memory access arbiters.

Fig. 26 shows a breakdown of the 162.75 kB of on-chip SRAM used by the chip. Memory controller consists of arbiter

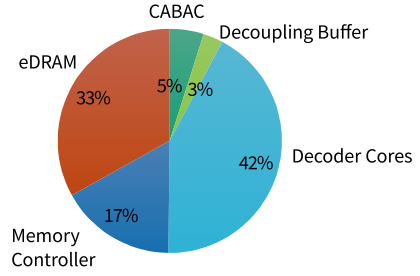


Fig. 25. Energy breakdown of chip for decoding  $1920 \times 1080$  frames.

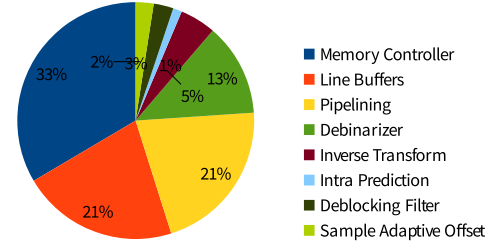


Fig. 26. SRAM bits breakdown of chip.

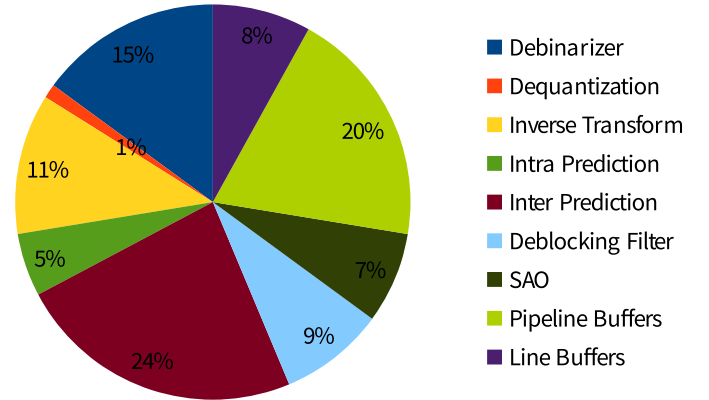


Fig. 27. Energy breakdown for individual decoder core.

for writing back the decoded pixels to the frame buffer and a cache which uses SRAM for both data and tags. Pipelining consists of buffers between: 1) debinarizer and prediction; 2) inverse transform and prediction; and 3) prediction and deblocking filter. Across all four decoders, by using finer grained pipelining, the pipeline buffer accounts for 273 kb (68 kb per decoder), which is 6.4 times smaller than the pipeline buffer in [7].

Finally, each decoder core require 235 kgates and 18.3 kB of on-chip SRAM. Figs. 27 and 28 show the energy and area breakdown for an individual decoder core. The energy breakdown was obtained from a postsynthesis analysis using 5 ms of switching activity. The decoder cores were designed for a minimum throughput of 2 pixel/cycle and synthesized at 25 MHz. More details on the design of the individual blocks in the decoder are in [23].

#### B. Comparison With State of the Art

Fig. 29 compares this chip against state-of-the-art H.265/HEVC video decoders [5], [6], [24]–[26]. All the other designs target higher resolutions for applications with  $>50$ -mW power budgets. The use of eDRAM, along with the



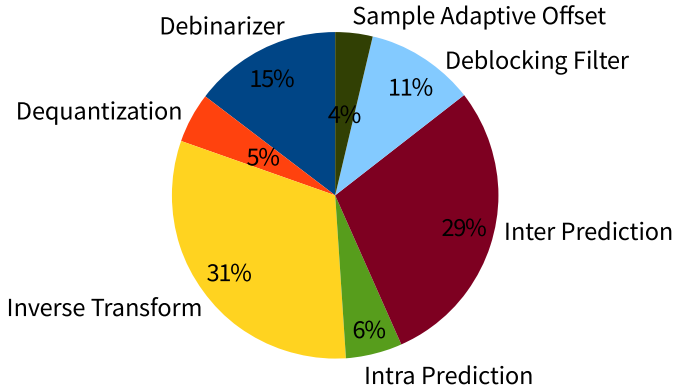


Fig. 28. Logic area breakdown of pixel decoder core.

	This Work	ISSCC 2013	A-SSCC 2013	ESSCIRC 2014	ISSCC 2016	ISSCC 2012
Standard	H.265/HEVC	H.265/HEVC WD4	H.265/HEVC	H.265/HEVC, multistandard	H.265/HEVC	H.264/AVC MP/MVC
Gate Count	1122K	715K	446K	3454K	2887K	1338K
SRAM	162.75kB	124kB	10.2kB	154kB	396kB	79.9kB
Technology	40nm/1.1V	40nm/0.9V	90nm/1V	28nm/0.9V	40nm/1V	65nm/1.2V
Max Throughput	1920x1080 @24fps	3840x2160 @30fps	1920x1080 @35fps	3840x2160 @60fps	7640x4320 @120fps	7640x4320 @60fps
Frame buffer Storage	128b eDRAM	32b DDR3	n/a	32b LPDDR3	64b DDR3	64b DDR3
Core Power [mW]	21.2 [I] 14.6 [B]	76	36.9	104	690	410
Frame buffer Power [mW]	9.4 [I] 10.3 [B]	219	n/a	n/a	n/a	2520
Core energy [nJ/pixel]	0.25 [I] 0.45 [B]	0.31	0.59	0.20	0.15 - 0.25	0.21
Frame buffer energy [nJ/pixel]	0.11 [I] 0.32 [B]	0.88	n/a	n/a	n/a	1.27
System energy [nJ/pixel]	0.35 [I] 0.77 [B]	1.19	n/a	n/a	n/a	1.48

Fig. 29. Comparison with the state of the art [5], [6], [24]–[26].

TABLE IV

COMPARISON WITH PREVIOUS WORK AT 1920 × 1080 24 frames/s.

\*—POWER FOR PREVIOUS WORK IS ESTIMATED BY SCALING THE FREQUENCY

	This work	Previous Work [7, 24]
Standard	H.265/HEVC	H.265/HEVC Working Draft 4
Logic gate count	1122 kgates	715 kgates
SRAM	162.75 kB	124 kB
Frame buffer	128b eDRAM	32b DDR3
Technology	40nm LP	40nm GP
Core voltage	1.1 V	0.9 V
Frequency	80 MHz/20 MHz	40 MHz
Core power	14.6 mW	36 mW *
Frame buffer power	10.3 mW	150 mW *
System power	24.9 mW	186 mW *

reduced refresh power and memory access, allows this paper to meet the stringent power budgets for wearable devices. We also compare this chip with our previous work [7] scaled to 1920 × 1080 resolution in Table IV. The frequency of the previous chip is scaled down to 40 MHz to match the throughput of this chip. Voltage and technology scaling is not applied. For energy comparison, the leakage power of the core and the background power of DRAM are kept constant and only the active components of both are scaled down for the reduced throughput. This paper has six times lower energy/pixel than the previous work.

This paper uses four parallel decoder cores as compared with the single decoder core of the previous work. This enables

energy reduction using voltage and frequency scaling at the cost of increased gate count and the SRAM. The current test chip only uses frequency scaling; future work can use voltage scaling to achieve further reduction in energy/pixel. We note that the current chip has a lower gate count and an SRAM per decoder core. The SRAM reduction is achieved by efficient pipelining as explained in Section II-A. The gate count reduction is due to differences in the video coding standard (Working Draft 4 versus Final) and register transfer logic-level improvements such as replacing large register arrays with the SRAM.

## VI. CONCLUSION

In this paper, we demonstrated several techniques that reduce the energy consumption of data movement to improve the energy efficiency of a fully integrated H.265/HEVC video decoder targeted at wearable devices. The eDRAM is used to reduce the cost of off-chip memory access and overall system footprint. We then reduce the energy of the eDRAM itself by addressing both its refresh power and memory access. The refresh power is reduced by 50% by reducing the number of eDRAM macros that are enabled at a given time. The number of accesses to the eDRAM is reduced by 2.7 times by using a compressed format to store data in the frame, address, and decoupling buffers. Together, these techniques reduce the overall power consumption of eDRAM by two times, which enable the entire H.265/HEVC video decoder system to consume between 24.9 and 30.6 mW for real-time high-definition decoding, which is well within the 50-mW power budget for wearable devices.

## ACKNOWLEDGMENT

The authors would like to thank the TSMC University Shuttle Program for chip fabrication. They would also like to thank A. Paidimarri, M. Price, and P. Raina for helpful discussion on design and CAD tools.

## REFERENCES

- [1] M. Aleksić, “Deep learning for mobile and embedded devices,” in *Proc. Symp. VLSI Circuits Short Course 1 Mach. Learn. Circuit Designers*, 2017.
- [2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the high efficiency video coding (HEVC) standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
- [3] V. Sze, M. Budagavi, and G. J. Sullivan, Eds., *High Efficiency Video Coding (HEVC): Algorithms and Architectures* (Integrated Circuits and Systems). Cham, Switzerland: Springer, 2014, pp. 1–375.
- [4] V. Sze, D. F. Finchelstein, M. E. Sinangil, and A. P. Chandrakasan, “A 0.7-V 1.8-mW H.264/AVC 720p video decoder,” *IEEE J. Solid-State Circuits*, vol. 44, no. 11, pp. 2943–2956, Nov. 2009.
- [5] C.-H. Tsai, H.-T. Wang, C.-L. Liu, Y. Li, and C.-Y. Lee, “A 446.6 K-gates 0.55–1.2 V H. 265/HEVC decoder for next generation video applications,” in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2013, pp. 305–308.
- [6] C.-C. Ju *et al.*, “A 0.2 nJ/pixel 4K 60fps main-10 HEVC decoder with multi-format capabilities for UHD-TV applications,” in *Proc. 40th Eur. Solid State Circuits Conf. (ESSCIRC)*, Sep. 2014, pp. 195–198.
- [7] M. Tikekar, C.-T. Huang, C. Juvekar, V. Sze, and A. P. Chandrakasan, “A 249-Mpixel/s HEVC video-decoder chip for 4K ultra-HD applications,” *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 61–72, Jan. 2014.
- [8] D. Zhou *et al.*, “An 8K H.265/HEVC video decoder chip with a new system pipeline design,” *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 113–126, Jan. 2017.

- [9] M. Tikekar, V. Sze, and A. Chandrakasan, "A fully-integrated energy-efficient H.265/HEVC decoder with eDRAM for wearable devices," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Jun. 2017, pp. 230–231.
- [10] S. Ghosh, "Modeling of retention time for high-speed embedded dynamic random access memories," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 9, pp. 2596–2604, Sep. 2014.
- [11] Y. H. Chen and V. Sze, "A deeply pipelined CABAC decoder for HEVC supporting level 6.2 high-tier applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 5, pp. 856–868, May 2015.
- [12] M. Tikekar, C.-T. Huang, V. Sze, and A. Chandrakasan, "Energy and area-efficient hardware implementation of HEVC inverse transform and dequantization," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 2100–2104.
- [13] D. F. Finchelstein, V. Sze, and A. P. Chandrakasan, "Multicore processing and efficient on-chip caching for H.264 and future video decoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 11, pp. 1704–1713, Nov. 2009.
- [14] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Cambridge, 2014, pp. 609–622.
- [15] L. Guo, D. Zhou, and S. Goto, "A new reference frame recompression algorithm and its VLSI architecture for UHDTV video codec," *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2323–2332, Dec. 2014.
- [16] D. Zhou, L. Guo, J. Zhou, and S. Goto, "Reducing power consumption of HEVC codec with lossless reference frame recompression," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2014, pp. 2120–2124.
- [17] M. Budagavi and M. Zhou, "Video coding using compressed reference frames," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2008, pp. 1165–1168.
- [18] A. R. Alameldeen and D. A. Wood, "Adaptive cache compression for high-performance processors," in *Proc. 31st Annu. Int. Symp. Comput. Archit.*, Jun. 2004, pp. 212–223.
- [19] S. Sardashti, A. Seznec, and D. A. Wood, "Skewed compressed caches," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2014, pp. 331–342.
- [20] G. Pekhimenko, V. Seshadri, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Base-delta-immediate compression: Practical data compression for on-chip caches," in *Proc. 21st Int. Conf. Parallel Archit. Compilation Techn. (PACT)*, 2012, pp. 377–388.
- [21] F. Bossen, *Common Test Conditions and Software Reference Configurations*, document JCTVC-H1100, Feb. 2012.
- [22] V. Sze and M. Budagavi, "High throughput CABAC entropy coding in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1778–1791, Dec. 2012.
- [23] M. Tikekar, "Energy-efficient video decoding using data statistics," Ph.D. thesis, Dept. Elect. Eng. Comput. Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2017.
- [24] C.-T. Huang, M. Tikekar, C. Juvekar, V. Sze, and A. Chandrakasan, "A 249Mpixel/s HEVC video-decoder chip for quad full HD applications," in *ISSCC Dig. Tech. Papers*, Feb. 2013, pp. 162–163.
- [25] D. Zhou *et al.*, "A 4Gpixel/s 8/10b H. 265/HEVC video decoder chip for 8K Ultra HD applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 266–268.
- [26] D. Zhou, J. Zhou, J. Zhu, P. Liu, and S. Goto, "A 2Gpixel/s H.264/AVC HP/MVC video decoder chip for super hi-vision and 3DTV/FTV applications," in *ISSCC Dig. Tech. Papers*, Feb. 2012, pp. 224–226.



**Mehul Tikekar** (S'11–M'18) received the B.Tech. degree in electrical engineering from IIT Bombay, Mumbai, India, in 2010, and the S.M. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2012 and 2017, respectively.

Since 2017, he has been with Analog Devices, Boston, MA, USA, where he is currently a Research Scientist and an Engineer. His current research interests include circuits, architectures, and algorithms for video coding, signal processing, and

machine learning.

Dr. Tikekar was a recipient of the Massachusetts Institute of Technology Presidential Fellowship in 2011.



**Vivienne Sze** (S'04–M'10–SM'16) received the B.A.Sc. (Hons) degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. and Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2006 and 2010 respectively. In 2011, she received the Jin-Au Kong Outstanding Doctoral Thesis Prize in Electrical Engineering at MIT.

She is an Associate Professor at MIT in the Electrical Engineering and Computer Science Department.

Her research interests include energy-aware signal processing algorithms, and low-power circuit and system design for portable multimedia applications including computer vision, deep learning, autonomous navigation and video processing/coding. Prior to joining MIT, she was a Member of Technical Staff in the Systems and Applications R&D Center at Texas Instruments (TI), Dallas, TX, where she designed low-power algorithms and architectures for video coding. She also represented TI in the JCT-VC committee of ITU-T and ISO/IEC standards body during the development of High Efficiency Video Coding (HEVC), which received a Primetime Emmy Engineering Award. Within the committee, she was the primary coordinator of the core experiment on coefficient scanning and coding, and has chaired/vice-chaired several ad hoc groups on entropy coding. She is a co-editor of "*High Efficiency Video Coding (HEVC): Algorithms and Architectures*" (Springer, 2014).

Prof. Sze is a recipient of the 2017 Qualcomm Faculty Award, 2016 Google Faculty Research Award, 2016 AFOSR Young Investigator Research Program (YIP) Award, 2016 3M Non-Tenured Faculty Award, 2014 DARPA Young Faculty Award, 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2017 CICC Outstanding Invited Paper Award, the 2016 IEEE Micro Top Picks Award and the 2008 A-SSCC Outstanding Design Award. Prof. Sze is a Distinguished Lecturer of the IEEE Solid-State Circuits Society (SSCS), and currently serves on SSCS AdCom and the technical program committees for VLSI Symposium and MICRO.



**Anantha P. Chandrakasan** (F'04) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 1989, 1990, and 1994, respectively.

Since 1994, he has been with MIT, Cambridge, MA, USA, where he is currently the Vannevar Bush Professor of electrical engineering and computer science. He was the Director of the Microsystems Technology Laboratories, Massachusetts Institute of Technology (MIT), from 2006 to 2011. From 2011 to 2017, he was the Head of the Department of Electrical Engineering and Computer Science, MIT, where he has been the Dean of the School of Engineering since 2017. He has co-authored the book *Low Power Digital CMOS Design* (Kluwer Academic Publishers, 1995), *Digital Integrated Circuits* (Pearson Prentice-Hall, 2003, second edition), and *Sub-Threshold Design for Ultra-Low Power Systems* (Springer, 2006). His current research interests include ultralow-power circuit and system design, energy harvesting, energy-efficient RF circuits, and hardware security.

Dr. Chandrakasan was a co-recipient of several awards, including the 2007 International Solid-State Circuits Conference (ISSCC) Beatrice Winner Award for Editorial Excellence and the ISSCC Jack Kilby Award for Outstanding Student Paper in 2007, 2008, and 2009. He received the 2009 Semiconductor Industry Association University Researcher Award and the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits. He has served in various roles for the IEEE ISSCC, including the Program Chair, the Signal Processing Sub-Committee Chair, and the Technology Directions Sub-Committee Chair. He served as the Conference Chair of ISSCC from 2010 till 2018. In 2015, he was elected to the National Academy of Engineering.