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

Mehul Tikekar, Student Member, IEEE, Vivienne Sze, Senior Member, IEEE, and Anantha Chandrakasan, Fellow, IEEE

Abstract—This paper proposes a fully-integrated H.265/HEVC video decoder that supports real-time video playback within the 50mW power budget of wearable devices such as smart watches and virtual reality (VR) headsets. Specifically, this work focuses on reducing data movement to and from off-chip memory as it dominates energy consumption, consuming 2.8 to 6× more energy than processing in most video decoders. Embedded DRAM (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 2× fewer eDRAM macros, reducing refresh power by 33%; 2) eDRAM macros are powered up on-demand to further reduce refresh power by 33%; 3) syntax elements are distributed to four decoder cores in a partially compressed form to reduce decoupling buffer power by 4×. These approaches reduce eDRAM power by 2× 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 24.9–30.6mW for decoding 1920×1080 video at 24–50 fps.

Index Terms—H.265/HEVC, video coding, embedded DRAM, reference frame compression, 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/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 DRAM. Some of the more recent work [7, 8] has demonstrated techniques such as caching, reference frame compression and DRAM-aware memory mapping to reduce off-chip memory power. However, even with these techniques, memory accesses require 2.8× to 6× 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 2× energy saving in eDRAM itself and 6× energy savings compared to a decoder system using off-chip DDR3 memory [9]. While eDRAM reduces system power and physical footprint relative to traditional DRAM, it requires more frequent refresh due to two factors. Firstly, DRAM uses larger bit-cell capacitance and access transistors with lower leakage than eDRAM. Hence, eDRAM bitcells need more frequent refresh (on the order of microseconds) compared to DRAM (on the order of milliseconds) [10]. Secondly, for typical HEVC decoding workloads, tens of eDRAM macros are needed to meet the storage requirement, but just 1 to 2 macros are sufficient to meet the bandwidth requirement. All the macros need to be refreshed to retain stored data even though only 1 to 2 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 eDRAM is dominated by refresh energy. Accordingly, in this work 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 work makes the following contributions:

- reduce energy per memory by using eDRAM to eliminate off-chip memory
- reduce refresh power of eDRAM by reducing the number of enabled eDRAM macros using reference frame compression and an on-demand power up scheme
- reduce number of memory accesses by using partial compression in the decoupling buffer, and reference frame compression 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, while Section IV discusses techniques to reduce 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 clock domains: (1) Frontend and memory domain contains the CABAC entropy decoder [11] to generate the binary symbols (bins) of the syntax elements (e.g., motion vectors, prediction modes, transform coefficients) from the compressed bitstream, and frame buffer plus associated logic to store the reference frames for motion compensation. (2) Backend domain contains the decoder cores that generate the pixels from bins of the syntax elements. To balance the throughput, the frontend operates at a higher clock frequency than the backend as the
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 work, four equally size FIFOs with a maximum depth of 8 is 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 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\cite{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 embedded DRAM (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:

- **Frame buffer**: 18 eDRAM macros to store the two previous and one current frame (6 macros per frame).
- **Address buffer**: One eDRAM macro to store the look up table to translate pixel location in frames to byte address in the frame buffer. This is necessary to support lossless reference frame compression as discussed in Section III-A.
- **Co-located motion vector (MV) buffer**: One eDRAM macro to store the co-located motion vectors from previous frames to support motion vector parsing for the current frame.
- **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 compare to DRAM. Section III will discuss techniques to address the eDRAM refresh power in order to reduce the refresh power of the system. While eDRAM consumes less energy per access compared to 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, eDRAM stands between SRAM and DRAM in terms of energy/access and density. For example, in 28 nm technology, eDRAM has $321 \times$ lower energy/access than DRAM and $2.85 \times$ higher density than SRAM\cite{14}. The key challenge with 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 to DRAM process, refresh requests need to be made more frequently and so, refresh power is more significant on eDRAM than DRAM. SRAM does not need to be refreshed as the data is actively maintained by cross-coupled inverters. Further, for video decoding application, the instantaneous read-write bandwidth requirement can be satisfied by 2 to 3 eDRAM macros in the frame buffer, while 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\cite{1}). Accordingly, we focus on reducing refresh power as a means to reduce energy consumption of eDRAM. 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 is 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

Reference frame compression (RFC) is a popular technique used on video decoders with DRAM main memory to reduce off-chip bandwidth\cite{15, 17}. Data compression for memory savings has also been explored for processor caches\cite{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, RFC must use lossless compression which makes the compressed data variable length. In previous work\cite{15, 16}, the compressed data is 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 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 is 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 work, 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

\footnote{The refresh power reductions techniques proposed in this work becomes 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%.}
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.

(a) Lightweight RFC Algorithm: A lightweight compression technique is applied on 4×4 pixel blocks. Each 4×4 block is compressed to three elements:

1. \( M \): minimum of the 16 pixel values [8-bit: 0 to 255]
2. \( R \): (log-range) number of bits required to represent the delta above the minimum [4-bit: 0 to 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

\[
2 = \text{floor}(\log(x + 1))
\]

of the compressed data, is stored separately in the address 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 4×4 blocks in a 1920×1080 frame. Compression is achieved since the pixels in a 4×4 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 blocks 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 is large per pixel. A block size of 4×4 provides the best tradeoff with the lowest bits/pixel.

The lightweight lossless RFC algorithm achieves a compression of 1.2×5× 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 k gates of logic area for compression and decompression.

![Diagram](image-url)
at a throughput of one 4×4 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, RFC has < 1% power and area overhead but saves total power by 16%.

![Table II](image)

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

<table>
<thead>
<tr>
<th>Compression method</th>
<th>minimum-delta</th>
<th>intra-prediction + DPCM + coding [15]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data savings</td>
<td>50%</td>
<td>60%</td>
</tr>
<tr>
<td>Logic Area</td>
<td>8 kgates</td>
<td>80 kgates</td>
</tr>
<tr>
<td>Throughput</td>
<td>32 pixel/cycle</td>
<td>32 pixel/cycle</td>
</tr>
</tbody>
</table>

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 5 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. 24 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 second 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 × 26) to 128 bits which enables the address buffer to fit in a single 0.5MB eDRAM macro.

The loop filter is designed to output pixels in units of 24 4×4 blocks composed of 16 luma and 8 chroma blocks as shown in Fig. 13. When writing pixels to the frame buffer, the compressed data (minimum and deltas) is written to the frame eDRAM after aligning to 128-bit eDRAM word. The starting address of the first 4×4 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 4×4 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. Maximum number of macros is needed when no compression is achieved (R = 8 for all 4×4 blocks). Frames with a resolution of 1920×1080 pixels require 6 macros. In the best case (R = 0 for all blocks), the compressed frame needs less than one macro. In a simple scheme, the maximum 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 macros 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 eDRAM consumes less energy per access than 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 4 pixel decoder cores (Dec core 1-4). The frontend and backend frequencies can be configured to balance throughputs; nominally frontend operates at 4 times the clock frequency as the backend. The decoder cores process 4 consecutive rows of CTUs in the frame as shown in Fig. 3.

To address the mismatch in processing order of entropy decoder and the 4 Dec cores, a decoupling buffer is needed between them. To keep all 4 Dec cores running, the buffer needs to store entropy decoder output of 4 rows of CTUs. An additional 4 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 1 entropy decoder and 1 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 entropy decoder as measured by number of binary symbols (bins) per CTU for one intra frame in a 1920x1080 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. 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.
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 20kbit SRAM along with a hit/miss determination logic. The tag-file is 3-way set associative with FIFO update method. The cache line in the data SRAM is 128b 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 eDRAM memory. However, the inter-prediction modules see at most a 4 cycle latency on their read requests, as they run at $4 \times$ lower clock frequency. Overall, this 3-way set-associative, $2 \times$ parallel cache reduces the frame buffer bandwidth by $2.1 \times$ while using 12.7 k gates of logic and 54 kB of SRAM.

V. TEST CHIP RESULTS

The test chip shown in Fig. 20 was fabricated in 40nm CMOS process [9]. The chip was tested using four 1920$\times$1080 video sequences (Kimono, ParkScene, Cactus, and BQTerrace) that were encoded with 2 reference frames across various quantization levels (22, 27, 32 and 37). The test chip can operate from 0.8V to 1.1V (eDRAM fixed at 1.1V) with the frontend/memory clock domain frequency ranging from 30.3MHz to 76.9MHz and a backend domain frequency ranging from 7.6MHz to 19.2MHz.

At maximum frequency, the chip can decode 1920$\times$1080 video at 24 to 50 frames per second depending on encoding parameters. The chip consumes 30.6mW (0.35nJ/pixel) for I frames (only intra-prediction is used), and 24.9mW (0.77nJ/pixel) for B frames (both intra 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 $4 \times$ 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 fps while consuming only 9.5 mW of power.

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

Fig. 24 shows the reduction in eDRAM bandwidth due to caching for the reference frame and using partial compression format. Reference frame compression leads to a slight increase in eDRAM bandwidth due to the use of the address buffer. The overall eDRAM bandwidth is reduced by $2.7 \times$.
TABLE III
SUMMARY OF CHIP SPECIFICATIONS

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

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

A. Energy and Area Breakdown

Fig. 25 shows breakdown of energy consumed for decoding 1920×1080 frames. The power for eDRAM macros in the frame buffer accounts for only 33% of the total power after applying the eDRAM optimizations in Section 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 arbiters 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; (3) prediction and deblocking filter. Across all four decoders, by using finer grained pipelining, the pipeline buffer accounts for 273kbits (68kbits per decoder), which is 6.4× smaller than the pipeline buffer in [7].

Finally, each decoder core requires 235 k gates and 18.3kB of on-chip SRAM. Fig. 27 and Fig. 28 shows the energy and area breakdown for an individual decoder core. The energy breakdown was obtained from post-synthesis analysis using 5 ms of switching activity. The decoder cores were designed for a minimum throughput of 2 pixel/cycle and synthesized at 25MHz. 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 > 50mW power budgets. The use of eDRAM, along with the reduced refresh power and memory access, allows this work 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, leakage power of the core and background power of DRAM are kept constant and only the active components of
VI. CONCLUSIONS

In this work, 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. 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 reduce by 2.7× 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 2×, which enable the entire H.265/HEVC video decoder system to consume between 24.9 mW to 30.6 mW for real-time high definition decoding, which is well within the 50 mW power budget for wearable devices.

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/co-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 AFRF Non-Tenured Faculty Award, 2014 DARPA Young Faculty Award, 2007 DAC/JSSC Student Design Contest Award and a recipient of the 2017 CICC Outstanding Invited Paper Award, the 2016 IEEE Micro Top Picks Award and the 2008 A-JSSC 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.

Mehul Tikekar (S’11–M’18) received the B.Tech degree in electrical engineering from the Indian Institute of Technology Bombay, Mumbai, India, in 2010, and the S.M. and Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, in 2012 and 2017 respectively. His research interests include circuits, architectures and algorithms for video coding, signal processing and machine learning. Since August 2017, he has been with Analog Devices, Boston, MA where he is a Research Scientist and Engineer.

Table IV

<table>
<thead>
<tr>
<th>Storage</th>
<th>This Work</th>
<th>Previous Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic gates</td>
<td>122 k gates</td>
<td>715 k gates</td>
</tr>
<tr>
<td>SRAM</td>
<td>162.75 kB</td>
<td>124 kB</td>
</tr>
<tr>
<td>Technology</td>
<td>40 mm/1.1 V</td>
<td>40 nm/0.9 V</td>
</tr>
<tr>
<td>Max throughput</td>
<td>3840x2160 @30 fps</td>
<td>3840x2160 @60 fps</td>
</tr>
<tr>
<td>Frame buffer</td>
<td>32b eDRAM</td>
<td>32b DDR3</td>
</tr>
<tr>
<td>Core power [mW]</td>
<td>12.2 [I]</td>
<td>14.6 [I]</td>
</tr>
<tr>
<td>Frame buffer power [mW]</td>
<td>9.4 [I]</td>
<td>10.3 [I]</td>
</tr>
<tr>
<td>Core energy [nJ/pixel]</td>
<td>0.25 [I]</td>
<td>0.45 [I]</td>
</tr>
<tr>
<td>Frame buffer energy [nJ/pixel]</td>
<td>0.11 [I]</td>
<td>0.32 [I]</td>
</tr>
<tr>
<td>System energy [nJ/pixel]</td>
<td>0.35 [I]</td>
<td>0.77 [I]</td>
</tr>
</tbody>
</table>

* Power for previous work is estimated by scaling the frequency.
Anantha Chandrakasan (F04) received the B.S., M.S., and Ph.D. degrees from the University of California at Berkeley, Berkeley, CA, USA, in 1989, 1990, and 1994, respectively, all in electrical engineering and computer sciences.

He was the Director of Microsystems Technology Laboratories, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, from 2006 to 2011, and the Head of the Electrical Engineering and Computer Science Department, MIT, from 2011 to 2017. Since 1994, he has been with MIT, where he is currently the Dean of Engineering. His current research interests include micropower digital and mixed-signal integrated circuit design, wireless microsensor system design, portable multimedia devices, energy efficient radios, and emerging technologies.

Dr. Chandrakasan received the 2009 Semiconductor Industry Association University Researcher Award and the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits. He was the ISSCC Conference Chair from 2010 to 2018.

REFERENCES


