S & M 3065

Memory-efficient Very Large Scale Integration Architecture of 2D Algebraic-integer-based Daubechies Discrete Wavelet Transform

Tiancai Lan,[1] Chih-Hsien Hsia,[2] Po-Ting Lai,[3]
Hsien-Wei Tseng,[4*] and Cheng-Fu Yang[5,6**]

[1]School of Mathematics and Information Engineering, Longyan University, Longyan 364012, China
[2]Department of Computer Science and Information Engineering, National Ilan University, Ilan 260, Taiwan
[3]Department of CS12/HPC1/SRD2, MediaTek, Hsinchu City 300, Taiwan
[4]College of Artificial Intelligence, Yango University, Mawei District, Fujian 350015, China
[5]Department of Chemical and Materials Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
[6]Department of Aeronautical Engineering, Chaoyang University of Technology, Taichung 413, Taiwan

(Received July 4, 2022; accepted August 25, 2022)

Keywords: algebraic integer, Daubechies wavelet, transpose memory, interlaced read scan algorithm

Low memory requirement and reduced latency have been two major concerns in the implementation of the 2D Daubechies discrete wavelet transform. In this study, a multilevel 2D Daubechies-4 (Daub-4) wavelet filter architecture based on an algebraic integer (AI) is investigated. This architecture can improve the traditional Daub-4 very large scale integration (VLSI) architecture design and reduce the number of adders in a 1D Daub-4 filter module architecture. The is because the AI-based multilevel architecture does not require any multipliers and can improve the efficiency of accurate calculations. In addition, to solve the problem of the large transpose memory generated by multimedia chip design, we examine the uses of $N \times N$ image inputs in the calculation of the Daub-4 filter by importing them into the interlaced read scan algorithm. This investigated architecture not only reduces the size of the transpose memory from N^2 to 10 or 21 (in the Daub-4 and Daub-6 modes, respectively) but also speeds up the sensing and reading of signals and the calculations. We also show that when a field-programmable gate array is combined with the investigated system, it can enhance the implementation of 2D multilevel AI-based Daub-4 and Daub-6 VLSI architectures.

1. Introduction

In recent years, the transpose memory (TM) cost has been a very important problem in multimedia IC design for the 2D discrete wavelet transform (DWT) architecture.(1,2) This architecture is based on the Daubechies scheme and produces both time domain and frequency domain characteristics. The DWT architecture is also suitable for applications performing image or video compression.(3–6) The problem of the signal accuracy of DWT architectures based on the Daubechies DWT has been researched and discussed.(7–9) For example, the Daubechies-4 (Daub-4) DWT is often applied to increase the smoothness of information, and the Daubechies-6

*Corresponding author: e-mail: hsienwei.tseng@gmail.com
**Corresponding author: e-mail: cfyang@nuk.edu.tw
https://doi.org/10.18494/SAM4025

(Daub-6) DWT is often used to analyze long signals.(10) Wavelets based on Daub-4 or Daub-6 have been used to process medical images, especially in medical image compression and texture analysis, because the details of the images are very important.(11,12) The Daub-4 transform has been used to increase the calculation accuracy, because it is reasonably simple to implement and effective at preprocessing data from 32 sensors.(13) When the 2D discrete Daub-4 architecture is combined with different algorithms, it can have different functions. For example, when it is combined with the target coverage algorithm, the system can identify the optimal position of each sensor to cover a maximum number of targets.(14) Madishetty *et al.*(15) studied a multilevel 2D Daubechies wavelet transform based on an algebraic integer (AI). The two methods that they investigated avoided the intermediate reconstruction steps in the design process, and it was only necessary to perform the reconstruction at the end of the analysis step.(15) In addition, the very large scale integration (VLSI) architecture does not use multipliers except in the final reorganization stage, which can increase its speed and reduce the area used by hardware.(10,12,15–22)

The Daubechies wavelet transform can be used to compress sensing data by employing a sparsifying linear transformation for different imaging systems, including those for optical imaging, radio astronomy, computed tomography, and magnetic resonance imaging.(21) Casson investigated an analog domain signal processing circuit, which was used to approximate the output of the DWT architecture used in combination with ultralow-power wearable sensors.(22) Therefore, in this study, we investigated an architecture to combine the above-mentioned hardware to reduce the use of TM and the reading time and enhance signal sensing and the accuracy of calculations. In this study, the novelty of the investigated system, which includes the DWT architecture, the AI architecture, and the VLSI architecture, is investigated to improve the efficiency of use of TM in the above hardware and to change the filter structure to reduce the signal sensing and reading times and increase the calculation speed. Moreover, we show that by using the AI architecture, the hardware cost and the numbers of adders and multipliers can be further reduced.

In Sect. 2, we introduce the traditional 2D multilevel Daubechies wavelet transformation architecture based on an AI architecture and present its coefficients. In Sect. 3, we investigate the 2D multilevel Daub-4 and Daub-6 wavelet transform VLSI architectures, and we introduce the interlaced read scan algorithm (IRSA) used for the calculation. In Sect. 4, we discuss and analyze the results obtained by implementing the investigated architecture on a field-programmable gate array (FPGA). Finally, a conclusion is given in Sect. 5.

2. Daubechies Filter Based on AI

2.1 Theoretical background

The Daubechies filter has many different forms and can be designed for input signals with values of 1 to 20.(23–25) According to the duration of the input signal, we consider a simple example and express the input signal $A^{(Daub\text{-}4)}$ as(7,12,14)

$$A^{(Daub\text{-}4)} = \frac{1}{4\sqrt{2}}\left[1+\sqrt{3}\,3+\sqrt{3}\,3-\sqrt{3}\,1-\sqrt{3}\right]^T. \tag{1}$$

However, an AI can be expressed as the root of a real number or a complex polynomial[26] and can also be used to define a coding mapping, that is, a constant can be used to accurately represent a specific irrational number and consider the roots of the polynomial. The AI is selected from an integer set $\mathbb{Z}$, which contains different integer algebras, and the AI can be expressed as

$$y = a + b\times\zeta \text{ and } y = c + d\times\zeta_1 + e\times\zeta_2 + f\times\zeta_1\zeta_2, \tag{2}$$

where a and b are integers. The AI coding is calculated on the basis of this set. Note that this set is sufficient to represent the coefficients of the Daub-4 and Daub-6 filters. Additionally, AI coding can be delayed until the final stage of the calculations, and the filter coefficients of Daub-4 and Daub-6 can be expressed as

$$A^{(Daub\text{-}4)} = \left[1+\zeta_3+\zeta_3-\zeta_1-\zeta\right]^T, \tag{3}$$

$$F^{(Daub\text{-}6)} = \begin{bmatrix} 1+\zeta_1+\zeta_2 \\ 5+\zeta_1+3\zeta_2 \\ 10-2\zeta_1+2\zeta_2 \\ 10-2\zeta_1-2\zeta_2 \\ 5+\zeta_1-3\zeta_2 \\ 1+\zeta_1-\zeta_2 \end{bmatrix}. \tag{4}$$

Therefore, both the Daub-4 and Daub-6 filters can also be expressed as the following equations, where F_1 and F_ζ in the Daub-4 equation represent the two outputs generated by the calculations. For Daub-6, after the filter operation is completed, three parts of the output are generated: F'_1, F_{ζ_1} and F_{ζ_2}.

$$F^{(Daub\text{-}4)} = F_1 + \zeta\cdot F_\zeta, \tag{5}$$

$$F^{(Daub\text{-}6)} = F'_1 + \zeta_1\cdot F_{\zeta_1} + \zeta_2\cdot F_{\zeta_2}, \tag{6}$$

where $F_1 = [1\ 3\ 3\ 1]^T$, $F_\zeta = [1\ 1\ -1\ -1]^T$, $F'_1 = [1\ 5\ 10\ 10\ 5\ 1]^T$, $F_{\zeta_1} = [1\ 1\ -2\ -2\ 1\ 1]^T$, $F_{\zeta_2} = [1\ 3\ 2\ -2\ -3\ -1]^T$.

2.2 2D filter

In this study, we use the Daub-4 filter to describe the multilevel architecture, the 2D encoding operation process, and the results based on integer algebra. The operation architecture and results of Daub-6 will be the same as those of Daub-4. Here, V and C denote a matrix and column, respectively; thus, v is a vector point and c is a column point. The symbols represent the two-sampling operation in the vertical and horizontal directions. The symbol $\oplus$ is defined as

$$V \oplus C \triangleq (2 \downarrow 1)\ [v{*}c_0\ v{*}c_1\ v{*}c_2\ \dots\ v{*}c_{N-1}] = [v{*}c_0\ v{*}c_2\ v{*}c_4\ \dots\ v{*}c_{N-2}], \tag{7}$$

where * denotes convolution. The similar $\ominus$ operation can be written as

$$v \ominus C \triangleq (v \oplus C^T)^T. \tag{8}$$

For the first-level 2D Daub-4 analysis, the formula is

$$\beta_1{}^2 \cdot A_1 = F^{(Daub\text{-}4)} \ominus F^{(Daub\text{-}4)} \oplus A_0, \tag{9}$$

where A_0 is the original matrix value of the input image. Substituting Eq. (5) into Eq. (9), we obtain

$$\beta_1{}^2 \cdot A_1 = (F_1 + \zeta \cdot F_\zeta) \ominus (\mathrm{F}_1 + \zeta \cdot F_\zeta) \oplus A_0 = F_1 \ominus F_1 \oplus A_0 + \zeta \cdot F_\zeta \ominus F_1 \oplus A_0 + \zeta \cdot F_1 \ominus F_\zeta \oplus A_0 + \zeta^2 \cdot F_\zeta \ominus F_\zeta \oplus A_0. \tag{10}$$

When $\zeta^2 = 3$ is substituted in Eq. (10), it can be simplified to

$$\beta_1{}^2 \cdot A_1 = A_1{}^{(1)} + \zeta \cdot A_1{}^{(\zeta)}, \tag{11}$$

where $A_1{}^{(1)}$ and $A_1{}^{(\zeta)}$ are given in Fig. 1.

In further analysis operations, a second-level analysis is performed as follows:

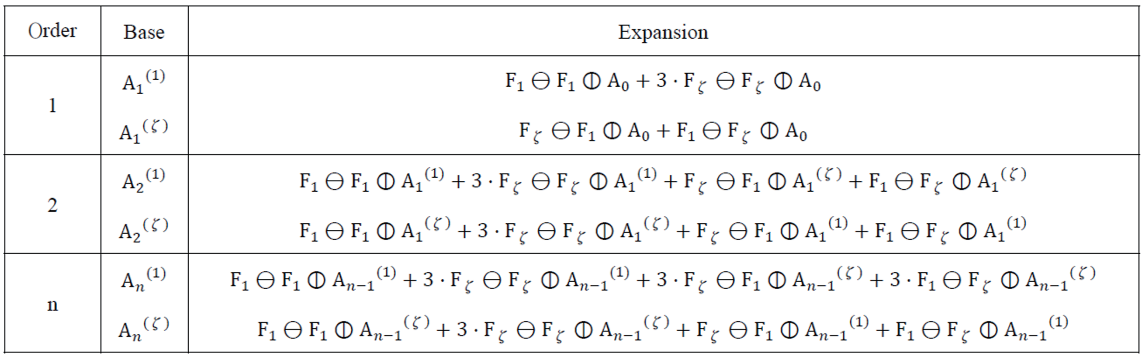
$$\beta_1{}^2 \cdot A_2 = F^{(Daub\text{-}4)} \ominus F^{(Daub\text{-}4)} \oplus A_1. \tag{12}$$

Substituting Eq. (11) into Eq. (12), we obtain

$$\beta_1{}^4 \cdot A_2 = F^{(Daub\text{-}4)} \ominus F^{(Daub\text{-}4)} \oplus (A_1{}^{(1)} + \zeta \cdot A_1{}^{(\zeta)}) = A_2{}^{(1)} + \zeta \cdot A_2{}^{(\zeta)}. \tag{13}$$

The fully expanded formulas of $A_2{}^{(1)}$ and $A_2{}^{(\zeta)}$ are also given in Fig. 1.

Equation (11) can be used to express the results of analyzing the remaining levels, for example, the analysis results of the nth level can be expressed as



Order	Base	Expansion
1	$A_1^{(1)}$	$F_1 \ominus F_1 \obar A_0 + 3 \cdot F_\zeta \ominus F_\zeta \obar A_0$
	$A_1^{(\zeta)}$	$F_\zeta \ominus F_1 \obar A_0 + F_1 \ominus F_\zeta \obar A_0$
2	$A_2^{(1)}$	$F_1 \ominus F_1 \obar A_1^{(1)} + 3 \cdot F_\zeta \ominus F_\zeta \obar A_1^{(1)} + F_\zeta \ominus F_1 \obar A_1^{(\zeta)} + F_1 \ominus F_\zeta \obar A_1^{(\zeta)}$
	$A_2^{(\zeta)}$	$F_1 \ominus F_1 \obar A_1^{(\zeta)} + 3 \cdot F_\zeta \ominus F_\zeta \obar A_1^{(\zeta)} + F_\zeta \ominus F_1 \obar A_1^{(1)} + F_1 \ominus F_\zeta \obar A_1^{(1)}$
n	$A_n^{(1)}$	$F_1 \ominus F_1 \obar A_{n-1}^{(1)} + 3 \cdot F_\zeta \ominus F_\zeta \obar A_{n-1}^{(1)} + 3 \cdot F_\zeta \ominus F_1 \obar A_{n-1}^{(\zeta)} + 3 \cdot F_1 \ominus F_\zeta \obar A_{n-1}^{(\zeta)}$
	$A_n^{(\zeta)}$	$F_1 \ominus F_1 \obar A_{n-1}^{(\zeta)} + 3 \cdot F_\zeta \ominus F_\zeta \obar A_{n-1}^{(\zeta)} + F_\zeta \ominus F_1 \obar A_{n-1}^{(1)} + F_1 \ominus F_\zeta \obar A_{n-1}^{(1)}$

Fig. 1. Daub-4 filter.

$$\beta_1^{2n} \cdot A_n = A_n^{(1)} + \zeta \cdot A_n^{(\zeta)}, \quad n \geq 2, \tag{14}$$

where $A_2^{(1)}$ and $A_2^{(\zeta)}$ are given in Fig. 1. Note that the required multiplier of 3 is achieved by using shift and add (SAA), as $3 \cdot m = (m \ll 1) + m$, where m is an integer used to eliminate the need for a multiplier and reduce hardware resource consumption.

2.3 Multilevel filter analysis architecture

The AI-encoded Daub-4 filter is calculated using the extended integers. Thus, when a value is input and the first level of the AI wavelet transformation has been analyzed, two parts of the output are generated, both of which are related to its basic elements. In Fig. 2, the AI-based 2D fourth-level Daub-4 filter architecture is taken as an example. This architecture includes N point A_0 input signals and the output results $A_1^{(1)}$ and $A_1^{(\zeta)}$ after AI encoding. The output results of the two parts are calculated on the basis of the previous input set $\{1, \zeta\}$, and the values of these two parts are calculated in the combination block and in the final reorganization stage to obtain the final output image. The AI-based Daub-4 filter VLSI architecture requires six adders to form an AI Daub-4 filter,[13] and each integer calculation AI block contains three AI Daub-4 filters. The VLSI architecture of the device generates 18 adders in the AI block, as shown in Fig. 3.

3. VLSI Architecture of Improved Daubechies Hardware Filter Based on AI

Figure 4 shows our improved Daub-4 AI filter with the VLSI architecture investigated in this study, in which only three adders are required. Compared with the previous architectures, which require six adders,[13] it can reduce the complexity by 50%. Each group of AI blocks contains three Daub-4 AI filters. Therefore, only nine adders remain in the AI block. The framework can calculate the integer, and the integer contains the ζ coefficient separately.

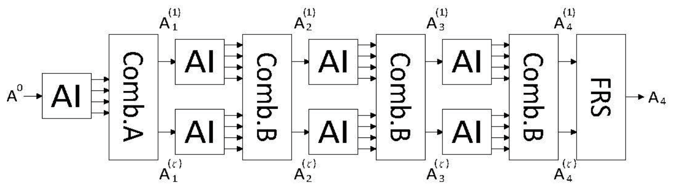


Fig. 2. Block diagram of 2D multilevel filter based on AI.

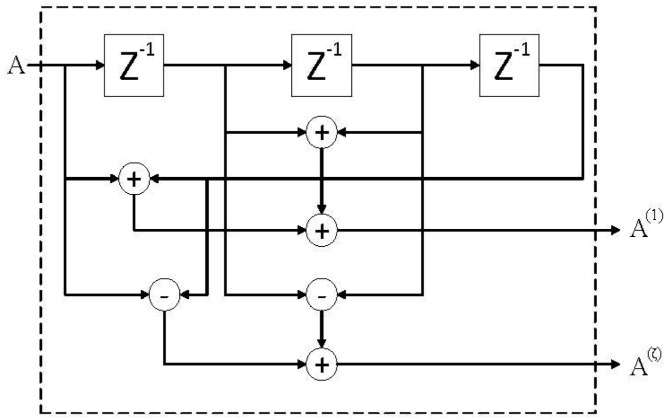


Fig. 3. VLSI architecture of Daub-4 AI filter.

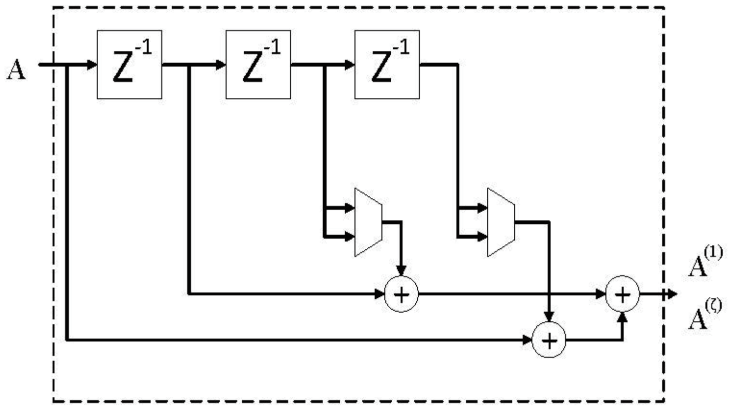


Fig. 4. Improved Daub-4 AI filter VLSI architecture.

3.1 AI block architecture

As shown in Fig. 5(a), there are three Daub-4 filter architectures in a 2D AI block: one horizontal filter and two vertical filters. After each analysis, high- and low-frequency signals are output. The two vertical Daub-4 filter architectures are required to calculate the two output values from the horizontal direction (the second-dimension calculation), then the vertical Daub-4 filter architectures calculate them separately for four output results. In the investigated improved AI VLSI architecture, a first in, first out (FIFO) memory architecture is used to

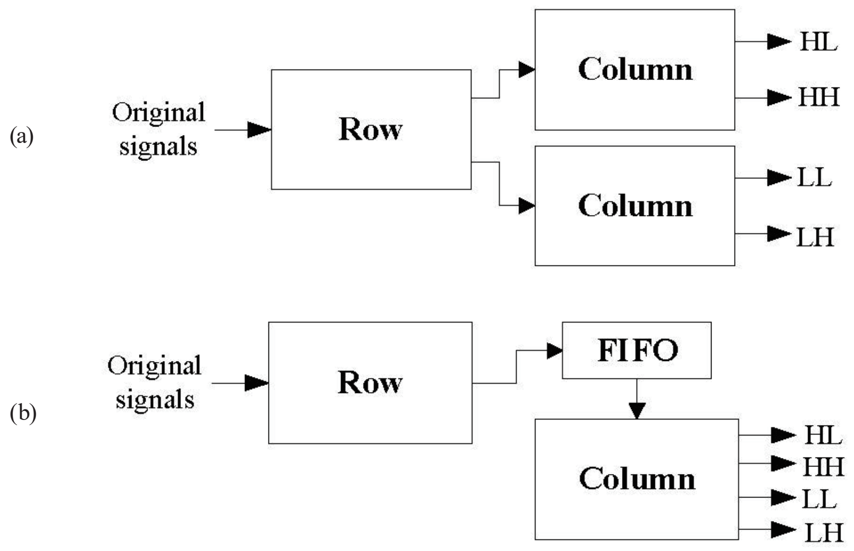


Fig. 5. 2D first-level filter architecture: (a) traditional architecture and (b) improved architecture.

replace the vertical data. As shown in Fig. 5(b), the output data from the horizontal direction is stored in the FIFO memory, and only one vertical Daub-4 filter is needed to read and calculate the four values. Regardless of the increase in the amount of input data, the FIFO memory need only be used 10 times. Compared with the previously reported architectures, the investigated architecture can reduce the number of adders in the AI block by 12.

3.2 IRSA

A 2D DWT is composed of two 1D DWTs and TM. The DWT requires a large number of calculation processes, making it prone to a large hardware cost, a long critical path, and large TM use when the resolution of the input image is increased.[27–34] As shown in Fig. 6, we first use an AI-based 2D Daub-4 filter as an example to illustrate the IRSA.

We assume that an image with a matrix of size $N \times N$ is input and that the output result is stored in the TM. After the process of the first Daub-4 filter, the high and low frequencies are read out to the second Daub-4 filter one by one, and four sub-bands are obtained. For the output results, the time required to access and save data in the TM increases the time required for the outputs of the four sub-bands to be obtained by the processes of the 2D DWT. Using this investigated architecture, the IRSA can solve the above problems and is applied to the Daubechies AI architecture to reduce the use of TM.

Taking a 6×6 input image as an example, Fig. 7(a) shows the AI-based 2D Daud-4 traditional input reading architecture. The black pixels represent the original input image signals, and the white pixels are the signals from the image boundary extension. During the first-dimensional AI-based Daub-4 filter signal reading process, the four pixel values in the red frame in the left frame of Fig. 7(a) are used to calculate the first dimension of the output (as shown by the red pixels in the middle frame). Due to the sub-sampling, four pixels are read simultaneously to

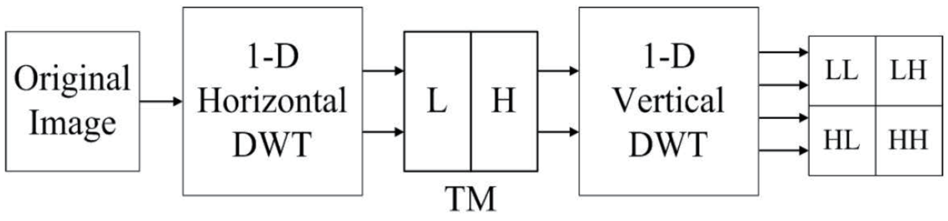


Fig. 6. Traditional 2D DWT.

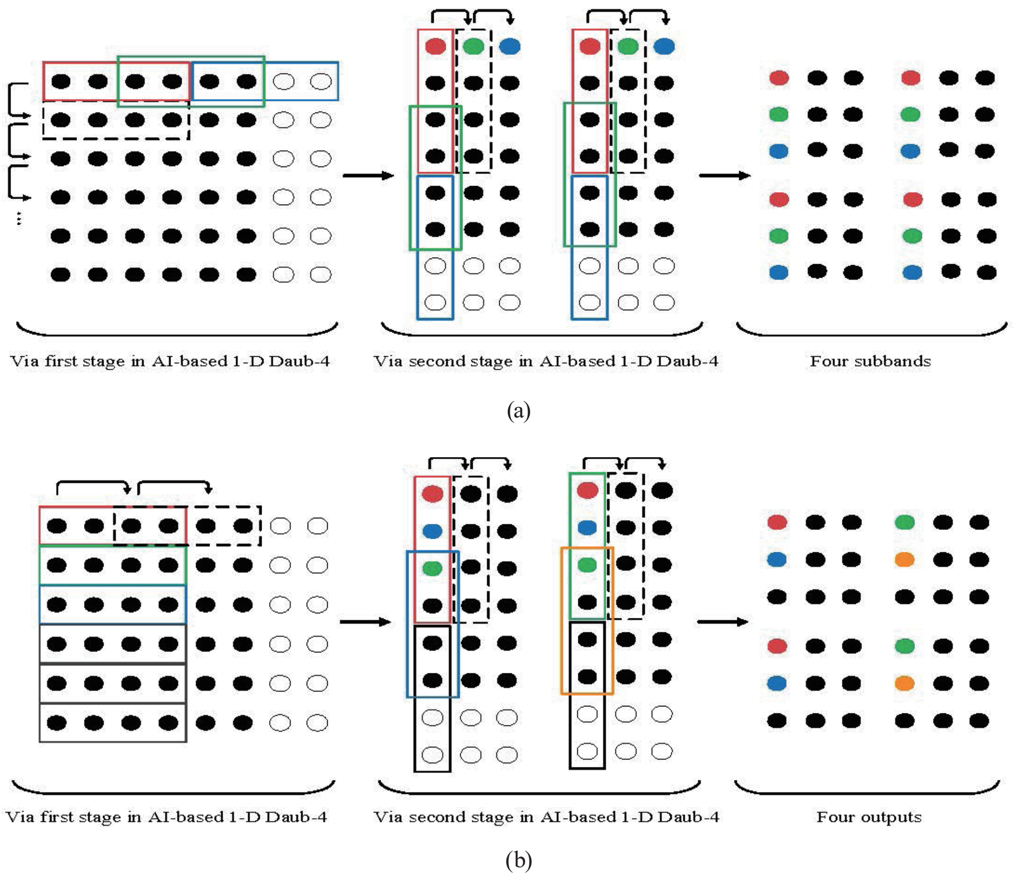

Via first stage in AI-based 1-D Daub-4
Via second stage in AI-based 1-D Daub-4
Four subbands
(a)
Via first stage in AI-based 1-D Daub-4
Via second stage in AI-based 1-D Daub-4
Four outputs
(b)

Fig. 7. (Color online) IRSA: (a) Traditional input reading architecture and (b) improved input reading architecture.

calculate every two pixels each time. The order is based on the red, green, and blue borders. Then, the next column is read until the end of the calculation of the last column. Next, the red, green, and blue pixels in the middle frame are their output results, which are used to generate the second-dimensional output [as shown in the right frame of Fig. 7(a)], and the reading order is based on red, green, and blue frames. Then, the next line is read until the calculation of the last line is completed, and the red, green, and blue pixels in the right frame are the output results. The output through the first dimension is stored in the TM. The middle frame shows the second-dimensional AI-based Daub-4 filter signal reading process, which reads four pixels from the

TM. After calculation, the output results of four different sub-bands shown in the right frame of Fig. 7(a) are generated.

The IRSA used in this study is shown in Fig. 7(b), and the first-dimensional Daub-4 filter based on the AI is shown in the left frame. First, the four input image pixels in the red frame are calculated. Then the four input pixels in the green border of the next row are read for use in the calculation. This sequence is based on the red, green, and blue borders until the last row, and then the four pixels in each row below are read. This rule is applied until the input pixels of the last line are read. Then, the input pixels in the first line of the next column (as shown by the black dashed border in the upper left corner of Fig. 7(b)) are processed using the same rule. For an input image with a size of $N \times N$, the red, green, and blue pixels in the middle part are based on the output of the first-dimensional AI-based Daub-4 filter in the previous step and stored in the TM. To calculate the output of the second-dimensional AI-based Daub-4 filter, the storage size of the TM is reduced from $N \times N$ to 10. Next, the four pixels in the red frame are read from the TM, and the two red pixels in the right frame are generated. These processes can speed up the final output and synchronize the first and second dimensions of the AI-based Daub-4 calculation. Then, the new pixels calculated from the first dimension are continuously stored in the TM and overwrite the old pixels. The second-dimensional Daub-4 filter continues to read the following pixels for use in calculations. This calculation method is based on the red, green, blue, and yellow borders, which output the red, green, blue, and yellow pixels in the right frame of Fig. 7(b), respectively.

As shown in Fig. 8, the storage size of the used TM is 10, which can be used to store the two output results of the first dimension. The input pixels are stored one at a time in the order 1, 2, 3, … 10. When a signal is stored in pixel 7, the Daub-4 filter of the second dimension uses pixels 1, 3, 5, and 7 to calculate the output of the two pixels of the second dimension simultaneously and, at the same time, pixels 8 and 9 are also stored in the TM in sequence. When the above signals are calculated, pixels 2, 4, 6, and 8 are used to calculate the two output pixels. At this time, pixels 10 and 11 are stored in the TM, and pixel 11 is stored to substitute the data in the position of pixel 1. Because of the sampling, four pixels are read simultaneously and, also, every two pixels are calculated simultaneously. Thus, pixels 5, 7, 9, and 11 are read for use in the calculation, then pixels 12 and 13 replace pixels 2 and 3 and store them in the TM. Therefore, the IRSA signal-reading architecture can greatly reduce the use of TM from N^2 to 10 bytes in this study.

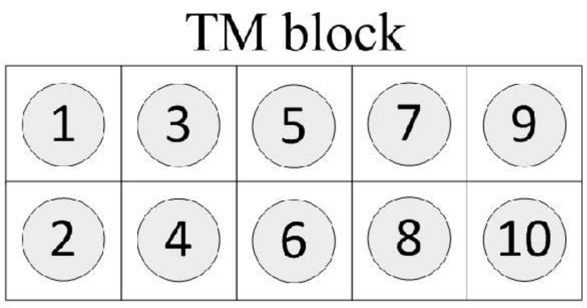


Fig. 8. TM architecture.

4. Experimental Results

In this research, we improve the VLSI architecture of the AI-based Daub-4 filter. The hardware architecture is implemented on the Xilinx FPGA chip and uses the ZedBoard platform. 512 × 512 input images are used for testing, and the hardware test results are compared with MATLAB simulation results.

4.1 VLSI hardware architecture

In this research, we use the Xilinx 7 series ZedBoard Zynq-7000 ARM/FPGA SoC development board to implement the calculations. The AI-based Daub-4 filter is an adder used in the traditional 2D architecture. The investigated AI-based Daub-4 filter has the advantage of enhancing the performance as the architecture changes from 2D's first level (Level 1) architecture to a multilevel one. As shown in Table 1, the traditional 2D Level 1 architecture requires 21 adders and the multilevel (Level 4) architecture requires 147 adders. In contrast, the VLSI architecture requires fewer adders, and the hardware cost is 43% lower for all architectures, regardless of the level. Therefore, the IRSA improves the efficiency of the investigated architecture, greatly reducing the amount of computing time and the hardware cost.(9,10,13,15,16) This algorithm is used to change the reading and memory access architectures and reduce the amount of used TM. With increasing resolution of the input images, the traditional architecture has a higher hardware cost. In past studies, the impacts on the output signals caused delays of the output signals. In contrast, the improved reading architecture in this study only needs 10 adders. The investigated architecture can also be extended to AI-based Daub-6 filters, as shown in Table 2.

Table 1
Number of filter adders used in Daub-4 architecture.

	Original Daub-4	This work	Cost reduction (%)
AI filter	6	3	50
AI block	18	6	33
Level 1	21	9	43
Level 2	63	27	43
Level 3	105	45	43
Level 4	147	63	43
TM (bytes)	N^2	10	

Table 2
Number of filter adders used in Daub-6 architecture.

	Original Daub-6	This work	Cost reduction (%)
AI filter	12	5	42
AI block	48	15	31
Level 1	57	24	43
Level 2	288	123	43
Level 3	519	222	43
Level 4	750	321	43
TM (bytes)	N^2	21	

Comparing the results obtained from the Daub-4 and Daub-6 architectures in this research with the results obtained from other studies, we find that the improved architecture has the advantage of not requiring multipliers and only a small number of adders are needed, and the IRSA further improves the architecture (Tables 3 and 4). The investigated architecture has reduced hardware requirements without affecting the delay time of the outputs. For example, by comparing the results of the investigated Daub-4 architecture in this study with the results in Refs. (21) and (35), it can be seen that the investigated architecture does not need multipliers and only two more adders are needed. Moreover, compared with the results in Refs. (10), (15), and (36), the investigated Daub-4 architecture needs fewer adders.

4.2 Image quality evaluation

In this section, we demonstrate that the image quality can be maintained when using the proposed architecture and that the test images of Lena, Goldhill, Boat, and Bridge are improved by the AI-based Daub-4 filters. The software/hardware simulation environment in this study is a PC with an Inter Celeron G1840 CPU (4.00 GHz), 16 GB of RAM, Windows 7, and ISE Design Suite 14.7, with MATLAB R2016a used as the test platform. Figure 9 shows, from left to right, the original images to the results of 2D third-level analysis. The output images are improved by the AI-based Daub-4 filter. The improved architecture can also be imported into other Daubechies filters, such as the Daub-6 filter, to reduce the number of adders, and then the IRSA can be used to further reduce the use of TM.

Table 5 shows the peak signal-to-noise ratio (PSNR) values of the images shown in Fig. 9 output by the improved Daub-4 filter. The size of the original images is 512 × 512 pixels, and the output of each level of analysis is only one-quarter of the size of the previous level to maintain the image quality: 256 × 256 pixels for Level 1, 128 × 128 pixels for Level 2, and 64 × 64 pixels

Table 3
Comparisons of first-order 2D Daub-4 filter architecture.

Investigated	Ours	(10)	(15)	(21)	(35)	(36)
Multipliers	0	0	0	2	3	0
Adders	6	9	18	4	4	30
TM (bytes)	10	—	N^2	—	—	$N^2/4 + 2N$
Latency	$4Ta$	$6Ta$	$5Ta$	—	—	—
CT	$3N^2/4$	—	$3N^2/4$	—	—	$N^2/2$

Latency represents time from signal input to output, *Ta* and *Tm*: numbers of adders and multipliers that have passed, respectively, CT: time required to calculate image.

Table 4
Comparisons of first-level and 2D Daub-6 filter architecture.

Investigated	Ours	(10)	(15)	(17)	(21)	(37)
Multipliers	0	0	0	0	4	0
Adders	15	18	48	48	6	16
TM (bytes)	21	—	N^2	N^2	—	—
Latency	$6Ta$	$5Ta$	$6Ta$	$6Ta$	—	—
CT	$3N^2/4$	—	$3N^2/4$	$3N^2/4$	—	—

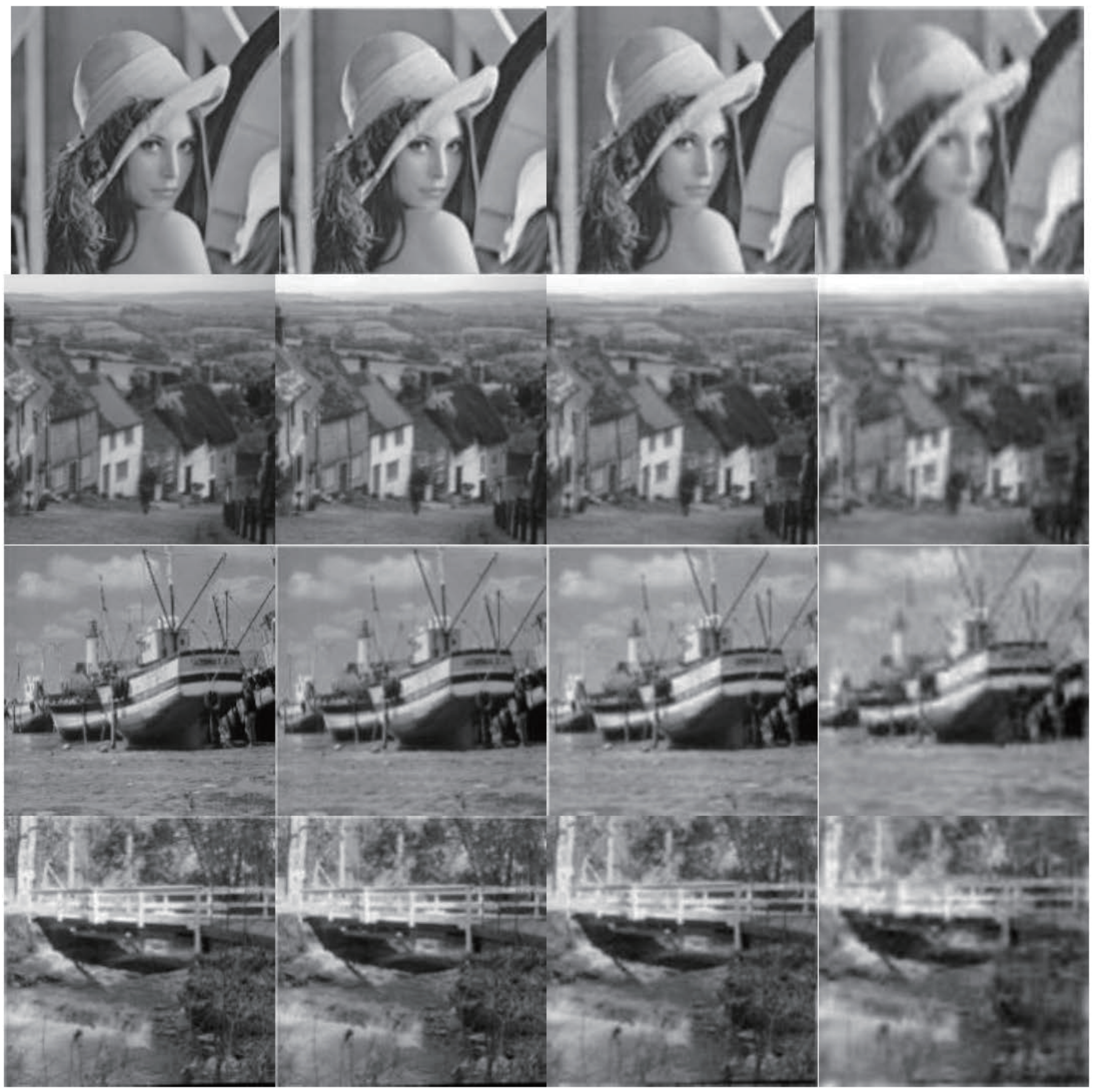


Fig. 9. Test results from original image to third-level analysis. (a) Original input image and results of (b) first-level, (c) second-level, and (d) third-level analyses.

Table 5
PSNR values of each ordered image after Daub-4 analysis.

Original Image (dB)	Level 1	Level 2	Level 3
Lena	71.11	65.58	66.27
Goldhill	69.72	64.31	63.12
Boat	72.48	70.35	65.81
Bridge	67.49	65.27	62.18

for Level 3. These results demonstrate that the investigated algorithm requires less memory to store the processed images and does not increase the PSNR value.

5. Conclusions

The system investigated in this research successfully used an AI-based 2D Daub-4 filter with the IRSA. We showed that the investigated IRSA architecture can implement the VLSI architectures of the AI-based Daub-4 and Daub-6 2D multilevel Daubechies DWT on an FPGA and that they had higher efficiency than the traditional Daub-4 VLSI architecture. The investigated system also had more efficient use of the TM in the traditional Daub-4 VLSI architecture owing to the new filter structure to reduce the amount of signal reading and increase the calculation speed. Two long-standing problems with the VLSI hardware architecture have been effectively solved in this study. First, the numbers of adders and multipliers were reduced. Second, the size of TM was reduced from N^2 in traditional architectures to 10 or 21 (in the Daub-4 and Daub-6 modes, respectively). Therefore, the investigated architecture can be applied to other Daubechies filters with different basis functions to increase the signal accuracy and speed and reduce the hardware cost.

Acknowledgments

This work was supported by the Natural Science Foundation of Fujian Province (2020J01366), Longyan Science and Technology Project (2019LYF5001) and by Project Nos. MOST 110–2622-E-390–002 and MOST 110–2221-E-390–020.

References

1 M. Vetterli and C. Herley: IEEE Trans. Signal Process. **40** (1992) 2207. https://doi.org/10.1109/78.157221
2 C. H. Hsia, J. M. Guo, and J. S. Chiang: IEEE Trans. Circuits Syst. Video Technol. **19** (2009) 1202. https://doi.org/10.1109/TCSVT.2009.2020259
3 M. Rabbani and R. Joshi: Signal Process. Image Commun. **17** (2002) 3. https://doi.org/10.1016/S0923–5965(01)00024–8
4 M. B. Amor, F. Kammoun, and N. Masmodi: 2016 Int. Image Process. Applications and Systems (IPAS) (2016) 1–4. https://doi.org/10.1109/IPAS.2016.7880121
5 S. K. Mishra, L. N. Tripathy, and S. C. Swain: 2017 Int. Conf. Innovative Mechanisms for Industry Applications (ICIMIA) (2017) 295–301. https://doi.org/10.1109/ICIMIA.2017.7975622
6 D. P. Ladumor, I. N. Trivedi, R. H. Bhesdadiya, and P. Jangir: 2017 3rd Int. Conf. Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB) (2017) 283–288. https://doi.org/10.1109/AEEICB.2017.7972430
7 K. A. Wahid, V. S. Dimitrov, G. A. Jullien, and W. Badawy: 36th Asilomar Conf. Signals, Systems and Computers (2002) 967–971. https://doi.org/10.1109/ACSSC.2002.1197320
8 J. P. Andrew, P. O. Ogunbona, and F. J. Paoloni: IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP '94) (1994) V/589-V/592. https://doi.org/10.1109/ICASSP.1994.389443
9 B. K. Mohanty, A. Mahajan, and P. K. Meher: IEEE Trans. Circuits Syst. II: Express Briefs **59** (2012) 434. https://doi.org/10.1109/TCSII.2012.2200169
10 M. A. Islam and K. A. Wahid: IEEE Trans. Circuits Syst. II: Express Briefs **57** (2010) 716. https://doi.org/10.1109/TCSII.2010.2056111
11 K. Wahid, S.B. Ko, and D. Teng: 2008 IEEE Int. Joint Conf. Neural Networks (IEEE World Congress on Computational Intelligence) (2008) 2761–2765. https://doi.org/10.1109/IJCNN.2008.4634186
12 K. A. Wahid, V. S. Dimitrov, G. A. Jullien, and W. Badawy: Conf. Record of the 36th Asilomar Conf. Signals, Systems and Computers (2002) 967–971. https://doi.org/10.1109/ACSSC.2002.1197320
13 H. S. Abdel-Aty-Zohdy, S. Roth, and E. Mebrahtu: IEEE 56th Int. Midwest Symposium on Circuits and Systems (MWSCAS) (2013) 616–620. https://doi.org/10.1109/MWSCAS.2013.6674724.
14 T. Ganesan and P. Rajarajeswari: Int. J. Commun. Netw. Distrib. Syst. **28** (2022) 337. https://doi.org/10.1504/IJCNDS.2022.122170

15 S. K. Madishetty, A. Madanayake, and R. J. Cintra: IEEE Trans. Circuits Syst. I Regul. Pap. **60** (2013) 1455. https://doi.org/10.1109/TCSI.2012.2221171
16 S. K. Madishetty, A. Madanayake, R. J. Cintra, and V. S. Dimitrov: IEEE Trans. Circuits Syst. I Regul. Pap. **61** (2014) 1984. https://doi.org/10.1109/TCSI.2014.2298283
17 S. Mohammadi and A. Javadi: 2010 Int. Conf. Signal Acquisition and Processing (2010) 145–150. https://doi.org/10.1109/ICSAP.2010.47
18 A. Madanayake, R. J. Cintra, and D. Onen: IEEE Trans. Circuits Syst. Video Technol. **22** (2012) 915. https://doi.org/10.1109/TCSVT.2011.2181232
19 P. Balakrishnan, M. M. Hasan, and K. A. Wahid: Can. J. Electr. Comput. Eng. **37** (2014) 127–134. https://doi.org/10.1109/CJECE.2014.2316227
20 S. C. Lai, Y. P. Yeh, and S. F. Lei: IEEE Trans. Circuits and System. II: Analog and Digital Signal Process. **59** (2012) 511. https://doi.org/10.1109/TCSII.2012.2204115
21 N. Dwork, D. O'Connor, C. A. Baron, E. M. I. Johnson, A. B. Kerr, J. M. Pauly, and P. E. Z. Larson: Signal Image Video Process. **15** (2021) 1407. https://doi.org/10.1007/s11760-021-01872-y
22 A. J. Casson: Sensors **15** (2015) 31914. https://doi.org/10.3390/s151229897
23 K. A. Wahid, M. A. Islam, and S.-B. Ko: IEEE Int. Symp. Circuits and Systems (2011) 2157–2160. https://doi.org/10.1109/ISCAS.2011.5938026
24 M. Oltean and M. Nafornita: IEEE Int. Conf. Communications (2010) 343–346. https://doi.org/10.1109/ICCOMM.2010.5509085
25 D. N. Vizireanu and R. O. Preda: IEEE Int. Conf. Telecommunication in Modern Satellite, Cable and Broadcasting Services (2005) 518–521. https://doi.org/10.1109/TELSKS.2005.1572166
26 R. Baghaie and V. Dimitrov: Comput. Math. Appl. **41** (2001) 1403. https://doi.org/10.1016/S0898–1221(01)00105–5
27 C. Diou, L. Torres, and M. Robert: IEEE Int. Conf. Emerging Technologies and Factory Automation (2001) 179–186. https://doi.org/10.1109/ETFA.2001.997684
28 K. Andra, C. Chakrabarti, and T. Acharya: IEEE Trans. Signal Process. **50** (2002) 966. https://doi.org/10.1109/78.992147
29 P. Y. Chen: IEEE Trans. Computers **53** (2004) 386. https://doi.org/10.1109/TC.2004.1268396
30 C. T. Huang, P. C. Tseng, and L. G. Chen: IEEE Trans. Signal Process. **53** (2005) 1575. https://doi.org/10.1109/TSP.2005.843704.
31 M. Vishwanath, R. M. Owens, M. J. Irwin, and J. Irwin: IEEE Trans. Circuits Syst. II: Analog Digital Signal Process. **42** (1995) 305–316. https://doi.org/10.1109/82.386170
32 C. H. Hsia, J. S. Chiang, and J. M. Guo: IEEE Trans. Circuits Syst. Video Technol. **23** (2013) 671. https://doi.org/10.1109/TCSVT.2012.2211953
33 C. H. Hsia, J. M. Guo, and J. S. Chiang: IEEE Trans. Circuits Syst. Video Technol. **19** (2009) 1202. https://doi.org/10.1109/TCSVT.2009.2020259
34 C. H. Hsia: J. Internet Technol. **15** (2014) 1083. https://doi.org/10.6138/JIT.2014.15.7.01
35 P. Balakrishnan and K. Wahid: IEEE Canadian Conf. Electrical and Computer Engineering (2013) 1–4. https://doi.org/10.1109/CCECE.2013.6567727
36 B. Das and S. Banerjee: IEEE Proc. Circuits, Devices and Systems (2005) 17–24. https://doi.org/10.1049/ip-cds:20040817
37 M. M. Hasan and K. A. Wahid: IEEE Trans. Circuits Syst. II Express Briefs **64** (2017) 585. https://doi.org/10.1109/TCSII.2016.2584091