A 530 Mpixels/s 4096x2160@60fps H.264/AVC High Profile Video Decoder Chip

Dajiang Zhou, *Member, IEEE*, Jinjia Zhou, Xun He, Jiayi Zhu, Ji Kong, Peilin Liu, *Member, IEEE*, and Satoshi Goto, *Fellow, IEEE*

***Abstract*—The increased resolution of Quad Full High Definition (QFHD) offers significantly enhanced visual experience. However, the corresponding huge data throughput of up to 530 Mpixels/s greatly challenges the design of real-time video decoder VLSI with the extensive requirement on both DRAM bandwidth and computational power. In this work, a lossless frame recompression technique and a partial MB reordering scheme are proposed to save the DRAM access of a QFHD video decoder chip. Besides, pipelining and parallelization techniques such as NAL/slice-parallel entropy decoding are implemented to efficiently enhance its computational power. The chip supporting H.264/AVC high profile is fabricated in 90 nm CMOS and verified. It delivers a maximum throughput of 4096x2160@60fps, which is at least 4.3 times higher than the state-of-the-art. DRAM bandwidth requirement is reduced by typically 51%, which fits the design into a 64-bit LPDDR SDRAM interface and results in 58% DRAM power saving. Meanwhile, the core energy is saved by 54% by pipelining and parallelization.**

***Index Terms*—DRAM bandwidth, embedded compression, frame recompression, H.264/AVC, QFHD, ultra high definition, video decoder.**

I. Introduction

WHILE 1080p HD has already become a current standard for various video applications including TV broadcasting and home entertainment, even higher specifications such as the 4Kx2K Quad Full High Definition (QFHD) format, which delivers at least four times the data throughput of HD, have been targeted by next-generation applications. To store and transmit these mass video contents, video encoding and decoding technologies are indispensable. Compared with the previous coding standards, H.264/AVC [1], which provides over two times higher compression ratio with better video quality, is a promising tool for compressing these massive data. With the enhanced coding efficiency, the complexity of H.264/AVC is also much higher than that of the previous standards. As a result, the huge data throughput along with the high algorithm complexity greatly challenges the design of QFHD video decoder VLSI with the extensive requirement on both memory bandwidth and computational power.

Thanks to the continuous advancement of VLSI technology, it is now possible to integrate an ever-increasing number of transistors into one chip, which provides much stronger computational power. However, the off-chip memory (DRAM) bandwidth is still critically limited to the number and physical design issues of I/O pins. Recently, design techniques such as VBSMC [2], 2-D MC cache [3] and optimized DRAM controller [4] and [5] have been applied in video decoder chips, and contribute to over 50% DRAM bandwidth reduction. Even with these optimizations, real-time H.264/AVC high-profile decoding for 1080p@60fps requires a DRAM configuration of 32-bit DDR-400 (1.6 GB/s) [6]. For QFHD 3840x2160@60fps or 4096x2160@60fps, the required DRAM bandwidth increases by at least 4 to 4.3 times proportionally to the throughput, to be near 7 GB/s, which may require a DRAM configuration beyond 64-bit DDR2-800. Considering a complete video SoC [7] will integrate other bandwidth-hungry components in addition to the video decoder, such as video post-processing and display, the total bandwidth requirement can hardly be handled even with the fastest DDR3 SDRAM. Therefore, in designing QFHD video decoder systems, the off-chip DRAM bandwidth should be regarded as a primary performance bottleneck.

Besides, DRAM traffic dominates the power consumption of video decoder systems, since the power dissipated for DRAM access is usually several times larger than that for the video decoder chip core. Furthermore, the number of pins for DRAM connection, which composes a large portion in a video decoder chip's total pin count, has a significant impact on both the wafer and package cost of the chip. Therefore, the reduction of DRAM bandwidth can also contribute to fabrication cost saving.

Most of the recent high-throughput video decoder chips [6], [8]–[10] were targeted for 1080p applications. These chips requires 100 to 120 MHz to perform 1080p@30fps decoding, which means it is infeasible to simply scale up their clock frequencies by over 8 times to achieve QFHD decoding.

In this paper, design of a 4096x2160@60fps video decoder chip for H.264/AVC high profile is presented. To solve the DRAM bandwidth problem, two techniques, partial MB reordering (PMBR) and variable-compression-ratio lossless frame recompression (VCR-LFRC), are proposed. The PMBR scheme allows part of the decoder pipeline to work in a modified MB scanning order, which saves the reference frame read and line buffer access bandwidth. The VCR-LFRC architecture compresses, locates and restores the frames stored

Manuscript received August 23, 2010; revised November 14, 2010; accepted December 18, 2010. Date of publication March 10, 2011; date of current version March 25, 2011. This paper was approved by Guest Editor Makoto Nagata. This research was supported in part by the Knowledge Cluster Initiative (2nd Stage) and Waseda University Ambient SoC Global COE Program of MEXT, Japan, and by the JST CREST Project. The work of D. Zhou was supported by the Japan Society of the Promotion of Science.

D. Zhou, J. Zhou, X. He, and S. Goto are with the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan (e-mail: zhou@fuji.waseda.jp).

J. Zhu, J. Kong, and P. Liu are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China.

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.2011.2109550

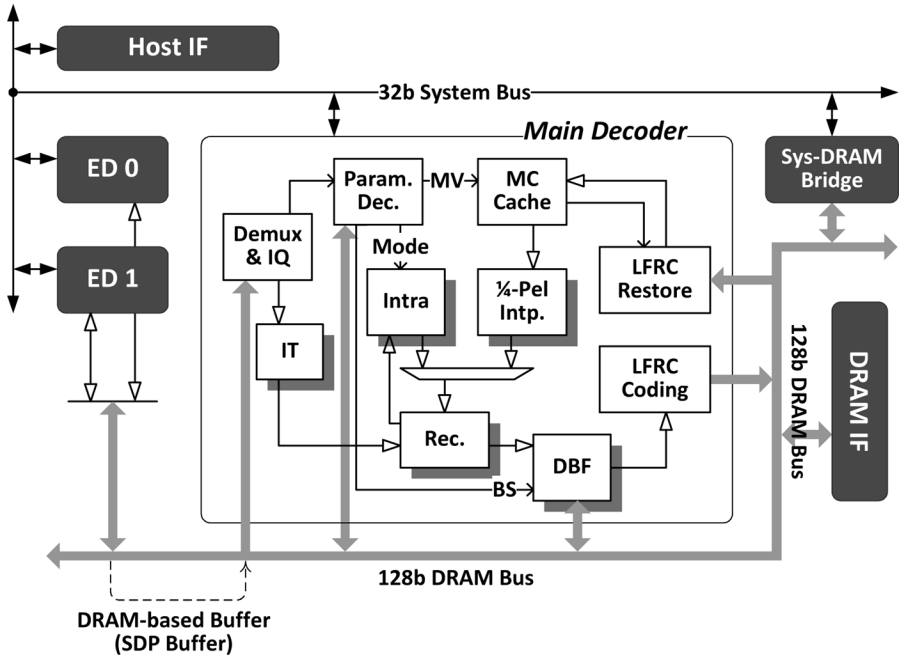


Fig. 1. QFHD H.264/AVC video decoder chip block diagram.

in the DRAM, and thereby reduces the bandwidth for frame data access. Moreover, various pipelining and parallelization techniques for decoder core optimization are also presented.

The rest of this paper is organized as follows. Section II gives an overview of the proposed video decoder system. Sections III and IV present the DRAM bandwidth optimization techniques of PMBR and VCR-LFRC respectively. Section V presents the pipelining and parallelization techniques for decoder core optimization. Section VI shows the chip implementation results. Finally, the conclusion is given in Section VII.

II. System Overview

Fig. 1 shows the block diagram of the chip. The video decoder receives configurations and bit streams from the external host processor through a host interface which is connected to various components on a 32-bit system bus. On the other side, a 128-bit bus is dedicated for DRAM access and is connected to the external memory chips through a DRAM interface optimized for video applications [4]. The two bus systems are linked by a system-DRAM bridge. Additional components for a complete video SoC, such like post-processing and display, can also be added into this architecture by separating configuration and DRAM access on these two bus channels.

For decoding a bit stream, it is first loaded into the entropy decoder engines (EDs) for CABAC/CAVLC decoding. The results are then packaged and stored into a buffer system established in the DRAM (SDP buffer) for use by the main decoder, which contains the computational components subsequent to entropy decoding. The buffer system is designed for three purposes: 1) to balance the variation of decoding speed between EDs (fast for P/B, but slow for I frames) and the main decoder (fast for I, but slow for P/B frames), 2) to facilitate PMBR (Section III), and 3) to facilitate NAL/slice-parallel entropy decoding (Section V-A).

The main decoder pipeline processes each MB in ideally 64 clock cycles. For DRAM bandwidth saving, the main decoder works in a reordered MB scanning sequence (Section III), while coding and restoring components are added after the deblocking filter (DBF) and prior the the MC cache respectively, for lossless frame recompression (Section IV).

III. Partial MB Reordering

To reduce the MC reference frame read (RFR) bandwidth that composes the largest portion of the total DRAM bandwidth of a video decoder, an effective approach is to reuse the overlapped data read from the reference frames. In H.264/AVC, the use of variable-block-size inter prediction usually results in a much larger reference block than the current block, which significantly increases the overlapping of reference samples. The previous works on 2-D MC cache [3], [5] contributed to near 80% RFR bandwidth reduction from non-optimization, or 60% reduction from VBSMC [2], by efficiently reusing the overlapped reference samples inside each MB and between horizontally neighboring MBs (horizontal reuse), as shown in Fig. 2(a). However, under the standard line-by-line raster MB scanning order, the cache is incapable of reusing the overlapped reference samples between vertically adjacent MBs (vertical reuse), unless a whole row of reference blocks is buffered into the cache. For QFHD decoding with biprediction, the data memory requirement for vertical reuse can be estimated to be at least 192KB (16x4Kx1.5x2), while the total cache memory requirement can be even larger if the tag memory and the influence of multiple reference frames are taken into consideration. Consequently, the use of this memory may increase the total area of a decoder by several times.

One method to reach a balance between vertical reuse and cache memory size is to use a modified MB scanning order. In [11], Chen *et al.* discussed an MB reordering method for the

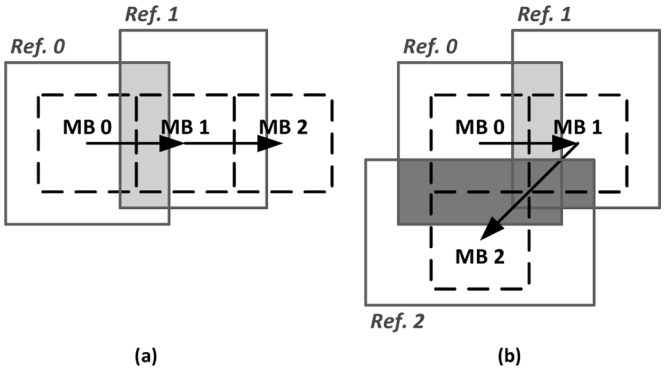


Fig. 2. Comparison between the raster and zig-zag MB scanning orders (a) Raster MB scanning: horizontal reuse only. (b) Zig-zag MB scanning: horizontal and vertical reuse.

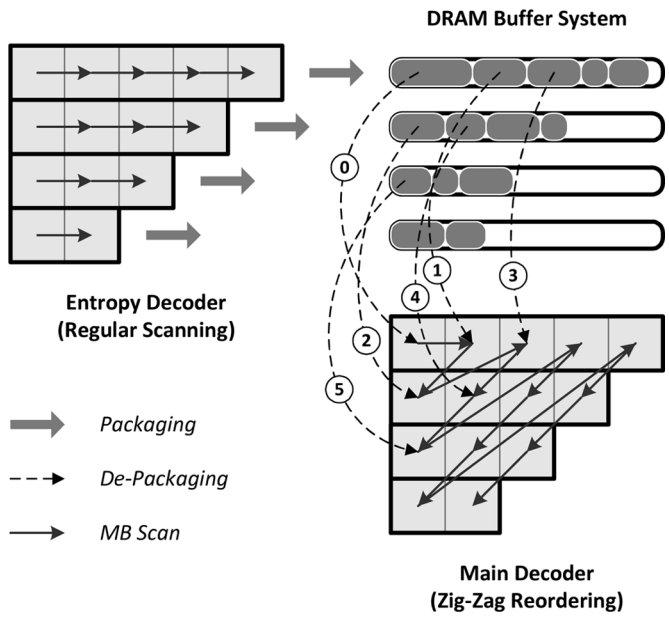


Fig. 3. Partial MB reordering.

video encoder to reuse vertically neighboring search windows. This method is based on an MB-pipelining model, in which the processing orders of all the pipeline stages including motion estimation and entropy coding are modified. This model works for MPEG-4 or H.264/AVC (CAVLC only) encoding, but it is hard to apply it to a video decoder because of the bit-to-bit dependency of entropy decoding that has to follow the same MB scanning order as the bit stream has been encoded.

To solve this problem, a partial MB reordering (PMBR) scheme is proposed to enable the decoding pipeline to work in two different MB scanning orders simultaneously. As shown in Fig. 1, the decoder pipeline is divided into the entropy decoder engines (EDs), and the main decoder that includes the computational components subsequent to entropy decoding. Fig. 3 shows the MB scanning orders for these two parts. The EDs follow the regular raster decoding sequence to conform to the standard. Meanwhile, the main decoder is connected to EDs via the SDP (slice data package) buffer established in the DRAM. The outputs of each ED are stored into 4 independent FIFO-like SDP sub-buffers according to the mod-4-value of the MB's row number (vertical address), and the main decoder can thereby select its input across different rows. Consequently, MB scanning of the main decoder pipeline can be reordered to be a zig-zag sequence performed inside each 4-row-of-MB region of a picture, so that for each MB, both its horizontally and vertically adjacent MBs either were recently, or will soon be processed, for vertical reuse (Fig. 2(b)) with a reasonable cache size. The zig-zag sequence also ensures that the neighboring MBs (left, upper-left, upper and upper-right) required for intra and MV decoding are processed prior to the current MB, for conformance to the standard. By applying PMBR, around 20% of the RFR bandwidth can be saved.

Besides vertical reuse, PMBR also contributes to saving the line buffer read/write (LBRW) bandwidth for buffering the last-4-line pixels required by the deblocking filter. After PMBR is applied, the last-4-line information of the upper three rows of MBs in each 4-row-of-MB region will soon be consumed by the deblocking filter, instead of waiting until a whole row of MBs is processed. Therefore, 3/4 of the LBRW access can be bypassed via a small on-chip memory.

The number (N) of MB rows across which the zig-zag reordering is performed can also be replaced with numbers other than 4. As shown in Table I, the larger N is, the larger portion of the vertically overlapped reference samples can be reused, and the more LBRW access can be bypassed. In the meanwhile, with a larger N, the required cache size to achieve optimum results increases drastically. In this work, $N = 4$ is selected as a balance.

TABLE I
IMPACT OF NUMBER (N) OF MB ROWS ACROSS WHICH THE ZIG-ZAG REORDERING IS PERFORMED

N =	1[1)]	2	4	8
Cache size per reference list	2x2 MBs	2x4 MBs	4x6 MBs	8x10 MBs
Data memory size[2)]	3KB	6KB	18KB	60KB
RFR vertical reuse ratio	0%	50%	75%	87.5%
LBRW reduction ratio	0%	50%	75%	87.5%

[1)] No reordering.
[2)] Containing both luma and chroma samples, and for two reference lists.

The data stored in the SDP buffer are the decoded MB-layer syntax elements (listed in 7.3.5 of [1]) such as MB types, intra prediction modes, reference indices, motion vector differences, and residuals. The syntax elements (SEs) are packaged using variable length coding before stored into the buffer, and de-packaged by the Demux component of the main decoder. Specifically for the residuals, the run/level values of each non-zero coefficient, instead of the original residual SEs specified by CAVLC or CABAC, are transmitted for complexity reduction.

For each slice, after the entropy decoding process is launched, the start addresses of the 4 SDP sub-buffers are returned from the ED to the host processor. These addresses are then used by the processor to configure the main decoder, for the latter to identify the start position of SDP read.

For frames partitioned into multiple slices, it is possible that the MBs in a 4-row-of-MB region belong to various slices. In this case, the MB scanning of the main decoder still follows the zig-zag order, but the order only applies to the MBs inside

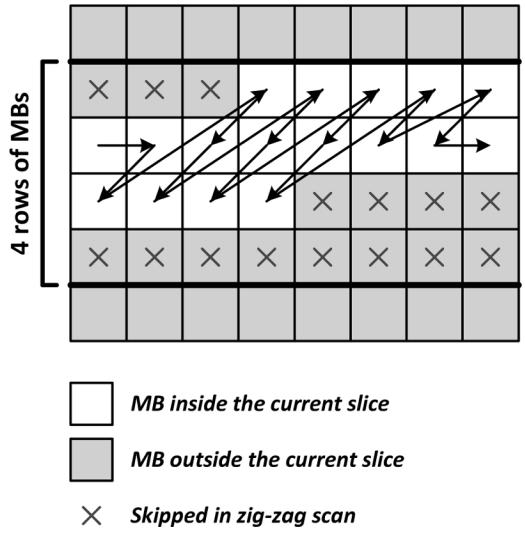


Fig. 4. Zig-zag reordering for frames partitioned into multiple slices.

the current slice, while those outside the current slice should be skipped, as shown in Fig. 4.

IV. VCR Lossless Frame Recompression

The basic idea of frame recompression (FRC) is to compress the reference frames before storing them into the DRAM, and then decompress the required data after fetching them back for motion compensation, so as to save the DRAM traffic. The design of FRC architectures is challenged by several critical constraints. 1) Random access must be supported within the compressed frames, otherwise extra memory traffic will be incurred because every time only a certain part of the frame data is needed. 2) DRAM access is usually with a long latency therefore the algorithms with high data dependency must be carefully used. 3) Real-time video decoding, especially for the QFHD application, requires high-speed compression and decompression circuits with relatively low power and area overhead for FRC.

As a result, most of the previous works [12]–[16] on this topic adopted a fixed-compression-ratio (FCR) model, which is to divide the original frames into small partitions and compress each of them to be equally sized. Although this model provides a straightforward approach to support random access, its shortcomings are also apparent. One one hand, some partitions may have a higher potential for compression, whereas can only be compressed in the designated but relatively lower compression ratio, which leads to the degradation of compression efficiency. On the other hand, image quality loss is inevitable for partitions incapable for fitting into the designated compression ratio in a lossless manner, while the consequent drift error from the quality loss of reference frames can be an even more critical problem.

In [17], a frame recompression architecture that can support lossless mode for the video encoder was discussed. But the large partition size (16x16 pixels) applied for this architecture makes it inefficient for a video decoder, which has much more complex frame access patterns than the video encoder. Furthermore, its low processing speed (CIF@30fps/10 MHz) limits its applications in high-throughput systems such as for QFHD.

In this work, a variable-compression-ratio (VCR) lossless frame recompression (LFRC) scheme is applied, which can achieve considerable DRAM bandwidth reduction for the frame write (FW) and reference frame read (RFR) traffic. To support random access with VCR storage, a memory mapping method and the corresponding architecture framework are proposed and presented in Section IV-A. To achieve the requirement of high throughput, a DPCM-based semi-fixed-length coding method with reduced algorithm dependency is proposed for compressing and decompressing the frame data, which is presented in Section IV-B. Section IV-C discusses the video output problem with VCR-LFRC.

A. Memory Mapping for VCR-LFRC

Figs. 5 and 6 show the two levels of the memory mapping for supporting random access with VCR, which is similar to that in [18] and [19]. A *partition* is a basic unit for compression, consisting of an 8x4 luma block and two corresponding 4x2 chroma blocks under 4:2:0 sampling. Without compression, a partition requires 384 bits for storage. As shown in Fig. 5(a), a frame is divided into *groups*, each of which contains 4 vertically adjacent partitions.

After compression, the mapping of groups keeps unchanged so that each partition can be randomly located to the group level. Inside one group, partitions are compressed and organized compactly with part of the group area left blank, as shown in Fig. 5(b). The blank space decreases the opportunity for consecutive vertical access which leads to extra scheduling overhead of the DRAM controller. Therefore, as shown in Fig. 5(c), for each pair of vertically adjacent groups, the data in the upper and lower groups stick to the same end, so as to connect with each other. Moreover, to comply with the data flow of H.264/AVC's deblocking filter, the groups are designed to be with a 4-pixel vertical offset to the MBs.

To fetch a compressed partition without useless DRAM access, its length (size in bits) and start address inside a group are also required, as shown in Fig. 6. Therefore, the lengths of partitions are also stored into the DRAM, so that two steps for fetching lengths and compressed partitions can be processed in a pipeline to solve the latency problem. The start addresses of all the partitions inside a group can be derived by accumulating the lengths.

As shown in Fig. 7(a), for the first three partitions in each group, the corresponding lengths (L0, L1 and L2) are recorded in 9 bits for each to cover the range from 0 to 384 bits. If the partition size after compression is equal to or larger than 384 bits, the length will be specified as 384 and the original representation for this partition, instead of the compressed one, will be stored. For the length of the last partition (L3), only the number of remaining words (0 to 3, 128 bits for each), instead of the detailed size in bits, is needed to be recorded in 2 bits. Therefore, the four partition lengths of one group can be packed into 32 bits (including three reserved bits). As shown in Fig. 7(b),

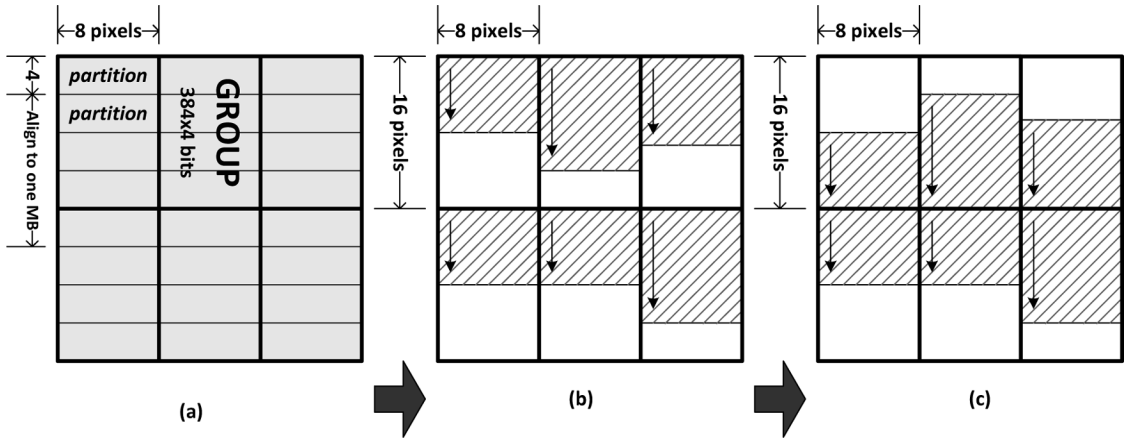


Fig. 5. Group level memory mapping for supporting random access with VCR. (a) Original full mapping. (b) Mapping of compressed groups. (c) Symmetric mapping of compressed groups.

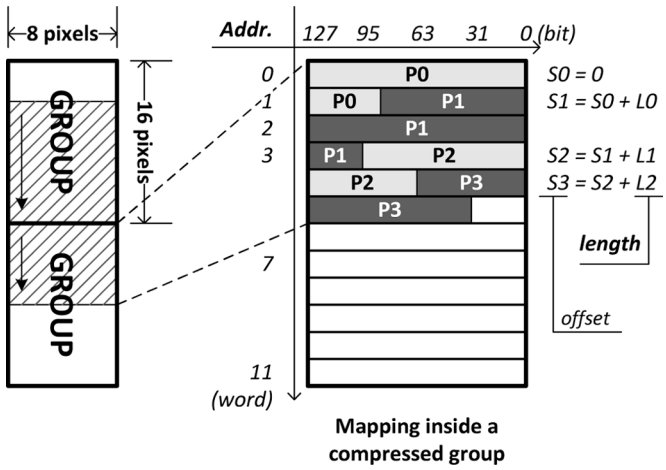


Fig. 6. Partition level memory mapping for supporting random access with VCR.

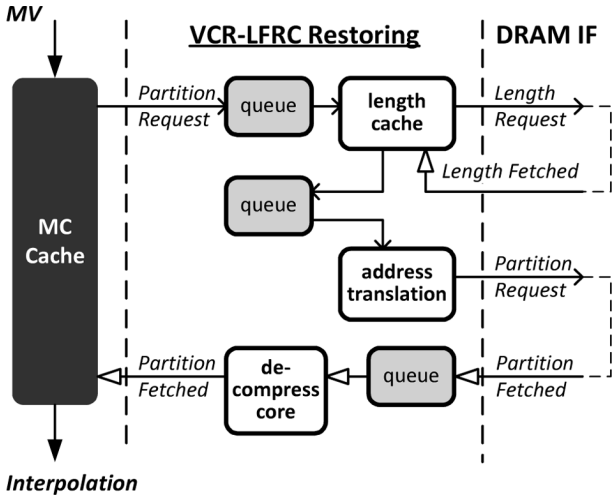


Fig. 8. Block diagram of VCR-LFRC restoring component.

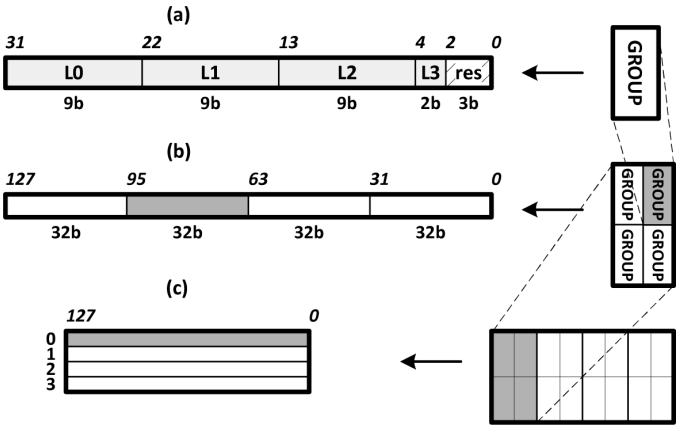


Fig. 7. (a) Bit allocation, (b) word organization, and (c) cache line organization, for partition lengths. (a) Lengths of 1 group. (b) One word: lengths of 4 groups. (c) One cache line: 4 words.

the lengths of a 2x2 array of groups are further packed into a 128-bit word for storage in the DRAM.

To reduce the DRAM bandwidth overhead for loading lengths, a dedicated length cache is implemented. As shown in Fig. 7(c), each cache line contains 4 words or the lengths of a 8x2 array of groups. The length cache is fully associative and follows the FIFO replacement policy to buffer 32 latest loaded cache lines, with a total size of 2KB.

Fig. 8 shows the block diagram of the VCR-LFRC restoring component inserted transparently between the MC cache and the DRAM interface. The request for partitions from the MC cache is first checked by the length cache unit. If the lengths of the requested partitions are missing, the length cache issues DRAM request to fetch them. The lengths are then used by an address translation unit to transfer the partition request into the compressed domain, and issued to the DRAM interface. Finally, the fetched partitions are restored by the decompress core unit, and sent back to the MC cache. To conceal the DRAM access latency, FIFO queues are inserted between the units for pipelining.

B. DPCM-Based Semi-Fixed-Length Coding

For 4096x2160@60fps decoding at 175 MHz, the required throughput of the VCR-LFRC compression unit should be at least 3 pixels/cycle (4096x2160x60/175M). For B frames, the throughput requirement for the decompression unit, which depends on the data traffic from the DRAM to MC cache, can

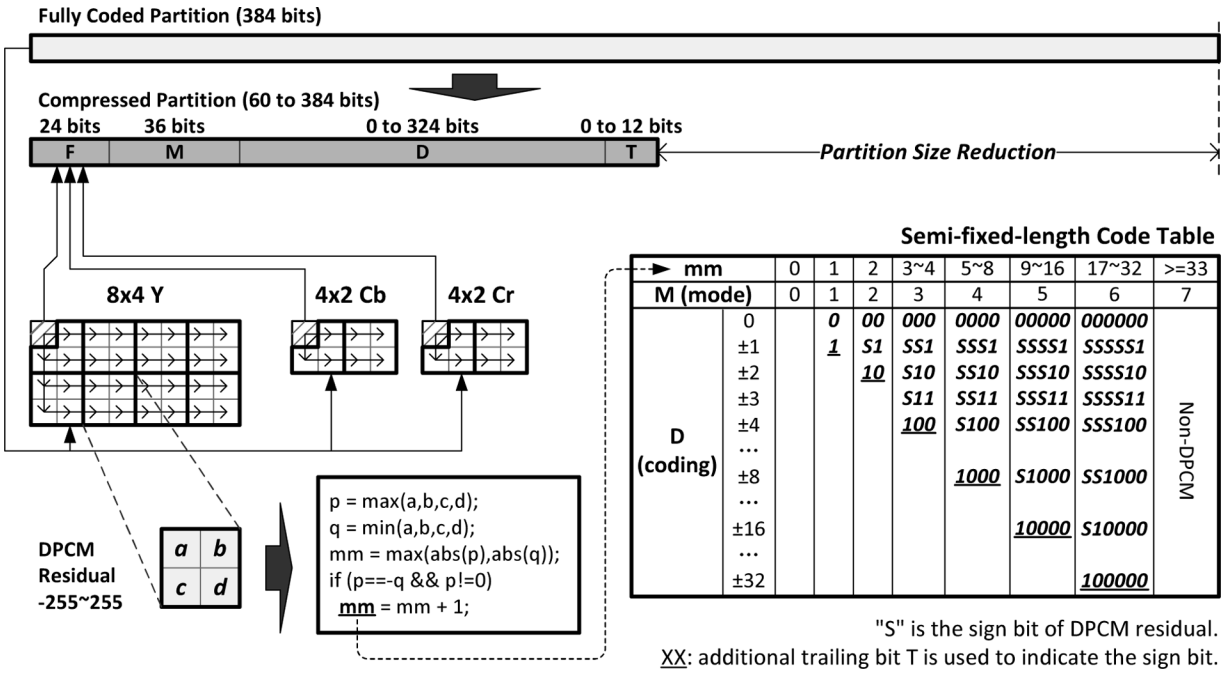


Fig. 9. DPCM-based semi-fixed-length coding algorithm.

be more than two times higher than that for compression. Further taking into consideration the usually uneven distribution of the decompression workload, a high parallelism of around 10 pixels/cycle, or 15 samples/cycle under 4:2:0 sampling, is required to ensure real-time processing.

Therefore, an efficient coding algorithm is needed to support architecture design. Variable length coding (VLC) is a widely used method for low-cost lossless data coding. But the VLC algorithms are usually with very high bit-to-bit and word-to-word data dependency, which results in either a long critical path of the arithmetic circuits, or a multi-cycle pipeline filled with bubbles.

To solve this problem, a DPCM-based semi-fixed-length coding algorithm is proposed, as shown in Fig. 9. For each partition, DPCM scan is first performed on the Y, Cb and Cr blocks of the partition to reduce spatial redundancy. The DPCM residuals are then divided into 2x2 sub-blocks and a coding mode (M) for each sub-block is decided according to its maximum and minimum residual values. Based on M, residuals inside sub-blocks are coded (to be D) according to the semi-fixed-length code table, except that for each of the Y, Cb and Cr blocks the top-left sample keeps its original 8-bit representation (F) to serve as a starting point for DPCM scan. For M between 2 and 6, the values representable by D can either be within $[-2^{M-2}, 2^{M-1}]$ or $[-2^{M-1}, 2^{M-2}]$. Therefore, for each sub-block, if any of the residuals equal to 2^{M-1} or -2^{M-1}, a trailing bit (T) will be added to denote the sign of the specific residual(s). The bits in F, M, D and T are assembled to be the finally compressed representation of the partition.

For each coded partition, the bit lengths of F and M are fixed. Then the length of D for each sub-block is fixed with M given. The length of T can be derived by subtracting the partition length to the sum of F, M and D. As a result, the extensive dependency in conventional VLC such as Exp-Golomb and Unary can be replaced by the slight dependency between M and D, so that the parallelism of the decompression unit can be significantly improved. Moreover, the proposed algorithm also outperforms VLC on coding efficiency by utilizing the locality of DPCM residuals, instead of coding every residual in a universal form.

Fig. 10 shows the 5-stage pipelined architecture for VCR-LFRC decompress core. In stage 1, the input data are latched to three registers for the F (DPCM start points), M (coding modes), and $D\&T$ (coded residuals and trailing bits) parts of the compressed partition. In stage 2, a cascaded tree of small barrel shifters (BS) splits the compactly stored coded residuals (D) for a 4x4 block to the 2x2 sub-block level, which is further split and decoded to independent residuals in stage 3. In stage 4, trailing bits (T) are decoded and used to determine the signs of the residuals. Finally, DPCM inverse scan is performed in stage 5 to generate the complete samples. Owing to the reduced algorithm dependency of semi-fixed-length coding, the pipeline achieves a parallelism of 16 samples, or 10.7 pixels per cycle, which can meet the requirement of our QFHD video decoder.

Fig. 11 shows the the DRAM bandwidth reduction results by applying VCR-LFRC. According to various sequence features and QP values, bandwidth reduction for frame write ranges from 43% to 65%, while reference frame read bandwidth is saved by 37% to 62%.

C. Video Output With VCR-LFRC

For a complete video system, it is meaningful to consider how the decoded video data are output. This can be difficult because video display usually follows a line-by-line pixel flow while the proposed VCR-LFRC compresses the frames based on partitions which are 2-D blocks. This problem is considered according to two types of organizations of the video system.

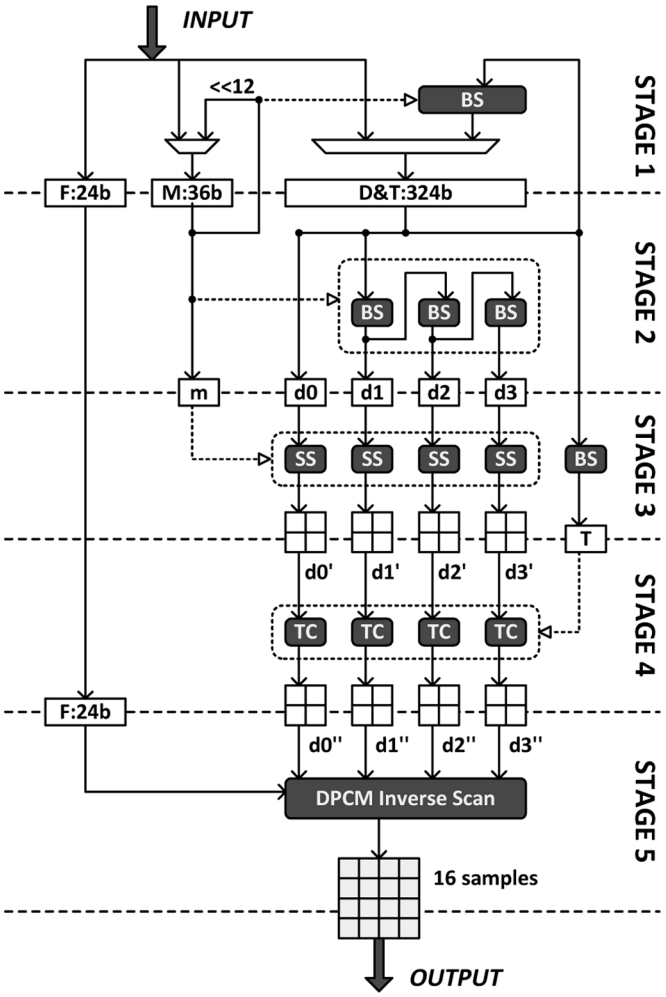


Fig. 10. 5-stage pipelined architecture for VCR-LFRC decompress core. BS denotes "barrel shifter". SS denotes "sub-block splitting". TC denotes "trailing bit compensation".

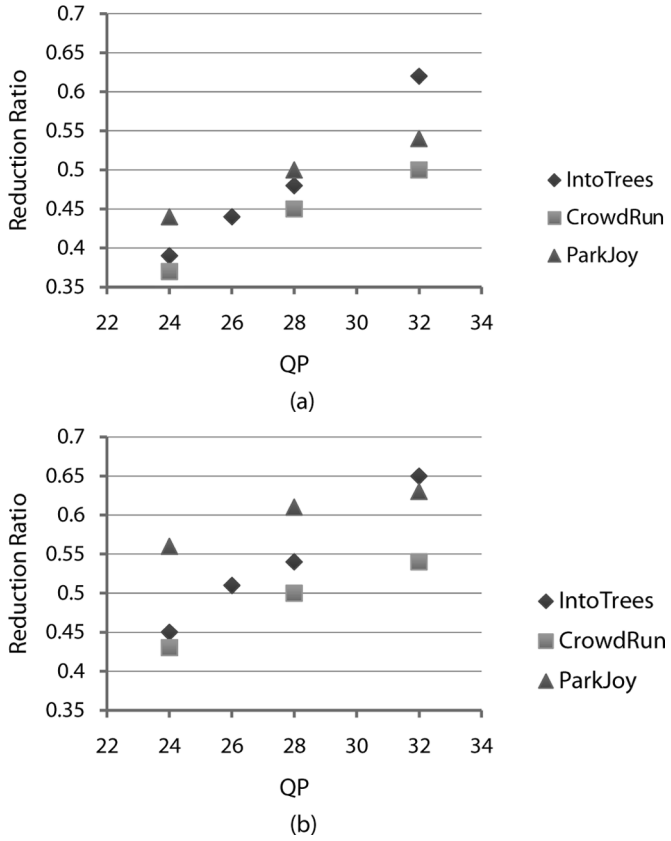


Fig. 11. DRAM bandwidth reduction for reference frame read (RFR) and frame write (FW) by applying VCR-LFRC. (a) RFR bandwidth reduction. (b) FW bandwidth reduction.

In the type 1 organization, two separate buffers are used to store the reference frames and display frames, respectively. In this case, the proposed VCR-LFRC method can be applied to the reference frame buffer. Meanwhile, a line-based frame recompression method such as [20] that fits the line-by-line flow of video display, can be suitable for reducing the DRAM bandwidth of the display frame buffer.

In the type 2 organization, the decoded frames are not directly displayed. Instead, post-processing units such as the video scaler, out-loop deblocking filter and image enhancer read the decoded frames from the DRAM and store back the processing results for display. In this case, the data input flow of these units can be partition-by-partition, so that the same decompress unit as designed for motion compensation can be used before post-processing. Then, the data output flow of the post-processing units can be organized to be a line-by-line compatible format, either with or without line-based frame recompression, for display.

V. Pipelining and Parallelization

This section presents the pipelining and parallelization techniques to meet the computational power requirement of the QFHD video decoder. In Section V-A, a NAL/slice-parallel entropy decoding scheme is discussed. Section V-B presents the parallelization of the main decoder.

A. NAL/Slice-Parallel Entropy Decoding

CABAC is the performance bottleneck of entropy decoding. Owing to the high bit-to-bit algorithm dependency, most of the previous CABAC decoder architectures can process only one regular bin in each cycle, which are unable to meet the requirement for QFHD applications. In [21], a multi-bin architecture based on branch selection was proposed, but at the cost of a long critical path and an area increase faster than the performance gain. Some other contributions [22]–[24] focused on low-dependency CABAC algorithms. Due to their modifications on the decoding specification, however, design for the current standard can not yet benefit from these progresses.

Therefore, a scheme for parallelizing multiple independent entropy decoder engines (EDs) is considered. Usually for multi-core video encoders, a frame is divided into multiple slices and in each slice one thread of entropy coding can be performed, so as to achieve a high overall performance. Nevertheless, this method can not be applied to this work, since a generic video decoder should give support to various configurations, instead of assuming whether or how the frames are partitioned to slices.

For the proposed scheme, parallelism of entropy decoding is achieved by utilizing the NAL (network abstraction layer) structure of H.264/AVC. As shown in Fig. 12, prior to entropy de-

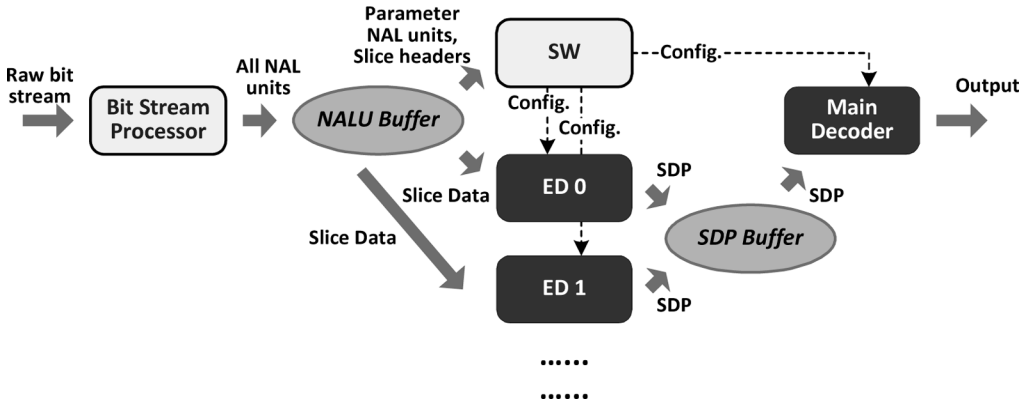


Fig. 12. The framework for NAL/slice-parallel entropy decoding.

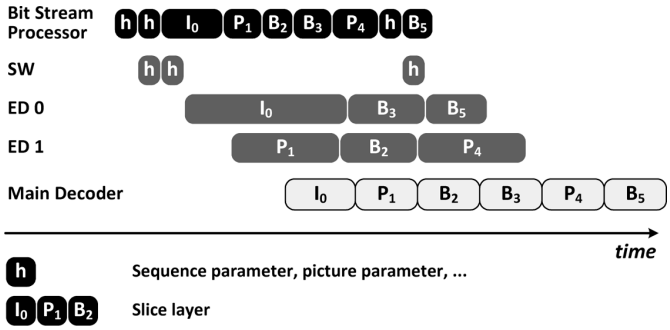


Fig. 13. An example of the NAL/slice-parallel processing schedule.

coding, the source bit stream first goes through a bit stream processor (BSP). BSP divides the stream into NAL units (NALU) by searching the NAL start code with a speed of one byte per cycle. The results are then written into an NALU buffer. The NALUs are classified into two categories including parameter NALUs and slice NALUs. The parameter NALUs containing the sequence- or picture-level parameters, together with the slice header part of the slice NALUs, are processed by the software (SW) built on the host processor, which generates the configurations for EDs and the main decoder. The slice data part of slice NALUs is processed by the multiple EDs, and full parallelism can be achieved when more than N-1 slice NALUs have been stored in the NALU buffer, where N equals to the number of EDs. The results of EDs are packaged and written into an SDP (slice data package) buffer for use by the main decoder.

Fig. 13 shows an example of the NAL/slice-parallel processing schedule, where two EDs are parallelized. ED0 and ED1 can start processing I_0 and P_1, when only part of the corresponding NALUs are ready in the NALU buffer. I_0 is larger in size than P_1. Therefore, even if ED1 is launched posterior to ED0, the former can complete the first slice prior to the latter. In this case, ED1 will start processing the next slice B_2, instead of waiting for ED0, to avoid pipeline bubbles. The main decoder always sticks to the standard slice order, and can also start processing when only part of the results of EDs are ready in the SDP buffer.

In this design, two EDs are parallelized to obtain an overall entropy decoding throughput of over 240 Mbit/s at 175 MHz for 4096x2160@60fps, while even higher throughput can be achieved by increasing the clock frequency.

B. Main Decoder Parallelization

The main decoder subsequent to EDs, as shown in Fig. 1, processes each MB in ideally 64 clock cycles. The IT, Parameter Decoder, Reconstruct, Deblocking Filter (DBF) and LFRC Coding components deliver a constant throughput for each MB in 64 cycles. A 4x4-block-based processing flow is applied for these components, which outperforms the conventional MB-based flow by eliminating the stall cycles across MB boundaries. The Demux, IQ, MC Cache, Interpolation, Intra and LFRC Restoring components deliver variable throughput according to the sequence features, with the average processing time for each MB less than 64 cycles. The proposed architecture achieves high area and power efficiency mainly because of two reasons.

Firstly, most components of the main decoder are designed to share the same and fixed throughput of 64 cycles/MB, while the previous designs usually consist of components with different speeds. Considering the system performance is restricted by its slowest component, the fixed-throughput design effectively reduces the idle time of faster components. As shown in Table II, in the proposed main decoder architecture, most components can achieve high pipeline utilization from 85.5% to 93.9%. The utilization is lower for B frames, because the DRAM bandwidth instead of core performance becomes the system bottleneck during B frame decoding. It is also relatively hard for the Interpolation and LFRC Restoring components to achieve high pipeline utilization for average cases, since the data throughput for motion compensation varies significantly according to the sequence features and the architecture should therefore be over-designed for the worst cases.

The high parallelism is the other factor that contributes to improving the core energy efficiency, which is also discussed in a previous work for a 720p H.264/AVC decoder [25]. Compared to other designs which process one MB in around 300 clock cycles, the proposed high-parallelism pipeline requires more silicon area for computational logic. However, with parallelism enhanced, control logic does not usually tend to increase. As a result, in spite of the increase of total area, area and power efficiency can be improved. Besides, in the IT, Intra, Interpolation,

TABLE II
PIPELINE UTILIZATION OF MAJOR COMPONENTS OF THE MAIN DECODER FOR VARIOUS FRAME TYPES (INTOTREES.264, IBBP 10 FRAMES, 166 MHZ).

Component	Cycles/MB	Average pipeline utilization (%)		
		I frame	P frames	B frames
IT	64	91.1	93.4	85.5
Param. Dec.	64	91.1	93.4	85.5
Intra	0∼66	93.9	40.8	29.6
MC Intp.	0∼65	0	25.8	44.4
Rec.	64	91.1	93.4	85.5
DBF	64	91.1	93.4	85.5
LFRC Coding	64	91.1	93.4	85.5
LFRC Restore	*[1)]	0	26.8	49.7

[1)] 3 cycles/partition.

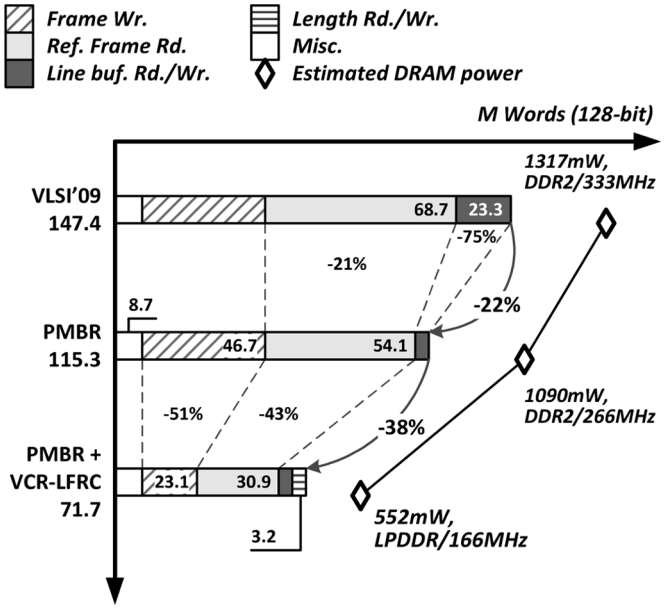


Fig. 14. DRAM bandwidth and DRAM power reduction (IntoTrees.264, IBBP 60 frames).

Reconstruct and DBF components, independent architectures are implemented to process luma and chroma samples in parallel. This results in a 1.5x throughput improvement with slight area increase, since the chroma logic is originally implemented as independent circuits in most of the previous luma-chroma serial designs.

VI. IMPLEMENTATION RESULTS

A. DRAM Bandwidth Saving

Fig. 14 summarizes the DRAM bandwidth reduction results. By applying PMBR, 21% bandwidth for reference frame read and 75% for line buffer read/write can be saved, which consequently contributes to an overall bandwidth reduction of 22%. After applying VCR-LFRC, the bandwidth for reference frame read is further reduced by 43% on the basis of PMBR, while the frame write bandwidth is also reduced by 51%. Due to the use of a length cache, the overhead for length read/write is very small, only 3.2 M words/s. Taking this part into consideration, total DRAM bandwidth decreases by 38% compared to PMBR alone, or 51% compared to only using the optimization techniques in [6] (MC cache and DRAM controller optimization).

The consequent DRAM power reduction is estimated by applying the model in [26], with the configurations shown in Table IV. For [6], the average bandwidth is 147.4 M words/s (one word contains 16 bytes). Considering the DRAM control overhead and the unbalanced distribution of bandwidth requirement, a 64-bit DDR2 SDRAM configuration at no less than 333 MHz is needed to ensure real-time decoding, which dissipates 1317 mW. By using PMBR, the required frequency and power dissipation is reduced to 266 MHz and 1090 mW, respectively. By combining PMBR and VCR-LFRC, the DRAM frequency can be reduced to 166 MHz. Since the frequency is lower than 200 MHz, the system can fit into an LPDDR configuration for further power saving. As a result, while the total DRAM bandwidth is reduced by 51%, DRAM power drops to 555 mW, saved by 58% from [6].

Fig. 15 shows the decoding time reduction by DRAM bandwidth saving. When VCR-LFRC is off (PMBR on), DRAM access is a system bottleneck for B frames, which severely restricts the overall performance. After VCR-LFRC is turned on, decoding time for B frames is saved by an average of 25% and the processing speed for various frame types can be almost balanced. Consequently, the system can easily ensure the time budget for real-time performance.

TABLE III
CHIP SPECIFICATION

Technology	SMIC 90nm G 1P9M CMOS
Supply voltage	1.0V Core, 1.8V/2.5V I/O
Die size	4x4mm^2 (incl. DDR PHY, DLL and PLL)
Package	176-pin LQFP
Logic gate count	662K
On-chip memory	59.6KB
External memory	64-bit LPDDR/DDR2 SDRAM
Max. throughput	530Mpixels/s, 4096x2160@60fps
Core power	189mW@175MHz, 4096x2160@60fps
	176mW@166MHz, 3840x2160@60fps
	48mW@36MHz, 1920x1080@60fps

B. Chip Features

A prototype chip of the QFHD video decoder [27] is fabricated with SMIC 90 nm G 1P9M CMOS technology. Fig. 16 and Table III show the chip micrograph and specification, respectively. A 64-bit LPDDR/DDR2 SDRAM interface is implemented on the chip for connection with the external memory. The chip die size is 4x4 mm^2 including DDR PHY, DDL, PLL, and the decoder core that contains 662K logic gates and 59.6KB on-chip memory. PCB-based est shows the chip is able to work at up to 220 MHz. At 175 MHz, the target throughput of 530 Mpixels/s is achieved. Power dissipation of the decoder core for 4096x2160, 3840x2160 and 1920x1080 at 60fps are 189 mW, 176 mW and 48 mW, respectively.

Fig. 17 shows the breakdown of core power dissipation. The power consumed by the LFRC coding and restoring components

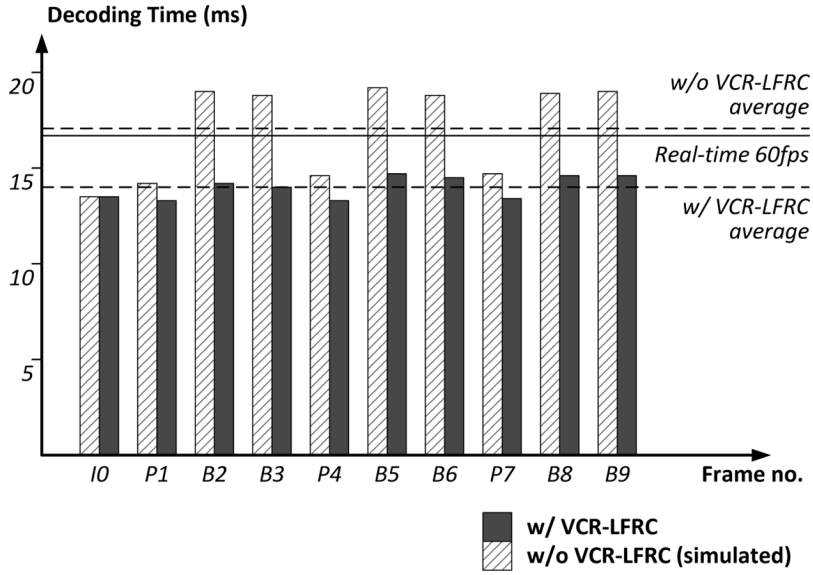


Fig. 15. Decoding performance w/ and w/o VCR-LFRC (IntoTrees.264, IBBP 10 frames, 166 MHz).

TABLE IV
CONFIGURATIONS FOR DRAM POWER ESTIMATION

	VLSI'09 [6]	PMBR	PMBR + VCR-LFRC
Organization	1.8V 16b/256Mb x4		
Type	DDR2	DDR2	LPDDR
Frequency	333MHz	266MHz	166MHz
Total BW (M words)	147.4	115.3	71.7
Read clock cycles %	25%	23%	24%
Write clock cycles %	19%	20%	19%
Page hit rate %	75%	75%	75%
Total DRAM power	1317mW	1090mW	552mW

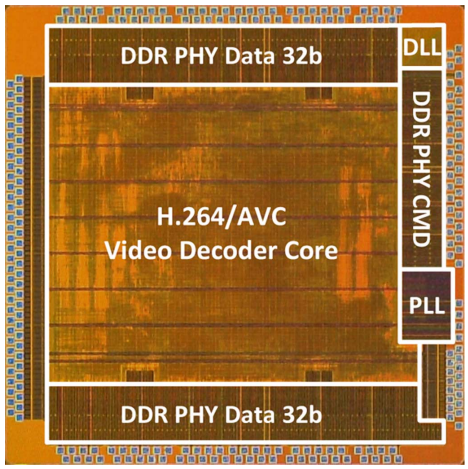


Fig. 16. Chip micrograph.

composes only 10% of the total core power, which is very small compared to the DRAM power reduction by applying VCR-LFRC.

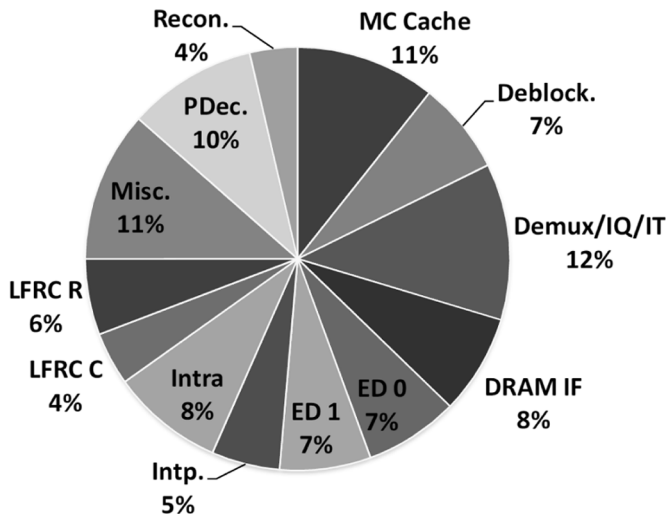


Fig. 17. Breakdown of core power consumption (166 MHz, IBBP frames, post-synthesis simulation).

C. Chip Comparison

Table V shows the comparison between this work and the state-of-the-art. The proposed video decoder delivers a maximum throughput of 530 Mpixels/s for real-time 4096x2160@60fps decoding at 175 MHz, which is 4.3 to 8.5 times faster than the previous chips [2], [6], [10]. With the decoding throughput significantly enhanced, this design keeps a comparative DRAM configuration as the previous works, which is achieved by the proposed DRAM bandwidth reduction techniques of PMBR and VCR-LFRC. The normalized DRAM power is saved by at least 58%, from 2.64 pJ/pixel to 1.11 pJ/pixel. Meanwhile, owing to the efficient parallelization techniques, normalized core energy with technology scaling is reduced by at least 54%, from 0.79 pJ/pixel to 0.36 pJ/pixel. As

TABLE V
COMPARISON WITH THE STATE-OF-THE-ART VIDEO DECODER CHIPS

	This work	VLSI'09 [6]	A-SSCC'08 [10]	JSSC'07 [2]
Video format(s)	H.264 HP	H.264 HP, MPEG-1/2, AVS	H.264 HP, MPEG-2, VC-1	H.264 MP
Max. resolution	4096x2160@60fps	1920x1080@60fps	1920x1080@60fps	1920x1080@30fps
Max. throughput	530Mpixels/s	124Mpixels/s	124Mpixels/s	62Mpixels/s
Clock frequency	175MHz	200MHz	200MHz	120MHz
Logic gate count	662k	367k	515k	160k
On-chip memory	59.6kB	11.0kB	65.3kB	4.5kB
DRAM configuration	64-bit DDR	32-bit DDR	-	32-bit DDR + 32-bit SDR
Technology	90nm/1.0V	0.13μm/1.2V	90nm/1.0V	0.18μm/1.8V
Core power	189mW	257mW	317mW	320mW
Scaled and normalized core power	0.36pJ/pixel	1.0pJ/pixel[1)]	2.56pJ/pixel	0.79pJ/pixel[2)]
Normalized DRAM power[3)]	1.11pJ/pixel	2.65pJ/pixel	-	-

[1)] $\text{Power}_{90} = \text{Power}_{130}/(V_{130}/V_{90})^2/(C_{130}/C_{90}) = \text{Power}_{130}/2.08$.
[2)] $\text{Power}_{90} = \text{Power}_{180}/(V_{180}/V_{90})^2/(C_{180}/C_{90}) = \text{Power}_{180}/6.48$.
[3)] Derived from Fig. 14.

a result, the overall system power efficiency can be more than doubled.

VII. CONCLUSION

This paper discusses the design of an H.264/AVC high profile video decoder in 90 nm CMOS. Its maximum throughput reaches 4096x2160@60fps, or 530 Mpixels at 175 MHz, which is at least 4.3 times faster than previous designs. To reduce the huge DRAM bandwidth requirement, the PMBR and VCR-LFRC techniques are proposed, which typically save bandwidth by 22% and 38% respectively. Therefore, a total bandwidth reduction of 51% is achieved, which leads to 58% saving of DRAM power. Meanwhile, by using various efficient parallelization techniques, the core energy is also saved by 54%, when compared to previous works with technology scaling taken into consideration.

REFERENCES

[1] Joint Video Team, "Advanced video coding for generic audiovisual services," ITU-T Recommendation H.264 & ISO/IEC 14496-10, 2005.

[2] 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 160K 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.

[3] X. Chen, P. Liu, J. Zhu, D. Zhou, and S. Goto, "Block-pipelining cache for motion compensation in high definition H.264/AVC video decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2009, pp. 1069–1072.

[4] J. Zhu, P. Liu, and D. Zhou, "An SDRAM controller optimized for high definition video coding application," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2008, pp. 3518–3521.

[5] T.-D. Chuang, L.-M. Chang, T.-W. Chiu, Y.-H. Chen, and L.-G. Chen, "Bandwidth-efficient cache-based motion compensation architecture with DRAM-friendly data access control," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2009, pp. 2009–2012.

[6] D. Zhou, Z. You, J. Zhu, J. Kong, Y. Hong, X. Chen, X. He, C. Xu, H. Zhang, J. Zhou, N. Deng, P. Liu, and S. Goto, "A 1080p@60fps multi-standard video decoder chip designed for power and cost efficiency in a system perspective," in *Symp. VLSI Circuits Dig. Tech. Papers*, 2009, pp. 262–263.

[7] C.-C. Ju *et al.*, "A multi-format Blu-ray player SoC in 90 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2009, pp. 152–153.

[8] C. Lin, J. Guo, H. Chang, Y. Yang, J. Chen, M. Tsai, and J. Wang, "A 160kgate 4.5KB SRAM H.264 video decoder for HDTV applications," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2006, pp. 1596–1605.

[9] 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 252kgate/71mW multi-standard multi-channel video decoder for high definition video applications," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2007, pp. 282–603.

[10] C.-C. Ju, T.-M. Liu, Y.-C. Chang, C.-M. Wang, H.-M. Lin, S. Cheng, C.-C. Chen, F. Chiu, K.-S. Lin, C.-B. Wu, S. Liang, S.-J. Wang, G. Chen, T. Hsiao, and C.-H. Wang, "A 125Mpixels/sec full-HD MPEG-2/H.264/VC-1 video decoder for Blu-ray applications," in *IEEE Asian Solid-State Circuits Conf. Dig. Tech. Papers*, 2008, pp. 9–12.

[11] C.-Y. Chen, C.-T. Huang, Y.-H. Chen, and L.-G. Chen, "Level C+ data reuse scheme for motion estimation with corresponding coding orders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 4, pp. 553–558, Apr. 2006.

[12] T. Y. Lee, "A new frame recompression algorithm and its hardware design for MPEG-2 video decoders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 6, pp. 529–534, Jun. 2003.

[13] Y. Lee, C.-E. Rhee, and H.-J. Lee, "A new frame recompression algorithm integrated with H.264 video compression," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2007, pp. 1621–1624.

[14] M. Budagavi and M. Zhou, "Video coding using compressed reference frames," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2008, pp. 1165–1168.

[15] Y. Ivanov and D. Moloney, "Reference frame compression using embedded reconstruction patterns for H.264/AVC decoder," in *Proc. Int. Conf. Digital Telecom.*, 2008, pp. 168–173.

[16] Y.-D. Wu, Y. Li, and C.-Y. Lee, "A novel embedded bandwidth-aware frame compressor for mobile video applications," in *Proc. Int. Symp. Intell. Signal Process. Commun. Syst.*, 2009, pp. 1–4.

[17] C.-C. Cheng, P.-C. Tseng, and L.-G. Chen, "Multimode embedded compression codec engine for power-aware video coding system," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 2, pp. 141–150, Feb. 2009.

[18] X. Bao, D. Zhou, and S. Goto, "A lossless frame recompression scheme for reducing DRAM power in video encoding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2010, pp. 677–680.

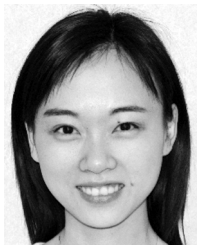
[19] X. Bao, D. Zhou, P. Liu, and S. Goto, "An advanced hierarchical motion estimation scheme with lossless frame recompression for ultra high definition video coding," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2010, pp. 820–825.

[20] M. Uchiyama, K. Oikawa, N. Date, and S. Koto, "A rate-controllable near-lossless data compression IP for HDTV decoder LSI in 65nm CMOS," in *IEEE Asian Solid-State Circuits Conf. Dig. Tech. Papers*, 2009, pp. 201–204.
[21] P.-C. Lin, T.-D. Chuang, and L.-G. Chen, "A branch selection multi-symbol high throughput CABAC decoder architecture for H.264/AVC," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2009, pp. 365–368.
[22] V. Sze, A. Chandrakasan, M. Budagavi, and M. Zhou, "Parallel CABAC for low power video coding," in *Proc. IEEE Int. Conf. Image Process.*, 2008, pp. 2096–2099.
[23] V. Sze and A. Chandrakasan, "A high throughput CABAC algorithm using syntax element partitioning," in *Proc. IEEE Int. Conf. Image Process.*, 2009, pp. 773–776.
[24] J.-H. Lin and K. Parhi, "Parallelization of context-based adaptive binary arithmetic coders," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3702–3711, Oct. 2006.
[25] V. Sze, D. Finchelstein, M. Sinangil, and A. 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.
[26] [Online]. Available: http://www.micron.com/support/part_info/powercalc
[27] D. Zhou, J. Zhou, X. He, J. Kong, J. Zhu, P. Liu, and S. Goto, "A 530 Mpixels/s 4096x2160@60fps H.264/AVC high profile video decoder chip," in *Symp. VLSI Circuits Dig. Tech. Papers*, 2010, pp. 171–172.

Dajiang Zhou (S'08–M'10) received the B.E. and M.E. degrees in electronic engineering from Shanghai Jiao Tong University, China, in 2005 and 2008, respectively. He received the Ph.D. degree from Waseda University, Japan, in 2010. His doctoral thesis focused on the DRAM bandwidth problems with high-throughput video decoder VLSI.

He is currently a researcher at the Graduate School of Information, Production and Systems, Waseda University. His interests are in algorithms and VLSI architectures for multimedia and communication signal processing.

Dr. Zhou received the research fellowship of the Japan Society for the Promotion of Science from 2009–2011. He was a recipient of the design contest award at ISLPED 2010. He has published 30 technical papers and holds eight patents.



Jinjia Zhou received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, China, in 2007, and the M.E. degree from Waseda University, Japan, in 2010, where she is currently a Ph.D. candidate. Her research interests are in algorithms and VLSI architectures for video coding.

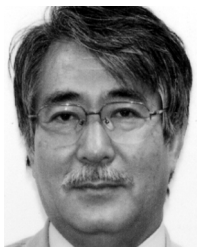
Xun He received the B.E. and M.E. degrees from the Department of Electronic Engineering of Shanghai Jiao Tong University, China, in 2006 and 2009, respectively. He is currently a Ph.D. candidate at the Graduate School of Information, Production and Systems, Waseda University, Japan. His research interests include video coding technologies and low power multicore processor architectures.

Jiayi Zhu received the B.E. degree in electronic engineering in 2006, and the M.E. degree in microelectronics in 2009, both from Shanghai Jiao Tong University, China, where he is currently working toward the Ph.D. degree. His research interests are in algorithms and VLSI architectures for video coding.

Ji Kong received the B.E. degree from the University of Electronic Science and Technology of China, in 2005, and the M.E. degree from Shanghai Jiao Tong University, China, in 2008, where he is currently pursuing the Ph.D. degree. His research interests are in VLSI and computer architecture for multimedia processing.

Peilin Liu (M'07) received the Doctor of Engineering degree from the University of Tokyo, Japan, in 1998, where she worked as a researcher in 1999. From 1999 to 2003, she was a senior researcher at the Central Research Institute of Fujitsu, Tokyo, Japan, where her research topics included multimedia processing, IC design and high-performance processor architecture.

She joined Shanghai Jiao Tong University, China in 2003, and is currently a Professor in the Department of Electronic Engineering. She directs the MediaSoC Lab, and is responsible for a series of projects, including HDTV SoC platform development and high-performance portable audio/video systems, etc.



Satoshi Goto (S'69–M'77–SM'84–F'86) received the B.E. and M.E. degrees in electronics and communication engineering from Waseda University, Japan, in 1968 and 1970, respectively. He received the Doctor of Engineering degree from Waseda University in 1981.

He joined NEC Laboratories in 1970, where he worked on LSI design, multimedia system and software as GM and VP. Since 2003, he has been a Professor in the Graduate School of Information, Production and Systems, Waseda University, Kitakyushu, Japan. His main interests are in VLSI design methodologies for multimedia and mobile applications. He has published seven books and more than 200 technical papers in international journals and conferences.

Dr. Goto served as GC of ICCAD and ASPDAC and was a board member of the IEEE CAS Society. He is a Fellow of IEEE and IEICE, and a member of the Academy Engineering Society of Japan.