A Low-Cost Slice Interleaving DSC Decoder Architecture for Real-Time 8K Video Decoding

Shifu Wu and Bevan Baas
Department of Electrical and Computer Engineering
University of California, Davis
{ucdwu, bbaas}@ucdavis.edu

***Abstract*—High resolution and high frame rate video including 4K and 8K is increasingly popular, however its real-time decoding using H.264 and HEVC is challenging due to its high hardware computational cost and large memory requirement. In contrast, the Display Stream Compression (DSC) video decoder requires much smaller hardware and easily supports very high pixel rates. We present a low-cost DSC decoder utilizing a slice interleaving architecture, as well as four designs that utilize the architecture implemented and synthesized in a 28 nm CMOS standard cell process. The designs are able to perform real-time decoding at frame rates up to 94–107 frames per second (fps) for 8K UHD and 376–430 fps for 4K UHD, both in 4:4:4 mode with a throughput of 3 pixels per clock cycle. The frame rates are doubled in native 4:2:2 and 4:2:0 modes. The designs have gate counts of 161K–282K in minimum-sized NAND2 equivalent gates and main memory of 36.9KB–54.7KB to support 8K UHD.**

I. Introduction

Ultra high definition (UHD) video resolutions of 4K UHD (3840×2160) and 8K UHD (7680×4320) are more prevalent, and higher frame rate is attracting more interests, both of which make real-time video decoding challenging, due to ultra high throughput requirement. H.264/AVC [1] and H.265/HEVC [2] are widely used in video decoding. Though decoders of both standards are able to achieve high decompression rate, the computation complexity is very high and large amounts of on-chip and off-chip memory is required. HEVC is even more complex than H.264, because of more advanced techniques used to increase performance. There have been several high performance video chips of H.264 [3], [4] and HEVC [5], [6] targeting at 4K or 8K resolution, all of which are able to perform real-time video decoding.

Display Stream Compression (DSC) [7], [8], a new video compression standard released by Video Electronics Standards Association (VESA), is designed to enable low-cost hardware implementation of visually lossless video compression over display links. The compressed bit rate of DSC can be configured to 8 bits per pixel (*bpp*) or higher (6bpp or higher in native 4:2:0 mode). Comparing to H.264 and HEVC, a remarkable advantage of DSC is that only one picture line memory is required and no off chip memory is needed. Besides, DSC does not contain computation intensive blocks.

To the best of our knowledge, this work is the first published hardware architecture description of DSC decoder. A low-cost slice interleaving decoding architecture of DSC v1.2a is presented. Based on the proposed architecture, four designs are implemented in 28 nm CMOS, all of which are able to perform real-time decoding of 8K UHD videos.

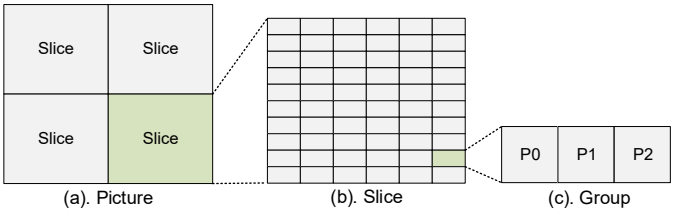

Fig. 1. Relationship of picture, slice, group and pixel: (a) A picture configured into two rows and two columns of slices, (b) A slice of size 18×10 contains 60 groups, and (c) A group contains three pixels.

II. DSC Decoding Algorithm Overview

DSC decoder takes compressed bitstream as input and reconstructs image output. An image, which is also called a picture or a video frame, can be configured into one or more identical sized non-overlap rectangular slices and each slice is independently decoded. The number of columns of slices in an image is called *slices per line*. The decoding of one slice is performed on a group basis, which is three neighboring pixels of the same slice line. Fig. 1 shows the relationship of picture, slice, group and pixel.

Fig. 2 [7], [8] illustrates the decoding process of one slice. *Rate Buffer* buffers the compressed bitstream and sends up to four muxwords to *Substream Demultiplexer* in each group time. Substream Demultiplexer splits single bitstream into three or four substreams, each of which is stored in a different funnel shifter. The *VLC Entropy Decoding* (VLD) block takes substream bits from funnel shifter and decodes flatness information and encoding mode of current group. Each group is encoded in either predictive mode (P-mode) or indexed color history (*ICH*) mode (ICH-mode). Based on the encoding mode, VLD further decodes quantized residuals in P-mode or ICH indices in ICH-mode. *Rate Control* algorithm generates quantization parameter (*Qp*) used in each group. Three predictors are used to predict pixel values: block prediction (BP), modified median adaptive prediction (MMAP) and midpoint prediction (MPP). ICH contains 32 recently used pixels, which are stored in a shift register. If P-mode is selected, pixels are reconstructed using the predicted values and quantized residuals, otherwise, the pixels pointed to by ICH indices are used as the reconstructed pixels. *Line Buffer*

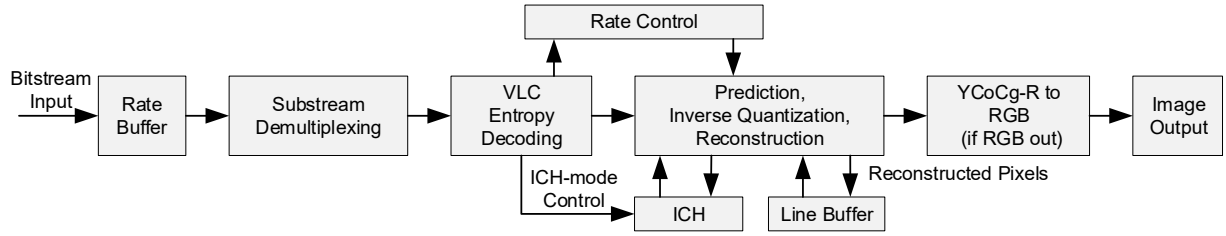

Fig. 2. Block diagram of DSC decoding process.

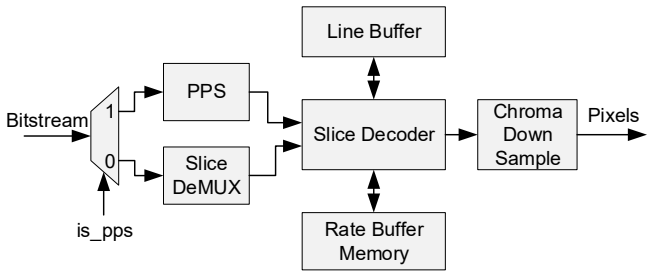

Fig. 3. Block diagram of DSC decoder architecture.

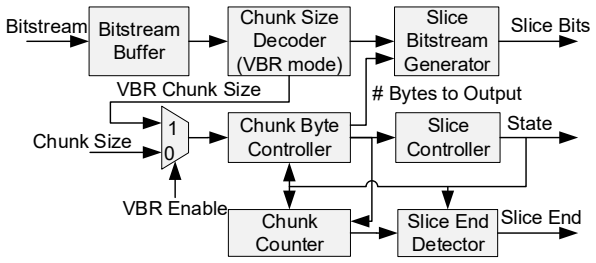

Fig. 4. Block diagram of slice demultiplexer.

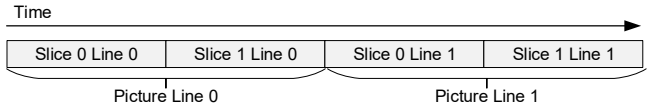

Fig. 5. Slice interleaved decoding of first two lines of a picture which is configured into 2 slices per line.

stores previous line reconstructed pixels, which are used in non-first line ICH and MMAP. If RGB pixel format is desired in output, decoder converts reconstructed pixels' format from YCoCg-R to RGB.

III. Slice Interleaving Architecture

The block diagram of the proposed decoder architecture is shown in Fig. 3. *Picture Parameter Set* (PPS) is a register of 94 bytes which stores the configuration parameters. *Slice Demultiplexer* (Slice DeMUX) decodes size and states of chunk, which is a portion of bitstream containing a set of data bytes. *Slice Decoder* is the major component of DSC decoder which decodes slices. It contains the blocks of DSC decoding process, as shown in Fig. 2. Line buffer and rate buffer are the main memories in DSC decoder. PPS bits are indicated by *is_pps* flag and enter decoder before compressed bitstream. Chroma downsample is applied to pixels only in simple 4:2:2 mode. Fig. 6 shows the 5-stage pipeline of major data paths in the proposed architecture.

A. Slice Demultiplexer

Fig. 4 describes the major blocks of slice demultiplexer. It has four functions: 1) Decode chunk size at the start of each chunk if decoder runs in variable bit rate (VBR) mode, 2) Send bitstream to slice decoder, 3) Determine the owner slice of current bitstream chunk using slice controller, and 4) Detect the boundary of bitstream between neighboring slices, which is critical to make sure correctness of rate buffer writes and reads at the end of slice.

B. One Slice Per Line–No Interleaving

When slice width is configured to be equal to picture width, there is only one column of slice, namely one slice per line. All compressed bits of a slice in picture enter decoder before any bit of later slices. In addition, all the pixels of a slice are decoded before any pixel of later slices. For a DSC decoder which only supports one slice per line, no slice interleaving is needed. Decoder starts decoding next slice only when current slice decoding is done. The slice controller in slice demultiplexer is simplified, since all the chunks belong to the slice being decoded.

C. Multiple Slices Per Line–Slice Interleaving

When a picture is configured into more than one slice per line, the bitstream of different slices in the same row are multiplexed. For example, in the case of two slices per line, the order of bitstream input is *(slice0 chunk0, slice1 chunk0, slice0 chunk1, slice1 chunk1, ...)*. Meanwhile, DSC decoded pixels output in raster scan order, which means the decoding of second picture line will start only when all pixels of first picture line are decoded. Fig. 5 shows the timing of decoding first two picture lines in two slices per line configuration. Therefore, decoder has to keep switching between slices in every slice line.

To decode bitstream compressed with multiple slices per line configuration, slice interleaving architecture is used. First, the decoding state of one slice should be stored when decoder switches to decode another slice and be recovered when switching back to decode the next slice line. Therefore, all the registers that store the decoding state should be replicated as many times as the maximum supported number of slices per line. Fig. 7 describes an example of no interleaving and two-way interleaving architecture. Comparing to no interleaving architecture, slice interleaving architecture uses same combi-

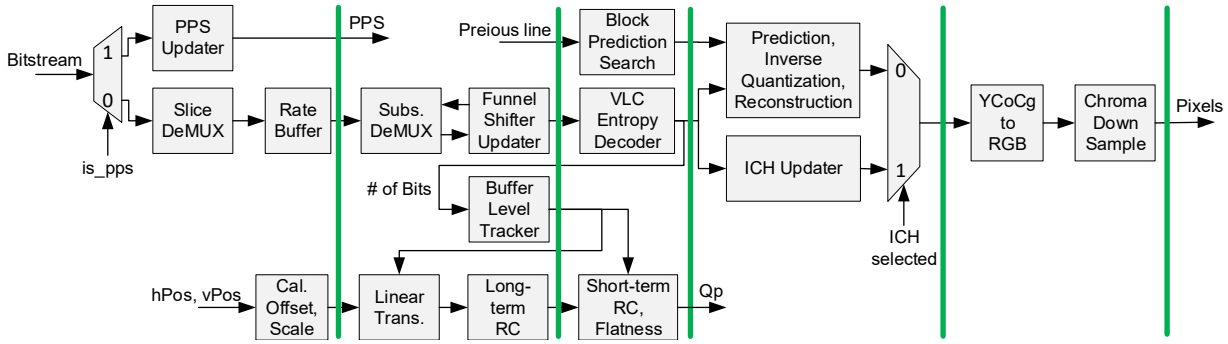

Fig. 6. Pipeline diagram of major data paths in proposed DSC decoder architecture.

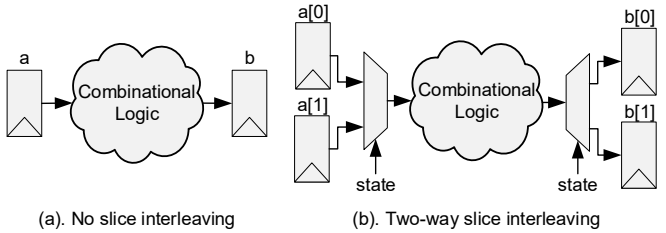

Fig. 7. Comparison of (a) No slice interleaving architecture, and (b) Two-way slice interleaving architecture.

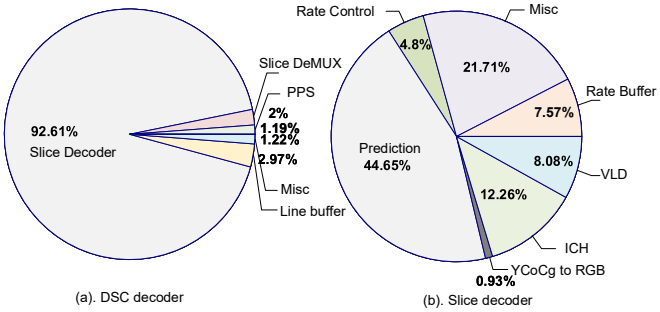

Fig. 8. Logic area utilization in (a) DSC decoder, and (b) Slice decoder.

national logic, which is shared by all slices of the same row, while registers are replicated for each slice. At the starting point of combinational path, a multiplexer is needed to select the source register value. Similarly, a demultiplexer is used to select the destination register at the end point of path. Second, line buffer and rate buffer have same number of banks as the maximum supported slices per line. The data of different slices is stored in different banks, since slices are independently decoded. For a decoder supporting up to four slices per line, each bank is used by a different slice when decoding four slices per line, while two banks are used by each slice when decoding two slices per line.

Two controllers are used to enable slice interleaving architecture: 1) *state*, which is generated in slice demultiplexer, indicates the owner slice of current chunk and controls the write of rate buffer, and 2) *pixel_state* indicates the owner slice of current decoded slice line and controls rate buffer read and rest of slice decoder.

D. Main Memories

Line buffer memory has a size of one picture line, which is 30.9KB and 45KB to support 8K UHD with maximum bits per component (*bpc*) of 10 bits and 16 bits, respectively.

Rate buffer size is *rc_model_size* – *initial_offset* + $\lceil$*initial_xmit_delay* * *bits_per_pixel*$\rceil$ + *groupsPerLine* * *first_line_bpg_offset* number of bits. By adopting the suggested parameter values from DSC, the equation can be reduced to 10,240 + *slice_width* * 5. Therefore, the minimum rate buffer size is 5.94KB to support one slice per line in 8K UHD. When multiple slices per line is supported, each column of slices should have separate rate buffer bank(s). Thus, rate buffer size is increased to (10,240 + *slice_width* * 5) * *slices_per_line*, which is 7.2KB and 9.7KB to support up to 2 and 4 slices per line in 8K UHD, respectively.

IV. Implementation Results and Analysis

A. Summary

Based on the proposed architecture, four designs are implemented in register transfer level (RTL) using Verilog. The designs support all mandatory and optional operations of DSC v1.2a decoding process and are verified by comparing to the DSC v1.2a reference C model. Key features include support of simple 4:4:4 (both RGB and YUV output formats), simple 4:2:2, native 4:2:2 and native 4:2:0 modes with both constant and variable bit rate modes. Design I and II support one slice per line without using slice interleaving. Design III and IV use slice interleaving architecture and support one and multiple slices per line. Design I supports maximum bits per component of 10 bits and the other three designs support up to 16 bits. Design III supports one and two slices per line, while design IV supports one, two and four slices per line. The designs are synthesized with a 28 nm CMOS standard cell library using typical-typical corner device and 1.2 V supply voltage.

B. Results and Analysis

1) Area Utilization: Fig. 8 shows the logic area utilization of design II. Slice decoder is the most complex block in entire decoder and takes 92.61% of total logic area. In slice decoder, prediction block occupies much more logic area than other blocks.

TABLE I
COMPARISON WITH 8K H.264 AND HEVC VIDEO DECODER CHIPS

	Standard	Tech (nm)	Voltage (V)	Clock (MHz)	Max Frame Rate (fps)		Throughput (Gpixel/s)	Gate Count (K)	On-Chip Main Memory (KB)	Max bpc
					4K UHD	8K UHD				
ISSCC'12 [4]	H.264	65	1.2	340 (1251*)	883*	60 (220*)	2 (7.36*)	1338	79.9	-
JSSC'17 [6]	HEVC	40	1.0	300 (427*)	683*	120 (170*)	4 (5.70*)	2887	396	10
Design I	DSC	28	1.2	1190	430	107	3.55	**161**	**36.9**	10
Design II	DSC	28	1.2	1120	405	101	3.35	**231**	**50.9**	16
Design III	DSC	28	1.2	1087	393	98	3.25	**258**	**52.2**	16
Design IV	DSC	28	1.2	1040	376	94	3.11	**282**	**54.7**	16

* Estimated value when scaled to 1.2V 28 nm technology using scale factors from [9].

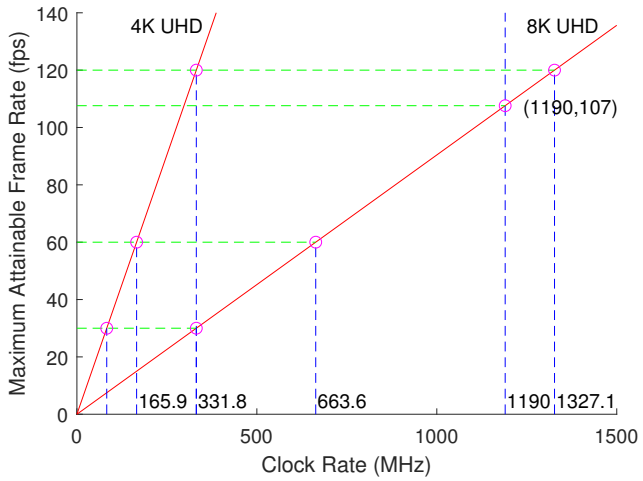

Fig. 9. Maximum attainable frame rate when decoding 4K and 8K UHD videos with a throughput of 3 pixels per clock cycle for 4:4:4 pixels. For native 4:2:2 and 4:2:0 pixel formats, frame rates are double these values.

2) Gate Count: Table I reports gate count in minimum-sized NAND2 equivalent logic gate, which is 161K–282K. The gate count of design I is 1.43× smaller than design II, due to the reduced component bit width of pixels. Though maximum supported component bit width are the same, design III and IV have larger gate count than design II. This is the effect of both the extra register selection logic and the fact that some logic can't be shared between slices and is replicated.

3) Maximum Frequency: The designs have maximum frequency of 1.04–1.19 GHz. Increasing maximum component bit width slows down the maximum clock frequency, as shown in the result of design I and II. Design III and IV are slower than design II, since register selection logic makes extra contribution to path delay. Furthermore, design IV requires larger register selection logic than design III, thus maximum frequency is smaller.

4) Throughput and Maximum Frame Rate: The throughput is 3 pixels per clock cycle in simple 4:4:4 and simple 4:2:2 modes, and 6 pixels per clock cycle in native 4:2:2 and 4:2:0 mode. Fig. 9 shows the maximum attainable frame rate with regards to clock rate when decoding 4K UHD and 8K UHD videos in 4:4:4 mode. The maximum achievable frame rates are 376–430 frames per second (fps) in 4K UHD and 94–107 fps in 8K UHD.

5) Comparison with H.264 and HEVC Decoders: When scaled to same technology as of the proposed designs, H.264 decoder [4] and HEVC decoder [6] have higher estimated throughput and maximum frame rate than the four designs of this work. Comparing to H.264 and HEVC decoders, the designs of this work have much smaller hardware cost. The designs have gate count of 4.7×–8.3× smaller than [4] and 10.2×–17.9× smaller than [6], while on-chip main memory is 1.46×–2.16× and 7.23×–10.7× smaller, respectively.

V. CONCLUSION

This paper presents a slice interleaving decoding architecture which is able to decode one and multiple slices per line. Four designs that fully support DSC v1.2a decoding algorithm are implemented and synthesized with a 28 nm standard cell library. The designs have very low hardware cost in terms of gate count and on-chip main memory, and are able to perform real-time decoding for 4K UHD and 8K UHD videos with frame rates up to 376–430 fps and 94–107 fps, respectively.

REFERENCES

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

[2] G. J. Sullivan *et al.*, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[3] D. Zhou *et al.*, "A 530 Mpixels/s 4096x2160 60fps H.264/AVC high profile video decoder chip," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 777–788, April 2011.

[4] D. Zhou, J. Zhou *et al.*, "A 2Gpixel/s H.264/AVC HP/MVC video decoder chip for super Hi-Vision and 3DTV/FTV applications," in *2012 IEEE International Solid-State Circuits Conference*, Feb 2012, pp. 224–226.

[5] C. T. Huang *et al.*, "A 249Mpixel/s HEVC video-decoder chip for quad full HD applications," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb 2013, pp. 162–163.

[6] D. Zhou *et al.*, "An 8K H.265/HEVC video decoder chip with a new system pipeline design," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 113–126, Jan 2017.

[7] V. E. S. Association, "VESA Display Stream Compression (DSC) standard v1.2a," Jan 2017. [Online]. Available: http://vesa.org

[8] F. G. Walls and A. S. MacInnis, "VESA Display Stream Compression for television and cinema applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, no. 4, pp. 460–470, Dec 2016.

[9] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration, the VLSI Journal*, vol. 58, pp. 74–81, 2017.