第 23 卷第 7 期　　半 导 体 学 报　　Vol. 23, No. 7
2002 年 7 月　　CHINESE JOURNAL OF SEMICONDUCTORS　　July, 2002

VLSI Implementation of a Wavelet Image Coder

Qiao Shijie and Wang Guoyu

(Department of Electronic Engineering, Xi' an Jiaotong University, Xi' an 710049, China)

Abstract: A modular architecture for two dimension (2-D) discrete wavelet transform (DWT) is designed. The image data can be wavelet transformed in real time, and the structure can be easily scaled up to higher levels of DWT. A fast zerotree image coding (FZIC) algorithm is proposed by using a simple sequential scan order and two flag maps. The VLSI structure for FZIC is then presented. By combining 2-D DWT and FZIC, a wavelet image coder is finally designed. The coder is programmed, simulated, synthesized, and successfully verified by ALTERA CPLD.

Key words: discrete wavelet transform; zerotree image coding; VLSI; Verilog HDL; CPLD
EEACC: 2570
CLC number: TN432　　**Document code:** A　　**Article ID:** 0253-4177(2002)07-0695-06

1 Introduction

The discrete wavelet transform (DWT) has been applied extensively to digital image processing due to its capability for multiresolution representation. However, the real time two dimension (2-D) DWT requires heavy computations, and therefore heavy hardware. The first architecture for 2-D DWT was presented by Lewis[1], dedicated to the Daubechies 4-tap filter. Vishwanath proposed a systolic-parallel architecture for 2-D DWT[2], however, the controller becomes more complex when the architecture is scaled up to higher levels of DWT. In this paper we proposed an efficient modular architecture for 2-D DWT. The architecture is modular so that it can be easily scaled up to the higher levels of DWT.

The embedded zerotree wavelet (EZW) coding, introduced by Shapiro[3], is a very effective method to code the data after DWT. Said then developed an improved algorithm SPIHT[4]. SPIHT requires three lists, which cost a little bit. Su[5] and Lin[6] used flag maps to reduce the memory requirement. However, the recursive scan orders in both algorithms are still complicated for hardware. In this paper, by using a simple sequential scan order and two flag maps, we proposed a fast zerotree image coding (FZIC) algorithm. The FZIC algorithm is efficient and suitable for hardware implementation. Based on FZIC, an efficient architecture for zerotree coding is presented.

By combining the structures of the 2-D DWT with the FZIC, we presented a wavelet image coder. The Verilog HDL models of the two architectures were programmed. Then extensive simulation has been carried out to optimize the design. The coder has been synthesized to ALTERA CPLD. A demonstration PCB module has been built. The experiment result shows that both architectures are efficient and suitable for VLSI implementation.

Qiao Shijie male, postgraduate student. His research interests focus on wavelet image coding algorithms and VLSI implementation.
Wang Guoyu male, professor. His research interests include CMOS image sensor, VLSI implementation for image/voice compression and communication, and SOC.

Received 18 December 2001, revised manuscript received 11 March 2002

2 Architecture for 2-D DWT

The input signals in 1-D DWT are transformed into a low (L) and a high (H) component signals by a 1-D filter bank, which consists of a low pass (LP) filter $h(n)$ and a high pass (HP) filter $g(n)$. The 2-D DWT can be calculated by using a separable approach. First, the 1-D DWT is performed on each row, and then on each column. So the 2-D DWT decomposes the image data into four subbands, i. e., low-low (LL), high-low (HL), low-high (LH), and high-high (HH). Figure 1 shows a basic module for one level 2-D DWT. 2-D DWT with more levels can be implemented by assembling the basic units properly.

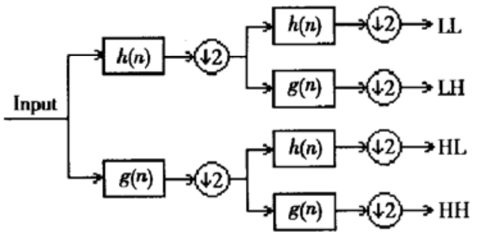

Fig. 1 Filter bank structure for 2-D DWT

The proposed architecture for 2-D DWT of two levels is shown in Fig. 2. The architecture is modular. In each level, the architecture consists of three parallel filter banks, one for horizontal DWT, the others for vertical DWT. The structures of the three filter banks are the same. The image data "d_{in}" is inputted to the unit of "Input Delay1". The input delay unit mimics the filter operation of a filter window, which moves the input data. The data output from the "Input Delay1" is then fed into the horizontal LP and HP filters. The LP and HP filters' outputs are stored in the LP and HP register banks separately for vertical DWT. The read and write operations of the register banks are controlled by the R/W controller. Once the data stored in the register banks are sufficient for calculating the vertical DWT, the data are read and fed into the vertical filter banks. The vertical DWT then starts calculating, while the horizontal DWT continues calculating. Except the LL subband, which is fed to the input delay unit of the next stage for further decomposition, all other subbands, i. e., HL, LH, and HH, are outputted as DWT coefficients. Once the basic modules for one-level 2-D DWT are ready, the 2-D DWT can be implemented by assembling the basic modules properly.

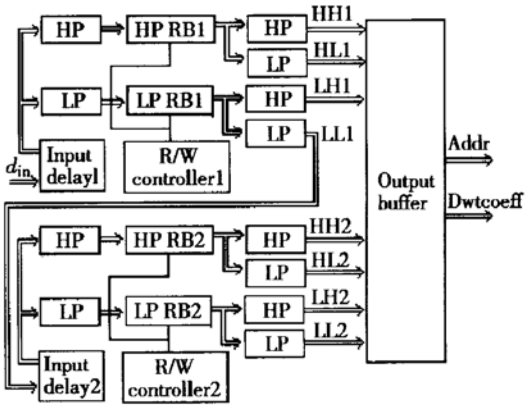

Fig. 2 Architecture for 2-D DWT

The DWT coefficient output from 2-D DWT are then collected by the output buffer. According to the levels and subbands the coefficients located in, the module generates the proper address for the coefficients. The DWT coefficients are then stored in a memory in raster scan manner subband by subband for later coding.

The verilog HDL modules with 5 levels 2D DWT for 512 × 512 image are designed. The simulation results show that the coefficients outputted from the architecture are the same as the coefficients outputted from C^{++} programs. The architecture is synthesized to ALTERA APEX20-K300EQC240. The synthesis results are shown in Table 1. The system clock of the 2-D DWT is 40.6MHz. The input images can be wavelet transformed in real time.

Table 1 2D DWT synthesized results

Image size	512 × 512
Image data	8bit(unsigned)
DWT coefficient	16bit(signed)
IOs	47
LCs	6634
Clock rate	40.6MHz

3 FZIC and its VLSI implementation

3.1 FZIC algorithm

The FZIC is similar to the SPIHT. However, since there are no lists in FZIC, different coding procedures are developed. The zerotree structure, which is the same as Shapiro's, is shown in Fig. 3 (a). The tree symbols in FZIC are defined as follows[4].

$H(i,j)$: set of all tree roots.

$O(i,j)$: set of all offspring of node (i,j).

$D(i,j)$: set of all descendent of node (i,j).

$L(i,j) = D(i,j) - O(i,j)$.

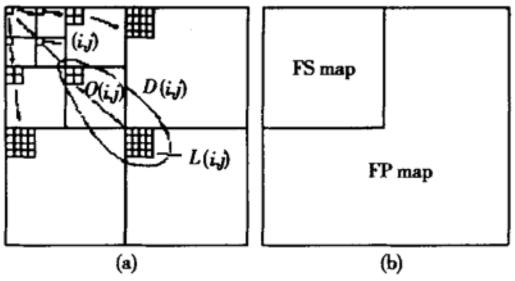

Fig. 3 (a) Zerotree structure; (b) FP and FS map

The set partitioning rules[4] are used to partition the set of pixels into subsets. The initial partition is formed with the set $\{(i,j)\}$ and $D(i,j)$, $(i,j) \in H$. In the coding process, if $D(i,j)$ becomes significant, it is partitioned into $L(i,j)$ and single-element sets $\{(k, l)\}$, $(k, l) \in O(i,j)$. Whereas if $L(i,j)$ is significant, it is partitioned into sets $\{D(k, l)\}$, $(k, l) \in O(i,j)$. As in LZC, two flag maps, 1bit FP maps and 3bit FS maps, are used to store the significance information for individual pixels and sets (D and L), respectively. The size of FP is the same as the image. However, since the highest frequency subbands do not have any child, the size of FS is only a quarter of the image, as shown in Fig. 3(b).

If the coefficient at (i,j) is significant, FP (i,j) is set to '1', otherwise, '0'. However, if the set $D(i,j)$ is significant, FS(i,j) is set to "001", otherwise, "000". If the set $L(i,j)$ is significant, FS(i,j) is set to "011", otherwise, "010". If the set at (i,j) is belong to an insignificant set D or L, FS (i,j) is set to "111", which means that no need to code the element at all.

To determine that a set is significant or not with respect to a given threshold $T = 2^n$, the following function is used:

$$S_n(\Gamma) = \begin{cases} 1 & \max\limits_{(i,j) \in \tau} \{|P(i,j)|\} \geqslant T \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

In ERZ and LZC, the trees are scanned in a recursive scan order. The recursive scan order is not suitable for hardware implementation. To simplify the hardware design, in the FZIC algorithm, the tree sets are scanned in raster scan manner one subband by one subband so that the parents must be scanned before the children. The raster scan manner is simple, and easy to implement in hardware.

The FZIC algorithm is summarized into four coding steps. The pseudo code of the FZIC is presented as follows.

```
(1) Initialization:
    Clear FP: set all FP($i,j$) = 0;
    Clear FS: If $(i, j) \in H$, then set FS($i,j$) =
              "000";
              Otherwise, set FS($i,j$) = "111";
    Output $n = \log_2\{\max|P(i,j)|\}$
(2) For each $(i,j)$ in $H$ do:
    If FP($i,j$) = 1 then output the $n$th most significant bit of $|P(i,j)|$. Otherwise output Sn($i,j$). If Sn($i,j$) = 1 then output the sign of $P(i,j)$ and set FP $(i,j)$ = 1;
(3) For each $(i,j)$ in FS do:
    If FS($i,j$) = "000" then output Sn($D(i,j)$);
      If Sn ($D(i,j)$) = 1 then
        Encode the children pixels of $O(i,j)$;
        If $L(i,j)$ = NULL then set FS($i,j$) = "001";
        Else output Sn($L(i,j)$);
          If Sn ($L(i,j)$) = 0 then set FS($i,j$) = "010";
          Else set FS($i,j$) = "011";
            For each($k, l$) $\in O(i,j)$
            Set FS$(k,l)$ = "000";
    Else if FS$(i,j)$ = "001" or "011" then
        Encode the children pixels of $O(i,j)$;
    Else if FS$(i,j)$ = "010" then
        Encode the children pixels of $O(i,j)$;
        Output Sn$(L(i,j))$;
        If Sn$(L(i,j))$ = 1 then set FS$(i,j)$ = "011";
            For each $(k,l) \in O(i,j)$ set FS$(k,l)$ = "000";
(4) $n = n-1$, go to step 2.
```

The process encoding the children pixels in step 3 is the same as in step 2 to code pixels in H.

The test FZIC codec is written in C++ language. The test image is 8 bpp, 512 × 512 Lena. The image is wavelet transformed to 5 levels, then coded by the FZIC. The test results are compared with EZW shown in Table 2. Without arithmetic coding, the FZIC achieves about the same performance as EZW does. The simple sequential scan order and the two flag maps make the FZIC algorithm efficient and suitable for hardware implementation.

Table 2 PSNR results of FZIC and EZW

Compress ratio	FZIC	EZW
16: 1	36. 2	36. 3
32: 1	33. 1	33. 2
64: 1	30. 1	30. 2

3. 2 Proposed architecture for FZIC

The architecture for FZIC is shown in Fig. 4. The coder reads DWT coefficient from memory DwtRam, codes the coefficient using the FZIC algorithm and outputs the coded bit streams to the output buffer, i. e. , FIFO. The functions performed by each of the modules are discussed below.

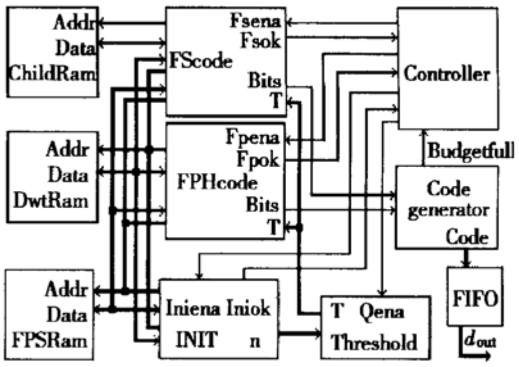

Fig. 4 Architecture for FZIC

3. 2. 1 Memory modules

The system contains three memory modules, i. e. , DwtRam, ChildRam and FPSRam. The DWT coefficients are stored in DwtRam. The DwtRam is a SRAM of N^2 words, each 16bit long. The 1bit FP and 3bit FS are combined together and stored in FPSRam. The FPSRam is a SRAM of N^2 words, each 4bit long.

In order to determine that a set is significant or not, all descendants of the node may be fetched and compared with the threshold, and the insignificant set identification may become extremely slow. To overcome this problem, another memory module ChildRam is used. The ChildRam is a SRAM, which is similar to the "largest descendant map"[7] and contains the largest descendant of each node that has descendant. The size of the memory is $N^2/4$ words, each 16bit long. When a particular set $D(i,j)$ is checked for significance, the corresponding value is read from ChildRam and compared with the threshold. However, when a particular set $L(i,j)$ is checked for significance, the corresponding value of the child nodes $(k,l) \in O(i,j)$ are read from ChildRam and compared with threshold. The memory fetch operations are simplified. The coding process is also speeded up.

3.2.2 INIT module

This module initializes the coding procedures. It clears the FP and FS flag maps and outputs the initial value of n to the threshold generator to generate the initial threshold. The value of n is also output to the code generator as coded bit streams. When the system has been initialized, the signal Iniok goes high. Thereafter, the main coding procedure starts.

3.2.3 Threshold module

This module generates the threshold values required for comparisons during various coding runs. The initial threshold T_0 is set to 2^n, where n is input from INIT module. Thereafter, when the coder goes from step 3 to step 4, the signal Qena goes high and the threshold value is reduced by a factor of 2.

3.2.4 FPHcode module

The FPHcode module codes the individual coefficients in H. The functions performed by this module can be divided into the following steps:

(1) Generate the address of the coefficient being coding and the write/read signal for DwtRam and FPSRam;

(2) Read FP value from FPSRam and perform the coding operations described in the coding step 2. It essentially includes comparing FP with '1', reading DWT coefficient from DwtRam, and finding the nth significant bit of the coefficient or comparing the coefficient with the threshold to generate a sequence of bits. If the coefficient becomes significant, the corresponding flag FP is set to '1'.

(3) If all coefficients in H are coded, the signal Fpok goes high.

3.2.5 FScode module

The tree sets and the individual coefficients partitioned from the sets are coded in the FScode module. The functions performed by this module are described as follows:

(1) Generate the address of the sets being coding and the write/read signal for FPSRam.

(2) Read FS from FPSRam and perform the coding operations described in the coding step 3. It involves generating all the control signals for memory modules and coding the insignificant sets D, L and the individual child coefficients separately. To code insignificant D or L, values are read from ChildRam and compared with the threshold. The coded bits are generated and the FS flag maps are set to proper values. To code the child coefficients, values are read from DwtRam and coded in the same way as described in the FPH-code module.

(3) If all the sets in FS are coded, the signal Fsok goes high.

3.2.6 Code generator

The coded bit streams outputted from FP and FS module are collected by the code generator and made to be 8-bit code words. The code words are then fed into the output buffer, i.e., FIFO. During the coding procedures, the module continuously monitors the bit streams and checks whether the budget is full. Once the budget is full, the signal Budgetfull goes high, which means the desired bit rate has been met.

3.2.7 Controller

The controller decides the time to make transitions from one coding step to another and generates enable signals for other modules. Initially, it enables the INIT module to initialize the coding system, then it repeatedly generate the enable signals for FPHcode, FScode and Threshold generator modules, until the signal Budgetfull goes high, and then the coding process ends.

The Verilog HDL modules of the architecture for the FZIC are designed. The simulation results show that the bit streams outputted from the architecture the same as the bit streams outputted from C++ programs. The architecture is synthesized to ALTERA APEX20K100QC240. The Synthesis results are shown in Table 3. The system clock of the FZIC achieves 45.0MHz. The DWT coefficients can be coded in real time.

Table 3 FZIC synthesized results

Image size	512 × 512
DWT coefficient	16bit (signed)
DWT levels	5
IOs	114
LCs	2991
Clock rate	45.0MHz

4 Hardware implementation of wavelet image coder

By combining the architectures of the 2-D DWT with the FZIC, a wavelet image coder is presented. The coding system is shown in Fig. 5. The image data captured by digital camera are stored in image memory. The data are then transformed to 5 levels by 2-D DWT. The coefficients are coded by the FZIC, and the bit streams are collected by the host computer. In the host computer, the coded data are then decoded and inversely transformed. The reconstructed images are displayed on the CRT.

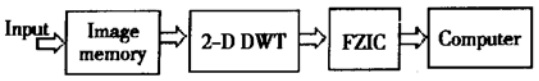

Fig. 5　Wavelet image coder

A demonstration PCB module has been built. The system clock of the PCB is set to 16MHz. The system can code 512 × 512 images in real time. The receiver programs are programmed in C^{++} language. The synthesized programs are downloaded to the CPLD chips and the images are coded and displayed in real time. The experiment result shows that the bit streams outputted from the coder the same as the bit streams outputted from the C^{++} programs. Both the architectures are efficient and suitable for VLSI implementation.

References

[1] Lewis A S, Knowles G. VLSI architecture for 2-D daubechr ies wavelet transform without multipliers. Electron Lett, 1991, 27(2): 171
[2] Vishwanath M, Owens R M, Irwin M J. VLSI architecture for the discrete wavelet transforms. IEEE Trans Circuit Syst, 1995, 42(5): 305
[3] Shapiro J M. Embedded image coding using zerotree of wavelet coefficients. IEEE Trans Signal Processing, 1993, 41(12): 3445
[4] Said A, Pearlman W A. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. IEEE Trans Circuit Syst Video Technol, 1996, 6(3): 243
[5] Su C Y, Wu B F. Image coding based on embedded recursive zerotree. ISMIP' 97, 1997: 387
[6] Lin W K, Ng B W-H, Burgess N. Reduced memory zerotree coding algorithm for hardware implementation. IEEE ICMCS' 99, June 1999: 231
[7] Shapiro J M. A fast technique for identifying zerotree in the EZW algorithm. ICASSP' 96, 1996: 1455

小波图像编码的 VLSI 实现

乔世杰　王国裕

(西安交通大学电子工程系，西安　710049)

摘要：设计了一种模块化的二维离散小波变换(2-D DWT) 的 VLSI 结构. 该结构可以实时完成小波变换, 且很容易扩展. 针对零树编码硬件实现方面的不足, 利用一种简单的顺序扫描方式和两个标志阵列, 设计了一种适合硬件实现的快速零树编码算法(FZIC) 和 FZIC 硬件实现的 VLSI 结构, 编写了 2-D DWT 和 FZIC 硬件结构的 Verilog HDL 模型, 并进行了仿真和逻辑综合. 结合 2-D DWT 和 FZIC, 实现了小波图像编码系统, 并用 ALTERA CPLD 成功进行了验证.

关键词：离散小波变换；零树编码；VLSI；Verilog HDL；CPLD
EEACC: 2570
中图分类号：TN432　　**文献标识码**：A　　**文章编号**：0253-4177(2002)07-0695-06

乔世杰　男, 博士研究生, 研究兴趣包括小波变换图像编码算法及其 VLSI 实现等.
王国裕　男, 教授, 博士生导师, 研究领域包括 CMOS 摄像器件、图像/声音压缩及通讯的 VLSI 实现, SOC 等.

2001-12-18 收到, 2002-03-11 定稿