

Hardware Design of On-Line Jawi Character Recognition Chip using Discrete Wavelet Transform

Zaidi. R, Rosli. S, Mashkuri. Y
 Department of Computer System and Technology
 University of Malaya, Malaysia
 zaidi@um.edu.my

Abstract

In this paper, we will discuss further on our previous works [1] on developing a recognition chip of Jawi characters using Chain Code Algorithm and problems occurred. An explanation of Jawi characters have been made in [1]. We will also discuss our second approach i.e using discrete wavelet transform (DWT) to extract coefficients from the characters based on their features and using these coefficients to develop a unique code. This unique code will be used to recognize it. Hardware model entities integration is presented in the last section. Paper orientation will be like this. First we will discuss on why we choose DWT & way of implement it and followed by the problems we faced in using chain code algorithm. The full implementation of the design will be explained in the last section, followed by a conclusion.

1. Why Discrete Wavelet Transform (DWT)

This paper will be our second paper on developing a specific chip of Jawi character Recognition but using different approach. Our first idea [1] on this work has meets several problems especially on preprocessing part, and we will explain these later in the next section.

Discrete Wavelet Transform is a tool with a discrete manner to extract wavelet coefficients which is also used by Mowlaei. etl [3] in their research on isolated arab/farsi character. The implementation of DWT in hardware modelling using VHDL was made by Zaidi .etl [2] and shows that it can be fit into this research as our main entity with a few modifications in the design(referto figure 7).

An equation below can describe DWT manner:

$$\Phi_{(sl)}(x) = 2^{-\frac{s}{2}} \Phi(2^{-s} x - l) \quad (1)$$

In order to get the code from different resolutions, a multiscalar tool will be used as shown in Equation 2 below:

$$W(x) = \sum_{k=-1}^{N-2} (-1)^k c_{k+1} \Phi(2x + k) \quad (2)$$

where $W(x)$ is the scalar function for wavelet analyzer Φ , and C_k is the wavelet coefficient. In this approach, a pyramid algorithm which is developed by Mallat [4] will be used. The feature of this algorithm is shown below:

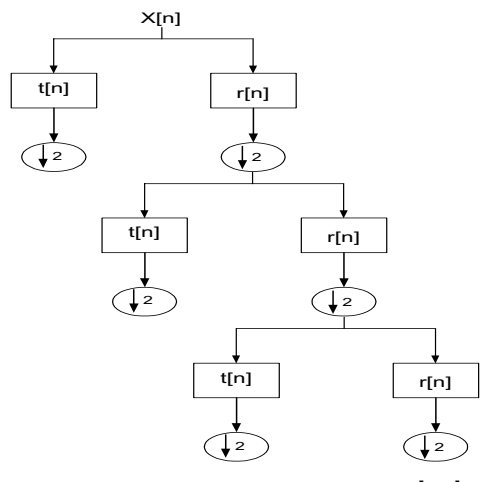


Figure 1. Pyramid Algorithm

where $t[n]$ and $r[n]$ is the highpass and lowpass function as explained by Edwards [5] which can be expressed below:

$$a_i = \frac{1}{2} \sum_{j=1}^N c_{2i-j+1} f_j \quad i=1, \dots, \frac{N}{2} \quad (3)$$

$$b_i = \frac{1}{2} \sum_{j=1}^N (-1)^{j+1} c_{j+2-2i} f_j \quad i=1, \dots, \frac{N}{2} \quad (4)$$

where a is the highpass function and b is the lowpass function.

In this work, coefficients produced by lowpass filter are useful and will be stored and only coefficients produced by highpass will be used in the next stage. For example if we have $X[n]$ with 8 parameters, it will produce 2 set of coefficients (i.e 4 lowpass & 4 highpass) and this 4 highpass will be processed and also produces 2 set of data (i.e 2 lowpass & 2 highpass) and this process will be carried on until we get a single size of data (i.e 1 lowpass & 1 highpass). Data with highfrequency contains hidden information and have to be processed more to extract it.

With all these characteristics and also implementation steps (to be discussed), we adamantly sure this DWT will be able to overcome the problems occurred in using Chain Code Algorithm.

1.1. Implementation Steps of Coefficients Extraction

The whole idea is to process the whole pixels in the image and extract only the **last** produced highpass coefficient for every step. For example if we have 600x600 pixel size of image, it will produce 150x150 set of 4x4 matrix of coefficients. At the end we only get 150 coefficients by diagonally extract them. Below is an algorithm on how it will be implemented.

1. Read image
2. Get image size
3. if image row is not modulo 4 then
 - 3.1 padding with zero by it differences
 end if
- 4.0 if image column is not modulo 4 then
 - 4.1 padding with zero by it differences
 end if
- 5.0 Get new image size
- 6.0 Divide the image into small matrices 4x4 in horizontally and vertically
 - 6.1 for each matrix compute DWT coefficient and stored in original locations.
 - 6.2 repeat 6.1 until all matrices process
- 7.0 fetch the coefficients using this rule location must (i) row modulo 4 is 0 (ii) column modulo 4 is 0
 - 7.1 Store these coefficient in new matrix

The DWT coefficients pictured us the essence of the image and also can be viewed as image in different dimension.

2. Problems of Chain Code Algorithm

The problems of applying Chain Code Algorithm in hardware implementation as our first attempt can be stated as below:

- (a) the manuscripts are the old ones and they carry a lot of noises and non-important items
 - our input of this online hardware is handwritten of old manuscript with a lot of degradation process taken place such a yellowish paper and dotted due to inadequate measurements of preservation processes as shown in Figure 2 and these will create lot of noises when it turn to digital format of 256 gray levels.



Figure 2. Original Manuscript of Hang Tuah Story

- (b) thinning process can't achieved 100% accurate due to noises
 - since the manuscripts have been degraded, they will give problems in thinning process because of inaccurate defining of character prime pixels as shown in Figure 3 and 4.

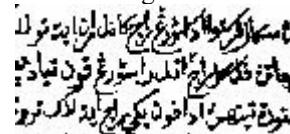


Figure 3 Before Thinning

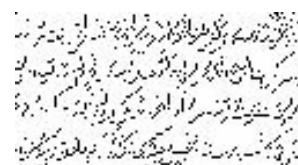


Figure 4. After thinning (image blurred because the effect of zoom in process)

- (c) identify unique code (accurate representation) will not achieve due to broken link of character image as shown in Figure 4. It is very hard to preserve the details in this situation where the noise and degraded items seem to be part of the image without sacrifices the features of character as concluded by Louisa Lam etl [6]

The above three problems are enough for us to abandon this algorithm and move to our second approach (as explained earlier in section 1.0).

3. Processing Algorithm

Figure 5 shows a process flow on how we will process our scanned image. In this flow chart, we will not process the secondaries separately because our DWT will process it as a one character and get its coefficient.

Scanner specification will be on 256 gray levels and will be parallel connected to our hardware and serial connection to computer as described in Figure 6.

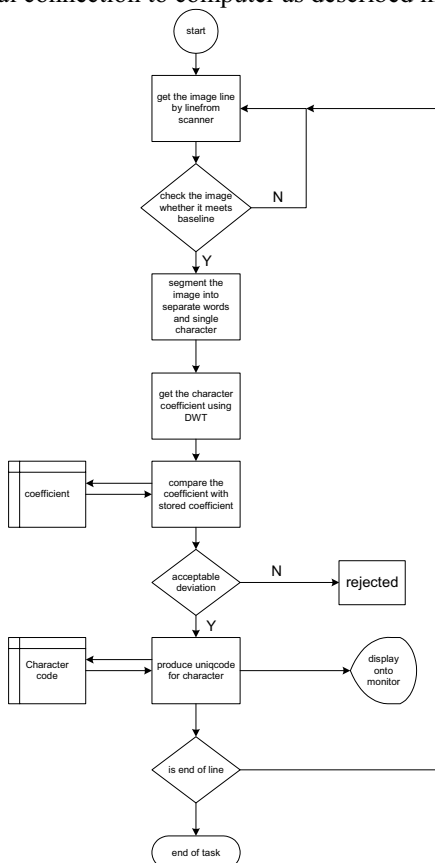


Figure 5. Process Flow of On-line Processing

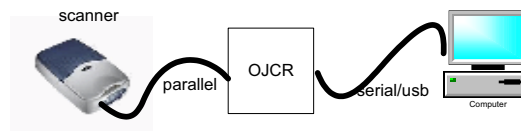


Figure 6. Connection between scanner, OJCR and Computer

4. Design of hardware

Several characteristics have to be considered before a design can be made. They were:

- (a) Data Format
 - image is scanned for 256 gray level. In this case, the format of data in is just 8 bits of positive integer with LSB
- (b) Bus Width
 - in transferring a data from one entity to another entity in the hardware we have to look into bus specifications with consideration made to speed. Since our data is only 8 bit, our bus will also be 8 bit in one-time transfer
- (c) Number of entities
 - a limited number of entities will be useful in keeping a high speed of processing, hence our design we will try to avoid of unnecessary entity.

A Combination of entities can be viewed in Figure 7. Our approach is to design and test these entities independently and then integrate them into actual design.

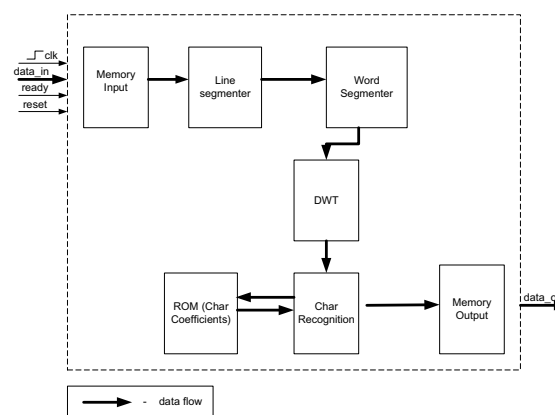


Figure 7. Combination of Entities

Our design implementation will be made using VHDL language and code listing below is our top Register

Transfer Logic RTL (represent the overview of the diagram in Figure 7.0)

-- On-line Jawi Character Recognition (OJCR)

entity OJCR is

```
port(clk : in bit;
     ready : in bit;
     reset : in bit;
     data_in : in bit_8;
     data_out: out bit_8;
);
end OJCR;
```

5. Result of DWT

A result of DWT for character ξ is shown in Figure 8(a). Only the numbers other than ± 4.0000 and ± 0.0000 will used in the comparison stage as shown in Figure 8(b)

4.0000	-0.0000	0.0000	0.0000	0.0000	4.0000	-	4.0000	-0.0000
0.0000	4.0000	-0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	-	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	0.0000	-0.0000	-0.0000	0.0000	-0.0000	-0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4.0000	-0.0000	0.0000	0.0000	0.0000	3.2500	-	3.7500	0.1071
0.0000	4.0000	-0.0000	0.0218	0.0289	-0.6754	0.2096	0.2405	
-0.0000	-0.0000	-0.0000	-0.0000	-0.0000	0.2872	-	0.0683	0.0523
-0.0000	-0.0000	-0.0000	0.6166	0.2279	0.5403	0.5150	-0.1970	
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-	0.7148	-0.0280
0.0000	0.0000	0.0000	0.2304	-0.6309	0.1870	0.0308	-0.0106	
0.0000	0.0000	0.0000	0.0000	0.0000	0.4516	-	0.0000	0.0000
0.0000	0.0000	0.0000	0.6589	0.2179	-0.0435	0.0000	0.0000	
4.0000	-0.0000	0.0000	-0.0482	-0.3388	2.7500	-	2.7500	-0.4314
0.0000	3.7500	-0.2740	0.4511	-0.6563	0.6254	0.2401	0.1432	
-0.0000	-0.0000	-0.0000	0.1319	-0.0379	0.1905	-0.6260	0.5157	
-0.0000	-0.2038	-0.4448	0.7469	-0.2193	-0.0478	1.0837	-0.4189	
0.0000	0.0000	0.0000	0.0000	0.0000	-0.6609	0.0405	0.1591	
0.0000	0.0000	0.0000	0.9221	-0.0715	0.0308	0.2324	-0.6415	
0.0000	0.0000	0.0000	0.1870	-0.0329	-0.0106	0.1870	0.4516	
0.0000	-0.2304	-0.6309	0.2304	0.0840	0.6435	0.6589	-0.0125	
4.0000	-0.0000	0.0000	0.2405	-0.2096	4.0000	-	4.0000	-0.0000
0.0000	2.7500	-0.1071	0.0000	0.0000	0.0000	0.0000	0.0000	
-0.0000	-0.0000	-0.0000	-0.4862	-0.5022	-0.0000	-	-0.0000	-0.0000
-0.0000	-0.0396	-0.7515	0.0000	-0.0000	-0.0000	0.0000	-0.0000	
0.0000	0.0000	0.0000	-0.0433	-0.6638	0.0000	0.0000	0.0000	
0.0000	-0.0433	-0.6638	0.0000	0.0000	0.0000	0.0000	0.0000	
0.0000	0.0000	0.0000	-0.0125	-0.6744	0.0000	0.0000	0.0000	
0.0000	0.6715	-0.6917	0.0000	0.0000	0.0000	0.0000	0.0000	
4.0000	-0.0000	0.0000	0.0482	0.3388	3.2500	4.0000	-0.0000	
0.0000	3.2500	0.2740	0.7439	-0.0970	-0.4177	0.0000	0.0000	
-0.0000	-0.0000	-0.0000	-0.6865	-0.5167	0.6066	-	-0.0000	-0.0000
-0.0000	-0.3507	-0.1098	0.0181	-0.2495	0.3315	0.0000	-0.0000	
0.0000	0.0000	0.0000	-0.0433	-0.6638	0.0000	0.0000	0.0000	
0.0000	-0.0433	-0.6638	0.4845	-0.6589	-0.0125	0.0000	0.0000	
0.0000	0.0000	0.0000	-0.2304	-0.6309	-0.6744	0.0000	0.0000	
0.0000	0.1870	-0.0329	0.1870	-0.0329	0.0000	0.0000	0.0000	
1.0000	1.2419	0.4003	0.4003	0.0488	0.5000	4.0000	-0.0000	
0.0488	1.0000	1.2419	0.7182	0.3089	0.4963	0.0000	0.0000	
0.2602	-0.6150	0.5610	0.5610	-0.2061	0.2746	-	-0.0000	-0.0000
0.2061	0.2602	-0.6150	0.3039	-0.0808	-0.0058	0.0000	-0.0000	
-0.1744	0.0435	0	0	0	0.0435	0	0.0000	0.0000
0	-0.1744	0.0435	0	0	0	0.0000	0.0000	
0	-0.4845	0.1744	0.1744	0.4410	0	-	0.0000	0.0000
0.4410	0	-0.4845	0.4845	0.6589	-0.2179	0.0000	0.0000	

Figure 8(a). Before Extraction

3.7500	-0.2740	-0.2038	-0.4448	-0.2304	-0.6309	2.7500	-0.1071	-0.0396	-0.7515
-0.0433	-0.6638	0.6715	-0.6917	3.2500	0.2740	-0.3507	-0.1098	-0.0433	-0.6638
0.1870	-0.0329	1.0000	1.2419	0.4003	0.0488	1.0000	1.2419	0.2602	-0.6150
-0.2061	0.2602	-0.6150	-0.1744	0.0435	-0.1744	0.0435	-0.4845	0.1744	0.4410
-0.4845	3.2500	-0.0218	0.0289	-0.6754	0.2872	-0.6166	0.2279	0.5403	-0.2304
-0.6309	0.1870	0.4516	-0.6589	0.2179	-0.0435	-0.0482	-0.3388	2.7500	-0.4511
0.6563	0.6254	0.1319	-0.0379	0.1905	0.7469	-0.2193	-0.0478	-0.6609	0.9221
-0.0715	0.0308	0.1870	-0.0329	-0.0106	-0.2304	0.0840	0.6435	0.2405	-
0.2096	-0.4862	-0.5022	-0.0433	-0.6638	-	-0.0125	-0.6744	0.0482	
0.3388	3.2500	0.7439	-0.0970	-0.4177	-0.6865	-0.5167	0.6066	-0.0181	-
-0.2495	0.3315	-0.0433	-0.6638	0.4845	-0.6589	-0.0125	-0.2304	-0.6309	-
0.6744	0.1870	-0.0329	0.4003	0.0488	0.5000	0.7182	0.3089	0.4963	
0.5610	-0.2061	0.2746	-0.3039	-0.0808	-0.0058	0.0435	0.1744	0.4410	
0.4845	0.6589	-0.2179	3.7500	0.1071	-0.2096	0.2405	0.0683	0.0523	
0.5150	-0.1970	0.7148	-0.0280	0.0308	-0.0106	2.7500	-0.4314	-0.2401	
0.1432	-0.6260	0.5157	1.0837	-0.4189	0.0405	0.1591	-0.2324	-0.6415	0.1870
0.4516	-0.6589	-0.0125							

Figure 8(b). After Eliminating Unnecessary numbers

A process of extracting and producing a unique number from this result will be conducted in later stage (as shown in figure. 5). In this stage, the unique numbers will be made into the same size thus will make the comparison easier.

6. Conclusion

Accuracy of our algorithm is depends on the pre processing stage taken before the DWT that involves line, word and char segmentation. If these processes are in the right shape, there will be no trade off between performance and accuracy. We hope with this new technique and experience earned by using Chain Code Algorithm, we can realize our design into the workable chip.

7. References

- [1] Z. Razak, O.A Rahim, M. Y. Idna, N. M. Noor, M. Yaacob, "VHDL Implementation of Jawi Character Recognition via Chain Code Algorithm", in Proceedings of SPIE Vol. 5286 Third International Symposium on Multispectral Image Processing and Pattern Recognition, edited by Hangqing Lu, Tianxu Zhang, (SPIE Bellingham, WA 2003) pp. 457.
- [2] Z. Razak, M. Yaacob, "VHDL Development of Discrete Wavelet Transformation", Malaysian Journal of Computer Science, Vol. 15, No.1, June 2002, pp. 84-92
- [3] Mowlaei, A.; Faez, K.; Haghghat, A.T, "Feature extraction with wavelet transform for recognition of isolated handwritten Farsi/Arabic characters and numerals", 14th International Conference on Digital Signal Processing, 2002, Volume: 2, 1-3 July 2002,Pages:923 – 926
- [4] Mallat, S., "A Theory for Multiresolution Signal Decompositions, The Wavelet Representationn," IEEE Trans. Pattern Analysis And Machine Intelligence, Vol 2, pp. 674-693, 1989.

[5] Edwards. T., “Discrete Wavelet Transforms: Theory And Implementation.” Research Report, Stanford University. (September 1991) .

[6] Louisa Lam etl, “Thinning Methodologies – A Comprehensive Survey”, IEEE Transcation on Pattern Analysis and Machine Intelligence, Vol 14, No. 9, September 1992, pp. 869-885.