NOVEL 3D-WPP ALGORITHMS FOR PARALLEL HEVC ENCODING

Ziyu Wen[⋆]*, Bichuan Guo*[⋆]*, Jiashuo Liu*[⋆]*, Jisheng Li*[⋆]*, Yao Lu*[†]*, Jiangtao Wen*[⋆]

[⋆]Tsinghua University [†]University of California, San Diego

ABSTRACT

Although wavefront parallel processing (WPP) proposed in the HEVC standard and various inter frame WPP algorithms can achieve comparatively high parallelism, their scalability for its parallelism is still very limited due to various dependencies introduced in spatial and temporal prediction in HEVC. In this paper, we propose three types of 3 Dimensional WPP (3D-WPP) algorithms that can significantly improve the parallelism, while achieving good tradeoffs between implementation complexity, determinism, and rate-distortion (RD) performance. Experimental results show that the proposed algorithms can lead to up to 2.8x speed up compared with existing inter frame WPP methods. While the Simple 3D-WPP and Static 3D-WPP algorithm may introduce an BD rate loss between 0 to 4.9% as compared with existing algorithms, the more complex Dynamic 3D-WPP algorithm achieves better parallelism with virtually no coding performance loss.

Index Terms— HEVC, 3D-WPP, Wavefront Parallel Processing

1. INTRODUCTION

The High Efficiency Video Coding (HEVC) standard developed by the Joint Collaborative Team on Video Coding (JCT-VC) can achieve 50% bitrate savings as compared with H.264/AVC [1] [2]. Due to its high computational complexity, parallel processing for HEVC is especially important. Several parallel processing friendly tools, such as tiles and wavefront parallel processing (WPP) were introduced. Among these, WPP has been reported to achieve a good tradeoff between parallelism and coding performance [3]. Later, several inter-frame WPP methods were introduced to further improve the parallelism scalability [3][4][5]. However, the actual parallelism achieved by the existing methods is still quite limited, e.g. 13 for 1080p video and 7 for 720p video in our experiment.

In this paper, we propose an inter frame WPP framework named 3-Dimensional WPP (3D-WPP), where the delay of the wavefronts between frames may be Largest Coding Units (CTUs) rather than in rows. This leads to a truly 3 dimensional (time, x, y) wavefront of concurrently running CTUs and a large increase in parallelism. The main contribution of this paper are:

1) We proposed Simple 3D-WPP and Static 3D-WPP algorithm with significant speed-ups and limited RD (rate-distortion) loss compared to the existing algorithms.

2) We analyzed the RD performance of the existing inter frame WPP algorithms and proposed a Dynamic 3D-WPP that archives superior performances in both the speed and quality aspects.

Email: {wen-zy13, gbc11, liujs13, lijs12}@mails.tsinghua.edu.cn, luyao@ucsd.edu, jtwen@tsinghua.edu.cn. This work is supported by the National Science Fund for Distinguished Young Scholars of China (Grant No. 61125102), the State Key Program of National Natural Science of China (Grant No. 61133008) and Nanjing Yunyan Information Technology Ltd.

3) We analyzed the in-loop deblocking and SAO (Sample Adaptive Offset) filters in inter frame WPP, and proposed a compatible method with 3D-WPP.

The paper is organized as the follows. In Section 2, we briefly review approaches for parallel processing in HEVC and H.264/AVC encoding. 3D-WPP is described in Section 3, including simple, static and dynamic algorithms as well as in-loop filters. Section 4 contains the experimental results, while Section 5 is the conclusions.

2. BACKGROUND

2.1. Wavefront parallel processing (WPP)

In wavefront parallel processing of HEVC [6], the encoding of a largest coding unit (CTU) is only dependent on the reconstructed CTU on its top-right, and its previous CTU. In the CTU level, by initializing the context model for each line, and getting the prediction information from the top-right CTU, each CTU can start encoding immediately after its top-right CTU is encoded, forming a wavefront of CTUs that can be processed in parallel. Compared with slices and tiles, the absence of a hard boundary in WPP leads to significant RD performance improvement while maintaining good parallelism [3].

In reality, the actual encoding time of each CTU by different processing units is highly unbalanced, as shown in Fig. 2 for a 1080p frame. As a result, the theoretical limits of WPP parallelism can hardly be reached in practical systems.

2.2. Related Work

For H.264/AVC, Zhao et al. proposed a wavefront parallel encoding method in [7] and [8]. A similar implementation of this approach is included in the x264 [9] open source H.264 encoder. Finchelstein et al.[10] examined multicore and larger reference area for H.264 decoding. Azevedo and Meenderinck et al.[11][12] introduced inter frame wavefront for H.264 decoding as well as an analysis of various parallel approaches in H.264/AVC. Zhao et al.[13] described a coding unit (CU) level parallel approach for HEVC intra encoding, while Shen and Chen et al.[14][15] introduced a fast H.264 to HEVC transcoder with wavefront parallelism. Chi et al.[3] thoroughly examined the parallel scalabilities and efficiencies of different HEVC parallelization approaches, and proposed a inter-frame wavefront method named Overlapped WaveFront (OWF), mainly from a decoder's perspective. Chen et al.[4] extended OWF to a method named Inter-Frame Wavefront (IFW) by introducing multiple B frames in the inter frame parallelism pipeline, while SAO is disabled. The open source x265 encoder [5] implements the inter-frame wavefront parallelism inherited from x264[9]. However, our experiments show that both [4] and [5] suffer from limited parallel scalability.

2.3. OWF and IFW

In this section, we briefly introduce the existing wavefront parallel algorithms, including (intra frame) WPP, OWF [3], and IFW [4]. In this paper, IFW refers to the algorithm in [4], while inter frame WPP is used in contrast to intra frame WPP.

WPP, OWF, IFW and our proposed methods in later section share the same reference and dependency map for intra prediction. So here we only discuss the reference and dependency for inter predictions. $C_{i,j,k}$ represents the k-th CTU in j-th line of i-th frame in encoding order (i, j, and k start from 0). $Ref_F(C_{i,j,k})$ is the reference area for inter prediction of $C_{i,j,k}$ using method F. Therefore,

$$\begin{aligned} Ref_{WPP}(C_{i,j,k}) = \{C_{i',j',k'} | \\ 0 \le i' < i, 0 \le j' < H, 0 \le k' < W\} \end{aligned} \tag{1}$$

$$\begin{aligned} Ref_{OWF}(C_{i,j,k}) = \{C_{i',j',k'} | \\ i' = i-1, 0 \le j' \le j + L_H, 0 \le k' < W\} \end{aligned} \tag{2}$$

$$\begin{aligned} Ref_{IFW}(C_{i,j,k}) = \{C_{i',j',k'} | \\ i' \in RefList(i), 0 \le j' \le j + L_H, 0 \le k' < W\} \end{aligned} \tag{3}$$

where W and H are the frame width and height in CTU, and *RefList(i)* is the set of reference frames for frame i. L_H is small non-negative integer parameter representing a safe range, which is about 1/4 frame height in CTU in OWF, and is set as small as 0 or 1 in IFW.

According to the reference area, we can find the dependent set $Dep_F(C_{i,j,k})$ for inter prediction of $C_{i,j,k}$ after transitive reduction:

$$Dep_{WPP}(C_{i,j,k}) = \{C_{i-1,H-1,W-1}\} \tag{4}$$

$$Dep_{OWF}(C_{i,j,k}) = \{C_{i-1,j+L_H,W-1}\} \tag{5}$$

$$Dep_{IFW}(C_{i,j,k}) = \{C_{i',j+L_H,W-1} | i' \in RefList(i)\} \tag{6}$$

The open source project x265[5] has a similar mechanism as IFW.

3. 3D WAVEFRONT PARALLEL PROCESSING

In this section, we will introduce the highly paralleled 3D-WPP algorithms, with compatible in-loop deblock and SAO filters. The Simple 3D-WPP is easier to implement. The Static 3D-WPP extends the reference area and hence improves the RD performance while keeping the process deterministic. Finally, the Dynamic 3D-WPP compensates for almost all the RD losses introduced by 3D parallelism and is optimal in both speed and quality.

3.1. Dependency of 3D-WPP

The existing methods, namely OWF and IWF, settle the reference and dependency in rows. For example, $Dep_{OWF,IWF}(C_{i,0,0}) = \{C_{i',L_H,W-1}\}$, which means the encoding process of frame i can only be started after the reconstruction of $C_{i',L_H,W-1}$, when almost half of the frame i' has been reconstructed in WPP pattern. However, CTUs near $C_{i',L_H,W-1}$ cannot be referenced by $C_{i,0,0}$. So such reference and dependency are not only redundant but also reduce the parallelism.

$$\begin{aligned} Dep_{Sim3D}(C_{i,j,k}) = Dep_{Sta3D}(C_{i,j,k}) = Dep_{Dyn3D}(C_{i,j,k}) \\ = \{C_{i',j+L_H,k+L_W} | i' \in RefList(i)\} \end{aligned} \tag{7}$$

The dependency set of 3D-WPP is given in Eq.7. In 3D-WPP, $C_{i,j,k}$ can start encoding right after $C_{i',j+L_H,k+L_W}$ is reconstructed, where L_H and L_W are small non-negative integers indicating the safe area. The reduction of redundant dependency results in significant improvement in parallelism.

3.2. Simple 3D-WPP

Our proposed Simple 3D-WPP algorithm achieves significantly larger parallelism than the existing algorithm. Using Simple 3D-WPP, the reference area is simply restricted in a rectangular area given in Eq. 8, which is easy and simple to implement.

$$\begin{aligned} Ref_{Sim3D}(C_{i,j,k}) = \{C_{i',j',k'} | i' \in RefList(i), \\ 0 \le j' \le j + L_H, 0 \le k' \le k + L_W\} \end{aligned} \tag{8}$$

Fig.1 shows the dependency and reference of previous algorithms and 3D-WPP. For simplicity, we only demonstrate a reference frame pair *(Ref(erence), Cur(rent))*, which could be (I, P), (I, B), (P, P), (P, B) or (B, B). An arrow $A \rightarrow B$ indicates that CTU B depends on CTU A. The reference area of Simple 3D-WPP can be represented as Area 1. We can see in the figure that, given the same *Ref*, compared to previous algorithms, the *Cur* of 3D-WPP can not only have more CTUs to process, but also enable other frames referencing Cur to start encoding, both of which improve the parallelism.

3.3. Static 3D-WPP

In Static 3D-WPP, all the CTUs which are guaranteed to be reconstructed when $C_{i',j+L_H,k+L_W}$ is done, can be referenced by $C_{i,j,k}$, as described in Eq. 9. Note that the $Ref_{Sta3D}(C_{i,j,k})$ is the largest possible reference area to be reconstructed deterministically.

$$\begin{aligned} Ref_{Sta3D}(C_{i,j,k}) = \{C_{i',j',k'} | i' \in RefList(i), \\ 0 \le j' \le j + L_H, 0 \le k' < W, \\ 2k' - j' \ge 2k + 2L_W - j - L_H\} \end{aligned} \tag{9}$$

In Fig.1, compared to Simple 3D-WPP, Area 2 is the additional reference area utilized by Static 3D-WPP. Such enlarged reference area contributes to RD performance gain, while introducing implementation complexity.

While implementing the restriction of reference area in HEVC, motion vectors (MV) in the Advanced Motion Vector Prediction (AMVP) mode should be clipped into the reference area, and Merge Candidates who have MVs out of the reference area should be discarded. It is desirable to check more merge candidates, in case some of them are discarded due to 3D-WPP.

It is also important to set earlier frames in encoding order and upper rows in one frame with higher priority, to maximize the parallelism.

3.4. Dynamic 3D-WPP

While the Simple and Static 3D-WPPs have exploited the parallelism in the dependency map, the reference area can be dynamically enlarged using the algorithm in this subsection to further improve RD performance.

$$\begin{aligned} Ref_{Dyn3D}(C_{i,j,k}) = \{C_{i',j',k'} | C_{i',j',k'} \text{ is } reconstructed\} \\ = Ref_{Sta3D}(C_{i,j,k}) \cup \{C_{i',j',k'} | C_{i',j',k'} \in Aera3\} \end{aligned} \tag{10}$$

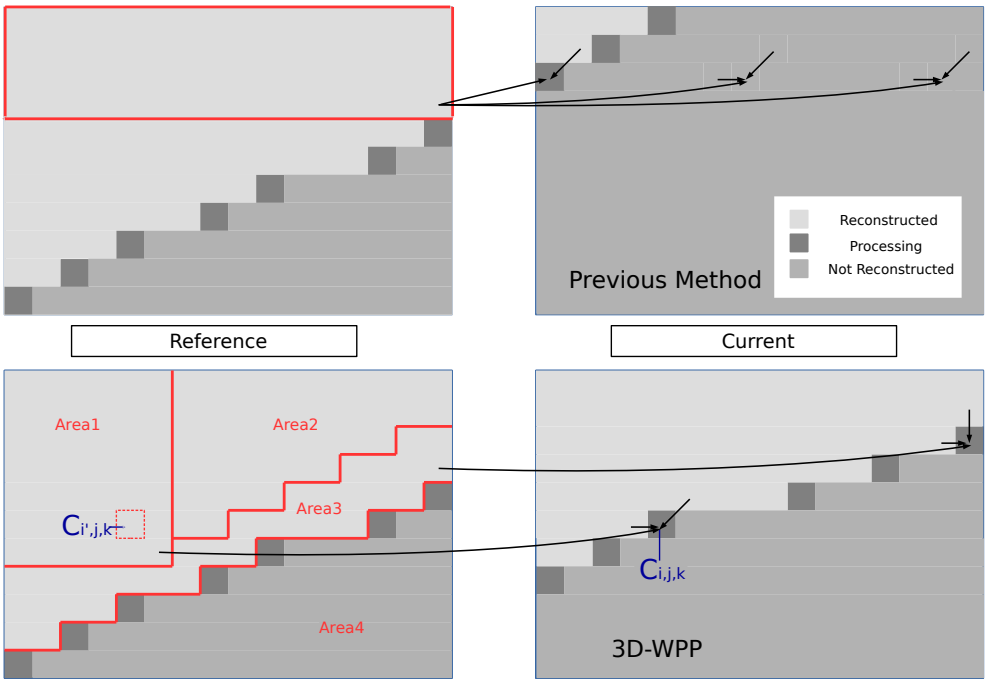

Fig. 1. Dependency and reference of previous method (upper) and 3D-WPP (bottom)

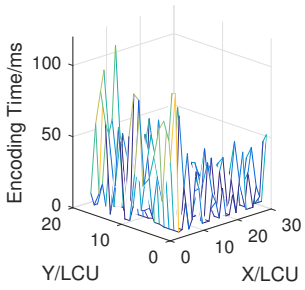

Fig. 2. Encoding Time of CTUs

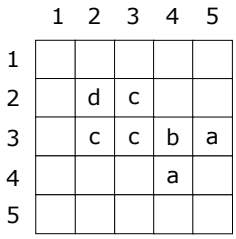

Fig. 3. Deblock Filter and SAO

As in Eq.10, when encoding $C_{i,j,k}$, all the CTUs which are reconstructed can be referenced. As shown in Fig.1, the Area 3 is the additional reference area utilized by Dynamic 3D-WPP. The boundary between Area 3 and Area 4 changes dynamically. Area 3 exists when 1) the encoding time of each CTU is unbalanced, as is usually the case; and 2) when the parallelism the encoder achieves exceeds the number of cores of the machine, which is content dependent.

The core idea of Dynamic 3D-WPP is to turn the inevitable parallelism loss from unbalanced CTU time and the surplus parallelism in a finite-core machine into RD performance gain. Such optimization is very useful when encoding speed is of higher priority than quality, such as in the real-time encoding scenario.

Experiments in the later section shows that even a small slice of Area 3 may contribute a lot to the RD performance. And since checking whether $C_{i',j',k'}$ is reconstructed does not require a mutex lock, no multi-threading overhead is introduced in this mechanism.

Algorithm 1 3D-WPP with In-Loop Filter

for Each $C_{i,j,k}$ **do**
  for Each $i' \in$ *RefList(i)* **do**
    wait until $C_{i',j+L_H,k+L_W}$ is reconstructed
  end for
  encode $C_{i,j,k}$
  deblock $C_{i,j-1,k}$ (vertical), $C_{i,j-1,k-1}$ (horizontal)
  compute SAO parameters for $C_{i,j-1,k-1}$
  apply SAO filter, extend CTU border for $C_{i,j-2,k-2}$
  mark $C_{i,j-2,k-2}$ as reconstructed and **issue signal**
  if $k = W - 1$ or $j = H - 1$ **then**
    do filter for the remaining CTUs in the row or colomn
    mark and **issue signal**
  end if
end for

3.5. In-Loop Filters

In [5], the encoder has to wait for all CTUs in a row to be encoded before applying the deblocking filter, and the SAO filter introduces another full row of delay. In [4], SAO is disabled to reduce delay.

As shown in Algorithm 1, instead of applying row filters to row j, 3D-WPP applies CTU filters to each CTU $C_{i,j,k}$. A vertical deblocking filter is applied first, followed by a horizontal deblocking filter after one column delay. Since HEVC's deblocking decisions are made based on a four-sample segment between adjacent blocks, we need adjacent blocks to be encoded beforehand to apply the deblock filter. Specifically, we deblock $C_{i,j-1,k}$ vertically and $C_{i,j-1,k-1}$ horizontally right after encoding $C_{i,j,k}$.

After horizontally deblocking a CTU, the encoder computes its SAO parameters and applies it. A few bottom rows and rightmost columns of the CTU are excluded in SAO parameter computation, since their pixels won't be decided until adjacent CTUs are deblocked. This further reduces the row/column delay in the sense that we can compute a CTU's SAO parameters as soon as itself is deblocked. However, since it requires all deblocked pixels to apply the SAO filter, the previous exclusion method can not be used for SAO filter application, resulting in a higher delay comparing to SAO computation.

The processes are pipelined according to their row delays and column delays. Each row uses its own entropy coder for SAO RD cost optimization, working in a similar pattern as wavefronts.

For example, as in Fig.3, when $C_{i,4,4}$ finishes its encoding, $C_{i,3,5}$ have also been encoded (marked as a). Therefore we can apply the vertical deblocking filter to $C_{i,3,4}$ (marked as b), and the horizontal deblocking filter to $C_{i,3,3}$. This means CTUs above $C_{i,3,3}$ (e.g. $C_{i,2,3}$) or on its left side (e.g. $C_{i,3,2}$) have also been deblocked (marked as c). As explained above, we can compute their SAO parameters. One more row and column delay are added for applying the SAO filters, marked as d in the figure.

Table 1. BD rate (%) compared with previous method

Frame Type	Size	Video	Prev1	Sim1	Sta1	Dyn1	Prev2	Sim2	Sta2	Dyn2
IPPPPP	2160p	Cobra	0.00	0.08	0.06	-0.03	-0.10	-0.10	-0.10	-0.11
	2160p	Suzie	0.00	1.54	0.91	-0.11	-0.40	-0.35	-0.39	-0.40
	1080p	Tennis	0.00	1.26	0.86	0.31	-0.07	0.41	0.38	0.39
	1080p	RedKayak	0.00	0.08	0.07	0.05	-0.02	0.03	0.03	0.03
	720p	DucksTakeOff	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01
	720p	Parkrun	0.00	0.08	0.08	0.08	0.00	0.08	0.08	0.08
IBBBBP	2160p	Cobra	0.00	4.92	3.67	1.19	-1.67	2.17	0.37	-0.27
	2160p	Suzie	0.00	3.14	1.93	-0.47	-1.50	-0.30	-0.59	-1.51
	1080p	Tennis	0.00	2.37	1.74	0.16	-0.97	-0.51	-0.47	-0.79
	1080p	RedKayak	0.00	0.35	0.35	-0.09	-0.20	-0.15	-0.15	-0.22
	720p	DucksTakeOff	0.00	0.01	0.01	0.02	-0.01	0.01	0.01	0.01
	720p	Parkrun	0.00	0.15	0.15	0.16	0.00	0.16	0.16	0.16

Table 2. Speed-up ratio (x) of previous method and 3D-WPP

Frame Type	Size	Video	Prev1	Sim1	Sta1	Dyn1	Prev2	Sim2	Sta2	Dyn2	Dyn1/Prev1
IPPPPP	2160p	Cobra	18.81	32.20	32.19	31.92	17.64	31.89	31.88	31.67	1.70
	2160p	Suzie	16.85	29.50	29.29	29.47	15.31	29.00	29.18	29.07	1.75
	1080p	Tennis	10.14	23.01	23.01	22.69	9.54	18.92	18.93	18.64	2.24
	1080p	RedKayak	9.95	24.64	24.72	24.47	9.40	20.55	20.47	20.28	2.46
	720p	DucksTakeOff	6.01	12.75	12.74	12.67	5.68	10.47	10.44	10.38	2.11
	720p	Parkrun	6.18	13.05	13.07	12.99	5.88	10.79	10.76	10.71	2.10
IBBBBP	2160p	Cobra	29.71	32.06	31.96	31.65	28.50	31.90	31.98	31.68	1.07
	2160p	Suzie	27.69	30.04	29.60	29.90	25.95	29.96	30.33	29.92	1.08
	1080p	Tennis	13.36	28.61	28.32	28.39	12.72	27.85	27.76	27.45	2.13
	1080p	RedKayak	13.29	30.56	30.52	30.15	12.77	29.81	29.77	29.32	2.27
	720p	DucksTakeOff	7.98	18.57	18.58	18.43	7.56	15.65	15.74	15.57	2.31
	720p	Parkrun	7.10	20.06	20.12	19.96	6.75	16.87	16.86	16.73	2.81

We have shown that the deblock and SAO filter and can be reduced from row level to CTU level, and applied in 3D-WPP pattern. Options like applying SAO with pixels before deblock filter can further reduce the delay, at the cost of RD loss.

4. EXPERIMENTS

Since it is difficult to implement inter frame WPP on the HEVC Test Model (HM) [16] which has a fixed GOP structure without frame level parallelism, we have implemented 3D-WPP on x265[5] v1.4. The Intra period was set to 120, the CTU size was 64. The encodings were conducted using the medium preset, with QPs $\{22, 27, 32, 37\}$. The experiments was conducted on a 36-core (2 x Intel Xeon E5-2699 v3 @ 2.30GHz, hyper-threading off) server with 128GB RAM.

Table 2 shows the speed up ratio compared to a single thread run of the previous methods. The index Prev denotes the previous algorithms, either IFW or the x265 pattern. The index Sim, Sta and Dyn denotes Simple, Static and Dynamic 3D-WPP respectively. The index *x* in Prev*x* or Dyn*x* denotes $L_H = x$ in IFW or $L_H = L_W = x$ in 3D-WPP.

In the experiment, 3D-WPP achieved up to 1.7x speed-up for 4k, 2.4x for 1080p and 2.8x speed-up for 720p video compared with previous algorithms. For 4k video, the overall speed-up ratio was almost identical to the number of cores of our machine. The three methods of 3D-WPP achieved almost the same parallelism, while increasing L_H and L_W slightly reduced the parallelism.

Table 1 shows the RD loss of existing algorithms and 3D-WPP. In most cases, Simple and Static 3D-WPP have good RD performance compared with previous method. When they do not, the Dynamic 3D-WPP made up up to 4% in BD rate loss of 3D-WPP, and in some case had better RD performance than the previous algorithms. In addition, increasing L_H and L_W by one, even in Prev and Sim, could lead to significant RD improvement, which could also be attributed to Area 3, where a small additional reference area contributed significantly to the quality.

From the comparison with Prev1 and Dyn2, we can conclude that the 3D-WPP has not only higher parallelism but also better RD performance than the existing strategies.

5. CONCLUSIONS

In this paper, we proposed three 3D-WPP algorithms for parallel HEVC encoding. Compared with the existing inter frame WPP techniques like IFW and the x265 mechanism, the proposed algorithms improve the level of parallelism with modest RD performance losses.

Experimental results show that 3D-WPP has up to 2.8x speed-up compared to the previous algorithms. The Simple 3D-WPP is easy to implement with good quality in most cases. The Static 3D-WPP improves the RD performance while maintaining determinism. And the Dynamic 3D-WPP is the best in both speed and quality, compensating almost all the RD loss from 3D-parallelism.

Further optimizations of the 3D-WPP algorithms on multicore system in terms of task scheduling, spatial and temporal locality, cache misses, and etc. are also important.

6. REFERENCES

[1] G.J. Sullivan, J.R. Ohm, W.J. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[2] J.R. Ohm, G.J. Sullivan, H. Schwarz, T.K. Tan, and T. Wiegand, "Comparison of the Coding Efficiency of Video Coding Standards–Including High Efficiency Video Coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

[3] Chi Ching Chi, Mauricio Alvarez-Mesa, Ben Juurlink, Gordon Clare, Félix Henry, Stéphane Pateux, and Thomas Schierl, "Parallel scalability and efficiency of HEVC parallelization approaches," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1827–1838, 2012.

[4] Keji Chen, Jun Sun, Yizhou Duan, and Zongming Guo, "A novel wavefront-based high parallel solution for HEVC," *Circuits and Systems for Video Technology, IEEE Transactions on*.

[5] "x265," `http://x265.org`.

[6] F Henry and S Pateux, "Wavefront parallel processing," Tech. Rep., Tech. Rep. JCTVC-E196, 2011.

[7] Zhuo Zhao and Ping Liang, "A highly efficient parallel algorithm for H. 264 video encoder," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, 2006, vol. 5, pp. V–V.

[8] Zhuo Zhao and Ping Liang, "Data partition for wavefront parallelization of H. 264 video encoder," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*. IEEE, 2006, pp. 4–pp.

[9] "x264," `http://www.videolan.org/developers/x264.html`.

[10] Daniel F Finchelstein, Vivienne Sze, and Anantha P Chandrakasan, "Multicore processing and efficient on-chip caching for H. 264 and future video decoders," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 19, no. 11, pp. 1704–1713, 2009.

[11] Arnaldo Azevedo, Ben Juurlink, Cor Meenderinck, Andrei Terechko, Jan Hoogerbrugge, Mauricio Alvarez, Alex Ramirez, and Mateo Valero, "A highly scalable parallel implementation of H. 264," in *Transactions on High-Performance Embedded Architectures and Compilers IV*, pp. 111–134. Springer, 2011.

[12] Cor Meenderinck, Arnaldo Azevedo, Ben Juurlink, Mauricio Alvarez Mesa, and Alex Ramirez, "Parallel scalability of video decoders," *Journal of Signal Processing Systems*, vol. 57, no. 2, pp. 173–194, 2009.

[13] Yanan Zhao, Li Song, Xiangwen Wang, Min Chen, and Jia Wang, "Efficient realization of parallel HEVC intra encoding," in *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. IEEE, 2013, pp. 1–6.

[14] Tong Shen, Yao Lu, Ziyu Wen, Linxi Zou, Yucong Chen, and Jiangtao Wen, "Ultra fast H. 264/AVC to HEVC transcoder," in *Data Compression Conference (DCC), 2013*. IEEE, 2013, pp. 241–250.

[15] Yucong Chen, Ziyu Wen, Jiangtao Wen, Minhao Tang, and Pin Tao, "Efficient software H.264/AVC to HEVC transcoding on distributed multicore processors," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 25, no. 8, pp. 1423–1434, 2015.

[16] "HEVC Test Model (HM) ," `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/`.