IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 4 July 2022, accepted 17 July 2022, date of publication 27 July 2022, date of current version 3 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3194261

RESEARCH ARTICLE

Hardware Accelerator Design of DCT Algorithm With Unique-Group Cosine Coefficients for Mel-Scale Frequency Cepstral Coefficients

SHIN-CHI LAI[1,2], (Member, IEEE), YING-HSIU HUNG[3], YI-CHANG ZHU[4], SZU-TING WANG[4], QI-XIAN HUANG[5], MING-HWA SHEU[3], AND WEN-HO JUANG[6]

[1]Department of Automation Engineering, National Formosa University, Huwei 632301, Taiwan
[2]Smart Machinery and Intelligent Manufacturing Research Center, National Formosa University, Huwei 632301, Taiwan
[3]Department of Electronics Engineering, National Yunlin University of Science and Technology, Douliu 64002, Taiwan
[4]Doctor's Program of Smart Industry Technology Research and Design, National Formosa University, Huwei 632301, Taiwan
[5]Department of Electrical Engineering, National Tsing Hua University, Hsinchu 300044, Taiwan
[6]Department of Computer Science and Information Engineering, National Formosa University, Huwei 632301, Taiwan

Corresponding author: Wen-Ho Juang (riverjuang@nfu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 110-2622-E-224 -006, Grant 110-2222-E-150 -001, Grant 110-2221-E-150 -045, and Grant 109-2221-E-150 -043; in part by the Smart Machinery and Intelligent Manufacturing Research Center; in part by the Higher Education Sprout Project, National Formosa University, Yunlin, Taiwan; and in part by the Ministry of Education (MOE) Female Researching Talent Cultivation Project for Science, Technology, Engineering, and Mathematics (STEM) field.

ABSTRACT This study presents a compact L-points discrete cosine transform (DCT) hardware accelerator for M-points Mel-scale Frequency Cepstral Coefficients (MFCC). The main contributions of this work can be summarized as 1) proposing an algorithm with lower complexity; 2) achieving higher accuracy performance; 3) implementing a low-cost accelerator with a unique group of cosine coefficients. For algorithm derivation, the proposed method converts the original formula into the type IV of discrete cosine transform (DCT-IV) with a preprocessing procedure. The kernel computation of DCT-IV can be further derived into the same cosine multiplication with the proposed preprocessing. Therefore, a total of $(M\text{-}1)$ $(L\text{-}1)$ additions, $(M\text{-}1)$ L multiplications, and L coefficients are required for the computation. Compared with Jo *et al.*'s algorithm, the proposed method respectively reduces the number of additions and multiplications by 42.32 % and 41.67 %. Instead, the number of coefficients is increased by 33.33 %. Moreover, the proposed algorithm exhibits a higher peak signal-to-noise ratio (PSNR) value which is achieved at 90.1dB with a 16-bit coefficient word length. For hardware realization, the FPGA implementation results show that it can operate at a clock rate of 135.85 MHz and requires only 113 combinational elements, 87 registers, 3 DSP multipliers, 64×16 bits RAM and 32×16 bits ROM. Overall, it would be a good choice for integrating MFCC applications in the future.

INDEX TERMS Discrete cosine transform (DCT), hardware accelerator, Mel-scale frequency cepstral coefficients (MFCC).

I. INTRODUCTION

Recently, Mel-scale frequency cepstral coefficients (MFCC) [1], [2] have been used to generate the feature vectors of sounds in speech recognition by combining them with convolutional neural network (CNN) and deep neural network (DNN) models [3], [4]. Moreover, the MFCC-based deep learning recognition algorithm has been widely used in heart sound recognition [4], semantic analysis [3], emotion analysis [5], and keyword detection [6]–[8]. Both fast Fourier transform (FFT) and discrete cosine transform (DCT) are very computationally intensive in MFCC processing. There are some well-known fast algorithms, such as radix-2 and mixed-radix FFT which have been developed and used for MFCC [9]. However, the DCT computation in MFCC has not received too much attention.

The associate editor coordinating the review of this manuscript and approving it for publication was Farid Boussaid.

Moreover, some problems with sliding and hopping DFTs were developed in [10]–[12]. For DCT computation, most of the studies such as [13], [14] focused on the derivation of modified DCT (MDCT) and modified discrete sine transform (MDST) by converting the formula to the type IV of DCT in the applications of audio codec. However, not too much attention has been paid to the definition of DCT in MFCC computation.

Recently, most approaches [9], [15]–[19] compute the DCT coefficients by using the direct mapping method. This means that storing the cosine coefficient in memory requires a set ROM whose size is $M \times L \times$ word length. Moreover, a fully parallelized DCT design [16] also requires a large number of hardware resources. To reduce the memory usage and hardware cost, Jo *et al.*'s method developed an approximate calculation in (16), [15] for sine coefficients and used one multiplier–accumulator (MAC) in kernel processing unit. Compared with conventional designs, reduction of memory usage and hardware cost can be improved at the expense of accuracy and complexity on DCT computation. In this work, an improved DCT algorithm extended from [20] is proposed to reduce the cost of memory usage and increase the accuracy of computation. A unique group of cosine coefficients is derived and mapped into a small size memory with a memory address generator. It is worth noting that there is no extra approximate calculation cost compared with Jo *et al.* [15]. The proposed method converts the original DCT formula into DCT-IV using a preprocessing procedure. The kernel computation of DCT-IV is further derived in the same cosine multiplication with the preprocessing in the proposed method. Overall, only $(M\text{-}1) \times (L\text{-}1)$ additions, $(M\text{-}1) \times L$ multiplications, and L coefficients are needed for the computation.

The rest of this paper is organized as follows: Section II gives an overview of the previous work. In Section III, presents the proposed algorithm in detail, and designs the compact architecture with memory address and sign exchange generator. Section IV presents the comparison results between previous works and the proposed method. Finally, in Section V the conclusions are presented.

II. PREVIOUS WORK AND ITS REALIZATION

The definition of the DCT conversion in MFCC is given in (1), where M and L are set as 13 and 32, respectively, in [10]. After substituting $n = l\text{-}1$ into (1), the equation can be written as (2), where n ranges from 0 to L-1.

$$\mathrm{C}[m] = \sum_{l=1}^{L} \mathrm{X}[l] \cos\left(\frac{m(l-0.5)\pi}{L}\right),\quad m = 1, \ldots, M-1. \tag{1}$$

$$\mathrm{C}[m] = \sum_{n=0}^{L-1} \mathrm{X}[n+1] \cos\left(\frac{2m(2n+1)\pi}{4L}\right),\quad m = 1, \ldots, M-1. \tag{2}$$

A. DIRECT COMPUTATION IN IMPLEMENTATION

As shown in (2), the input sequence $\mathrm{X}[n+1]$ is stored in a RAM, and the cosine function is generally realized in a ROM. After performing multiplication and summation, the coefficients of $\mathrm{C}[m]$ is calculated by different m values. In the implementation, the numbers of m and n must be sequentially counted. For the cosine function, two indices m and n directly change the cosine coefficients and then a total of $(M\text{-}1) \times L$ cosine coefficients are needed to store in ROM. To obtain the correct cosine coefficients, an address generator is developed according to the proposed algorithm which is presented in Section II (B).

B. TRIGONOMETRIC APPROXIMATION METHOD

A unified lookup table (LUT) using the trigonometric approximation method as shown in (3) is adopted in the implementation of Jo *et al.* [15] by (4), where $0 \leq \text{addr} < 2\text{L}$, $\text{L} = 32$, $\alpha = \text{addr AND } 110000_{(2)}$, $\beta = \text{addr AND } 001100_{(2)}$, and $\gamma = \text{addr AND } 000011_{(2)}$.

$$\sin(\mathrm{A}+\mathrm{B}+\mathrm{C}) \approx \sin(\mathrm{A}+\mathrm{B}) + \cos(\mathrm{A})\sin(\mathrm{C}). \tag{3}$$

$$\sin\left(\frac{\pi}{2}\cdot\frac{addr}{2L}\right) = \sin\left(\frac{\pi}{2}\cdot\frac{\alpha+\beta+\gamma}{2L}\right) \approx \sin\left(\frac{\pi}{2}\cdot\frac{\alpha+\beta}{2L}\right) + \cos\left(\frac{\pi}{2}\cdot\frac{\alpha}{2L}\right) \times \sin\left(\frac{\pi}{2}\cdot\frac{\gamma}{2L}\right). \tag{4}$$

Let the L value be a power-of-two in the trigonometric function, and you get the best of ROM sizes. However, the trigonometric approximation must pay the price of lower accuracy and added computational complexity. To convert the cosine function into a sine function, (2) can be further derived as:

$$\mathrm{C}[m] = \sum_{n=0}^{L-1} \mathrm{X}[n+1] \sin\left(\frac{\pi}{2}\cdot\frac{(4mn+2m+2L)}{2L}\right). \tag{5}$$

Unlike the direct computation, the value generated by the address generator ranges from 0 to $2L$-1, but the value of $(4mn+2m+2L)$ would have a larger dynamic range in computation. Thus, additional operations are needed to convert the angle between 0 and $\pi/2$. Table 1 shows the pseudo algorithm for generating the essential memory address, where ***addr*** is the memory address, **SIN_origin** and **SIN_approx** are the non-approximation and approximation of sine functions, respectively, as defined in (4). Based on (4), (5), and Table 1, it can be easily observed that the memory size depends on the computation of $(2^4 + 2^2 + 2^2)$, and 24 trigonometric coefficients are stored in three different ROMs. Compared to the conventional method, this method can realize the sine and cosine functions, and reduces the memory size by 62.5 %, *i.e.*, 40 coefficients in the implementation. The advantage lies in the use of the coefficient memory and not in the computational complexity. In terms of computational complexity, this method requires a total of takes $2(M\text{-}1) \times L$ multiplications and $(M\text{-}1) \times (2L\text{-}1)$ additions excluding the

TABLE 1. Proposed pseudo code for memory address and sine coefficient generators based on formula (5).

```
addr_init = mod(4mn+2m+2L, 8L);
if (addr_init > 4L)
    addr_tmp = addr_init – 4L;
    Sign_exch = -1;
else
    addr_tmp = addr_init;
    Sign_exch = 1;
end
if (addr_tmp > 2L)
    addr = 4L – addr_tmp;
else
    addr = addr_tmp;
end
SIN_origin = Sign_exch · sin(addr·π/2);
α = addr AND 110000(2);
β = addr AND 001100(2);
γ = addr AND 000011(2);
tmp0 = sin((α+β)/(2L)·(π/2));
tmp1 = cos(α/(2L)·(π/2));
tmp2 = sin(γ/(2L)·(π/2));
SIN_approx = Sign_exch · (tmp0 + tmp1·tmp2);
```

memory address generator operation. Hence, two issues are required to address 1) proposing a novel algorithm with lower complexity; 2) implementing a low-cost and high-accuracy accelerator with a unique group of cosine coefficients.

III. PROPOSED DCT ALGORITHM AND ARCHITECTURE DESIGN

Since a power-of-two (L value) is used in the DCT computation, a unique-group of cosine coefficients leads to a smaller memory requirement, which is a better solution than the trigonometric approximation method. Moreover, it can be mapped into a low-cost and compact structure for hardware implementation.

A. ALGORITHM DERIVATION

Applying the sum and difference identity into (2), we obtain the following:

$$\begin{aligned} C[m] = &\sum_{n=0}^{L-1} X[n+1]\cos\left(\frac{(2m+1)(2n+1)\pi}{4L}\right) \\ &\times \cos\left(\frac{(2n+1)\pi}{4L}\right) \\ &+\sum_{n=0}^{L-1} X[n+1]\sin\left(\frac{(2m+1)(2n+1)\pi}{4L}\right) \\ &\times \sin\left(\frac{(2n+1)\pi}{4L}\right). \end{aligned} \tag{6}$$

The sine part of (6) can be derived into (7), and the original DCT computation is further converted from (2) to the type-IV of the DCT kernel, as shown in (8), where the preprocessing method of $X_1^m[n]$ is defined as (9).

$$\begin{aligned} &\sum_{n=0}^{L-1} X[n+1]\sin\left(\frac{(2m+1)(2n+1)\pi}{4L}\right) \\ &\quad\times \sin\left(\frac{(2n+1)\pi}{4L}\right) \\ &\quad= \sum_{n=0}^{L-1} X[L-n]\times(-1)^m\cos\left(\frac{(2m+1)(2n+1)\pi}{4L}\right) \\ &\quad\times \cos\left(\frac{(2n+1)\pi}{4L}\right) \end{aligned} \tag{7}$$

$$C[m] = \sum_{n=0}^{L-1} X_1^m[n]\cos\left(\frac{(2m+1)(2n+1)\pi}{4L}\right) \tag{8}$$

$$X_1^m[n] = \left(X[n+1]+(-1)^m X[L-n]\right)\cos\left(\frac{\pi}{2}\cdot\frac{2n+1}{2L}\right) \tag{9}$$

It can be observed that the values of the cosine function, *i.e.*, $(2n+1)$ in the numerator are all odd numbers and they are also co-prime with $(2L)$ in the denominator. This result means that only L memory sizes are required for the implementation. Looking at the angle of the cosine function in (8) and (9), the main difference is the multiplication of $(2m+1)$ which is also an odd value and a co-prime value to $(2L)$. For this reason, these cosine coefficients would have many repetitions rotated around a circle. Therefore, (8) can be rewritten into (10)–(13) with the same cosine function by changing and reordering the index n into n^, where the symbol $S0$ is applied to determine the sign exchange operation of the cosine function. If the angle is greater than $\pi/2$, it is converted to a value less than $\pi/2$ according to the symmetry of the trigonometric identity.

$$C[m] = \sum_{\hat{n}=0}^{L-1} X_1^m[\hat{n}]\times(-1)^{S0}\times\cos\left(\frac{\pi}{2}\cdot\frac{2\hat{n}+1}{2L}\right) \tag{10}$$

$$addr_{mn} = \mathrm{mod}\left((2mn+m+n), 4L\right) \tag{11}$$

$$\hat{n} \equiv \begin{cases} addr_{mn}, & 0 \le addr_{mn} < L \\ 2L-1-addr_{mn}, & L \le addr_{mn} < 2L \\ addr_{mn}-2L, & 2L \le addr_{mn} < 3L \\ 4L-1-addr_{mn}, & 3L \le addr_{mn} < 4L \end{cases} \tag{12}$$

$$S0 \equiv \begin{cases} 0, & 0 \le addr_{mn} < L \\ 1, & L \le addr_{mn} < 2L \\ 1, & 2L \le addr_{mn} < 3L \\ 0, & 3L \le addr_{mn} < 4L \end{cases} \tag{13}$$

Based on the above derivations, the proposed algorithm has a very simple computational kernel with the same multiplication of cosine function in (9) and (10). The number of the cosine coefficients can be completely shared with the preprocessing procedure and the computational kernel. For preprocessing, $2L$ additions, $2L$ multiplications, and L coefficients are required. In the kernel design, there are $(M\text{-}1)\times(L\text{-}1)$ additions and $(M\text{-}1)\times L$ multiplications are required for computation. Figure 1 shows the flowchart of the

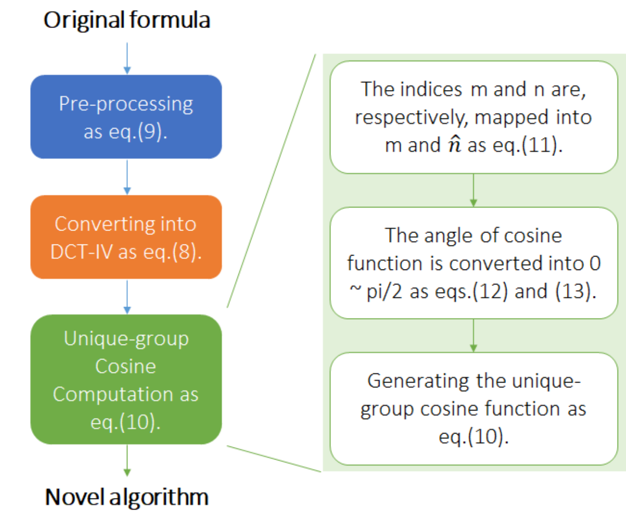

FIGURE 1. Flowchart of the proposed algorithm.

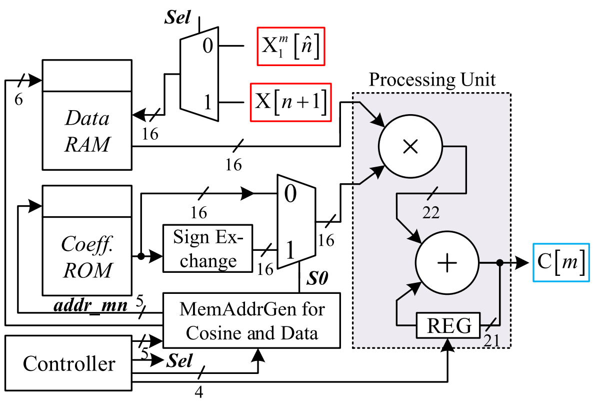

FIGURE 2. Proposed compact hardware accelerator design.

proposed novel algorithm. Table 2 shows the corresponding pseudocode with a unique group of cosine coefficients and the function of the memory address generator.

B. COMPACT ARCHITECTURE DESIGN

The definitions of (9)–(13) can be directly mapped into a hardware structure in terms of a controller, a memory address generator, and a processing unit (PU) as shown in Figure 2. For the pre-processing in (9), two-input data is first fed into the data memory. Then, the data is extracted to PU to perform the operations of sign exchange, addition, and multiplication. Next, the controller assigns the address generator to access the data memory (RAM) and the sine coefficient memory (ROM). Results are received from the accumulator after PU completes all computations. Finally, all DCT coefficients, *i.e.*, C[*m*] are well calculated and written back to the data memory.

C. MEMORY ADDRESS AND SIGN EXCHANGE GENERATOR DESIGN

In the proposed design, the memory address generator should follow two different counting methods for the data memory

TABLE 2. Pseudo code for the proposed algorithm with memory address generator.

```
#01  M = 13; L = 32;
#02  for n = 0 : L – 1
#03    FixedCOS (n) = cos((2n+1)π/(4L));
#04  end
#05  for m = 1 : M – 1
#06    Acc = 0;
#07    for n = 0 : L – 1
#08      [addr_mn, S0] = MemAddrGen(m, n, L);
#09      X1(n) = (X(n) + X(L–1–n)*(–1)^m) * FixedCOS (n);
#10      C(m) = X1(n)*( –1)^S0*FixedCOS(addr_mn) + Acc;
#11      Acc = C(m);
#12    end
#13  end
#14
#15  Function [ addr_mn, S0 ] = MemAddrGen(m, n, L)
#16  addr_init = mod(2mn+m+n, 4L);
#17  if (addr_init < L)
#18    addr_mn = addr_init;
#19    S0 = 0;
#20  else if (addr_init < 2L)
#21    addr_mn = 2L – 1 – addr_init;
#22    S0 = 1;
#23  else if (addr_init < 3L)
#24    addr_mn = addr_init – 2L;
#25    S0 = 1;
#26  else
#27    addr_mn = 4L – 1 – addr_init;
#28    S0 = 0;
#29  end
```

and the coefficient memory, as shown in Figure 2. The first one is to provide a sequential count for accessing the data memory. For the coefficient memory, the other address generator follows the proposed derivation as shown in (11) and (12) as well as (#15) to (#29) in Table 2.

In the hardware implementation of the modulo operation, a 9-bit result for calculating $(2mn + m + n)$ would be generated first, and then this temporary result would be stored from bit [6] to bit[0]. Finally, this temporary 5-bit memory address would follow the four conditions in (12) to decide whether to execute the NOT operation or keep the current situation. As mentioned earlier, the proposed memory address generator is clearly a low-cost design. Similar to the memory address generator design, the sign exchange operation also follows the same rule in (13) so it can be directly mapped to one XOR operation to produce a 1-bit *S0* value listed as #19, #22, #25, and #28 in Table 2. Figure 3 shows that the proposed memory address generator is designed for the cosine function.

IV. COMPARISION AND DISCUSSION

In this section, the performance metrics in terms of complexity, accuracy, and resource cost are used to evaluate the difference between the proposed algorithm and previous works. The key parameters of *M* and *L* are set here to 13 and 32, respectively. The number of additions, multiplications,

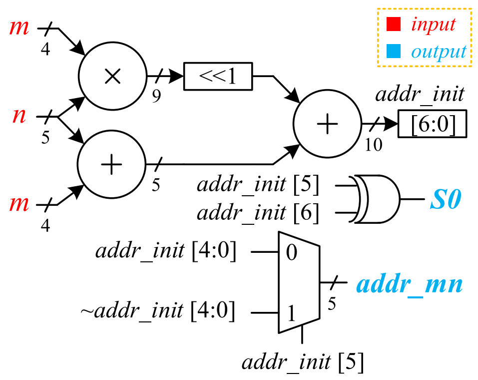

FIGURE 3. Proposed memory address generator for cosine function.

TABLE 3. (A) Complexity analysis of preprocessing computation between previous and proposed designs. (B). Complexity analysis of kernel computation between previous and proposed designs. (C). Complexity analysis of all computations between previous and proposed designs.

(a)

Method	Preprocessing Computation		
	Add	Mup	Coeff
Direct [18, 19]	0	0	0
[15]	0	0	0
[20]	2*L*	2*L*	*L*
Proposed	2*L*	2*L*	*L*

(b)

Method	Add	Mup	Coeff
Direct [18, 19]	(*M*-1)(*L*-1)	(*M*-1)*L*	(*M*-1)*L*
[15]	(*M*-1)(2*L*-1)	(*M*-1)2*L*	24
[20]	(*M*-1)(2*L*-1)	(*M*-1)2*L*	24
Proposed	(*M*-1)(*L*-1)	(*M*-1)*L*	0

(c)

Method	Pre-processing			Kernel		
	Add	Mup	Coeff	Add	Mup	Coeff
Direct [18, 19]	0	0	0	372	384	384
[15]	0	0	0	756	768	24
[20]	64	64	32	756	768	24
Proposed	64	64	32	372	384	0

TABLE 4. PSNR analysis with different word lengths of coefficients for various algorithms.

WL	10	12	14	16	18	20
Direct [18, 19]	68.4	78.9	86.9	89.8	90.6	90.7
[15]	57.6	56.9	56.6	56.6	56.6	56.6
[20]	54.2	53.3	52.9	52.8	52.8	52.8
Proposed	69.7	79.3	87.6	90.1	90.6	90.8

and coefficients are carefully considered when calculating the complexity. The peak signal-to-noise ratio (PSNR) is used to estimate the impact on accuracy for different word lengths (WL).

TABLE 5. Word length selection for proposed hardware accelerator design.

Node	WL	Sign Bit	Integer Bit	Fraction Bit
$X[l]$	16	1	1	14
$X_1[\hat{n}]$	16	1	2	13
COS	16	0	0	16
R	21	1	4	16
$C[m]$	16	1	4	11

TABLE 6. PSNR analysis for various algorithms.

WL of Coeffs	10 bit	12 bit	14 bit	16 bit	18 bit	20 bit
Ideal*	68.4	78.9	86.9	89.8	90.6	90.7
Direct [18, 19]	62.0	64.6	65.3	65.5	65.5	65.5
[15]	57.4	57.7	57.5	57.5	57.5	57.5
[20]	54.9	54.1	53.8	53.7	53.6	53.6
Proposed	60.9	62.8	63.5	63.7	63.7	63.7

TABLE 7. Hardware cost analysis for various DCT accelerator kernel design.

Method	Direct	2016 [15]	2019 [16]	2021 [20]	Proposed
Register	1	1	15×(2+6)	1	1
Adder	1	2	15×1	1	1
Multiplier	1	2	15×1	1	1
ROM	6144 bits	384 bits	360 words	896 bits	512 bits
AddrGen	Easiest	Hard	Easiest	Hard	Easy
DTPT	1	1	15	1	1

A. COMPUTATIONAL COMPLEXITY

Table 3 lists the analytical results for different algorithms. Here, "Add", "Mup", and "Coeff" are presented as addition operation, multiplication operation, and trigonometric coefficient, respectively. Note that both the direct method and [16]–[19] follow the definition derived in (2). Table 3 (a) summarizes that no operations are required in the preprocessing for the direct method and the Jo *et al.* [15] method. However, in the method of Lai *et al.* [20] and the proposed method, 64 additions and 64 multiplications are required. In addition, 32 cosine coefficients are required for conversion to a DCT-IV kernel computation in Table 3 (b). For the kernel part in Table 3 (c), the direct method requires 372 additions, 384 multiplications and 384 coefficients, respectively.

To reduce the number of coefficients used, the trigonometric approximation method of Jo *et al.* [15] requires only 24 sine coefficients but increases the numbers of addition and multiplication by 103.22 % and 100 %, respectively. The proposed algorithm requires the less number of additions and multiplications compared with Jo *et al.* [15]. By using unique-group cosine coefficients, no additional cosine coefficients are required in the kernel design. Overall, the proposed

TABLE 8. FPGA implementation results of the proposed DCT accelerator.

Hardware Resource	Hardware Block (Altera FPGA Device: Cyclone IV E EP4CE22F17C6)						
	Controller	Address Generator	Memory	Processing unit		Others	Total
				Sign Exchange	Computation		
Combinational	22	42	0	18	21	1	104
Registers	23	7	0	33	22	2	87
DSP Elements 9-bits Multipliers	0	0	0	0	2	0	2
DSP 18x18 Simple Multipliers	0	0	0	0	1	0	1
RAM Size(bits)	0	0	64x16	0	0	0	64x16
ROM Size(bits)	0	0	32x16	0	0	0	32x16
Latency Per Transformation	**64** clocks for data input + **3** clocks for control setting + **32**×**13** clocks for DCT kernel computation = **483** clocks. Latency = 483 clocks / 135.85MHz						3.56 µs
Power Consumption (PC) Per Transformation	0.05mW (Static PC @ 10MHz) + 0.73mW(Dynamic PC @ 10MHz)						0.78mW
	0.05mW (Static PC @ 100MHz) + 7.12mW(Dynamic PC @ 100MHz)						7.17mW

method reduces the number of additions and multiplications by 42.32 % and 41.67 %, respectively, although it increases the number of coefficients by 33.33 %, compared with the algorithm of Jo *et al.* [15].

B. COMPUTATIONAL ACCURACY

For accuracy analysis, 10^5 times DCT operations are required in different algorithms. The input, output, and cosine coefficient are set to 16-bit, 16-bit, and variable word length, respectively. The other computational nodes are all simulated by floating-point format in MATLAB. Table 4 shows the PSNR comparison results, and makes it clear that the proposed algorithm has outstanding performance as well as a direct method. The algorithm of Jo *et al.* [15] applies approximate computation so the PSNR values with different WL of coefficients are approximately 56 dB which are lower than those of the proposed method.

In this paper, a dynamic range analysis is performed for all computational nodes in the proposed algorithm. Then, the WL of input and output are both set to 16 bits. By averaging 10^6 random input sets, the WL information of all computational nodes can be obtained as listed in Table 5. To be specific, the WL of the register (R) is set to 21-bit to maintain the accuracy of the accumulator.

Following the results of Table 5, Table 6 shows the additional analysis by averaging 10^4 random input sets with truncation for each computational node. Here, the ideal computation (Ideal) is excluded without truncation for all internal nodes. Unlike all the truncation nodes of the different algorithms, ''Ideal'' uses 16-bit word lengths for input and output but retains floating computation for the other computational nodes. Changing different WL of coefficients (WL of Coeffs), it shows that the proposed algorithm has better performance than [15] and [20]. Furthermore, it is suggested to set the WL of coefficients of the proposed algorithm to 16-bit, because the results show that the PSNR values become saturated. If a higher PSNR value is needed for the proposed algorithm, the WL of the fractional part of the accumulated register (R) can be adjusted from 16-bit to 20-bit in the implementation.

C. HARDWARE COST AND RESOURCE USAGE

Table 7 lists the hardware cost results of various algorithms. The DCT design by Abed *et al.* [17], provides only a rough hardware block with multiplier, adder, and address logic. For fast parallel computation, MFCC computation implements 15 DCT blocks ($M = 24$, $L = 16$) as shown in Figure 4, [16] by Boujelben and Bahoura. Each DCT block includes a ROM, a MAC unit, and an output register for calculating 15 MFCC coefficients. Boujelben and Bahoura's design requires a total of 24×15 words to store the cosine values, although it has outstanding data throughput per transformation (DTPT). In contrast, the direct method only requires one register, one adder, and one multiplier to recursively accumulate the multiplication result as (2); however, it still requires 384×16 bits for coefficient ROM. Jo *et al.*'s design [15] also requires an extra adder and a multiplier for the trigonometric approximation so that eventually one register, two adders, and two multipliers are needed for the implementation. This also means that the critical path is longer than the proposed design due to the sequential computation, *i.e.,* sine coefficients must be generated before the multiplication and accumulation are calculated. The hardware cost of Lai *et al.*'s algorithm [20] is comparable to that of the direct method. It requires fewer coefficients in ROM but an additional memory address generator (AddrGen) design. To have a compact design and to handle a larger number of memory generators, the proposed method uses less hardware cost with a simple memory address generator compared with previous methods.

Table 8 shows the FPGA implementation results of the proposed DCT hardware accelerator. The memory address generator requires exactly 42 combinational elements and 7 registers in the implementation. In total, the proposed accelerator requires 104 combinational elements, 87 registers, 3 DSP multipliers, 64 × 16 bits RAM and 32 × 16 ROM for implementation. Moreover, it can operate at a clock rate of 135.85 MHz. As for the computational latency per transformation, it takes a total of 3.56 μs to compute 13 coefficients. As for the power consumption, the proposed hardware would consume 7.17 mW under the condition of a 100 MHz clock rate. Based on these outstanding results, the proposed method would be a simple and compact design for future MFCC applications.

V. CONCLUSION

In this study, a new DCT design with unique-group cosine coefficients for MFCC is presented. It not only reduces the use of the coefficient ROM, but also has low hardware cost compared to previous works. In addition, the proposed method achieves higher computational accuracy in our experiments and will be a better solution for integrating the whole MFCC in the future.

REFERENCES

[1] V. Tiwari, "MFCC and its applications in speaker recognition," *Int. J. Emerg. Technol.*, vol. 1, no. 1, pp. 19–22, 2010.

[2] L. Muda, M. Begam, and I. Elamvazuthi, "Voice recognition algorithms using mel frequency cepstral coefficient and dynamic time warping techniques," *J. Comput.*, vol. 2, pp. 138–143, Mar. 2010.

[3] M. Yousefi and J. H. L. Hansen, "Block-based high performance CNN architectures for frame-level overlapping speech detection," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 28–40, 2021.

[4] T. H. Chowdhury, K. N. Poudel, and Y. Hu, "Time-frequency analysis, denoising, compression, segmentation, and classification of PCG signals," *IEEE Access*, vol. 8, pp. 160882–160890, 2020.

[5] M. B. Er, "A novel approach for classification of speech emotions based on deep and acoustic features," *IEEE Access*, vol. 8, pp. 221640–221653, 2020.

[6] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," 2017, *arXiv:1711.07128*.

[7] A. Gruenstein, R. Alvarez, C. Thornton, and M. Ghodrat, "A cascade architecture for keyword spotting on mobile devices," 2017, *arXiv:1712.03603*.

[8] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*.

[9] V.-L. Dao, V.-D. Nguyen, H.-D. Nguyen, and V.-P. Hoang, "Hardware implementation of MFCC feature extraction for speech recognition on FPGA," in *Advances in Information and Communication Technology* (Advances in Intelligent Systems and Computing). Cham, Switzerland: Springer, Dec. 2016, pp. 248–254.

[10] W.-H. Juang and S.-C. Lai, "A non-feedback-loop and low-computation-complexity algorithm design for a novel 2-D sliding DFT computation," *IEEE Access*, vol. 7, pp. 104912–104920, 2019.

[11] W.-H. Juang, S.-C. Lai, C.-H. Luo, and S.-Y. Lee, "VLSI architecture for novel hopping discrete Fourier transform computation," *IEEE Access*, vol. 6, pp. 30491–30500, 2018.

[12] S.-C. Lai, W.-H. Juang, Y.-S. Lee, S.-H. Chen, K.-H. Chen, C.-C. Tsai, and C.-H. Lee, "Hybrid architecture design for calculating variable-length Fourier transform," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 63, no. 3, pp. 279–283, Mar. 2016.

[13] S.-C. Lai, Y.-P. Yeh, W.-C. Tseng, and S.-F. Lei, "Low-cost and high-accuracy design of fast recursive MDCT/MDST/IMDCT/IMDST algorithms and their realization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 1, pp. 65–69, Jan. 2012.

[14] S.-F. Lei, S.-C. Lai, P.-Y. Cheng, and C.-H. Luo, "Low complexity and fast computation for recursive MDCT and IMDCT algorithms," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57, no. 7, pp. 571–575, Jul. 2010.

[15] J. Jo, H. Yoo, and I.-C. Park, "Energy-efficient floating-point MFCC extraction architecture for speech recognition systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 2, pp. 754–758, Feb. 2016.

[16] O. Boujelben and M. Bahoura, "Efficient FPGA-based architecture of an automatic wheeze detector using a combination of MFCC and SVM algorithms," *J. Syst. Archit.*, vol. 88, pp. 54–64, Aug. 2018.

[17] S. Abed, B. Jamil Mohd, and M. H. Ai Shayeji, "Implementation of speech feature extraction for low-resource devices," *IET Circuits, Devices Syst.*, vol. 13, no. 6, pp. 863–872, Sep. 2019.

[18] Q. Li, Y. Yang, T. Lan, H. Zhu, Q. Wei, F. Qiao, X. Liu, and H. Yang, "MSP-MFCC: Energy-efficient MFCC feature extraction method with mixed-signal processing architecture for wearable speech recognition applications," *IEEE Access*, vol. 8, pp. 48720–48730, 2020.

[19] M. Fariselli, M. Rusci, J. Cambonie, and E. Flamand, "Integer-only approximated MFCC for ultra-low power audio NN processing on multi-core MCUs," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4.

[20] S.-C. Lai, Y.-H. Hung, Y.-C. Zhu, S.-T. Wang, M.-H. Sheu, and W.-H. Juang, "Improved transform algorithm of direct computation of discrete cosine for mel-scale frequency cepstral coefficient," in *Proc. Int. Conf. Electron. Commun., Internet Things Big Data (ICEIB)*, Yunlin, Taiwan, Dec. 2021, pp. 185–187.

SHIN-CHI LAI (Member, IEEE) was born in Taichung, Taiwan, in 1980. He received the B.S. degree in electronic engineering from Chienkuo Technology University, Changhua, Taiwan, in 2002, the M.S. degree in electronic engineering from the National Yunlin University of Science and Technology, Yunlin, Taiwan, in 2005, and the Ph.D. degree from the National Cheng Kung University, Tainan, Taiwan, in 2011. From October 2011 to July 2013, he had been an Assistant Research Fellow with the Department of Electrical Engineering, National Cheng Kung University. From August 2013 to July 2016, he had been an Assistant Professor at the Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan. From August 2016 to July 2019, he was an Associate Professor with the Department of Computer Science and Information Engineering. From August 2019 to July 2021, he has been a Full Professor with the Department of Computer Science and Information Engineering, Nanhua University. He is currently a Full Professor with the Department of Automation Engineering, National Formosa University. His main research interests include signal processing and its circuit design, especially for speech, audio, biomedical, and multimedia applications.

YING-HSIU HUNG received the B.S. degree from the Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan, and the M.S. degree from the Department of Biomedical Engineering, National Yang Ming Chiao Tung University, Taipei, Taiwan. She is currently pursuing the Ph.D. degree in electronic engineering with the National Yunlin University of Science and Technology, Yunlin, Taiwan. Her research interests include deep learning, data augmentation, and digital signal processing.

YI-CHANG ZHU received the B.S. degree from the Department of Computer Science and Information Engineering, Nanhua University, Chiayi, Taiwan, and the M.S. degree from the Master Program of Green Technology for Sustainability, Nanhua University. He is currently pursuing the Ph.D. degree with the Program of Smart Industry Technology Research and Design, National Formosa University, Yunlin, Taiwan. His research interests include microcontroller unit application, digital signal processing, and digital integrated circuit design.

SZU-TING WANG received the B.S. degree from the Department of Computer Science and Information Management, Providence University, Taichung, Taiwan, and the M.S. degree from the Department of Information Engineering and Computer Science, Feng Chia University, Taichung. She is currently pursuing the Ph.D. degree with the Program of Smart Industry Technology Research and Design, National Formosa University, Yunlin, Taiwan. Her research interests include image processing, digital signal processing, and deep learning.

QI-XIAN HUANG is currently pursuing the Ph.D. degree with the National Tsing Hua University, Hsinchu, Taiwan. He used to work as an Intern at Taiwan Semiconductor Manufacturing Corporation, and later to Qualcomm Semiconductor Corporation to be a Senior Firmware Engineer, and also worked as a Researcher at the Artificial Intelligence Research Center, National Chengchi University, Taipei, Taiwan. His research interests include deep learning for computer vision algorithm and applications, beyond 5G, 6G networks, and the Internet of Things security.

MING-HWA SHEU received the M.S. and Ph.D. degrees in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, in 1989 and 1993, respectively. He worked as a Review Committee of the Engineering Department, Ministry of Science & Technology (MOST). From 2008 to 2011, he served as the Chairperson for the Department of Electronic Engineering. From 2015 to 2018, he worked as a Supervisor of Taiwan IC Design Association. He is currently a Full Professor with the Department of Electronic Engineering, National Yunlin University of Science and Technology, Yunlin, Taiwan. His research interests include CAD/VLSI, digital signal process, algorithm analysis, and embedded systems. He has served as the Committee Chair for the E. E. Course Planning for Technical High School, Ministry of Education, Taiwan.

WEN-HO JUANG was born in Taiwan. He received the M.S. degree from the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, in 2010, and the Ph.D. degree from the National Cheng Kung University, in January 2019. From October 2011 to September 2014, he had been an Engineer with the Data Center Networking BU, MediaTek Inc., Hsinchu, Taiwan. He is currently an Assistant Professor with the Department of Computer Science and Information Engineering, National Formosa University, Yunlin, Taiwan. His research interests include digital signal processing, VLSI/FPGA system design, microprocessor applications, and the IoT application in agriculture for smart farming.