Tree-Structure Architecture and VLSI Implementation for Vector Quantization Algorithms

Chung-Wei Ku, Liang-Gee Chen[1], Tzi-Dar Chiueh, and Her-Ming Jong
Room 332, Department of Electrical Engineering
National Taiwan University, Taipei, Taiwan, R. O. C.
TEL:(886)2-363-5252 ext. 332
FAX:(886)2-363-8247
email: f80157@cc.ee.ntu.edu.tw

abstract

A tree-structure based architecture for vector quantization (VQ) is proposed and implemented by VLSI in this paper. The proposed folded-tree architecture is designed by 0.8 μm CMOS VLSI technology. The die size is 4.65×5.21 mm^2 and estimated clock rate is about 34MHz, which satisfies most real-time applications. Since it is basically a mean squared error computation circuit, various kinds of VQ algorithms can apply on the proposed architecture.

Introduction

In the future, video compression will be broadly applied in many situations, from very low bit-rate video telephone to high quality HDTV. Vector quantization (VQ) is deeply researched and discussed for image coding recently. For the sake of real-time requirements, it is necessary to design a specific hardware with high-speed processing ability for VQ algorithms. Many variations of VQ algorithms have been proposed to improve the performance. Most of these algorithms either design a codebook with particular structure to speed up the encoding process, or divide the whole codebook into many sub-codebooks and the searching within a smaller sub-codebook costs much less time. However, many variants can be thought as the VQ algorithms with "memory" and there are strong relationships between current and previous coding tasks. The implementation of VQ should consider the properties of data dependency and be more flexible to many applications.

The operation common to all variants of VQ is the computation of mean squared error (MSE) between input data and VQ codewords. To calculate MSE as soon as possible, the computation element with fast operation ability and the support of smooth data flow are both important. Since VQ is both "I/O bound" and "computation bound" with poor data reuse, it is not easy to implement VQ in hardware. Some available works include bit-serial approaches [1], multiply and accumulate (MAC) approach [2], and systolic array approach [3]. In order to solve the realization problems such as large I/O bandwidth, complicated hardware, and irregular data flow, etc., most of the previous approaches change data sequence to alleviate these problems while sacrificing latency and throughput at the same time. Jehng, Chen, and Chiueh [4] have developed a tree-structured architecture for block matching motion estimation algorithm. In this paper, this structure is extended to VQ algorithms and the proposed architecture is designed by VLSI at last.

Previous Algorithms

VQ is a mapping process from input vector (generally 4 × 4 block for image coding) to some typical patterns which are generated previously. Many kinds of VQ algorithms have been developed to reduce the coding error or shorten the encoding time. Full-search and tree-search VQ are the most fundamental operations among all the variations.

The basic operation of full-search VQ can be described as follows:

```
Loop 1:  For each of the N codewords
Loop 2:      For each part of the K dimensions
                 Accumulate the squared error.
```

The algorithm of tree-search VQ is very similar except that the first loop is a binary search process rather than the sequential search in full-search. Tree-search VQ decreases the encoding time significantly but the resulted performance becomes suboptimal compared with full-

[1]Prof. Chen is working for AT&T Bell Lab. currently.

[2]This work is supported by National Science Council, R. O. C., under Grant NSC-83-0404-E-002 -008.

search algorithm. In other methods, for example, finite-state VQ algorithm, the searching range is reduced by the relationship of state-functions. Full-search or tree-search is still necessary but the encoding time is much shorter due to the small size of state-codebook. In addition, the result of current encoding determines which state-codebook should be chosen for the next encoding. As a result, memory access becomes irregular and the control strategy is more complex. There are similar situations in other VQ algorithms with memories. The implementation of such VQ algorithms should offer short latency which makes processing efficient and controller simple. In other words, short latency is beneficial for the algorithms with irregular memory access and complicated control procedures because next task can not begin until the result of current task is produced. Intuitively a binary tree-structure gives the shortest latency.

Proposed Architecture

A. Full-Tree Structure and Previous Approaches

The basic operation in VQ is the calculation of mean squared error (MSE). Full-search VQ can be formulated as follows:

$$\begin{aligned} & \min_i \sum_{k=0}^{15}(X_k - C_{i,k})^2 \\ = & \min_i \sum_{k=0}^{15}(X_k^2 - 2X_kC_{i,k} + C_{i,k}^2) \\ = & \max_i[(\sum_{k=0}^{15} X_kC_{i,k}) - \frac{1}{2}\sum_{k=0}^{15} C_{i,k}^2] \\ = & \max_i[(\sum_{k=0}^{15} X_kA_{i,k}) + B]. \end{aligned} \quad (1)$$

Where k means the kth dimension and i means the ith codeword. X is the input vector and C is the codeword. $A_{i,k} = C_{i,k}$ and $B = -\frac{1}{2}\sum_{k=0}^{15} C_{i,k}^2$. Similar result can be obtained for tree-search VQ:

$$\begin{aligned} & sign[\sum_{k=0}^{15}(X_k - C_{0,k})^2 - \sum_{k=0}^{15}(X_k - C_{1,k})^2] \\ = & sign[\sum_{k=0}^{15}(2X_k - C_{0,k} - C_{1,k})(C_{1,k} - C_{0,k})] \\ = & sign[2\sum_{k=0}^{15} X_k(C_{1,k} - C_{0,k}) - \sum_{k=0}^{15}(C_{0,k}^2 - C_{1,k}^2)] \\ = & sign[\sum_{k=0}^{15} X_k(C_{1,k} - C_{0,k}) - \frac{1}{2}\sum_{k=0}^{15}(C_{0,k}^2 - C_{1,k}^2)] \\ = & sign[(\sum_{k=0}^{15} X_kA_k) + B], \end{aligned} \quad (2)$$

where k means the kth dimension and X means the input vector. C_0, C_1 are the left and right codewords at some level of the tree, and $A_k = (C_{1,k} - C0, k)$, $B = -\frac{1}{2}\sum_{k=0}^{15}(C_{0,k}^2 - C_{1,k}^2)$. The sign in equation (2) determines the next search is toward to left or right.

Clearly the operations of full-search and tree-search VQ can be expressed by the sum of products from the above formulations. In order to compute the summation of absolute errors, Jehng [4] has proposed a tree-structured architecture for block matching algorithm. A similar architecture is applied for VQ with little modifications, as shown in Figure 1. At the bottom, there are sixteen multipliers and each multiplier calculates the product of input data X and codeword A_i in each dimension, as described in equation (1) and (2). A binary adder-tree generates the mean squared error as soon as possible because it has the smallest height. At last, the offset term, B, is added to produce the correct result. Unfortunately, the full-tree architecture can not be implemented due to its huge input bandwidth and chip size. Large amount of memory modules and pin count are caused and it is impossible to fulfil the full-tree architecture. Some previous works [1][2] change data flow for implementation. In Figure 2 (a) the data flow of bit-serial approach is demostrated. The sixteen multipliers are substituted by serial-parallel multipliers. As a result, the bandwidth of input and the complexity of multipliers are both reduced. The disadvantages of bit-serial approach is its long latency and the overhead of those circuits for data format reordering. Similarly, the data flow diagram of MAC approach is shown in Figure 2 (b). There is a skewing circuit before the MAC computation core for bit-level pipeline which increases MAC's throughput. MAC approach also has the problems of long latency and extra circuit overhead. Besides, for all the block processing algorithms, line delay element is essential because most of the image data is transmitted or stored in raster scan format. Both bit-serial and MAC approaches must convert these raster scanned data into sequential order and calculate these input one by one, as depicted in Figure 2 (c). It is obvious that the data flow is not smooth since the bottleneck may occur in the multiplexer. We will propose another strategy to solve these problems and keep the flexibilities for ordinary VQ algorithms.

B. Tree-Folding and Interleaving Methods

Another approach called "tree-folding" is suggested and employed in this paper. Figure 3 displays the $\frac{1}{4}$ folded tree and its data flow. In fact, the MAC approach can be regarded as the $\frac{1}{16}$ folded-tree. The proposed tree-folding technique lowers both input bandwidth and hardware complexity at the same time. Since there is no data skewing or reordering in this approach, short laten-

cy is guaranteed. Sixteen interleaved memory modules is necessary for full-tree structure but the $\frac{1}{4}$ folded-tree needs only four modules. Each pixel in screen is stored in one memory module and the input of the architecture is supported by these interleaved memories. These modules are just the same as the line delay elements, which exists originally for block processing algorithms such as VQ or others. Taking advantage of these line delays as interleaved memories, we make data flow more regular without any extra circuits.

Pipeline interleaving can increase the throughput of the system, many papers have discussed about it [1][4]. It can be applied on the proposed architecture, too. However, it makes control complex and needs more extra circuits for data buffering. Besides, pipeline interleaving can not be executed in the algorithms with strong data dependency, such as finite-state VQ algorithms. Short latency is more important in these cases because high throughput is still available under simple control strategies.

C. Efficiency of the Proposed Architecture

Compared with bit-serial and MAC approaches, the proposed folded-tree architecture has the shortest latency and the smoothest data flow, as mentioned before. For those algorithms with irregular memory access, high-speed processing requires short latency. The latency of bit-serial or MAC approach is seriously prolonged by skewing or reordering time. Besides, the computation of a complete MSE must accumulate all the sixteen inner product terms, so the actual throughput of the system is the throughput of a complete MSE. The proposed architecture generally generates a meaningful result every four clocks, which is also the fastest. Although higher cost of hardware is spent, it is not a serious problem due to the regularities of modules and modern VLSI technology.

VLSI Implementation

To realize the proposed architecture, we implement the folded-tree structured architecture by Genesil, which is a silicon compiler tool. In the chip, there are four multipliers, an adder-tree, and an accumulator computing the mean squared error. The comparator produces the minimum finally and reports a message to external controller whether a new minimum is generated in full-search algorithm. For tree-search algorithm, the comparison is executed with zero and the decision toward to left or right is made from the result. An external signal determines the mode of full-search or tree-search. Each multipler has two input ports. One input is the encoding data with eight bits resolution and the other is the codeword data. the codeword data is expressed by nine bits because it is the difference of two codewords in tree-search algorithm. Multipliers occupy most of the chip's area obviously. In order to simplify the design methodology, input data is expressed in non-negative integer form and sign-magnitude form for codeword data. Therefore, the design of 2's complement multiplier is avoided and the speed of the chip is increased.

Layout of the full chip is displayed in Figure 4. Total chip size is about $4.65 \times 5.21mm^2$ which is small due to the regularities of modules and advanced $0.8\mu m$ CMOS technology. The chip encloses only 22709 transistors and 100 pins; pin count is the dominant reason to enlarge the size of chip. Simulation of its function is passed and estimated clock rate is about 34MHz. In term of the speed for many applications or the utilization of hardware, it is both efficient and simple.

Conclusion

In this paper, a tree-structure based architecture for vector quantization algorithms has been proposed. Tree-folding technique and interleaving strategy are applied to reduce the hardware complexity and increase the throughput. Memory interleaving actually is only line delay which exists originally therefore there is no overhead . The proposed architecture has shorter latency and more flexibilities compared with bit-serial and MAC approaches. It also owns high throughput and smooth data flow. The proposed architecture gives short latency which is necessary to simplify the design of controller. A $4.65 \times 5.21mm^2$ sized chip is designed with about 34MHz clock rate. Obviously the chip has regular modules and appropriate bandwidth. Its short latency and high throughput satisfies the requirements of many compression applications. Since it is basically a mean squared error computation circuit, it can be used for various kinds of VQ, including full-search VQ, tree-search VQ, and other VQ algorithms.

References

[1] R. Jain, A. Madisetti, and R. L. Baker, "An integrated circuit design for pruned tree-search vector quantization encoding with an off-chip controller", *IEEE Tran. Circuits and Systems for Video Tech.* , vol. 2, No. 2, pp. 147-157, Jun. 1992.

[2] R. K. Kolagotla, S.-S. Yu, and J. F. JáJá, "VLSI implementation of a tree searched vector quantizer", *IEEE Tran. Signal Processing*, vol. 41, No. 2, pp. 901-905, Feb. 1993.

[3] G. A. Davidson, P. R. Cappello, and A. Gersho, "Systolic architecture for vector quantization", *IEEE Tran. ASSP*, vol. 36, No. 10, pp. 1651-1664, Oct. 1988.

[4] Y.-S. Jehng, L.-G. Chen, and T.-D. Chiueh, "An efficient and simple VLSI tree architecture for motion estimation algorithms", *IEEE Tran. Signal Processing*, vol. 41, No. 2, pp. 889-900, Feb. 1993.

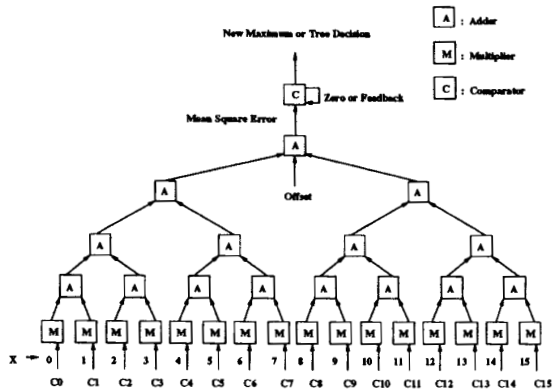

Figure 1: The proposed tree-structure architecture

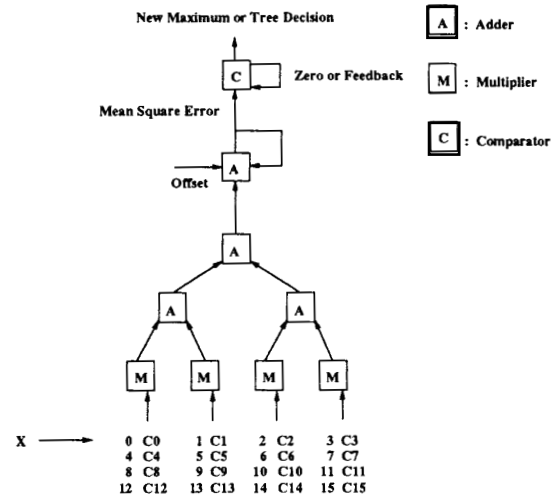

Figure 3: The folded-tree architecture and its data flow

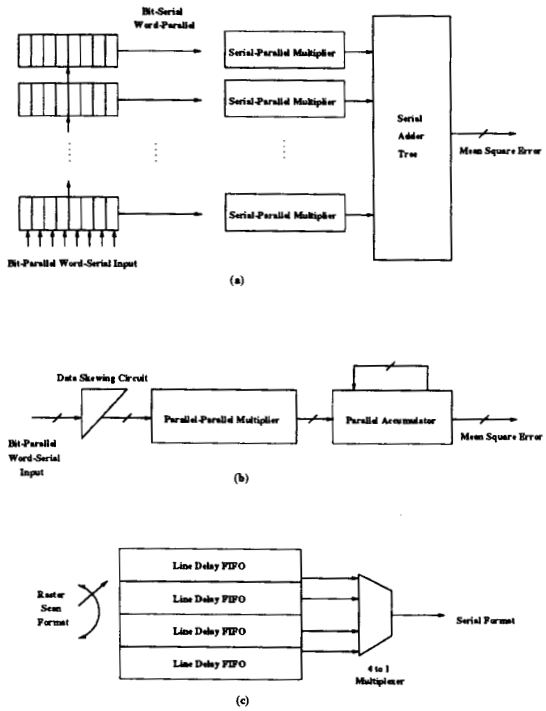

Figure 2: The data flow of bit-serial and MAC approaches

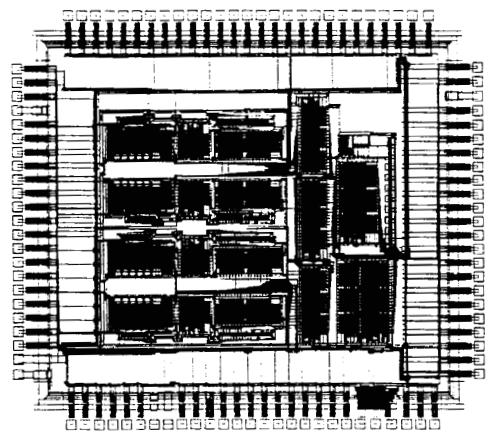

Figure 4: The layout of the VQ processor