Display Stream Compression Encoder Architectures for Real-time 4K and 8K Video Encoding

Shifu Wu, Snehlata Gutgutia*, Massimo Alioto* and Bevan Baas
Department of Electrical and Computer Engineering
University of California, Davis
*Department of Electrical and Computer Engineering
National University of Singapore

***Abstract*—The Display Stream Compression (DSC) standard enables visually-lossless video compression with a much lower hardware cost than H.264 and HEVC albeit with a lower level of compression. We present the first published DSC encoder architectures—including a Single Slice architecture which supports one slice per line encoding and a Slice Interleaving architecture which supports one or more slices per line. Seven designs that fully support the DSC v1.2a standard have been implemented and synthesized in a 28 nm CMOS standard cell library. The designs are able to perform real-time encoding at frame rates up to 35–40 frames per second (fps) for 8K UHD (7680×4320), 140–160 fps for 4K UHD (3840×2160), and 560–642 fps for 1080p (1920×1080) video in 4:4:4 mode. In native 4:2:2 and 4:2:0 modes, frame rates are doubled to 70–80 fps, 280–320 fps and 1120–1284 fps for 8K UHD, 4K UHD and 1080p, respectively. The designs require 194–433 K minimum-sized NAND2 equivalent gates and main memory of 31.9–68.5 KB to support 8K UHD.**

I. Introduction

Ultra high definition (UHD) video resolutions of 4K UHD (3840×2160) and 8K UHD (7680×4320) and high frame rates resulting in pixel rates of over 1 billion pixels/sec are becoming increasingly common, however their transmission and storage is challenging. Hardware cost is an important consideration for video encoders that support UHD video compression. H.264/AVC [1] and H.265/HEVC [2] provide excellent compression but utilize large number of logic gates, large on-chip and off-chip memories.

In contrast, the Display Stream Compression (DSC) [3], [4] standard, which is released by the Video Electronics Standards Association (VESA), requires a much smaller amount of hardware which enables many new applications of video compression albeit with a lower level of compression. The compressed bit rate of DSC can be programmed to 8 bits per pixel (bpp) or higher (6 bpp or higher for 4:2:0 pictures). DSC features only a single picture line of memory storage without necessitating off-chip memory. In addition, the computational complexity of the DSC encoding algorithm is relatively low.

To the best of our knowledge, this work is the first published description of a DSC hardware encoder architecture, although a DSC decoder architecture was reported recently [5]. Two general DSC encoder architectures are presented: 1) A *Single Slice* architecture which is capable of encoding pictures with one column of slice, and 2) a *Slice Interleaving* architecture which can encode pictures containing one or more columns of slices. In total, seven designs are implemented in 28 nm CMOS utilizing these two architectures.

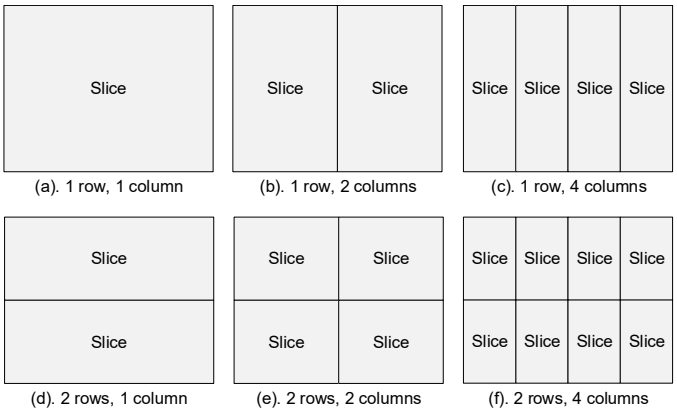

Fig. 1. Example of pictures configured to 1 (top) or 2 (bottom) rows, and 1 (left), 2 (middle) or 4 (right) columns of slices.

The reminder of this paper is organized as follows. Section II provides an overview of the DSC encoding process. Section III covers the Single Slice architecture, and section IV describes the Slice Interleaving architecture. Section V evaluates implementation results. Section VI concludes the paper.

II. Display Stream Compression Encoding

DSC encodes each picture, also called an image, independently. A picture can be configured into one or more contiguous identically-sized rectangular non-overlapping *slices*. A slice can have a size up to an entire picture, as shown in Fig. 1(a), or there can be multiple rows of slices and/or columns of slices in a single picture, as shown in Fig. 1(b–f). The number of columns of slices per picture, *slices per line*, is an key parameter that strongly affects the encoding architecture. However, the number of rows of slices per picture does not impact the hardware architecture.

Figure 2 [3], [4] illustrates the DSC encoding process. The encoding is performed on a *group* basis, which is three neighboring pixels of the same slice line. There are in total 7 main sub-processes applied as follows:

1) *RGB to YCoCg-R* converts the RGB format pixel into a reversible YCoCg format.
2) *Flatness Determination* estimates whether upcoming pixels are *"flat"*.

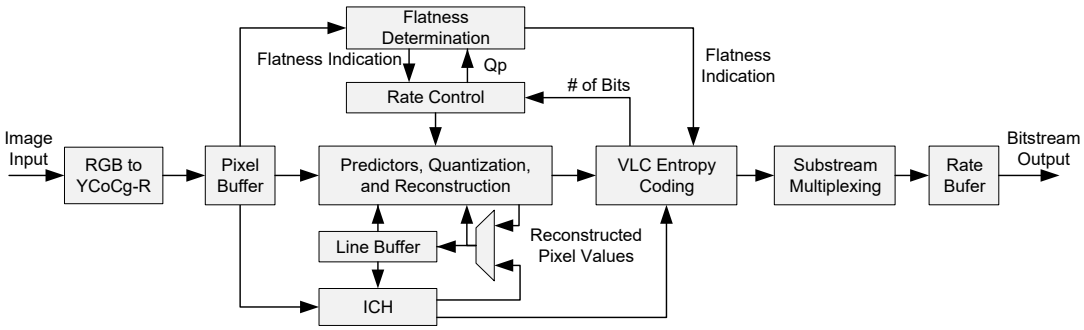

Fig. 2. Block diagram of the entire DSC encoding process.

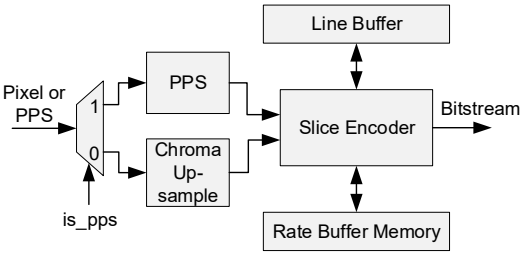

Fig. 3. Block diagram of the single slice architecture.

3) *Rate Control* manages Rate Buffer fullness and adjusts quantization level to maximize perceived quality. Quantization parameter (Qp) quickly drops if the upcoming group is flat.
4) *Predictors, Quantization, and Reconstruction (Prediction)* uses three predictors to estimate sample values: modified median adaptive prediction (MMAP), block prediction (BP) and midpoint prediction (MPP). The residual, which is the difference between the original pixel value and the predicted pixel value, is quantized. Pixel reconstruction is based on quantized residual and predicted value.
5) *Indexed Color History (ICH)* keeps track of 32 recently-used reconstructed pixel values which are used to code current pixel values in ICH mode.
6) *VLC Entropy Coding (VLC)* encodes each group with quantized residuals if prediction mode (P-mode) is selected, or ICH indices if ICH mode is selected.
7) *Substream Multiplexing* packs the encoded variable length syntax elements from the VLC into fixed-size packets, called *muxword*, which are multiplexed into a single bitstream.

In addition, the DSC encoder contains three major memories:

1) *Line Buffer* stores reconstructed pixels of the previous line, which are used in MMAP and ICH.
2) *Balance FIFO* buffers muxwords of each substream.
3) *Rate Buffer* converts a variable number of bits used to encode groups into a constant-rate bitstream output.

III. The Single Slice Architecture

Figure 3 shows the proposed *Single Slice* architecture. Before encoding starts, *Picture Parameter Set* (PPS) data

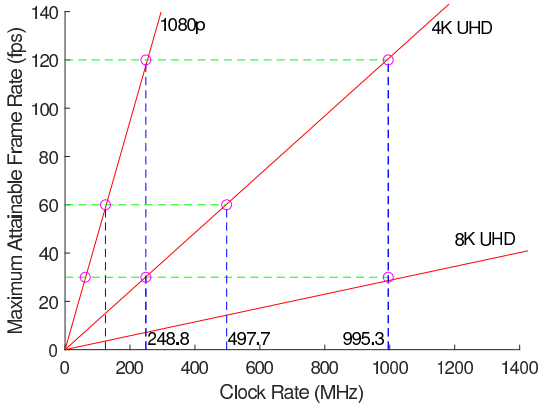

Fig. 4. Maximum attainable encoding frame rates of 1080p, 4K UHD and 8K UHD videos with a throughput of 1 pixel per clock cycle for 4:4:4 pixels.

enters the encoder and are stored in PPS registers, which have a total size of 94 bytes. When PPS is ready, pixel data enters the encoder in raster scan order. A *Chroma Upsample* process, which converts pixels from 4:2:2 format to 4:4:4 format by interpolating chroma values, is applied to pixels if the encoder is working in simple 4:2:2 mode. The *Slice Encoder*, which consists of sub-processes of the entire encoding process described in Section II, operates in units of slices and compresses pixels into a bitstream.

A. Key Features

The presented Single Slice architecture is capable of encoding all DSC-specified pixel formats: 4:4:4 (both RGB and YCbCr format), 4:2:2, and 4:2:0; in addition to both variable bit rate (VBR) and constant bit rate (CBR) bitstream modes. This architecture supports an encoding of 1 slice per line and delivers a throughput of 1 pixel per clock cycle in 4:4:4 mode. The throughput is effectively doubled in native 4:2:2 and 4:2:0 modes since two neighboring pixels are packed into one container pixel.

This architecture is able to handle special cases including: a partial group at the end of a slice line, pixel replication for slices that exceed the right boundary of a picture, and midpoint padding for slices that go beyond the bottom boundary of picture. Encoding automatically stalls when new pixel data is unavailable or buffers are full, and resumes once new pixels come and buffers are ready for writes.

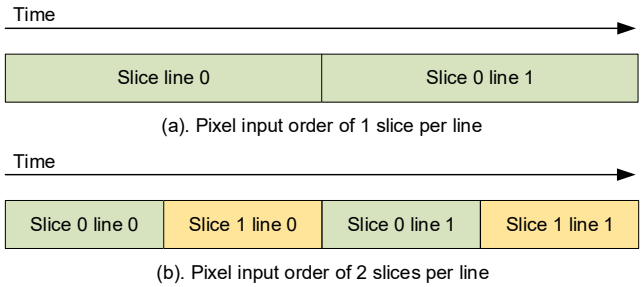

Fig. 5. Pixel input order of the first two picture lines when configured into 1 and 2 slices per line.

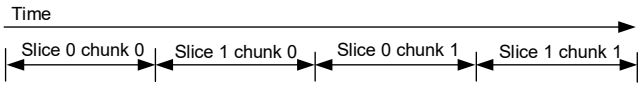

Fig. 6. Bitstream chunk output order when encoding 2 slices per line.

B. Maximum Attainable Frame Rates

Fig. 4 plots the linear relationship between clock rate and maximum achievable frame rates with 1 pixel per cycle throughput when encoding 4:4:4 pixels. The encoder can achieve 120 frames per second (fps) for 1080p (1920×1080) videos at a clock rate of 248.8 MHz. At 995.3 MHz, the encoder is able to encode 4K UHD at 120 fps, and 8K UHD at 30 fps. For a given clock rate, the maximum frame rates are doubled in native 4:2:2 and native 4:2:0 modes.

C. Memories

1) Line Buffer: The buffer's size is one picture line of pixels. Since picture width is effectively halved in native 4:2:2 and 4:2:0 modes, and the fourth component is needed only in native 4:2:2 mode, we use a half picture width size for odd- and even- position luma memory, whereas chroma memory sizes are one full picture line. In 4:4:4 mode, both half-sized memories are used to store luma sample values. The total memory size to support 8K UHD is 30 KB for 10 bits per pixel component (bpc) and 45 KB for 16 bpc.

2) Balance FIFOs: Four FIFOs are needed, one for each substream, to accumulate *muxWordSize* + *maxSeSize* – *1* syntax elements before removing the muxwords. 522 and 1040 bytes are required to support 10 bpc and 16 bpc, respectively.

3) Rate Buffer: DSC [3] standard's Appendix E gives a minimum buffer size bound, which is equal to *rc_model_size* – *initial_offset* + ⌈*initial_xmit_delay* * *bits_per_pixel*⌉ + *groupsPerLine* * *first_line_bpg_offset* number of bits. By applying the suggested variable values from DSC, this equation is reduced to *fixed_size* + *picture width* * 5 bits, where *fixed_size* is equal to 10,240. Therefore, the minimum buffer size is 3680 and 6080 bytes to support resolutions up to 4K UHD and 8K UHD, respectively.

IV. The Slice Interleaving Architecture

When a picture is configured to one slice per line, as shown in Fig. 1(a, d), all the pixels of one slice enter the encoder before any later slices. However, in a multiple-slice-per-line configuration, as shown in Fig. 1(b, c, e, f), pixels of slices in the same row enter the encoder in an interleaved manner. Fig. 5 describes the order of pixel input for the first two picture lines in one and two slices-per-line configurations. The bitstream output is also interleaved in units of *chunks* across slices in the same row, as indicated in Fig. 6.

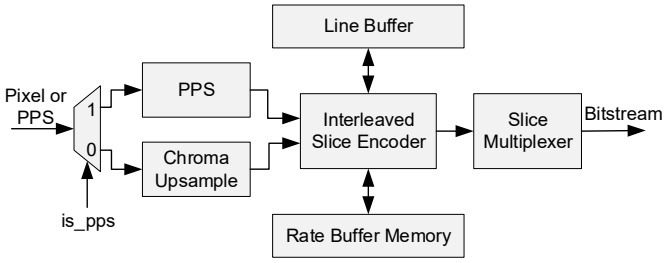

Fig. 7. Block diagram of the slice interleaving architecture.

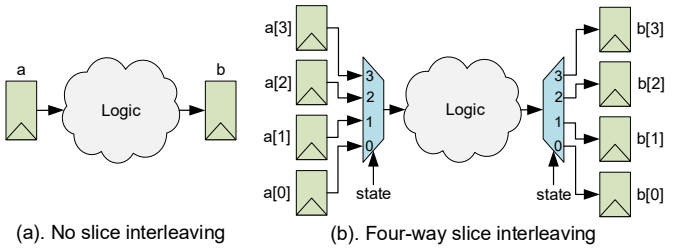

Fig. 8. Example of (a) no interleaving and (b) four-way slice interleaving.

To address the pixel and bitstream interleaving challenges incurred in multiple slices per line encoding, we present the *Slice Interleaving* architecture. The block diagram of this architecture is shown in Fig. 7, which is equivalent to Fig. 3 with two major changes: Slice Encoder is replaced with *Interleaved Slice Encoder*, and *Slice Multiplexer* is added. This architecture maintains all features of the Single Slice architecture, with an additional capability of encoding multiple slices per line.

A. Interleaved Slice Encoder

The Interleaved Slice Encoder is made up of all the sub-processes in the DSC encoding algorithm, as described in Section II. When encoding pictures with one slice per line, it behaves the same as the Slice Encoder in the Single Slice architecture. In case of multiple slices per line, it switches to the next slice after encoding one slice line of the current slice. For two slices per line, as shown in Fig 5(b), line 0 of slice 0 is first encoded, then the encoder works on line 0 of slice 1. Once done, the slice encoder switches back to slice 0 and continues to encode line 1. This process repeats for all slice lines.

As different slices are independently encoded, the slice encoder records the encoding state of current slice before switching to the next slice, and recovers the state when switching back in the next slice line. This is achieved by slice interleaving encoding, which shares the combination logic for all column slices, while using separate registers to store the encoding state of each slice. Registers storing the encoding

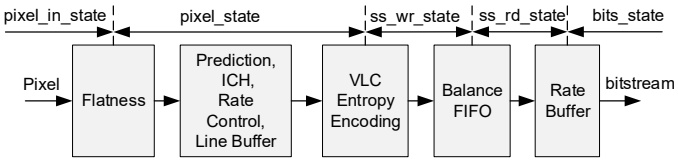

Fig. 9. Slice interleaving controllers.

state of the current slice are updated only when the slice encoder works on the current slice.

Fig. 8 explains generic scenario of a register-logic-register path without slice interleaving and a registers-multiplexer-logic-demultiplexer-registers path with four-way slice interleaving. Registers *a* and *b* are replicated three times, therefore it supports up to four slices per line. The slice controller, *state*, selects the registers of the current slice for reads and writes.

B. Slice Multiplexer

The Slice Multiplexer handles the aforementioned bitstream interleaving challenge in encoding multiple slices per line. It multiplexes compressed slices in the same row into a single bitstream. First, a control signal *bits_state* is generated, which indicates the slice that the current chunk belongs to. Then, a fixed number of bits of the slice selected by *bits_state* are read from Rate Buffer in each cycle. Once one chunk of bits is removed, *bits_state* value is updated, pointing to the next slice. This process repeats until all compressed data are read from the Rate Buffer.

C. Slice Control

Fig. 9 shows the following five slice controllers used for register selection in different sub-processes of the Interleaved Slice Encoder:

1) *pixel_in_state* indicates the current column slice which pixels entering the Interleaved Slice Encoder belong to, used in flatness check.
2) *pixel_state* decides the column slice to be processed by flatness adjustment, Prediction, ICH, Rate Control, Line Buffer, and VLC.
3) *ss_wr_state* controls Balance FIFO writes.
4) *ss_rd_state* controls Balance FIFO reads and Rate Buffer writes.
5) *bits_state* selects the slice for the current bitstream chunk output.

D. Memory Architecture

Support for multiple-slice-per-line encoding has two impacts on memory architecture: multiple memory banks and increased memory size.

1) Multi-bank Memory: To ensure correct data access, separate Line Buffer and Rate Buffer memory banks for different column slices are required. Memory bank allocation adapts to different slice-per-line configurations, as shown in Fig. 10. For one slice per line, all four memory banks are used by the only slice in each row, while two banks are allocated to each slice when encoding two slices per line. In the case of

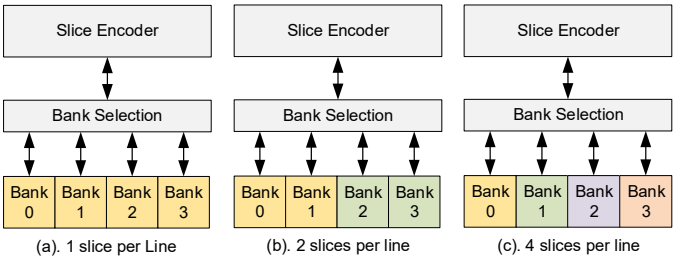

Fig. 10. Multi-bank memory shared by column slices when encoding 1, 2, and 4 slices per line in a four-way interleaved slice encoder.

pictures configured into four slices per line, each slice accesses a separate memory bank.

2) Memory Size: The Line Buffer size does not change with the number of slices per line, and it is always equal to one picture line pixel storage. However, since each column of slice needs a separate rate buffer, the *fixed_size* term in Rate Buffer size equation duplicates for each slice. Therefore, moving from the Single Slice architecture to a four-way Slice Interleaving architecture, the Rate Buffer size increases from 6080 bytes to 9920 bytes to support 8K UHD encoding. Balance FIFOs are replicated for each column of slice.

V. Implementation Results and Analysis

A. Summary

To exploit effects of different pixel component bit depths of uncompressed video, five designs supporting up to 8, 10, 12, 14 and 16 bpc are implemented using the Single Slice architecture: *Single_8bpc*, *Single_10bpc*, *Single_12bpc*, *Single_14bpc* and *Single_16bpc*. In addition, two Slice Interleaving architecture designs are implemented: *Interleave2_16bpc* and *Interleave4_16bpc*, which supports up to 2 and 4 slices per line, respectively. All designs support 8K UHD resolution, and are designed at the register transfer level in Verilog and synthesized with a 28 nm CMOS standard cell library using typical-typical corner devices with a 1.2 V operating voltage.

B. Maximum Frame Rates

Real-time encoding capability is a vital consideration for video encoders. Columns 6–8 of Table I list maximum frame rates of different designs encoding 1080p, 4K UHD and 8K UHD in 4:4:4 mode, respectively. All designs are able to perform real-time encoding with maximum frame rates up to 560–642 fps for 1080p, 140–160 fps for 4K UHD and 35–40 fps for 8K UHD.

C. Area and Delay

Area breakdown analysis identifies the complexity of different blocks. Fig. 11 shows the logic area utilization of the entire DSC encoder and Slice Encoder in *Single_10bpc*. The Slice Encoder occupies 95 percent of the logic area in the DSC encoder, while Prediction and ICH are the most complex blocks in the Slice Encoder and occupy 35.71 and 36.45 percent of total logic area, respectively.

Fig. 12 plots the critical path delay, logic area and register area of different designs, normalized by *Single_16bpc*. For

TABLE I
COMPARISON OF DSC ENCODERS AND A H.264&HEVC VIDEO CODEC

Design	Standard	Tech (nm)	Voltage (V)	Clock (MHz)	Max Frame Rate (fps)			Gate Count(K)	On-chip Main Memory (KB)	Off-chip Memory
					1080p	4K	8K			
ISSCC'15 [6]	H.264&HEVC	28	0.9	494	-	30	-	3558	308	32-b LPDDR3
Single_8bpc	DSC	28	1.2	1333	642	160	40	**194.2**	**31.9**	None
Single_10bpc	DSC	28	1.2	1333	642	160	40	**233.6**	**38.0**	None
Single_12bpc	DSC	28	1.2	1282	618	154	38	**276.9**	**44.3**	None
Single_14bpc	DSC	28	1.2	1265	610	152	38	**318.0**	**50.6**	None
Single_16bpc	DSC	28	1.2	1205	581	145	36	**322.3**	**55.0**	None
Interleave2_16bpc	DSC	28	1.2	1190	573	143	35	**366.2**	**60.4**	None
Interleave4_16bpc	DSC	28	1.2	1162	560	140	35	**433.5**	**68.5**	None

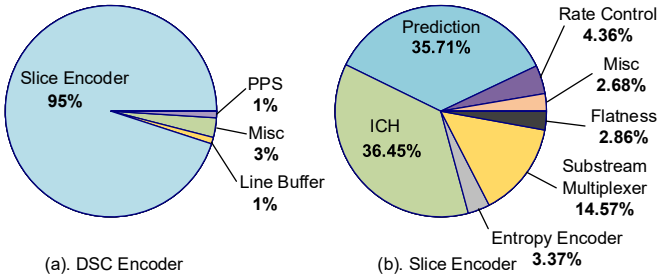

Fig. 11. Logic area utilization in (a) DSC encoder and (b) slice encoder.

Single Slice architecture designs, an increase of the maximum supported component bit depth yields increased delay, larger logic and register area. Comparing to *Single_16bpc*, *Single_8pc* has 1.114×, 1.659× and 1.559× smaller delay, logic area and register area, respectively. With maximum bpc of 16, Slice Interleaving designs have longer delay, while occupying larger logic and register area than Single Slice designs. The register selection multiplexers and demultiplexers contribute to increased delay and logic area, resulting in an increase of 1.038× and 1.136× for *Interleave2_16bpc*, 1.064× and 1.345× for *Interleave4_16bpc*, both of which are compared to *Single_16bpc*. Since registers storing encoding state are replicated, the register area of *Interleave2_16bpc* and *Interleave4_16bpc* are 1.583× and 2.717× larger than *Single_16bpc*.

D. Gate Count and Memory Size

Table I shows all DSC encoder designs have a small hardware cost with gate counts of 194.2–433.5 K and on-chip memory sizes of 31.9–68.5 KB, which is 8.2×–18.3× and 4.5×–9.6× smaller than an H.264&HEVC codec [6]. The Line Buffer and Balance FIFO size increase as maximum bpc increases, resulting in total memory size growth of 23.1 KB from *Single_8bpc* to *Single_16bpc*. On-chip memory of *Interleave2_16bpc* and *Interleave4_16bpc* is 1.1× and 1.25×larger than that of *Single_16bpc*, as separate Rate Buffers and Balance FIFOs are needed for each column slice.

VI. CONCLUSION

This paper presents two hardware architectures that fully support DSC v1.2a encoding: a Single Slice architecture and a Slice Interleaving architecture. Based on the proposed architectures, 7 designs are implemented and evaluated, all of which have low hardware costs in terms of logic gate counts and memory requirements. These designs are able to perform real-time encoding at frame rates up to 560–642 fps, 140–160 fps and 35–40 fps in 4:4:4 mode, and 70–80 fps, 280–320 fps and 1120–1284 fps in native 4:2:2 and 4:2:0 modes for 1080p, 4K UHD and 8K UHD , respectively.

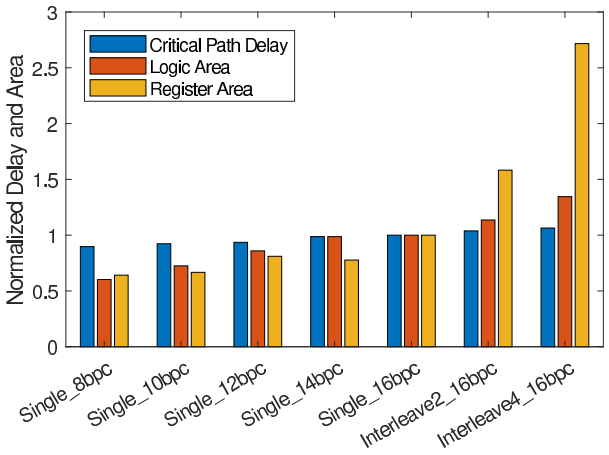

Fig. 12. Normalized delay, logic and register area in single slice architecture and slice interleaving architecture designs.

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

[4] 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.

[5] S. Wu and B. Baas, "A low-cost slice interleaving DSC decoder architecture for real-time 8K video decoding," in *2018 IEEE International Midwest Symposium on Circuits and Systems*, Aug 2018.

[6] C. C. Ju *et al.*, "A 0.5nJ/pixel 4K H.265/HEVC codec LSI for multi-format smartphone applications," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb 2015, pp. 1–3.