A new area and power efficient DCT circuits using sporadic logarithmic shifters

Jiajia Chen¹, Yujia Wang², Juan Zhao³, and Susanto Rahardja¹a)

Abstract Discrete Cosine Transform (DCT) is by far the most widely adopted transformation in digital image and video processing. Particularly for applications in mobile and smart devices, the required DCT needs to be realized with small circuit area and low power cost. In this paper, a new area- and power-efficient DCT architecture design is proposed. To reduce the area and power cost, temporal redundancy in matrix-vector multiplication is eliminated by time-multiplexing reconfigurable multipliers. To further reduce the complexity, a new minimization algorithm is proposed to maximize the utilization of the newly developed sporadic programmable shifters. Our experimental results on FPGA show significant circuit area reduction by at least 39.0% and power-efficiency improvement by at least 12.3% over existing advanced DCT circuits reported in the literature.

Keywords: discrete cosine transform, digital signal processing

Classification: Integrated circuits

1. Introduction

In the past decades, image and video coding standards such as JPEG [1], MPEG-2 [2], and H.263 [3] had been developed. Subsequently, newer video coding standards such as H.264/AVC and the most recent High Efficient Video Coding (HEVC) [4, 5, 6, 7] were proposed in the last decade. HEVC [8] was finalized in 2013 and is set to replace H.264/AVC [9]. With the growing subscription of mobile multimedia services which involve the H.265/HEVC standards [10], it becomes imperative to improve the hardware implementation such that it can consume less power which is essential particularly for mobile devices. One of the essential cores of these video codecs is Discrete Cosine Transform (DCT). DCT is able to offer lossy compression, efficient representation of the video data and in addition, it allows subsequent manipulation that eventually provides a highly efficient video compression [11].

Owing to these applications, several DCT designs have been proposed [12, 13, 14, 15]. These methods explore different DCT architectures to achieve lower multiplication complexity. To further reduce the computational cost and the hardware complexity, non-exact DCTs were introduced [16] including integer DCT [9, 17, 18] and DCT approximations [19, 20, 21]. In [21], the DCT is implemented through the Walsh–Hadamard transform (WHT) followed by Givens rotations. The proposed scheme derives an adaptive algorithm, which allows computing of four different approximations ranging from the complete DCT to the WHT. Besides the above exact and approximated DCTs, some recent works [22, 23] focus on the DCT circuit optimization. In [22], the design simplifies the implementation by truncating a couple of least significant bits (LSB), most significant bits (MSB), and some zero columns in the DCT coefficients. In [23], the proposed architecture reuses the hardware resources by rearranging the order of input data for different transform sizes while still exploiting the butterfly property. Despite these efforts, more efficient solutions are continuously in demand particularly in view of the limited power budget and area constraint of the mobile devices. In the hardware implementation of both exact and approximated DCTs, multiplication is the core operation which consumes significant circuit area and power. Unfortunately, the hardware savings and power efficiency can hardly be improved further with the conventional logic operators.

In this paper, a new circuit design that uses low complexity reconfigurable multipliers (RM) is proposed. The design can be used to implement exact or approximated DCTs. To minimize the complexity, new Sporadic Logarithmic Shifters (SLS) [24] are applied into the RMs to replace the conventional barrel shifters [25] and logarithmic shifters [26]. It has been shown in [24] that the complexity of SLSs drops further when its total amount of different shifts decreases. To achieve this, a new algorithm is proposed to search for the double base number system (DBNS) [27] representations for the given DCT coefficients to minimize the total number of shifts. The proposed designs are prototyped on field programmable gate array (FPGA) devices. Experimental results have shown that there is a significant circuit area saving by at least 39.0% and power-efficiency improvement by at least 12.3%, over some competing DCT circuits reported in the literature for 16-point and 32-point DCT.

2. Proposed DCT architecture with reconfigurable multipliers

Type II DCT can be represented as:

$$F(u, v) = \sqrt{\frac{N}{N}} \sum_{j=0}^{N-1} a_i \cos \left( \frac{\pi}{N} (2j + 1) \right) X_{(i+u)} X_{(j+v)}$$

(1)
The Hardware Architecture Proposed for the Implementation of the DCT Computation

The hardware architecture proposed for the implementation of the DCT computation is shown in Fig. 2. It consists of a partial sum block, a multiplexer array, SLSs and a carry save adder (CSA) at the bottom.

Table I. Computation of M-point DCT by reconfigurable multiplier

| Step | Computation |
|------|-------------|
| 1    | \(x(0) \times c(v, 0)\) the 1st column of \(C\), the \(v\)-th RM performs \(x(0) \times c(v, 0)\). |
| 2    | \(x(1) \times c(v, 1)\) the 2nd column of \(C\), the \(v\)-th RM performs \(x(1) \times c(v, 1)\). |
| ...  | ...         |
| \(M\) | \(x(M-1) \times c(v, M-1)\) the last column of \(C\). The \(v\)-th RM performs \(x(M-1) \times c(v, M-1)\). |
| \(M+1\) | The output is computed by an accumulator, i.e., \(F(v) = \sum_{i=0}^{M-1} x(i) \times c(v, i)\). |

Each RM block in Fig. 3 consists of a network of adders, SLSs and multiplexers. The \(v\)-th RM is to configure one of the DCT coefficients from \(c(v, 0)\) to \(c(v, M-1)\) at one time. In the \(j\)-th sampling clock cycle, the RM is configured to multiply the \(j\)-th coefficient \(c(v, j)\) with the input sample \(x(j)\). Several low complexity RMs have been proposed in the literature [28] and [29]. Fig. 4 shows the proposed architecture of one RM block for the computation of \(F(v)\). It consists of a partial sum block, a multiplexer array, SLSs and a carry save adder (CSA) at the bottom.
shift counts is minimized. Knowing that each fixed point value has many different DBNS representations [27], the search space for the optimal set of presentations for the DCT coefficients is huge. Therefore, an effective algorithm is needed to search for the representations which can maximize the sporadicity of SLs.

For the v-th RM configuring \( c(v, 0) \) to \( c(v, M - 1) \), let \( sd_j \) denote the number of different DBNS representations existing for the \( j \)-th DCT coefficient. \( DBNS_j \) is the set containing all these \( sd_j \) possible representation and \( DBNS_j, \omega \) is the chosen representation for the \( j \)-th DCT coefficient. Furthermore, let \( shift(DBNS_j, \omega, t) \) represent the count of shifts applied to the \( t \)-th DBNS term in \( DBNS_j, \omega \), with \( t \in [1, T] \). Let \( \text{unique} \{ \cdot \} \) denote the function which returns the number of unique integers in its argument. The total number of unique shift values required by the \( t \)-th SLS in the \( v \)-th RM, \( shift(t) \) can be expressed as

\[
    shift(t) = \text{unique} \{ shift(DBNS_j, \omega, t) \cup shift(DBNS_j, 2, t) \cup \ldots \cup shift(DBNS_j, M-1, t) \}. \tag{5}
\]

Let \( sol \) denote the DBNS representations for \( c(v, 0) \) to \( c(v, M - 1) \) with the minimized total number of distinctive shift values for all \( T \) SLSs. Thus, the problem of finding \( sol \) can be formulated as an integer linear programming problem with the following objective function:

\[
    sol = \min_{DBNS \in \text{DBNS}_v} \sum_{t=1}^{T} shift(t) \tag{6}
\]

We use the total number of shift counts required in all the SLSs as a measure of optimality for any candidate solution. This optimality is denoted as

\[
    opt = \sum_{t=1}^{T} shift(t) \tag{7}
\]

Any \( j \)-th coefficient has \( sd_j \) different DBNS representations, so the total number of different representation sets for all coefficients \( c(v, 0) \) to \( c(v, M - 1) \) is \( \prod_{j=0}^{M-1} sd_j \). This huge search space makes exhaustive search impractical. In our method, we adopted Genetic Algorithm (GA) [30] to solve (6). By introducing stochastic search [31] to escape local minima, GA converges the solution towards greater optimality uniformly from many different initial representations. GA is initialized with a population of \( K \) randomly selected solutions as the parent set. Each of these \( K \) solutions is a set of DBNS representations for all the \( M \) DCT coefficients from \( c(v, 0) \) to \( c(v, M - 1) \). Next, the representations in the parent set evolve a new generation of solutions through an iterative process in GA. Two operations, namely crossover and mutation, are performed to evolve new solutions.

The solutions in the parent set are paired to perform the crossover first. In our method, \( S \) solutions are first randomly selected from the parent set. Two solutions with the least \( opt \) in these \( S \) randomly selected solutions are paired and the remaining \( S - 2 \) solutions are returned to the parent set. Next, another \( S \) unpaired solutions in the parent set are randomly selected and two of them with the least \( opt \) are paired. This pairing continues until all the \( K \) solutions in
the parent set are paired. Let $DBNS_A$ and $DBNS_B$ be two solutions in a pair. Crossover is performed by swapping one representation for the same coefficient, i.e., the two $DBNS_j$ from $DBNS_A$ and $DBNS_B$, starting from $j = 0$. If the two $DBNS_j$ from $DBNS_A$ and $DBNS_B$ are the same, swapping for $DBNS_{j+1}$ is checked. The two new solutions generated by a crossover are added into the next generation of the parent set. Once a successful crossover between $DBNS_A$ and $DBNS_B$ is made, the remaining representations for the same coefficient will not be swapped. The crossover is performed for all paired solutions in the parent set of the current generation. Mutation is performed after crossover, where one representation of a DCT coefficient in a solution is changed to yield a new candidate solution. To reduce the search space, only one randomly selected representation is changed in one solution.

After crossover and mutation, more candidate solutions are added into the parent set. Those $K$ solutions with less opt are selected for the new parent set in the next iteration. The iterations continue until no more significant improvement can be contributed or the maximum allowed number of iterations is reached. Once sol is found, the DBNS representations for coefficients from $c(v,0)$ to $c(v,M−1)$ are decided. All distinct $3^B$ terms in the selected DBNS representations will be generated from an adder network called partial sum block. The required shift counts for each SLS in the RMs can be determined by the $2^B$ terms.

**Algorithm 1.** GA based minimization algorithm

```plaintext
GA(\{c(v,0) to c(v,M−1)\}, ε, max)
(c(v,0) to c(v,M−1)) = scale(\{c(v,0) to c(v,M−1), b\})
Initialize opt.decrement = ∞;
Initialize Num.iteration = 0;
parent_set = random_generate(c(v,0) to c(v,M−1), K);
average_opt = optimality_compute(parent_set)/K;
while (opt.decrement ≥ ε or Num.iteration < max) {
    parent_set = crossover(parent_set, S, K);
    parent_set = mutation(parent_set);
    parent_set = select(K, parent_set);
    new_opt = optimality_compute(parent_set)/K;
    opt.decrement = average_opt − new_opt;
    average_opt = new_opt;
    Num.iteration = Num.iteration + 1;
} end while; // end iterations
best_solution = select(1, parent_set);
return best_solution;
```

Algorithm 1 shows the proposed GA based minimization algorithm to find the DBNS representations. The $scale$ function scales the fractional DCT coefficients by $b$-bit followed by the rounding. The $random_generate$ function randomly generates $K$ solutions. The $optimality_compute$ function computes the optimality of every solution in the parent set according to (7). The $while$ function executes a loop until the $opt$ drops to be less than the threshold $ε$ or the max iterations have been executed. The last $select$ function picks up the best solution from the parent set. The proposed approach is summarized in the pseudo code of Algorithm 2, with GA as the sub-function.

**Algorithm 2.** Procedure for the design of RMs

```plaintext
RM(C) {
    DBNS = GA(\{c(v,0) to c(v,M−1)\}, ε, max);
    PS_set = partial_sum(DBNS); // generate partial sum set
design_MCM(PS_set); // design partial sum block
    (s) = shift.counts(DBNS); // extract shifts from DBNS
    SLS(s);
    return RM;
}
```

In Algorithm 2, the function $GA(\{c(v,0) to c(v,M−1)\}, ε, max)$ calls Algorithm 1 to select one representation for each coefficient. The partial sums (PS) represented by the $3^B$ terms in $DBNS$ are extracted using the function $partial_sum$ and stored in $PS_set$. The function $design_MCM(PS_set)$ optimizes the implementation of partial sum block by the MCM design proposed in [32]. Next, the $shift.counts(DBNS)$ function extracts the shift information. The function $SLS(s)$ is called to design the SLS units according to [24].

During each iteration, the parent set maintains $K$ solutions which can form $K/2$ pairs. In each pair, the crossover attempt starts from the two representations for the first coefficient. The worst case happens when the representations of the last coefficient from the two paired solutions are different. In this case, $M$ checks are performed before one crossover is made in one pair. This brings the worst case complexity of crossover to be $O(MK/2)$. Similarly, one mutation is performed for each of the $K$ solutions. Therefore, the complexity of mutation for $K$ candidate solutions is $O(K)$. The overall complexity of crossover and mutation becomes $O(MK/2 + K)$ for one iteration. From our experimental results of algorithmic convergence, the number of iterations, $Num.iteration$, is well bounded and independent of $M$. Hence, the overall complexity can be approximated by the complexity of single iteration, $O(MK/2 + K)$. In addition, it is reasonable to assume $K$ is proportional to $M$. The complexity to design one RM in the M-point DCT can be simplified into $O(M^2)$ which is quadratic. Given that there is a total of $M$ parallel RMs, the overall complexity to design the RM based M-point DCT is therefore $O(M^3)$. In practical applications, the DCT length $M$ rarely exceeds 128, and therefore this cubic complexity is acceptable. The computation efficiency is tested by running the proposed algorithm on 32-point DCT. It takes only 6.7 s on a PC equipped with Intel i7-4500CPU running at 1.8 GHz and 16 GB RAM.

4. Experimental results and discussions

4.1 Maximize the sporadicity of the SLSs

In this sub-section, we adopt the minimization algorithm proposed in Section 3 to minimize the total number of shift counts needed in the SLSs for 16-point and 32-point DCT. Input signal is assumed to be 8-bit. In these designs, the parameters in the GA function are set as $K = 50$, $S = 10$, $ε = 5$ and $max = 10$. After running our minimization algorithm, the total number of shift counts needed for the SLSs designed in the RMs for 16-point and 32-point DCT are presented in Table II. These results are compared with the number of shifts needed by full range programmable
shifters (FRPS) [25]. Average sporadicity is computed by averaging the sporadicities of all SLSs used in the design.

From Table II, it is clear that the proposed minimization algorithm has successfully represented the DCT coefficients in the appropriate DBNS where the total number of shift counts is minimized. Compared with the required shift counts in the conventional FRPS, the average sporadicity is increased to above 70%.

### 4.2 Experimental results of the DCT designs on FPGA

In this section, we adopt our proposed design described in Algorithm 2 to develop the hardware architecture of 16-point DCT. The procedure is repeated for 32-point DCT. These 16-point and 32-point RM based DCT designs are mapped onto FPGA. The area, delay and power results of these architectures are evaluated and presented in Table III. These results are compared with three relevant methods, [6], [9] and [14]. 8-bit wordlength is assumed for the input sample. All the designs are described in Verilog code and implemented onto Xilinx Spartan VI XC6LX75 (package FGG676) FPGA device. The supply voltage is set at 1.2 V. The power consumptions by the DCT architectures on this device are computed using Xilinx XPower Analyzer.

| designs       | Minimized total shift counts | Total shift counts for FRPS | sporadicity |
|---------------|------------------------------|-----------------------------|-------------|
| 16-point      | 59                           | 248                         | 73.79%      |
| 32-point      | 181                          | 648                         | 72.07%      |

**Table III.** Comparison of total shift counts and sporadicity of the SLSs designs

From the FPGA mapping results in Table III, the areas of the proposed designs are 67.2%, 49.7% and 80.1% lower than the areas of [6], [9] and [14], respectively, for 16-point DCT. In addition, for 32-point DCT the reductions are 83.3%, 39.0% and 77.4%, respectively. Similarly, its total power consumption over [6], [9] and [14] are also reduced by 61.3%, 47.0% and 78.5% respectively, for 16-point DCT, and for the 32-point DCT, the reductions are 21.5%, 12.3% and 69.2% respectively. These double-digit percentage savings in area and power consumption testify that the proposed RMs and the SLSs are significantly smaller and more power-efficient than the conventional logic operator network adopted such as the recursive kernel of [14] and multiplier block units in [9]. Because the RM and SLS both fall in the critical path of our design, the delay of our proposed design turns out to be longer than those of [6] in FPGA implementation for 24.9% and 22.3% respectively for 16-point and 32-point. Although RM could cause longer delay compared with the traditional multipliers, the benefits in area and power reductions are more significant.

### 5. Conclusions

This paper presents a novel approach to design RM based DCT architectures. The DCT coefficients are represented in the non-canonical minimum double base number representations, where the shift information of the partial sums can be explicitly accessed. The algorithm selects one DBNS representation for each DCT coefficient, such that the total number of distinct shift counts in every RM is minimized. This leads to an efficient realization of RMs with the high sporadicity SLSs, which are much simpler than conventional shifters. Significant area reduction and power savings have been observed in our experimental results for the implementation of the proposed 16-point and 32-point DCT designs on FPGA platforms.

**Acknowledgment**

This work is supported by Nanjing Selected Project Grant MCA19005.

### References

[1] G. K. Wallace: “The JPEG still picture compression standard,” IEEE Trans. Consum. Electron. 38 (1992) xviii (DOI: 10.1109/30.125072).

[2] M. Bosi, et al.: ISO/IEC MPEG-2 Advanced Audio Coding, document (1992).

[3] K. Rijjske: “H.263: Video coding for low-bit-rate communication,” IEEE Commun. Mag. 34 (1996) 42 (DOI: 10.1109/35.556485).

[4] C.-K. Fong, et al.: “Recursive integer cosine transform for HEVC and future video coding standards,” IEEE Trans. Circuits Syst. Video Technol. 27 (2017) 326 (DOI: 10.1109/TCSVT.2015.2513664).

[5] G. J. Sullivan, et al.: “Overview of the high efficiency video coding (HEVC) standard,” IEEE Trans. Circuits Syst. Video Technol. 22 (2012) 1649 (DOI: 10.1109/TCSVT.2012.2221919).

[6] M. Budagavi, et al.: “Core transform design in the high efficiency video coding (HEVC) standard,” IEEE J. Sel. Topics Signal Process. 7 (2013) 1029 (DOI: 10.1109/JSTSP.2013.2270429).

[7] G. J. Sullivan, et al.: “Standardized extensions of high efficiency video coding,” IEEE J. Sel. Topics Signal Process. 7 (2013) 1001 (DOI: 10.1109/JSTSP.2013.2283657).

[8] B. Bross, et al.: High Efficiency Video Coding (HEVC) Text Specification Draft 10 (for FDIS and Consent), JCT-VC L1003 (2013).

[9] P. K. Meher, et al.: “Efficient integer DCT architectures for HEVC,” IEEE Trans. Circuits Syst. Video Technol. 24 (2014) 168 (DOI: 10.1109/TCSVT.2013.2276662).

[10] H. Liang, et al.: “Area-efficient HEVC IDCT/IDST architecture for 8K×4K video decoding,” IEICE Electron. Express 13 (2016) 20160019 (DOI: 10.1587/exle.13.20160019).

[11] R. Rajapaksha, et al.: “VLSI computational architectures for the arithmetic cosine transform,” IEEE Trans. Comput. 64 (2015) 2708 (DOI: 10.1109/TC.2014.2366732).

[12] M. T. Heideman: Multiplicative Complexity, Convolution, and the DFT (Springer-Verlag, New York, NY, USA, 1988).

[13] C. Loeffler, et al.: “Practical fast 1-D DCT algorithms with 11 multiplications,” ICASSP (1989) 988 (DOI: 10.1109/ICASSP.1989.266596).

[14] S. An: “Recursive algorithm, architectures and FPGA implementation of the two-dimensional discrete cosine transform,” IET
[15] Y. Shen and H. Oh: “Pipelined implementation of AI-based Loeffler DCT,” IEICE Electron. Express 10 (2013) 20130319 (DOI: 10.1587/exle.10.20130319).

[16] V. de A. Coutinho, et al.: “Low-complexity multidimensional DCT approximations for high-order tensor data decorrelation,” IEEE Trans. Image Process. 26 (2017) 2296 (DOI: 10.1109/TIP.2017.2679442).

[17] R. J. Cintra: “An integer approximation method for discrete sinusoidal transforms,” Circuits Syst. Signal Process. 30 (2011) 1481 (DOI: 10.1007/s00034-011-9318-5).

[18] R. J. Cintra, et al.: “Low-complexity 8-point DCT approximations based on integer functions,” Signal Process. 99 (2014) 201 (DOI: 10.1016/j.sigpro.2013.12.027).

[19] R. J. Cintra and F. M. Bayer: “A DCT approximation for image compression,” IEEE Signal Process. Lett. 18 (2011) 579 (DOI: 10.1109/LSP.2011.2163394).

[20] F. M. Bayer and R. J. Cintra: “DCT-like transform for image compression requires 14 additions only,” Electron. Lett. 48 (2012) 919 (DOI: 10.1049/el.2012.1148).

[21] M. Masera, et al.: “Adaptive approximated DCT architectures for HEVC,” IEEE Trans. Circuits Syst. Video Technol. 27 (2017) 2714 (DOI: 10.1109/TCSVT.2016.2595320).

[22] H. Sun, et al.: “Approximate DCT design for video encoding based on novel truncation scheme,” IEEE Trans. Circuits Syst. I, Reg. Papers 66 (2019) 1517 (DOI: 10.1109/TCSI.2018.2882474).

[23] M. Zheng, et al.: “A reconfigurable architecture for discrete cosine transform in video coding,” IEEE Trans. Circuits Syst. Video Technol. Early Access (2019) 1 (DOI: 10.1109/TCSVT.2019.2896294).

[24] J. Chen, et al.: “New hardware and power efficient sporadic logarithmic shifters for DSP applications,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 37 (2018) 896 (DOI: 10.1109/TCAD.2017.2740300).

[25] R. Rajalakshmi and P. A. Priya: “Design and analysis of a 4-bit low power universal barrel-shifter in 16 nm FinFET technology,” IEEE ICACCCCT (2014) 527 (DOI: 10.1109/ICACCCCT.2014.7019141).

[26] J. Y. S. Low and C.-H. Chang: “A VLSI efficient programmable power-of-two scaler for $2^n - 1$, $2^n$, $2^n + 1$ RNS,” IEEE Trans. Circuits Syst. I, Reg. Papers 59 (2012) 2911 (DOI: 10.1109/TCSI.2012.2206491).

[31] K. Deep, et al.: “A real coded genetic algorithm for solving integer and mixed integer optimization problems,” Appl. Math. Comput. 212 (2009) 505 (DOI: 10.1016/j.amc.2009.02.044).

[32] J. Ding, et al.: “A new paradigm of common subexpression elimination by unification of addition and subtraction,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 35 (2016) 1605 (DOI: 10.1109/TCAD.2016.2527700).