Energy Saving Analysis for the Wavelet Transform Processing

Héctor Silva-López[1], Sergio Suárez Guerra

Center of Computational Research, National Technical Institute, México, A.P. 75-476, C.P. 07738, Zacatenco, DF, México.
hsl@fis.cinvestav.mx[1], ssuarez@cic.ipn.mx

Abstract. The applications of the wavelet transform which are executed in mobile devices usually operate with batteries of limited capacity. The problem to be solved is how to increase the efficiency of the batteries used so that their time life can be measured in days instead of hours. On the other hand, the ability to adapt the changes of the work charge conditions for a real-time system with energy saving is handled in this article using an algorithm for graphic applications (by means of discrete wavelet transform) which presents a considerable energy saving in its processing. These applications will be executed in a variable voltage processor. The algorithm selects the way of execution of each level of discrete wavelet transform so that the energy saving is maximized and that the deadline is not lost. The problem is presented as a problem of dynamic optimization with discrete constraints. The experimental results were focused on images that showed that the filters Haar, Daub4 and Daub6 had a smaller execution time for the algorithm presented. These filters had a smaller execution time for smaller images size. The energy saving is greater for smaller images and smaller for bigger ones. An extra saving is obtained if the algorithm is executed in a shared memory environment.

1 Introduction

Although wavelet have their roots in approximation theory [7] and signal processing [14], they have recently been applied to many problems in computer graphics. These graphics applications include image editing [4], image compression [12], and image querying [3]. These applications impose strict quality of service requirements in the form of timing constraints. Ignoring energy consumption, operating the CPU at its highest speed operation quickly drains the batteries. Thus there is a tradeoff between reduced energy consumption and quality of service.

Voltage scaling technology has the potential to exploit such variability in the case of meeting timing constraints. By adjusting the operating voltage of the processor, the energy consumption and speed can be controlled [1]. Power regulator and variable voltage

A. Gelbukh, S. Suárez, H. Calvo (Eds.)
Advances in Computer Science and Engineering
Research in Copmuting Science 29, 2007, pp. 285-294

Received 02/07/07
Acepted 19/10/07
Final Version 20/10/07

processors with response times in the microseconds range are available [10]. Fast response time makes it practical to dynamically adjust the voltage at run time.

Recently, researches have attempted to apply Dynamic Voltage Scaling (DVS) to video decoding to reduce power [5, 8, 9, 11]. These studies present approaches that predict the decoding time of incoming frames or Group of Pictures (GOPs), and reduce or increase the processor setting based on this prediction. As a result, idle processing time, which occurs when a specific frame decoding completes earlier than its playout time, is minimized. In [6] an alternative method called Dynamic is proposed as an improvement to these techniques. The Dynamic approach is designed to perform well even with high motion videos by dynamically adapting its prediction model based on the decoding experience of the particular video clip being played. The same authors present another alternative method called frame data computation aware (FDCA) in [2]. FDCA dynamically extracts useful frame characteristics while a frame is being decoded and uses this information to estimate the decoding time.

The objective of this work is to develop a new DVS technique for the energy saving. An algorithm that can be used for any graphic application that uses the wavelet transform for its implementation is presented. Some energy saving will be obtained for each level of the transformed one, and the sum of energy savings of all the levels is considered the total energy saving.

This article is structured as follows: In section 2, a general description of wavelet transform is given. In section 3, the problem is formulated and the algorithm is described. Section 4 shows a practical example to demonstrate the function of the algorithm. The simulation of experiments is described in section 5. And finally, the conclusions are described in section 6.

2 Wavelet Transform

Of the many processes available for image compression, two of the most popular transformations are the Discrete Cosine Transform (DCT) used in the common JPEG format, and the Discrete Wavelet Transform (DWT) used in the newer JPEG 2000 format. The DWT differs from the traditional DCT in several fundamental ways. The DCT operates by splitting the image into 8x8 blocks that are transformed independently [13]. Through this transformation process, the energy compaction property of the DCT ensures that the energy of the original data is concentrated in only a few of the transformed coefficients, which are used for further quantization and encoding [15]. It is the discarding of the lower-energy coefficients that result in image compression and the subsequent loss of image quality. Unfortunately, the rigid 8x8 block nature of the DCT makes it particularly susceptible to introducing compression artifacts (extraneous noise) around sharp edges in an image. This is the “halo effect” seen in over compressed web images. Because the

artifacts become more pronounced at higher compression ratios, JPEG's suitability for line drawings and cartoon-like images is significantly impaired.

In contrast to the DCT, the DWT operates over the entire image, eliminating artifacts like those caused by the 8x8 DCT blocking. Like the DCT, the fundamental wavelet transform is completely reversible, meaning that if the forward and reverse transforms are applied in sequence, the resulting data will be identical to the original. In addition, the DWT is based on sub-band coding where the image is analyzed and filtered to produce image components at different frequency sub-bands [18]. This produces significant energy compaction that is later exploited in the compression process. The wavelet's two-dimensional nature results in the image visually being divided into quarters with each pass through the wavelet transformation. A key effect of this transformation is that all of the high pass quadrants in the image contain essentially equivalent data [16]. In the field of wavelets, the Haar wavelet is traditionally used for rudimentary image compression because of its algorithmic simplicity and low computational complexity due to an integer based design [18].

When the wavelet is applied and through its filtering process produced the scaling function coefficients (low frequency) and wavelet coefficients (high frequency). From the application of the Haar wavelet, it is evident that the scaling function coefficients is simply the average of two consecutive pixel values, while the corresponding wavelet coefficient is the difference between the same two pixel values. The scaling function coefficients appear to contain all the image data, while the wavelet coefficients appear to be zero (black).

Figure 1 presents the process of the wavelet transform. It begins with an original image, when the transformed by row occurs, the image is divided in two halves, the first half corresponds to the scaling function and the second half corresponds to the wavelet transform; each is called half sub-band.

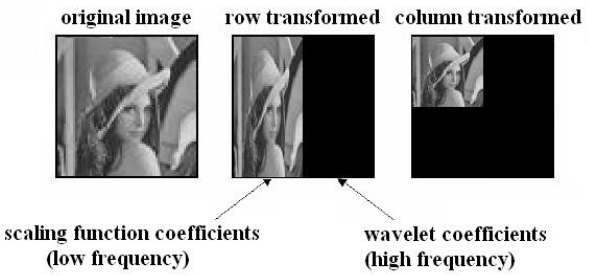

Figure 1.First Level Transform.

When the following transformed by column is performed, this image is divided in four equal parts or sub-bands. The first quadrant or sub-band corresponds to the scaling function and the other three sub-bands, to the wavelet transform; in this first step, four sub-

bands are obtained. This process is repeated for the scaling function for the second level and so on successively. In the end, the image is seen as it is shown in figure 2.

3 Formulation of the Problem

The problem can be formulated as follows. Each time a new image arrives at or leaves the system, the problem is to determine the mode (speed) of execution of the sub-bands such that no sub-band misses its deadline and the energy savings of the system is maximized. Each image in the system execute in a discrete voltage/frequency processor. Note that a solution to this problem must be computed each time a new image arrives or leaves the system, therefore a solution with probably cause deadlines to be missed.

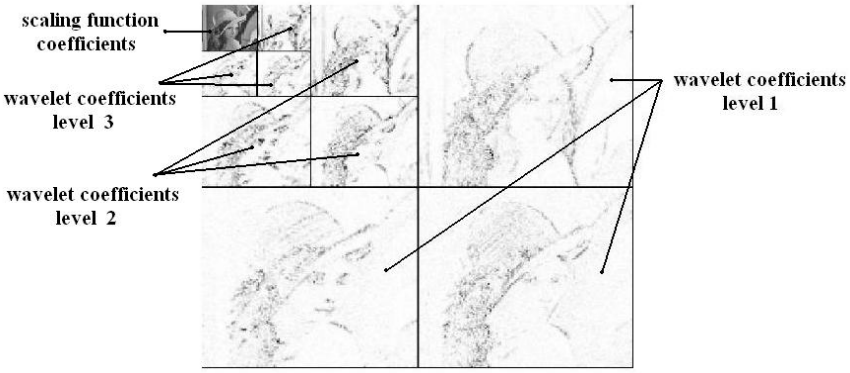

Figura 2. Wavelet Transform for three levels.

The problem is formulated as:

$$\max_{\{u(k)\}_{k=0}^{2}} J = \sum_{k=0}^{2} S_k[u(k)]$$

subject to $x(k+1) = x(k) - u(k)$

$x(0) = 1.0,$

with : $x(3) = 0,$

$u(k) \le x(k),$

$u(k) \in \{1.0, 0.8, 0.6, 0.4, 0.15\}$ *for* $k = 0,1,2$

where

$k = corresponds\ to\ sub-bands$

$S_k[u(k)] = energy\ saving$

$x(k) = \text{variable of state}$

$u(k) = control\ variable\ (speed\ to\ select)$.

Bellman's optimality principle is used by compute the state variable in each stage, as follows:

Step 1: x(3) is calculated as follow:

$$J_3^*\{x(3)\} = 0$$

Step 2: x(2) $\in \{1.0,0.8,0.6,0.4,0.15\}$ is:

$$J_2^*\{x(2)\} = \max_{u(2)}\{S_2[u(2) + J_3^*\{x(2) - u(2)\}\},$$

where:

$$x(3) = 0 = x(2) - u(2), u(2) \le x(2), u(2) \in \{0.15,0.4,0.6,0.8,1.0\}$$

Step 3: x(1) $\in \{1.0,0.8,0.6,0.4,0.15\}$, the equation that corresponds is:

$$J_1^*\{x(1)\} = \max_{u(1)}\{S_1[u(1) + J_2^*\{x(1) - u(1)\}\},$$

where:

$$u(1) \le x(1),\ u(1) \in \{0.15,0.4,0.6,0.8,1.0\}$$

Step 4: x(0)=1.0 is:

$$J_0^*\{1.0\} = \max_{u(0)}\{S_0[u(0) + J_1^*\{1.0 - u(0)\}\},$$

where:

$$u(0) \le 5,\ u(0) \in \{0.15,0.4,0.6,0.8,1.0\}$$

4 Example of the Algorithm

To illustrate the execution of the proposed algorithm, we consider the image Lena and we use five discrete speed levels {1.0, 0.8, 0.6, 0.4, 0.15}, in a discrete voltage/frequency processor. The CPU utilization can be measured while the system is under various states of loading. Obviously there is no way (yet) to measure CPU utilization directly. CPU utilization must be derived from measured changes in the period of the background loop. The average background loop period should be measured under various system loads and then the CPU utilization can be obtained. The CPU utilization is defined as the time 'not' spent executing the idle task (is a task with the absolute lowest priority in a multi-tasking system). The amount of time spent executing the idle task can be represented as a ratio of the period of the idle task in an unloaded CPU to the period of the idle task under some known load.

% time in the idle task = (average period of the background task with no load) * 100%
/ (average period of the background task with some load) [1]

Based on DVS technique, we adjust processor speed for each sub-band with slack reclamation.

Table 1 shows the time in microseconds for each sub-band of the image. Five levels are considered. The average execution time of the idle task is 195 microseconds, this time was calculated in the same was as the one presented in [17] for which an Analyzer of Logical State was used to measure the data that flow through the data and direction bus when a task is being executed in background. The time obtained is the average period of the task in background without load. Immediately, the time of the idle task with load is obtained, when each sub-band of the second column of table 1 is executed, and finally, the third column is obtained from equation 1. The fourth column is the result of subtracting the third column from 100 in order to obtain the use of the CPU.

After the executing the first sub-band of the level 1, to the maximum speed, we can obtain the percentage of utilization of this sub-band, the percentage idle is assigned to sub-bands 2 and 3, the executing speed the these sub-bands depending the result of first sub-band. Later, for the others levels, calculating the percentage of utilization for the first sub-band of each level and assigning the percentage idle to the two following sub-bands. In the table 2 is presented the percentage of energy saving level by level. The total energy saving of the all image is 44.19 %.

Sub-bands	T (microseconds)	% Idle	% CPU
1	974	20.02	79.97
2	535	36.49	63.55
3	388	50.26	49.74
4	859	22.70	77.30
5	417	46.76	53.24
6	353	55.24	44.76
7	595	32.77	67.23
8	368	52.99	47.01
9	287	67.94	32.06
10	445	43.82	56.18
11	313	62.30	37.70
12	267	73.03	26.96
13	342	57.02	42.98
14	281	69.39	30.61
15	247	78.95	21.05

Table 1. Sub-bands vs. Average background loop Period.

The energy saving per level of the third column of table 2 was the one that was obtained per sub-band. For the fourth column, the percentage of the total energy saving per level was calculated on the basis of the maximum saving that could be obtained per level with respect to the obtained from the third column. For example, for the first level, the

maximum saving that could be obtained is 20,02 for the first sub-band, 36,49 for second sub-band and 50,26 for the third sub-band, which gives a maximum total of 106.77. The total saving obtained per sub-band was of 40,04; then the percentage of saving energy is obtained from a rule of three which is 37.50% for the first level, and so on for the other levels.

Level	Sub-band	% Energy Saving	Total
1	1	0	
	2	20.02	
	3	20.02	= 37.50 %
2	4	0	
	5	22.70	
	6	22.70	= 36.40 %
3	7	0	
	8	32.77	
	9	32.77	= 42.64 %
4	10	0	
	11	43.82	
	12	43.82	= 48-91%
5	13	0	
	14	57.02	
	15	57.02	= 55.53 %

Table 2. Energy Saving for level.

5 Simulation Experiments

The algorithm proposed on the basis of our criterion of optimization presented in this paper is verified in this experiment simulation, using different images and different wavelet filters. The objective of this experiment simulation is to measure the energy saving for different image sizes. Each image size will be observed for different wavelet filters. It can be observed that different execution times are obtained if different wavelet filters are used. The execution time for each level is physically measured in microseconds, having used a Laptop Sony, model VGN-T350P, with a processor Intel Centrino at 3,2 GHZ, with 512MB of RAM and running in the Operating System Fedora Linux version 5.0. The used function to measure the time is psched_get_time.

Figure 3 presents the execution times obtained when performing the wavelet transform for each level, using different filters. It can be seen that Villa2 filter obtained the worst time and that the Haar filter obtained the best one, within the first 3 levels. For the rest of the levels, all the filters tended to have an equal time, tending to zero, mainly because the transformed in level 5 was made only for a block of 32x32 pixels. This Figure 3 shows that the three best filters are the Haar, Daub4 and Daub6, and it is from these filters that the execution time for each level, using different image sizes as shown in figure 4, will be obtained.

Figure 4 shows that the smaller the image, the smaller execution time per level. This same behavior can be observed for the filters Daub4 (figure 5) and Daub6 (figure 6). The difference between the execution times for the three types of filters is conserved in these three figures. It is important to mention that the times (for each filter) were almost equal when they were proven with other images, which lead to conclude that these times are constant for any type of image of the same size.

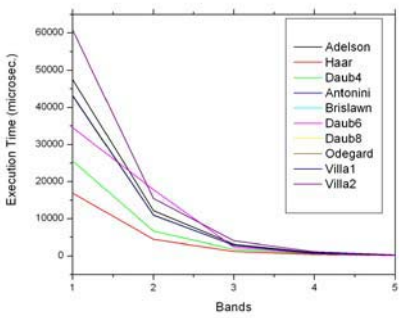

Figure 3. Execution time for different filters

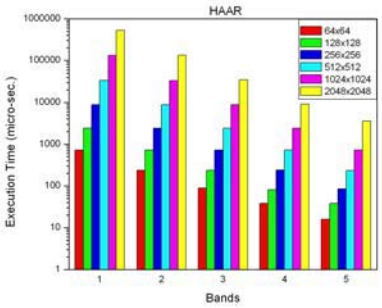

Figure 4. Execution time for filter Haar.

The energy saving for these three filters, using different image sizes, is presented in figure 7. For small image sizes, a greater energy saving is observed and for bigger image sizes a smaller energy saving is obtained.

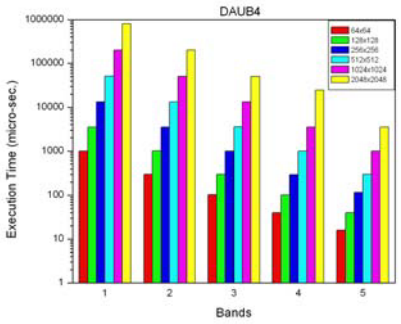

Figure 5. Execution time for filter Daub4.

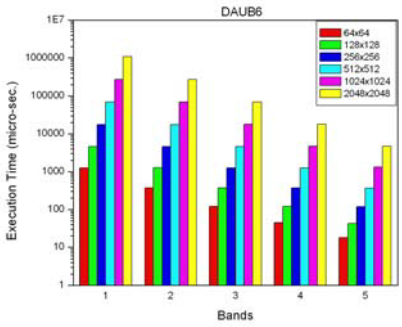

Figure 6. Execution time for filter Daub6

The last step consisted of running the algorithm in parallel form in order to observe how much the energy saving would be reduced in this environment. The algorithm was

programmed to be executed by shared memory. The Pthread library, which fulfills with the standards POSIX, and it allowed working with two threads of execution at the same time. Figure 8 shows that there was an extra energy saving of the order of 28.77% in average.

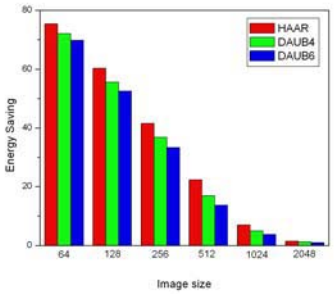

Figure 7. Energy saving for different size images.

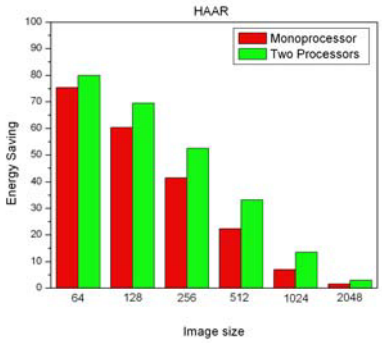

Figure 8. Energy saving for two threads.

6 Conclusions

A method of optimization for the discrete wavelet transform applied to images running in a processor of variable speed is being proposed in this article. The problem is presented as a linear problem with discrete constraints. The proposed approximate solution is based on the Bellman equation. An energy saving of the 44.19% is obtained for an image. Comparing the execution times among the different filters it is observed that the Villa1 filter is the worst and that the Haar filter is the best, but approaching the fifth level, the execution time tends to zero for all the filters. Taking into account only the three best filters, the execution time for different image sizes was obtained; thus observing that for small image sizes, the smaller execution time was obtained. The execution time for other images was also obtained and the results lead to conclude that these times are equal for any type of image of the same size. The energy saving for different image sizes was obtained, thus concluding that a greater saving energy occurs for small image sizes, whereas for a bigger size, a smaller energy is saved. Finally, when executing the algorithm in a shared memory environment, an extra saving of the order of 28.77% in average was obtained.

References

[1] A. Chandrakasan, S. Sheng and R. W. Brodersen, "Low-power CMOS digital design", IEEE Journal of Solid State Circuirs, 27, pp. 473-484, April 1992.

[2] B. Lee, E. Nurvitadhi, R. Dixit, C. Yu, and M. Kim, "Dynamic Voltage Scaling Techniques for power efficient video decoding", Journal of Systems Arquitecture, pp. 633-652, Available online 18 April 2005.

[3] Charles E. Jacobs, Adam Finkelstein, and David h. Salesin, "Fast multiresolution image querying", In Proceedings of SIGGRAPH 95, pages 277-286, ACM, New York 1995.

[4] Debora Berman, Jason Bartell, and David Salesin, "Multiresolution painting and compositing", In Proceedings of SIGGRAPH 94, pages 85-90, ACM, New York, 1994.

[5] D. Son, C. Yu, and H. Kim, "Dynamic Voltage Scaling on MPEG Decoding", International Conference of Parallel and Distributed Systems (ICPADS), June 2001.

[6] E. Nurvitadhi, B. Lee, C. Yu, and M. Kim, "A Comparative Study of Dynamic Voltage Scaling Techniques for Low-Power Video Decoding", International Conference on Embedded Systems and Applications (ESA), Jun 23-26, 2003.

[7] Ingrid Daubechies, "Orthonormal bases of compactly supported wavelets", Communications on Pure and Applied Mathematics, 41(7): 909-996, October 1988.

[8] J. Pouwelse, K. Langendoen, R. Lagendijk, and H. Sips, "Power-Aware Video Decoding", Picture Coding Symposium (PCS'01), Seoul, Korea, April 2001.

[9] J. Pouwelse, K. Langedoen, and H. Sips, "Dynamic Voltage Scaling on a Low-Power Microprocessor", 7th ACM Int. Confe, on Mobile Computing and Networking (Mobicom), pp. 251-259, Rome Italy, July 2001.

[10] M. Fleischmann, "Crusoe power management reduces the operating power with LongRun", in Hot Chips 12, Aug. 2000.

[11] M. Mesarina and Y. Turner, "Reduced Energy Decoding of MPEG Stream", ACM/SPIE Multimedia Computing and Networking 2002 (MMCN'02), San Jose CA, 18-25 January 2002.

[12] R. Devore, B. Jawerth, and B. Lucier, "Image compression Through wavelet transform coding", IEEE Transactions on Information Theory, 38(2):719-746, March 1992.

[13] Santa-Cruz, D, T. Ebrahimi, J. Askelöf, M. Larsson and C. A. Christopoulos, "JPEG 2000 Still Image Coding Versus Other Standards," *Proceedings of SPIE* 89 *45th Annual Meeting, Applications of Digital Image Processing XXIII*, Vol. 4115, San Diego, CA, July 30-August 4, 2000, pp. 446-454.

[14] Stephane Mallat, "A theory for multiresolution signal decomposition: The wavelet representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674-693, July 1989.

[15] Subramanya, S.R., "Image Compression Techniques," *IEEE Potentials*, Vol. 20, No. 1, February/March 2001.

[16] Topiwala, P.N., Wavelet Image and Video Compression, Kluwer Academic Publishers, 1998.

[17] Trader Michael, "Embedded Real Time Techniques for Calculating CPU Utilization", Spring Embedded Systems Conference, Course ESC-449, San Francisco, USA, March 2004.

[18] Villasenor, J., Belzer, B. and J. Liao, "Wavelet Filter Evaluation for Image Compression", *IEEE Transactions on Image Processing*, Vol. 4, No. 8, pp. 1053-1060, August 1995.

[19] Welstead, S, Fractal and Wavelet Image Compression Techniques, SPIE Optical Engineering Press, 1999.