*electronics*

MDPI

*Article*

# Area-Time Efficient Two-Dimensional Reconfigurable Integer DCT Architecture for HEVC

**Pramod Kumar Meher [1], Siew-Kei Lam [2], Thambipillai Srikanthan [2], Dong Hwan Kim [3] and Sang Yoon Park [3,*]**

[1] Sandhaan Labs Private Limited, Bhubaneswar 751016, Odisha, India; pkmeher@sandhaanlabs.in
[2] School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore; siewkei_lam@pmail.ntu.edu.sg (S.-K.L.); astsrikan@ntu.edu.sg (T.S.)
[3] Department of Electronic Engineering, Myongji University, Yongin 17058, Korea; kdhwan1995@naver.com
* Correspondence: sypark@mju.ac.kr; Tel.: +82-31-330-6751

**Abstract:** In this paper, we present area-time efficient reconfigurable architectures for the implementation of the integer discrete cosine transform (DCT), which supports all the transform lengths to be used in High Efficiency Video Coding (HEVC). We propose three 1D reconfigurable architectures that can be configured for the computation of the DCT of any of the prescribed lengths such as 4, 8, 16, and 32. It is shown that matrix multiplication schemes involving fewer adders can be used to derive parallel architectures for 1D integer DCT of different lengths. A novel transposition buffer is designed to be used for the proposed 2D DCT architecture, which offers double the throughput without increasing the size of the transposition buffer. We determine the optimal pipeline locations in the proposed design through the precise estimation of propagation delays and the critical path so that the area-delay-product is optimized and all the output samples are obtained in the same cycle in spite of the recursive nature of the structure. Implementation results show that the proposed 2D integer DCT architectures provide significantly higher throughput per unit area than the existing designs for HEVC.

**Keywords:** discrete cosine transform (DCT); High Efficiency Video Coding (HEVC); H.265; integer DCT; video coding



**Citation:** Meher, P.K.; Lam, S.-K.; Srikanthan, T.; Kim, D.H.; Park, S.Y. Area-Time Efficient Two-Dimensional Reconfigurable Integer DCT Architecture for HEVC. *Electronics* **2021**, *10*, 603. https://doi.org/10.3390/electronics10050603

Academic Editor: Inhee Lee

Received: 26 January 2021
Accepted: 26 February 2021
Published: 5 March 2021



## 1. Introduction

The discrete cosine transform (DCT) is a core operation in video compression due to its near-optimal de-correlation efficiency [1,2]. The Joint Collaborative Team-Video Coding (JCT-VC) has defined the integer DCTs for transform lengths, $N = 4$, 8, 16, and 32 for the video coding standard H.265/HEVC (High Efficiency Video Coding) [3,4]. Low-cost hardware implementations of the integer DCTs for HEVC were proposed in [5–13].

Ahmed et al. [8] proposed a multiplier-less lifting-based approach applied to a sparse decomposition of the DCT matrices. Shen et al. [14] proposed another multiplier-less implementation using a multiple constant multiplication (MCM) approach for the integer DCT of lengths four and eight. For the DCT of lengths 16 and 32, however, they used conventional multipliers. Park et al. [9] used Chen's factorization of the DCT where the butterfly operation of the DCT computation was implemented by the processing element (PE) with only shifters, adders, and multiplexors. Budagavi and Sze [10] proposed a unified hardware structure that can be used for forward, as well as inverse DCTs. Efficient MCM-based architectures and transposition buffers for integer DCT for HEVC were proposed in [12,13]. However, the transposition unit has a significant overhead of complexity in this design.

As a major deviation from the earlier video coding standards, HEVC allows DCTs of four different lengths. The DCT unit for HEVC is therefore required to be configurable to support different transform lengths, $N = 4$, 8, 16, and 32. Based on this requirement, in this paper, we propose efficient reconfigurable structures of the DCT unit that can be used

for integer DCTs of different lengths. Specifically, three reconfigurable DCT architectures based on a constant matrix multiplication (CMM) and the MCM are proposed. 2D integer DCT architectures having novel transposition buffers are also proposed. The innovations of the proposed architectures are summarized as follows.

- A hardware-oriented algorithm for integer DCT computation for HEVC is proposed.
- Three different flexible hardware architectures for the integer DCT are proposed, each with advantages in terms of area, delay, or power.
- A novel matrix-vector-product unit that involves fewer adders than the existing method [12,13] is proposed.
- A novel 2D integer DCT architecture having double the throughput and less latency (without increasing the size of the transposition buffer) is proposed.
- A novel low-cost pipeline strategy is proposed to reduce the critical path of the proposed 1D integer DCT.
- Comparisons with existing methods are provided in terms of various metrics such as gate counts, maximum usable frequency, throughput, latency, etc.

The rest of the paper is organized as follows. In the next section, we discuss the key features of the integer DCT for HEVC. We describe three proposed reconfigurable architectures of 1D integer DCT in Section 3. In Section 4, we discuss the implementation of 2D DCT. In Section 5, we compare the area and time complexities of the proposed designs with those of the existing designs. Section 6 presents the conclusions.

## 2. Key Features of Integer DCT for HEVC

The $N$-point integer DCT of an input vector $\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]$ is given by:

$$\begin{bmatrix} y(0) \\ y(1) \\ \cdot \\ \cdot \\ \cdot \\ y(N-2) \\ y(N-1) \end{bmatrix} = \mathbf{C}_N \begin{bmatrix} x(0) \\ x(1) \\ \cdot \\ \cdot \\ \cdot \\ x(N-2) \\ x(N-1) \end{bmatrix} \tag{1}$$

where the integer DCT kernel $\mathbf{C}_N$ is an $N \times N$ matrix of integers and $\mathbf{y} = [y(0), y(1), \ldots, y(N-1)]$ is an output vector. The kernel matrices $\mathbf{C}_N$ for $N = 4, 8, 16$, and 32 for HEVC were defined by JCT-VC [3]. The $N$-point integer DCT for HEVC can be computed by a partial butterfly approach using an $(N/2)$-point DCT and a product of the $(N/2) \times (N/2)$ matrix by an $(N/2)$-point vector as:

$$\begin{bmatrix} y(0) \\ y(2) \\ \cdot \\ \cdot \\ \cdot \\ y(N-4) \\ y(N-2) \end{bmatrix} = \mathbf{C}_{N/2} \begin{bmatrix} a(0) \\ a(1) \\ \cdot \\ \cdot \\ \cdot \\ a(N/2-2) \\ a(N/2-1) \end{bmatrix} \tag{2a}$$

and

$$\begin{bmatrix} y(1) \\ y(3) \\ \cdot \\ \cdot \\ \cdot \\ y(N-3) \\ y(N-1) \end{bmatrix} = \mathbf{S}_{N/2} \begin{bmatrix} b(0) \\ b(1) \\ \cdot \\ \cdot \\ \cdot \\ b(N/2-2) \\ b(N/2-1) \end{bmatrix} \tag{2b}$$

where:

$$a(i) = x(i) + x(N-i-1) \tag{3a}$$
$$b(i) = x(i) - x(N-i-1), \tag{3b}$$

for $i = 0, 1, \cdots, N/2-1$. $\mathbf{C}_{N/2}$ is an $(N/2)$-point integer DCT kernel of size $(N/2) \times (N/2)$. $\mathbf{S}_{N/2}$ is also an integer matrix of size $(N/2) \times (N/2)$ derived from the first $N/2$ columns of the odd-indexed rows of $\mathbf{C}_N$, such that the $(i,j)^{\text{th}}$ entry of $\mathbf{S}_{N/2}$ can be defined as:

$$s_{N/2}^{i,j} = c_N^{2i+1,j} \text{ for } 0 \leq i, j \leq N/2-1 \tag{4}$$

where $c_N^{2i+1,j}$ is the $(2i+1, j)$th entry of $\mathbf{C}_N$. Note that all even DCT outputs are given by Equation (2a), while odd DCT outputs are given by Equation (2b). An $(N/2)$-point DCT given by Equation (2a) is again decomposed into an $(N/4)$-point DCT and a product of the integer matrix $\mathbf{S}_{N/4}$ of size $(N/4) \times (N/4)$ with an $(N/4)$-point vector. This decomposition can continue recursively till the $N$-point DCT is expressed in terms of the four-point DCT and a product of the $4 \times 4$ matrix and four-point vector.

## 3. Proposed Reconfigurable Architectures for 1D Integer DCT

In this section, we propose three reconfigurable architectures for the computation of the 1D integer DCT of different lengths for HEVC.

### *3.1. Proposed 1D DCT Architecture-1*

Figure 1 shows the proposed 1D DCT Architecture-1. The architecture in Figure 1a is for the integer DCTs of lengths $N = 8$, 16, and 32, whereas the architecture in Figure 1b is for the computation of the four-point integer DCT. Figure 1c describes the structure of the shift-add unit in Figure 1b. Sixteen- or eight- or four-point integer DCTs can be computed by the 32-point DCT structure by setting the *sel* control signal to the MUX unit appropriately. The input adder unit (IAU) in Figure 1a computes $a(i)$ and $b(i)$ for $i = 0, 1, \cdots, N/2-1$ by Equation (3). The MUX unit, which consists of $N/2$ 2:1 MUXes, selects either $a(i)$ or $x(i)$ for $i = 0, 1, \cdots, N/2-1$, depending on whether it is used to compute Equation (2a) or the DCT of a lower size, respectively. Odd-indexed coefficients of $y(i)$ for $i = 1, 3, \cdots, N-1$ are computed by the matrix-vector-product unit (MVPU) according to Equation (2b). The computation of Equation (2b) could be realized as a CMM problem [15–18]. We find that the algorithm of Boullis and Tisserand [18] is very efficient when it is used for the computation of the MVPU. We generated the MVPU algorithm for $\mathbf{S}_4 \cdot \mathbf{b}_4$, $\mathbf{S}_8 \cdot \mathbf{b}_8$, and $\mathbf{S}_{16} \cdot \mathbf{b}_{16}$ and list these in Table 1, where $\mathbf{b}_{N/2} = [b(0), b(1), \cdots b(N/2-1)]$. The complexity of the proposed 1D DCT Architecture-1 is compared to that in [12] in Table 2 since the DCT computation in [12] also uses a partial butterfly approach based on Equations (2a) and (2b). Table 1 of [12] shows the algorithm of the reference DCT architecture to be compared. Specifically, the IAUs of the two structures are the same, and the MVPU of the proposed structure corresponds to the MCM of [12]. The total number of adders listed in the seventh column of Table 2 can be referred to Table 2 of [12]. The proposed Architecture-1 uses 14.6% fewer adders compared with [12] for $N = 32$. Note that pipeline registers are not used for the implementation of the MVPU. The structure requires only input registers after the IAU to hold the incoming data.

**Table 1.** Computation of $\mathbf{S}_{N/2} \cdot \mathbf{b}_{N/2}$ in the matrix-vector-product unit ($b_i = b(i)$ and $y_{2\times i+1} = y(2 \times i + 1)$ for $i = 0, 1, 2, ..., N/2 - 1$).

| Computation of $\mathbf{S}_4 \cdot \mathbf{b}_4$ in the matrix-vector-product unit (MVPU) for 8-point DCT computation |
|---|
| $t_9 = b_0 + 4b_0$; $t_8 = b_1 - 8b_3$; $t_7 = 4b_2 - b_2$; $t_6 = 2b_0 - b_3$; $t_5 = 2b_1 - b_3$; $t_4 = b_1 - 2b_2 - 16b_3$; $t_3 = b_0 - 2t_5$; $t_2 = 4t_7 + 8b_3 - b_2$; $t_1 = 8b_0 + b_1 + 4b_1 - 2b_0$; |
| $y_1 = 16t_1 + t_3 + 16t_7 - t_4 - 8b_0$; $y_3 = 16t_9 - 8t_2 - 2t_8 - t_9 - 16b_1 - b_2 - 2b_3$; $y_5 = 8t_1 + t_6 + 16b_2 - t_4 - 64t_5 - 4b_3$; $y_7 = t_2 + 16t_3 + t_6 + 16t_8 + 64b_2 - 2b_1$; |
| Computation of $\mathbf{S}_8 \cdot \mathbf{b}_8$ in the MVPU for 16-point DCT computation |
| $t_{21} = b_3 - b_6$; $t_{20} = b_2 + b_7$; $t_{19} = b_0 - b_5$; $t_{18} = b_0 + b_1$; $t_{17} = b_2 - b_4$; $t_{16} = 4b_0 + 2b_5 - b_6$; $t_{15} = b_0 - 4b_2 - 2b_4$; $t_{14} = 2b_3 + 4b_5 - b_7$; $t_{13} = 2b_1 - 4b_4 + b_5$; |
| $t_{12} = 2b_0 - 4b_1 - b_3$; $t_{11} = b_4 + 4b_6 + 2b_7$; $t_{10} = b_1 + 2b_2 - 4b_7$; $t_9 = b_2 + 4b_3 + 2b_6$; $t_8 = 16t_{12} + b_0 + 64b_4 + 8b_6 - 8t_{20}$; $t_7 = 16t_{11} + 8t_{18} + 64b_3 + b_7 - 8b_5$; |
| $t_6 = 8t_{18} + 8b_4 - 16t_9 - 64b_5 - b_6$; $t_5 = 16t_{13} + b_1 - 8t_{21} - 64b_2 - 8b_7$; $t_4 = 16t_{16} + 8t_{20} + b_5 - 64b_1 - 8b_4$; $t_3 = 16t_{10} + 8t_{19} + b_2 + 8b_3 - 64b_6$; |
| $t_2 = 16t_{14} + 64b_0 + 8b_1 + b_3 - 8t_{17}$; $t_1 = 16t_{15} + 8t_{21} + 64b_7 - b_4 - 8b_5$; $y_1 = 128t_{18} + 2b_0 + b_4 - t_5 - t_6 - t_{14} - 32b_0 - b_5$; |
| $y_3 = t_9 + 128t_{19} + b_1 + b_3 + 32b_5 - t_7 - t_8 - 2b_5$; $y_5 = t_1 + t_2 + t_{10} + b_5 + 2b_6 - 128t_{21} - 32b_6 - b_7$; $y_7 = t_3 + t_4 + b_1 + 2b_4 - t_{12} - 128t_{17} - 32b_4 - b_7$; |
| $y_9 = t_4 + b_0 + 2b_3 + 128b_3 - t_3 - t_{11} - 32b_3 - 128b_5 - b_6$; $y_{11} = t_2 + 32b_1 + b_2 - t_1 - t_{16} - b_0 - 2b_1 - 128b_1 - 128b_4$; |
| $y_{13} = t_8 + t_{13} + 128t_{20} + 2b_2 - t_7 - 32b_2 - b_4 - b_6$; $y_{15} = t_6 + t_{15} + 128b_6 + 32b_7 - t_5 - b_2 - b_3 - 2b_7 - 128b_7$; |
| Computation of $\mathbf{S}_{16} \cdot \mathbf{b}_{16}$ in the MVPU for 32-point DCT computation |
| $t_{108} = b_8 + b_{14}$; $t_{107} = b_0 - b_{10}$; $t_{106} = b_{12} - 8b_{15}$; $t_{105} = b_8 - b_{13}$; $t_{104} = b_9 - b_{15}$; $t_{103} = b_2 - 16b_8$; $t_{102} = 2b_0 + b_5$; $t_{101} = b_0 - 2b_{14}$; $t_{100} = b_5 - b_7$; |
| $t_{99} = 8b_7 + b_{11}$; $t_{98} = 2b_7 - b_{10}$; $t_{97} = 32b_5 - b_{13}$; $t_{96} = b_8 - 2b_9$; $t_{95} = b_3 - 8b_9$; $t_{94} = b_{10} + 32b_{13}$; $t_{93} = b_6 + 8b_{11}$; $t_{92} = 2b_8 - b_{14}$; $t_{91} = b_8 + b_9$; |
| $t_{90} = b_4 + 2b_{12}$; $t_{89} = b_4 - 32b_8$; $t_{88} = 4b_4 - b_{10}$; $t_{87} = b_1 + 2b_3$; $t_{86} = b_{13} + b_{15}$; $t_{85} = b_1 - 2b_4$; $t_{84} = 2b_1 - b_8$; $t_{83} = 4b_5 - b_8$; $t_{82} = b_5 - 8b_{14}$; |
| $t_{81} = 2b_{11} + b_{12}$; $t_{80} = 8b_6 - b_{13}$; $t_{79} = b_1 + b_7$; $t_{78} = b_2 + 2b_{10}$; $t_{77} = b_8 - 16b_9$; $t_{76} = b_{14} + 2b_{15}$; $t_{75} = b_6 - b_8$; $t_{74} = b_1 + b_{15}$; $t_{73} = b_{11} - 8b_{14}$; |
| $t_{72} = 16b_6 + b_7$; $t_{71} = 2b_6 - b_{15}$; $t_{70} = 2b_1 - b_{11}$; $t_{69} = b_3 + 2b_4$; $t_{68} = b_0 - b_6$; $t_{67} = b_6 - 2b_7$; $t_{66} = b_2 + b_7$; $t_{65} = b_7 - 8b_9$; $t_{64} = b_0 - b_{12}$; |
| $t_{63} = b_4 - 2b_{12}$; $t_{62} = 16b_2 - b_5$; $t_{61} = b_6 - b_7$; $t_{60} = 2b_{11} - b_{12}$; $t_{59} = 4b_1 - b_3$; $t_{58} = 8b_3 - b_7$; $t_{57} = b_0 + b_{14}$; $t_{56} = b_0 + 4b_{13}$; $t_{55} = 2b_0 + b_{14}$; |
| $t_{54} = t_{57} - 2b_9$; $t_{53} = t_{91} + 2b_2$; $t_{52} = t_{55} + 2b_{12}$; $t_{51} = 2t_{70} + b_9$; $t_{50} = t_{87} + 4b_6$; $t_{49} = t_{61} - 2b_{13}$; $t_{48} = b_6 - 2b_{11} - b_{13}$; $t_{47} = b_2 + 2b_4 + b_9$; |
| $t_{46} = 4b_4 - t_{90}$; $t_{45} = t_{74} + 2b_{10}$; $t_{44} = 2b_3 + b_{14} - b_5$; $t_{43} = b_1 + b_{10} - 2b_{12}$; $t_{42} = t_{99} + 4b_{11}$; $t_{41} = t_{103} - 8b_6$; $t_{40} = 4b_3 - t_{69} - 2b_{11}$; |
| $t_{39} = 2b_3 + b_{11} - 2t_{81}$; $t_{38} = 16b_3 + 2b_4 - t_{67}$; $t_{37} = 2t_{63} - t_{60}$; $t_{36} = 2b_2 + 4b_5 + 2b_{13} - b_5$; $t_{35} = 4b_2 + 2b_5 + 2b_{10} - b_2$; $t_{34} = 2t_{78} - b_{10} - 2b_{13}$; |
| $t_{33} = 2b_5 + 4b_{13} - 2b_{10} - b_{13}$; $t_{32} = 8b_2 + 64b_9 - b_{10} - 16b_{13}$; $t_{31} = t_{62} - 8t_{80}$; $t_{30} = b_3 + 4b_9 + 2b_{15} - b_{14}$; $t_{29} = 16b_7 + 64b_{12} + b_{15} - 8b_8$; |
| $t_{28} = t_{56} + 4t_{84} - 8b_6$; $t_{27} = 4t_{66} + 8b_9 + 8b_{14} - b_{15}$; $t_{26} = 4t_{75} + b_1 + 8b_4 + 8b_{12}$; $t_{25} = t_{58} + 4b_4 + 4b_9 - 4b_7$; $t_{24} = 4t_{85} + 4b_0 + b_9 - 8b_3$; |
| $t_{23} = 4t_{107} - t_{65} - 8b_1$; $t_{22} = t_{93} + 4b_{14} - 8b_{12} - 4b_{15}$; $t_{21} = 4t_{71} + t_{83} - 8b_{14}$; $t_{20} = t_{77} + 32t_{85} + 8b_0 - 8t_{76}$; $t_{19} = t_{72} + 16t_{101} + 8b_{15} - 8b_1 - 64b_{11}$; |
| $t_{18} = 8t_{98} + 16t_{102} + 8b_{15} - 64b_1 - b_{13}$; $t_{17} = 8t_{82} + 8b_0 + 16b_{10} - t_{103} - 32b_{15}$; $t_{16} = 4t_{46} + 2t_{86} + 64b_0 - t_{94} - b_3 - 8b_3 - 32b_7$; |
| $t_{15} = 4t_{40} + t_{78} + 32t_{98} - 2b_8 - b_{11} - 32b_{15}$; $t_{14} = 4t_{37} + 2t_{100} + 32b_0 + b_4 - t_{97} - 64b_8$; $t_{13} = 4t_{36} + 32t_{74} + 2t_{104} + 64b_6 - t_{105}$; |
| $t_{12} = 4t_{39} + t_{102} + 2t_{103} - 32b_2 - b_{12} - 64b_{15}$; $t_{11} = 4t_{33} + 32t_{91} + t_{107} + 64b_1 - 2t_{108}$; $t_{10} = 4t_{34} + 32t_{54} + t_{66} - 2t_{68}$; |
| $t_9 = 4t_{35} + 32t_{61} + 2t_{79} + b_5 + 64b_{14} - b_{15}$; $t_8 = 2t_{52} + 2t_{88} + t_{95} + 8b_{13} - 32t_{33} - 16t_{85} - 2t_{89}$; $t_7 = 2t_{38} + t_{73} + 64b_0 + 2b_2 + 8b_3 + 8b_{10} - 32t_{34} - 2t_{95}$; |
| $t_6 = 32t_{36} + 2t_{51} + t_{63} + 2t_{80} + 8t_{106} + 4b_8 + 32b_{12} - 8b_5$; $t_5 = 32t_{35} + 8t_{42} + 2t_{50} + 4b_{15} - t_{81} - 2t_{82} - 8b_2$; |
| $t_4 = 32t_{37} + 2t_{49} + 4t_{52} + 16t_{83} + t_{85} - 8b_6 - 32b_6$; $t_3 = 2t_{45} + 32t_{46} + 16t_{56} + b_9 - 4t_{38} - 2t_{59} - 32b_1$; |
| $t_2 = 32t_{39} + 2t_{54} + 4t_{62} + t_{93} + 2t_{106} - 8t_{108} - 32b_{14}$; $t_1 = 2t_{53} + 4t_{87} - 32t_{40} - 2t_{99} - 8t_{104} - 32b_9 - 64b_{10} - b_{14}$; |
| $y_1 = t_5 + 2t_{11} + t_{49} + t_{69} + 4t_{72} + t_{108} + 128b_0 + 32b_{12} - 8t_{24} - 2t_{24}$; $y_3 = t_2 + 2t_{16} + 4t_{41} + t_{84} - 2t_{23} - 8t_{23} - t_{47} - t_{86} - 128b_{10}$; |
| $y_5 = 2t_{31} + 64t_{55} + 4b_1 + 2b_{12} - t_4 - t_{15} - t_{18} - t_{30} - 64t_{105}$; $y_7 = t_8 + 2t_{20} + 2t_{48} + 64t_{64} + t_{64} + 128b_9 - t_9 - t_{41} - 8t_{58} - 4b_3$; |
| $y_9 = t_7 + t_{13} + t_{19} + t_{102} + 128b_{14} - 16t_{25} - 2t_{43} - t_{89} - 4b_{11}$; $y_{11} = t_{16} + 2t_{17} + 8t_{65} + b_4 + 64b_5 + 4b_6 + 2b_7 - t_2 - t_{31} - t_{51} - 128b_8$; |
| $y_{13} = t_4 + 8t_{27} + 2t_{27} + t_{44} + b_0 + 64b_0 + 4b_5 + b_{10} - 2t_{15} - t_{96} - 32b_1 - 128b_2$; $y_{15} = t_7 + 2t_{26} + 8t_{26} + t_{90} + 64t_{92} + 2t_{102} + t_{104} - 2t_{13} - t_{57} - 32b_{11}$; |
| $y_{17} = 2t_{10} + 8t_{25} + 2t_{30} + 16t_{42} + t_{68} + t_{73} - t_6 - t_{45} - 16t_{59}$; $y_{19} = 2t_{14} + t_{71} + 4t_{94} - t_1 - 8t_{28} - 2t_{28} - t_{43} - 32t_{76} - t_{100}$; |
| $y_{21} = 2t_{18} + t_{32} + 64t_{67} + b_6 + b_{11} - t_3 - t_{12} - 4t_{92} - 2t_{96} - 2b_4 - 64b_{10}$; $y_{23} = t_6 + t_{10} + t_{20} + b_{11} - 2t_{29} - 2t_{44} - 64t_{70} - 64t_{75} - t_{88}$; |
| $y_{25} = t_5 + 2t_{19} + t_{29} + 2t_{47} + 4b_{12} - t_{11} - 64t_{71} - t_{95} - 64b_3 - b_{13}$; $y_{27} = t_1 + t_{14} + 2t_{32} + 64t_{66} - t_{17} - t_{50} - 2t_{101} - 64b_1 - b_{12} - 128b_{15}$; |
| $y_{29} = t_3 + 8t_{21} + 2t_{21} + t_{48} + t_{101} + b_2 - 2t_{12} - 4t_{97} - b_7 - 64b_7 - 32b_9$; $y_{31} = t_8 + 2t_9 + 4t_{77} - 2t_{22} - 8t_{22} - t_{53} - t_{60} - t_{79} - 32b_3 - 128b_{15}$; |

Adding pipeline registers in the critical path is the general practice in VLSI design. However, in the recursive design as in the proposed DCT, the locations of pipeline registers need to be carefully decided. Furthermore, the pipelining in the MVPU may significantly increase the silicon area. We could find optimal pipeline locations through precise estimation of propagation delays in the critical path so that area-delay-product can be optimized and all the output samples can be obtained in the same cycle even in the recursive structure. Architecture-1 for the eight-point DCT involves four pipeline registers before the MVPU to reduce the propagation delay, as shown in Figure 1a. In order to obtain all eight DCT coefficients in the same cycle, four more registers are inserted after the first four adders in the four-point integer DCT unit, as shown in Figure 1b. Similarly, eight and 16 registers are located before the MVPU for 16- and 32-point DCTs, respectively. Note that Architecture-1

involves two pipeline stages for any of the DCT sizes, $N = 4$, 8, 16, and 32. The array of AND gates is used to disable the IAU, pipeline registers, and MVPU in the computation of the DCT of a lower size in order to reduce the power consumption.

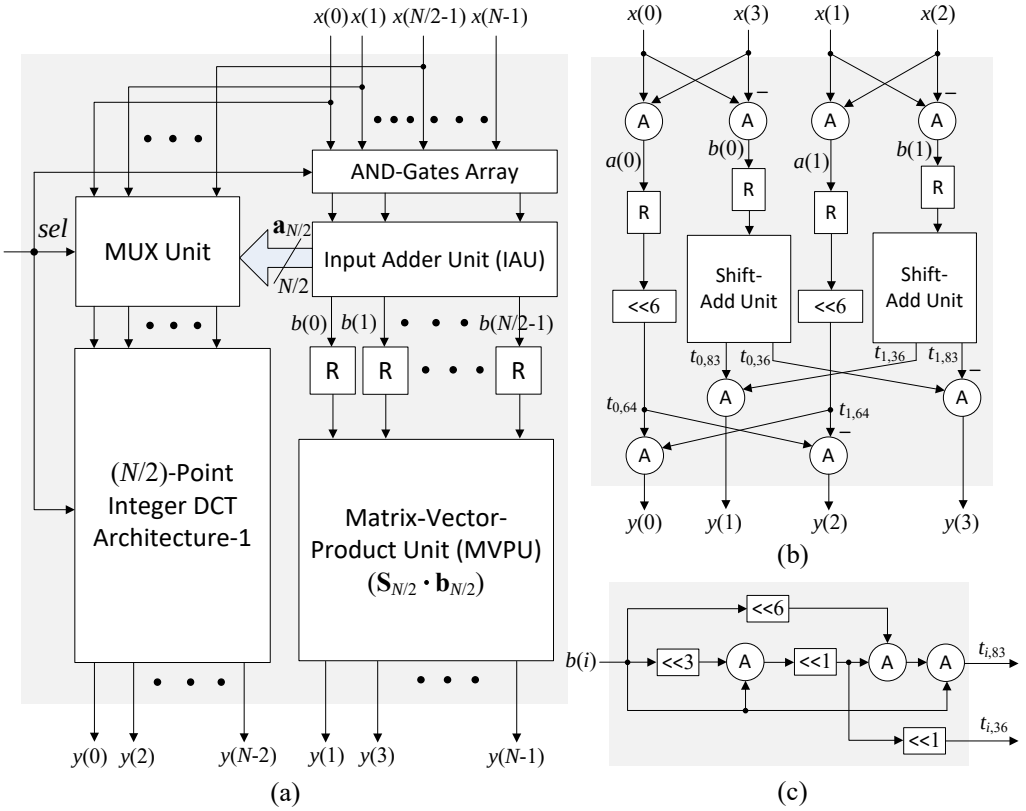


**Figure 1.** (**a**) Proposed Architecture-1 for $N = 8$, 16, and 32. (**b**) Four-point integer DCT architecture. (**c**) Shift-add unit used in (**b**).

**Table 2.** Complexity comparison in terms of the number of adders. IAU, input adder unit.

| *N* | IAU | MVPU | | *N*/2-DCT | | Total | |
|---|---|---|---|---|---|---|---|
| | | [12] | **Figure 1** | [12] | **Figure 1** | [12] | **Figure 1** |
| 4 | 4 | 10 | | | | 14 | |
| 8 | 8 | 28 | 33 | 14 | | 50 | 55 |
| 16 | 16 | 120 | 112 | 50 | 55 | 186 | 183 |
| 32 | 32 | 464 | 367 | 186 | 183 | 682 | 582 |

### *3.2. Proposed 1D DCT Architecture-2*

The proposed Architecture-2 is an extension of Architecture-1, as shown in Figure 2, which incorporates coarse-grained reconfiguration with additional hardware units to maintain the same throughput rate for all the transform lengths. It uses an extra $(N/2)$-point architecture over the structure of Figure 1a, which takes the input $[x(N/2),...,x(N-1)]$. The output of this additional $(N/2)$-point architecture is multiplexed with the output of the MVPU by MUX Unit-2. The 32-point architecture can compute one 32-point DCT, two 16-point DCTs, four 8-point DCTs, and eight 4-point DCTs, while the throughput remains the same as the 32 DCT coefficients per cycle irrespective of the desired transform size.

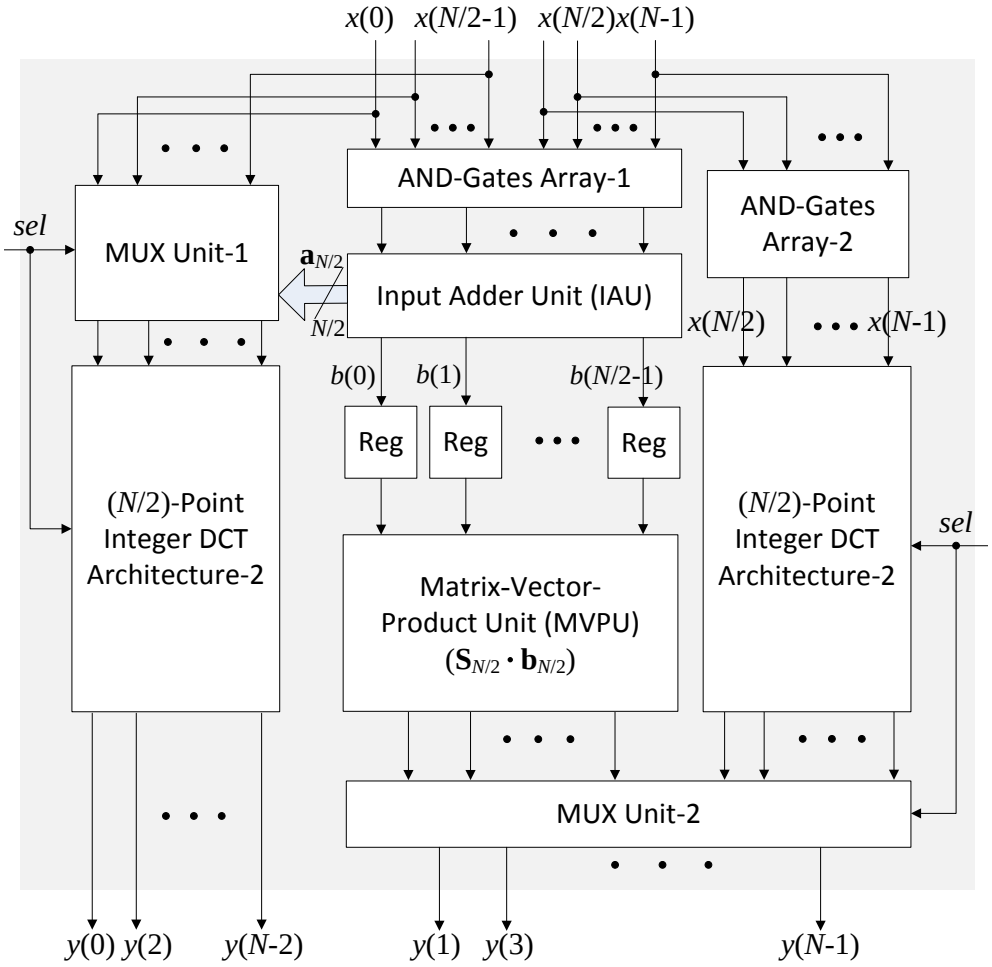


**Figure 2.** Proposed Architecture-2 for $N = 8$, 16, and 32. The four-point integer DCT architecture is same as the one shown in Figure 1b.

### *3.3. Proposed 1D DCT Architecture-3*

Figure 3 shows the proposed Architecture-3 for 1D DCT. It incorporates a fine-grained reconfiguration for different transform lengths. For the computation of the $N$-point DCT, the $(N/2)$-point integer DCT unit computes Equation (2a), whereas the configurable shift-add units (CSAU) and the output adder unit (OAU) perform the computation of (2b). In order to reuse this architecture for the computation of the $(N/2)$-point DCT, the $(N/2)$-point integer DCT unit computes an $(N/2)$-point DCT, and the CSAU and OAU compute the other $(N/2)$-point DCT providing two $(N/2)$-point DCTs. Similarly, for the computation of the $(N/4)$-point DCT, the $(N/2)$-point DCT unit computes two $(N/4)$-point DCTs, and the CSAU and OAU compute the other two $(N/4)$-point DCTs, producing four $(N/4)$-point DCTs. Figure 4a shows the structure of CSAU-1, which is one out of four CSAUs used for the eight-point DCT structure. For the computation of the four-point DCT, CSAU-1 multiplies $x(5)$ with $[64, 36, 64, 83]$ in the second column of $\mathbf{C}_4$ (ignoring the signs of the coefficients). This is part of the four-point DCT computation of $[x(4),x(5),x(6),x(7)]$. For the computation of the eight-point DCT, CSAU-1 multiplies $b(1)$ with $[75, 18, 89, 50]$, which is the second column of $\mathbf{S}_4$. As shown in Figure 4a, the CSAU can be designed based on the SAU with several MUXes; however, it does not need any additional adder. Each CSAU is designed differently so that the last four MUXes have different combinations of coefficients corresponding to the desired configuration. Since the signs of the coefficients in each configuration may also be different, the OAU is implemented using add/subunits instead of adders or subtractors, as shown in Figure 4b. When the 16-point DCT structure is to be used for the computation of the four-point DCT,

the intermediate results of Stage-2 in the three-stage adder-tree in the OAU are directed to the output $y(i)$, unlike the computation for eight- and 16-point DCTs. Therefore, the MUX Unit-3, which involves an array of 2:1 MUXes, is used to select the desired outputs of Stage-2 or outputs of Stage-3. Similarly, a MUX Unit-3 consisting of 3:1 MUXes is used to select the desired outputs of Stages-2, 3, and 4 for the 32-point DCT structure. Since MUX Unit-3 is not required for the eight-point DCT structure, it is indicated by dashed lines as shown in Figure 3.

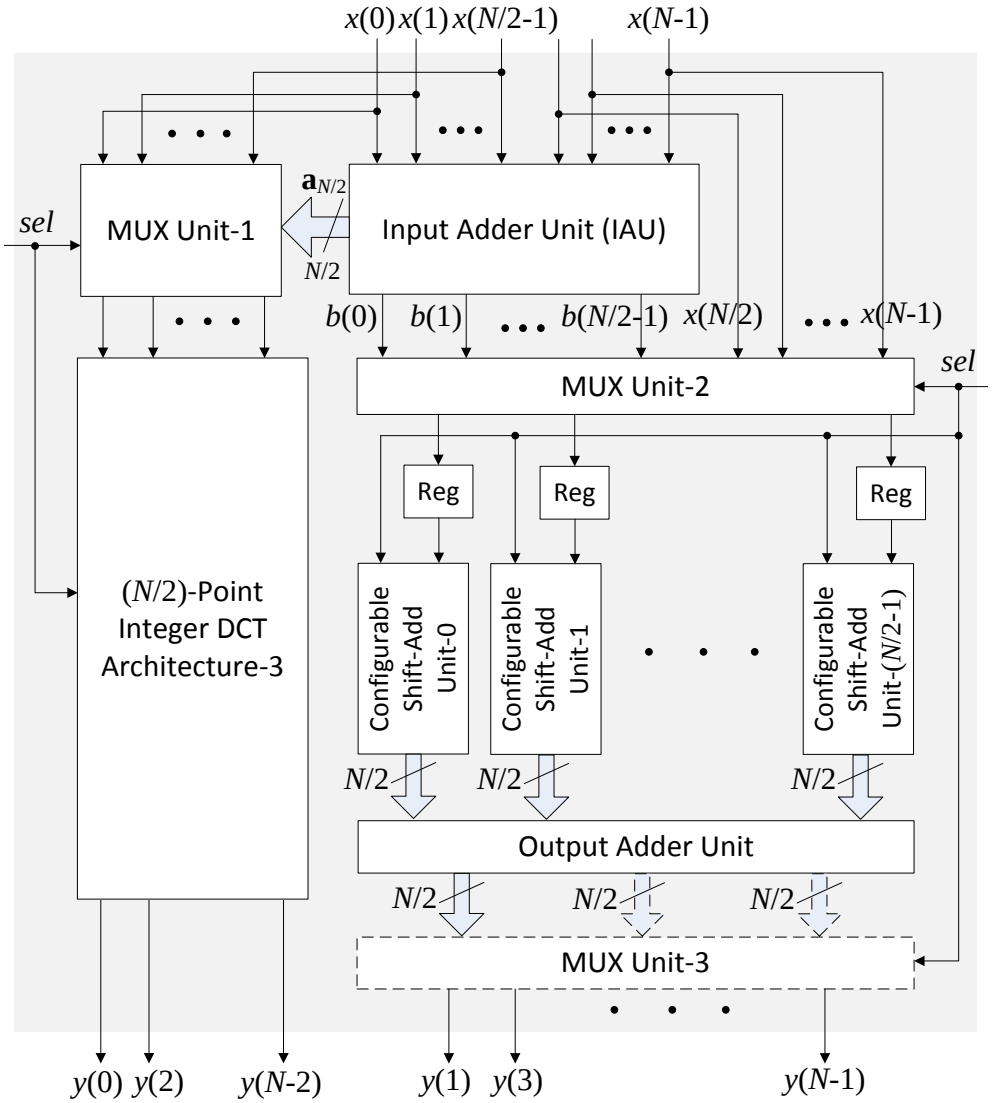


**Figure 3.** Proposed Architecture-3 for $N = 8$, 16, and 32. The number of output samples of the output adder unit (OAU) is $N/2$, $N$, and $3N/2$ for $N = 8$, 16, and 32, respectively. The four-point integer DCT architecture is the same as the one shown in Figure 1b.

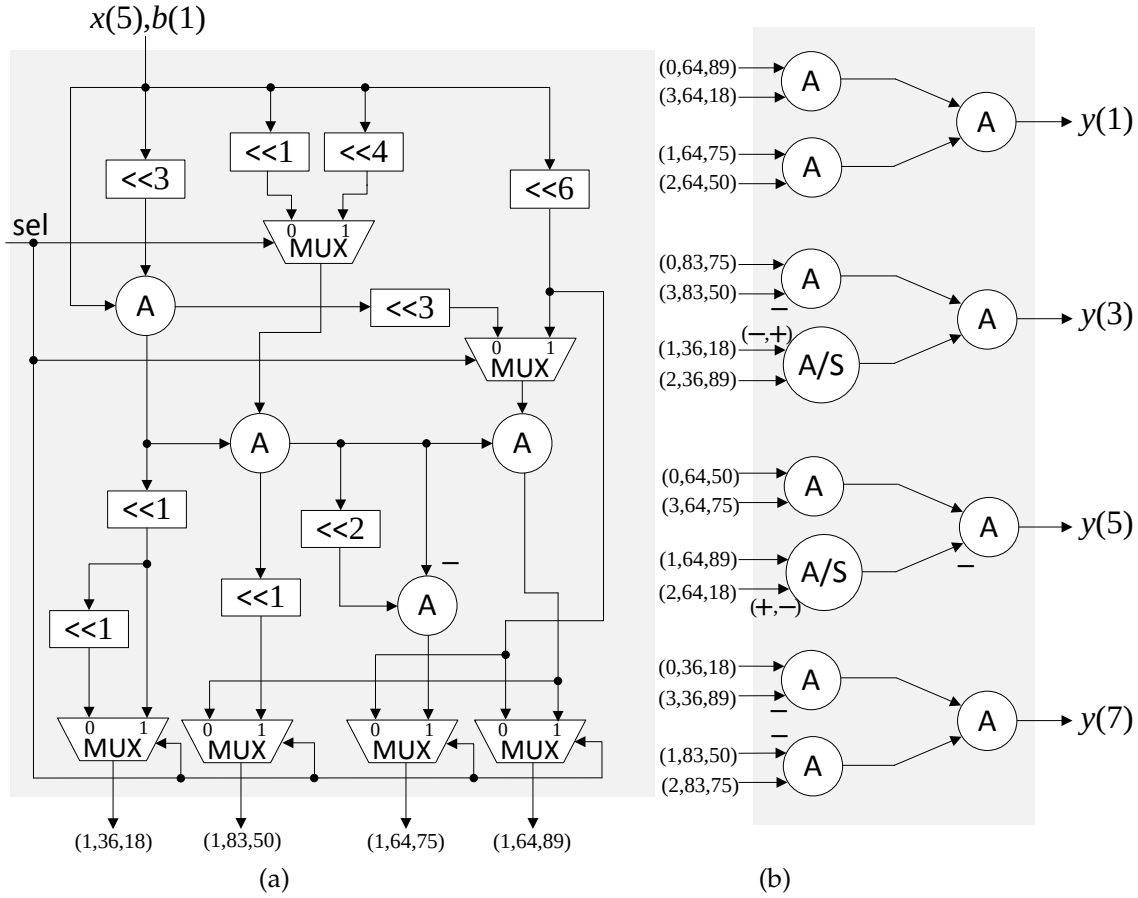


**Figure 4.** (**a**) Configurable Shift-Add Unit (CSAU)-1 for the eight-point DCT structure. (**b**) The OAU for the eight-point DCT structure. Input of the OAU $(a, b, c)$ means the output of CSAU-$a$ where $b$ and $c$ are used for the four- or eight-point DCT computations according to *sel*, respectively.

## 4. High-Throughput 2D Integer DCT Architecture

Figure 5 shows the proposed architecture for 2D integer DCT using $N$-point reconfigurable 1D integer DCTs. It consists of two sections corresponding to two stages of 2D DCT computation by row-column decomposition based on the separable property of 2D DCT. The input section of the proposed structure consists of two $N$-point 1D integer DCTs, to compute the 1D DCT of all the $N$ columns of the $N \times N$ input matrix. The first $N$-point DCT unit (S1-A) computes the DCTs of the first $(N/2)$ columns of the $N \times N$ input matrix, while the second $N$-point DCT unit (S1-B) computes the DCTs of the last $(N/2)$ columns of the input matrix. A transposition unit consisting of four $(N/2) \times (N/2)$ buffers is employed to reorder the DCT coefficients of different columns to be fed to the output section row-wise. The output section consists of two 1D parallel $N$-point DCT units, to compute the 1D DCT of all the $N$ rows of the $N \times N$ intermediate output matrix. The first $N$-point DCT unit (S2-A) computes the DCT of even rows of the intermediate matrix (available from *SR_Even_A* and *SR_Even_B*). The second $N$-point DCT unit (S2-B) computes the DCT of the odd rows in the intermediate matrix (available from *SR_Odd_A* and *SR_Odd_B*).

In each cycle, two columns of the $N \times N$ input matrix **x** are fed to the input section of the proposed 2D DCT architecture as inputs. For example, if Column-0 and column-$(N/2)$ of the input matrix are fed, respectively, to the DCT units S1-A and S1-B in the input section in the current clock cycle, then Column-1 and column-$(N/2+1)$ are fed concurrently to S1-A and S1-B, respectively, in the next clock cycle. Hence, the entire $N \times N$ input matrix is fed to the proposed architecture in $N/2$ consecutive clock cycles. The first two rows of 2D

DCT from the output section are available after $(N/2+2)$ clock cycles where $N/2$ clock cycles are for transposition and two clock cycles for the input and output sections.

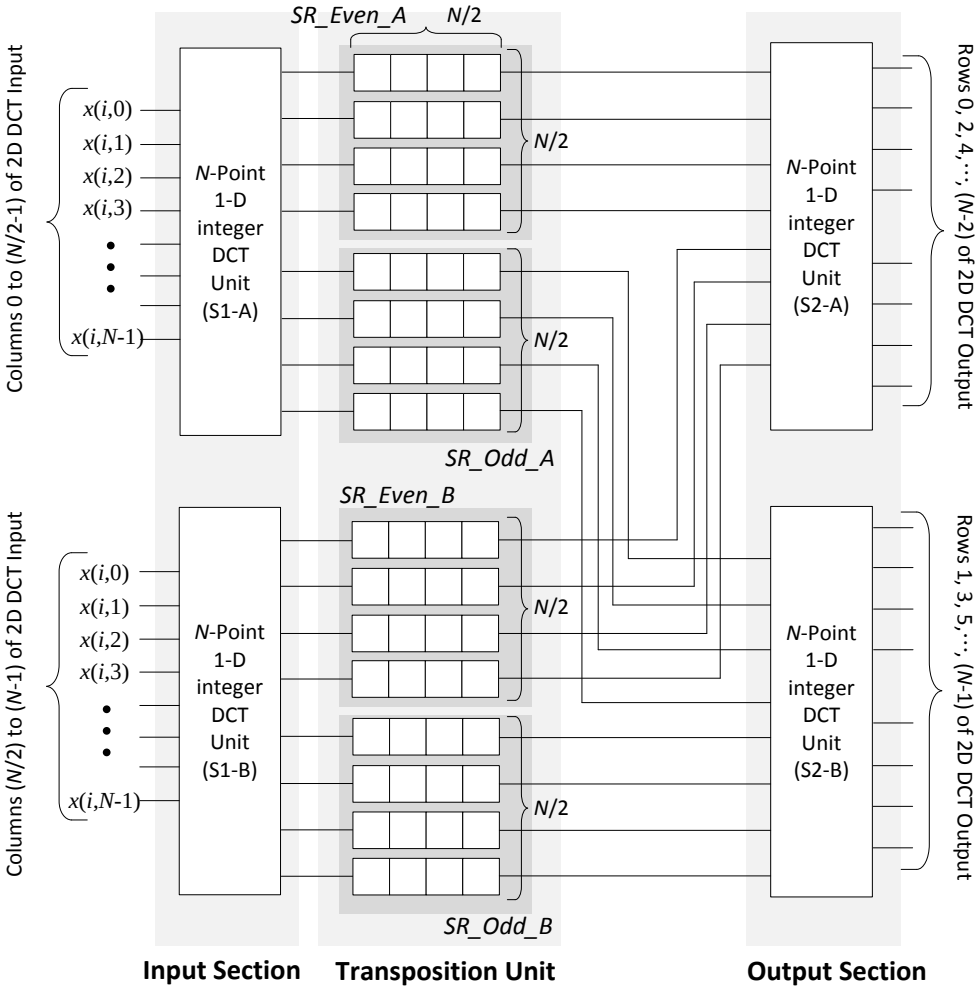


**Figure 5.** Proposed architecture for the 2D integer DCT of size $N \times N$.

In every $N/2$ cycle, the proposed 2D DCT architecture can process a new $N \times N$ input matrix, resulting in a throughput rate of $2N$ DCT coefficients per clock cycle. Therefore, the 2D DCT architecture using 32-point Architecture-1 has 8, 16, 32, and 64 coefficients per clock cycle for the 4-, 8-, 16-, and 32-point DCT computations, respectively. Furthermore, the 2D DCT architecture employing 32-point Architecture-2 or 3 can produce 64 DCT coefficients per clock cycle irrespective of the transform size.

The conventional way to obtain double the throughput is to use double the hardware comprised of two DCT units and the transposition buffers. However, the proposed 2D DCT transposition technique does not increase the transposition buffer size to increase the throughput. The $32 \times 32$ transposition buffer occupies a 2.1 times larger area than the 32-point integer DCT unit; therefore, the savings of the transposition buffer result in the significant savings in the total silicon area.

## 5. Implementation Results

The proposed 2D architectures using 32-point 1D Architectures-1, -2, and -3 are coded in VHDL and synthesized by Synopsys Design Compiler using the TSMC 90nm CMOS library. Table 3 lists the gate count, maximum usable frequency (MUF), samples/cycle, throughput (TPT), latency, power consumption, and energy per sample (EPS) of the proposed 2D architectures and existing architectures [6,7,11–13]. Throughput per second per gate (TSG) (Ksamples/s/gate) is also listed in the last column in Table 3. Architecture-

$n$ in Table 3 means 2D architectures based on conventional transposition by a $32 \times 32$ transposition buffer and two proposed 1D architectures-$n$. The architecture-$n^{\dagger}$ means 2D architectures using four 1D architectures-$n$ and four $(N/2) \times (N/2)$ transposition buffers as proposed in Section 4. Existing designs [6,7,13] and proposed Architectures-1 and -$1^{\dagger}$ have different throughputs depending on the transform length. In Table 3, note that the TPT and TSG values of these designs were calculated based on the maximum throughput in the case of the 32-point DCT computation, and we specifically indicated $*$ with the values. Architectures-1 and 2 can have a 400 MHz MUF, whereas the MUF of Architecture-3 is found to be 380 MHz. Architecture-$2^{\dagger}$ offers the smallest EPS among the proposed and existing architectures, which have a constant throughput. The throughput of Architecture-1 varies with the DCT size; however, Architectures-2 and -3 yield 32 samples in every cycle, resulting in 12.80 and 12.16 giga samples per second (GSPS) for all different DCT sizes. The 32-point architecture-$n^{\dagger}$ uses four $16 \times 16$ transposition buffers, which involves the same area as $32 \times 32$ transposition buffers used in 32-point architecture-$n$, but has double the throughput of 64 samples per cycle and half the transposition latency. Therefore, Architectures-$2^{\dagger}$ and -$3^{\dagger}$ have 25.60 and 24.33 GSPS for all the transform sizes. The TSGs of the 2D architectures using the proposed Architecture-1 are the best for the computation of the 32-point DCT, but decrease for DCTs of lower sizes. The 2D architectures using 1D Architecture-3 offer better TSGs than those using 1D Architecture-2, but have a lower throughput. Architecture-$n^{\dagger}$ provides 1.32 times better TSG than architecture-$n$ on average for 32-point DCT computations of Architectures 1, 2, and 3. The proposed 2D architectures offer higher throughput, as well as lower latencies and, accordingly, higher TSGs than the existing 2D architectures.

**Table 3.** Comparison of different 2D integer DCT architectures.

| Design | Tech. (nm) | Gates ($K$) | MUF (MHz) | Samples/ Cycle | TPT (GSPS) | Latency | Power (mW) | EPS (mW×ns) | TSG (KSPSPG) |
|---|---|---|---|---|---|---|---|---|---|
| Zhao et al. [6] | 45 | 205 | 333 | 32 | 4.54 * | 38 | – | – | 22.09 * |
| Zhu et al. [7] | 90 | 412 | 311 | 4/8/16/32 | 9.95 * | 10/14/22/38 | 30.50 | 3.06 | 24.13 * |
| Meher et al. [12] | 90 | 463 | 310 | 32 | 9.92 | 32 | 86.27 | 8.69 | 21.41 |
| Sun et al. [11] | 90 | 155 | 312 | 4 | 1.24 | – | – | – | 8.04 |
| Park et al. [13] | 90 | 402 | 336 | 4/8/16/32 | 10.77 * | 4/8/16/32 | 69.05 | 6.40 | 26.77 * |
| Architecture-1 | 90 | 310 | 400 | 4/8/16/32 | 12.80 * | 6/10/18/34 | 39.15 | 3.05 | 41.20 * |
| Architecture-2 | 90 | 423 | 400 | 32 | 12.80 | 6/10/18/34 | 68.07 | 5.31 | 30.26 |
| Architecture-3 | 90 | 367 | 380 | 32 | 12.16 | 6/10/18/34 | 71.65 | 5.88 | 33.08 |
| Architecture-$1^{\dagger}$ | 90 | 453 | 400 | 8/16/32/64 | 25.60 * | 4/6/10/18 | 74.82 | 2.92 | 56.43 * |
| Architecture-$2^{\dagger}$ | 90 | 662 | 400 | 64 | 25.60 | 4/6/10/18 | 127.62 | 4.98 | 38.66 |
| Architecture-$3^{\dagger}$ | 90 | 561 | 380 | 64 | 24.33 | 4/6/10/18 | 138.23 | 5.68 | 43.34 |

MUF: maximum usable frequency. TPT: throughput. EPS: energy per sample. TSG: TPT per second per gate. KSPSPG: Ksamples/s/gate. For designs where the throughput varies with the transform length, the maximum throughput is shown in the TPT and TSG columns, with an * next to the value.

## 6. Summary and Conclusions

In this paper, we propose area-time efficient architectures for 2D integer DCT to be used in HEVC. Three reconfigurable architectures that support the computation of integer DCT of different sizes are proposed. We propose a novel transposition buffer and a 2D DCT architecture, which provides double the throughput and half the transposition latency without increasing the size of the transposition buffer. Implementation results show that the proposed architectures provide more throughput per unit area than the existing integer DCT architectures for HEVC. The DCT is a basic transform that can be used for the implementation of other widely used transforms such as discrete Fourier transform (DFT), discrete Hartley transform (DHT), and discrete sine transform (DST).

**Author Contributions:** Conceptualization, P.K.M., S.-K.L., T.S., and S.Y.P.; methodology, P.K.M., S.-K.L., and S.Y.P.; validation, S.Y.P. and D.H.K.; writing—original draft preparation, S.-K.L. and S.Y.P.; writing—review and editing, P.K.M., D.H.K., and S.Y.P. All authors read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1060820).

**Acknowledgments:** The EDA Tool was supported by the IC Design Education Center.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Chen, J.; Liu, S.; Deng, G.; Rahardja, S. Hardware Efficient Integer Discrete Cosine Transform for Efficient Image/Video Compression. *IEEE Access* **2019**, *7*, 152635–152645. [CrossRef]
2. Ahmed, N.; Natarajan, T.; Rao, K. Discrete cosine transform. *IEEE Trans. Comput.* **1974**, *100*, 90–93. [CrossRef]
3. JCTVC-G495, CE10: Core Transform Design for HEVC: Proposal for Current HEVC Transform. Available online: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=3752 (accessed on 4 March 2021).
4. Bross, B.; Han, W.; Ohm, J.; Sullivan, G.; Wang, Y.; Wiegand, T. High Efficiency Video Coding HEVC Text Specification Draft 10, JCTVC-L1003. Available online: http://phenix.int-evry.fr/jct/doc_end_user/documents/8_San%20Jose/wg11/JCTVC-H1003-v22.zip (accessed on 4 March 2021).
5. Singhadia, A.; Bante, P.; Chakrabarti, I. A Novel Algorithmic Approach for Efficient Realization of 2D-DCT Architecture for HEVC. *IEEE Trans. Consum. Electron.* **2019**, *65*, 264–273. [CrossRef]
6. Zhao, W.; Onoye, T.; Song, T. High-Performance Multiplierless Transform Architecture for HEVC. In Proceedings of the IEEE International Symposium on Circuits and Systems, Beijing, China, 19–23 May 2013; pp. 1668–1671.
7. Zhu, J.; Liu, Z.; Wang, D. Fully Pipelined DCT/IDCT/Hadamard Unified Transform Architecture for HEVC Codec. In Proceedings of the IEEE International Symposium on Circuits and Systems, Beijing, China, 19–23 May 2013; pp. 677–680.
8. Ahmed, A.; Shahid, M.U.; ur Rehman, A. *N*-point DCT VLSI architecture for emerging HEVC standard. *VLSI Des.* **2012**, *2012*, 752024.
9. Park, J.S.; Nam, W.J.; Han, S.M.; Lee, S. 2D large inverse transform (16 × 16, 32 × 32) for HEVC (High Efficiency Video Coding). *J. Semicond. Technol. Sci.* **2012**, *12*, 203–211. [CrossRef]
10. Budagavi, M.; Sze, V. Unified forward + inverse transform architecture for HEVC. In Proceedings of the IEEE International Conference on Image Processing, Orlando, FL, USA, 30 September–3 October 2012; pp. 209–212.
11. Sun, H.; Zhou, D.; Zhu, J.; Kimura, S.; Goto, S. An area-efficient 4/8/16/32-point inverse DCT architecture for UHDTV HEVC decoder. In Proceedings of the IEEE International Conference on Visual Communications and Image Processing, Valletta, Malta, 7–10 December 2014; pp. 197–200.
12. Meher, P.K.; Park, S.Y.; Mohanty, B.K.; Lim, K.S.; Yeo, C. Efficient Integer DCT Architectures for HEVC. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 168–178. [CrossRef]
13. Park, S.Y.; Meher, P.K. Flexible integer DCT architectures for HEVC. In Proceedings of the IEEE International Symposium on Circuits and Systems, Beijing, China, 19–23 May 2013; pp. 1376–1379.
14. Shen, S.; Shen, W.; Fan, Y.; Zeng, X. A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards. In Proceedings of the IEEE International Conference on Multimedia and Expo, Melbourne, VIC, Australia, 9–13 July 2012; pp. 788–793.
15. Software/Hardware Generation of DSP Algorithms. Available online: http://www.spiral.net/ (accessed on 4 March 2021).
16. Potkonjak, M.; Srivastava, M.B.; Chandrakasan, A.P. Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **1996**, *15*, 151–165. [CrossRef]
17. Macleod, M.D.; Dempster, A.G. Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers. *Electron. Lett.* **2004**, *40*, 651–652. [CrossRef]
18. Boullis, N.; Tisserand, A. Some optimizations of hardware multiplication by constant matrices. *IEEE Trans. Comput.* **2005**, *54*, 1271–1282. [CrossRef]