Design of Hierarchical Architecture of Multilevel Discrete Wavelet Transform Using VHDL Language

Waleed Fawwaz Shareef

Received on:4/8/2009
Accepted on:7/1/2010

Abstract

The wide spread of devices that use image processing in its functions, like cellular phone and digital cameras, increases the need for specialized processors for these functions as a replacement for software programs that consume more time and resources. This paper presents a hardware description for discrete wavelet transform (DWT) module in VHDL language. The design involves the forward DWT (fDWT) and its inverse (iDWT) characterized by variable number of transformation levels, ranging from one level to seven levels. Each one of these two modules is designed as hierarchical scheme that uses one-dimensional processing module twice to represent two-dimensional processing. The module can be used repeatedly on the same image for multilevel processing. Three versions of the design are presented (v64, v128 and v256), each one adapted different image size. Synthesis process showed that the design frequency is about 56MHz. The simulation process showed that the maximum possible rounding error is about 0.012%. This resolution with the variable number of processing level adapts this design to fit in many applications. Finally, a comparison of the proposed design with other related work is presented, considering performance and specifications.

Keywords: wavelet, image coding, VHDL.

تصميم معمارية هيكلية لتحويل الموجة DWT متعدد المستويات بأستخدام لغة VHDL

الخلاصة

مع انتشار الاجهزة الحديثة التي تتضمن وظائفها معالجة الصور الرقمية, كالهواتف الخلوية و اجهزة التصويرالرقمية, ازدادت الحاجة الى توفر معالجات متخصصة تقوم بهذه الوظائف كبديل للبرمجيات التي تستغرق وقتا و مواردا اكثر. هذا البحث يقدم وصفا للكيان المادي لوحدة DWT بلغة VHDL. التصميم يتضمن DWT الامامي وDWT العكسي و يتميز بمقدار متغير لعدد المستويات من واحد الى سبعة. كل وحدة منهما مصممة بصورة تركيبية من وحدات اصغر تقوم بعملية معالجة احادية البعد لتكوين وحدة ثنائية البعد. هذه الوحدة يمكن استخدامها تكراريا على نفس الصورة لتوفير معالجة متعددة المستويات. صممت ثلاثة نماذج مختلفة (v64 و v256 و v128) كل منها مخصصة لحجم صورة مختلف. عملية تحليل التصميم اظهرت ان تردد التصميم هو تقريبا 56MHz. عملية المحاكاة اظهرت ان اكبر خطأ تقريبي ممكن هو بقيمة 0.012%. ان الدقة العالية مع العدد المتغير لمستوى المعالجة

* Control and Systems Engineering Department, University of Technology/Baghdad

https://doi.org/10.30684/etj.28.7.7
2412-0758/University of Technology-Iraq, Baghdad, Iraq

يجعل هذا التصميم ملائما لكثير من التطبيقات. اخيرا يقدم البحث مقارنة بين التصميم المقترح مع تصاميم اخرى مشابهة من حيث الاداء و المواصفات.

1. Introduction

The increasing growth in demand for image and video data transmission, increase the stress upon developing new image compression technology. Among the several compression standards available, the JPEG image compression standard is in wide spread use today [1]. JPEG uses the Discrete Cosine Transform (DCT) as the transform, applied to 8-by-8 blocks of image data. The newer standard JPEG2000 is based on the Discrete Wavelet Transform (DWT) [2]. Wavelet Transform offers multi-resolution image analysis, which appears to be well matched to the low level characteristic of human vision. The DCT is essentially unique but wavelet transform has many possible realizations [3].

The most basic wavelet transform is the Haar discrete transform as it encapsulates the basic concepts of wavelet transforms used today. The Haar transform works well (provides a relatively sparse wavelet representation) for signals that are approximately piecewise constant [4].

The Haar wavelet transform describes the image in terms of a coarse overall shape, plus some details that range from broad to narrow. The Haar wavelet transform is applied iteratively on an image to generate multilevel decomposition. At level l decomposition, $3l + 1$ sub-bands are produced.

A one-level Haar wavelet transform decomposes the original image (see **Fig. 1.a**) into four sub-bands: LL, LH, HL and HH (see **Fig. 1.b**). The Haar wavelet coefficients are computed by Eq(1) and Eq(2), respectively [5].

$$L(i) = f(x(i)) = \frac{x(i)+x(i+1)}{\sqrt{2}} \quad \text{... (1)}$$

$$H(i) = g(x(i)) = \frac{x(i)-x(i+1)}{\sqrt{2}} \quad \text{...(2)}$$

where, $x(i)$ and $x(i + 1)$ are the current and next pixels values of an image, respectively.

The original pixels values could be retrieved in iDWT process from the High and Low components according to the next Eq(3) and Eq(4) below:

$$x(i) = \frac{L(i)+H(i)}{\sqrt{2}} \quad \text{....(3)}$$

$$x(i+1) = \frac{L(i)-H(i)}{\sqrt{2}} \quad \text{...(4)}$$

In this paper, the hardware description of DWT chip through the use of VHDL language is addressed. The DWT module specifications; (iDWT, fDWT, and theirs algorithms) are reviewed in the next section. Section three discusses simulation results, whereas final conclusions are presented in section four.

2 Design layout and algorithms

The overall design is composed of two-dimensional DWT module (DWT-2D) designed as synthesisable VHDL code, in addition to memory unit designed for simulation only. The DWT-2D module is composed of one-dimensional DWT module (DWT-1D) which represents the main part of the design (see **Fig. 2**).

The discussion and figures below focus on the fDWT architecture, however the design of the fDWT and iDWT are almost identical.

2.1 Algorithms of fDWT/iDWT modules

Generally the proposed algorithm of the fDWT is divided into three phases: *initializing* phase, *horizontal pass* phase, and *vertical pass* phase.

2.1.1 Initialization phase

Upon applying the *Reset* signal the initialization phase is start. During the initializing phase the user must provide DWT-2D module with necessary information of the input image, in the first four clock cycles, according to the next sequence:

1. Start address of memory space occupied by the image.
2. Start address of memory space specified for temporary data storage (to store intermediate data of horizontal pass results).
3. Style used to store image (either *row by row* or *column by column*).
4. Number of the required transformation levels.

This information is presented to the DWT-2D module through *Control Bus*. The image size is configurable only before programming the target device (FPGA, CPLD, etc.), that must be assigned to one of the next three values: 256,128 and 64. This make the final design comes out in three versions as shown later in simulation results.

Then DWT-2D module apply the internal reset signal *int_Reset* at the DWT-1D then provide it with the necessary information (through *int_Control Bus*) to perform horizontal or vertical pass, as follow:

1. Start address of memory space occupied by the current line of pixels.
2. Start address of memory space for temporary data storage for the current line of pixels.
3. Line type (0 for horizontal pass / 1 for vertical pass).
4. Width of current line. (Equal to image size at level one, then halved at every new level).
5. Number of lines in current image section. (Equal to image size at level one, then halved at every new level).

The piece of code that employed to implement the *reset phase* and *receiving address vector phase* of DWT-1D module is shown below:

```
IF int_reset = '1' THEN
   ready <= '1';
   state <= TS_READY;
   access_bus <= '0';
   counter<=0;
ELSE
   IF clk'event AND clk = '1'
THEN
     CASE state IS
     WHEN TS_READY =>
                 CASE ctrl_sig IS
WHEN "000" => src_start_offset <=
                 unsigned(ctrl_data);
WHEN "001" =>src_step <=
                 unsigned(ctrl_data);
WHEN "010" =>dst_start_offset_lo
                 <=
                 unsigned(ctrl_data);
WHEN "011" =>dst_start_offset_hi
                 <=
                 unsigned(ctrl_data);
```

```
WHEN "100"=>dst_step_lo <=
                 unsigned(ctrl_data);
WHEN "101"=>dst_step_hi <=
                 unsigned(ctrl_data);
WHEN "110" =>elem_count <=
                 unsigned(ctrl_data);
                 state <= TS_Read_a;
                 int_ready <= '0'; i
<=1 1;
WHEN OTHERS =>
END CASE;
....
```

2.1.2 Horizontal/Vertical pass phase

During the *Horizontal Pass* and *Vertical Pass* phases, the Low and High components for fDWT (or *x(i)* and *x(i+1)* for the iDWT) are calculated. The calculations is performed according Eq(5) through Eq(8) instead of Eq(1) through Eq(4) to avoids the extra calculation needs to divide by the value $\sqrt{2}$.

$$L(i) = \frac{x(i)+x(i+1)}{2} \quad .... (5)$$

$$H(i) = \frac{x(i)-x(i+1)}{2} \quad ...... (6)$$

$$x(i) = L(i) + H(i) \quad ...(7)$$

$$x(i+1) = L(i) - H(i) \quad ...(8)$$

The piece of code that employed to implement Eq(5) and Eq(6) is shown below (note that the division by two is implemented as shift to right):

```
FUNCTION calculate
(x, y : signed(7 DOWNTO 0);
        islow:std_logic) RETURN
signed IS
VARIABLE    TEMP1:signed(8
downto 0);
VARIABLE TEMP2:signed(7
downto 0);
BEGIN
IF islow='1' then
temp1:=( x(7) & x )+(y(7) & y)+
to_signed(1, 8);
temp2:= temp1(8 downto 1);  return
temp2 ;
ELSE
temp1:=( x(7) & x )-(y(7) & y)+
to_signed(1, 8);
temp2:= temp1(8 downto 1); return
temp2;
END IF;  END;
```

The design employs a similar code in the iDWT for Eq(7) and Eq(8) wihout shift to right operation.

After manipulating one line of the input image, DWT-1D module generates *int_Ready* signal to the DWT-2D module which in turn provide the DWT-1D module with the information for the next line if any. If all lines in the *Horizontal Pass* are finished then the process of *Vertical Pass* is initiated.

2.1.3 Level updating

If more than one transformation level is required, the overall process is repeated. The DWT-2D module reinitializes the DWT-1D module with the necessary information of the new image (which represent the LL section of the image from the previous level) to start new horizontal and vertical passes phases. When no more levels are exist, DWT-2D generate *Ready* signal to the outside indicating process end. The complete process is demonstrated in the flowchart shown in **Fig. 3**.

2.2 Memory Unit Design

Memory unit needs to be of size twice as the image size. For simulation purpose, a memory was modelled using non-synthesisable VHDL code for simulation only. During simulation, the recorded process time (see Table 3) includes extra delay inserted to simulate delay time presented by actual memory devices. This would give flexibility in choosing the memory speed depending on the DWT maximum frequency. Memory *Read* and *write* processes are each given three clock cycles.

3. Simulation Results

Simulation was implemented for various image sizes and transformation levels for each size. **Fig.4** shows three copies of "*cameraman*" image (256×256, 128×128 and 64×64 respectively) with two different transformation levels for each size.

To estimate the rounding error produced by the arithmetic operations, a comparison was made between the original image and the resultant image after the fDWT and iDWT processes. The comparison includes cases shown in **Fig.4** and two other cases. Comparison results (shown in **Table 1**) illustrated that rounding error increases with increasing number of transformation level. The maximum rounding error was recorded in 7th transformation level with image size 256×256. It should be noted that actual error values for fDWT module are half of these viewed in **Table 1**, since the values in the table represents rounding error retrieved by subtracting pixels values of original image from those of image resultant from fDWT and iDWT module respectively.

The results presented in this section were produced by simulating the proposed architecture in ModelSim4.5 environment. The architecture itself was designed in the hardware description language VHDL in the ISE4.1 environment. The selected target technology was Xilinx Virtex II FPGA-XC2V10000. (The choice of target device has no large influence on synthesis and simulation results).

For illustration purposes, MATLAB^{TM} image toolbox (Version 7) was used to display the simulation data as pictures.

Finally, *device utilization summary* and *timing summary* for the three versions of the design (as stated by the *Place and route* reports) are shown in **Table 2**, whereas **Table 3** lists number of clock cycles required for coding completion (for several permutation of image size and transformation level number). Finally, **Table 4** show results of *performance*, *time*, and *size utilization* regarding other related works.

4 Conclusions

In this paper, a hardware description for discrete wavelet transform (DWT) module in VHDL language is presented. The modules performs 2-dimentional wavelet transform to images of size 64X64, 128X128, and 256X256 pixels. Synthesis process showed that the three versions is similar in the maximum frequency (about 56MHZ) and the used slice of the target device (about 600 slices), while the difference comes out in the number of clock cycles required for coding. In addition to image size, number of clock cycles is also dependent on the number of levels required and it ranges from 62,729 to 1,320,827clock cycles. The simulation process shows that a maximum rounding error of 0.012% was recorded in 7th transformation level with image size 256×256. According to the comparison results, the proposed design takes longer time to perform processing but utilize less slices in the target device. The competitive size of this design helps the module to fit in many embedded and mobile applications, while it's fine resolution and the variable level of transformation makes the design profitable to a wider range of applications.

5- References

[1]-MALLAT, S.G.: "**A theory for multi resolution signal decomposition: The wavelet representation**", IEEE Trans. Pat. Anal. Mach. Intell., 1989, 11, (7), pp. 674-693.

[2]-Rafael C. Gonzalez, Richard E. Woods "**Digital Image Processing**" Addison-Wesley publishing company, 1993

[3]-Faizal Iqbal, "**Wavelet Transform Based Image Compression On FPGA**", M. Sc. thesis, College of Engineering, Florida State University, 2004.

[4]-Ivan W. Selesnick, Polytechnic University "**Wavelet Transforms, A Quick Study**" , *Physics Today magazine*, October,2007.

[5]-D. S. Taubmann, M. W. Marcellin, "***JPEG2000 Image Compression Fundamentals, Standards and Practice***", Kluwer Academic Publishers, 2002.

[6]-Dhaha Dia, Medien Zeghid, Taoufik Saidani, Mohamed Atri, Belgacem Bouallegue, Mohsen Machhout and Rached Tourki "***Multi-level Discrete Wavelet Transform Architecture Design***" Proceedings of the World Congress on Engineering 2009 Vol I WCE 2009, July 1 - 3, 2009, London, U.K.

[7]-R.J.C. Palero, R.G. Gironés and A.S. Cortes "***A Novel FPGA Architecture of a 2-D Wavelet Transform***" Journal of VLSI Signal Processing 42, 273-284, August 4, 2005.

[8]-A. Benkrid, D. Crookes and K. Benkrid "Design ***and Implementation of Generic 2-D Biorthogonal Discrete Wavelet Transform on an FPGA***" IEEE, Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2001.

[9]-T. Acharya and C. Chakrabarti "***A Survey Lifting-based Discrete Wavelet Transform Architectures***" Journal of VLSI Signal Processing 42, 321-339,13 February 2006.

[10]]M.E. Angelopoulou, P.Y.K. Cheung, K. Masselos and Y. Andreopoulous "***Implementation and comparison of 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs***" Journal of Signal Systems 51, 3-21, 2008.

Table (1) Maximum and mean rounding error for various image size and transformation level number (as applied in Fig. 4)

Version	No. of level	Max e %	Mean e %
v64	1	0.011719	0.00191
v64	2	0.023438	0.003806
v64	3	0.03125	0.005668
v128	1	0.011719	0.001901
v128	3	0.03125	0.005691
v128	5	0.042969	0.009826
v256	2	0.023438	0.003813
v256	5	0.046875	0.009865
v256	7	0.050781	0.012673

Table (2) Device utilization summary and timing summary for the three versions of the design (as stated by the Place and route reports)

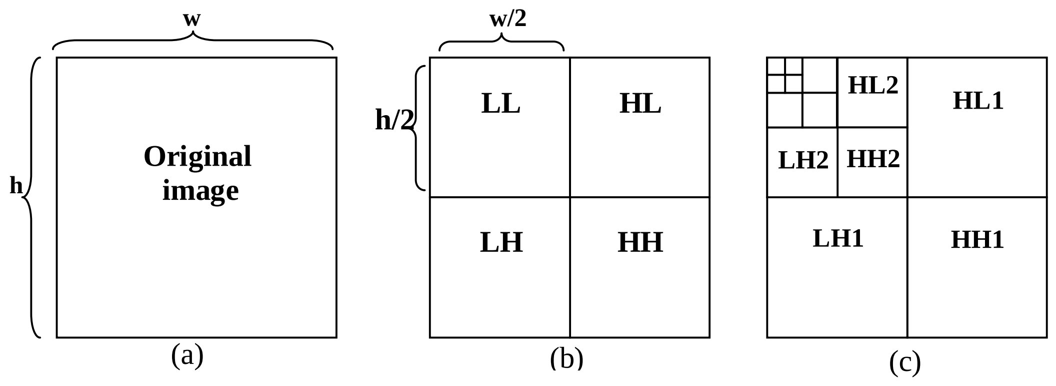
		Image size		
		64	128	256
Device utilization criteria	No. of External GCLKIOBs (out of 4)	1	1	1
	No.of External IOBs (out of 512)	40	44	48
	No.of SLICEs (out of 12288)	516	547	622
	No.of GCLKs (1 out of 40)	1	1	1
Timing criteria	Maximum frequency (MHz)	57	56	55
	Number of clock required for coding completion	See Table 3		

Table (3) Number of clock cycles required for coding completion (for all possible permutation of image size and transformation level numbers).

Version	No. of level	No. of ck cycles
v64	1	62,729
v64	2	78,732
v64	3	82,895
v64	4	84,018
v64	5	84,341
v64	6	84,444
v128	1	248,329
v128	2	311,052
v128	3	327,055
v128	4	331,218
v128	5	332,341
v128	6	332,664
v128	7	332,767
v256	1	988,169
v256	2	1,236,492
v256	5	1,319,381
v256	7	1,320,827

Table (4) Comparison of the proposed design with other related works

Parameters	Proposed v256	[6]	[7]	[8]	[9]	[10]
Image size	256×256	256×256	512×512	256×256	N/A	256×256
Input data Precision	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits
Device	XC2V10000	XCV600E	XCV600E	XCV600E	APEX20KE	XC4VLX15
Computation time (one level)	17.953 ms	2.36 ms	5.88 ms	N/A	N/A	N/A
Number Slices	622	1835	2554	4720	7726	2646
Frequency	55 Mhz	108 Mhz	45 Mhz	75 Mhz	66.8 Mhz	117,6 Mhz



Figure(1) Wavelet multiresolution property of an image: (a) represents original image, (b) a one-level decomposition produces 4 sub-bands, namely LL, LH, HL and HH, (c) a four-level decomposition produces 13 sub-bands

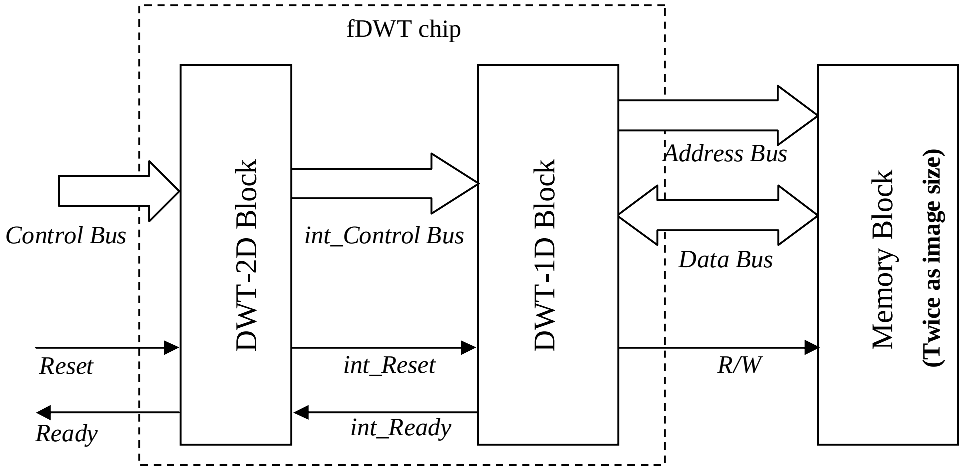


Figure (2) Layout of the proposed architecture for the fDWT

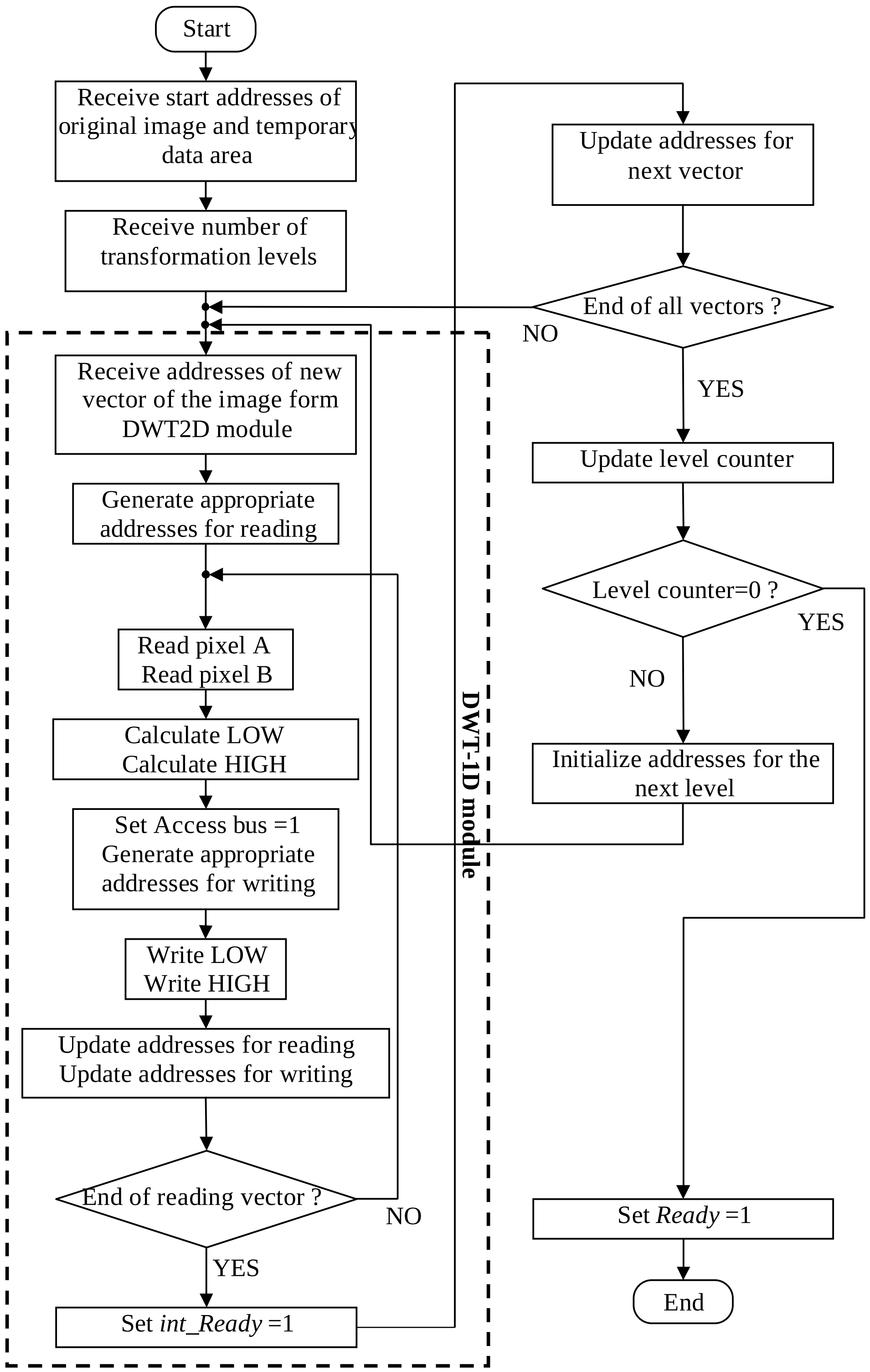
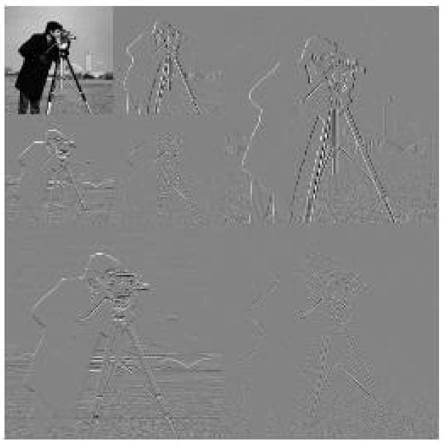
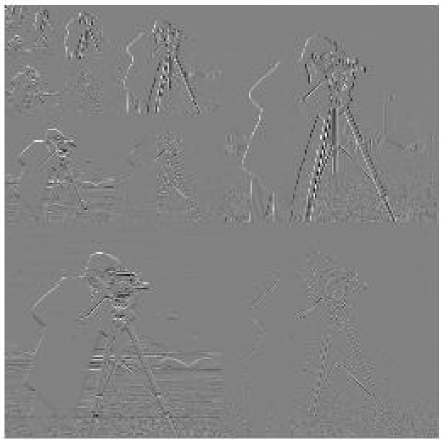

Start
Receive start addresses of original image and temporary data area
Receive number of transformation levels
Receive addresses of new vector of the image form DWT2D module
Generate appropriate addresses for reading
Read pixel A
Read pixel B
Calculate LOW
Calculate HIGH
Set Access bus =1
Generate appropriate addresses for writing
Write LOW
Write HIGH
Update addresses for reading
Update addresses for writing
End of reading vector ?
NO
YES
Set *int_Ready* =1
DWT-1D module
Update addresses for next vector
End of all vectors ?
NO
YES
Update level counter
Level counter=0 ?
YES
NO
Initialize addresses for the next level
Set *Ready* =1
End

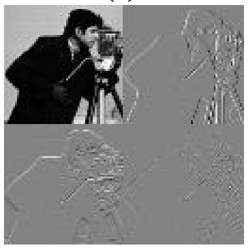
Figure (3) Flowchart represents algorithm executed by DWT-2D module

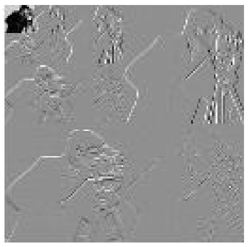
(a)


(b)


(c)

(d)


(e)


(f)

(g)

(h)

(i)

Figure (4)“*Cameraman.jpg*”
(a) Original image of size 256×256, (b) its Three-level DWT and (c) Seven-level DWT
(d) Original image of size 128×128, (e) its One-level DWT and (f) Three-level DWT
(g) Original image of size 64×64, (h) its One-level DWT and (i) Two-level DWT