Multicore Processing and Efficient On-Chip Caching for H.264 and Future Video Decoders

Daniel F. Finchelstein, *Member, IEEE,* Vivienne Sze, *Student Member, IEEE,* and Anantha P. Chandrakasan, *Fellow, IEEE*

***Abstract*—Performance requirements for video decoding will continue to rise in the future due to the adoption of higher resolutions and faster frame rates. Multicore processing is an effective way to handle the resulting increase in computation. For power-constrained applications such as mobile devices, extra performance can be traded-off for lower power consumption via voltage scaling. As memory power is a significant part of system power, it is also important to reduce unnecessary on-chip and off-chip memory accesses. This paper proposes several techniques that enable multiple parallel decoders to process a single video sequence; the paper also demonstrates several on-chip caching schemes. First, we describe techniques that can be applied to the existing H.264 standard, such as multiframe processing. Second, with an eye toward future video standards, we propose replacing the traditional raster-scan processing with an interleaved macroblock ordering; this can increase parallelism with minimal impact on coding efficiency and latency. The proposed architectures allow N parallel hardware decoders to achieve a speedup of up to a factor of N. For example, if $N = 3$, the proposed multiple frame and interleaved entropy slice multicore processing techniques can achieve performance improvements of 2.64× and 2.91×, respectively. This extra hardware performance can be used to decode higher definition videos. Alternatively, it can be traded-off for dynamic power savings of 60% relative to a single nominal-voltage decoder. Finally, on-chip caching methods are presented that significantly reduce off-chip memory bandwidth, leading to a further increase in performance and energy efficiency. Data-forwarding caches can reduce off-chip memory reads by 53%, while using a last-frame cache can eliminate 80% of the off-chip reads. The proposed techniques were validated and benchmarked using full-system Verilog hardware simulations based on an existing decoder; they should also be applicable to most other decoder architectures. The metrics used to evaluate the ideas in this paper are performance, power, area, memory efficiency, coding efficiency, and input latency.**

***Index Terms*—H.264, low-power, multicore, parallelism, video decoders.**

Manuscript received January 31, 2009; revised May 21, 2009. First version published September 1, 2009; current version published October 30, 2009. This work was funded by Nokia, and Texas Instruments Incorporated. Chip fabrication was provided by Texas Instruments Incorporated. The work of V. Sze was supported by the Texas Instruments Graduate Women's Fellowship for Leadership Activities in Microelectronics and the Natural Sciences and Engineering Research Council. This paper was recommended by Associate Editor M. Mattavelli.

D. Finchelstein is with Nvidia Corporation, Santa Clara, CA 95050 USA (e-mail: dfinchel@alum.mit.edu).

V. Sze and A. P. Chandrakasan are with the Microsystems Technology Laboratories, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: sze@alum.mit.edu; anantha@mtl.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/TCSVT.2009.2031459

I. Introduction

THE INCREASING demand for higher definition and faster frame rate video is making high-performance and low-power critical in hardware video decoders. Mobile multimedia devices such as smart phones are energy-constrained, so reducing their power is critical for extending video playback times. For wired devices such as set-top boxes, speed is critical for high-quality video playback. Ideally, a video decoder would not require a different architecture for these two types of applications. This would help reduce design time and lower implementation costs.

Pipelining and parallelism, two well-known hardware architecture techniques, can be used to achieve these high-performance requirements. Pipelining increases computation concurrency by reducing the datapath between registers. This allows a circuit to be clocked at a higher frequency, and thus process data faster. One disadvantage of pipelining is the increase in pipeline registers and control complexity. Parallelism increases concurrency by distributing computation amongst several identical hardware units. The main cost of parallelism is an increase in chip area and muxing/demuxing logic to feed all the units and collect their results.

The power of a given video decoder architecture can be minimized by lowering the supply voltage. First, the decoder's clock frequency is set to the lowest value that still guarantees the current computation workload can be met. Next, the supply voltage is reduced to the minimal value that still allows the circuit to operate at the chosen frequency. Voltage scaling reduces dynamic energy consumption by a quadratic factor. This comes at a cost of increased circuit delay, as the currents decrease with supply voltage. Specifically, the circuit suffers a linear increase in delay above the threshold voltage; as the supply voltage approaches the sub-threshold region, the circuit begins to suffer an exponential increase in delay [1]. This decreased speed can be a challenge for real-time applications such as video decoding where on average a new frame must be computed every 33 ms for a frame rate of 30 frames per second (fps).

Video decoding also requires a significant amount of on-chip and off-chip memory bandwidth, for both motion compensation (MC) and last-line accessing. Therefore, memory system optimization can reduce total power in the decoder system, which includes both the decoder application-specific integrated circuit (ASIC) and the off-chip frame buffer memory. One effective way to reduce memory power is the use of

on-chip caching. This technique trades off an increase in chip area for a reduction in more power-hungry off-chip accesses.

A. Related Work

State-of-the-art H.264 ASIC video decoders have used microarchitectural techniques such as pipelining and parallelism to increase throughput and thus reduce power consumption of digital logic [2]–[6]. In this paper, parallelism is applied at the system level by replicating the entire decoder pipeline.

In [2], the performance bottleneck was identified to be the entropy decoding (ED) unit. This is because context-adaptive variable-length coding (CAVLC) processes an inherently serial bitstream and cannot be easily parallelized. This is also seen in [7], where everything but the ED unit was replicated by a factor of 8. One way to overcome the ED performance bottleneck is to run it at a faster frequency, as suggested in [8] and [9]. However, the ED unit must be run at a higher voltage than the rest of the system, so it will lower the overall energy efficiency. Also, even at the maximum frequency allowed by the underlying transistor technology, the ED unit might not be able to run fast enough to meet the highest performance demands.

A system-level approach to increasing decoder throughput is to break the input stream into slices that can be processed in parallel by multiple cores, which has been proposed by [10] and [11]. [10] proposes breaking up each frame into completely independent slices; this method was described for MPEG-2 but is also applicable within the H.264 standard at the cost of lower coding efficiency, as will be shown in Section II-A. [11] proposes breaking up each frame into "entropy" slices where only the ED portion is independent; this method is not H.264 compliant.

State-of-the-art H.264 decoders have also used last-line and MC caching to reduce power consumption of the memory subsystem [2]–[6]. Another technique to reduce off-chip frame buffer (OCFB) bandwidth (BW) is to compress the reference frames [12]. The idea of [12] is not H.264 compliant and must be performed in the same way at the encoder and decoder.

B. Main Contributions of This Paper

The main contributions of this paper fall in two categories: 1) multicore parallelism (Sections II and III) and 2) on-chip memory caching (Section IV).

Multicore parallelism consists of replicating an existing video decoder (DEC) architecture, as shown in Fig. 1. Each of the parallel DEC processes different parts of the bitstream, and together they produce an output video.

First, we describe a way of decoding multiple H.264 frames simultaneously, while achieving a linear improvement in performance with no loss in coding efficiency. Second, through coexploration of algorithm and architecture, we develop interleaved entropy slice (IES) processing; this method also achieves a linear increase in throughput with negligible impact on coding efficiency, and has lower input latency than the multiple frame technique.

Third, we show how using either a last-frame cache (LFC) or data-forwarding cache (DFC) can drastically reduce OCFB read BW. Fourth, we show how IES processing increases data locality and reduces the BW of a full-last-line cache (FLLC).

Finally, the different techniques are evaluated for speed, area, power, latency, and coding loss, and the results are summarized in Section V. The proposed architectures were implemented using Verilog and the coding loss was simulated using the H.264 reference software [13]. The underlying DEC architecture used for all the analysis is based on [2] and [8].

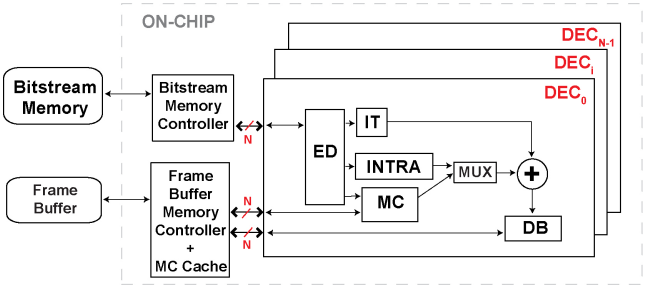


Fig. 1. Parallel video decoder architecture.

II. Video Decoder Replication for H.264

Previously published H.264 DEC have used parallelism within the DEC units [for example, deblocking filter (DB) or MC] to increase system performance. In this section, we will describe different ways in which two or more DEC can process a H.264 video in parallel and therefore increase system performance. The goal of these techniques is to enable N DEC to execute concurrently, in order to achieve a performance improvement of up to N. These techniques are also cumulative, so they could be used together to expose even more parallelism. While this section deals only with H.264-compliant video processing, Section III will describe other ways to expose the desired parallelism by slightly modifying the H.264 algorithm. The metrics used to evaluate the ideas are performance, power, area, memory efficiency, coding efficiency, and input latency.

A. Slice Parallelism

A scheme that enables high-level DEC parallelism is the division of a frame into two or more H.264 slices at the video encoder (ENC). Each slice can then be processed by a separate DEC core. Parallel slice processing relies on the ability of the DEC's ED to parse two or more slices simultaneously, and also assumes that the ENC divides each frame into enough slices to exploit parallelism at the DEC.

In the H.264 standard [14], each slice is preceded by a small 32-bit delimiter code, as shown in Fig. 2. If the DEC can afford to buffer an entire encoded frame of the input stream and quickly parse for the start code of all slices, then it can simultaneously read all the slices from this input buffer. This idea is similar to the parallel MPEG-2 decoder described in [10].

This scheme trades off increased parallelism for a decrease in coding efficiency. We evaluated the impact of slice parallelism by encoding 150 frames of four different video

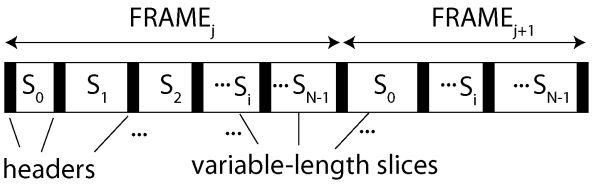


Fig. 2. Start of slices can be found by parsing for headers.

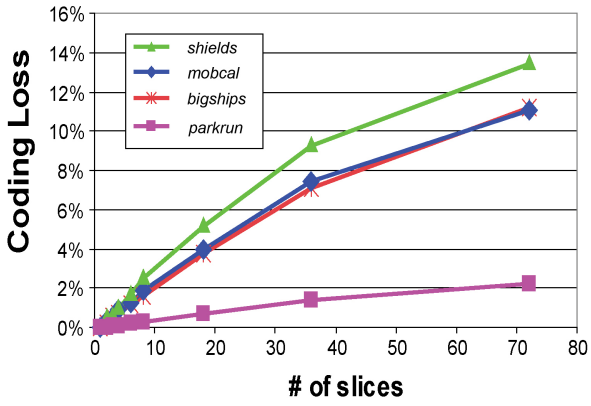


Fig. 3. CAVLC coding loss increases with number of H.264 slices in a 720 p frame.

sequences and separating each frame into a fixed number of slices, using the JM reference software [13] with QP = 27. The result is shown in Fig. 3. Relative to having single-slice frames, the coding efficiency decreases because the redundancy across the slice borders is not exploited by the ENC. Furthermore, the size of the slice header information is constant while the size of the slice body decreases because it contains fewer macroblocks (MBs), which are blocks of 16 × 16 pixels. For example, when dividing a 720 p frame into three slices, the CAVLC coding method suffers an average 0.41% coding loss, when measured under common conditions [15].

Besides the loss in coding efficiency, another disadvantage of the slice partitioning scheme is that the FLLCs need to be replicated together with each DEC, since they operate on completely different regions of the frame. This causes the area overhead of parallelism to be nearly proportional to the degree of parallelism, since the DEC pipeline is replicated, but not the memory controller logic. In some DEC implementations the on-chip cache dominates the active area in [2, 75%], so replicating the FLLC incurs a large area cost.

The performance improvement of H.264 slice multicore parallelism is shown in Fig. 16. Ideally, the performance improvement of slice parallelism with N decoders is at most N. However, there are two reasons why the performance does not reach this peak. First, the workload is not evenly distributed amongst the parallel slices, especially since they operate on disjoint regions of the frame which could have different coding characteristics. Second, the increase in total bits per MB due to loss in coding efficiency (more nonzero coefficients, for example) leads to an increase in ED computation cycles.

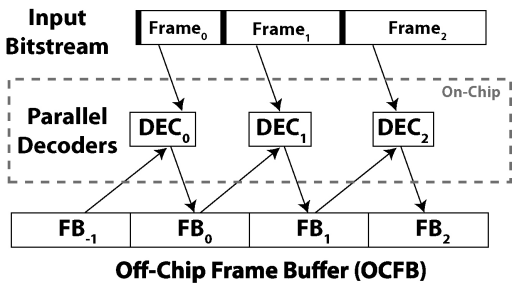


Fig. 4. Three parallel video decoders processing three consecutive frames of the same video.

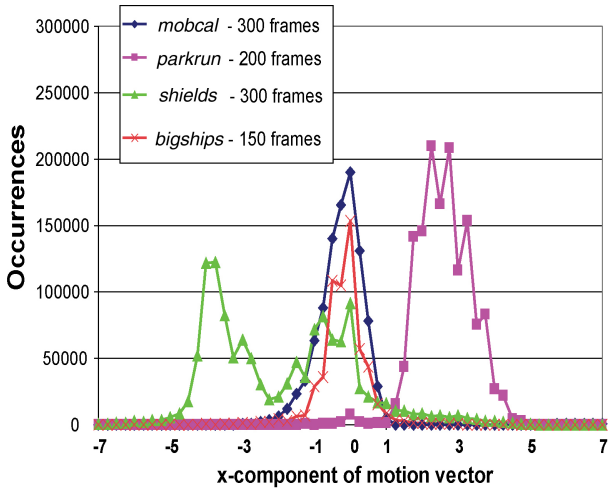
B. Frame Parallelism

In this section, we show how to process N consecutive H.264 frames in parallel, without requiring the ENC to perform any special operations, such as splitting up frames into N slices. The simultaneous parsing of several frames relies on input buffering and searching for delimiters, similar to the discussion of Section II-A. However, note that this technique requires buffering N frames, so it will incur a higher input latency than the buffering of N slices.

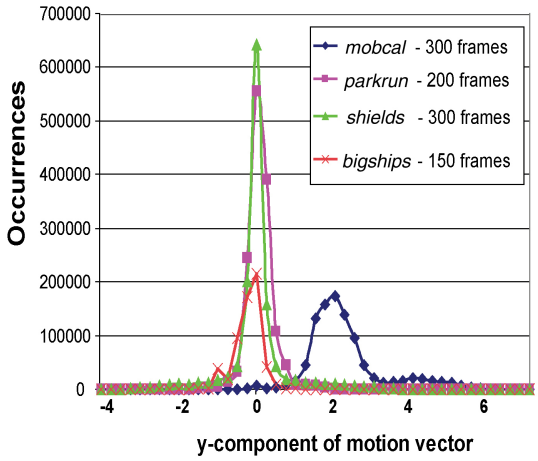
Several consecutive frames can be processed in parallel by N different DEC, as shown in Fig. 4. The main cost of multiframe processing is the area overhead of parallelism, which is proportional to the degree of parallelism, just as in Section II-A. If these frames are all I-frames (spatially predicted), then they can be processed independently from each other. However, when these frames are P-frames (temporally predicted), DEC_i requires data from frame buffer (FB) location FB_{i-1}, which was produced by DEC_{i-1}. If we synchronize all the parallel DEC, such that DEC_i lags sufficiently behind DEC_{i-1}, then the data from FB_{i-1} is usually valid.

If the motion vector (MV) in DEC_i requires pixels not yet decoded by DEC_{i-1}, then concurrency suffers and we must stall DEC_i. This could happen if the y-component of the MV is a large positive number. The performance decrease due to these stalls was simulated to be less than 1% for $N = 3$, across 100 frames of a 720 p "*Mobcal*" video sequence. The relatively small number of stalls for the simulated videos can be understood by examining the statistics of their vertical motion vectors. As shown in Fig. 5(b), the y-motion vectors for various videos are typically small and have a very tight spread, which minimizes stalling.

Frame multicore decoding can increase the DEC performance by up to a factor of N. The parallel frame processing architecture was implemented in Verilog using the core of [2] for each of the DEC. The architecture was then verified for different video sequences and varying degrees of parallelism. Fig. 16 shows how the maximum clock period increases for a given resolution as we process more frames in parallel. For a given resolution, a larger clock period is made possible by an increase in performance, since fewer cycles are needed to decode a given workload. This increase is nearly linear, but is limited by the workload imbalance across the various sets of frames running on each of the parallel DEC.



(a)



(b)

Fig. 5. Distribution of vertical motion vectors for several conformance videos showing a tight spread.

III. Decoder Replication for Future Video Standards

In this section, we propose some changes to the H.264 algorithm which allows two or more DEC to process a video in parallel and therefore increase system performance. As in Section II, the goal of these techniques is to enable N DEC to execute concurrently, in order to achieve performance improvement of up to N. The multicore ideas in this section offer several advantages over the techniques in Section II, such as better workload distribution among the parallel cores, lower cache area requirements, smaller loss in coding efficiency, and a much smaller input buffering latency.

A. Diagonal Macroblock Processing

The H.264 coding standard processes the MB of video frames in raster-scan order. In order to exploit spatial redundancy, each MB is coded differentially with respect to its already decoded neighbors to the left, top-left, top, and top-right. The redundancy between neighbors is present in both pixel values and prediction information (motion vectors, number of coded coefficients, etc.).

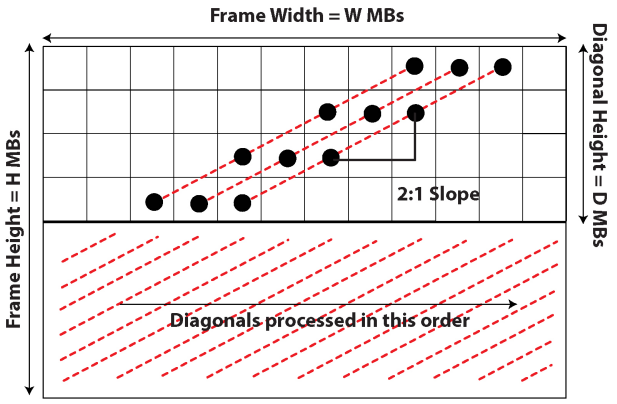


Fig. 6. Processing order is on a 2 to 1 diagonal.

We could instantiate multiple DEC to process the MB on a 2:1 diagonal as shown in Fig. 6. This is similar to the processing order described in [7]. The diagonal height D could be set to anywhere from 1 to H (frame height). The different diagonals are ordered from left to right. Setting $D = 1$ corresponds to the typical raster-scan processing order.

If diagonal processing is used, all the MB on a diagonal can be decoded concurrently since there are no dependencies between them. If all MB had similar processing workloads, the scheme described in this section could speed up the DEC by N (degree of DEC replication). In reality, the workload per MB does vary, so the performance improvement is lower than the increase in area. The diagonal height D of each region of diagonals can be set to N, since no further parallel DEC hardware is available. Note that the top line of MB in each region of diagonals is still coded with respect to the MB in the region of diagonals just above, in order to maintain good coding efficiency.

A limitation to implementing this scheme is that the coded MB in H.264 arrive in raster-scan order from the bitstream. One solution would be to modify the algorithm and reorder the MB in a 2:1 diagonal order at the ENC. For example, the MB in each diagonal could be transmitted from top-right to bottom-left in the bitstream. Another ordering could transmit MB on even diagonals top-right to bottom-left and MB on odd diagonals from bottom-left to top-right.

This reordering would require a change in the H.264 standard, so that both the ENC and DEC now process MB in a diagonal order. The CAVLC entropy coding efficiency would not suffer, since each MB can be coded in the same way as for the raster-scan ordering of H.264. Therefore, the reordered CAVLC bitstream would contain the same bits within the MB, but the MB would just be rearranged in a different order.

However, even if diagonal reordering is used, we still cannot scan ahead to the next MB since the current MB has variable length and there are no MB delimiters. This critical challenge is addressed in the next section.

B. Interleaved Entropy Slice (IES) Parallelism

In order to enable DEC parallelism when using the diagonal scanning order of Section III-A, we propose the following solution. The bitstream can be split into N different IES; for

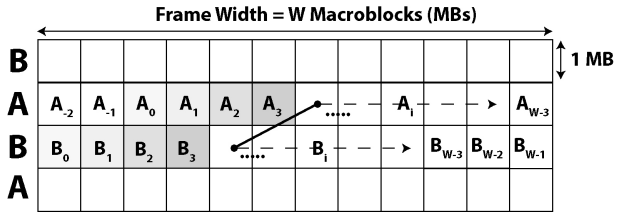


Fig. 7. Interleaved entropy slices (IESs) with diagonal dependencies.

example, when $N = 2$, IES A and B in Fig. 7. Therefore, each of N parallel DEC would be assigned to an entire MB line, and the IES would be interleaved amongst these lines. Just as slices are separated for H.264, the bitstream could be split into different IES.

There are several key differences between IES and the entropy slices of [11]. First, IES allows for context selection across slice boundaries, which improves coding efficiency relative to [11]. Second, IES are interleaved to enable better parallel processing and memory locality. Finally, IES allows the full decoding path to be parallelized; the decoded syntax elements can be fed directly to the rest of the decoder which can begin processing immediately, avoiding the need for intermediate buffering which is necessary for traditional slice partitioning.

The processing of IES would be synchronized to ensure that the 2:1 diagonal order is maintained. As a result, each DEC must trail the DEC above. However, if one IES has a higher instantaneous processing workload than the IES above it, the DEC above can move forward and proceed further ahead, so that stalling is minimized.

This approach is different than the one used in [7]. In that paper, the ED processing was done in the usual raster-scan order and all the syntax elements were buffered for one frame. The diagonal processing could only start after the entire frame was processed by ED. In the IES approach, which would be enabled by a change in the H.264 algorithm, even the ED processing is done in parallel, which speeds up the ED operation and does not require buffering any syntax elements.

This technique is similar to the dual macroblock pipeline of [9]. In that paper, the authors duplicate the MB processing hardware at the encoder, whereas in this paper we replicate the DEC at the decoder. Parallel processing at the decoder is more challenging than at the encoder since the input is a variable-length bitstream and the MB are transmitted in a fixed raster-scan order. The encoder has the flexibility to process MB in any order, whereas interleaved processing at the decoder requires a change in the H.264 standard.

It is worth considering how the use of IES affects the entropy coding efficiency. Once again, if the video uses CAVLC, the bitstream size will only be slightly affected, since the macroblocks are coded in the same way as the raster-scan order of H.264. The only coding overhead is the 32 bits used for the slice header and at most 7 extra bits for byte alignment between slices. As we see in Fig. 8, this scheme offers much better coding efficiency than using CAVLC with H.264 slices, since there is no loss in coding efficiency at the borders between IES.

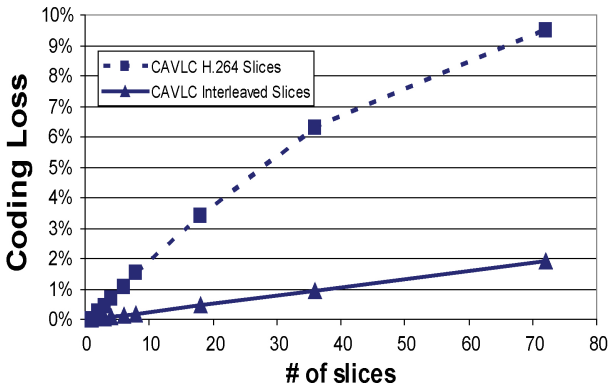


Fig. 8. Average CAVLC coding efficiency of IES relative to parallel H.264 slice processing of Section II-A for 150 frames of four different videos: "*Bigships*," "*Mobcal*," "*Shields*," and "*Parkrun*." The coding efficiency curve for H.264 slices is the average of the curves shown in Fig. 3.

To evaluate the actual performance of a real system, we implemented the IES parallelism scheme in Verilog and evaluated it for several videos and degrees of parallelism. The performance of IES multicore decoding is shown in Fig. 9 for varying N. Ideally, the IES parallelism technique can speed up the DEC performance by up to a factor of N. This was also shown in [7], where an 8-core decoder achieved a 7.5× improvement in throughput. In reality, an exact linear increase in performance cannot be achieved due to varying slice workloads (as discussed in Section II-A) and stalls due to synchronization between the DEC. The increased performance of multicore IES can be mapped to a larger clock period for a given workload. This allows lower voltage operation and the corresponding power savings are shown in Fig. 9. The area costs associated with multicore IES are also shown in Fig. 9. Unlike the multicore ideas of Section II, only one last-line cache is needed independent of the number of cores, as will be shown in Section IV-D. As a result, the total area costs of this scheme do not increase linearly with the degree of parallelism.

Fig. 9 shows that the rate of power reduction decreases as we add more and more DEC cores, eventually leveling off. For example, we need 3 cores and 39% more area to save 60% of the power relative to a single-core DEC. However, a 25-core implementation only saves 38% of the power relative to a 3-core DEC, but uses 4.1× the area. Fig. 16 shows that IES perform better than regular H.264 slices, and there are several reasons for that. First, the workload variation is not as large between interleaved slices since they cover similar regions of a frame; as N increases, however, the variation in interleaved slice workloads also gets larger. Second, IES parallelism does not suffer from a large coding penalty, so the ED performance does not suffer as a result.

IV. Memory Optimization

The memory subsystem is critical to both the performance and power of a DEC. A state-of-the-art OCFB dynamic random access memory (DRAM) such as the one in [16] can consume as much power as the DEC processing itself, as

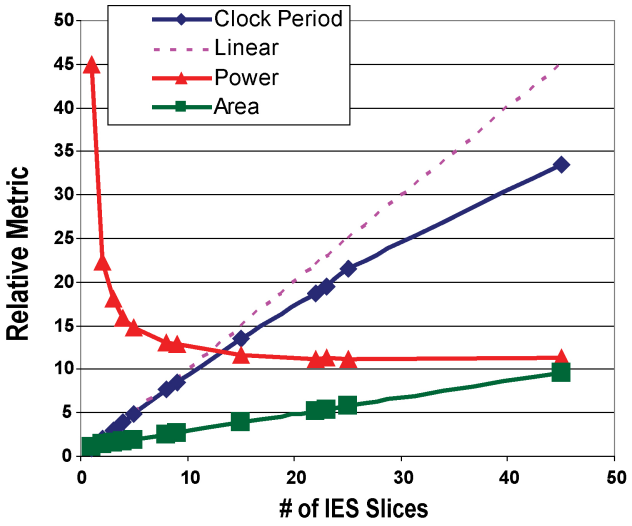


Fig. 9. Performance of IES multicore decoding. The power is normalized relative to a single decoder running at the nominal supply voltage. The area increase assumes caches make up 75% of the area of a single DEC and the memory controller takes 23% of the logic [2].

estimated in [8]. Additionally, off-chip accesses require further power to charge high-capacitance bondwires and PCB traces. Therefore, reducing off-chip memory accesses is important for minimizing system power.

The off-chip *write* BW of a DEC is bounded on the low side by the number of pixels output by the decoding process. However, the off-chip *read* BW of the MC block is typically higher than the write BW due to the use of a six-tap interpolating filter. The following sections propose different ways to reduce the MC off-chip read BW with the help of on-chip caches.

A. Existing Memory Optimization Techniques

The work in [8] discusses two main caching techniques for reducing off-chip BW of a hardware DEC: FLLC and MC caches.

The top-neighbor dependency requires each MB to refer to the MB above in the last line. The use of a FLLC allows us to fetch this data from on-chip SRAMs rather than getting the previously processed data from a large off-chip memory. As a result, the off-chip memory BW is reduced by 26%. For 720 p resolutions, the area cost of this technique is 138 kbits of on-chip SRAM [8], whereas for 1080 p the FLLC size increase to 207 kbits.

To further reduce off-chip memory BW, the MC data read from the previous frame can also be cached on-chip. In [2], the MC block identifies the horizontal and vertical overlap of interpolation area between two adjacent 4×4 blocks of pixels with identical MV. This data is cached on-chip in flip-flops, so as not to be read twice from the OCFB. This technique was shown to reduce the total off-chip BW by 19%. If the more general MC cache of [8] is used, a further 33% of OCFB read BW can be saved with a moderately sized cache (512 Bytes). A much larger MC cache (32 kBytes for 720 p resolutions) is needed to increase the read BW savings from 33% to 56% over the simple MC caching of [2]. The following sections present new caching ideas that can be used independently or together with existing techniques.

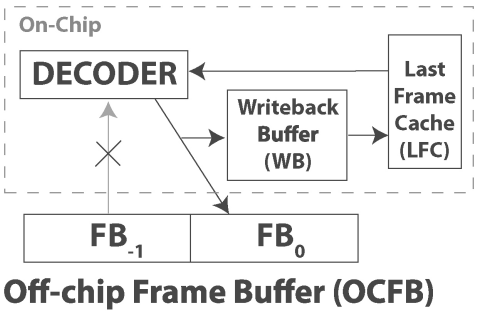


Fig. 10. Caching an entire frame on chip for MC.

B. Last-Frame Cache (LFC) for Motion Compensation

During MC, most of the pixels are read from the previous frame, as opposed to being read from even earlier frames. If we can store the last reference frame in an on-chip LFC, we can avoid going off-chip for the majority of MC reads. This caching architecture is described in Fig. 10(a), which shows how reads from FB_{-1} are replaced with reads from the LFC. This scheme requires a WB in order to not overwrite the data at the current location in the LFC, which is needed for MC. To understand the need for a WB, let us assume that there is no WB and the MV for the current block at location (x, y) is $(-10, -10)$. In this case, the data from the last frame at location $(x-10, y-10)$ would no longer be found in the LFC, since it would have already been overwritten by the block at location $(x-10, y-10)$ from the current frame.

One overhead of this LFC scheme is the significant additional area of the WB and the LFC. There is also a power overhead, since each decoded pixel is now written to the LFC (as well as to the OCFB), and written to and read from the WB, all of this just to avoid reading it back from the OCFB.

For 720 p resolutions, the size of the LFC would be 1.4 MBytes, with an area of 2.7 mm^2 if implemented with high-density embedded DRAM (eDRAM) [16]. The size of the WB depends on how many misses we are willing to tolerate in the LFC. Fig. 11 shows how the hit rate of the LFC varies with the size of the WB, as simulated in Verilog. If there is no WB, the videos with more movement from left to right or up to down will have more LFC misses. For example, for the "*Shields*" video, the LFC hit rate with no WB is 65% because the movement is from left to right. A small WB with the size of 1 MB can improve the hit rate up to about 93%. To eliminate the remaining misses, the entire row of MB above must be buffered by the WB, which explains the last jump up to 100% when the WB size is 80 MB. A miss occurs in the LFC when the block being fetched has a much smaller y-coordinate than the current block being processed. In the case of this miss, the MC data was already overwritten by a recently decoded block which was evicted from the WB. If this happens, the data must be fetched from FB_{-1}. If the reference frame is not the last frame, the LFC is also bypassed and the data is fetched from the OCFB. The work in [17] shows that the previous frame is chosen 80% of the time as the reference frame, as averaged

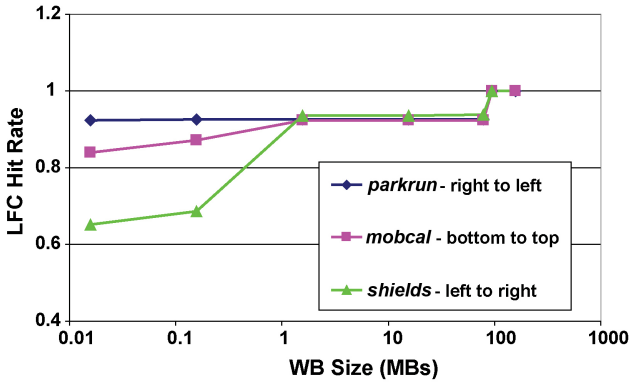


Fig. 11. Hit rate of last-frame cache versus size of writeback cache for different 720 p videos. For each video, the type of motion is described, in order to help explain the differences in hit rates.

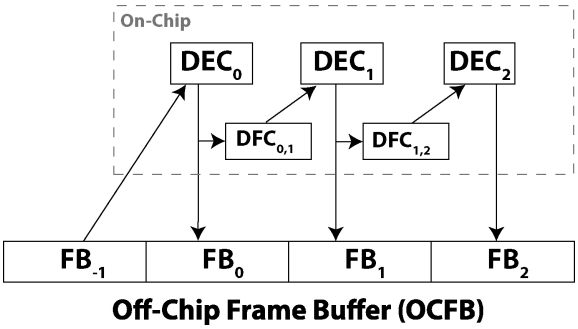


Fig. 12. Motion compensation (MC) DFC for $N = 3$.

over 10 different videos of common intermediate format (CIF) resolution.

C. Motion Compensation Data-Forwarding Caches (DFCs)

If we allow N parallel DEC to operate concurrently on N consecutive frames, as in Section II-B, we can forward MC data between them using on-chip DFC, as shown in Fig. 12. This will avoid most off-chip MC reads for all DEC but DEC_0, greatly reducing off-chip read BW (up to 67% for $N = 3$) and power. The OCFB BW only depends on the video frame rate and resolution, and is independent of the number of DEC used.

In general, DEC_i and DEC_{i-1} need to be synchronized, such that DEC_i lags sufficiently behind DEC_{i-1}, similar to the discussion in Section II-B. Conversely, if DEC_{i-1} gets too far ahead of DEC_i, the temporal locality is lost, and the MC data will be read from the OCFB instead of from $DFC_{i-1,i}$. In that case, we can stall DEC_{i-1} in order to maximize the hit rate of the DFC. These two constraints can be handled with the help of low and high-watermarks.

In order to evaluate the performance impact and hit rate of these DFC, we implemented the DFC in Verilog and placed them between the DEC described in Section II-B. The performance impact of stalling at these watermarks was simulated for a "*Mobcal*" video sequence of 100 frames. The overall loss in throughput for $N = 3$ was less than 8%.

The DFC need to store about 32–64 lines of pixels to minimize the cache miss rate, so their on-chip area can be quite large for high-resolution, highly parallel DEC. To understand

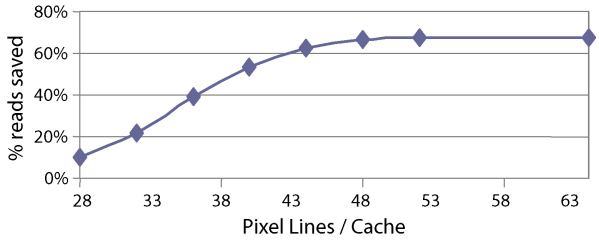


Fig. 13. Reduction in off-chip reads versus size of MC DFC for $N = 3$.

the trade-off between the size of the DFC and the hit rate, we simulated the DFC system for 100 frames of the "*Mobcal*" video. The result is shown in Fig. 13. As expected, a really large cache will have near 100% hit rate, leading to 67% reduction in off-chip MC reads for $N = 3$. The hit rate drops off significantly for DFC sizes of less than 32 lines, since the vertical MV can easily fall outside this range. For $N = 3$ and 720 p resolution, the total area of the two 64-line DFC is about 1 mm^2, assuming high-density 65 nm SRAMs.

D. Last-Line Caching for Interleaved Entropy Slices (IESs)

In addition to enabling parallel processing, the IES of Section III-B also allow for better memory efficiency than the raster-scan processing in H.264. This section shows how the IES processing order can reduce accesses to the large FLLC discussed in Section IV-A. For example, when decoding B_i in Fig. 14, the data from MB A_{i-1}, A_{i-2}, A_{i-3} can be kept in a much smaller cache since those MB were recently processed by DEC_A and have high-temporal locality.

The caches that pass data vertically between decoders, such as DEC_A to DEC_B in Fig. 14, are implemented as FIFOs. A deeper FIFO could better handle workload variation between the IES by allowing DEC_A to advance several MB ahead of DEC_B and thus reduce stall cycles and increase throughput. The caches that pass data horizontally within each decoder only need to hold the information for 1 MB, and are unchanged from the H.264 raster-scan implementation. However, when we process A_i, the FLLC is still needed to hold the data that is passed from DEC_C to DEC_A, since DEC_C writes this data long before DEC_A can process it. The depth of the FLLC FIFO should therefore be about as large as the frame width in order to prevent deadlock. The caching of data for IES processing is similar to the one used in the encoder of [9].

To evaluate the performance impact of sizing the FIFOs of Fig. 14, we implemented the IES caches in Verilog and placed them together with the system of Section III-B. When simulating intra frames for $N = 3$, we found that a FIFO depth of four 4×4 edges (one MB edge) only has a 3% performance penalty, whereas a minimally sized FIFO can reduce system performance by almost 25%. This trade-off is illustrated in Fig. 15.

The FLLC FIFO is read by DEC_0 and written to by DEC_{N-1}, so if a single-ported memory is used, the accesses will need to be shared. The total size of the IES inter-slice FIFOs is independent of the degree of parallelism N, since the FLLC is not replicated with each DEC. This implies that the total area overhead of DEC parallelism with diagonal

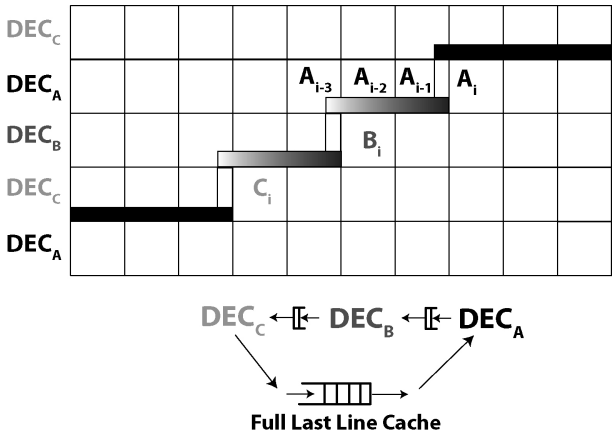


Fig. 14. Caches used for IES processing with three DEC.

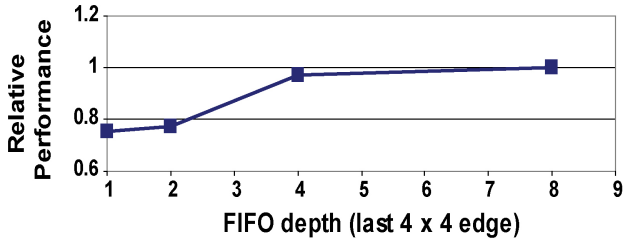


Fig. 15. Impact of FIFO sizing on parallel IES performance.

processing is not a factor of N, as was the case for the techniques in Section II. For the DEC of [2], the area of the FLLC SRAMs was three times larger than the rest of the DEC logic. As a result, for $N = 3$, the area increase due to parallelism would be about 50% and not 200%. This analysis only includes the DEC of Fig. 1, and does not include the area of the frame buffer and bitstream memory controller.

If N is the number of parallel IES DEC, the number of accesses to the large FLLC are reduced to $1/N$ of the original. These accesses are replaced with accesses to much smaller FIFOs that hold the information for about 1 MB. This uses less energy than accessing a large memory that stores 80 MB for 720 p, or 120 MB for 1080 p. This reduction in FLLC accesses allows the designer to even eliminate the area-hungry FLLC and just use the large off-chip memory where the frame buffer is stored.

It is interesting to note that diagonal processing enabled by IES can reduce FLLC accesses even when only one DEC is used (no DEC replication). This would require the single DEC to alternate between different IES whenever one of the FIFOs in Fig. 14 stalls.

The improved data locality of IES processing can also benefit the MC cache described in Section IV-A, enabling a higher hit rate. Specifically, IES enables vertically neighboring MB to be processed simultaneously. The read areas of these MB overlap which enables an improved hit rate and consequently reduce OCFB BW. For example, the hit rate of a 2 kB MC cache was simulated to be 5% larger than for an equally sized MC cache of a DEC that uses regular raster-scan MB ordering.

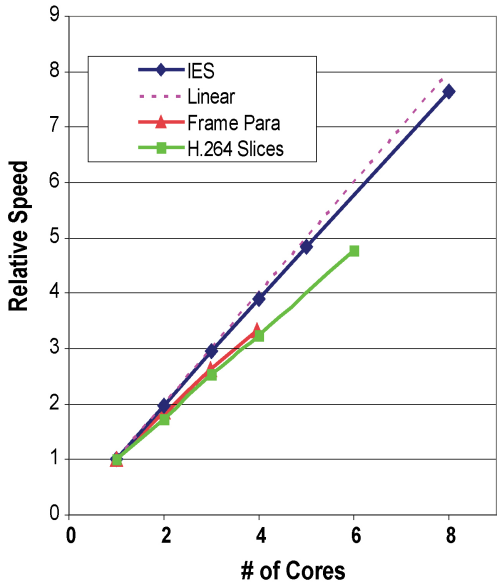


Fig. 16. Three different multicore architectures show nearly linear performance gains. The multicore performance of H.264 slices is slightly lower because of the extra processing required by the CAVLC and also the unbalanced slice workload due to uneven image characteristics across the slices.

V. Results

The high-level parallelism in this paper was achieved by replicating a full decoder of an arbitrary architecture. The different architectures we proposed were implemented, verified, and benchmarked in Verilog. Fig. 16 shows that all multicore architectures achieve a near-linear speedup and corresponding clock frequency reduction for a given resolution. However, as was shown in Fig. 9, extending the level of multicore parallelism to much higher than 3 achieves relatively small power savings at the cost of a much larger area. As a result, we compare these different multicore architectures for $N = 3$, as shown in Table I.

Their relative coding efficiency was quantified by running several experiments using a modified version of the reference H.264 software. The results are summarized and compared in Table I. The table lists the performance achieved when the decoder is replicated three times, relative to the performance of a single decoder. A 3-core implementation was found in Section III-B to be a good trade-off of power savings versus area. As discussed previously, this performance can be traded-off for a slower clock and lower voltage, and the equivalent power savings are also shown in Table I. These dynamic power savings assume the original single decoder runs at full voltage and is voltage-scaled when parallelism is enabled.

In addition to video decoder parallelism, several on-chip caching techniques were introduced that significantly reduce the off-chip memory bandwidth requirements. The different caching ideas were implemented, verified, and benchmarked in Verilog. They are summarized and compared in Table II. The first two techniques reduce off-chip frame buffer bandwidth by using large on-chip caches. The third technique takes advantage of IES processing to provide better data locality and thus minimize accesses to the FLLC.

TABLE I
VIDEO DECODER PARALLELISM (N = 3, 720P) COMPARISON DONE FOR A FIXED WORKLOAD FOR DIFFERENT TECHNIQUES RELATIVE TO A SINGLE-CORE DECODER AT FULL VOLTAGE

Parallelism Technique	H.264 Slices	H.264 Frames	Interleaved Entropy Slices
Paper Section	II-A	II-B	III-B
Degree of Parallelism	3	3	3
Relative Performance	2.51×	2.64×	2.91×
Equivalent Dynamic Power Savings	58%	59%	60%
CAVLC Coding Loss	0.41%	0%	0.05%
Relative Last-Line Size	3.00×	3.00×	1.03×
Relative Logic Area	2.54×	2.54×	2.54×
Input Buffering Latency (ms)	22	66	22
H.264 Compliance	Yes	Yes	No

TABLE II
SUMMARY AND COMPARISON OF DIFFERENT DEC CACHING TECHNIQUES. RESULTS SHOWN FOR 720 P ARE RELATIVE TO A DECODER WITH ONLY A FLLC

Caching Technique	LFC with 32-line WB	48-line DFC *N=3*	1-MB FIFOs for IES *N=3*
Paper Section	IV-B	IV-C	IV-D
Cache Type	eDRAM	SRAM	FIFO Flip-Flops
Cache Size (kB)	963	123	0.43
Silicon Area in 65nm (mm^2)	2.7	1.0	0.029
OCFB MC BW Reduction	80%	53%	>0%*
FLLC BW Reduction	0%	0%	67%
Memory Access Power Savings	60%	44%	65%
H.264 Compliance	Yes	Yes	No

*OCFB BW can be reduced further if IES processing is combined with a MC cache.

The memory power savings listed in Table II are with respect to the power of the accesses which the cache helps to reduce. To calculate the exact energy savings based on the different cache hit rates, we simulated and estimated the energies for the different types of memories involved. The normalized energy per bit for each of the types of memories are as follows: 1) 1 for a FIFO flip-flop from the synthesis library; 2) 19 for a large eDRAM [16]; 3) 51 for a large static random access memory (SRAM) [2]; and 4) 672 for an off-chip synchronous dynamic random access memory (SDRAM) [18], and 10 pF/pin.

VI. CONCLUSION

In order to handle the high-computation load of modern high-definition hardware decoders, parallelism needs to be exposed wherever possible. In this paper, we presented several ways to enable high-level parallelism and provide a clear performance-area trade-off. If performance, power, area, memory efficiency, coding efficiency, and input latency are key concerns for the video decoder designer, we recommend choosing the proposed IES architecture. In all of these metrics, IES processing provides comparable or better results relative to the other techniques, though it requires a slight change in the video standard. To optimize the memory system, a larger cache (LFC) can reduce more off-chip bandwidth but requires the most area; a good power–area compromise is provided by the DFC of Section IV-C.

ACKNOWLEDGMENTS

The authors would like to thank M. Budagavi and T. Ono for their valuable feedback on this paper.

REFERENCES

[1] J. M. Rabaey, A. P. Chandrakasan, B. Nikolic, and J. M. Rabaey, in *Digital Integrated Circuits*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, Dec. 2002.

[2] D. F. Finchelstein, V. Sze, M. E. Sinangil, Y. Koken, and A. P. Chandrakasan, "A low-power 0.7-V H.264 720p video decoder," in *Proc. IEEE Asian Solid State Circuits Conf.*, Fukuoka, Japan, Nov. 3–5, 2008, pp. 173–176.

[3] C.-D. Chien, C.-C. Lin, Y.-H. Shih, H.-C. Chen, C.-J. Huang, C.-Y. Yu, C.-L. Chen, C.-H. Cheng, and J.-I. Guo, "A 252 kgate/71 mW multi-standard multi-channel video decoder for high definition video applications," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, San Francisco, CA, Feb. 11–15, 2007, pp. 282–603.

[4] S. Na, W. Hwangbo, J. Kim, S. Lee, and C.-M. Kyung, "1.8 mW, hybrid-pipelined H.264/AVC decoder for mobile devices," in *Proc. IEEE Asian Solid State Circuits Conf.*, Jeju, South Korea, Nov. 12–14, 2007, pp. 192–195.

[5] T.-M. Liu, T. Lin, S. Wang, W. P. Lee, K. Hou, J. Yang, and C. Lee, "A 125 uW, fully scalable MPEG-2 and H.264/AVC video decoder for mobile applications," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 161–169, Jan. 2007.

[6] C.-C. Lin, J.-W. Chen, H.-C. Chang, Y.-C. Yang, Y.-H. O. Yang, M.-C. Tsai, J.-I. Guo, and J.-S. Wang, "A 160 K gates/4.5 KB SRAM H.264 video decoder for HDTV applications," *IEEE J. Solid-State Circuits*, vol. 42, no. 1, pp. 170–182, Jan. 2007.

[7] S. Nomura, F. Tachibana, T. Fujita, H. C. K. T. Usui, F. Yamane, Y. Miyamoto, C. Kumtornkittikul, H. Hara, T. Yamashita, J. Tanabe, M. Uchiyama, Y. Tsuboi, T. Miyamori, T. Kitahara, H. Sato, Y. Homma, S. Matsumoto, K. Seki, Y. Watanabe, M. Hamada, and M. Takahashi, "A 9.7 mW AAC-decoding, 620 mW H.264 720 p 60 fps decoding, 8-core media processor with embedded forward-body-biasing and power-gating circuit in 65 nm CMOS technology," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, San Francisco, CA, Feb. 2008, pp. 262–612.

[8] V. Sze, D. F. Finchelstein, M. E. Sinangil, and A. P. Chandrakasan, "A 0.7-V 1.8 mW H.264/AVC 720 p video decoder," *IEEE J. Solid-State Circuits*, vol. 44, no. 11, Nov. 2009.

[9] K. Iwata, S. Mochizuki, T. Shibayama, F. Izuhara, H. Ueda, K. Hosogi, H. Nakata, M. Ehama, T. Kengaku, T. Nakazawa, and H. Watanabe, "A 256 mW full-HD H.264 high-profile CODEC featuring dual macroblock-pipeline architecture in 65 nm CMOS," in *Proc. Symp. Very-Large-Scale Integr. Circuits Dig. Tech. Papers*, Honolulu, HI, Jun. 18–20, 2008, pp. 102–103.

[10] L. Phillips, S. V. Naimpally, R. Meyer, and S. Inoue, "Parallel architecture for a high definition television video decoder having multiple independent frame memories," U.S. Patent No. 5 510 842, Matsushita Electric Corporation of America, Apr. 23, 1996.

[11] J. Zhao and A. Segall, "VCEG-AI32: New results using entropy slices for parallel decoding," in *Proc. 35th Meet. ITU-T Study Group 16 Question 6, Video Coding Experts Group (VCEG)*, Berlin, Germany, Jul. 2008.
[12] M. Budagavi and M. Zhou, "Video coding using compressed reference frames," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, Las Vegas, NV, 2008, pp. 1165–1168.
[13] *H.264/AVC Reference Software, JM 12.0* [Online]. Available: http://iphome.hhi.de/suehring/tml/
[14] *Advanced Video Coding for Generic Audiovisual Services*, document Rec. ITU-T H.264.doc, ITU-T, 2003.
[15] T. K. Tan, G. Sullivan, and T. Wedi, *Recommended Simulation Common Conditions for Coding Efficiency Experiments Revision 1*, document ITU-T SG16/Q6.doc, ITU-T VCEG-AE010, Jan. 2007.
[16] K. Hardee, F. Jones, D. Butler, M. Parris, M. Mound, H. Calendar, G. Jones, L. Aldrich, C. Gruenschlaeger, M. Miyabayashil, K. Taniguchi, and I. Arakawa, "A 0.6-V 205 MHz 19.5 ns tRC 16 Mb embedded DRAM," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 15–19, 2004, pp. 200–522.
[17] Y.-W. Huang, B.-Y. Hsieh, T.-C. Wang, S.-Y. Chient, S.-Y. Ma, C.-F. Shen, and L.-G. Chen, "Analysis and reduction of reference frames for motion estimation in MPEG-4 AVC/JVT/H.264," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, vol. 3. Apr. 6–10, 2003, pp. 145–148.
[18] *Micron Mobile DRAM, The Secret to Longer Life* [Online]. Available: http://www.micron.com

Daniel F. Finchelstein (M'09) received the Ph.D. and M.S. degrees in 2009 and 2005, respectively, from the Massachusetts Institute of Technology, and the B.A.Sc. degree in 2003 from the University of Waterloo. His doctoral thesis focused on energy and memory efficient parallel video processing.

He is currently working in the 3-D graphics performance group at Nvidia Corporation. His research interests include energy-efficient and high-performance digital circuits and systems. His recent focus has been on parallel processors and memory architectures for video and graphics processing. He has been an engineering intern on 8 different occasions at companies like IBM, ATI, Sun, and Nvidia.

Dr. Finchelstein received student design contest awards at both A-SSCC (2008) and DAC/ISSCC (2006), and has several conference and journal publications. He was awarded the Natural Sciences and Engineering Research Council of Canada (NSERC) graduate scholarship from 2003 to 2008. He also received the University of Waterloo Sanford Fleming Medal for ranking first in the graduating class. He holds one U.S. patent related to cryptographic hardware.

Vivienne Sze (S'04) received the B.A.Sc. (Hons.) degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. degree from the Massachusetts Institute of Technology (MIT), Cambridge, in 2006, where she is currently a doctoral candidate.

From May 2007 to August 2007, she worked in the DSP Solutions Research and Development Center at Texas Instruments, Dallas, TX, designing low-power video coding algorithms. From May 2002 to August 2003, she worked at Snowbush Microelectronics, Toronto, ON, Canada, as an IC Design Engineer. Her research interests include low-power circuit and system design, and low-power algorithms for video compression. Her work has resulted in several contributions to the ITU-T Video Coding Experts Group (VCEG) for the next generation video coding standard.

Ms. Sze was a recipient of the 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2008 A-SSCC Outstanding Design Award. She received the Julie Payette - Natural Sciences and Engineering Research Council of Canada (NSERC) Research Scholarship in 2004, the NSERC Postgraduate Scholarship in 2007, and the Texas Instruments Graduate Woman's Fellowship for Leadership in Microelectronics in 2008. She was awarded the University of Toronto Adel S. Sedra Gold Medal and W.S. Wilson Medal in 2004.

Anantha P. Chandrakasan (S'87–M'95–SM'01–F'04) received the B.S, M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been with the Massachusetts Institute of Technology, Cambridge, where he is currently the Joseph F. and Nancy P. Keithley Professor of Electrical Engineering. His research interests include low power digital integrated circuit design, wireless microsensors, ultrawideband radios, and emerging technologies. He is a co-author of Low Power Digital CMOS Design (Kluwer Academic Publishers, 1995), Digital Integrated Circuits (Pearson Prentice-Hall, 2003, 2nd edition), and Sub-threshold Design for Ultralow Power Systems (Springer 2006). He is also a co-editor of Low Power CMOS Design (IEEE Press, 1998), Design of High-Performance Microprocessor Circuits (IEEE Press, 2000), and Leakage in Nanometer CMOS Technologies (Springer, 2005).

Dr. Chandrakasan was a co-recipient of several awards including the 1993 IEEE Communications Society's Best Tutorial Paper Award, the IEEE Electron Devices Society's 1997 Paul Rappaport Award for the Best Paper in an EDS publication during 1997, the 1999 DAC Design Contest Award, the 2004 DAC/ISSCC Student Design Contest Award, the 2007 ISSCC Beatrice Winner Award for Editorial Excellence and the 2007 ISSCC Jack Kilby Award for Outstanding Student Paper. He has served as a technical program co-chair for the 1997 International Symposium on Low Power Electronics and Design (ISLPED), VLSI Design '98, and the 1998 IEEE Workshop on Signal Processing Systems. He was the Signal Processing Sub-committee Chair for ISSCC 1999-2001, the Program Vice-Chair for ISSCC 2002, the Program Chair for ISSCC 2003, and the Technology Directions Sub-committee Chair for ISSCC 2004-2009. He was an Associate Editor for the IEEE Journal of Solid-State Circuits <DCL: Note IEEE journal titles are cap/small caps> from 1998 to 2001. He served on SSCS AdCom from 2000 to 2007 and he was the meetings committee chair from 2004 to 2007. He is the Conference Chair for ISSCC 2010. He is the Director of the MIT Microsystems Technology Laboratories.