HIGH-THROUGHPUT AND LOW-COST HARDWARE-ORIENTED INTEGER TRANSFORMS FOR HEVC

Trang Thi Thu Do, Yih Han Tan and Chuohao Yeo

Institute for Infocomm Research

ABSTRACT

To achieve a target bit-rate reduction of 50% over H.264/AVC while maintaining equivalent perceptual video quality, High Efficiency Video Coding (HEVC) includes several new coding tools including a new set of integer transforms. Since these transforms are more complex than the H.264/AVC transforms, it is more challenging to design and develop high-performance integer transform hardware for HEVC. In this paper, we propose a series of high-throughput and low-cost hardware-oriented HEVC transform algorithms by using a butterfly structure and replacing multiplications by additions and shift operations in a way that minimizes critical computation paths. Compared to the algorithms using other methodologies like multiplierless multiple constant multiplication (MMCM) or decomposition to sparse matrices, our algorithms achieve around 20% shorter critical paths while consuming relatively small numbers of additions and shift operations. Hardware designs applying our proposed algorithms can increase their throughput by 20% while maintaining a reasonable resource consumption compared to when applying other algorithms.

Index Terms— High Efficiency Video Coding (HEVC), Discrete Cosine Transform (DCT), Discrete Sine Transform (DST), Critical Path Optimization, Multiplierless Multiple Constant Multiplication (MMCM).

1. INTRODUCTION

The Discrete Cosine Transform (DCT) has been one of the most essential tools in image and video coding due to its ability to perform near-optimal de-correlation [1]. However, the DCT uses floating point arithmetic, which could cause mismatches between its forward and inverse transforms. In order to avoid mismatches and to avoid the complexity of the floating point arithmetic, integer transforms that are integer approximations of the DCT are used in the latest video coding standards such as H.264/Advanced Video Coding [2] and High Efficiency Video Coding (HEVC) [3].

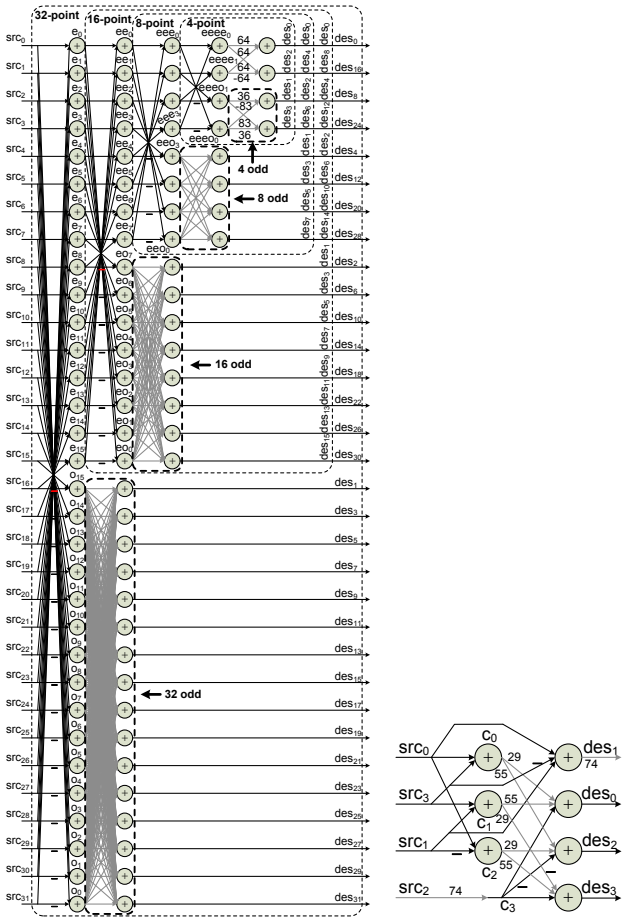
Besides the DCT-based integer transforms, HEVC uses an alternative transform which is an integer approximation of the Discrete Sine Transform (DST) for coding the luma residual blocks in intra-picture prediction modes [4].

HEVC is more complex than its predecessor H.264/AVC due to the inclusion of tools to achieve its target of a 50% bit-rate reduction over H.264/AVC for equal perceptual video quality [3]. HEVC decoder complexity is about 1.5 times of that of H.264/AVC [5], while HEVC encoders are expected to be several times more complex than H.264/AVC encoders [6]. Regarding transform complexity, HEVC supports up to 32×32 transforms with 7-bit multipliers while H.264/AVC only supports up to 8×8 transforms with 4-bit multipliers. The maximum input bit-depth to a 1-D transform in HEVC is 16 [3], while that of H.264/AVC is 12 [2]. Due to these increased operational requirements, it is more challenging to design high-performance integer transform hardware for HEVC. It should be noted that performance of integer transform designs is mostly measured by throughput, which is computed as the amount of data processed in one unit of time, since integer transform designs having higher throughput can support the encoding and decoding of higher resolution videos. For designs that have further approximations or pruning for the forward transforms, peak signal-to-noise ratio (PSNR) or Bjøntegaard distortion-rate (BD-rate) [7] can be used as an additional factor for performance measurement.

To achieve high throughput with fixed data sizes, transform processing time needs to be minimized. In hardware designs, processing time depends on critical path lengths of transform designs. Therefore, for high throughput, we need to minimize the critical paths of the transform algorithms.

There have been previous reported works on high throughput and/or efficient integer transforms for HEVC [8–16]. Except for some of the designs which mainly targets resource sharing for area efficiency ([8–10]), most of them target high throughput by simplifying implementation algorithms and minimizing operation count, i.e., resource consumption.

In particular, there have been one reported design that exploits the similarity between the H.264/AVC and HEVC 8×8 transform matrices to develop the 8×8 transform [11]. As H.264/AVC only supports the transform sizes of up to 8×8, this method is not applicable to the HEVC larger size transforms like 16×16 and 32×32. The remaining designs decompose the transforms into odd and even parts together with butterfly blocks like the Partial Butterfly (PB) implementation in the HEVC reference software, or HM [17]. To implement the odd parts, the designs use either (1) multiplier-



(a) 4, 8, 16, 32-point PB DCT algorithms (b) Fast DST algorithm

Fig. 1. Data flows of the forward transform algorithms used in the HEVC reference software. *src* (source or input) and *des* (destination or output) are residual and transformed data, respectively. Each arrow represents a data flow. Each gray arrow represents a data flow with a multiplication.

less multiple constant multiplication (MMCM) [18] to replace multiplications with additions and shift operations in a way that minimizes the operation count [15, 16]; or (2) decompose odd matrices into sparse matrices for ease of implementation [12–14]. By using MMCM, the first option efficiently saves hardware resources. By using sparse matrices, the second option easily implements the transforms by additions and shift operations instead of multiplications. It also enables hardware sharing for multiple coding standards [12].

However, simplifying the implementation algorithms and minimizing operation count does not ensure the shortest critical paths for the algorithms. This is because in hardware implementation, operations can be designed to perform in parallel. In this paper, we provide a series of high-throughput and low-cost hardware-oriented HEVC integer transform algorithms which have critical computation path lengths of 80% of those of the best competitors, and consume reasonable numbers of additions and shift operations. The algorithms support the HEVC 4×4, 8×8, 16×16 and 32×32 DCTs, and 4×4 DST.

The rest of the paper is organized as follows. The HEVC transforms and their implementation in the HM are reviewed in Section 2. The proposed forward transform implementation algorithms are introduced in Section 3. Results are discussed, followed by the concluding remarks in Section 4.

2. INTEGER TRANSFORMS

2.1. Integer Transforms in HEVC

Like in H.264/AVC, the transform operations in HEVC are based on the DCT, following Equations 1 and 2.

$$W = H_f X H_f^T, \quad (1) \qquad X' = H_i W' H_i^T, \quad (2)$$

where H_f and H_i are the core forward/inverse transform matrices; X and W are residual and transformed blocks, which are input and output of forward transform, respectively; and W' and X' are de-quantized and reconstructed residual blocks, which are input and output of inverse transform, respectively.

The core inverse transform matrices (H_is) were derived by approximating the coefficients of the scaled inverse DCT matrices to integer values under the considerations of maximizing the precision and proximity to orthogonality and limiting the dynamic range for transform computation [3]. The forward transform matrices (H_fs) used in the HM are the transpose of the H_is. In total, there are four inverse transform matrices and four corresponding forward transform matrices to support four different transform sizes. The four forward DCT matrices can be found in [19].

The HEVC DST also follows Equations 1 and 2, where its forward matrix is the transpose of its inverse matrix. The forward DST matrix can be found in [3].

2.2. Integer Transforms in HEVC Reference Software

A 2-D forward/inverse transform of a block can be computed by repeatedly applying 1-D forward/inverse transform algorithms to all the rows and columns of the block. The forward transformation of a residual block includes two stages. In the first stage, each row of residual data is 1-D transformed by applying a 1-D transform algorithm; while in the second stage, all the columns of the result in the first stage are transformed using the same algorithm. Figure 1 shows the original 1-D Partial Butterfly (PB) forward DCTs and the fast 1-D forward DST used in the HM. Each DCT algorithm can be divided into a butterfly block, an even part and an odd part. In Figure 1a, the butterfly blocks and the even parts are on the left and on top of the odd parts, respectively. It is found that the 16-, 8- and 4-point PB algorithms have the same data flow structures with the even parts of the 32-, 16- and 8-point algorithms, respectively. Table 1 shows the operation counts and critical

Table 1. Resource consumptions and critical path lengths of the 1-D transform algorithms in HEVC reference software.

HM transform	Resource			Critical Path		
algorithms	Muls	Adds	Total*	Muls	Adds	Total*
4-point Partial Butterfly	6	8	44	1	2	8
8-point Partial Butterfly	22	28	160	1	3	9
16-point Partial Butterfly	86	100	616	1	4	10
32-point Partial Butterfly	342	372	2424	1	5	11
4-point fast DST	11	8	74	1	3	9

Muls: multiplications; Adds: addition/subtractions; *: Total add count when only adds (without muls) are used. As the multipliers are smaller than 2^7, 6 2-input adds can be used to implement each mul.

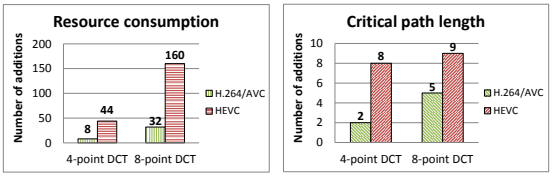


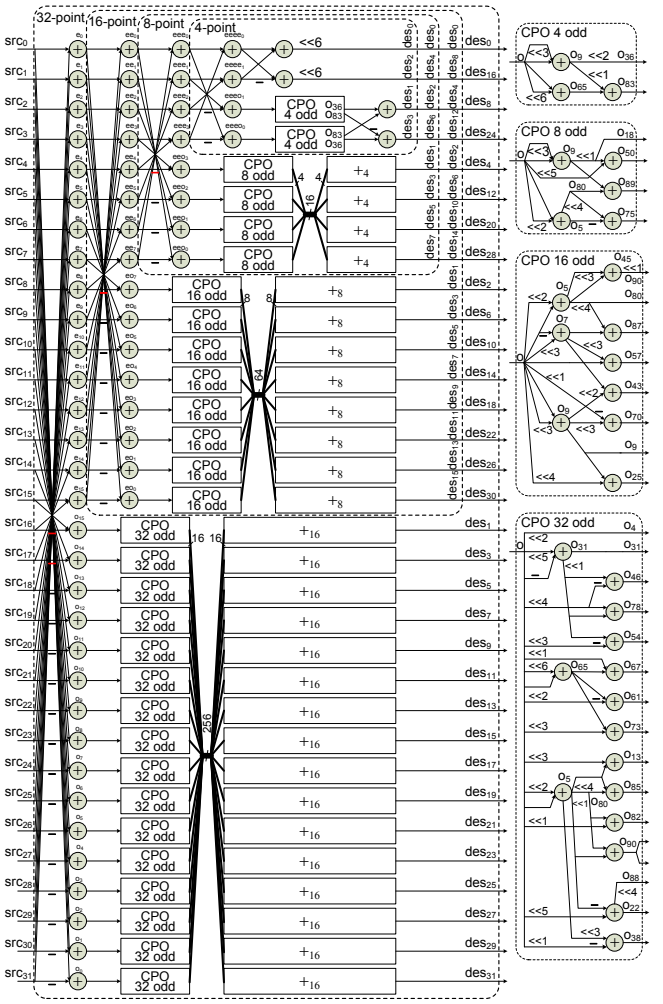
Fig. 2. Resource consumptions and critical path lengths of the 4- and 8-point transform algorithms used in the H.264/AVC and HEVC reference softwares.

path lengths of the 1-D transform algorithms in the HM. As can be seen, when the transform size increases, the operation count also increases drastically. Figure 2 further illustrates the significant increase in the resource consumption and critical path length of the HEVC transforms over those of H.264/AVC [20] when comparing the 4- and 8-point algorithms.

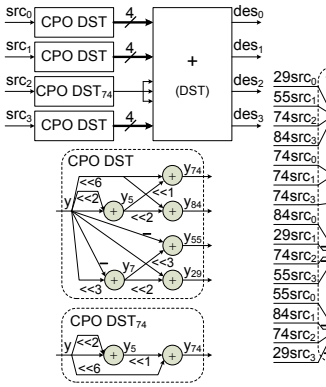
Due to the complexity of HEVC transforms, much design effort is needed to achieve high-throughput transforms for HEVC. To achieve high throughput, processing time for the transform algorithms needs to be minimized. In hardware implementation, processing time depends on critical path length. Therefore, transform implementation algorithms which has the shortest critical computation path and consume a reasonable resource is strongly desired.

3. PROPOSED TRANSFORM DESIGNS

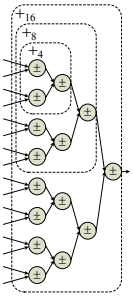
Taking critical path minimization into account, we reimplement the HEVC transform algorithms by using just additions and shifts. For the DCTs, like the MMCM and sparse matrix methods, we first decompose the transforms using the partial butterfly structure. We then compute the minimum critical path length and find an algorithm that achieves it with a reasonable operation count for each of the odd parts. The method to find these algorithms starts with multiplication-to-addition conversions that minimize the operation counts for each multiplication. It then arranges the operations into the binary adding tree for each multiplication to ensure achieving the minimum critical path length. It finally re-arranges the



(a) Proposed DCTs. A "CPO N odd" block has $\frac{N}{2}$ outputs, with output $o_n = n \times o$, where o is the input and n is an integer multiplier. Addition block $+_M$ (Figure 3c) has M inputs which are connected to M outputs from M different CPO $2M$ blocks. Please refer to HM [17] for inputs and addition/subtractions required in the addition blocks.



(b) Proposed DST. Blocks CPO DST and DST_{74} output $y_n = n \times y$, where y is the their input and n is an integer multiplier. Block $+_{(DST)}$ has 15 inputs corresponding to the outputs from the CPO blocks. Block $+_4$ (Figure 3c) for the DST consists only additions.



(c) Addition blocks. Each operator can do addition or subtraction.

Fig. 3. Proposed forward transform algorithms for HEVC.

Table 2. Resource consumptions and critical path lengths of the proposed 1-D CPO transform algorithms for HEVC.

	Resource		Critical Path	
Proposed transform algorithms	Adds	Shift	Adds	Shift
4-point CPO DCT	14	8	4	2
8-point CPO DCT	54	28	5	2
16-point CPO DCT	198	92	6	2
32-point CPO DCT	710	268	7	2
4-point CPO DST	31	24	4	2

Adds: 2-input addition/subtractions; CPO: critical path optimization.

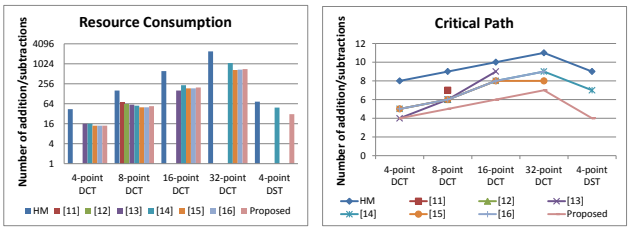


Fig. 4. Resource consumptions and critical path lengths of different algorithms. Missing bars/points at an algorithm means that the algorithm does not support the transform.

operations to maximize the common operation counts among the multiplications to further reduce the total operation count. Since each of the odd parts always has a longer minimum critical path than its associated even part, the overall transform algorithms found will achieve the minimum critical path lengths among algorithms using the partial butterfly structure. For the DST, we also use the same methodology to find an algorithm that achieves the minimum critical path length for its multiple constant multiplication.

For example, the 4-point odd part (Figure 1a) has $eeeo_1$ and $eeeo_0$ as the inputs and des_1 and des_3 as the outputs. Each input needs to be multiplied by 36 and 83 before being added or subtracted. The multiplication by 36 can be implemented as $x \times 36 = x \ll 5 + x \ll 2$. The multiplication by 83 can be implemented as $x \times 83 = (x \ll 6 + x \ll 4) + (x \ll 1 + x)$, which needs at least 2 addition/subtraction (add) stages to implement. The critical path of the 4-point odd part then includes a) at least 2 add stages for multiplications by 36 and 83 and b) 1 add stage for adding 2 branches of multiplications by 36 and 83. Therefore, the minimum critical path length of the 4-point odd part is 3 adds. Since this is longer than that of the 4-point even part (1 adds), the minimum critical path length of the 4-point DCTs using the butterfly structure is the total of that of the odd part and 1 add stage for the butterfly, which becomes 4 adds.

Figure 3 illustrates the proposed critical-path-optimization (CPO) implementation for the HEVC transform algorithms. Table 2 shows their operation counts and critical path lengths. Our algorithms achieve the minimum critical path lengths for the transform algorithms using the PB structure.

Table 3. Comparison of resource consumption and critical path length among different HEVC 1-D transform algorithms.

Types$^{\#}$	HM$^{\&}$	[11]	[12]	[13]$^{\approx}$	[14]	[15]	[16]	Pr*
4-p DCT	44/8	-	-	16/4	16/5	14/5	14/5	14/4
8-p DCT	160/9	72/7	60/6	60/6	56/6	50/6	50/6	54/5
16-p DCT	616/10	-	-	160/9	232/8	186/8	186/8	198/6
32-p DCT	2424/11	-	-	-	1080/9	666/8	682/9	710/7
4-p DST	74/9	-	-	-	49/7	-	-	31/4

Each cell contains resource consumption/critical path length of an algorithm, measured by addition/subtraction count. $^{\#}$: Transform types; $^{\&}$: The algorithms used in the HEVC reference software; $^{\approx}$: The design neglects some complex parts of the transforms; *: The proposed algorithms; -: the algorithm does not support the corresponding transform.

4. DISCUSSIONS AND CONCLUSIONS

Table 3 and Figure 4 show that our transforms achieve the shortest critical paths and consume relatively small resources compared to those of the reported HEVC 1-D transform algorithms [11–16] and the original algorithms used in the HM.

As can be seen from Table 3, our DCT algorithms require critical path lengths of 57% while consuming resources of 32% of those of the PB algorithms in the HM on average. Our 4-point DCT is the best among the reported designs as it requires a critical path length of 80% of that of [14–16], while consuming a resource amount of 88% of that of [14] or equal to that of [15, 16]. Compared to [13], which uses transform approximation, our 4-point DCT requires 88% operation count while having the same critical path length. Our DST also achieves a critical path of 57% while consuming only 63% resource compared to those of [14].

Our 8-, 16- and 32-point DCTs require 5, 6 and 7 adds in their critical paths, which are 17%, 25% and 22% less than that of the best critical path competitors for the 8-point DCT [12–16], for the 16-point DCT [14–16] and for the 32-point DCT [15], respectively. Our operation counts are also smaller than that of most of the designs. However, they are from 6% to 8% more than that of the least-resource-consumption designs [15] for the 8-, 16- and 32-point DCTs and [16] for the 8- and 16-point DCTs.

In this paper, we proposed a set of implementation algorithms for the HEVC transforms including the 4×4, 8×8, 16×16, and 32×32 DCTs and 4×4 DST. Compared to the reported algorithms, our algorithms achieve around 20% shorter critical paths while consuming relatively small numbers of additions and shift operations. Shorter critical path leads to shorter processing time, faster speed, and subsequently higher throughput. Hardware designs when applying our proposed algorithms can increase their throughput by 20% while maintaining a reasonable resource consumption compared to when applying other implementation algorithms.

5. REFERENCES

[1] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computer*, vol. C-23, no. 1, pp. 90-93, Jan. 1974.

[2] ITU-T and ISO/IEC, "ITU-T Rec. H.264 and ISO/IEC 14496-10:2009: Advanced Video Coding," Mar. 2010.

[3] G.J. Sullivan, J. 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*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[4] C. Yeo, Y.H. Tan, Z. Li and S. Rahardja, "Mode-Dependent Transforms for Coding Directional Intra Prediction Residuals," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 4, pp. 545-554, Apr. 2012.

[5] Y.J. Ahn, W.J. Han, and D.G. Sim, "Study of Decoder Complexity for HEVC and AVC Standards based on tool-by-tool comparison," in *Proceedings of the SPIE, Applications of Digital Image Processing XXXV*, vol. 8499, article id. 84990X, Oct. 2012.

[6] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC Complexity and Implementation Analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685-1696, Dec. 2012.

[7] G. Bjøntegaard, "Calculation of Average PSNR Differences between RD Curves," ITU-T Q6/16 Doc. VCEG-M33, Apr. 2001.

[8] M. Budagavi and V. Sze, "Unified forward+inverse transform architecture for HEVC," in *Proceedings of the 19th IEEE International Conference on Image Processing (ICIP)*, pp. 209-212, Sep.-Oct. 2012.

[9] M. Budagavi, A. Fuldseth, G. Bjøntegaard, V. Sze and M. Sadafale, "Core Transform Design for the High Efficiency Video Coding (HEVC) Standard," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1029-1041, Dec. 2013.

[10] P.-T. Chiang and T.S. Chang, "A reconfigurable inverse transform architecture design for HEVC decoder," *IEEE International Symposium on Circuits and Systems (ISCAS'13)*, pp. 1006-1009, May 2013.

[11] M. Martuza and K. Wahid, "A cost effective implementation of 8×8 transform of HEVC from H.264/AVC," *Proceedings of 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE '12)*, pp. 1-4, May 2012.

[12] R. Rithe, C.-C. Cheng, and A.P. Chandrakasan, "Quad Full-HD Transform Engine for Dual-Standard Low-Power Video Coding," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 11, pp. 2724-2736, Nov. 2012.

[13] F. Belghith, H. Loukil, and N. Masmoudi, "Free multiplication integer transformation for the HEVC standard," in *Proceedings of the 10th IEEE International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 1-5, Mar. 2013.

[14] C. Fan, F. Li, G. Shi, L. Zhou, and H. Yang, "A Low Complexity Multiplierless Transform Coding for HEVC," in *Proceedings of 13th Pacific-Rim Conference on Multimedia*, pp. 578-586, Dec. 2012.

[15] M. Tikekar, C.-T. Huang, C. Juvekar, V. Sze, and A.P. Chandrakasan, "A 249-Mpixel/s HEVC Video-Decoder Chip for 4K Ultra-HD Applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 61-72, Jan. 2014.

[16] P.K. Meher, S.Y. Park, B.K. Mohanty, K.S. Lim, and C. Yeo, "Efficient Integer DCT Architectures for HEVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 1, pp. 168-178, Jan. 2014.

[17] JCT-VC, "HM Reference Software," available online: `https://hevc.hhi.fraunhofer.de/svn/svn?HEVCSoftware/`

[18] Y. Voronenko and M. Puschel, "Multiplierless multiple constant multiplication," ACM Transaction on Algorithms, vol. 3, no. 2, pp. 11, May 2007.

[19] A. Fuldseth, G. Bjøntegaard, M. Budagavi and V. Sze, "CE10: Core transform design for HEVC," JCTVC-G495, Nov. 2011.

[20] T.T.T. Do and T.M. Le, "High Throughput Area-Efficient SoC-Based Forward/Inverse Integer Transforms for H.264/AVC," *IEEE International Symposium on Circuits and Systems (ISCAS'10)*, pp. 4113-4116, May 2010.