HIGH THROUGHPUT BLOCK CODING IN THE HTJ2K COMPRESSION STANDARD

David Taubman, Aous Naman and Reji Mathew

School of Electrical Engineering and Telecommunications,
The University of New South Wales, Sydney, Australia.

ABSTRACT

This paper describes the block coding algorithm that underpins the new High Throughput JPEG 2000 (HTJ2K) standard. The objective of HTJ2K is to overcome the computational complexity of the original block coding algorithm, by providing a drop-in replacement that preserves as much of the JPEG 2000 feature set as possible, while allowing reversible transcoding to/from the original format. We show how the new standard achieves these goals, with high coding efficiency, and extremely high throughput in software.

***Index Terms*—** JPEG 2000, JPH, Image coding, Low complexity, Vectorization

1. INTRODUCTION

JPEG 2000 offers both high coding efficiency and a rich set of features, including: scalability; region-of-interest accessibility; and optimized rate control without iterative encoding. The code-stream contains embedded subsets representing a hierarchy of spatial resolutions and a fine-grained progression of quality/SNR operating points. Accessibility arises from the fact that the sub-bands produced by wavelet transformation are partitioned into code-blocks that are coded independently. These properties facilitate extremely efficient interactive access to large compressed images, where both decoding and communication are limited to a region or resolution of interest. Beyond these core features of JPEG 2000 Part-1 [1], Part-2 [2] defines rich multi-component transforms, with applications in hyperspectral imagery, multi-focal microscopy and compression of medical volumes, along with non-linear transforms that support efficient HDR compression, and many other tools. Part-9, also known as JPIP [3, 4], defines transport protocols and request/response syntax to support efficient interactive browsing of imagery, volumes and even video over lossy and lossless networks, leveraging heavily from the random access and scalability of JPEG 2000.

Recently, the JPEG committee, formally known as Working Group 1 (WG1) of ISO/IEC JTC1/SC29, has finalized its work on a new Part-15 to the JPEG 2000 family, known as High Throughput JPEG 2000 (HTJ2K). The HTJ2K standard defines an alternate block coding algorithm that can be used as a drop-in replacement for the original algorithm of Part-1, preserving most features from all parts of the JPEG 2000 family, while offering much lower computational complexity. HTJ2K also defines a wrapping file format known as JPH, which may become a more common name for the technology.

The purpose of this paper is to provide an overview of the High Throughput (HT) block coding algorithm in HTJ2K, explaining how the algorithm can be used in encoding and transcoding applications, and to provide experimental evidence for the coding efficiency and throughput of the entire system, in a software environment.

2. HTJ2K STANDARDIZATION

The HT block coder in HTJ2K is ultimately derived from an algorithm known as Fast Block Coding with Optimized Truncation (FBCOT), which was first proposed in [5] for the JPEG-XS standard for lightweight, ultra-low latency video compression. While FBCOT met the low latency requirements of JPEG-XS, with high coding efficiency, a different approach was selected for that standard, primarily due to a hard requirement to fit the solution within a specified FPGA deployment target.

The FBCOT algorithm in [5], which is further documented in [6], presented sufficient evidence for initiation of the HTJ2K project within JPEG. A public Call for Proposals (CfP) [7] was issued in June 2017, with evaluations in April 2018, which resulted in a variant of the FBCOT algorithm [8] being adopted for the first working draft of the standard. A series of core experiments led to further improvements in coding efficiency without any loss in throughput [9], with the Draft International Standard released for ballot in October 2018.

The key requirements expressed in the CfP [7] are: a) support for reversible transcoding with existing JPEG 2000 code-streams; b) a loss of no more than 15% in coding efficiency; and c) an average increase in block coder throughput (encoder and decoder) of at least 10x, for an optimized software implementation, in comparison to the JPEG 2000 Part-1 block coding algorithm . These requirements were not only met, but considerably exceeded, as demonstrated by a formal WG1 evaluation; key results from this evaluation are summarised in Figs. 1, 2.

Note, however, that the formal WG1 performance evaluation involved only the block coding module itself. Images and video frames from a test suite were individually compressed using JPEG 2000 Part-1, then reversibly transcoded to use the new HT block coding algorithm, allowing relative changes in coding efficiency and throughput to be measured in a well defined test harness. By contrast with this approach, an objective of this paper is to shed light on the coding efficiency and throughput of a complete HTJ2K system, including wavelet transform, colour transformation, and all components required to generate and parse the complete code-stream syntax.

3. FBCOT AND THE HT BLOCK CODER

We use the term FBCOT to refer to a complete compression system based on the HT block encoder, while HTJ2K refers only to the code-stream syntax and decoding procedures that are defined in JPEG 2000 Part-15. The distinction between FBCOT and HTJ2K is that only the decoding process is the subject of standardization. Other fast alternatives to the JPEG 2000 block coder have previously been proposed [10, 11], but without considering the lossless transcoding and rate control options that form part of the FBCOT approach.

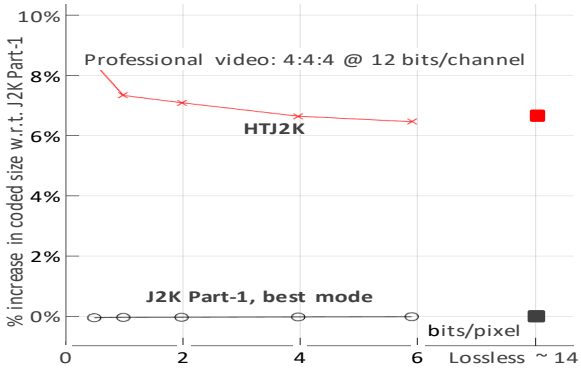

Fig. 1. JPEG formal evaluation of HTJ2K coding efficiency.

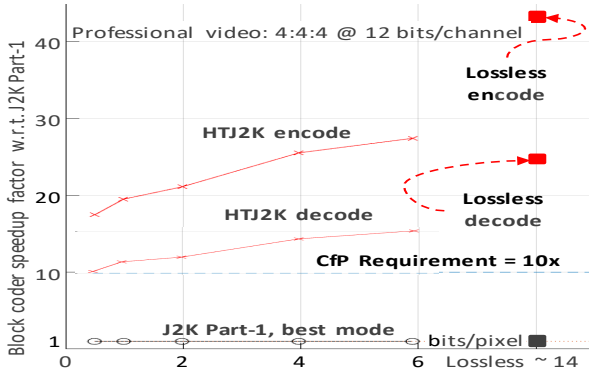

Fig. 2. JPEG formal evaluation of HTJ2K block coder throughput.

Fig. 3 illustrates the overall FBCOT architecture. FBCOT and HTJ2K substantially preserve the existing architecture and code-stream sytnax of JPEG 2000. Imagery is first subjected to any required multi-component transforms and/or non-linear point transforms, as allowed by Part-1 or Part-2 of JPEG 2000, after which transformed image components are processed by a reversible or irreversible Discrete Wavelet Transform (DWT), which decomposes each component into a hierarchy of detail sub-bands and one base (LL) sub-band. All sub-bands are partitioned into blocks whose size is no more than 4096 samples, with typical dimensions being 64×64 or 32×32; very wide and short blocks such as 1024×4 are also important for low latency applications. Each block is individually quantized (if irreversible) and coded, producing a block bit-stream comprising one or more coding passes. In the encoder, an optional Post-Compression Rate-Distortion optimization (PCRD-opt) phase is used to discard generated coding passes so as to achieve a rate or distortion target, which may be global (whole code-stream) or local (small window of code-blocks). Finally, the bits belonging to the selected coding passes from each code-block are assembled into J2K packets to form the final code-stream.

The original JPEG 2000 block coding algorithm, identified here as J2K-1, processes sub-band samples in coding passes, known as Cleanup (CUP), Significance Propagation (SP) and Magnitude Refinement (MR). Each CUP pass brings the quality of all samples in the code-block to a whole magnitude bit-plane boundary, while the SP and MR are refinement passes that raise only certain samples to the next bit-plane boundary. Truncating the representation at the end of an SP or MR pass effectively results in a data-dependent variable level of quantization. Since J2K1 makes three passes through the code-block for each bit-plane, the algorithm is computationally expensive. At high bit-rates, including lossless compression, the block encoding and decoding complexity of J2K1 can exceed that of all other JPEG 2000 processes by more than an order of magnitude.

The HT block coding algorithm also adopts a coding pass structure, with CUP, SP and MR coding passes, defined with respect to bit-planes p. However, the CUP pass associated with each bit-plane p fully encodes the magnitudes $\mu_p[n] = \left\lfloor \frac{|x[n]|}{2^p} \right\rfloor$ and the signs of those samples $x[n]$ for which $\mu_p[n] \neq 0$. This information completely subsumes that associated with all previous (larger p) coding passes, so that there is no point in emitting them to the code-stream. The HT refinement passes, SP and MR, encode the same information as their J2K1 counterparts, which allows a J2K1 codestream to be transcoded to the HTJ2K format without altering its quantized sample representation in any way. To fully recover this information, an HT block decoder must process at most one CUP, one SP and one MR pass, whereas a J2K1 block decoder may need to process a large number of passes.

An HT block encoder may generate any number of coding passes, yielding a collection of candidate truncation points, with associated distortion and length implications. The PCRD-opt algorithm can then select optimal truncation points from the available passes. The PCRD-opt algorithm itself is identical to that of the EBCOT [12] algorithm on which JPEG 2000 is based, but truncating an HT code-block at pass t_0 leaves behind at most one CUP, one SP and one MR pass, having indices $t \in [3\lfloor t_0/3 \rfloor, t_0]$.

As suggested in Fig. 3, high throughput HTJ2K encoding systems may employ "Complexity Control" procedures to determine a small set of coding passes that are actually worth generating, avoiding those that are likely to be discarded by the PCRD-opt algorithm. For example, a two-pass strategy may be employed, in which a first pass estimates lengths and distortions associated with the coding passes without actually performing them, after which a crude PCRD-opt implementation can be used to identify one, or perhaps two, HT Sets to actually generate, followed by a final PCRD-opt step for precise rate control.

4. THE HT BLOCK CODING ALGORITHM

The HT SP and MR passes are similar to those of the J2K1 block coder, operating in its "Bypass" mode, where significance propagation, sign and magnitude refinement symbols are not subjected to entropy coding – this is still effective because these symbols tend to have nearly uniform probability distributions. However, the HT versions of these coding passes rearrange the symbols so as to allow much faster software-based decoding via lookup tables, while ensuring that hardware implementations can always decode the CUP, SP and MR passes concurrently.

The HP CUP pass is by far the most important, since the CUP pass associated with bit-plane p encodes all information from all coarser bit-planes. A complexity challenged decoder may opt not to process SP and MR passes, with a small sacrifice in quality, but the CUP pass must be decoded. The design of the CUP pass is the result of an iterative exploration of the trade-off between coding efficiency and achievable throughput on modern CPU platforms, with specific attention to the exploitation of SIMD instruction sets for vector processing.

The HT CUP pass employs three different entropy coding techniques to exploit the most important forms of redundancy amongst neighbouring sub-band samples. Fig. 4 illustrates most of the relevant quantities. One important form of redundancy exploitation is significance coding, whereby the binary significance symbols

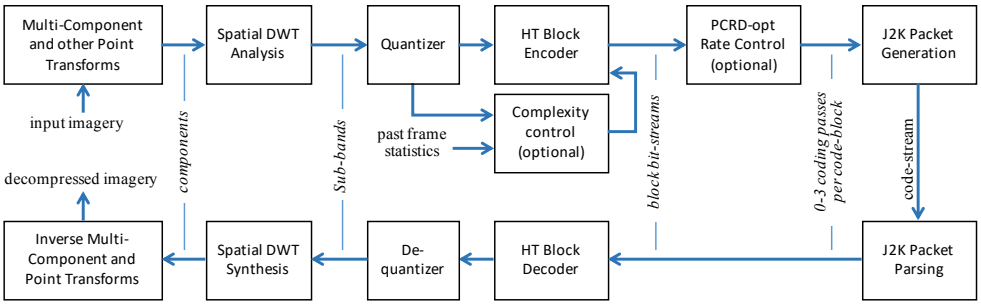

Fig. 3. High level architecture of FBCOT encoding and decoding processes.

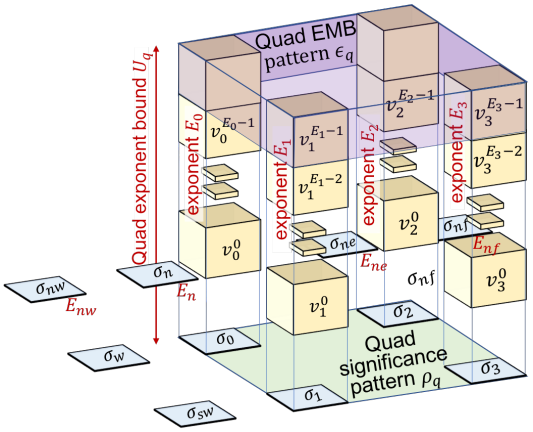

Fig. 4. Major quantities involved in the HT Cleanup coding pass.

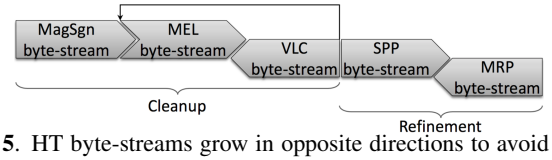

Fig. 5. HT byte-streams grow in opposite directions to avoid separate length signalling. MEL+VLC length encoded at end of Cleanup.

$\sigma[n] = \min\{1, \mu[n]\}$ of each sample[1] are subjected to variable length and adaptive coding. Samples are processed in 2×2 quads q, with an associated 4-bit significance pattern ρ_q, whose constituent significance symbols are denoted σ_0 through σ_3 in Fig. 4. The significance pattern ρ_q is coded with respect to a context c_q that depends on the significance of the quad's 6 causal neighbours; in the figure, these are denoted σ_{sw}, σ_w, σ_{nw}, σ_n, σ_{ne} and σ_{nf}, in clockwise order. Writing $\vee$ for the binary "OR" operator, we have

$$c_q = (\sigma_{nw} \vee \sigma_n) + 2(\sigma_{sw} \vee \sigma_w) + 4(\sigma_{ne} \vee \sigma_{nf}) \in [0, 7] \quad (1)$$

Quads for which $c_q = 0$ are treated specially as *All Zero Context* (AZC) quads, coding first a binary quad significance symbol $\sigma_q = \max\{1, \rho_q\}$, after which the significance pattern ρ_q is coded only if $\sigma_q = 1$.

A second important form of redundancy exploitation is magnitude exponent prediction. The magnitude exponent $E[n]$ associated with a significant sample n is defined by

$$E[n] = \begin{cases} 0 & \sigma[n] = 0 \\ \min\{E \in \mathbb{N} \mid 2^E > v[n]\} & \sigma[n] = 1 \end{cases} \quad (2)$$

where

$$v[n] = [2(\mu[n] - 1) + \text{sign}(x[n])] \cdot \sigma[n] \quad (3)$$

is a rearrangement of the sign and offset magnitude bits for sample n, known as its MagSgn value. $E[n]$ can be interpreted as the number of significant bits in $v[n]$, being 0 for insignificant samples. The HT CUP pass encodes a magnitude exponent bound U_q for each quad q, via an unsigned integer residual u_q. Specificallly,

$$U_q = u_q + \underbrace{\max\{1, \gamma_q \cdot \max\{E_{nw}, E_n, E_{ne}, E_{nf}\}\}}_{\kappa_q}. \quad (4)$$

The predictor κ_q depends on neighbouring exponents from the previous line, denoted E_{nw}, E_n, E_{ne} and E_{nf} in Fig. 4, together with γ_q, which is 1 for quads with more than one significant sample, and 0 otherwise. The bound U_q is required to be tight whenever the coded residual u_q is non-zero.

The third form of redundancy exploited by the HT CUP pass is the Exponent Mag Bound (EMB) pattern ϵ_q of each significant quad. As shown in the figure, ϵ_q can be understood as the most significant bit-plane of the U_q-bit representation of the quad's four MagSgn values. The EMB pattern is coded only when $u_q > 0$, since then the U_q bound is tight and all EMB bits are magnitude bits or 0; under these conditions, the EMB pattern ϵ_q cannot be 0 and generally exhibits significant statistical redundancy.

We now briefly introduce the three entropy coding tools of the HT CUP pass. The quad significance symbols σ_q for AZC quads q are coded using an adaptive run-length coder (MELCODE) similar to the one employed by the LOCO-I algorithm [13] employed by the JPEG-LS standard, but with only 13 states. The resulting code bits are packed into a separate *MEL bit-stream*.

For each non-AZC quad and each significant AZC quad, a context-adaptive variable length coding (CxtVLC) procedure is used to jointly encode the significance pattern ρ_q, a binary flag $u_q^{\text{off}} = \max\{1, u_q\}$, plus some or all of the quad's EMB pattern ϵ_q, using a separate VLC codebook for each context c_q. The CxtVLC codeword lengths are constrained to at most 7 bits, in order to simplify decoding. As a result, the EMB patterns associated with some combinations of c_q, ρ_q and $u_q^{\text{off}} = 1$ are only partially coded. The CxtVLC procedure is thus a variable-to-variable coding scheme, in which both codeword lengths and the number of symbols recovered from a decoded codeword are data dependent. It turns out that this adds no significant complexity to the algorithm, while allowing better exploitation of the redundancy associated with those symbols that are actually coded.

In software, CxtVLC encoding is best accomplished using a lookup table with 2048 entries, indexed by c_q, ρ_q and $\epsilon_q \cdot u_q^{\text{off}}$. In hardware, a much smaller lookup table can be employed, taking advantage of the fact that insignificant samples must have zero EMB

[1]Here, we drop the bit-plane p from our notation, interpreting $\mu[n] = \mu_p[n]$ as the sample's magnitude.

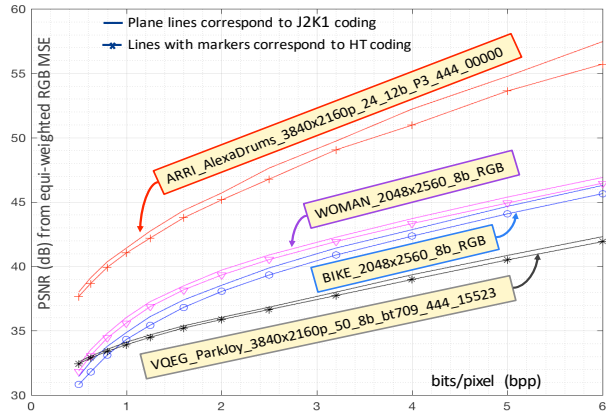

Fig. 6. Rate-distortion comparison of HT and J2K1 block coders in their optimal (default) modes. (BD-rate, BD-PSNR) values: Bike = (10.3%, −0.59dB); Woman = (6.8%, −0.39dB); Drums = (9.6%, −0.7dB); Park = (5.8%, −0.22dB).

bits. CxtVLC decoding can be accomplished using a lookup table with 1024 entries, indexed by c_q and the next 7 bits of the *VLC bit-stream*. It is worth emphasizing that only one lookup is required for every 4 samples, so hardware implementations with a throughput of 1 sample per clock cycle can interleave lookups from 4 separate code-blocks onto a single set of table resources. By contrast, implementations of the original JPEG standard require up to one VLC lookup per sample.

Whenever $u_q^{\text{off}} \neq 0$ the actual value of u_q must also be coded. A fixed variable length code, known as the *UVLC code*, is defined for this purpose, producing codewords with tightly bounded lengths. Prefix and suffix components of the UVLC codewords are interleaved across a pair of quads, so that software implementations can easily encode and decode a pair of quads together using a small lookup table. Together, the CxtVLC and UVLC codewords are interleaved into the *VLC bit-stream* on a quad-pair basis. Meanwhile, all bits from the MagSgn values $v[n]$ of significant samples, that cannot be deduced from the coded significance and EMB patterns are packed into a separate *MagSgn bit-stream*.

All HT bit-streams are subjected to byte-oriented bit-stuffing procedures to avoid the appearance of illegal 16-bit marker codes that are reserved by JPEG 2000 for resynchronization. Fig. 5 illustrates the complete set of resulting *byte-streams* that are associated with a full HT Set. These 5 separate byte-streams can be encoded and decoded asynchronously, which adds considerable concurrency to hardware implementations and also accelerates software deployments. All aspects of an encoder can be heavily vectorized on modern CPUs, with the exception only of MEL encoding, which consumes an average of around 0.25 clock cycles per sample on modern x86 processors, due to its compact state machine and the fact that AZC symbols are either rare or occur in runs.

Decoding of an HT CUP pass is best understood as a two-stage process. The first stage alternates on a line-pair basis between the steps of context formation and VLC decoding, assuming that MEL symbols have been decoded ahead of time, or batch processed. The first step is easily vectorized, while the second is inherently sequential, operating on 2x2 quads; nonetheless, the second step can be realized at a cost of approximately 3 clock cycles per sample on modern x86 processors. The second stage uses ρ_q, ϵ_q and u_q values produced by the first stage to reconstruct MagSgn values $v[n]$ from the MagSgn bit-stream. This stage also alternates on a line-pair basis between two steps: 1) formation of exponent predictors κ_q and bounds U_q, according to (4); and 2) unpacking bits from the MagSgn bit-stream, reconstructing MagSgn values and computing magnitude exponents for use in the next line-pair's first step. Both steps can be heavily vectorized.

Table 1. Throughputs obtained on a 4-core desktop platform with Intel i7-6700 (Skylake) CPU, 3.4GHz base clock, and 64-bit Ubuntu, processing the 292 frame full RGB HDR Drums test sequence – frame 0 was used in Fig. 6.

Conditions	ENC:J2K1	ENC:HT	DEC:J2K1	DEC:HT
2bpp vis i5x3	10.1 fps	81 fps	17.6 fps	148 fps = 3.7 GS/s
2bpp vis i9x7	9.7 fps	72 fps	17.2 fps	126 fps = 3.1 GS/s
2bpp mse i5x3	8.9 fps	75 fps	17.5 fps	130 fps = 3.2 GS/s
2bpp step i5x3	15.8 fps	114 fps	17.4 fps	126 fps = 3.1 GS/s
lossless r5x3	2.11 fps	67 fps	2.10 fps	67 fps = 1.7 GS/s

5. THROUGHPUT AND CODING EFFICIENCY RESULTS

Fig. 6 provides rate-distortion results for four readily available test images. Bike and Woman were amongst the most important images used in the original development of JPEG 2000; ParkJoy is part of a 4K VQEG test sequence derived from scanned film; and Drums is the first frame of an all digital 4K HDR test video, used in the development of a number of standards. The JPEG test file names appearing in the figure provide sufficient information to reproduce the results provided here. The comparison involves 64×64 code-blocks, 5 levels of irreversible 9×7 DWT and full PCRD-opt rate control, with 3 target bit-rates per octave. Bjontegaard delta-rate figures reported in the caption broadly confirm the conclusions of Fig. 1.

Table 1 reveals the very high throughput of HTJ2K, capable of reaching 3.7 Giga-samples/s at 2 bpp on a 4-core desktop CPU. The lossy results at 2 bpp correspond to a high quality, being above the maximum rate allowed for 4K Digital Cinema. We consider both the 9×7 transform of Part-1 and the irreversible LeGall 5×3 transform, using Part-2 transform extensions. Code-block dimensions are 32×128, which yields slightly higher throughput than 64×64, with less memory and lower latency. For the first 3 rows of the table, rate control is based on PCRD-opt, targeting either minimum MSE or visually weighted MSE (vis), with a complexity-control algorithm that is based on previous frame statistics [6] to avoid generating more than 2 HT Sets per code-block. The fourth row uses the fastest possible encoding strategy for video, where bit-rate is controlled loosely by quantization step size selection, and only one CUP pass is produced. All results are based on an experimental version of the popular Kakadu Software tools, with all hardware threads collaborating on each video frame to avoid the latency and memory demands of frame-parallel processing.

6. CONCLUSIONS

HTJ2K offers exceptionally high throughputs, with high coding efficiency, while preserving virtually all features of the JPEG 2000 family of standards. Decoding throughput appears to be higher than anything reported for the original JPEG algorithm, even for 8-bit content, while HTJ2K supports almost any precision, efficient region/resolution of interest decoding, interactive access via JPIP, and reversible transcoding to/from the quality progressive J2K1 representation, even on a block-by-block basis.

7. REFERENCES

[1] ISO/IEC, "15444-1:2016 Information technology – JPEG 2000 image coding system: Core coding system," 2016.

[2] ISO/IEC, "15444-2:2002 Information technology – JPEG 2000 image coding system: Extensions," 2002.

[3] ISO/IEC, "15444-9:2005 Information technology – JPEG 2000 image coding system: Interactivity tools, apis and protocols," 2005.

[4] D. Taubman and R. Prandolini, "Architecture, philosophy and performance of jpip: internet protocol standard for JPEG 2000," *Int. Symp. Visual Comm. and Image Proc.*, vol. 5150, pp. 649–663, July 2003.

[5] D. Taubman, A. Naman, R. Mathew, S-Y. Chien, and T-H. Tsai, "FBCOT: a fast block coding option for JPEG2000," Tech. Rep. M73021, ISO/IEC JTC1/SC29/WG1, October 2016.

[6] D. Taubman, A. Naman, and R. Mathew, "FBCOT: a fast block coding option for jpeg2000," in *SPIE Optics and Photonics: Applications of Digital Imaging*, August 2017, vol. 10396.

[7] JPEG Committee, "High throughput JPEG 2000 (HTJ2K): Call for proposals," Tech. Rep. N76037, ISO/IEC JTC1/SC29/WG1, June 2017, Available at: `https://www.jpeg.org`.

[8] D. Taubman, "FBCOT: Response to JPEG call for proposals on high throughput JPEG 2000," Tech. Rep. M79053, ISO/IEC JTC1/SC29/WG1, April 2018.

[9] D. Taubman, "HTJ2K CD1.3 contributrion: CxtVLC codeword extensions," Tech. Rep. M80018, ISO/IEC JTC1/SC29/WG1, July 2018.

[10] T. Richter, "Towards high-speed, low-complexity image coding: variants and modification of JPEG 2000," in *SPIE Optics and Photonics: Applications of Digital Imaging*, October 2012, vol. 849915.

[11] P. Enfedaque, F. Aulí-Llinàs, and J. C. Moure, "GPU Implementation of Bitplane Coding with Parallel Coefficient Processing for High Performance Image Compression," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 8, pp. 2272–2284, Aug. 2017.

[12] D.S. Taubman, "High performance scalable image compression with EBCOT," *IEEE Trans. Image Proc.*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[13] M.J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," *Proc. IEEE Data Compression Conf. (Snowbird)*, pp. 140–149, April 1996.