A Novel Transform Accelerator With Fast Kernel Selection and Efficient Transform Circuit

Zhijian Hao, Chenlong He, Jiaming Liu, *Graduate Student Member, IEEE*, Qi Zheng, Jinchang Xu, Peijun Ma, Xiaohua Ma, *Member, IEEE*, and Yue Hao, *Senior Member, IEEE*

***Abstract*— The introduction of multiple transform types into the Versatile Video Coding (VVC) standard has yielded notable encoding gains but also resulted in substantial computational burdens, posing two critical challenges for hardware implementation: fast kernel selection and efficient transform computation design. Existing studies typically address these challenges in isolation, lacking a holistic solution for VVC transform coding. In this paper, we presents a groundbreaking transform accelerator that unifies transform kernel selection and multiple transform circuit within a single framework. In terms of algorithms, driven by mechanistic analysis, we propose a decision tree-based kernel selection algorithm that ensures both high decision accuracy and computational efficiency. Additionally, we design a transfer matrix-based approximation algorithm for Discrete Sine Transform Type-7 and a matrix decomposition-based improved computation for Discrete Cosine Transform Type-2, significantly reducing the computational complexity. On the hardware front, we implement a high-precision and area-efficient transform accelerator, which integrates highly pipelined kernel selection and transform computation architectures. With multiple reuse and parallelism strategies, the accelerator demonstrates substantial resource efficiency advantages. Experimental results reveal that the proposed accelerator achieves a circuit resource reduction of over 44% with a slight performance degradation, while maintaining processing capabilities up to 8K@57 fps. To the best of our knowledge, this is the first comprehensive hardware solution for VVC transform coding that jointly addresses the challenges of kernel selection and transform circuit design.**

***Index Terms*— Video coding, transform accelerator, decision-tree, sparsity optimization, pipelined design.**

I. Introduction

TRANSFORM coding has been widely integrated in video coding due to its efficient spatial decorrelation capability. Cooperating with other coding tools [1], [2], [3], [4], it significantly reduces spatial correlations and achieves high compression rates while maintaining minimal quality loss.

Received 18 December 2024; revised 4 February 2025; accepted 16 February 2025. Date of publication 25 February 2025; date of current version 30 May 2025. This work was supported by China Postdoctoral Science Foundation under Grant 2024M762536. This article was recommended by Associate Editor X. Zeng. *(Corresponding authors: Xiaohua Ma; Yue Hao.)*

Zhijian Hao, Peijun Ma, Xiaohua Ma, and Yue Hao are with the State Key Discipline Laboratory of Wide Band-Gap Semiconductor Technology, School of Microelectronics, Xidian University, Xi'an 710071, China (e-mail: haozhijian@xidian.edu.cn; pjma@xidian.edu.cn; xhma@xidian.edu.cn; yhao@xidian.edu.cn).

Chenlong He, Jiaming Liu, and Qi Zheng are with the State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai 200433, China (e-mail: clhe22@m.fudan.edu.cn; liujm22@m.fudan.edu.cn; qzheng21@m.fudan.edu.cn).

Jinchang Xu is with the School of Computer Science, Peking University, Beijing 100871, China (e-mail: jinchang_xu@pku.edu.cn).

Digital Object Identifier 10.1109/TCSI.2025.3543575

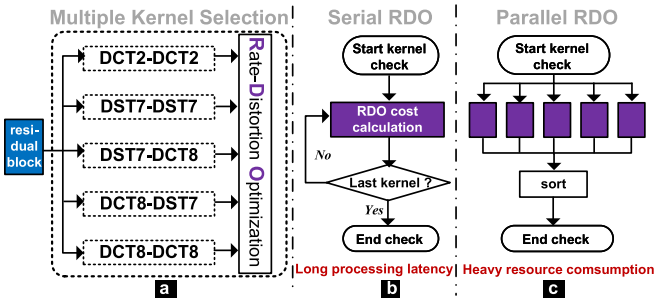


Fig. 1. (a) The Multiple Transform Selection in VVC. (b) Serial RDO for kernel selection. (c) Parallel RDO for kernel selection.

The Karhunen–Loève Transform (KLT) has been proven to be the most effective in removing spatial correlation from image signals. However, its practical application in real-world encoding is severely limited by high computational complexity, as it requires the calculation of eigenvalues and eigenvectors of the input signal. It was later proved in [5] that the Discrete Cosine Transform Type-2 (DCT2) can optimally approximate the KLT and is also more computationally efficient, resulting in widespread adoption in H.264 [6] and H.265. In 2012, Han et al. [7] demonstrated that for image signals modeled by a first-order Gaussian-Markov process, the Discrete Sine Transform Type-7 (DST7) provides better decorrelation than DCT2. This distribution assumption also aligns well with the residual signals generated by the diverse prediction modes in modern codecs. Consequently, the DST7 has been integrated within Versatile Video Coding (VVC) alongside DCT2 [8]. Furthermore, VVC also introduces a symmetric counterpart of DST7, known as DCT8. These three transforms serve as primary transform types, working collaboratively to achieve more effective decorrelation performance. Based on these three primary transform types, the VVC standard defines the transform scheme depicted in Fig. 1a, termed Multiple Transform Selection (MTS). MTS consists of five distinct transform kernels, each generated by the combination of two primary transforms in both the horizontal and vertical directions: DCT2-DCT2, DST7-DST7, DST7-DCT8, DCT8-DST7, and DCT8-DCT8. Considering the trade-off between complexity and coding gain, the combination of DCT2 with the other two types was not included in the MTS [9]. When encoding the residual block, the optimal transform kernel is selected from these five candidates using a Rate-Distortion Optimization (RDO) strategy. Consequently, MTS introduces two formidable hardware design challenges.

Challenge 1: Long Processing Latency or Heavy Resource Consumption When Deciding the Best Kernel. Typically, MTS utilizes RDO cost for decision-making and the cost calculation is computation-intensive, requiring the forward and inverse transforms/quantization and bitrate estimation operations. As shown in Fig. 1b and 1c, two implementation schemes are typically considered in the hardware design: serial processing and parallel processing. In serial processing, a single RDO cost circuit is employed with time-division multiplexing to make decisions, leading to long processing latency. In contrast, parallel processing instantiates five RDO cost calculation circuits, simultaneously evaluating the RDO costs of all five transform kernels to accelerate decision-making with a heavy resource consumption. Therefore, making efficient decisions within limited cycles and resources remains a critical challenge that requires urgent resolution.

Challenge 2: Heavy Resource Overhead of Transform Circuits. The MTS requires hardware implementations for transform types beyond DCT2, resulting in a threefold increase in resource consumption compared to the single DCT2 design. Given the inherently computation-intensive of transform operations, this leads to a significant rise in computational cost. Therefore, designing a single circuit to handle the computation for all three transform types has become a key concern.

These two challenges are critical to the hardware implementation efficiency of MTS. Several research efforts have been made to address these issues. **Regarding Challenge 1**, existing approaches can be broadly classified into two categories: ***Skip-based scheme*** and ***selection-based scheme***.

For skip-based scheme. Fu et al. [10] introduced a two-stage algorithm based on encoded information in 2019, achieving a Bjøntegaard Delta-Bit Rate (BD-BR) [11] loss of 0.16% and a 23% reduction in encoding time. In 2022, Saldanha et al. [12] employed decision trees to determine whether to skip MTS encoding, with a BD-BR loss of 0.23% and a timesaving of 11%. Later, Hao et al. [13] proposed a probability model-based skip algorithm for AV1 that accelerates the encoding process while preserving more encoding gains. It is crucial to note that when the skip conditions are not satisfied, these algorithms may fail to skip any transform kernels. In such a worst case, it necessitates checking all transform kernels, resulting in no reduction in hardware resource usage. Additionally, the variable computation cycles introduced would lead to pipeline bubbles, making such scheme impractical for hardware implementation.

For selection-based algorithms. In 2018, Lu et al. [14] used a sparse Laplacian matrix to estimate the approximate bitrate and introduced a fast selection method. However, due to the insufficient optimization of the weight parameters, this approach incurred significant encoding performance loss. In 2019, Su et al. [15] used neural networks to make decision among 16 transform kernels for AV1, achieving high precision but at the cost of excessive computational complexity, which hindered hardware implementation. That same year, Wang et al. [16] employed corner pixel selection for transform kernels but still retained three transform types, offering limited optimization in terms of hardware resource utilization.

Overall, skip-based scheme do not consistently reduce hardware resource consumption, while selection-based scheme are primarily designed for software acceleration and fail to effectively balance encoding performance with hardware overhead. As a result, there is still a lack of an efficient, hardware-friendly transform kernel selection approach.

In addressing Challenge 2, many studies have focused on designing efficient transform circuits, which can be broadly categorized into two approaches: ***independent optimization*** of individual transform type and ***joint optimization*** of multiple transform types.

Independent Optimization *of* ***Individual*** *Transform Circuits*. Due to the symmetry between DST7 and DCT8, they can share a common transform circuit. Consequently, optimization efforts primarily focused on the DCT2 [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29] and DST7 [30], [31], [32], [33], [34], [35] circuits. The classic optimization algorithm for **DCT2** is the butterfly algorithm [22], which reduces the number of multipliers by leveraging the symmetry of the transform matrix. Ahmed et al. [23] further reduced hardware resource usage by decomposing the DCT2 matrix into sub-matrices. Additionally, multiplier-free techniques [24], [25], [26] have been employed to reduce the hardware cost of remaining multipliers. Recent researches have focused on exploiting the reuse characteristics in the DCT2 computation process. Hao et al. [27] implemented a single circuit that supports 4∼64 point DCT2 transforms by treating small-sized transform matrices as subsets of larger ones. Zheng et al. [28] and Hao et al. [29] designed data reordering schemes that allow different transform sizes to share the same even-odd decomposition circuit. For the optimization of **DST7**, Mert et al. [30] developed a high-performance architecture supporting 4×4 and 8×8 sizes. Garrido et al. [31], [32] explored multiple FPGA implementation methods for DST7, including solutions based on general multipliers and shift-adders. Kammoun et al. followed a multiplier-free design principle and presented efficient one-dimensional (1D) and two-dimensional (2D)architecture designs in references [33] and [34]. To minimize hardware costs in multiplier-free designs, Fan et al. [35] used the N-Dimensional Reduced Adder Graph (RAG-n) algorithm to reduce the number of adders required, leading to significant resource savings.

Joint Optimization *of* ***Multiple*** *Transform Circuits*. The goal of joint optimization is to achieve multiple transform functions using a single circuit, thus optimizing resource usage to the maximum extent. This type of optimization typically relies on approximation techniques [36], [37]. Zeng et al. [36] proposed a method that simplifies the complexity of multiplications by adjusting matrix elements to the multiples of 2^3. Ahmed et al. [37] suggested converting the inverse DCT2 results into approximate forward DST7 results. While this approach achieves high approximation accuracy, it requires introducing inverse transform circuits into the forward transform pipeline, which ultimately does not lead to resource savings. Furthermore, this method necessitates modifications to the decoder, making it impractical for real-world applications. In general, the current approaches to

joint optimization of multiple transform circuits fall short of balancing coding performance and hardware constraints.

The above reviews on these two challenges underscore the pressing need for efficient hardware solutions for the fast transform kernel selection and the efficient transform circuit design. To bridge these gaps and deliver a comprehensive, high-performance solution for VVC transform coding, this paper introduces a novel transform accelerator, embedding a high-precision fast transform kernel selection and area-efficient multiple transform architectures. The key contributions of this work are summarized as follows:

- **A Decision-Tree-Based Transform Kernel Selection Scheme.** Through an in-depth exploration of the energy concentration mechanism in transform processes, we derive the necessary and sufficient conditions for achieving perfect energy concentration and introduce the concept of the Frequency Matching Factor (FMF). Based on this foundation, we develop a decision tree approach for efficient transform kernel selection. This method not only ensures hardware efficiency but also significantly enhances decision accuracy, demonstrating notable advantages over existing techniques.
- **Algorithm-Hardware Co-Optimization for Efficient DST7 and DCT2 circuit Designs.** This paper introduces two novel optimization techniques for DST7 and DCT2. Firstly, by defining a transfer matrix between DCT2 and DST7 and leveraging the least squares method to enhance its sparsity, we achieve a low-cost, high-precision computation of DST7. Secondly, we apply a matrix decomposition algorithm to effectively reduce the computational complexity of DCT2. These algorithmic innovations lead to highly efficient and resource-optimized transform circuit designs.
- **Deeply-Pipelined and Area-Efficient Transform Accelerator.** Building on the aforementioned optimizations, we design a deeply pipelined transform accelerator that seamlessly integrates both the transform kernel selection and transform circuits. Through the optimization of circuit microstructures and the application of various parallelism and reuse strategies, the proposed accelerator significantly outperforms existing solutions in terms of resource utilization, data throughput, and power consumption.

The rest of this paper is organized as follows. Section II describes the essential background of VVC transforms. Section III introduces the proposed algorithmic optimizations for kernel selection and transform circuit in detail. Section IV presents the related hardware design and implementation. Section V shows the results, which demonstrate the advantages of the proposed accelerator over existing works in the literature. Section VI concludes this paper.

II. Background

The MTS contains three primary transform types: DCT2, DST7, and DCT8. The following provides a brief overview of the transform, using DCT2 as a representative example.

TABLE I
BASIS FUNCTIONS OF DCT2/8 & DST7

Transform Types	Basis Function, $n, k = 0, 1, ..., N-1$
DCT2	$r(n,k) = \varepsilon_k \cdot \sqrt{\frac{2}{N}} \cdot cos(\frac{\pi \cdot k \cdot (2n+1)}{2N})$, where $\varepsilon_k = \begin{cases} \sqrt{\frac{2}{N}}, k = 0 \\ 1 \quad , k \neq 0 \end{cases}$
DST7	$r(n,k) = \sqrt{\frac{4}{2N+1}} \cdot sin(\frac{\pi \cdot (2k+1) \cdot (n+1)}{2N+1})$
DCT8	$r(n,k) = \sqrt{\frac{4}{2N+1}} \cdot cos(\frac{\pi \cdot (2k+1) \cdot (2n+1)}{4N+2})$

The 1D and 2D DCT2 are defined as (1) and (3),

$$Y(k) = \sum_{n=0}^{N-1} x(n) r(n.k), \tag{1}$$

$$\mathbf{r(n,\ k)} = \sqrt{\frac{2}{N}} \varepsilon_k cos\left[\frac{k(2n+1)\pi}{2N}\right], \tag{2}$$

$$F(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) r(m,n,k,l), \tag{3}$$

$$\mathbf{r(m,\ n,\ k,\ l)} = \epsilon_k \epsilon_l cos\left[\frac{k(2m+1)\pi}{2N}\right] cos\left[\frac{l(2n+1)\pi}{2N}\right],$$
$$\epsilon_k = \epsilon_l = \begin{cases} \sqrt{1/N}, & k,l = 0, \\ \sqrt{2/N}, & else \end{cases} \tag{4}$$

where m, n is the index of residual samples, k, l is the index of transform coefficients, and $m, n, k, l = 0, 1, \ldots, N-1$. $Y(k)$ and $F(k,l)$ are the transformed outputs, and N is the transform size. $r(n,k)$ and $r(m,n,k,l)$ are defined as the basis functions of 1D and 2D DCT2. Table I shows the basis functions of the three transform types employed in VVC. In practical video coding, by scaling and rounding to integer while preserving orthogonality, the 1D and 2D transforms can be expressed in matrix forms as (5) and (6):

$$Y = T \times X^T, \tag{5}$$

$$Y = T \times X \times T^T, \tag{6}$$

where T is the transform matrix and varies with transform types. It can be seen that the transform process is inherently a computation-intensive matrix multiplication. Thus, the optimization of the transform architecture relies on refining the corresponding matrix multiplications.

When reconstructing the residual block, the 2D inverse transform is performed according to (7):

$$x(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) s(m,n,k,l), \tag{7}$$

$$s(m,n,k,l) = r(m,n,k,l) = \epsilon_k \epsilon_l cos\left[\frac{k(2m+1)\pi}{2N}\right] cos\left[\frac{l(2n+1)\pi}{2N}\right], \tag{8}$$

where $s(m,n,k,l)$ is the basis function of the inverse transform. (7) reflects that the essence of the transform is to

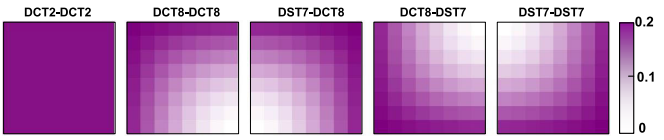


Fig. 2. The primary frequency basis images of five transform kernels.

decompose the spatial block into a set of weighted combinations of images in the frequency domain, and the transformed coefficient is the weight. It could be clearer by modifying the symbols used as (9):

$$X = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) S_{kl}, \tag{9}$$

S_{kl} is defined as the basis image corresponding to the frequency index (k, l). In this paper, we name the basis image at the position of $(0, 0)$ as the primary frequency basis image. When expanded into one-dimensional vectors, these basis images are orthogonal to each other [38]. Fig. 2 visually presents the primary frequency basis images of different transform kernels at the size of 8×8.

III. Hardware-Oriented Algorithmic Optimizations

This section provides a detailed description of the algorithmic optimizations targeting reducing the resource consumption, including a novel transform kernel selection algorithm, an approximate algorithm for DST7, and an area-efficient calculation scheme for DCT2.

A. Decision-Tree-Based Transform Kernel Selection

This section introduces a lightweight, high-precision solution for transform kernel selection.

The core function of a transform is to concentrate spatially scattered energy into low-frequency components in the frequency domain, with the kernel achieving the highest energy concentration deemed optimal. Through rigorous analysis, we determine that perfect energy concentration, where all energy is focused on the frequency position of $(0, 0)$, occurs if and only if the residual block is a scaled version of the primary frequency basis image. This condition can be mathematically formulated as:

$$X = f * S_{00}, \tag{10}$$

where X is the residual block, S_{00} is the primary frequency basis image as defined in (9), and f is a scaling factor with arbitrary values. The following is the formal proof of this assertion.

Necessity: Given that the transform process is lossless, the total spatial energy equals the total frequency energy:

$$\sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x^2(m,n) = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F^2(k,l) = E, \tag{11}$$

where E is the total energy, referring to the sum of squares of all terms in the spatial or frequency block. Assuming that the energy is concentrated at $(0, 0)$:

$$F(0,0)^2 = E, \tag{12}$$

with (11) and (12), the following equation (13) can be derived:

$$\sum_{k=0}^{N-1} \sum_{\substack{l=0 \\ (k,l)\neq(0,0)}}^{N-1} F^2(k,l) = E - F^2(0,0) = 0. \tag{13}$$

Therefore, it can be concluded as (14),

$$\begin{cases} F(k,l) \neq 0, & (k,l) = (0,0) \\ F(k,l) = 0, & (k,l) \neq (0,0) \end{cases} \tag{14}$$

Substituting (14) into (9), we can get:

$$X = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) S_{kl} = F(0,0) * S_{00}. \tag{15}$$

Therefore, (10) is satisfied, and $f = F(0, 0)$ in this case. Thus, the necessity is proven.

Sufficiency: With (10) and (9), we can get:

$$X = f * S_{00} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} F(k,l) S_{kl} \tag{16}$$

thus,

$$\left[f - F(0,0)\right] * S_{00} + \sum_{k=0}^{N-1} \sum_{\substack{l=0 \\ (k,l)\neq(0,0)}}^{N-1} F(k,l) S_{kl} = 0, \tag{17}$$

as mentioned in (9), the $S_{kl}s$ are orthogonal to each other. Therefore, (17) holds if and only if (18) is met.

$$\begin{cases} F(0,0) = f, \\ F(k,l) = 0, & (k,l) \neq (0,0). \end{cases} \tag{18}$$

It elucidates that only at the position of $(0, 0)$ are the coefficients nonzero, which means the residual block energy is concentrated on $F(0, 0)$. Thus, the sufficiency is proven.

When X is the scaling of the primary frequency basis image of the current transform kernel, the perfect energy concentration can be achieved. This conclusion easily extends to the notion that **the greater the similarity between the residual block and the scaling of the primary frequency basis image, the more pronounced the energy concentration effect becomes**.

Previous analysis reveals the importance of the similarity. To effectively characterize this similarity, we introduce the concept of FMF that we first proposed in [13]:

$$\begin{aligned} FMF &= \left| cos(\vec{X}, \vec{S_{00}}) \right| \ll 6, \\ &= \left| \frac{\vec{X} \cdot \vec{S_{00}}}{|\vec{X}| \times |\vec{S_{00}}|} \right| \ll 6. \end{aligned} \tag{19}$$

FMF is defined as the absolute value of cosine similarity between the residual block X and the primary frequency basis image S_{00}. In (19), $\vec{X}$ and $\vec{S_{00}}$ refer to the 1D vectors obtained by flattening X and S_{00}, respectively. It has been demonstrated to exhibit a strong correlation with the RDO cost in AV1 in [13]. To evaluate its effectiveness in VVC, we extract scatter plots depicting the relationship between FMF and RDO cost

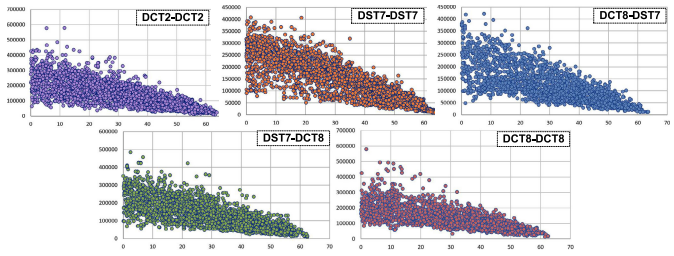


Fig. 3. Scatter diagrams between FMF and RDO cost of transform kernels in VVC, where the X-axis and Y-axis are corresponding to FMF and RDO cost. These scatter plots belong to 8×8 data encoded by the VTM for the *RaceHorses*.

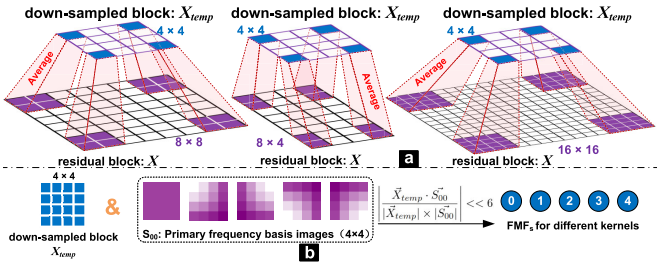


Fig. 4. Schematic diagram of down-sampling and FMF calculation under different sizes.

Algorithm 1 Fast Transform Kernel Selection

Input : residual block **X**, 5 primary frequency basis images $\mathbf{S}_{00}[i]$, trained decision tree **DT**, block width and height **w**, **h**

Output: The best transform kernel $\mathbf{T}_{best}$

1 **Step 1: the improved FMF acquisition process**;
2 /**** *down-sample X to* X_{temp}****/ ;
3 $\mathbf{w}_{stride} \leftarrow \mathbf{w}/4$;
4 $\mathbf{h}_{stride} \leftarrow \mathbf{h}/4$;
5 **for** $i \leftarrow 0$ **to** h **do**
6 average every $\mathbf{w}_{stride}$ samples in each row of X to X_{temp}
7 **end**
8 **for** $i \leftarrow 0$ **to** 4 **do**
9 average every $\mathbf{h}_{stride}$ samples in each column of X_{temp} to X_{temp}
10 **end**
11 /**** *FMF calculation*****/ ;
12 **for** $i \leftarrow 0$ **to** 4 **do**
13 $\mathbf{FMF}[\mathrm{i}] = \left| \frac{\vec{X}_{temp} \cdot S_{00}[\vec{i}]}{|\vec{X}_{temp}| \times |S_{00}[\vec{i}]|} \right| \ll 6$
14 **end**
15 **Step 2: the decision tree-based kernel selection**;
16 $\mathbf{T}_{best} \leftarrow \mathbf{DT}(\ \mathbf{FMF}[\mathrm{i}]\ )$;
17 **return:** $\mathbf{T}_{best}$

during the VVC encoding process, as illustrated in Fig. 3. The results reveal a similarly pronounced correspondence, reinforcing the utility of the FMF.

The method proposed in [13] relies on probabilistic models constructed for different transform sizes, which incurs significant computational overhead and hardware costs. This paper introduces a novel and hardware-friendly scheme, whose pseudo-code is presented in Algorithm 1. The method features two key innovations: **the improved FMF acquisition process** and **the decision tree-based kernel selection.**

1) Improved FMF Acquisition Process: As evident from (19), FMF computation primarily focuses on the operation $\vec{X} \cdot \vec{S_{00}}$, which involves intensive matrix dot-product calculations. Taking the 16×16 block size as an example, the FMF_0 computation for DCT2-DCT2 is relatively straightforward since S_{00} values are highly uniform. By summing $\vec{X}$ first and then multiplying, the number of required multipliers can be reduced to 1. However, for other transform kernels, due to the non-uniformity of S_{00}, FMF computation becomes more complex, requiring $16 \times 16 = 256$ multipliers per kernel. For five transform kernels, this amounts to $256 \times 4 + 1 = 1025$ multipliers. Considering additional block sizes (e.g., 4×4, 8×8, 4×16, etc.), the total number of multipliers reaches 3145, leading to excessive hardware costs. To address this, we propose an improved FMF acquisition process, as detailed in Algorithm 1, Line 1. The approach involves two procedures:

- *Residual Block Downsampling.* The residual block is down-sampled to a 4×4 block by averaging operations. For instance, the process is illustrated for 8×8, 8×4, and 16×16 block sizes in Fig. 4a. After this, all residual blocks are unified into 4×4 down-sampled blocks.
- *FMF Computation.* FMFs are calculated using the 4×4 S_{00} matrices corresponding to different transform kernels according to (19), as shown in Fig. 4b.

The motivation for downsampling is based on the observation that adjacent residual samples within a block typically exhibit minimal variation. Consequently, using local averages effectively captures the overall residual distribution. Ablation experiments in Section V-B validate that decision-making performance remains unaffected by downsampling. Furthermore, unifying all residual blocks to a 4×4 size reduces the required multipliers for FMF computation to $4 \times 4 \times 4 + 1 = 65$. Compared to the original computation, it achieves a multipliers reduction of **97.9%**, substantially lowering hardware costs.

2) Decision Tree-Based Kernel Selection: After obtaining the FMF information of the current residual block, a lightweight decision tree is used to select the optimal transform kernel. The training data is extracted from the VTM encoding process of sequences outside the Common Test Conditions (CTC), including {352×288: bus, foreman, mobile} and {704×576: crew, harbour}. The five FMF values serve as inputs, the RDO-selected optimal kernel is used as the labels collected across four Quantization Parameter (QP) values {22, 27, 32, 37}. The decision tree is trained using Gini impurity as the splitting criterion, with the structure constrained to only 10 split nodes. The decision-making performance will be presented in detail in Section V-B.

B. Algorithmic Optimizations for DST7

This section explores a low-cost and high-precision approximation scheme for DST7 by directly converting the results of DCT2 into DST7. By exploiting the separability, the 2D

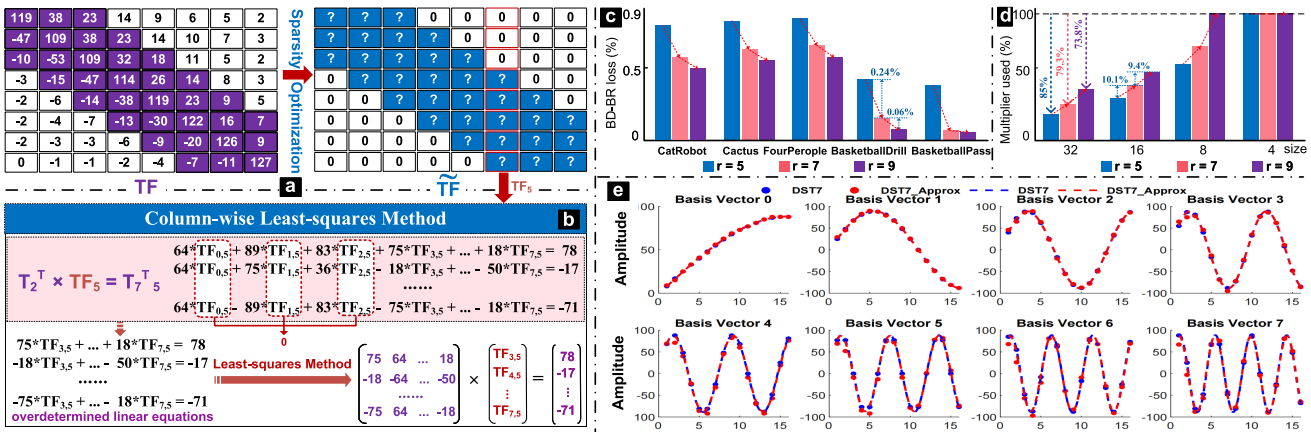


Fig. 5. The algorithmic optimizations for DST7. (a) The introduction of transfer matrix and the motivation of sparsity optimization. (b) Sparsity optimization of TF_5 with the least-squares method. (c) The Bd-BR loss of five sequences under different r. (d) The multiplier usage ratio of four sizes under different r. (e) Performance of approximation when $r = 7$.

transform is commonly decomposed into two sequential 1D transforms. Therefore, optimization efforts for transform architectures mainly focus on 1D transforms. To establish the relationship between the DCT2 and DST7, we begin with the following equations describing their respective 1D transform processes:

$$Y_2^T = X \times T_2^T, \qquad Y_7^T = X \times T_7^T, \tag{20}$$

where X is the residual block, T_2^T and T_7^T are DCT2 and DST7 transposed transform matrices, and Y_2^T and Y_7^T are transformed coefficient block of DCT2 and DST7. To reveal the relationship between of DCT2 and DST7, we introduce a transfer matrix TF that satisfies:

$$Y_2^T \times TF = Y_7^T \iff X \times T_2^T \times TF = X \times T_7^T, \tag{21}$$

simplifying it as:

$$T_2^T \times TF = T_7^T. \tag{22}$$

Given that T_2 is an orthogonal matrix, we know that $T_2^{-1} = \frac{T_2^T}{2^n}$, where 2^n is the scaling factor for integer transform. Similarly, the inverse matrix of T_2^T is $\frac{T_2}{2^n}$. Substituting this into (22) leads to:

$$\frac{1}{2^n} \times T_2 \times T_2^T \times TF = \frac{1}{2^n} \times T_2 \times T_7^T, \tag{23}$$

$$TF = \frac{1}{2^n} \times T_2 \times T_7^T. \tag{24}$$

Fig. 5a illustrates the transfer matrix TF computed from (24) with scaling and rounding. The matrix elements display a clear aggregation pattern, with the larger values clustering along the diagonal, while values further away from the diagonal tend to approach zero. Discarding the smaller values of less significance, is beneficial to reduce computational expenses with slightly affecting the accuracy of the transfer.

Building on the previous observation, we propose a **sparse optimization** approach for TF. It can be framed as a constrained optimization task, where the goal is to minimize the error, as defined in (26), between the original DST7 matrix T_7 and its approximation $\widetilde{T_7}$, derived from the sparse transfer matrix $\widetilde{TF}$. The optimization is subject to the constraint of limiting the number of non-zero elements in each column of $\widetilde{TF}$. The mathematical formulation is as follows:

$$\widetilde{T_7^T} = round\ (T_2^T \times \widetilde{TF} \times \frac{1}{2^n}); \tag{25}$$

$$E(\widetilde{TF}) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (T_7(i,j) - \widetilde{T_7}(i,j))^2 \tag{26}$$

$$\textbf{minimize}\ E(\widetilde{TF}),$$

$$\textbf{subject to}\ \widetilde{TF}_{i,j} = 0,\ |i-j| > (r-1)/2,$$

$$\widetilde{TF}_{i,j} \in \mathbb{Z} \cap [-2^n + 1, 2^n]. \tag{27}$$

The scaling factor $\frac{1}{2^n}$ in (25) enables the derived $\widetilde{TF}$ matrix to be scaled by 2_n compared to (22), ensuring that $\widetilde{TF}$ can be rounded to an integer matrix with higher precision. The variable r denotes the number of elements to be retained near the diagonal in each column of $\widetilde{TF}$. For instance, when $r = 5$, the diagonal and two neighboring elements on each side are preserved, as illustrated in Fig. 5a. Besides, note that the error $E(\widetilde{TF})$ is accumulated independently for each column. This allows the optimization problem to be decomposed into separate column-wise tasks, which helps reduce the complexity of the constrained optimization. In this study, we apply the **least squares method** to solve the constrained optimization for each column. To clarify, the optimization process for the sixth column $TF_{i,5}$ is shown in Fig. 5b. It is apparent that the system of equations is overdetermined. By expressing it as a matrix equation, the problem can be efficiently solved using MATLAB to find the least squares solution.

Following that, a set of experiments was conducted to determine the optimal value of r. The results, illustrated in Fig. 5c and 5d, were obtained using five video sequences of varying resolutions that were randomly selected as test cases. As depicted in Fig. 5c, increasing r leads to a consistent reduction in BD-BR loss across all sequences. However, the improvement diminishes with larger values of r. For example, in the case of *basketballDrill*, increasing r from 5 to 7 reduces performance loss by 0.24%, while the improvement is only 0.06% when r is increased from 7 to 9. In addition, multiplier usage ratios for the approximate transform compared to the exact transform are presented in Fig. 5d. The analysis reveals an approximately linear increase in multiplier usage

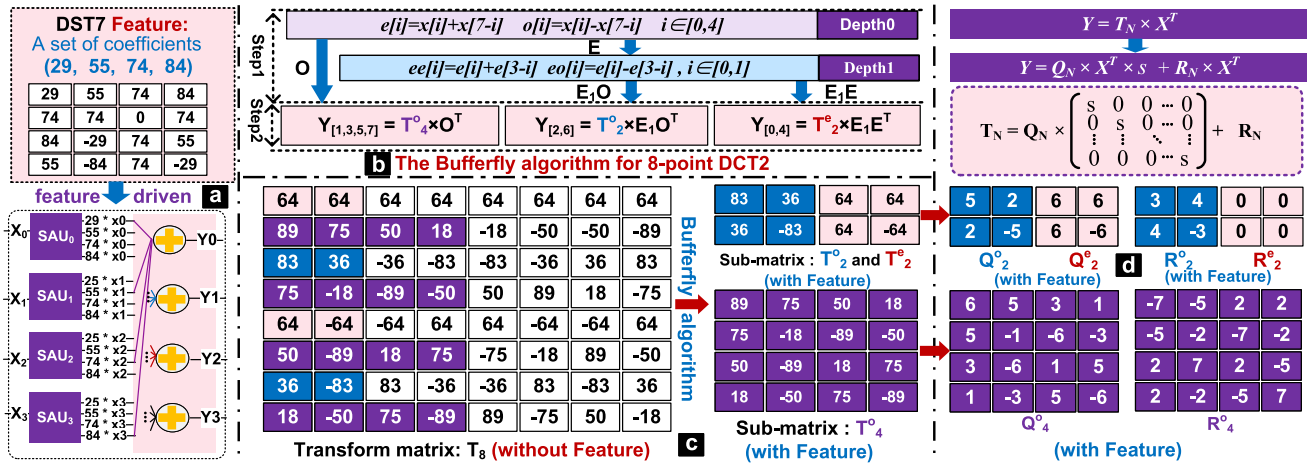


Fig. 6. (a) The beneficial feature and the SAU-based matrix multiplication. (b) The butterfly algorithm. (c) The 8-point DCT2 transform matrix and its sub-matrix after butterfly algorithm. (d) The proposed calculation scheme based on matrix decomposition.

as r grows. Based on a trade-off between performance gains and computational resource demands, $r = 7$ is identified as the optimal parameter.

In Fig. 5e, the approximation of the first eight basis vectors of the 16-point DST7 is presented for $r = 7$. It demonstrates that $\widetilde{T}_7$ derived from equation (25), closely aligns with the original matrix. For a 32-point DST7, an exact DST7 requires $32 \times 32 = 1024$ multipliers. By comparison, the proposed approximation, with $r = 7$, reduces this requirement to 212 multipliers, achieving a substantial reduction of **79.3%**. This highlights the effectiveness of the proposed method in balancing computational efficiency and precision.

C. Algorithmic Optimizations for DCT2

The optimization of DCT2 proposed in this study draws inspiration from the efficient hardware design of DST7, illustrated in Fig. 6a. A key feature of the DST7 transform matrix is that its row and column coefficients are derived from a single unified set, differing only in position and sign. This enables efficient Shift-Adder Unit (SAU)-based matrix multiplication, where N SAUs generate product terms that are accumulated through an adder tree to obtain the transformed results. However, Fig. 6c shows that DCT2 lacks this feature, rendering direct optimization using SAU-based matrix multiplication infeasible. Encouragingly, when the classical butterfly algorithm is applied to DCT2, the resulting sub-matrices exhibit this desirable property. Fig. 6b illustrates the 8-point DCT2 with the butterfly algorithm, where inputs corresponding to identical coefficients are grouped to compute the E and O vectors. These vectors are then multiplied by the sub-matrices to produce the transformed outputs. The emergence of this feature in the sub-matrices enables the remaining computations to be effectively optimized.

It should be highlighted that directly optimizing the sub-matrix computation via SAU-based matrix multiplication still leads to considerable resource overhead. This arises from the need to instantiate distinct SAUs for different sizes due to the variability in their elements. To address this and improve SAU reusability across sizes, a matrix decomposition-based calculation scheme for DCT2 is proposed, as shown in Fig. 6d, defined as:

$$T_N = Q_N \times s + R_N, \tag{28}$$

where s is a scaling factor that takes a power-of-two value (2^n), determined by the trade-off between the complexities of Q_N and R_N. As the s increases, the complexity of Q_N reduces, while that of R_N rises. The decomposition equation for each entry in $T_N(i, j)$ is expressed in (29):

$$\begin{cases} T_N(i,j) = Q_N(i,j) * s + R_N(i,j), \\ |R_N(i,j)| \le s/2. \end{cases} \tag{29}$$

Considering that the maximum entry of the DST7 and DCT2 matrix is 90. The absolute value ranges for the Q_N and R_N matrices are provided in Table II. Notably, when s is set to 16, their complexity become comparable. Under this condition, the optimized calculation scheme for DCT2 is described as (30):

$$Y^T = Q_N \times X^T \times s + R_N \times X^T. \tag{30}$$

Here, X is the inputs, and the resource-intensive matrix operation is replaced by two simpler matrix multiplications and shift operations. The limited range of Q_N and R_N allows for a lightweight SAU circuit, whose output $\{1x, 2x, 3x, 4x, 5x, 6x, 7x, 8x\}$ generates all the needed products across varying sizes, significantly enhancing reusability.

TABLE II
THE RANGE OF ABSOLUTE VALUE IN Q_N AND R_N MATRICES

Scale factor s	4	8	16	32	64
abs Q_N range	[0,23]	[0,11]	[0,6]	[0,3]	[0,1]
abs R_N range	[0,2]	[0,4]	[0,8]	[0,16]	[0,32]

IV. THE HARDWARE TRANSFORM ACCELERATOR IMPLEMENTATION

A. Overall Accelerator Design

We presents a highly-pipelined, area-efficient, and high-throughput transform accelerator, as illustrated in Fig. 7a.

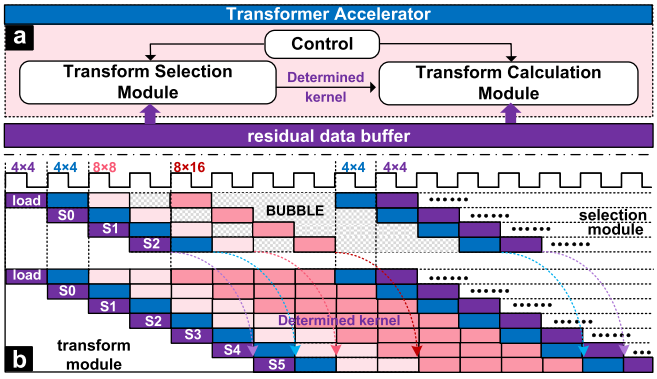


Fig. 7. The proposed data rearrangement scheme.

The accelerator integrates the transform kernel selection module and transform calculation module. The selection module determines the optimal transform kernel, while the computation module performs the corresponding transformation. To the best of our knowledge, this is the first work to provide a comprehensive solution to these two critical challenges.

With high degrees of pipelining, the selection module completes the kernel decision-making process in 3 cycles, and the calculation module executes the transform in 6 cycles. When processing a residual block, both modules initiate simultaneously. Taking the advantages of the DST7 approximation scheme, the transform module only requires the optimal kernel before entering the final pipeline stage, allowing ample time for the selection process. Fig. 7b provides a timing diagram using different residual blocks as examples, showing that the selection module's processing time is entirely concealed within the computation module's processing time, introducing no additional latency. Consequently, the accelerator's overall throughput equals that of the computation module.

B. The Proposed Kernel Selection Architecture

The proposed kernel selection architecture consists of three stages: downsampling, FMF calculation, and decision tree. The subsequent sections provide a detailed description of the data flow for each pipeline stage, with a particular emphasis on the parallelism and reusability strategies.

1) Downsampling Stage: **Stage0** is a dedicated pipeline stage for downsampling residual blocks to a 4×4 size, which consists of three main steps. First, residual samples are bound to the groups according to the size. Fig. 9a illustrates the binding process for an 8×8 block, and Fig. 4a provides examples for 8×4 and 16×16 blocks. Next, the grouped samples are processed through a multiple-output adder tree (M_O_adder16), capable of accumulating inputs for different scales, such as 2, 4, 8, and 16 input casesb. For cases with fewer than 16 inputs, the grouped samples are mapped to different input ports, and the corresponding result is derived via certain data path. For instance, when down-sampling to a 4×4 size from 4×8 or 8×4, two adjacent inputs are averaged, and thus only one adder is needed for the operation. As shown in the Fig. 9b, two input data values are loaded into the first two input ports, and the sum result is output through the $O_{0,1}$ ports. The purple data path in the figure illustrates the summation of four inputs, supporting cases such as 8×8, 4×16, and 16×4. Finally, the averaged results are selected and output based on the size signal. *Reusability:* By configuring inputs and outputs through the size signal, M_O adder16 supports the downsampling of residual blocks for varying sizes with exceptional reusability.

Parallelism: The accelerator instantiates 16 M_O_adder16s, which work in parallel to derive the 4×4 down-sampled block.

2) FMF Calculation: **Stage1** is responsible for calculating FMF based on Algorithm 1, Line 13, where the numerator $\vec{X}_{temp} \cdot \vec{S_{00}}[i]$ and denominator $|\vec{X}_{temp}|$ are the primary computational components. The denominator terms $|\vec{S_{00}}[i]|$ are fixed values that can be precomputed and hardcoded into the circuit. Notably, the denominator term $|\vec{X}_{temp}|$ is identical for all five FMF types and thus requires only a single computation. Considering the trade-off between precision and resources, this value is defined as an integer in this paper. As shown in the Fig. 8, after calculating the SUM, it needs to be square-rooted and rounded, which is typically a resource- and cycle-intensive operation in hardware design. To address this, we designed the computational circuit shown in Fig. 10. Assuming the maximum bit-width of the current result is 10 bits, with each bit initially set to 0, the module determines the value of each bit one by one. First, the module compares $(1 \ll 9)^2$ with SUM. If the former is larger, the most significant bit is set to 1; otherwise, it is set to 0, with the result marked as c[8] in Fig. 10. Next, the value of the second-highest bit is determined. Specifically, the module compares $(c[0] + 1 \ll 8)^2$ with SUM, which is equivalent to comparing $(c[0]^2 + T8 + c[0] \ll 9)$ with SUM according to (31). If the former is larger, the second-highest bit is set to 1; otherwise, it is set to 0, and the result is recorded as c[7]. This process continues until all bits are determined.

$$\begin{aligned}(c[0] + 1 \ll 8)^2 &= c[0]^2 + (1 \ll 8)^2 + 2 \times c[0] \times (1 \ll 8) \\ &= c[0]^2 + T8 + c[0] \ll 9. \end{aligned} \tag{31}$$

Reusability: To reduce computational overhead, we fully explored the reuse characteristics in the calculation of $\vec{X}_{temp} \cdot \vec{S_{00}}[i]$. Taking $S_{00}[0]$ as an example, we observe that all 16 elements in the matrix can be synthesized using just 3 basic elements $\{13, 25, 34\}$. Based on the concept of combining like-type terms, we develop the decomposition strategy illustrated in Fig. 11. This approach enables the functionality of 16 multipliers to be achieved with only 3, reducing multiplier resource usage by **81.25%**.

3) Decision Tree: **Stage2** is responsible for determining the optimal transform kernel, with decisions made at each node of the decision tree through comparison operations. In the original design, the decision tree directly compared FMF values. Considering that the FMF calculation involves division, this approach incurred significant hardware resource consumption. To address this, we optimized the comparison process at each node, as illustrated in Fig. 8, by multiplying the denominator terms with the node values. This modification eliminates the need for division, significantly reducing resource usage.

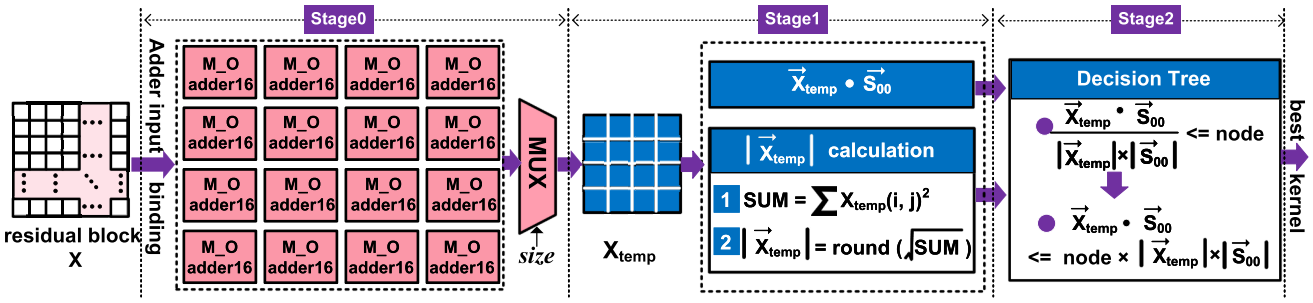


Fig. 8. The proposed pipelined architecture of transform kernel seletction.

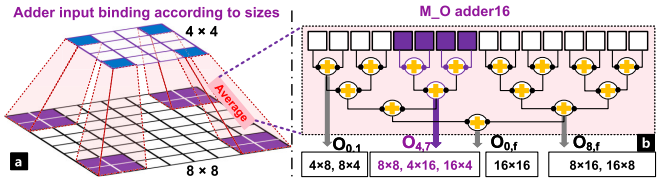


Fig. 9. Down-sample X to X_{temp}. (a) Binding residual data to the inputs of M_O adder16 according to sizes. (b) M_O adder16.

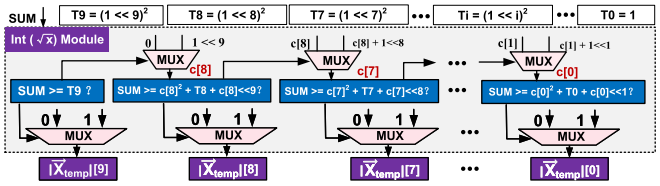


Fig. 10. The circuit of Int ($\sqrt{x}$) module.

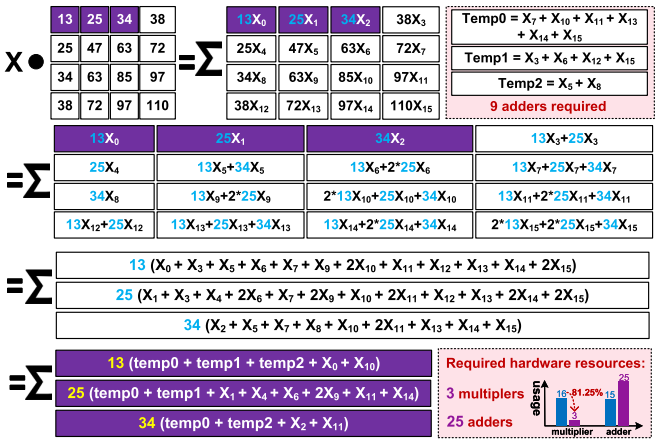


Fig. 11. The reuse characteristics and proposed decomposition strategy in the calculation of $\vec{X}_{temp} \cdot S_{00}[0]$.

C. The Proposed Transform Calculation Architecture

We introduce a highly parallel, extensively reusable, and deeply pipelined transform architecture in Fig. 12, designed to process 32 samples per cycle. For transforms smaller than 32 points, multiple inputs are concatenated to form a 32-point input. Specifically, it handles 8 sets of 4-point inputs, 4 sets of 8-point inputs, 2 sets of 16-point inputs, or 1 set of 32-point inputs per cycle. This parallelism achieves the highest degree of concurrency in the literature.

The accelerator's architecture is organized into six stages: even-odd decomposition, lightweight SAU calculation, selector tree, adder tree, DCT2 result integration, and transfer calculation. The first five stages are dedicated to DCT2, while the final stage is for DST7. The subsequent sections elaborate on the data flow through each stage, emphasizing parallelism and reusability strategies.

1) Even-Odd Decomposition Stage: **Stage0** corresponds to the even-odd decomposition stage, decomposing the inputs into E and O vectors using the butterfly algorithm, as shown in Fig. 6b. Accordingly, we design an even-odd decomposition circuit depicted in Fig. 12b, customized for the 8 sets of 4-point inputs case.

Reusability: Directly applying the decomposition circuit to different sizes can result in data misalignment. For instance, with a set of 32-point input, the correct $e[0]$ involves adding X_1 and X_{31}, but the circuit would mistakenly compute it as the sum of X_1 and X_3, leading to a malfunction. To ensure functional correctness and reusability, we propose a data rearrangement scheme depicted in Fig. 13, where X_{31} is relocated to the position of X_3. This scheme not only eliminates the data misalignment but also enables reuse of the decomposition circuit across different sizes, reducing the adder resource requirement by **75%**.

2) Lightweight SAU Calculation Stage: **Stage1** is responsible for the SAU computation, generating all the product terms involved in the matrix multiplications.

Reusability: As indicated in Table I, the absolute values fall within the range of [0, 8]. This allows for the design of an efficient, lightweight SAU that utilizes only three adders to compute the product terms, as illustrated in Fig. 12a.The reusability of this pipeline stage is achieved in two ways: firstly, the SAU can be leveraged across various transform sizes; secondly, there is no need to instantiate separate SAU circuits for the Q and R, as both share the same value range.

Parallelism: A total of 32 lightweight SAUs are instantiated in the accelerator, operating in parallel to generate the corresponding product terms for the 32 inputs.

3) Selector Tree: **Stage2** is the selector tree stage, selecting the product terms generated by the SAUs. According to the matrix operation rules, the corresponding SAU outputs are selected and passed to the next stage via Adder-tree Selectors (ATS), as depicted in Fig. 12c. Each ATS consists of a set of selectors, with the *size* acting as the control signal.

Parallelism: In this stage, the selection of sub-matrices Q and R occurs in parallel. Each sub-matrix requires 32 ATS units, resulting in a total of 64 ATS units operating concurrently.

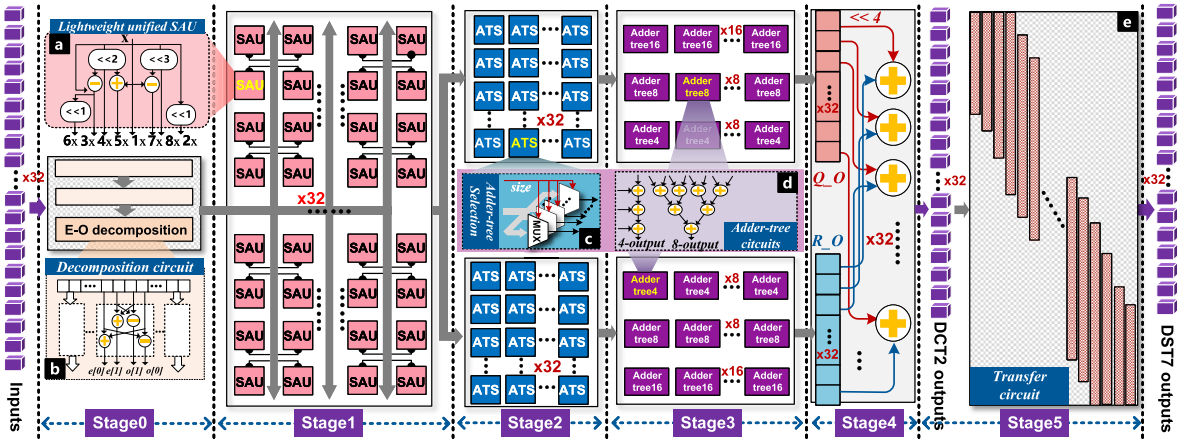


Fig. 12. The proposed architecture. (a) Lightweight unified SAU. (b) Decomposition circuit. (c) Adder tree selection. (d) Adder-tree circuit. (e) Transfer circuit.

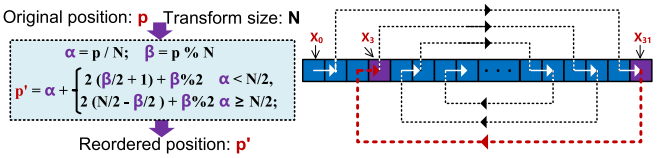


Fig. 13. The proposed data rearrangement scheme.

4) Adder Tree and DCT2 Result Integration Stages: **Stage3** is the adder tree stage, accumulating the selected product terms.

Parallelism: In this stage, three types of adder-trees are employed to efficiently perform accumulation operations: adder-tree4, adder-tree8, and adder-tree16 as shown in Fig. 12d, responsible for accumulating 4, 8, and 16 inputs, respectively. In the traditional DCT2, each output is obtained by accumulating 32 product terms, requiring 32 adder-tree32. However, thanks to the butterfly algorithm, after the even-odd decomposition, the actual sub-matrices involved in the computation are O, EO, EEO, and EEE, with sizes of 16×16, 8×8, 4×4 and 4×4, respectively. Consequently, instead of 32 adder-tree32, only 16 adder-tree16, 8 adder-tree8, and 8 adder-tree4 are enough to perform the accumulation. Since the sub-matrices Q and R are processed in parallel, the total number of adder-tree circuits is doubled. Thus, the proposed accelerator requires 32 adder-tree16, 16 adder-tree8, and 16 adder-tree4 circuits to perform the parallel accumulation.

Stage4 is the DCT2 result integration stage, responsible for calculating the final DCT2 results using the outcomes of sub-matrices Q and R according to (30).

5) Transfer Calculation Stage: **Stage5** is the transfer calculation stage, responsible for approximating the DCT2 results to DST7 results based on (21) through a sparse matrix multiplication. This stage requires 212 multipliers. In contrast, the exact DST7 calculation requires $32 \times 32 = 1024$ multipliers. Therefore, the approximation scheme reduces multiplier consumption by **79.3%**.

V. Evaluation

A. Experimental Setup

Performance evaluations are conducted using the VVC reference software VTM17.0 under all-intra configurations, with BD-BR as the evaluation metric. The key configurations related to transform coding are listed as follows:

```
-TS 1 -MTS 1 -MTSIntraMaxCand 4 -SBT 0
-LFNST 0 -RDOQ 1 -RDOQTS 1
```

The accelerator was implemented in Verilog HDL, verified through simulation in Vivado, and synthesized using Global Foundry's 28nm standard cell library to obtain resource, frequency, and power consumption performance.

B. Coding Performance Comparison

1) Comparison of the Kernel Selection Algorithm: We evaluate the proposed transform kernel selection algorithm by comparing its coding performance with similar algorithms. The evaluation metrics include BD-BR and encoding time savings (TS). Additionally, ablation experiments are conducted to compare the cases with and without downsampling, demonstrating that the downsampling operation significantly reduces computational cost without compromising coding efficiency. As shown in Table III, disabling MTS achieves a substantial 43.9% reduction in encoding time but incurs a significant 2.42% loss in coding performance—a trade-off that is typically unacceptable in real-world encoding scenarios. In contrast, the proposed method achieves a 26.5% acceleration with a negligible performance loss of only 0.19%. By retaining more than half of the acceleration gains while preserving virtually all coding performance, the method demonstrates exceptional decision-making efficiency and practical value.

The method in [12] is a skip algorithm that leverages encoded syntax elements to determine whether MTS can be bypassed for the current residual block. However, as discussed earlier, the algorithm cannot guarantee a stable skip. In the worst-case scenario, the encoder still processes all transform kernels, limiting its acceleration effect to a mere 5.01%. More critically, the method fails to incorporate the distribution characteristics of the residual into its decision-making process, resulting in a coding performance degradation of 0.23%. Work [16] proposes a selection algorithm that simply exploits the distribution characteristics of residual blocks to prune transform kernels. However, the pruning depth is limited, reducing 5 transform kernels to 3. While this approach preserves relatively high coding gains, it leaves multiple transform

TABLE III
BD-BR (%) AND TIME SAVING (%) OF THE PROPOSED SELECTION METHOD AND OTHER WORKS AGAINST ANCHOR

Sequences		Disable MTS		ISCAS'22 [12]		ICCCAS'22 [16]		Ours		Ours with D-S*	
		BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)	BD-BR (%)	TS (%)
BasketballDrill	C	1.53	44.82	0.25	6.03	0.01	23.34	0.04	31.43	0.04	32.45
BQMall		1.50	49.00	0.25	7.13	0.11	22.22	0.13	33.19	0.08	30.64
PartyScene		1.10	48.20	0.10	5.93	0.17	24.23	0.21	37.37	0.24	35.76
RaceHorsesC		2.03	45.44	0.17	4.81	0.30	21.63	0.30	32.59	0.32	31.91
FourPeople	E	3.24	46.16	0.33	4.29	0.14	22.90	0.16	30.31	0.18	30.17
Johnny		2.73	47.92	0.28	4.67	0.14	26.77	0.15	32.30	0.12	31.81
KristenAndSara		2.95	49.36	0.21	6.13	0.07	32.10	0.16	34.58	0.13	35.22
Cactus	B	2.70	41.21	0.23	3.89	0.25	20.00	0.31	24.52	0.32	24.96
MarketPlace		3.28	48.65	0.24	5.29	0.23	18.69	0.31	20.84	0.29	21.83
RitualDance		3.35	44.23	0.27	5.37	0.10	19.69	0.12	20.24	0.14	20.61
BastetballDrive		2.08	53.97	0.27	5.45	0.06	27.80	0.11	31.35	0.15	32.32
BQTerrace		1.37	49.74	0.11	4.45	0.19	14.62	0.21	27.44	0.20	26.76
Tango2	A1	3.49	37.82	0.22	6.04	0.12	2.62	0.19	13.10	0.22	17.41
FoodMarket4		2.70	25.61	0.27	2.72	0.02	-4.63	0.08	8.32	0.08	19.62
Campfire		1.97	37.31	0.17	3.25	0.29	12.90	0.29	20.02	0.31	20.19
CatRobot	A2	2.76	41.00	0.24	5.65	0.16	15.12	0.19	19.86	0.22	20.63
DaylightRoad2		2.85	39.28	0.20	6.21	0.28	20.28	0.28	20.28	0.24	21.39
ParkRunning3		1.92	40.46	0.21	2.76	0.39	15.25	0.39	15.25	0.39	23.33
Average		2.42	43.90	0.23	5.01	0.17	18.64	**0.19**	**25.17**	**0.19**	**26.50**

∗ D-S means downsampling.

kernels to be processed in the RDO, significantly constraining its acceleration potential. As shown in Table III, this method achieves only an 18.64% reduction in encoding time at a minor 0.17% performance loss. Checking 3 transform kernels necessitates instantiating three separate RDO circuits in hardware design, resulting in considerable resource overhead and posing significant challenges for practical implementation.

Compared to the previous two algorithms, the proposed method achieves superior acceleration. In contrast to [16], performance degradation is only 0.19% when more kernels are pruned, demonstrating a comparable coding efficiency and a reduced computational complexity. Moreover, the proposed approach integrates the hardware-friendly strategy, of downsampling-based FMF computation. As shown in Table III, this optimization not only avoids additional performance degradation but also accelerates the encoding process further, thereby enhancing overall efficiency.

2) Comparison of the DST7 Approximate Algorithm: A comparison with other works is conducted to validate the effectiveness of the proposed DST7 approximate algorithm, as shown in Table IV. Taking the number of sequences in each resolution class into consideration, the final average result is calculated using the following formula:

$$Average = \frac{1}{18}(6 \times (A1\&A2) + 5 \times B + 4 \times C + 3 \times E). \tag{32}$$

By restricting the transform type in VVC to DCT2 only, the coding performance incurs a loss of 2.42% compared to multiple types. Notably, when the proposed approximation method is applied within the framework of multiple transform types, the performance degradation is significantly mitigated, with only a 0.57% loss in coding efficiency, retaining most coding gains of the original scheme.

TABLE IV
COMPARISON OF THE DST7 APPROXIMATE ALGORITHMS

	DCT2	TCSVT [37]	ICIP [39]	ISCAS [36]	Ours
Performance	Evaluating metric: **BD-BR (%)**				
Hardware Friendly	✓	×	✓	✓	✓
A1 & A2	2.62	0.20	1.13	1.79	0.67
B	2.55	0.14	1.32	1.16	0.71
C	1.54	0.06	0.96	0.93	0.23
E	2.97	0.18	0.95	1.25	0.59
Average	**2.42**	**0.15**	**1.12**	**1.33**	**0.57**

TABLE V
THE OVERALL CODING PERFORMANCE OF INTEGRATING THE KERNEL SELECTION AND DST7 APPROXIMATION

Class	A1&A2	B	C	E	Average
BD-BR	0.80%	0.84%	0.76%	0.52%	**0.74%**

Approximation algorithms in the literature can be categorized into two types. **Neglecting hardware constraints** and **Hardware-friendly algorithms**. The first type does not adequately consider the practical hardware limitations when designing algorithms. For example, in [37], a high-precision private approximation architecture is proposed. While high accuracy is achieved, it requires embedding an inverse transform circuit in the forward pipeline and modifying the decoder, doubling circuit area and making it unsuitable for most embedded systems [41]. The second type is specifically designed to be hardware-friendly, facilitating easy integration into the forward transform pipeline without necessitating changes to the decoder, including [36], [39] and the proposed algorithm. While [36] and [39] present hardware-efficient, they incur coding performance loss greater than 1%. In contrast, our approach halves this loss, demonstrating its superior approximation accuracy.

3) Overall Coding Performance: The Table V presents the overall coding performance after integrating the two

TABLE VI
COMPARISON OF DIFFERENT TRANSFORM ARCHITECTURES

Design	TCAS-I [17]	TCAS-I [18]	TCAS-I [20]	Ours	TCSVT [35]	ISCAS [36]	Ours	TCE [30]	TVLSI [28]	ICASSP [40]	Ours
Technology(nm)	90	90	90	28	65	65	28	90	65	28	28
Frequency (MHz)	250	303	300	769	384	333	769	345	476	600	769
Parallelism	4∼32	4∼32	8	32	32	2∼8	32	8	32	2	32
Gate Count	111.2K	149K	166K	132K	275.4K	30.7K	51.9K	136.4K	329K	96.8K	184.4K
Power (mW)	16.83	26.96	23.22	42	68.6	*NA*	16.5	*NA*	95.2	*NA*	58.7
NGC	3.475∼27.8	4.66∼37.25	20.75	**4.125**	8.61	3.838 ∼ 15.35	**1.628**	7.05	10.281	48.4	**5.763**
NP	0.067	0.089	0.074	**0.054**	0.179	-	**0.021**	-	0.2	-	**0.076**
Sizes	4∼32	4∼32	4∼32	4∼32	4∼32	4∼32	4∼32	4,8	4∼32	4∼64	4∼32
Types	DCT2	DCT2	DCT2	DCT2	DCT8 DST7	DCT8 DST7	DCT8 DST7	DCT2/5/8 DST1/7	DCT2 DST7/DCT8	DCT2 DST7/DCT8	DCT2 DST7/DCT8

TABLE VII
HARDWARE PERFORMANCE OF THE PROPOSED SELECTION MODULE

Design	Proposed Selection Module
Frequency	769 MHz
Gates Count	27.7 K
Power	6.75 mW

optimization algorithms proposed in this study. It can be seen that when the transform kernel selection and the DST7 approximation working together, the overall coding loss is approximately equal to the sum of their individual performance losses, amounting to just 0.74%. This modest performance degradation achieves significant savings in both resources and latency, which greatly enhances the practical applicability of the VVC encoder.

C. Hardware Implementation Comparison

1) Hardware Performance of Selection Module: The three-stage selection module architecture proposed earlier was implemented in hardware, with its performance metrics summarized in the Table VII. The results indicate that the module requires only 27.7K equivalent gates to handle the selection of five transform kernels, achieving a power consumption of just 6.75 mW at an operating frequency of 769 MHz. As the first transform kernel selection design targeting Application-Specific Integrated Circuit (ASIC), direct comparisons with similar works are currently unavailable. However, when combined with the implementation results of the transform module, it is evident that the selection module avoids redundant instantiations of the RDO pipeline when consuming only 15% of the resources required by the transform module, demonstrating exceptional resource efficiency.

2) Comparison of the Transform Module: To highlight the hardware advantages of the proposed accelerator, we present a comparison with related works in terms of resources, throughput, and power in Table VI.

Resource: The proposed accelerator achieves a parallelism of 32 samples/cycle with the gate count of 184.4K. Compared to the latest design [28] with the same parallelism requiring 329K gates, our design reduces resource usage by **44%**. Additionally, the design in [35] supporting only two types at the same throughput has the gate count of 275.4K. Despite supporting more transform types, our accelerator achieves a **33%** reduction in gate count through algorithmic optimization.

Due to the varying specifications across different designs, the supported data parallelism may differ. To enable a fairer comparison, we adopt the Normalized equivalent Gate Count (NGC) defined as (33) to assess the efficiency of the design. Additionally, we compare the NGC of different works for DCT2 and DST7/DCT8.

$$NGC = Gate\ Count\ /\ Parallelism\ . \tag{33}$$

For DCT2, our accelerator achieves a 32-point data parallelism with only 132.5K equivalent gates, resulting in the NGC of 4.125. The data parallelisms of architectures in [17] and [18] are non-constant, resulting in smaller resource consumptions but lower speeds. Responsively, the NGC of [17] and [18] are 3.475∼27.8 and 4.66∼37.25. Compared to them, our accelerator reduces the normalized resources for DCT2 by **85.2%** and **88.9%** at their worst cases. The work in [20] is designed with fixed data parallelism and its NGC is 20.75. The proposed DCT2 design achieve a normalized area reduction of **80.1%**.

For DST7/DCT8, the computational resources in our work are primarily consumed in the transfer calculation stage, requiring 51.9K gates with a NGC of 1.628. There are two designs in [35] and [36]. Work in [35] reports a design of an NGC of 8.61 with 275.4K gates for 32-point parallelism. Compared to it, our design reduces the normalized resources by **81.1%**. For work [36], which supports an inconstant throughput and the maximum parallelism of only 8 points with an NGC ranging from 3.838 to 15.35, our accelerator achieves a normalized resource reduction of **57.6%** to **89.4%.**

Works [28], [30], and [40] support three transform types in VVC, with NGC of 17.05, 10.281, and 48.4, respectively. With supporting three types and a fixed 32-point throughput, the proposed design achieves an NGC of 5.763. Compared to these designs, our work reduces the normalized gate count by **66.2%**, **44%**, and **88%**, respectively.

Throughput: In addition to area efficiency, the proposed architecture also exhibits the highest processing performance among all designs. It achieves the highest levels of parallelism and operating frequency compared to existing works, making it capable of meeting the real-time encoding requirements of **8K@57fps**.

Power: The power efficiency of the proposed accelerator is evaluated using frequency-normalized dynamic power (NP) as defined in [42] and (34).

$$NP = Power \ / \ Frequency \ . \tag{34}$$

Table II shows that, compared with the works reporting power consumption in the literature, the proposed accelerator achieves NP savings of over **19.4%**, **30.3%**, **60%**, **88.3%**, and **62%** for DCT2, DST7, and three types, respectively.

VI. Conclusion

This paper comprehensively addresses the hardware design challenges in transform kernel selection and computation, proposing an advanced accelerator optimized for VVC transform coding. Adopting a co-optimization strategy between algorithms and hardware, the accelerator introduces a decision tree-based algorithm for efficient kernel selection and a transition matrix-based approximate DST7 algorithm for transform computation. On the hardware side, multiple parallelism and reuse strategies are leveraged to develop several optimized circuit designs. The resulting accelerator features a highly pipelined architecture, supporting real-time processing at 8K@57fps. Additionally, it demonstrates notable advantages in resource efficiency and power consumption compared to state-of-the-art designs, underscoring its effectiveness and hardware adaptability.

References

[1] M. M. Corrêa, B. H. Waskow, J. W. Goebel, D. M. Palomino, G. R. Corrêa, and L. V. Agostini, "A high-throughput hardware architecture for AV1 non-directional intra modes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1481–1494, May 2020.

[2] W. Kolodziejski, R. Domanski, and L. Agostini, "FastGW: A machine learning-based early skip for the AV1 global warped motion compensation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, pp. 1–12, 2024.

[3] G. Pastuszak, "Generative multi-symbol architecture of the binary arithmetic coder for UHDTV video encoders," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 3, pp. 891–902, Mar. 2020.

[4] F. L. L. Ramos, A. V. P. Saggiorato, B. Zatt, M. Porto, and S. Bampi, "Residual syntax elements analysis and design targeting high-throughput HEVC CABAC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 2, pp. 475–488, Feb. 2020.

[5] R. J. Clarke, "Relation between the Karhunen Loève and cosine transforms," *IEE Proc. F, Commun., Radar Signal Process.*, vol. 128, no. 6, pp. 359–360, Jan. 1981.

[6] H. Yu, Z. Lin, and F. Pan, "Applications and improvement of H.264 in medical video compression," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 12, pp. 2707–2716, Dec. 2005.

[7] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Trans. Image Process.*, vol. 21, pp. 1874–1884, 2012.

[8] T. Biatek, V. Lorcy, and P. Philippe, "Transform competition for temporal prediction in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 3, pp. 815–826, Mar. 2019.

[9] X. Zhao et al., "Transform coding in the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 10, pp. 3878–3890, Oct. 2021.

[10] T. Fu, H. Zhang, F. Mu, and H. Chen, "Two-stage fast multiple transform selection algorithm for VVC intra coding," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 61–66.

[11] G. Bjøntegaard. (2001). *Calculation of Average PSNR Differences Between RD-Curves*. [Online]. Available: https://api.semanticscholar.org/CorpusID:61598325

[12] M. Saldanha, G. Sanchez, C. Marcon, and L. Agostini, "Fast transform decision scheme for VVC intra-frame prediction using decision trees," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 1948–1952.

[13] Z. Hao et al., "Fast transform kernel selection based on frequency matching and probability model for AV1," *IEEE Trans. Broadcast.*, vol. 70, no. 2, pp. 693–707, Jun. 2024.

[14] K.-S. Lu, A. Ortega, D. Mukherjee, and Y. Chen, "Efficient rate-distortion approximation and transform type selection using Laplacian operators," in *Proc. Picture Coding Symp. (PCS)*, 2018, pp. 76–80.

[15] H. Su, M. Chen, A. Bokov, D. Mukherjee, Y. Wang, and Y. Chen, "Machine learning accelerated transform search for AV1," in *Proc. Picture Coding Symp. (PCS)*, 2019, pp. 1–5.

[16] Z. Wang, J. Wang, J. Yang, C. Luo, F. Liang, and K. Huang, "A fast transform algorithm for VVC intra coding," in *Proc. 11th Int. Conf. Commun., Circuits Syst. (ICCCAS)*, 2022, pp. 237–240.

[17] S. Chatterjee and K. Sarawadekar, "Approximated core transform architectures for HEVC using WHT-based decomposition method," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 11, pp. 4296–4308, Nov. 2019.

[18] H. Sun, Z. Cheng, A. M. Gharehbaghi, S. Kimura, and M. Fujita, "Approximate DCT design for video encoding based on novel truncation scheme," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 4, pp. 1517–1530, Apr. 2019.

[19] M. Jridi, A. Alfalou, and P. K. Meher, "A generalized algorithm and reconfigurable architecture for efficient and scalable orthogonal approximation of DCT," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 2, pp. 449–457, Feb. 2015.

[20] A. Shabani, M. Sabri, B. Khabbazan, and S. Timarchi, "Area and power-efficient variable-sized DCT architecture for HEVC using muxed-MCM problem," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1259–1268, Mar. 2021.

[21] J. Huang, T. Nandha Kumar, H. A. F. Almurib, and F. Lombardi, "A deterministic low-complexity approximate (multiplier-less) technique for DCT computation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 8, pp. 3001–3014, Aug. 2019.

[22] A. K. Prasoon and K. Rajan, "4 × 4 2-D DCT for H.264/AVC," in *Proc. Int. Conf. Adv. Comput., Commun. Control*, 2009, pp. 573–577.

[23] A. Ahmed, M. U. Shahid, and A. U. Rehman, "N point DCT VLSI architecture for emerging HEVC standard," *VLSI Des.*, vol. 2012, no. 1, Jan. 2012, Art. no. 752024.

[24] W. Zhao, T. Onoye, and T. Song, "High-performance multiplierless transform architecture for HEVC," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 1668–1671.

[25] M. Masera, G. Masera, and M. Martina, "An area-efficient variable-size fixed-point DCT architecture for HEVC encoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 1, pp. 232–242, Jan. 2020.

[26] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan. 2014.

[27] Z. Hao, F. Xu, G. Xiang, P. Zhang, X. Zeng, and Y. Fan, "A multiplier-less transform architecture with the diagonal data mapping transpose memory for the AVS3 standard," in *Proc. IEEE 14th Int. Conf. ASIC (ASICON)*, Oct. 2021, pp. 1–4.

[28] Z. Hao, H. Sun, G. Xiang, P. Zhang, X. Zeng, and Y. Fan, "A reconfigurable multiple transform selection architecture for VVC," in *Proc. IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, Feb. 2023, vol. 31, no. 5, pp. 658–669.

[29] M. Zheng, J. Zheng, Z. Chen, L. Wu, X. Yang, and N. Ling, "A reconfigurable architecture for discrete cosine transform in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 3, pp. 810–821, Mar. 2020.

[30] A. C. Mert, E. Kalali, and I. Hamzaoglu, "High performance 2D transform hardware for future video coding," *IEEE Trans. Consum. Electron.*, vol. 63, no. 2, pp. 117–125, May 2017.

[31] M. J. Garrido, F. Pescador, M. Chavarrías, P. J. Lobo, and C. Sanz, "A high performance FPGA-based architecture for the future video coding adaptive multiple core transform," *IEEE Trans. Consum. Electron.*, vol. 64, no. 1, pp. 53–60, Feb. 2018.

[32] M. J. Garrido, F. Pescador, M. Chavarrías, P. J. Lobo, C. Sanz, and P. Paz, "An FPGA-based architecture for the versatile video coding multiple transform selection core," *IEEE Access*, vol. 8, pp. 81887–81903, 2020.

[33] A. Kammoun, S. Ben Jdidia, F. Belghith, W. Hamidouche, J. F. Nezan, and N. Masmoudi, "An optimized hardware implementation of 4-point adaptive multiple transform design for post-HEVC," in *Proc. 4th Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, 2018, pp. 1–6.

[34] A. Kammoun, W. Hamidouche, F. Belghith, J.-F. Nezan, and N. Masmoudi, "Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard," *IEEE Trans. Consum. Electron.*, vol. 64, no. 4, pp. 424–432, Nov. 2018.
[35] Y. Fan, Y. Zeng, H. Sun, J. Katto, and X. Zeng, "A pipelined 2D transform architecture supporting mixed block sizes for the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 9, pp. 3289–3295, Sep. 2020.
[36] Y. Zeng, H. Sun, J. Katto, and Y. Fan, "Approximated reconfigurable transform architecture for VVC," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.
[37] A. Kammoun et al., "Forward-inverse 2D hardware implementation of approximate transform core for the VVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 11, pp. 4340–4354, Nov. 2020.
[38] O. Guleryuz and M. Orchard, "Optimized nonorthogonal transforms for image compression," *IEEE Trans. Image Process.*, vol. 6, pp. 507–522, 1997.
[39] L. Chang, Z. Liu, X. Ji, and D. Wang, "Coding sensitive based approximation algorithm for power efficient VBS-DCT VLSI design in HEVC hardwired intra encoder," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3001–3005.
[40] I. Farhat, W. Hamidouche, A. Grill, D. Menard, and O. Déforges, "Lightweight hardware implementation of VVC transform block for ASIC decoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1663–1667.
[41] Y. Zhang and C. Lu, "Efficient algorithm adaptations and fully parallel hardware architecture of H.265/HEVC intra encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3415–3429, Nov. 2019.
[42] M. Masera, M. Martina, and G. Masera, "Adaptive approximated DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2714–2725, Dec. 2017.

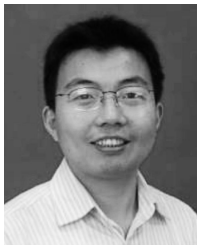
Zhijian Hao received the B.E. degree from the Department of Electronic Engineering, Xidian University, Shaanxi, China, in 2019, and the Ph.D. degree in microelectronics from Fudan University, Shanghai, China, in 2024. He is currently a Post-Doctoral Researcher with the College of Microelectronics, Xidian University. His research interests include image processing, video processing, video coding, and associated VLSI architecture.

Chenlong He received the B.S. degree from Fudan University, Shanghai, China, in 2022, where he is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Chips and Systems. His research interests include image processing, video processing, and video coding.

Jiaming Liu (Graduate Student Member, IEEE) received the B.E. degree in electronic science and technology and the M.S. degree in circuit and systems from Southwest Jiaotong University, Chengdu, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai, China. His research interests include image processing, channel coding, machine learning, and associated VLSI architecture.

Qi Zheng received the B.E. degree from the School of Microelectronics, Fudan University, Shanghai, China, in 2021. She is currently pursuing the Ph.D. degree with the State Key Laboratory of Integrated Chips and Systems, Fudan University. Her research interests include image and video processing, generative coding, and perceptual quality assessment.

Jinchang Xu received the B.E. degree from the School of Science, Shanghai University, Shanghai, China, in 2019, and the M.S. degree from the School of Software and Microelectronics, Peking University, Beijing, China, in 2022. His research interests include image and video processing and generative coding.



Peijun Ma received the Ph.D. degree in microelectronics and solid-state electronics from Xidian University, Xi'an, China, in 2001. He is currently a Professor with the School of Microelectronics, Xidian University.

Xiaohua Ma (Member, IEEE) received the Ph.D. degree in microelectronics and solid-state electronics from Xidian University, Xi'an, China, in 2007. He is currently a Professor and the Vice Dean with the School of Microelectronics, Xidian University.

Yue Hao (Senior Member, IEEE) received the B.S. degree in semiconductor physics and devices from Xidian University, Xi'an, China, in 1982, and the Ph.D. degree in computing mathematics from Xi'an Jiaotong University, Xi'an, in 1990. He is currently a Professor with the School of Microelectronics, Xidian University, where he is also the Vice President.