sensors

MDPI

Article

Digital Image Decoder for Efficient Hardware Implementation

Goran Savić [1,*], Milan Prokin [1], Vladimir Rajović [1] and Dragana Prokin [2]

[1] School of Electrical Engineering, University of Belgrade, 11000 Belgrade, Serbia
[2] The School of Electrical and Computer Engineering of Applied Studies, 11000 Belgrade, Serbia
* Correspondence: gsavic@etf.rs

Abstract: Increasing the resolution of digital images and the frame rate of video sequences leads to an increase in the amount of required logical and memory resources necessary for digital image and video decompression. Therefore, the development of new hardware architectures for digital image decoder with a reduced amount of utilized logical and memory resources become a necessity. In this paper, a digital image decoder for efficient hardware implementation, has been presented. Each block of the proposed digital image decoder has been described. Entropy decoder, decoding probability estimator, dequantizer and inverse subband transformer (parts of the digital image decoder) have been developed in such way which allows efficient hardware implementation with reduced amount of utilized logic and memory resources. It has been shown that proposed hardware realization of inverse subband transformer requires 20% lower memory capacity and uses less logic resources compared with the best state-of-the-art realizations. The proposed digital image decoder has been implemented in a low-cost FPGA device and it has been shown that it requires at least 32% less memory resources in comparison to the other state-of-the-art decoders which can process high-definition frame size. The proposed solution also requires effectively lower memory size than state-of-the-art architectures which process frame size or tile size smaller than high-definition size. The presented digital image decoder has maximum operating frequency comparable with the highest maximum operating frequencies among the state-of-the-art solutions.

Keywords: digital image decoder; efficient hardware implementation; image decompression

Citation: Savić, G.; Prokin, M.; Rajović, V.; Prokin, D. Digital Image Decoder for Efficient Hardware Implementation. *Sensors* **2022**, *22*, 9393. https://doi.org/10.3390/s22239393

Academic Editor: Kuo-Liang Chung

Received: 20 October 2022
Accepted: 24 November 2022
Published: 1 December 2022

1. Introduction

The development of new and improvement of existing techniques for the compression and decompression of digital images and videos is very topical today. There is a constant need to improve the quality and resolution of the digital image and the need to increase the frame rate and the duration of the video sequences. All of this results in an increase in the amount of required logical and memory resources for digital image and video processing and storage. Therefore, the improvement of existing techniques and the development of new techniques and hardware architectures for digital image and video compression and decompression, which will decrease the amount of required logical and memory resources, are the only answers to these challenges and many efforts are directed towards achieving that goal. Hardware implementation of 3-D DCT based image decoder with two algorithms to reduce the number of computations and the amount of utilized hardware resources has been presented in [1]. A flexible, line-based JPEG 2000 decoder with customizable level of parallelization without need to use external memory, has been described in [2]. FPGA implementation of a high-performance MPEG-4 simple profile video decoder, capable of parsing multiple bitstreams from different encoder sources has been proposed in [3]. Architecture design of an H.264/AVC decoder described in [4], allows efficient FPGA implementation. A flexible hardware JPEG 2000 decoder for digital cinema, presented in [5], intended for implementation in a single FPGA device, requires a reduced amount of logic and memory resources. A hardware JPEG 2000 decoder architecture based on the DCI specification, which can decode digital cinema frames without accessing any external

memory, supports the decoding process in accordance with the order of output images, with reduced storage resources for middle states and temporary image data, has been proposed in [6]. Design and implementation of an efficient memory video decoder with increased effective memory bandwidth has been presented in [7]. FPGA implementation of a full HD real-time high efficiency video coding main profile decoder, solving both real-time and power constraints, has been proposed in [8]. Hardware implementation of a full HD capable H.265/HEVC video decoder, presented in [9], targeted constraints related to hardware costs. Video decoder implemented on FPGAs using 3×3 and 2×2 networks-on-chip, with communication between the decoder modules performed via a network-on-chip, has been described in [10].

The block diagram of the state-of-the-art digital image decoder is shown in Figure 1. It consists of entropy decoder, decoding probability estimator, dequantizer and inverse subband transformer. The input compressed image is primarily received and processed by an entropy decoder which forwards its output data to the decoding probability estimator. The decoding probability estimator reconstructs the symbol probabilities within the specified contexts, sends them to the dequantizer and feeds them back to the entropy decoder. These data samples are processed by the dequantizer, which produces dequantized data samples in case of lossy compression or only forwards received data samples to the inverse subband transformer in case of lossless compression. Inverse subband transformer performs inverse filtering and composition of data samples received from the dequantizer and generates pixels of the output decompressed image at its output. As it has been shown in [11–13], the arithmetic coding ensures the highest compression ratio, which can theoretically remove all redundant information from the digital message. Arithmetic Q-coder has been presented in [14–17] and arithmetic Z-coder has been described in [18–21]. In the well-known JPEG 2000 still image compression standard, the MQ arithmetic coder is used, which is similar to QM-coder adopted in the original JPEG image compression standard described in [22–24]. The inverse process of the range encoding presented in [25], has been adopted as the basis of the decoding process implemented in the hardware realization proposed in this paper. The decoding process proposed in this paper is performed for every level of composition and every subband separately. Due to its high performance, uniform scalar quantizer with dead-zone [26] is used for quantization purposes very often. For that reason, it is adopted as a basis for the hardware realization of the dequantizer proposed in this paper.

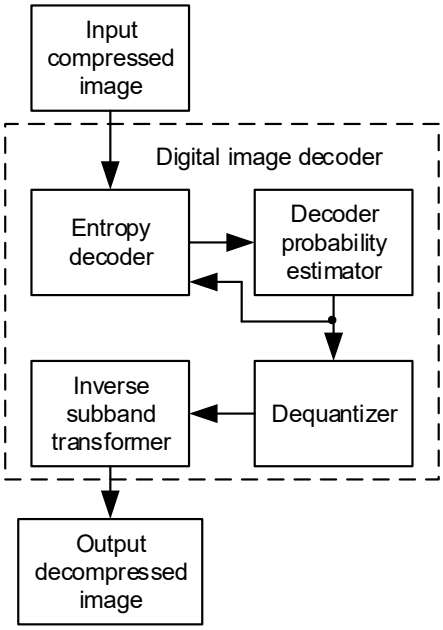

Figure 1. The block diagram of the state-of-the-art digital image decoder.

The inverse subband transformer within the proposed digital image decoder is based on two-dimensional (2-D) discrete wavelet transform (DWT) with Le Gall's 5/3 filters, which is also a part of the JPEG 2000 still image standard, due to its very good performances. The state-of-the-art hardware architectures of the 2-D DWT are mainly convolution-based or lifting-based. Convolution-based hardware architectures [27–32] are usually more complex and utilize larger amount of logic and memory resources. Lifting-based hardware architectures [33–38] are usually simpler, have lower computational complexity and utilize less amount of logic and memory resources. The most efficient hardware architectures of the 1-D DWT and 2-D DWT are described in [28] and [36–60]. The concept of the proposed 2-D DWT with 5/3 filters and its hardware architecture are presented in [61–63].

The digital image decoder for efficient hardware implementation presented in this paper has the same block diagram at the highest level of hierarchy as the state-of-the-art decoder shown in Figure 1. However, internal blocks of the proposed digital image decoder have been developed with intention to reduce the amount of utilized memory and logic resources and to optimize the hardware architecture of each internal block and to optimize the hardware architecture of the entire digital image decoder. Some initial research results, related to this topic, have been presented in [64].

This paper has the following structure: Section 2 describes the proposed entropy decoder and decoding probability estimator. The proposed dequantizer is presented in Section 3. Description of the proposed inverse subband transformer, based on two-dimensional (2-D) DWT, can be found in Section 4. Section 5 contains synthesis results of the hardware realization of the entire digital image decoder proposed in this paper. A brief conclusion is presented in Section 6.

2. Entropy Decoder and Decoding Probability Estimator

Hardware realization of entropy decoder and decoding probability estimator presented in this paper is based on the inverse process of the range encoding described in [25,65]. Decoding process is performed for every level of composition and every subband separately.

During the process of image compression, the samples of the components of the decomposed signal C (generated by direct subband transformer) had been split into magnitude M and sign S pairs:

$$M = |C|, \tag{1}$$

$$S = \begin{cases} 0, & C > 0 \\ 2, & C = 0 \\ 1, & C < 0 \end{cases}. \tag{2}$$

Magnitudes were then classified into magnitude-set indexes MS, which contained a group of magnitudes with similar values. A residual R had been defined as the difference between magnitude M and the lower limit of the sample of the component of the decomposed signal M_lower_limit:

$$R = M - M\_lower\_limit. \tag{3}$$

Magnitude-set indexes MS and the lower limits of the samples of the components of the decomposed signal M_lower_limit are determined based on Table 1.

In further process of image compression, MS, S and R had been separately encoded. In order to obtain a higher compression ratio, symbols had been defined based on contextual model which contained neighboring data samples, as shown in Figure 2. These contexts are also used in the process of decoding as a part of image decompression.

Table 1. Decoding the limit of the samples of the components of the decomposed signal.

Limits of the Samples (Inclusive)		Range of the Samples	*MS*
Lower	**Upper**		
0	0	1	0
1	1	1	1
2	2	1	2
3	3	1	3
4	5	2	4
6	7	2	5
8	11	4	6
12	15	4	7
16	23	8	8
24	31	8	9
32	47	16	10
48	63	16	11
64	95	32	12
96	127	32	13
128	191	64	14
192	255	64	15
256	383	128	16
384	511	128	17
512	767	256	18
768	1023	256	19
1024	1535	512	20
1536	2047	512	21
2048	3071	1024	22
3072	4095	1024	23
4096	6143	2048	24
6144	8191	2048	25
8192	12,287	4096	26
12,288	16,383	4096	27
16,384	24,575	8192	28
24,576	32,767	8192	29
32,768	49,151	16,384	30
49,152	65,535	16,384	31

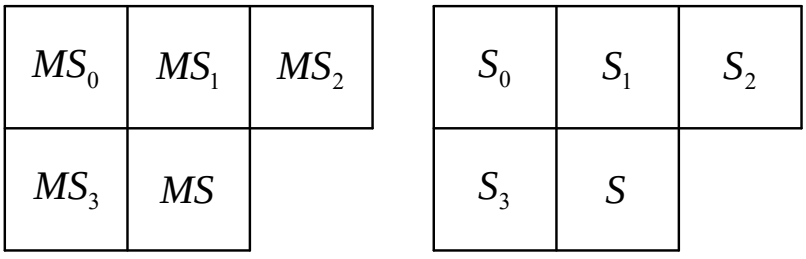

Figure 2. Neighboring magnitude-set indexes MS_i and signs S_i of already encoded data samples.

A flowchart of the entropy decoder and the decoding probability estimator, based on single-pass adaptive histograms with fast adaptation, is shown in Figure 3. The adaptation process starts from a uniform distribution and requires several data samples to complete. The adaptation time is proportional to the number of histogram bins and the difference between the uniform distribution and the exact distribution of the variable being decoded.

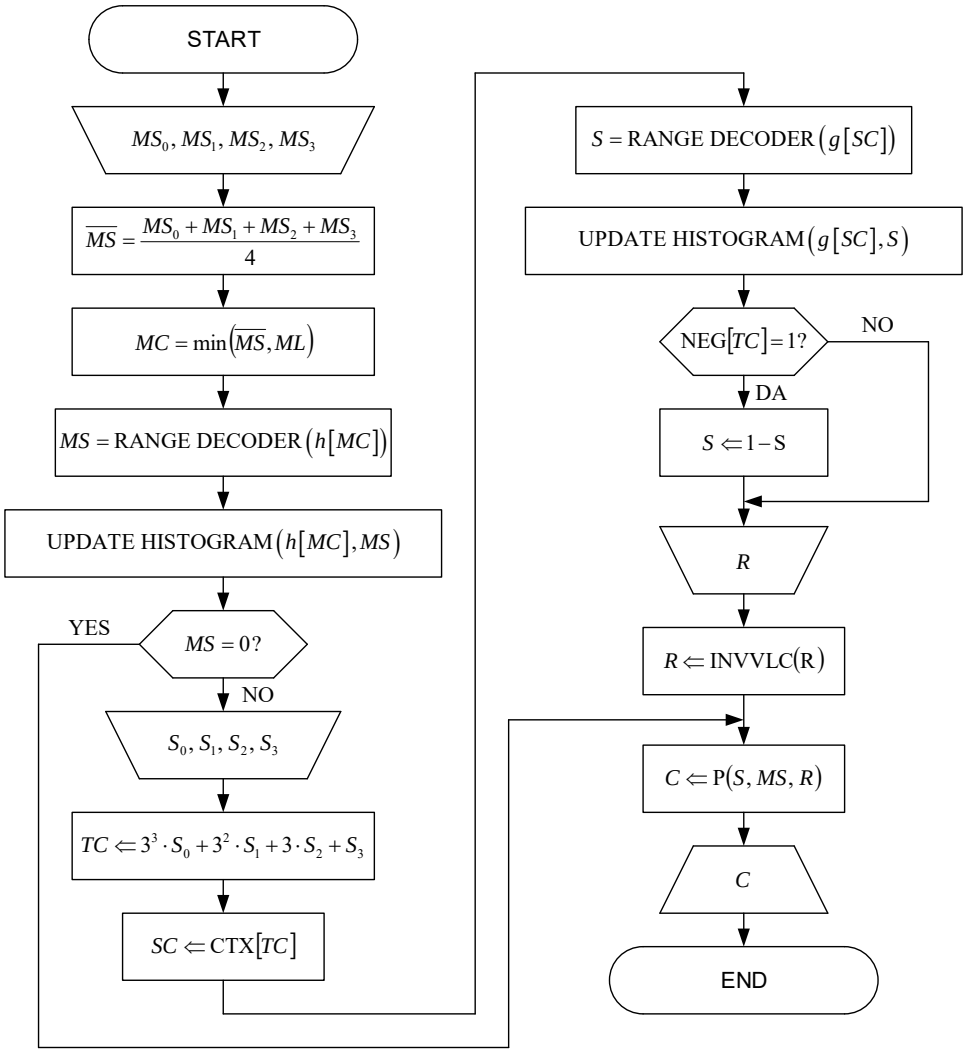

Figure 3. The flowchart of the entropy decoder and the decoding probability estimator.

First, the values of the neighborhood magnitude-set indexes MS_i (shown in Figure 2) of already encoded samples of the components of the decomposed signal are loaded and their mean value $\overline{MS}$ is calculated. Based on the calculated value $\overline{MS}$, the magnitude context MC,which represents the index of the appropriate adaptive magnitude histogram $h[MC]$, is determined, which is then used for the decoding of the magnitude-set index MS using the range decoder. The magnitude context MC is limited by a constant ML, with preferable value $ML = 4$, because the local variance can increase significantly near the sharp edges in the image, which would lead to a large number of histograms and their slow adaptation.

The number of magnitude histograms MH, i.e., the number of different magnitude contexts MC, is preferably limited to $MH = ML + 1 = 5$. After decoding the magnitude-set index MS, the magnitude histogram $h[MC]$ is updated.

In case of $MS = 0$, sign S is not decoded at all. In case of $MS \neq 0$, the neighborhood sign values S_i (shown in Figure 2) of already encoded samples of the components of the decomposed signal are loaded and then used for the decoding of a ternary context TC.

Based on the ternary context TC, the sign context SC is then determined using the CTX table represented as Table 2. The CTX table translates 81 different values of ternary contexts TC into a preferable number of five different values of sign context SC for each of the subbands, because a large number of different sign context SC values would lead to histograms that do not adapt at all, which also represents the number of sign histograms

SH. This very small number is justified by the fact that the more probable sign S is decoded, which is assured by appropriate examination of the sign and, if necessary, by inversion of the sign S using the NEG table represented as Table 3. Ternary contexts TC with $NS = \mathrm{NEG}[TC] = 0$ correspond to a higher probability of a positive sign $P(0)$ than a probability of a negative sign $P(1)$. Ternary contexts TC with $NS = \mathrm{NEG}[TC] = 1$ correspond to a higher probability of a negative sign $P(1)$ than the probability of a positive sign $P(0)$.

Table 2. CTX table for translating ternary context TC values into sign context SC.

S_0	S_1	S_2	S_3	TC	SC	S_0	S_1	S_2	S_3	TC	SC	S_0	S_1	S_2	S_3	TC	SC
0	0	0	0	0	0	1	0	0	0	27	2	2	0	0	0	54	1
0	0	0	1	1	0	1	0	0	1	28	1	2	0	0	1	55	0
0	0	0	2	2	0	1	0	0	2	29	2	2	0	0	2	56	0
0	0	1	0	3	4	1	0	1	0	30	0	2	0	1	0	57	1
0	0	1	1	4	1	1	0	1	1	31	1	2	0	1	1	58	0
0	0	1	2	5	2	1	0	1	2	32	1	2	0	1	2	59	2
0	0	2	0	6	1	1	0	2	0	33	1	2	0	2	0	60	0
0	0	2	1	7	0	1	0	2	1	34	0	2	0	2	1	61	0
0	0	2	2	8	1	1	0	2	2	35	0	2	0	2	2	62	0
0	1	0	0	9	0	1	0	0	0	36	1	2	0	0	0	63	0
0	1	0	1	10	1	1	1	0	1	37	1	2	1	0	1	64	0
0	1	0	2	11	1	1	1	0	2	38	1	2	1	0	2	65	0
0	1	1	0	12	1	1	1	1	0	39	0	2	1	1	0	66	0
0	1	1	1	13	0	1	1	1	1	40	1	2	1	1	1	67	1
0	1	1	2	14	0	1	1	1	2	41	0	2	1	1	2	68	1
0	1	2	0	15	0	1	1	2	0	42	1	2	1	2	0	69	0
0	1	2	1	16	4	1	1	2	1	43	0	2	1	2	1	70	0
0	1	2	2	17	0	1	1	2	2	44	0	2	1	2	2	71	3
0	2	0	0	18	4	1	2	0	0	45	3	2	2	0	0	72	0
0	2	0	1	19	2	1	2	0	1	46	0	2	2	0	1	73	4
0	2	0	2	20	2	1	2	0	2	47	1	2	2	0	2	74	3
0	2	1	0	21	1	1	2	1	0	48	1	2	2	1	0	75	2
0	2	1	1	22	0	1	2	1	1	49	2	2	2	1	1	76	2
0	2	1	2	23	3	1	2	1	2	50	2	2	2	1	2	77	0
0	2	2	0	24	0	1	2	2	0	51	0	2	2	2	0	78	2
0	2	2	1	25	0	1	2	2	1	52	0	2	2	2	1	79	1
0	2	2	2	26	1	1	2	2	2	53	1	2	2	2	2	80	0

The sign context SC represents the index of the appropriate adaptive sign histogram $g[SC]$ which is then used for decoding the sign S using a range decoder. After decoding the sign S, the sign histogram $g[SC]$ is updated.

After that, the encoded value of the residual is loaded and decoded using a decoder with a variable length code (INVVLC). Based on the already decoded values of the magnitude-set index MS, using Table 1 given for 16-bit values of the samples of the components of the decomposed signal, the lower limits of the samples of the components of the decomposed signal M_lower_limit are determined, which are then summed with the decoded value of the residual R, forming the decoded value of the magnitude M, as it is shown in Equation (4).

$$M = R + M\_lower\_limit. \tag{4}$$

Finally, at the very end of the decoding process, the decoded value of the samples of the components of the decomposed signal C is formed based on the already decoded magnitude M and sign S values.

The initialization flowchart for histograms with fast adaptation is shown in Figure 4. Each histogram bin corresponds to a single symbol x, which can be MS for a magnitude histogram or S for a sign histogram. State-of-the-art method for the probability $p(x)$

estimation of an occurrence of symbols x is based on the number $u(x)$ of occurrences of symbol x and the number of occurrences of all symbols $Total$.

Table 3. NEG table for inversion of the sign S.

TC	*P*(0)	*P*(1)	*NS*	*TC*	*P*(0)	*P*(1)	*NS*	*TC*	*P*(0)	*P*(1)	*NS*
0	0.5276	0.4724	0	27	0.6147	0.3853	0	54	0.4168	0.5832	1
1	0.5333	0.4667	0	28	0.4170	0.5830	1	55	0.5012	0.4988	0
2	0.4901	0.5099	1	29	0.6326	0.3674	0	56	0.5302	0.4698	0
3	0.2961	0.7039	1	30	0.4889	0.5111	1	57	0.5467	0.4533	0
4	0.4321	0.5679	1	31	0.4176	0.5824	1	58	0.5061	0.4939	0
5	0.6300	0.3700	0	32	0.4469	0.5531	1	59	0.4039	0.5961	1
6	0.4463	0.5537	1	33	0.5505	0.4495	0	60	0.5024	0.4976	0
7	0.4754	0.5246	1	34	0.5240	0.4760	0	61	0.4613	0.5387	1
8	0.4397	0.5603	1	35	0.4731	0.5269	1	62	0.4837	0.5163	1
9	0.5012	0.4988	0	36	0.4299	0.5701	1	63	0.5106	0.4894	0
10	0.5796	0.4204	0	37	0.5880	0.4120	0	64	0.5440	0.4560	0
11	0.4117	0.5883	1	38	0.5806	0.4194	0	65	0.5343	0.4657	0
12	0.5842	0.4158	0	39	0.4698	0.5302	1	66	0.4918	0.5082	1
13	0.5457	0.4543	0	40	0.4119	0.5881	1	67	0.4521	0.5479	1
14	0.5364	0.4636	0	41	0.5193	0.4807	0	68	0.5841	0.4159	0
15	0.5243	0.4757	0	42	0.4539	0.5461	1	69	0.5211	0.4789	0
16	0.7224	0.2776	0	43	0.4871	0.5129	1	70	0.4783	0.5217	1
17	0.5050	0.4950	0	44	0.4953	0.5047	1	71	0.6651	0.3349	0
18	0.7235	0.2765	0	45	0.3502	0.6498	1	72	0.4561	0.5439	1
19	0.3963	0.6037	1	46	0.4688	0.5312	1	73	0.6998	0.3002	0
20	0.6019	0.3981	0	47	0.5802	0.4198	0	74	0.6531	0.3469	0
21	0.4508	0.5492	1	48	0.4432	0.5568	1	75	0.6163	0.3837	0
22	0.5286	0.4714	0	49	0.3927	0.6073	1	76	0.5956	0.4044	0
23	0.6598	0.3402	0	50	0.6199	0.3801	0	77	0.5022	0.4978	0
24	0.4770	0.5230	1	51	0.5357	0.4643	0	78	0.6148	0.3852	0
25	0.5417	0.4583	0	52	0.4830	0.5170	1	79	0.4368	0.5632	1
26	0.4398	0.5602	1	53	0.4464	0.5536	1	80	0.5065	0.4935	0

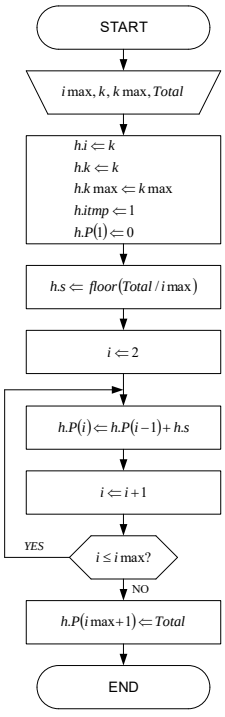

Figure 4. The initialization flowchart for histograms with fast adaptation.

Additionally, it is possible to define the cumulative probability $P(x)$ of all symbols y that precede the symbol x in the alphabet.

$$p(x) = \frac{u(x)}{Total}, \tag{5}$$

$$Total = \sum_{x} u(x), \tag{6}$$

$$P(x) = \sum_{y<x} p(y) = \frac{U(x)}{Total}, \tag{7}$$

$$U(x) = \sum_{y<x} u(y). \tag{8}$$

The main drawback of this simple method is that *Total* is an arbitrary integer, which means that division operation is necessary in order to calculate the probability $p(x)$. However, in the proposed hardware realization of the entropy decoder and decoder probability estimator, division operation is replaced by shift right operation for w bits, due to:

$$Total = 2^w. \tag{9}$$

Another drawback of this method is slow adaptation of the probability $p(x)$, due to averaging process. However, in the proposed hardware realization, the adaptation of the probability $p(x)$ is provided by low-pass filtering of the binary sequence $I(j)$ which represents the occurrence of a symbol x in a sequence y of symbols:

$$I(j) = \begin{cases} 1, \ y(j) = x \\ 0, \ y(j) \neq x \end{cases}. \tag{10}$$

The time response of mentioned low-pass filter is very important, since it is well-known that the bigger time constant of the low-pass filter provides more accurate steady-state estimation, while a smaller time constant provides faster estimation. This problem is especially pronounced at the beginning of the adaptation process, due to a lack of information. In order to avoid making a compromise in a fixed choice of a dominant pole of the low-pass filter, the variation of a dominant pole between minimum and maximum value is implemented.

According to the histogram initialization flowchart shown in Figure 4, the values of the variables are first loaded and, based on them, the variables within the histogram structure h are initialized. In that flowchart, the parameter i represents the histogram bin index, which can have values in the range from 1 to $i$max. The parameter $i$max represents the maximum value of the index i of the non-zero histogram, i.e., the total number of different symbols in the alphabet, which is preferably less than or equal to 32 for the magnitude histogram or equal to 2 for the sign histogram. The parameter $h.P()$ represents a string of cumulative probabilities:

$$h.P(i) = P(y|y < i) = \sum_{y<i} p(y). \tag{11}$$

The parameter $h.k$ is a reciprocal of an absolute dominant pole value of the low-pass filter. Variation of its value between $h.k$min and $h.k$max allows fast adaptation of the histogram after the start. The parameter $h.k$max represents the reciprocal value of the minimum absolute dominant pole of the low-pass filter and it is a fixed empirical parameter with preferable value less than *Total*. The parameter $h.k$min represents the reciprocal value of the maximum absolute dominant pole of the low-pass filter and it is a fixed parameter with preferable value $h.k\text{min} = 2$. The total number of symbols within the histogram increased by 1 is represented by the parameter $h.i$. Finally, the parameter $h.itmp$ represents the temporary value of the parameter $h.i$ before the parameter $h.k$ is changed.

After initializing the variables within the histogram structure h, in accordance with the flowchart shown in Figure 4, the step size $h.s$ is calculated, the index i is initialized and the histogram is initialized. This is followed by incrementing the index i and examining its value. The last step is the initialization of the last histogram bin.

Figure 5 shows an update flowchart for histogram with fast adaptation, based on the input of the symbol x and already described histogram structure h. Since the range decoder cannot operate with estimated zero probability $p(x) = 0$, even for symbols that do not occur at all, there is a need to modify the binary sequence $I(j)$. Another reason for modifying the binary sequence $I(j)$ is the fact that the modified probability $Mp(x) = Total \cdot p(x)$ is estimated using a fixed-point arithmetic. Adaptation of the probability $p(x)$ is performed by low-pass filtering of the modified binary sequence $MI(j)$ defined by Equation (12).

$$MI(j) = \begin{cases} Total - i\max, & y(j) = x \\ 1, & y(j) \neq x \end{cases}. \tag{12}$$

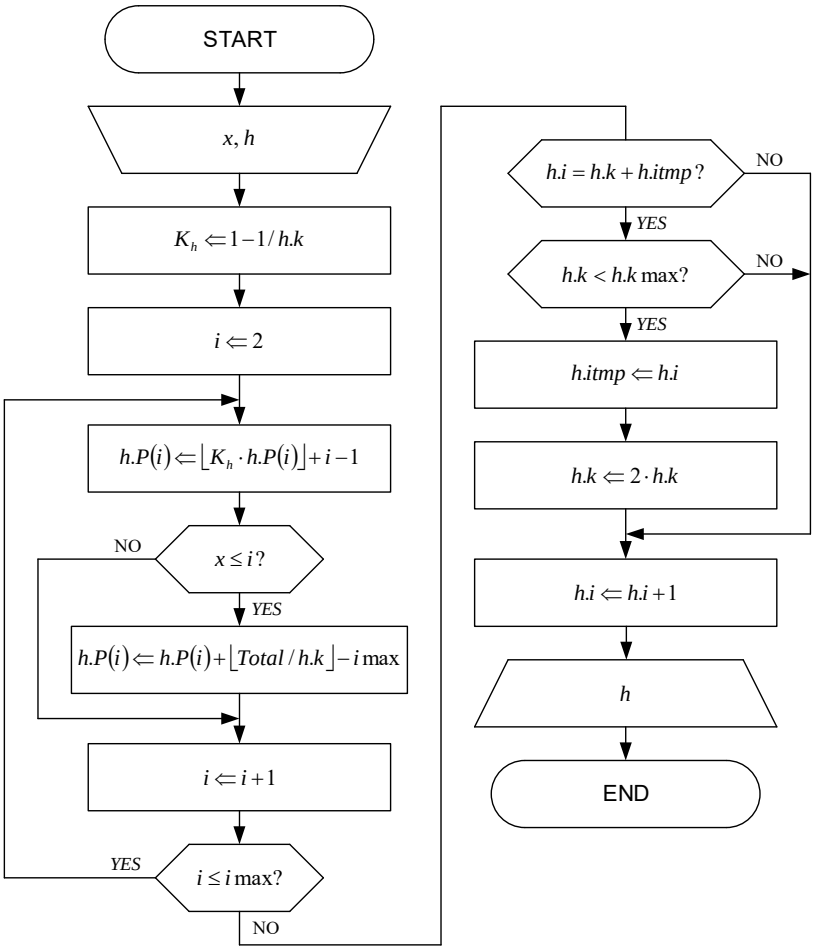

Figure 5. The update flowchart for histogram with fast adaptation.

The maximum probability $\max p(x)$ and the minimum probability $\min p(x)$ can be represented as:

$$\max p(x) = \frac{Total - i\max}{Total} < 1; \tag{13}$$

$$\min p(x) = \frac{1}{Total} > 0. \tag{14}$$

The preferable low-pass filter is the first order IIR filter in which the divide operation is avoided by keeping the parameter $h.k$ to be the power of two during its variation:

$$Mp(x) \Leftarrow Mp(x) \cdot \left(1 - \frac{1}{h.k}\right) + MI(j). \tag{15}$$

Instead of updating the modified probability $Mp(x)$, a modified cumulative probability $MP(x) = Total \cdot P(x)$ is updated, i.e., a string of cumulative probabilities $h.P()$ is updated. The constant K_h, which is used for the fast adaptation of histograms, and the histogram bin index i are initialized first. Then, $i-1$ is added to the cumulative probability $h.P(i)$ prescaled with a constant K_h, which is equivalent to adding one to a number $u(x)$. This is followed by an update of the cumulative probability $h.P(i)$, only for histograms with the index i greater than or equal to x, which is determined by the previous examination of the values of these parameters.

In the rest of the histogram update algorithm, the histogram is updated according to the following mathematical formulas:

$$h.k = \min\left\lfloor 2^{\lfloor \log_2 (h.i + h.k\min - 2) \rfloor}, h.k\max \right\rfloor; \tag{16}$$

$$h.k = \max(h.k, h.k\min). \tag{17}$$

where the preferable value $h.k\min = 2$, which is important for the first $h.k$ during the process of the fast adaptation.

The described method for the fast adaptation of histograms has significant advantages in comparison with state-of-the-art methods. Modifications of estimated probabilities are large at the beginning of the estimation process and much smaller later, which makes possible the detection of small local probability variations, which increases the compression ratio.

Figure 6 shows a flowchart of the state-of-the-art range decoder, which is together with the state-of-the-art range encoder described in [66–68]. Decoding is performed using a lookup table LUT (Equation (18)), which is compatible with Equations (19)–(22) for encoding symbol x (the symbol x had been encoded in the buffer of width $s = b^w$ in the form of a number i):

$$x = LUT\left(\frac{i+1}{s}\right); \tag{18}$$

$$i \in (\lfloor s \cdot P(x) \rfloor, \lfloor s \cdot (P(x) + p(x)) \rfloor); \tag{19}$$

$$\lfloor s \cdot P(x) \rfloor \leq i < \lfloor s \cdot (P(x) + p(x)) \rfloor; \tag{20}$$

$$s \cdot P(x) < i + 1 \leq s \cdot (P(x) + p(x)); \tag{21}$$

$$P(x) < \frac{i+1}{s} \leq P(x) + p(x). \tag{22}$$

In flowchart from Figure 6, following variables and constants are used:

- B = lower range limit;
- R = range;
- constant w_1 with preferable value 8;
- constant w_2 with preferable value 32;
- constant $TopValue = 1 << (w_2 - 1)$ with preferable value 40000000h;
- constant $BottomValue = TopValue >> w_1$ with preferable value 00400000h;
- constant $ExtraBits = (w_2 - 2)\%w_1 + 1$ with preferable value 4;
- constant $BottomLimit = (1 << w_1) - 1$ with preferable value 0FFh.

Operators <<, >>, %, | and &, used in that flowchart are borrowed from C/C++ programming language.

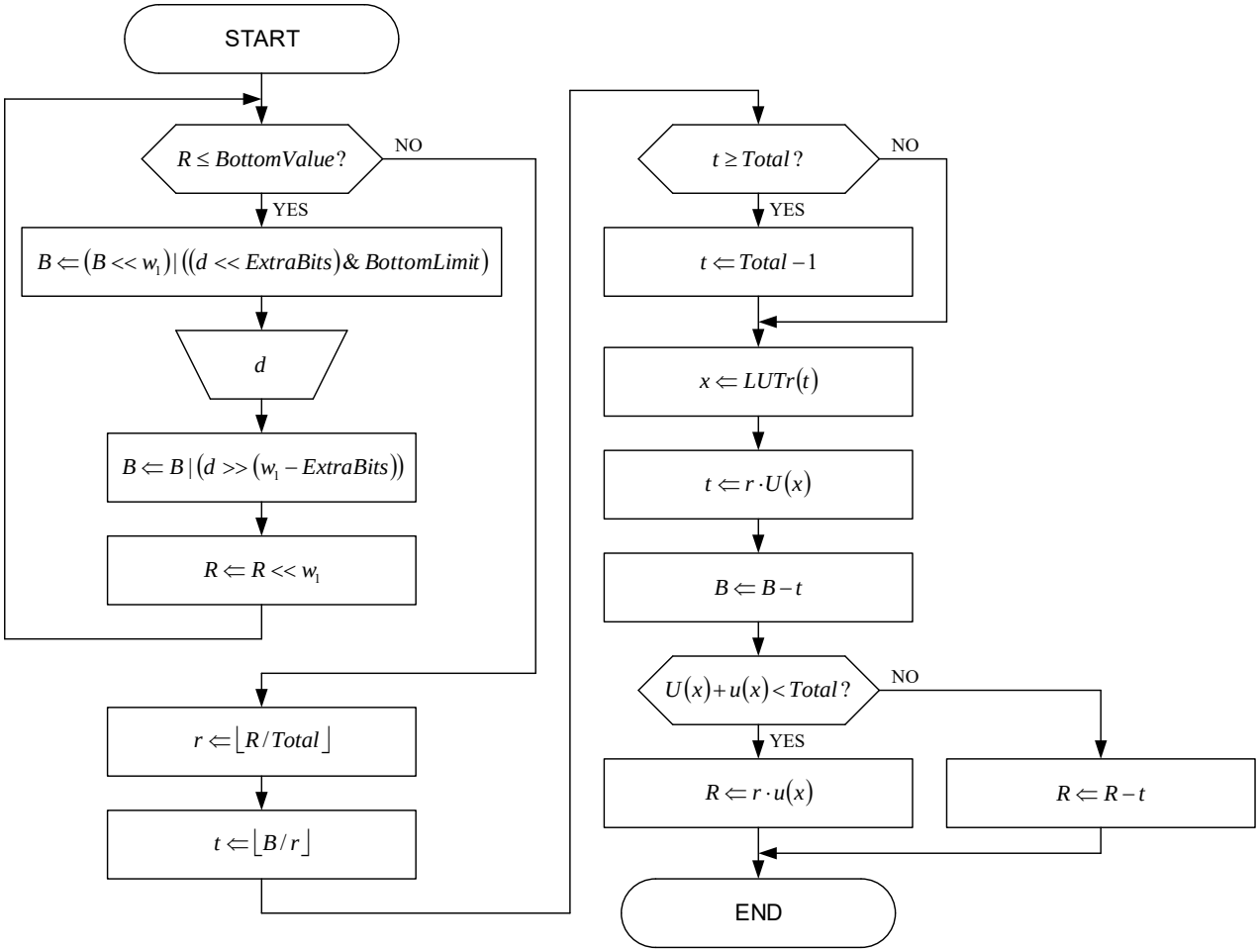

Figure 6. The flowchart of the state-of-the-art range decoder.

Floating point range decoder algorithm after the renormalization and without checking the boundary conditions is described with following equations:

$$t \Leftarrow B/R; \tag{23}$$

$$x \Leftarrow LUT(t); \tag{24}$$

$$t \Leftarrow R \cdot P(x); \tag{25}$$

$$B \Leftarrow B - t; \tag{26}$$

$$R \Leftarrow R \cdot p(x). \tag{27}$$

After introduction of the prescaled range r, the integer range decoder algorithm after the renormalization and without checking the boundary conditions becomes:

$$r \Leftarrow \left\lfloor \frac{R}{Total} \right\rfloor; \tag{28}$$

$$t \Leftarrow \left\lfloor \frac{B}{r} \right\rfloor; \tag{29}$$

$$x \Leftarrow LUTr(t); \tag{30}$$

$$t \Leftarrow r \cdot U(x); \tag{31}$$

$$B \Leftarrow B - t; \tag{32}$$

$$R \Leftarrow r \cdot u(x); \tag{33}$$

where:

$$LUTr(t \cdot Total) = LUTr\left(\frac{B}{r}\right) = LUT(t). \tag{34}$$

Digits of the symbol x in base b from the input buffer are input. First, the $2w_1 - ExtraBits$ bits are ignored according to the concept of extra bits. In this particular case, the first byte is a dummy one. Before start of the range decoding process, the following variables need to be initialized:

$$B = d >> (w_1 - ExtraBits); \tag{35}$$

$$R = 1 << ExtraBits. \tag{36}$$

The first part of the range decoding algorithm shown in Figure 6 performs renormalization before decoding, according to the initial examination block. Then, the appropriate bits are written into variable B and new symbol d is input in appropriate input block. After that, the variable B is updated using the appropriate shift operation and the variable R is updated by shifting.

The second part of the range decoding algorithm shown in Figure 6 updates the range. First, the prescaled range r for all symbols is updated using the first division operation. This is followed by deriving the cumulative number of occurrences t of the current symbol using the second division operation, and then limiting the value of t if corresponding condition is met. The next step is to find the appropriate symbol x based on the parameter t value and then to prescale the parameter t value. The parameter B value is updated, followed by the update of the parameter R value using the second multiplication operation with $u(x)$ for the current symbol x for all symbols except the last one. In the case of the last symbol, the parameter R value is updated using the subtraction operation. After the decoding of all data is completed, the final renormalization is performed.

In the state-of-the-art range decoder, the first division operation by $Total$ can be implemented with the shift right operation for w_3 bits in case when $Total = 2^{w_3}$, which is provided by the decoder probability estimator. However, the second division operation cannot be eliminated, which contributes to the increasing complexity of the decoder processor because a large number of existing digital signal processors do not support the division operation. Additionally, there are two multiplication operations per each symbol of the compressed image in the range decoder, which contributes to reducing the processing speed in general-purpose microprocessors. These drawbacks have been eliminated in the range decoder described in this paper.

Figure 7 shows the flowchart of the range decoder proposed in this paper without division operations and, optionally, without multiplication operations. The first division operation by $Total = 2^{w_3}$ (when calculating the parameter r value) is implemented by the shift right operation for w_3 bits, due to the fast adaptation of histograms described in this paper. The parameter r is then represented as $r = V \cdot 2^l$ and the first multiplication operation is implemented by multiplication with a small number V and shift left operation for l bits in order to calculate the value of the parameter t.

The second multiplication operation is performed when calculating the parameter R value by multiplying with a small number V and shift left operation for l bits. Both small number V multiplication operations are significantly simplified due to the small number of bits used to represent the number V. Furthermore, the multiplication with small, odd numbers, $V = 3$ or $V = 5$, can be implemented by the combination of shift and add operations, which completely eliminates the multiplication operations. The second division operation by r, when calculating the parameter t value, is implemented by the division operation with small number V and shift right operation for l bits. In this case, the division operation by constant small odd numbers $V = 3$, $V = 5$, $V = 9$, $V = 11$, $V = 13$ or $V = 15$ can be implemented with one multiplication operation and one shift right operation according to Table 4, as disclosed in [69,70]. Specially, the division operation by $V = 7$ is the most complex, because it requires the implementation of the addition operation of

049240249h and the addition operation with carry and 0h between the multiplication and shift right operations shown in Table 4.

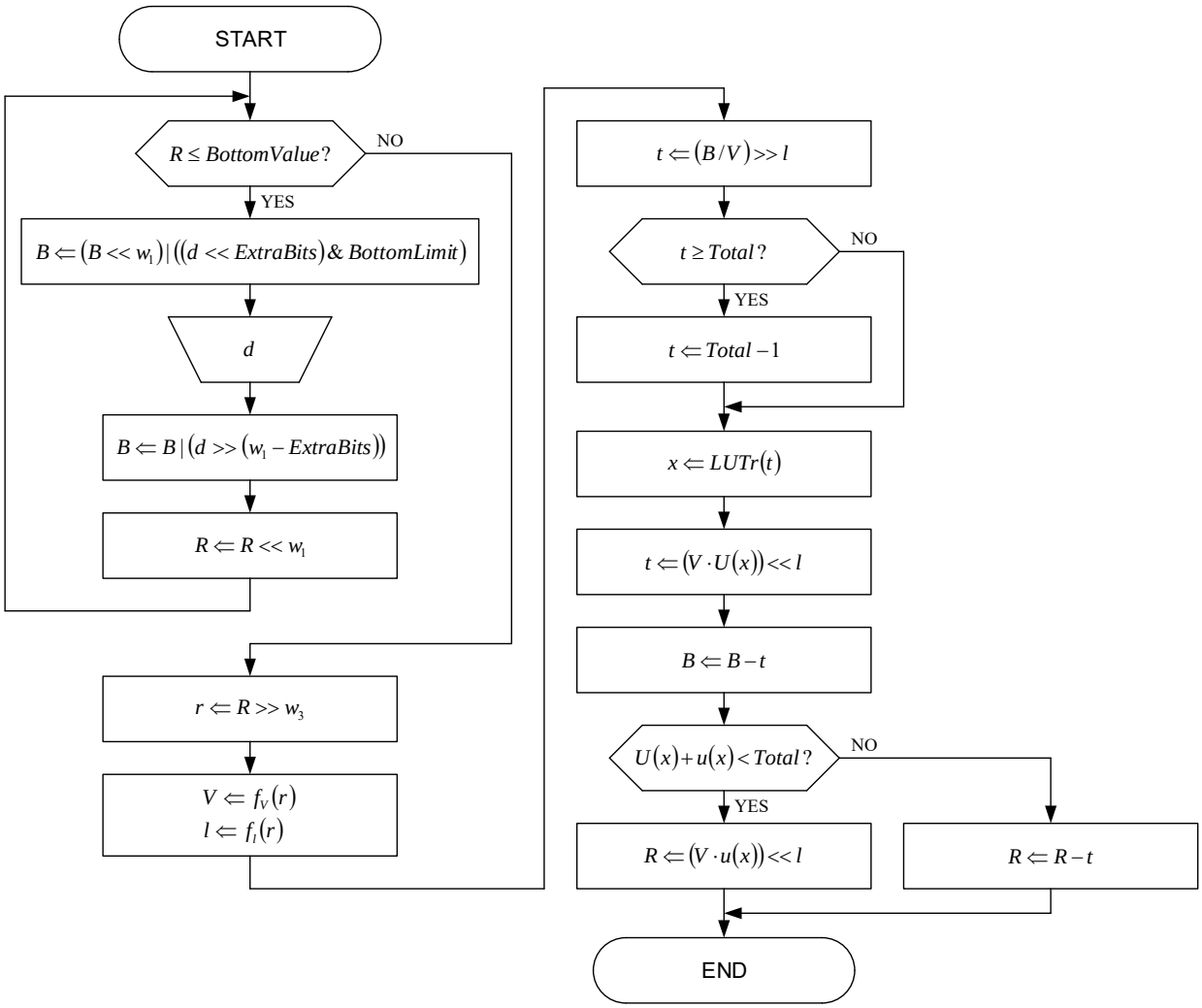

Figure 7. The flowchart of the proposed range decoder.

Table 4. Implementation of division operation with numbers 3, 5, 7, 9, 11, 13 and 15.

Divide by [Decimal Number]	Multiply by [Hexadecimal Number]	Right Shift for [Binary Digits]
3	0AAAAAAAB	1
5	0CCCCCCCD	2
7	049249249	1
9	038E38E39	1
11	0BA2E8BA3	3
13	04EC4EC4F	2
15	088888889	3

The approximations used in the implementation of multiplication or division operations in the proposed range decoder led to a smaller compression/decompression ratio. For example, by fixing $V = 1$, it is possible to completely eliminate all multiplication and division operations, but this also causes the largest approximation error and the largest decreasing of the compression/decompression ratio, but not more than 5%. On the other hand, if V is allowed to be $V = 1$ or $V = 3$, the compression/decompression ratio is decreased by less than 1%. Tables 5 and 6 show the difference in a number of multiplication

and division operations per decoded symbol between the state-of-the-art range decoder and the range decoder proposed in this paper. Although approximations, implemented in the proposed range decoder cause a negligible decrease of the compression/decompression ratio and, in contrast, they significantly reduce the hardware complexity of the realization.

Table 5. Number of multiplication and division operations per decoded symbol for $Total \neq 2^{w_3}$.

Operation Type	State-of-the-Art Range Decoder	Proposed Range Decoder $r=V\cdot 2^l$		
		V=1	V=3 V=5	$V\geq 7$
Multiply	2	0	1	3
Divide	2	1	1	1

Table 6. Number of multiplication and division operations per decoded symbol for $Total = 2^{w_3}$.

Operation Type	State-of-the-Art Range Decoder	Proposed Range Decoder $r=V\cdot 2^l$		
		V=1	V=3 V=5	$V\geq 7$
Multiply	2	0	1	3
Divide	1	0	0	0

3. Dequantizer

Dequantization is only performed in the case of lossy compression, while in the case of lossless compression data samples from the input of the dequantizer are simply routed to its output. Dequantizer proposed in this paper performs the process of dequantization for data samples which had been previously quantized with the uniform scalar quantizer with dead-zone, with quantization step Δ_b and dead-zone width $2\Delta_b$, as it is shown in Figure 8.

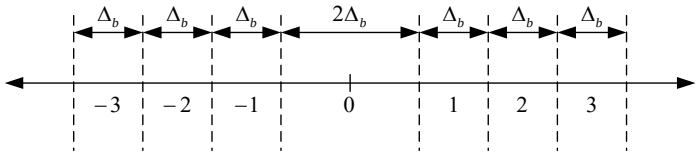

Figure 8. Illustration of the quantization process with uniform scalar quantizer with dead-zone.

Generally, each subband b (HH, HL, LH or LL) has its own quantization step Δ_b, calculated based on dynamic range of data samples which represent the components of the decomposed signal from subband b. This approach provides higher compression/decompression ratio. Equation (37) describes the quantization process with uniform scalar quantizer with dead-zone:

$$q_b = sign(y_b) \cdot \left\lfloor \frac{|y_b|}{\Delta_b} \right\rfloor \tag{37}$$

where y_b represents the component of the decomposed signal from subband b and q_b represents the resulted quantized value of data sample.

In order to avoid the division operation and to reduce the hardware complexity of the quantizer and dequantizer, for quantization steps for all four subbands from particular level of decomposition i, the values which represent the power of two are adopted:

$$\Delta_{LHi,HLi} = M \cdot 2^{E-i},\ \Delta_{HHi} = M \cdot 2^{E-i+1},\ \Delta_{LLi} = M \cdot 2^{E-i-1} \tag{38}$$

where M represents the mantissa (integer from the range $64 \leq M \leq 127$) and E represents the exponent (integer from the range $-6 \leq E \leq 6$).

Dequantized absolute values of data samples which represent the components of the decomposed signal from subbands HH, HL, LH or LL, at level i of composition, are calculated according to the following equations:

$$\left|y_{LHi\_deq}\right| = |q_{LHi}| \cdot M \cdot 2^{E-i} + M \cdot 2^{E-i-1}; \tag{39}$$

$$\left|y_{HLi\_deq}\right| = |q_{HLi}| \cdot M \cdot 2^{E-i} + M \cdot 2^{E-i-1}; \tag{40}$$

$$\left|y_{HHi\_deq}\right| = |q_{HHi}| \cdot M \cdot 2^{E-i+1} + M \cdot 2^{E-i}; \tag{41}$$

$$\left|y_{LLi\_deq}\right| = |q_{LLi}| \cdot M \cdot 2^{E-i-1}. \tag{42}$$

The hardware complexity of the dequantizer proposed in this paper is significantly reduced, since the multiplication operation by power of two is implemented by using permanently shifted hardware connections between input and output bit lines, and due to multiplication with narrow-range integer M, which is implemented by a simple lookup table.

4. Inverse Subband Transformer

Inverse subband transformer is an important part of digital image decoder from the aspect of memory resources utilization. Optimal realization of inverse subband transformer can make important contribution to reducing the capacity of used memory and the neglecting the importance of inverse subband transformer optimization could lead to a significant increase in the amount of utilized memory resources.

The proposed hardware realization of the inverse subband transformer is based on the 2-D DWT with 5/3 filters. Equation (43) describes one-dimensional (1-D) inverse low-pass Le Gall's 5/3 filter, while Equation (44) describes 1-D inverse high-pass Le Gall's 5/3 filter:

$$w_0[n] = \frac{1}{2}y_0[n-1] + y_0[n-2] + \frac{1}{2}y_0[n-3]; \tag{43}$$

$$w_1[n] = -\frac{1}{8}y_1[n] - \frac{1}{4}y_1[n-1] + \frac{3}{4}y_1[n-2] - \frac{1}{4}y_1[n-3] - \frac{1}{8}y_1[n-4]. \tag{44}$$

The basic building block utilized for 2-D DWT filtering is non-stationary hardware realization of the 1-D inverse 5/3 filter shown in Figure 9.

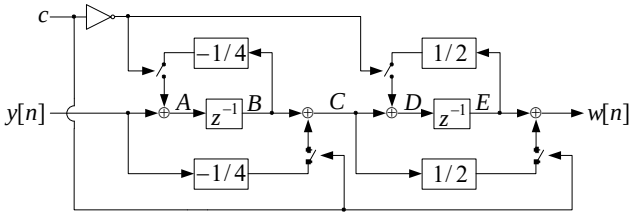

Figure 9. Non-stationary hardware realization of the 1-D inverse 5/3 filter.

The control signal c controls four switches, providing two different topologies of the filter: one topology for input data samples $y[n]$ with even indexes $n = 2p$ and another topology for input data samples $y[n]$ with odd indexes $n = 2p + 1$. The control signal c is at low level ($c = 0$) for every input data sample $y[n]$ with even index n when two upper switches are closed, while two lower switches are opened. Control signal c is at high level ($c = 1$) for every input data sample $y[n]$ with odd index n when two upper switches are opened, while two lower switches are closed. The time diagram of control signal c in the proposed 1-D inverse 5/3 filter is shown in Figure 10.

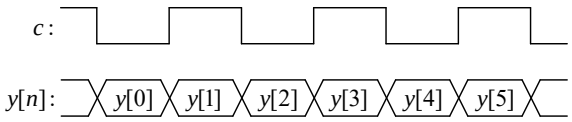

Figure 10. The time diagram of control signal c in the proposed 1-D inverse 5/3 filter.

The proposed 1-D inverse DWT 5/3 filter provides output data samples for even indexes $n = 2p$ and odd indexes $n = 2p + 1$ in an interleaved fashion, as shown in Figure 11.

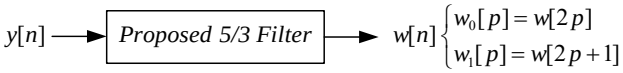

Figure 11. Block diagram of the proposed 1-D inverse 5/3 filter.

Hardware realization of 1-D inverse 5/3 filter from Figure 9 has been implemented on EP4CE115F29C7 FPGA device from Altera Cyclone IVE family [71]. The synthesis results for the proposed non-stationary filter realization and state-of-the-art convolution-based and lifting-based realizations (implemented on the same FPGA device), obtained using Altera Quartus II 10.0 software, are presented in Table 7.

Table 7. FPGA synthesis results of various implementations of 5/3 filter on Altera FPGA EP4CE115F29C7.

1-D Inverse DWT 5/3 Filter @ 85 °C Unrestricted Frequency	**Convolution [27–32]**	**Lifting [33–38]**	**Proposed**
Total logic elements	234	120	120
Total registers	139	72	48
Critical path delay [ns]	5.4	8.2	5
Max frequency [MHz]	197.7	128	212
Total power dissipation [mW] @ 80MHz	132.4	134.4	130.9

It can be seen that hardware implementation of the proposed non-stationary 1-D inverse 5/3 filter utilizes the lowest number of total logic elements and registers, has the shortest critical path delay, allows the highest maximum operating frequency and has the lowest total power dissipation in comparison with state-of-the-art realizations.

The block diagram of the proposed 2-D inverse DWT 5/3 architecture, with $J = 7$ levels of composition, is shown in Figure 12. The input data samples are the components of the decomposed signal $z_{HH}^{(j)}[m,n]$, $z_{HL}^{(j)}[m,n]$ and $z_{LH}^{(j)}[m,n]$ from level j (j=1, 2, . . . , 7) of composition and the components of the decomposed signal $z_{LL}^{(7)}[m,n]$ from level 7 of composition. The subband LL represents the data samples produced as the result of forward low-pass filtering over rows and forward low-pass filtering over columns within the direct subband transformer, which is a part of a digital image encoder. The subband HL represents the data samples produced as the result of forward low-pass filtering over rows and forward high-pass filtering over columns. The subband LH represents the data samples produced as the result of forward high-pass filtering over rows and forward low-pass filtering over columns. Finally, the subband HH represents the data samples produced as the result of forward high-pass filtering over rows and forward high-pass filtering over columns.

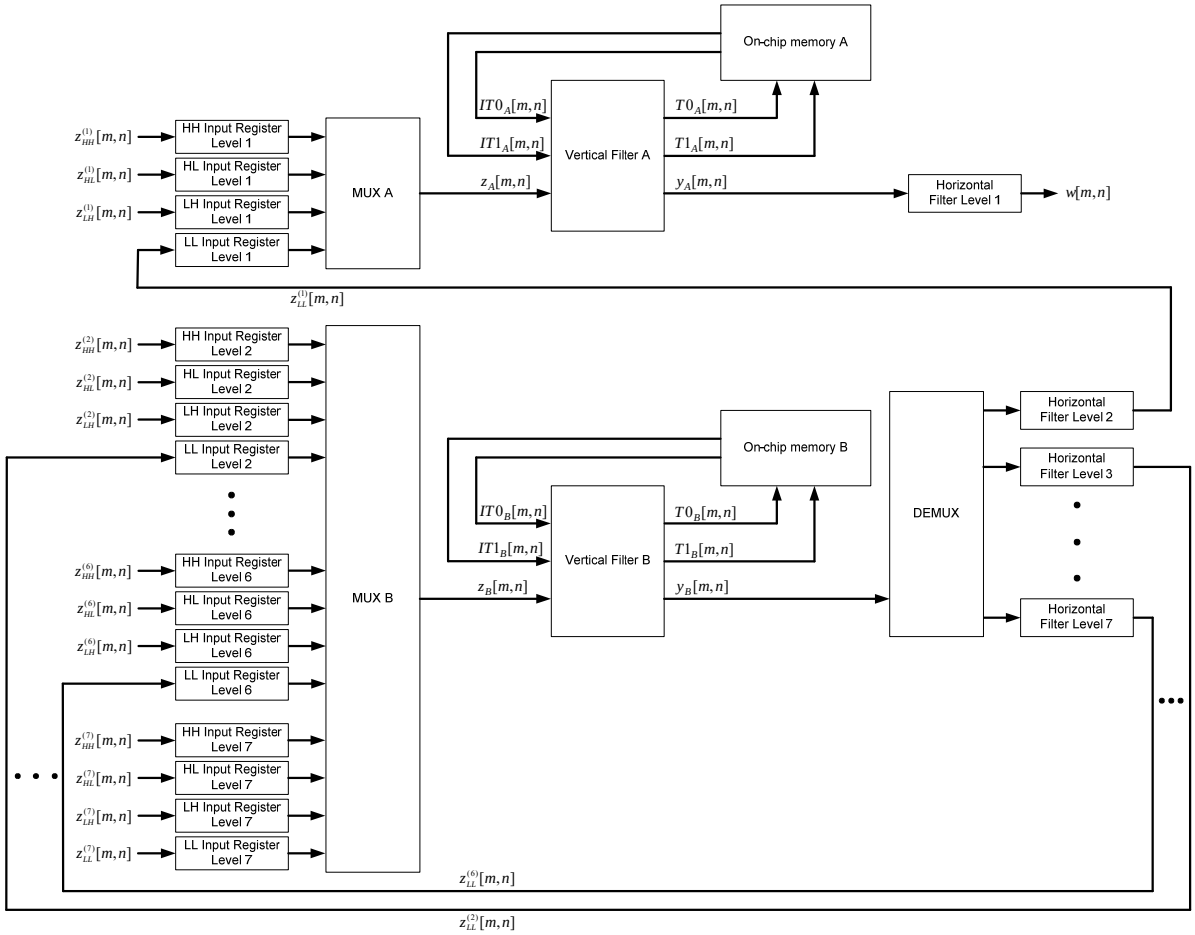

Figure 12. The block diagram of the proposed 2-D inverse DWT 5/3 architecture.

The input data samples from level 1 of composition are routed through a multiplexer "MUX A" generating data samples $z_A[m,n]$ shown in Equation (45), then vertically filtered by "Vertical Filter A", producing the data samples $y_A[m,n]$ shown in Equation (46), which are then horizontally filtered by "Horizontal Filter Level 1" generating the pixels of the reconstructed image $w[m,n]$. The sequence of data samples $y_A[m,n]$ contains high-pass ($y_H^{(1)}[m,k]$) and low-pass ($y_L^{(1)}[m,k]$) data components at level 1 which are to be horizontally filtered.

$$z_A[m,n] = \begin{cases} z_{LH}^{(1)}[m,k],\ for\, m = 2l\ and\ n = 2k \\ z_{LL}^{(1)}[m,k],\ for\, m = 2l\ and\ n = 2k+1 \\ z_{HH}^{(1)}[m,k],\ for\, m = 2l+1\ and\ n = 2k \\ z_{HL}^{(1)}[m,k],\ for\, m = 2l+1\ and\ n = 2k+1 \end{cases}; \tag{45}$$

$$y_A[m,n] = \begin{cases} y_H^{(1)}[m,k],\ for\, n = 2k \\ y_L^{(1)}[m,k],\ for\, n = 2k+1 \end{cases}. \tag{46}$$

The input data samples from level j (j = 2,3, . . . ,7) of composition are routed through a multiplexer "MUX B" generating data samples $z_B[m,n]$ shown in Equation (47), and then vertically filtered by "Vertical Filter B", producing the data samples $y_B[m,n]$ shown in Equation (48), which are then horizontally filtered by "Horizontal Filter Level j" generating the components of the decomposed signal $z_{LL}^{(j-1)}[m,n]$ (j = 2,3, . . . ,7), which are later used for inverse filtering at level $j-1$. The sequence of data samples $y_B[m,n]$ contains high-pass

($y_H^{(j)}[m,k]$) and low-pass ($y_L^{(j)}[m,k]$) data components at level j (j = 2,3, ... ,7) which are to be horizontally filtered.

$$z_B[m,n] = \begin{cases} z_{LH}^{(j)}[m,k],\ for\ m = 2l\ and\ n = 2k \\ z_{LL}^{(j)}[m,k],\ for\ m = 2l\ and\ n = 2k+1 \\ z_{HH}^{(j)}[m,k],\ for\ m = 2l+1\ and\ n = 2k \\ z_{HL}^{(j)}[m,k],\ for\ m = 2l+1\ and\ n = 2k+1 \end{cases}; \tag{47}$$

$$y_B[m,n] = \begin{cases} y_H^{(j)}[m,k],\ for\ n = 2k \\ y_L^{(j)}[m,k],\ for\ n = 2k+1 \end{cases}. \tag{48}$$

The time diagram of the 2-D inverse DWT 5/3 filtering at the beginning of even lines (starting from 0) for the first three levels of composition is shown in Figure 13. This pattern continues until the end of the even lines, and time diagram of the 2-D inverse DWT 5/3 filtering at the end of even lines for the first three levels of composition can be seen in Figure 14.

clk	VF Input Level 3	HF Input Level 3	VF Input Level 2	HF Input Level 2	VF Input Level 1	HF Input Level 1	Output Pixels
0	$z_{LH}^{(3)}[m^{(3)},0]$	$y_H^{(3)}[m^{(3)},0]$					
1	$z_{LL}^{(3)}[m^{(3)},0]$	$y_L^{(3)}[m^{(3)},0]$					
2	$z_{LH}^{(3)}[m^{(3)},1]$	$y_H^{(3)}[m^{(3)},1]$					
3	$z_{LL}^{(3)}[m^{(3)},1]$	$y_L^{(3)}[m^{(3)},1]$					
4	$z_{LH}^{(3)}[m^{(3)},2]$	$y_H^{(3)}[m^{(3)},2]$					
5			$z_{LH}^{(2)}[m^{(2)},0]$	$y_H^{(2)}[m^{(2)},0]$			
6			$z_{LL}^{(2)}[m^{(2)},0]$	$y_L^{(2)}[m^{(2)},0]$			
7			$z_{LH}^{(2)}[m^{(2)},1]$	$y_H^{(2)}[m^{(2)},1]$			
8			$z_{LL}^{(2)}[m^{(2)},1]$	$y_L^{(2)}[m^{(2)},1]$	$z_{LH}^{(1)}[m^{(1)},0]$	$y_H^{(1)}[m^{(1)},0]$	
9			$z_{LH}^{(2)}[m^{(2)},2]$	$y_H^{(2)}[m^{(2)},2]$	$z_{LL}^{(1)}[m^{(1)},0]$	$y_L^{(1)}[m^{(1)},0]$	
10	$z_{LL}^{(3)}[m^{(3)},2]$	$y_L^{(3)}[m^{(3)},2]$			$z_{LH}^{(1)}[m^{(1)},1]$	$y_H^{(1)}[m^{(1)},1]$	
11			$z_{LL}^{(2)}[m^{(2)},2]$	$y_L^{(2)}[m^{(2)},2]$	$z_{LL}^{(1)}[m^{(1)},1]$	$y_L^{(1)}[m^{(1)},1]$	$w[m,0]$
12					$z_{LH}^{(1)}[m^{(1)},2]$	$y_H^{(1)}[m^{(1)},2]$	$w[m,1]$
13			$z_{LH}^{(2)}[m^{(2)},3]$	$y_H^{(2)}[m^{(2)},3]$	$z_{LL}^{(1)}[m^{(1)},2]$	$y_L^{(1)}[m^{(1)},2]$	$w[m,2]$
14	$z_{LH}^{(3)}[m^{(3)},3]$	$y_H^{(3)}[m^{(3)},3]$			$z_{LH}^{(1)}[m^{(1)},3]$	$y_H^{(1)}[m^{(1)},3]$	$w[m,3]$
15			$z_{LL}^{(2)}[m^{(2)},3]$	$y_L^{(2)}[m^{(2)},3]$	$z_{LL}^{(1)}[m^{(1)},3]$	$y_L^{(1)}[m^{(1)},3]$	$w[m,4]$
16					$z_{LH}^{(1)}[m^{(1)},4]$	$y_H^{(1)}[m^{(1)},4]$	$w[m,5]$
17			$z_{LH}^{(2)}[m^{(2)},4]$	$y_H^{(2)}[m^{(2)},4]$	$z_{LL}^{(1)}[m^{(1)},4]$	$y_L^{(1)}[m^{(1)},4]$	$w[m,6]$
18	$z_{LL}^{(3)}[m^{(3)},3]$	$y_L^{(3)}[m^{(3)},3]$			$z_{LH}^{(1)}[m^{(1)},5]$	$y_H^{(1)}[m^{(1)},5]$	$w[m,7]$
19			$z_{LL}^{(2)}[m^{(2)},4]$	$y_L^{(2)}[m^{(2)},4]$	$z_{LL}^{(1)}[m^{(1)},5]$	$y_L^{(1)}[m^{(1)},5]$	$w[m,8]$
20					$z_{LH}^{(1)}[m^{(1)},6]$	$y_H^{(1)}[m^{(1)},6]$	$w[m,9]$
21			$z_{LH}^{(2)}[m^{(2)},5]$	$y_H^{(2)}[m^{(2)},5]$	$z_{LL}^{(1)}[m^{(1)},6]$	$y_L^{(1)}[m^{(1)},6]$	$w[m,10]$
22					$z_{LH}^{(1)}[m^{(1)},7]$	$y_H^{(1)}[m^{(1)},7]$	$w[m,11]$
23			$z_{LL}^{(2)}[m^{(2)},5]$	$y_L^{(2)}[m^{(2)},5]$	$z_{LL}^{(1)}[m^{(1)},7]$	$y_L^{(1)}[m^{(1)},7]$	$w[m,12]$
					$z_{LH}^{(1)}[m^{(1)},8]$	$y_H^{(1)}[m^{(1)},8]$	$w[m,13]$

Figure 13. The time diagram of the 2-D inverse DWT 5/3 filtering at the beginning of even lines.

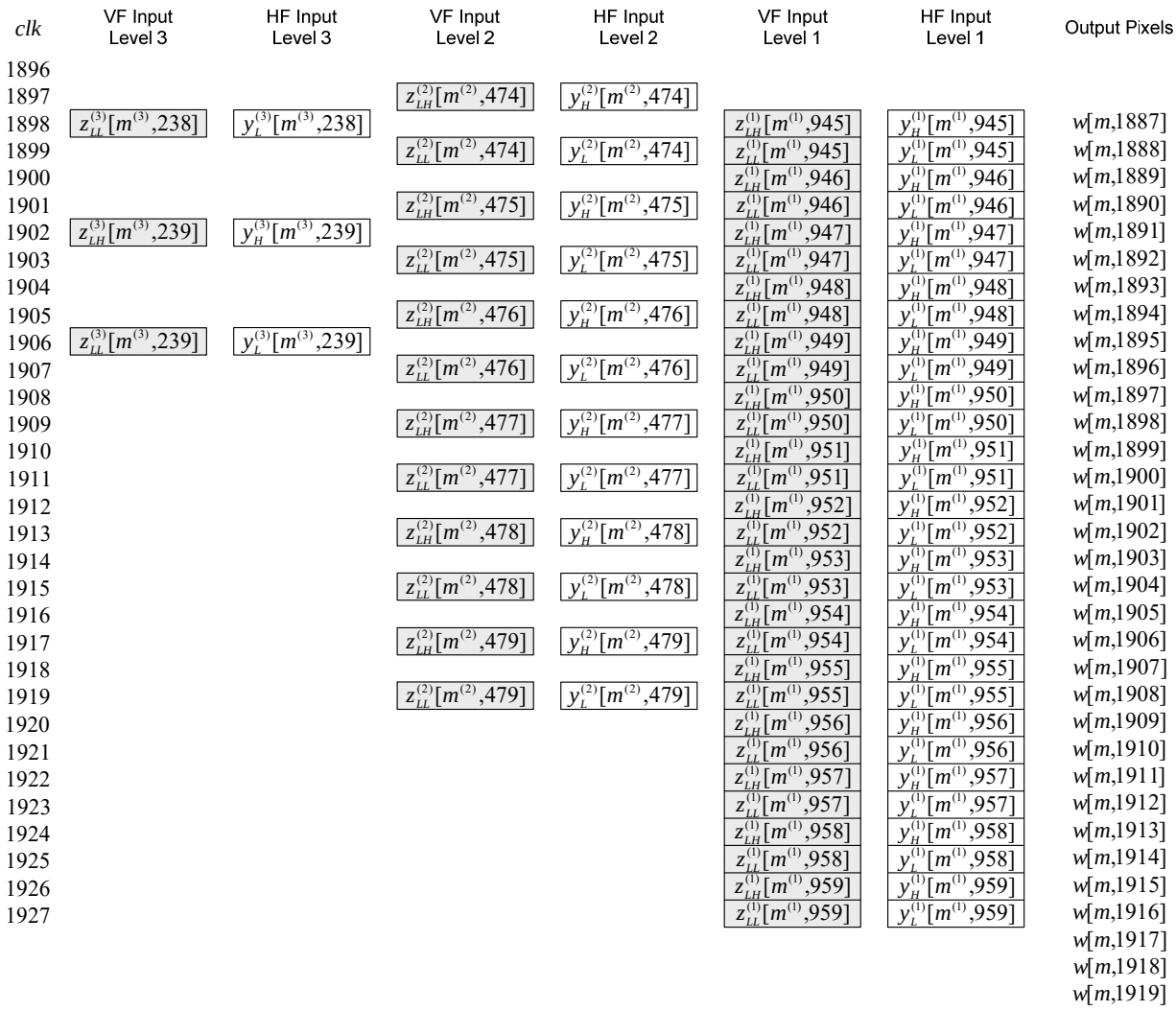

Figure 14. The time diagram of the 2-D inverse DWT 5/3 filtering at the end of even lines.

The time diagram of the 2-D inverse DWT 5/3 filtering of the odd lines is almost the same as for the even lines. There are only few differences: every even signal component (starting from 0) at the input of vertical filter belongs to the subband HH ($z_{HH}^{(1)}[m^{(1)}, n^{(1)}]$), every odd signal component at the input of vertical filter belongs to the subband HL ($z_{HL}^{(1)}[m^{(1)}, n^{(1)}]$) and the first level of composition is also the only level of composition, because the signal components from the subbands HH and HL are not generated based on the signal components from the previous levels of composition.

The time diagram of the beginning and the time diagram of the end of the line-wise inverse filtering for the high-definition resolution image are shown in Figures 15 and 16, respectively. Notation "$z_{LH}^{(j)}[m^{(j)}, n^{(j)}]$, $z_{LL}^{(j)}[m^{(j)}, n^{(j)}]$", for even lines (starting from 0) at level j, represents the following sequence of signal components: $z_{LH}^{(j)}[m^{(j)}, 0]$, $z_{LL}^{(j)}[m^{(j)}, 0]$, $z_{LH}^{(j)}[m^{(j)}, 1]$, $z_{LL}^{(j)}[m^{(j)}, 1]$, $z_{LH}^{(j)}[m^{(j)}, 2]$, $z_{LL}^{(j)}[m^{(j)}, 2]$, etc. Notation "$z_{HH}^{(j)}[m^{(j)}, n^{(j)}]$, $z_{HL}^{(j)}[m^{(j)}, n^{(j)}]$", for odd lines at level j, represents the following sequence of signal components: $z_{HH}^{(j)}[m^{(j)}, 0]$, $z_{HL}^{(j)}[m^{(j)}, 0]$, $z_{HH}^{(j)}[m^{(j)}, 1]$, $z_{HL}^{(j)}[m^{(j)}, 1]$, $z_{HH}^{(j)}[m^{(j)}, 2]$, $z_{HL}^{(j)}[m^{(j)}, 2]$, etc. All these signal components are vertically and then horizontally filtered by appropriate inverse filters.

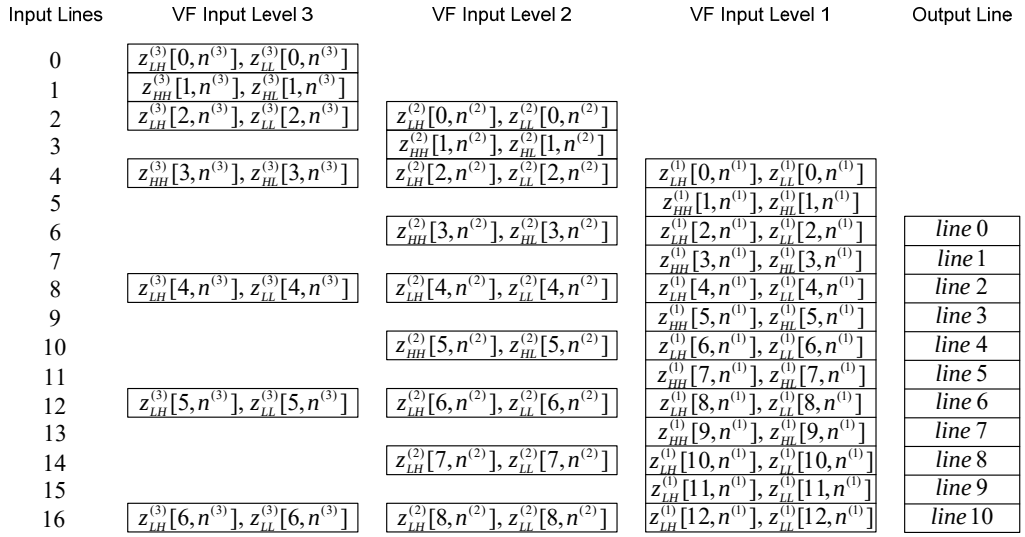

Figure 15. The time diagram of the beginning of the line-wise filtering.

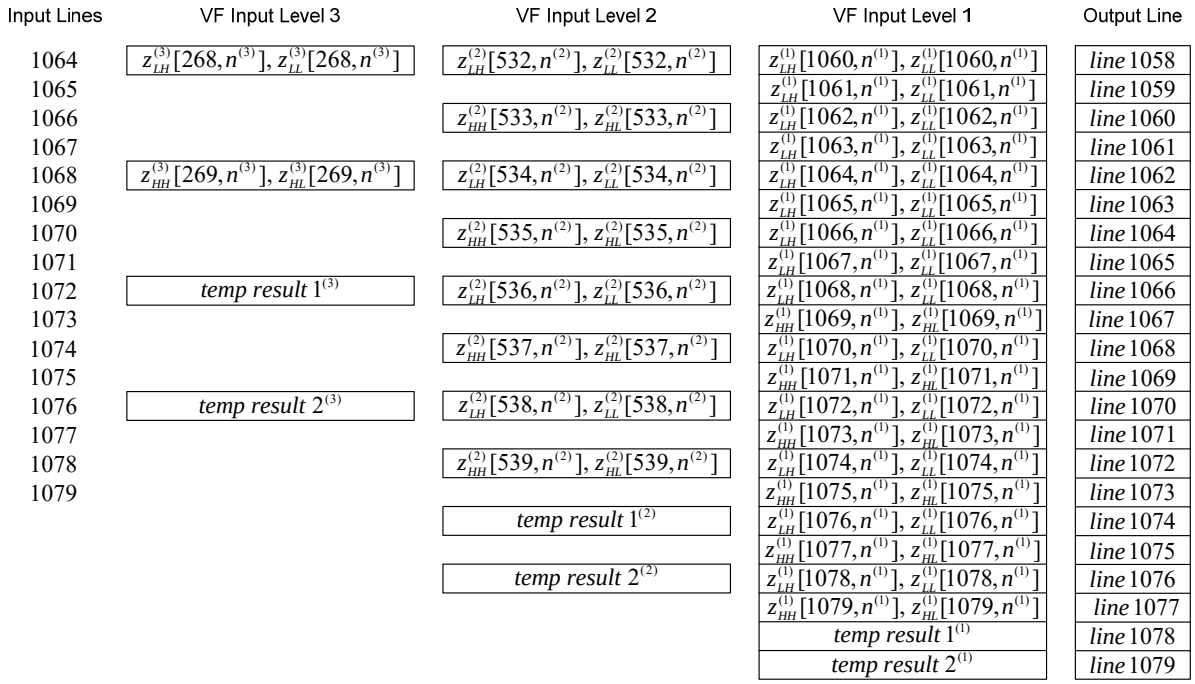

Figure 16. The time diagram of the end of the line-wise filtering.

The internal intermediate results 'temp result 1' and 'temp result 2' at the current level of composition are used for generating the last two lines of resulting signal components from the next level of composition.

In order to ensure the proper inverse 2-D DWT 5/3 filtering of $N \times N$ image, two lines of intermediate results have to be stored into on-chip memory (shown in Figure 17) at each level of composition. The intermediate results from level 1 of composition are stored into "On-chip memory A", which contains one FIFO buffer with capacity of $2N$ data samples, while the intermediate results from all other levels of composition are stored into "On-chip memory B", which contains six FIFO buffers (in case of $J = 7$ levels of composition) with capacity halved at every succeeding level, starting from capacity of N data samples at level

2. The total on-chip memory capacity needed for $N \times N$ image filtering with J levels of composition is:

$$2N + N + \frac{N}{2} + \ldots + \frac{N}{2^{J-2}} = 4N\left(1 - 2^{-J}\right). \tag{49}$$

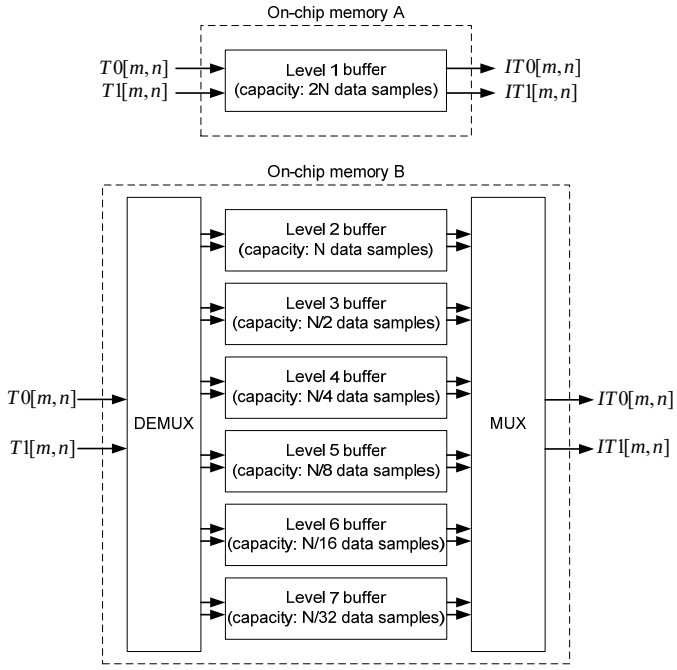

Figure 17. On-chip memory.

Due to the very low capacity of required memory, the proposed inverse 2-D DWT architecture does not require off-chip memory at all. The comparison between the proposed architecture and the best state-of-the-art 2-D inverse 5/3 DWT architectures so far published in the literature, in terms of required capacity of on-chip and off-chip memory, is presented in Table 8. It can be concluded that the proposed 2-D inverse 5/3 DWT architecture, for $N \times N$ image and $J \to \infty$ levels of composition, requires the total memory capacity of $4N$ data samples, which is 20% lower capacity compared with the best state-of-the-art architecture.

Table 8. Comparison of various 2-D inverse 5/3 DWT architectures.

Architecture	On-Chip Memory Capacity	Off-Chip Memory Capacity
Non-separable [37]	$10N$	0
SIMD [37]	N^2	0
Direct [38]	N^2	0
Systolic-parallel [38]	$14N$	0
[28]	$5N$	$N^2/4$
[40]	$7N(1-2^{-J})$	0
[36]	N^2+4N	0
RA [39]	$6N$	0
FA [41]	$3.5N$	$N^2/4$
PA [41]	$6N(1-2^{-J})+0.5N$	0
[42]	$\approx 7N$	0
[43]	$8N(1-2^{-J})$	0
[44]	$\approx 6.25N$	0
[45]	$2N$	$N^2/2$
[46]	$2N$	$N^2/4$
[46]	$3N(1-2^{-J})+2N$	0
[47,48]	$9N$	0
Proposed	$4N(1-2^{-J})$	0

5. Synthesis Results of the Hardware Implementation of the Proposed Digital Image Decoder

Described in this paper is a digital image decoder for efficient hardware implementation with three color planes (Y, U and V) and its functional correctness had been verified by implementation within Altera DE2-115 development board, produced by Terasic Technologies [72], on an EP4CE115F29C7 FPGA device. Synthesis results, which show the amount of utilized resources and the maximum operating frequency of the decoder are presented in Table 9.

Table 9. FPGA synthesis results of presented digital image decoder.

Parameter	Value
Number of logic elements	77,127
Memory size	1,884,207 bits
Number of multipliers	12
Maximum operating frequency at 85 °C	114.71 MHz

FPGA synthesis results of proposed digital image decoder have been compared with synthesis results of various state-of-the-art architectures of digital image decoders. The results of comparison are presented in Table 10.

Table 10. Comparison of FPGA synthesis results for various architectures of digital image decoder.

Architecture	Frame Size/ Tile Size	Memory Size [kbits]	Maximum Operating Frequency [MHz]
[1]	160 × 120	n/a	24.15
[2]	512 × 512	1424	89.9
[3]	704 × 576	594	105.6
[4]	1920 × 1080	433,357	42.8
[5]	512 × 512	1602	116.9
[6]	2048 × 1080	2710	n/a
[7]	1920 × 1080	6192	n/a
[8]	1920 × 1080	5182	180
[9]	1920 × 1080	3277	110
[10] 3 × 3 NoC	320 × 200	1842	n/a
[10] 2 × 2 NoC	320 × 200	1125	n/a
Proposed	1920 × 1080	1840	114.71

It can be seen that the proposed hardware architecture for digital image decoder requires at least 32% less memory resources in comparison to the other state-of-the-art decoders which can process HD frame size or HD tile size. Some state-of-the-art architectures which process frame size or tile size smaller than HD size require total memory size lower than the memory size of the proposed solution. However, when frame size or tile size is taken into account as well, it can be concluded that proposed digital image decoder architecture can process 7.9 times larger frame/tile size, while it only requires 29% greater memory size in comparison with [2]. Similarly, the proposed digital image decoder can process 5.1 times larger frame size while it requires only a 3.1 times greater memory size in comparison [3]. The proposed solution for digital image decoder can process 5.1 times larger frame/tile size than [5] but utilizes only 15% more memory resources. Finally, the proposed digital image decoder architecture can process 32.4 times larger frame size, while requires only 64% greater memory size in comparison with 2 × 2 NoC decoder from [10]. In comparison to all other state-of-the-art solutions, the proposed architecture requires less memory size although it can process larger frame size.

Additionally, it can be seen that the proposed solution for digital image decoder has lower maximum operating frequency than architectures from [5,8], but can operate at higher frequency than all other state-of-the-art architectures.

6. Conclusions

The digital image decoder for efficient hardware implementation presented in this paper has many advantages in comparison to state-of-the-art solutions. The proposed entropy decoder and decoder probability estimator for efficient hardware implementation reduces the hardware complexity compared to the other state-of-the-art solutions by reducing or completely eliminating multiplication and division operations. The hardware complexity of the proposed dequantizer is reduced, in comparison to the state-of-the-art solutions, due to using the multiplication operation by power of two (which is implemented by using permanently shifted hardware connections between input and output bit lines), and due to using the multiplication operation with narrow-range integer which is implemented by simple lookup table. The proposed novel hardware realization of the inverse subband transformer, which performs 2-D inverse 5/3 DWT, utilizes 20% less memory resources compared to the best realization so far published in the literature. As a basic building block for the 2-D inverse 5/3 DWT, non-stationary hardware realization of the 1-D inverse 5/3 DWT filter has been used. This realization utilizes the lowest number of logic elements and the lowest number of registers, has the lowest total power dissipation and allows the highest operating frequency in comparison to any other realizations from the literature. The proposed digital image decoder requires at least 32% less memory resources in comparison to the other state-of-the-art decoders from the literature which can process HD frame size and requires effectively lower memory size than state-of-the-art solutions which process frame size or tile size smaller than HD size. The presented solution for digital image decoder has maximum operating frequency comparable with the highest maximum operating frequencies among the state-of-the-art solutions.

Future work on proposed digital image decoder would include modifications and optimizations which would increase maximum operating frequency while maintaining the reduced amount of utilized logical and memory resources. Additionally, future work could include the upgrade of proposed digital image decoder for efficient hardware implementation so that it can support decompression of ultra-high-definition (UHD) resolution images.

Author Contributions: Conceptualization, G.S. and V.R.; methodology, G.S.; software, M.P., V.R. and D.P.; validation, G.S. and V.R.; formal analysis, G.S.; investigation, G.S., M.P. and V.R.; resources, G.S., M.P. and V.R.; data curation, G.S., M.P. and D.P.; writing—original draft preparation, G.S.; writing—review and editing, M.P. and D.P.; supervision, M.P.; project administration, M.P. and D.P.; funding acquisition, M.P. and D.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was conducted during research supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, contract number 451-03-68/2022-14/200103.

Data Availability Statement: Data available on request.

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

References

1. Bakr, M.; Salama, A.E. Implementation of 3D-DCT based video encoder/decoder system. In Proceedings of the 45th Midwest Symposium on Circuits and Systems (MWSCAS-2002), Tulsa, OK, USA, 4–7 August 2002.
2. Descampe, A.; Devaux, F. A flexible, line-based JPEG 2000 decoder for digital cinema. In Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, Dubrovnik, Croatia, 12–15 May 2004.
3. Schumacher, P.; Denolf, K.; Chilira-Rus, A.; Turney, R.; Fedele, N.; Vissers, K.; Bormans, J. A scalable, multi-stream MPEG-4 video decoder for conferencing and surveillance applications. In Proceedings of the IEEE International Conference on Image Processing, Genoa, Italy, 11–14 September 2005.
4. Warsaw, T.; Lukowiak, M. Architecture design of an H.264/AVC decoder for real-time FPGA implementation. In Proceedings of the IEEE 17th International Conference on Application-specific Systems, Architectures and Processors (ASAP'06), Steamboat Springs, CO, USA, 11–13 September 2006.
5. Descampe, A.; Devaux, F.; Rouvroy, G.; Legat, J.; Quisquater, J.; Macq, B. A Flexible Hardware JPEG 2000 Decoder for Digital Cinema. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 1397–1410. [CrossRef]

6. Xu, R.; Xiao, T.; Xu, C. A High-Performance JPEG2000 Decoder Based on FPGA According to DCI Specification. In Proceedings of the Symposium on Photonics and Optoelectronics, Chengdu, China, 19–21 June 2010.
7. Bonatto, A.; Negreiros, M.; Soares, A.; Susin, A. Towards an Efficient Memory Architecture for Video Decoding Systems. In Proceedings of the Brazilian Symposium on Computing System Engineering, Natal, Brazil, 5–7 November 2012.
8. Engelhardt, D.; Moller, J.; Hahlbeck, J.; Stabernack, B. FPGA implementation of a full HD real-time HEVC main profile decoder. *IEEE Trans. Cons. Electr.* **2014**, *60*, 476–484. [CrossRef]
9. Stabernack, B.; Moller, J.; Hahlbeck, J.; Brandenburg, J. Demonstrating an FPGA implementation of a full HD real-time HEVC decoder with memory optimizations for range extensions support. In Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP), Krakow, Poland, 23–25 September 2015.
10. Barge, I.; Ababei, C. H.264 video decoder implemented on FPGAs using 3×3 and 2×2 networks-on-chip. In Proceedings of the International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 4–6 December 2017.
11. Witten, I.H.; Neal, R.M.; Cleary, J.G. Arithmetic coding for data compression. *Commun. ACM* **1987**, *30*, 520–540. [CrossRef]
12. Moffat, A.; Neal, R.M.; Witten, I.H. Arithmetic coding revisited. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 28–30 March 1995; pp. 202–211.
13. Moffat, A.; Neal, R.M.; Witten, I.H. Arithmetic coding revisited. *ACM Trans. Inform. Syst.* **1998**, *16*, 256–294. [CrossRef]
14. Mitchell, J.L.; Pennebaker, W.B. Software implementations of the Q-coder. *IBM J. Res. Dev.* **1988**, *21*, 753–774. [CrossRef]
15. Pennebaker, W.B.; Mitchell, J.L.; Langdon, G.G.; Arps, R.B. An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM J. Res. Dev.* **1988**, *32*, 717–726. [CrossRef]
16. Pennebaker, W.B.; Mitchell, J.L. Probability Adaptation for Arithmetic Coders. U.S. Patent 4,933,883, 12 June 1990.
17. Pennebaker, W.B.; Mitchell, J.L. Probability Adaptation for Arithmetic Coders. U.S. Patent 4,935,882, 19 June 1990.
18. Bottou, L.; Howard, P.G.; Bengio, Y. The Z-coder adaptive binary coder. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 30 March–1 April 1998; pp. 13–22.
19. Bengio, Y.; Bottou, L.; Howard, P.G. Z-Coder: Fast Adaptive Binary Arithmetic Coder. U.S. Patent 6,188,334, 13 February 2001.
20. Bengio, Y.; Bottou, L.; Howard, P.G. Z-Coder: A Fast Adaptive Binary Arithmetic Coder. U.S. Patent 6,225,925, 1 May 2001.
21. Bengio, Y.; Bottou, L.; Howard, P.G. Z-Coder: A Fast Adaptive Binary Arithmetic Coder. U.S. Patent 6,281,817, 28 August 2001.
22. Wallace, G.K. The JPEG still picture compression standard. *IEEE Trans. Consum. Electron.* **1992**, *38*, 18–34. [CrossRef]
23. Ono, F.; Denki, M.; Kaisha, K. Coding Method of Image Information. U.S. Patent 5,059,976, 22 October 1991.
24. Ono, F.; Denki, M.; Kaisha, K. Coding System. U.S. Patent 5,307,062, 26 April 1994.
25. The Data Compression Resource on the Internet. Available online: http://www.data-compression.info/Algorithms/RC/ (accessed on 10 June 2021).
26. Soman, K.P.; Ramachandran, K.I.; Resmi, N.G. *Insight into Wavelets from Theory to Practice*; PHI Learning: Delhi, India, 2010.
27. Parhi, K.K.; Nishitani, T. VLSI architectures for discrete wavelet transforms. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **1993**, *1*, 191–202. [CrossRef]
28. Wu, P.C.; Chen, L.G. An efficient architecture for two-dimensional discrete wavelet transform. *IEEE Trans. Circuit Syst. Video Technol.* **2001**, *11*, 536–545.
29. Zervas, N.D.; Anagnostopoulos, G.P.; Spiliotopoulos, V.; Andreopoulos, Y.; Goutis, C.E. Evaluation of design alternatives for the 2-D-discrete wavelet transform. *IEEE Trans. Circuits Syst. Video Technol.* **2001**, *11*, 1246–1262. [CrossRef]
30. Cheng, C.; Parhi, K.K. High-speed VLSI implementation of 2-D discrete wavelet transform. *IEEE Trans. Signal Process.* **2008**, *56*, 393–403. [CrossRef]
31. Usha Bhanu, N.; Chilambuchelvan, A. Efficient VLSI architecture for discrete wavelet transform. *Int. J. Comput. Sci. Issues* **2011**, *1*, 32–36.
32. Ghantous, M.; Bayoumi, M. P^2E-DWT: A parallel and pipelined efficient VLSI architecture of 2-D discrete wavelet transform. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Rio de Janeiro, Brazil, 15–18 May 2011; pp. 941–944.
33. Liu, C.C.; Shiau, Y.H.; Jou, J.M. Design and implementation of a progressive image coding chip based on the lifted wavelet transform. In Proceedings of the 11th VLSI Design/CAD Symposium, Pingtung, China, 16-19 August 2000.
34. Jou, J.M.; Shiau, Y.H.; Liu, C.C. Efficient VLSI architectures for the biorthogonal wavelet transform by filter bank and lifting scheme. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sydney, Australia, 6–9 May 2001; pp. 529–532.
35. Lian, C.J.; Chen, K.F.; Chen, H.H.; Chen, L.G. Lifting based discrete wavelet transform architecture for JPEG2000. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Sydney, Australia, 6–9 May 2001; pp. 445–448.
36. Andra, K.; Chakrabarti, C.; Acharya, T. A VLSI architecture for lifting-based forward and inverse wavelet transform. *IEEE Trans. Signal Process.* **2002**, *50*, 966–977. [CrossRef]
37. Chakrabarti, C.; Vishwanath, M. Efficient realizations of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers. *IEEE Trans. Signal Process.* **1995**, *43*, 759–771. [CrossRef]
38. Vishwanath, M.; Owens, R.M.; Irwin, M.J. VLSI architectures for the discrete wavelet transform. *IEEE Trans. Circuits Syst. II* **1995**, *42*, 305–316. [CrossRef]
39. Liao, H.; Mandal, M.K.; Cockburn, B.F. Efficient implementation of lifting-based discrete wavelet transform. *Electron. Lett.* **2002**, *38*, 1010–1012. [CrossRef]

40. Tseng, P.C.; Huang, C.T.; Chen, L.G. Generic RAM-based architecture for two-dimensional discrete wavelet transform with line-based method. In Proceedings of the APCCAS, Asia-Pacific Conference on Circuits and Systems, Denpasar, Indonesia, 28–31 October 2002; pp. 363–366.
41. Xiong, C.Y.; Tian, J.; Liu, J. Efficient high-speed/low-power line-based architecture for two-dimensional discrete wavelet transform using lifting scheme. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 309–316. [CrossRef]
42. Mohanty, B.K.; Meher, P.K. Memory efficient modular VLSI architecture for highthroughput and low-latency implementation of multilevel lifting 2-D DWT. *IEEE Trans. Signal Process.* **2011**, *59*, 2072–2084. [CrossRef]
43. Aziz, S.M.; Pham, D.M. Efficient parallel architecture for multi-level forward discrete wavelet transform processors. *Comp. Elect. Eng.* **2012**, *38*, 1325–1335. [CrossRef]
44. Mohanty, B.K.; Meher, P.K. Memory-efficient high-speed convolution-based generic structure for multilevel 2-D DWT. *IEEE Trans. Circuits Syst. Video Technol.* **2013**, *23*, 353–363. [CrossRef]
45. Hsia, C.H.; Chiang, J.S.; Guo, J.M. Memory-efficient hardware architecture of 2-D dual-mode lifting-based discrete wavelet transform. *IEEE Trans. Circuits Syst. Video Technol.* **2013**, *23*, 671–683. [CrossRef]
46. Darji, A.D.; Kushwah, S.S.; Merch, S.N.; Chandorkar, A.N. High-performance hardware architectures for multi-level lifting-based discrete wavelet transform. *Eurasip J. Image Video Process.* **2014**, *47*, 1–19. [CrossRef]
47. Hsia, C.H.; Chiang, J.S.; Chang, S.H. An efficient VLSI architecture for 2-D dual-mode SMDWT. In Proceedings of the 2013 IEEE International Conference on Networking, Sensing and Control (ICNSC), Paris, France, 10–12 April 2013; pp. 775–779.
48. Hsia, C.H. A New VLSI Architecture Symmetric Mask-Based Discrete Wavelet Transform. *J. Internet Technol.* **2014**, *15*, 1083–1090.
49. Ballesteros, D.M.L.; Renza, D.; Pedraza, L.F. Hardware Design of the Discrete Wavelet Transform: An Analysis of Complexity, Accuracy and Operating Frequency. *Ing. Cienc.* **2016**, *12*, 129–148. [CrossRef]
50. Wang, H.; Wang, J.; Zhang, X. Architecture and Implementation of Shape Adaptive Discrete Wavelet Transform for Remote Sensing Image Onboard Compression. In Proceedings of the 3rd IEEE International Conference on Computer and Communications, Chengdu, China, 13–16 December 2017; pp. 1803–1808.
51. Basiri, M.A.M.; Noor, M.S. An Efficient VLSI Architecture for Convolution Based DWT Using MAC. In Proceedings of the 31st International Conference on VLSI Design and 17th International Conference on Embedded System, Pune, India, 6–10 January 2018; pp. 271–276.
52. Aziz, F.; Javed, S.; Gardezi, S.E.I.; Younis, C.J.; Alam, M. Design and Implementation of Efficient DA Architecture for LeGall 5/3 DWT. In Proceedings of the 2018 International Symposium on Recent Advances in Electrical Engineering (RAEE), Islamabad, Pakistan, 17–18 October 2018.
53. Ganapathi, H.; Kotha, S.R.; Telugu, K.S.R. A new approach for 1-D and 2-D DWT architectures using LUT based lifting and flipping cell. *Int. J. Electron. Commun.* **2018**, *97*, 165–177.
54. Gardezi, S.E.I.; Aziz, F.; Javed, S.; Younis, C.J.; Alam, M.; Massoud, Y. Design and VLSI Implementation of CSD based DA Architecture for 5/3 DWT. In Proceedings of the 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 8–12 January 2019; pp. 548–552.
55. Tausif, M.; Khan, E.; Mohd, H.; Reisslein, M. Lifting-Based Fractional Wavelet Filter: Energy-Efficient DWT Architecture for Low-Cost Wearable Sensors. *Adv. Multimed.* **2020**, *2020*, 8823689. [CrossRef]
56. Chakraborty, A.; Banerjee, A. A memory and area-efficient distributed arithmetic based modular VLSI architecture of 1D/2D reconfigurable 9/7 and 5/3 DWT filters for real-time image decomposition. *J. Real-Time Image Process.* **2020**, *17*, 1421–1446. [CrossRef]
57. Chakraborty, A.; Banerjee, A. A Memory Efficient, Multiplierless & Modular VLSI Architecture of 1D/2D Re-Configurable 9/7 & 5/3 DWT Filters Using Distributed Arithmetic. *J. Circuits Syst. Comput.* **2020**, *29*, 2050151.
58. Pinto, R.; Shama, K. An Efficient Architecture for Modifed Lifting-Based Discrete Wavelet Transform. *Sens. Imaging* **2020**, *21*, 53. [CrossRef]
59. Joshi, A. Hardware Implementation of Audio Watermarking Based on DWT Transform. In *Security and Privacy from a Legal, Ethical, and Technical Perspective*; IntechOpen: London, UK, 2019; pp. 1–17.
60. Tausif, M.; Jain, A.; Khan, E.; Hasan, M. Memory-efficient architecture for FrWF-based DWT of high-resolution images for IoMT applications. *Multimed. Tools Appl.* **2021**, *80*, 11177–11199. [CrossRef]
61. Rajović, V.; Savić, G.; Čeperković, V.; Prokin, M. Combined one-dimensional lowpass and highpass filters for subband transformer. *Electron. Lett.* **2013**, *49*, 1150–1152. [CrossRef]
62. Savić, G.; Prokin, M.; Rajović, V.; Prokin, D. Novel one-dimensional and two-dimensional forward discrete wavelet transform 5/3 filter architectures for efficient hardware implementation. *J. Real-Time Image Process.* **2019**, *16*, 1459–1478. [CrossRef]
63. Savić, G.; Prokin, M.; Rajović, V.; Prokin, D. High-Performance 1-D and 2-D Inverse DWT 5/3 Filter Architectures for Efficient Hardware Implementation. *Circuits Syst. Signal Process.* **2017**, *36*, 3674–3701. [CrossRef]
64. Savić, G.; Prokin, M.; Rajović, V.; Prokin, D. Efficient Hardware Realization of Digital Image Decoder. In Proceedings of the 25th Telecommunications Forum (TELFOR), Belgrade, Serbia, 21–22 November 2017; pp. 534–541.
65. Čeperković, V.; Pavlović, S.; Mirković, D.; Prokin, M. Fast Codec with High Compression Ratio and Minimum Required Resources. U.S. Patent 8,306,340, 6 November 2012.
66. Martin, G.N.N. Range encoding: An algorithm for removing redundancy from a digitised message. In Proceedings of the Video & Data Recording Conference, Southampton, UK, 24–27 July 1979.

67. Schindler, M. A fast renormalization for arithmetic coding. In Proceedings of the Data Compression Conference, Snowbird, UT, USA, 30 March–1 April 1998.
68. Range Encoder Homepage. Available online: http://www.compressconsult.com/rangecoder/ (accessed on 10 June 2021).
69. Magenheimer, D.J.; Peters, L.; Pettis, K.W.; Zuras, D. Integer multiplication and division on the HP precision architecture. *IEEE Trans. Comput.* **1988**, *37*, 980–990. [CrossRef]
70. Granlud, T.; Montgomery, P.L. Division by invariant integers using multiplication. *SIGPLAN Not.* **1994**, *29*, 61. [CrossRef]
71. Altera Press. *Cyclone IV Device Handbook—Volume 1*; Version 1.8; Altera Press: Blacksburg, VA, USA, 2013.
72. Terasic Technologies. *DE2-115 User Manual*; Version 2.1; Terasic Technologies: Hsinchu, Taiwan, 2012.