A Low Power Variable Length Decoder for MPEG-2 Based on Nonuniform Fine-Grain Table Partitioning

Seong Hwan Cho, Thucydides Xanthopoulos, *Member, IEEE*, and Anantha P. Chandrakasan, *Member, IEEE*

***Abstract*—Variable length coding is a widely used technique in digital video compression systems. Previous work related to variable length decoders (VLD's) are primarily aimed at high throughput applications, but the increased demand for portable multimedia systems has made power a very important factor. In this paper, a data-driven variable length decoding architecture is presented, which exploits the signal statistics of variable length codes to reduce power. The approach uses fine-grain lookup table (LUT) partitioning to reduce switched capacitance based on codeword frequency. The complete VLD for MPEG-2 has been fabricated and consumes 530 μW at 1.35 V with a video rate of 48-M discrete cosine transform samples/s using a 0.6-μm CMOS technology. More than an order of magnitude power reduction is demonstrated without performance loss compared to a conventional parallel decoding scheme with a single LUT.**

***Index Terms*—Huffman, low power, MPEG-2, table partitioning, variable length decoder.**

I. Introduction

VARIABLE length coding (e.g., Huffman coding [1]) is a widely used technique in video compression systems. It is often applied together with other lossy image-compression techniques to further compress the data. The main idea of variable length coding is to minimize the average codeword length by exploiting the statistics of the data. Shorter codewords are assigned to frequently occurring data while longer codewords are assigned to infrequently occurring data. Therefore, minimum average codeword length and bit-rate reduction can be achieved.

In the MPEG-2 standard [2], the variable length code (VLC) stream is composed of many types of macroblock-level data, resulting in 15 different VLC tables and a total of about 450 codewords. The discrete cosine transform (DCT) run-level coefficients are most frequent, accounting for more than 80% of the variable length bit stream.

In this paper, we discuss a method for designing low power variable length decoders (VLD's) for portable video compression systems. The concepts are applied to a VLD which decodes all the VLC's in the MPEG-2 standard for MP@ML video.

Manuscript received September 16, 1997; revised January 21, 1998 and May 30, 1998. This work was supported by the Sharp Corporation. The work of T. Xanthopoulos was supported under a National Semiconductor Graduate Fellowship.

The authors are with the Massachusetts Institute of Technology, Cambridge, MA 021389 USA.

Publisher Item Identifier S 1063-8210(99)03343-0.

II. Background

The basic function of a VLD is *table lookup*. Since the VLC's are encoded using a one-to-one mapping without any arithmetic operations, the same applies to the decoder, i.e., no arithmetic computation is necessary to decompress the data. Although the basic functionality lies in the table-lookup procedure, the key operation of the VLD involves detecting the VLC in the bit stream since the codewords in the bit stream do not have any explicit word boundaries. The VLD does not know the start of the next VLC until the previous VLC is decoded. Hence, the length must be provided as a feedback from the decoded codeword to determine the start of the next VLC. This recursive operation must be efficiently designed to achieve a low power VLD.

A. Previous Work

Various approaches have been presented to implement high-throughput VLD's. They can be largely divided into two categories: a binary tree search method and a parallel method. The binary tree search technique can be implemented based on a Huffman tree, which uses the principle of a token propagation in a reverse binary tree constructed from the original codes [3]. However, for high-performance systems, i.e., MPEG-2 or HDTV, this approach is not suitable since it can decode only 1 bit per cycle.

As opposed to the tree search method, the parallel approach processes more than 1 bit per cycle. Lei and Sun [5] proposed a parallel decoding architecture which decodes one VLC per cycle regardless of its length. It uses a barrel shifter and an accumulator to align the VLC at a fixed position. The aligned VLC is then fed to the lookup table (LUT) as the address. The LUT is implemented as a programmable logic array (PLA) instead of read only memory (ROM) to avoid wasted area and power. It is possible to use ROM without much area and power waste to implement the LUT [4], but additional address generating circuits are needed. Further improvement of the parallel method have been proposed in [6], which produces output at a higher rate. (i.e., more than one codeword per cycle) by using augmented PLA-based LUT. Other methods for designing an efficient VLD were also presented, such as adaptive tree search technique [7], area-efficient memory-based method [8], and programmable VLD's [9], [10]. These VLD's were mainly aimed at high throughput and power dissipation was not a focus.

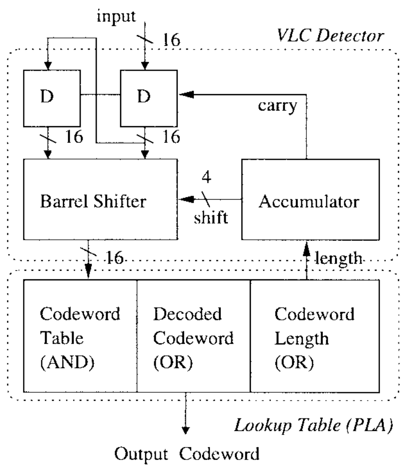


Fig. 1. Parallel decoding architecture proposed by Lei and Sun.

B. Low Power Approach

Various levels of design techniques can be employed to implement a low power VLD. In this paper, we deal mainly with architectural level power reduction. Once the architectural optimization for power is done, we step down to the structural level, where we choose the appropriate circuit style for further power reduction.

The most effective approach to lower the power consumption is reducing the supply voltage. However, as the supply voltage is reduced, the propagation delay of the circuit increases, limiting the amount of voltage scaling under a certain throughput constraint. From this voltage scaling point, the parallel method is preferred to the tree search method since the parallel approach processes multiple bits per cycle. In other words, we can run the parallel architecture at a slower clock frequency and lower voltage than the serial method at a given throughput.

At a high level, the parallel VLD can be decomposed into two components: the VLC detector and the LUT, as shown in Fig. 1. The VLC detector receives the input VLC's and generates an address for the LUT. To reduce additional circuit overhead, address generation is simply alignment of the VLC's at a fixed position so that the LUT uses the VLC itself as the address. The LUT receives the address from the VLC detector and produces the corresponding output codeword and length. The length is stored in the accumulator, which tells the barrel shifter how many bits are used. The barrel shifter then aligns the bits so that the next VLC is aligned. To achieve a low power operation, both components of the VLD must be optimized. A popular method of increasing the efficiency of the LUT is *prefix predecoding*, which is discussed in Section II-C.

C. Prefix Predecoding

In cases where the number of codewords in the table is large, there are some bits that are common to the long VLC's, which we call *prefix*. By exploiting these common prefixes, the size of the LUT can be reduced. Several approaches have exploited this prefix-predecoding method to efficiently decode the VLC's [11], [12], [14]. The basic idea of prefix predecoding is to group the VLC's by their common prefixes.

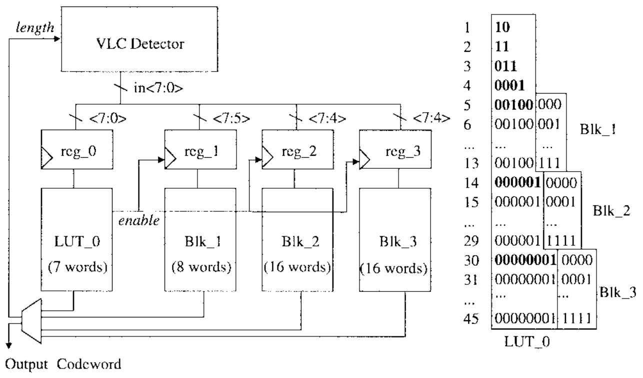


Fig. 2. Prefix predecoding architecture.

Fig. 2 shows an example of such a grouping. In this example, there are three prefixes: 00100, 000001, and 00 000 001. The highlighted bits in the figure indicate the VLC prefixes and short codes that are stored in LUT_0. The VLC's are decoded by the following steps: First, the VLC is fed to the first block LUT_0. If the VLC is a short-length code without any prefix, then the output codeword is produced from LUT_0 and the variable length detector is ready to decode the next VLC. On the other hand, if the input VLC is one of the long codes with a prefix, only the prefix is decoded in LUT_0. The remainder of the VLC is decoded in one of the subsequent blocks: Blk_1, Blk_2, or Blk_3. For example, if the input is a short VLC without a prefix "11," the decoding procedure is the same as the single LUT method. It is decoded in LUT_0 and the corresponding output is produced. If the input is a VLC with a prefix "00100001," LUT_0 decodes only the prefix ("00 100") and enables reg_1 to latch the remainder of the VLC. The remainder of the VLC ("001") is then decoded in the next block Blk_1.

With this prefix predecoding method, the size of the LUT is reduced because the prefixes are no longer redundant in the LUT. In this example, the number of prefixes stored in LUT_0 is reduced from 41 to 3 by using the predecoding technique. For MPEG-2 DCT ac coefficients, the majority of VLC's can be clustered by their prefixes and more than 50% area reduction can be achieved compared to a single table. In addition, we achieve a power reduction up to a factor of two since the switched capacitance is also reduced.

III. Nonuniform Fine-Grain Table Partitioning

A. Low Power Architecture

The VLC tables are often the most area and power intensive blocks of a VLD. In an MPEG-2 VLD system, about 80% of the area is consumed by the LUT. To lower the average power dissipation of the VLD, we will reduce the energy consumption per each codeword at a given throughput. The average energy consumption per codeword in the LUT can be modeled by the following equation:

$$E_{\text{LUT, avg}} = \sum_{i=1}^{n} P_{cwi} E_{cwi}. \tag{1}$$

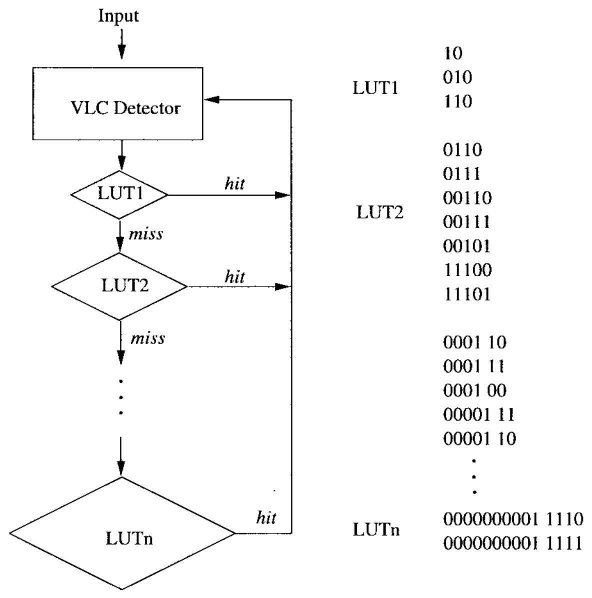


Fig. 3. Low power algorithm.

where P_{cwi} is the probability that codeword i will occur, E_{cwi} is the energy required to decode the codeword i, and n is the total number of codewords in the table.

In conventional approaches, the energy required to decode a VLC in a *LUT* does not vary much over the codeword probability. (i.e., $E_{cwi} \approx \mathrm{Constant},\ \forall i$). This is because the VLC table is implemented in a single LUT and the whole table has to be charged and discharged every cycle. The single LUT method does not exploit the fact that the short codewords are frequently occurring. Therefore, the average energy in (1) is dominated by codewords which have high probability of occurrence.

We propose a low power approach that exploits the variable length codeword statistics, making the energy to decode a codeword depend on the codeword probability. Fig. 3 shows the basic algorithm for low power variable length decoding. The main idea is to partition the single LUT into several variable size tables with respect to their energy consumption and frequency of occurrence. By having variable size tables, the energy required to decode a codeword will vary depending on the size of the partitioned table. Low power can be achieved if the dominant term in (1) is made small (i.e., E_{cwi} with high P_{cwi}). In our proposed approach, frequently occurring codewords use small tables with less switched capacitance and infrequent codewords use larger tables.

The flowchart of Fig. 3 works as follows. Given the codeword LUT decomposed into several tables, the VLC from the VLC detector goes through a series of tables looking for a match. If there is a match, the output word is produced and the next VLC is processed, going through the same procedure. In case of a miss, the matching process continues until the codeword is fully decoded. With this method, we no longer achieve a constant throughput. An additional buffer must be added in order to get a constant throughput rate. However, this variable output rate does not matter for MPEG-2 DCT coefficients since the decoded DCT coefficients are run length coded, which have to be buffered anyway. To achieve the maximum hit ratio, tables are ordered in order of descending probability. With this data-driven decoding process, (1) is modified as follows:

$$\begin{aligned}E_{\mathrm{LUT,avg}} = {} & Pr_1 E_{H1} + Pr_2(E_{H2} + E_{M1}) \\ & + Pr_3(E_{H3} + E_{M1} + E_{M2}) + \cdots \\ & + Pr_n\left(E_{Hn} + \sum_{i=1}^{n-1} E_{Mi}\right) + E_{\mathrm{overhead}}(n) \qquad (2)\end{aligned}$$

where Pr_i is the probability that table i is a hit (i.e., a match was found in table i), n is the total number of decomposed tables, E_{Hi} is the energy consumption of table i when there is a hit, and E_{Mi} is the energy when there is a miss. $E_{\mathrm{overhead}}(n)$ is the energy consumed by the circuit overhead introduced by the table partitioning, such as conditional branches and clock generation which increases with n. The average energy consumption is no longer a constant value, but the sum of the energy consumption in the decomposed tables weighed with the corresponding probability. To achieve minimum average energy, the first table must consume the least energy while having a high hit ratio. The hit ratio can be improved by allocating more codewords, but this will also increase the energy consumption. Hence, a solution which results in the lowest energy at a given throughput must be found.

B. Codewords Allocation per Each Table

There are two issues that need to be considered in the table partitioning to minimize the energy consumption. First, how many partitions should be used in the LUT and, second, how many codewords should be allocated per each decomposed table [13]. To illustrate the idea of the low power table partitioning, let us first consider an example of splitting the table into two tables. In the case of this two-way partitioning, the question is how many codewords should be assigned to the first table to achieve minimum energy. With the proposed algorithm, (2) reduces down to (3) as follows:

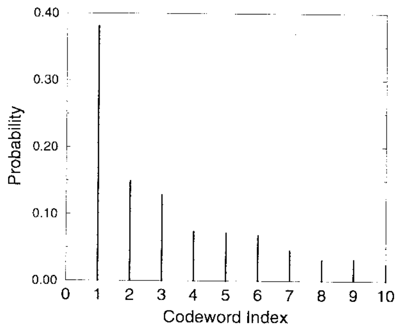
$$E_{\mathrm{LUT,avg}} = Pr_1 E_{H1} + (1 - Pr_1)(E_{H2} + E_{M1}). \qquad (3)$$

We will neglect the $E_{\mathrm{overhead}}(n)$ in this two-way partition scheme.

Consider a simple example with ten codewords, whose probability is shown in Fig. 4. Assuming that the first n codewords are in the first table, the energy required to decode a codeword in the first and second table is shown in Fig. 5(a), obtained from PLA simulation results using HSPICE. The average energy consumption is shown in Fig. 5(b), which is a plot of (3). Note that the energy consumption is calculated for two cases: $E_{\mathrm{miss}} = 0$ and $E_{\mathrm{miss}} = E_{\mathrm{hit}}$. In general, E_{hit} and E_{miss} depend on several factors, such as how the table is implemented and what the incoming bit stream is. Without loss of generality, it is reasonable to assume that E_{miss} will be the same as or less than E_{hit} since the switched capacitance will typically be larger in case of a hit than a miss. Hence, we consider (3) for two extreme cases: $E_{\mathrm{miss}} = 0$ and $E_{\mathrm{miss}} = E_{\mathrm{hit}}$. Each will represent a lower and an upper bound for (3). The actual energy plot will lie somewhere between

	1	10	
LUT1	2	010	
	3	110	?
	4	0110	
	5	0111	
	6	00110	
LUT2	7	00111	
	8	00101	
	9	11100	
	10	11101	

(a)



(b)

Fig. 4. Example of a VLC table. (a) VLC table. (b) Probability.

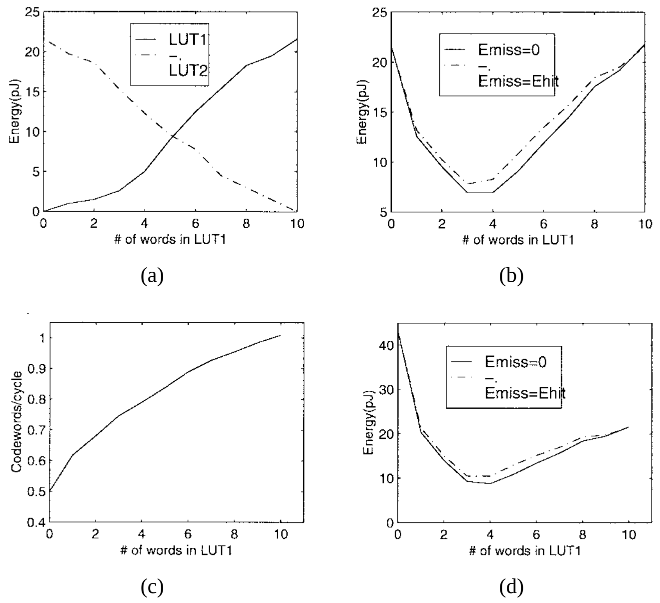


Fig. 5. Simulation results of two way partitioning. (a) Energy consumption in LUT_1, LUT_2. (b) Average energy consumption. (c) Throughput. (d) Energy at constant throughput.

the two curves. As can be seen in the graph, the average energy consumption is not sensitive to E_{miss}. For an optimum partitioned table, this is an obvious result since the table size and its energy will be increasing as the codeword goes through the series of tables and, hence, E_{miss} of the previous table will be negligible compared to E_{hit} or E_{miss} of the next table. In this example, the number of codewords that results lowest power consumption is around 3–4.

Another important issue that needs to be considered in the table partitioning is the *throughput*. Each conditional branch that the VLC goes through in the flow chart of Fig. 3 is a critical path. With this scheme, it would take n cycles to decode a codeword in the nth table. The average throughput can be represented by the following equation:

$$f_{\text{out}} = (Pr_1 + 2Pr_2 + 3Pr_3 + \cdots + n\,Pr_n)^{-1} f_a \quad (4)$$

where f_{out} is the average output codeword throughput (codewords/s), Pr_i is the probability that table i is a hit, and f_a is the clock frequency of the VLD. Fig. 5(c) shows the plot of (4), with a fixed f_a. As can be seen, the throughput decreases as the number of codewords in the first table decrease. Therefore, we have to consider the throughput together with the energy plot. Fig. 5(d) shows energy consumption at constant throughput. This graph was obtained by varying f_a and the supply voltage to get minimum energy at constant throughput for each number of codewords in LUT_1. The optimum number of codewords we get from this new graph is $n = 3$–4.

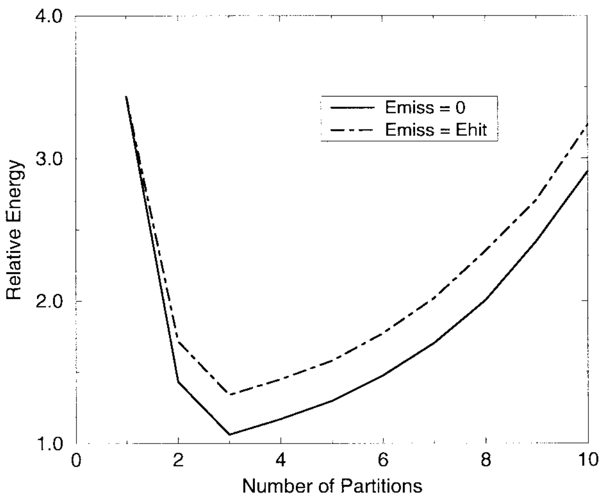


Fig. 6. Simulation results of M-way partitioning.

C. Number of Partitions

Now, let us consider the case of M-way partitioning. An optimum number of partitions M must be found to achieve lowest energy at a given throughput. Based on the simulation, which takes into account all the possible combinations of M-way partition, the results are plotted in Fig. 6. The graph shows the minimum energy for each partition with the overhead energy considered. The overhead circuitry is composed of latches at the input of each LUT, Hit, and $Miss$ detection circuit, which is a simple combinational logic and a delayed clock circuit which is implemented using a D-latch. The energy consumption of the overhead circuitry was measured using HSPICE and it was taken into account when calculating the total energy consumption. Although the size of the LUT (and, hence, the energy consumption) varies in a wide range, the energy consumption of the overhead circuitry does not vary much. Simulation results show that overhead circuitry consumes less than 20% of a LUT energy. The average energy consumption increases when the number of partitioning exceeds four, when the E_{overhead} in (2) starts to become dominant. The optimum number of partitions for this ten-codeword example is $n = 2$–3.

IV. VLC Detector Based Optimization

Another part of the VLD which consumes power is the VLC detector. Unlike the decomposed LUT, which is accessed by their probability, the VLC detector is operating on every cycle. From the result of prefix predecoding and fine-grain nonuniform table partitioning, the LUT is optimized to consume minimum energy. The energy has been lowered to an extent that now the VLC detector consumes about half of the total

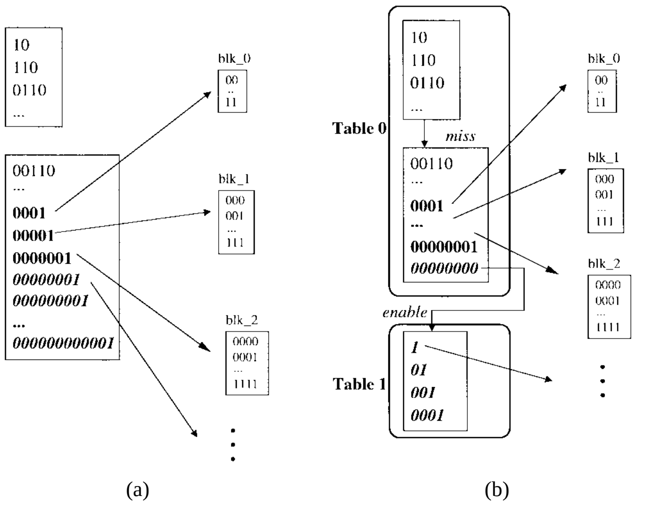


Fig. 7. Barrel-shifter-based table partitioning. (a) LUT without VLC detector reduction. (b) LUT with VLC detector reduction.

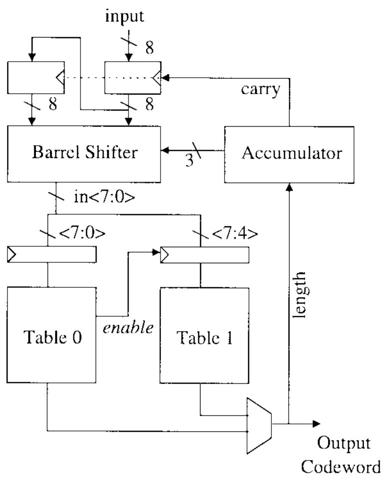


Fig. 8. Reduced VLC detector architecture.

energy to decode a codeword. In conventional approaches, the VLC detector was designed for the worst case input (i.e., the longest VLC) to achieve high throughput. Therefore, there was a lot of waste in the VLC detector energy consumption since only short VLC's occur most of the time. To minimize the average energy consumption in the VLC detector, we propose a method which exploits the signal statistics of the input VLC's. By reducing the size of the VLC detector and breaking up the variable length codeword into two or more parts, we can reduce the average energy consumption [15].

The following figures illustrate an example of the reduced VLC detector architecture. Fig. 7 shows the LUT and Fig. 8 shows the corresponding architecture. In Fig. 7(a), the table is reduced by prefix predecoding and fine-grain table-partitioning method. The highlighted bits indicate the prefixes of the VLC's. In this example, the maximum length of the VLC is 16 and, in conventional methods, the output bit width of the VLC detector must be at least 16. However, by decomposing the long length codes (denoted as italic bold bits), as shown in Fig. 7(b), we can reduce the size of the VLC detector. An additional table LUT_2 is added, but the size of the VLC detector can be reduced to eight and the size of LUT_1 is further reduced. If the VLC is a short codeword whose length is less than eight, then it is decoded in Table I in the same manner as before. If a VLC of length greater than eight comes in, e.g., a 14-bit VLC ("00 000 000 010 011"), then only the first eight bits ("00 000 000") are decoded in Table 0. (Note that this VLC has the prefix "0 000 000 001.") Since Table 0 does not have all the information to decode the VLC by receiving the first eight bits, it instructs the barrel shifter to shift eight bits and latches the rest of the bits ("010 011") to Table I. Table I then decodes the necessary bits for prefix decoding ("01") and enables one of the blocks to decode the remainder of the bits ("0011").

By reducing the size of the VLC detector, we have reduced the energy consumption for short codewords. However, we also decreased the throughput by one cycle for long codewords. Again, there is a tradeoff between energy and throughput. A small VLC detector is desirable for low power, but the opposite is preferred for high performance. An optimum solution, which results in a lowest power, can be achieved from the following analysis. The following notations are used:

$E_{\mathrm{VD,\,avg}}$	average energy consumption of the VLC detector;
$L_{\max}$	maximum length of the input VLC;
Y_b	number of output bits from the barrel shifter (VLC detector size);
$E(Y_b)$	energy consumption of the VLC detector when its size is Y_b;
f_a	operating frequency of the VLC detector;
f_{out}	throughput of the VLD (codewords/s);
$P_{cr}(x, y)$	sum of the occurring probabilities of VLC's, which have length greater than x and less than y;
Pr_i	probability of a block (decomposed as described in the previous section) being a hit;
n	$[L_{\max}/Y_b+1]$, where $[x]$ represents the integer part of x.

There are two cases of concern: $L_{\max} \leq Y_b$ and $L_{\max} \geq Y_b$.

Case 1: $L_{\max} \leq Y_b$

$$f_{\mathrm{out}} = (Pr_1 + 2\,Pr_2 + 3\,Pr_3 + \cdots)^{-1} \tag{5}$$

$$E_{\mathrm{VD,\,avg}} = E(Y_b). \tag{6}$$

This is when the size of the barrel shifter is larger than the maximum length of the VLC. It basically has the same structure as proposed in the previous section. The energy consumption of the VLC detector will be independent of the codeword probability.

Case 2: $L_{\max} \geq Y_b$*:*

$$\begin{aligned} f_{\mathrm{out}} = \{&P_{cr}(0, Y_b) + 2P_{cr}(Y_b, 2Y_b) + \cdots \\ &+ nP_{cr}((n-1)Y_b, nY_b)\}^{-1} f_0 \end{aligned} \tag{7}$$

$$\begin{aligned} \text{Energy} = [&P_{cr}(0, Y_b) + 2P_{cr}(Y_b, 2Y_b) + \cdots \\ &+ nP_{cr}((n-1)Y_b, nY_b)]E(Y_b). \end{aligned} \tag{8}$$

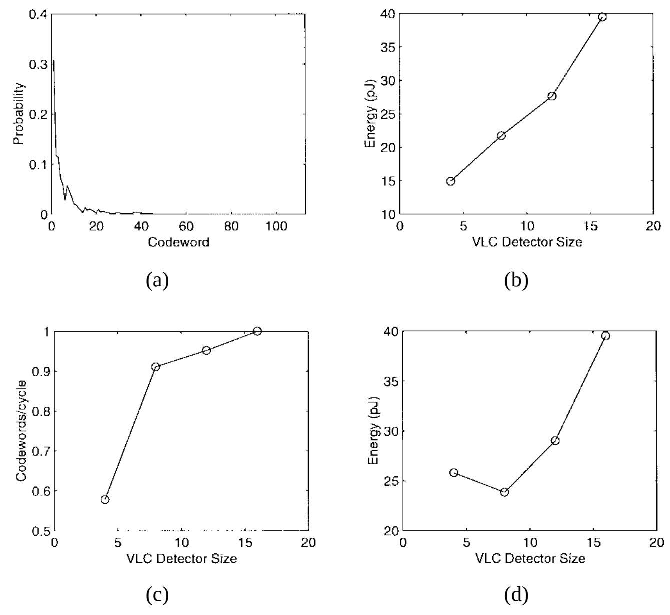


Fig. 9. Finding the optimum size of the barrel shifter. (a) Probability. (b) Energy consumption. (c) Throughput. (d) Energy at constant throughput.

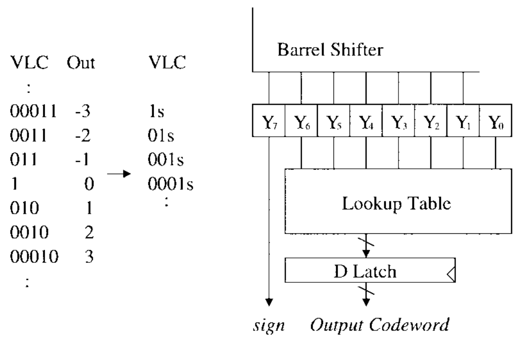


Fig. 10. VLC table with sign extension.

In this case, the decoding procedure cannot be done in one cycle for VLC's that have length greater than Y_b. The VLC must be decomposed and decoded in several cycles in order to fit the barrel shifter. Equations (7) and (8) represent the average throughput and energy consumption of the VLC detector. The sum of the probabilities indicates the average cycle it takes to decode a codeword.

The graphs in Fig. 9 show the result of the previous equations for the cases when $Y_b = 4$, 8, 12, and 16. The VLC detector is optimized for DCT VLC's since they are most frequent and have the longest average length VLC. Fig. 9(a) shows the codeword probability of the MPEG-2 DCT VLC's. The second graph shows the energy consumption of the VLC detector as we increase its size (8). Throughput is shown in the next graph, where (7) is plotted. The reason why we see a significant drop in throughput as Y_b is changed from 8 to 4 is because although the maximum length of the VLC is 16, the average length is about five, which means that $Y_b = 4$ will result an average of two cycles to decode a VLC. Energy is shown in the next graph and we see it decreases as the size of the VLC detector is reduced. To achieve the best result, we look at the energy consumption at constant throughput in Fig. 9(d). The minimum energy we get from the graph is when $Y_b = 8$. About 40% energy reduction is achieved at same performance compared to the architecture without VLC detector size reduction.

V. Implementation of VLD for MPEG-2 Tables

A. Sign-Extended VLC Table Reduction

In MPEG-2 VLC tables, there are tables which have sign extension. The sign-extended VLC's have the following format: $a_n a_{n-1}, \cdots, a_1 a_0 s$, where each a_i represents a bit in the VLC, and s denotes the last bit of the VLC, which is the sign of the output codeword. Fig. 10 shows an example of a VLC with the sign extension.

An easy way to decode this VLC table would be to implement the LUT for both cases of $s = 1$ and 0. However, this would double the area and power of the LUT. Another approach is to think of the whole VLC (without the last sign bit s) as a prefix and go through the prefix-based table-partitioning scheme. In this case, the area and power increases by only a small amount. On the other hand, the throughput decreases since we always need an extra cycle for the sign extension bit. An efficient way to decode the VLC with sign extension would be pipelining. The idea is to decode the sign bit one cycle later, together with the next VLC. Since the sign is the least significant bit of the VLC, it will always be placed one bit prior to the start of the next VLC. Hence, instead of reading the VLC input from the most significant bit of the barrel shifter output, the LUT will receive the bits from the second most significant bit. The most significant bit of the barrel shifter is used by the sign bit. Fig. 10 shows the example of VLC with sign extension and how it is decoded. Using this scheme, there is only one clock cycle increase in latency and the throughput stays the same.

B. Low Power VLD Architecture

The low power VLD possesses the architecture shown in Fig. 11. The table partitioning and VLD detector optimization has been applied together with the prefix-based decoding. The first three blocks—LUT_0, LUT_1, and LUT_2—decode the short codewords and prefixes. Blocks blk_0, blk_1, etc. store the remaining bits for prefix predecoding. When a VLC comes in, reg_0 is latched and it is fed to LUT_0. If it is a hit, then the corresponding output codeword is produced. If it is a miss, it goes on to the next table. Most of the time, LUT_0 will be accessed and energy will be saved. The energy consumption is further reduced by choosing static complimentary metal–oxide–semiconductor (CMOS) in implementing the LUT rather than a PLA. Although the area is larger, static CMOS has the advantage that its performance and power consumption are better at a low-voltage supply.

VI. Results

Table I shows the test results of the low power VLD chip. The chip is implemented using 0.6-μm CMOS process. The threshold voltages are 0.67 and 0.93 V for nMOS and pMOS,

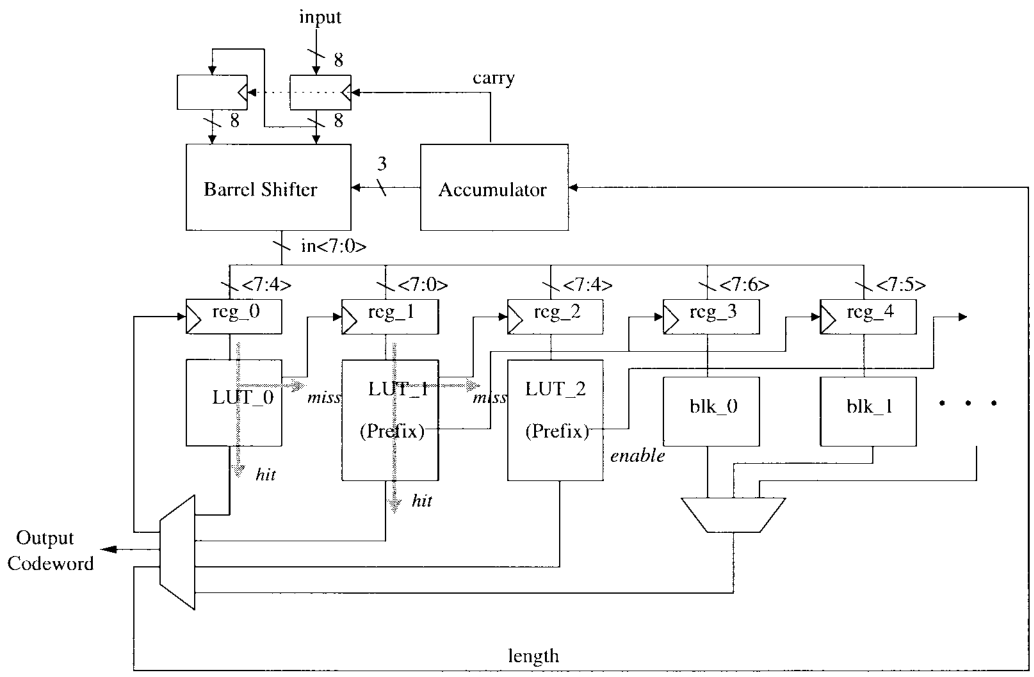


Fig. 11. Low power VLD architecture.

TABLE I
CHIP TEST RESULTS

Power	$0.531mW$ @ 1.35V
Area	$5.24mm^2$(core), $16.54mm^2$(full)
Frequency	$12.5Mhz$
Throughput	$7.57Mcodewords(VLCs)/sec$
Video rate	$48.75MDCTsamples/sec$

respectively. The area is determined by the 40 input and output pads, as can be seen in the chip photo shown in Fig. 14. The boxes with "B- -" indicate the VLC table, as denoted in ISO/IEC 13 818-2 [2]. The largest tables are B-14 and B-15, which are the DCT ac coefficients.

The probability and energy consumption of the block and codeword are shown in Figs. 12 and 13, based on the MPEG-2 DCT ac table. Each codeword is numbered and represented in the horizontal axis. In Fig. 12, two graphs are plotted. The solid line is the probability of individual codewords and the dotted line is the probability of the decomposed table.

The energy is plotted for three cases. First, the solid line represents the energy consumption of each table. As one can see, the energy consumption increases as the probability of a hit decreases. The energy consumption of each table was obtained by feeding in codewords of that particular block only. The energy correlation that may occur when the input bit stream is composed of different tables is neglected. There are three big jumps in the energy plot at $n = 4$, 13, and 27, each indicating a miss. The reason why the table energy does not increase monotonically is because of the different energy consumption of prefix predecoding blocks (i.e., blk_0, blk_1, etc.). Although the energy consumption of the first three LUT's (i.e., LUT_0, LUT_1, LUT_2) are in the increasing

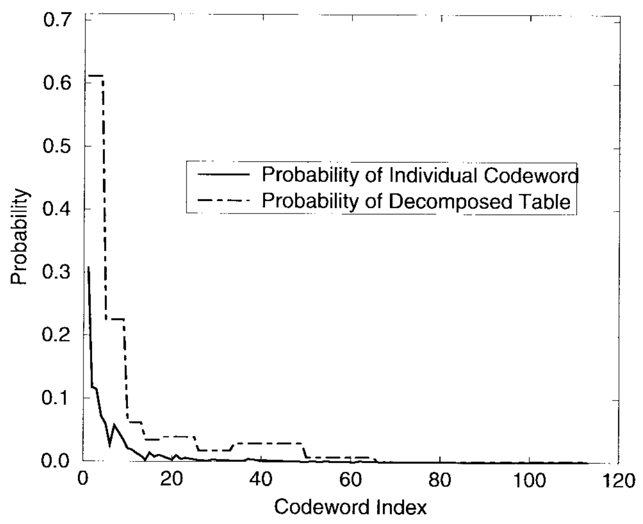


Fig. 12. Codeword probability.

order, the prefix predecoding blocks are not ordered by their energy consumption since the size of the prefix predecoding blocks are determined by the prefix and we cannot place them arbitrarily by their energy consumption. The dashed line represents the simulated result of the average energy. It is achieved by multiplying the table probability of Fig. 12 by the energy consumption in Fig. 13. The dotted line shows the energy consumption when the actual MPEG-2 DCT sequence are fed to the VLD. The actual result is slightly higher than the one we expected. This is due to the probability variation of the codewords and error that was included in obtaining the energy consumption of each table.

These results were obtained by running various types of MPEG-2 video data. Since our VLD decodes only the VLC's

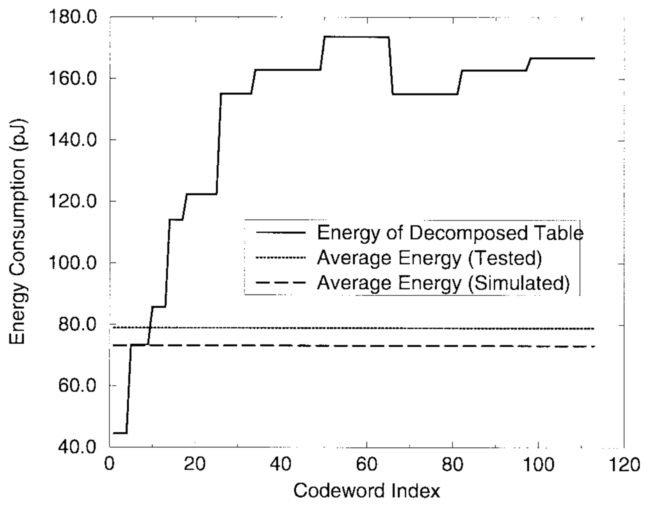


Fig. 13. Energy consumption.

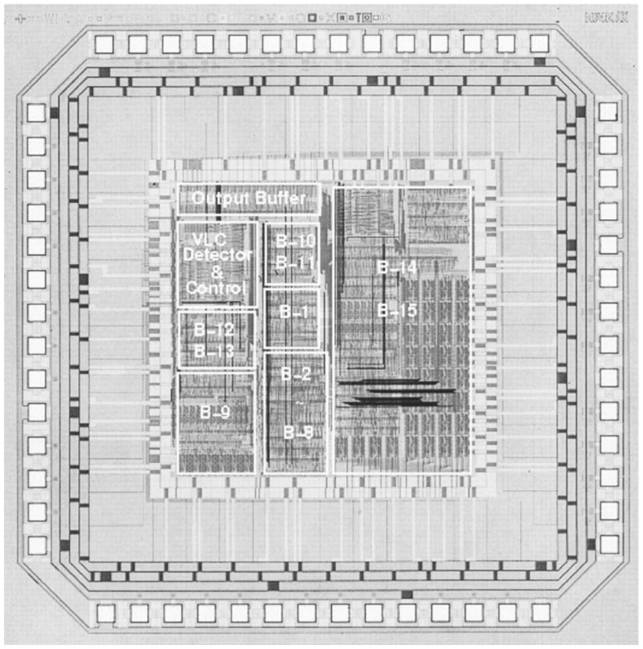


Fig. 14. Chip photo.

and does not perform bit-stream parsing or fixed-length code decoding, the VLC part of the bit stream was sorted to a buffer and then fed to the VLD. Knowing the required DCT sample rate for MP@ML video decompression, we decoded the VLC's at a rate which was enough for maximum MP@ML video bit rate given in [2].

VII. Conclusion

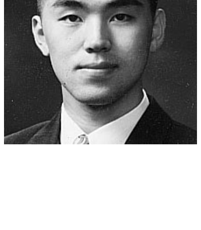
A low power VLD has been presented. Power reduction was achieved mainly by architecture level optimizations, where we decomposed the table into several variable size blocks by exploiting the statistics of the VLC. First, the table was reduced by clustering the codewords by their prefixes. It was then decomposed into variable size tables, which allowed the energy to vary according to the occurring frequency of the VLC. Next, the VLC detector was optimized to further reduce the energy consumption. A chip was implemented which decodes all 15 different MPEG-2 VLC tables. The actual power consumption for the DCT sequence was about 0.530 mW at 1.35 V. More than an order of magnitude power reduction was achieved compared to the result shown in [4], excluding the factor achieved by voltage scaling.

Acknowledgment

The authors thank Y. Yaoi for his valuable contributions.

References

[1] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE,* vol. 40, pp. 1098–1101, Sept. 1952.
[2] *Generic Coding of Moving Picture and Associated Audio*, Draft International Standard, Recommendation H.262, ISO/IEC 13818.
[3] A. Mukherjee, N. Ranganathan, and M. Bassiouni, "Efficient VLSI designs for data transformations of tree-based codes," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 306–314, Mar. 1991.
[4] E. Komoto and M. Seguchi, "A 110 MHz MPEG-2 variable length decoder LSI," in *Proc. VLSI Circuits Symp.*, 1994, pp. 71–72.
[5] S. M. Lei and M. T. Sun, "An entropy coding system for digital HDTV applications," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 1, pp. 147–155, Mar. 1991.
[6] Y. J. Jan, "A high speed variable length decoder for digital HDTV systems," *Signal Processing HDTV IV,* pp. 333–340, 1992.
[7] Y. Ooi, A. Taniguchi, and S. Demura, "A 162 Mbits/s variable length decoding circuit using an adaptive tree search technique," in *IEEE CICC,* 1994, pp. 107–110.
[8] H. Park and V. K. Prasanna, "Area efficient VLSI architectures for Huffman coding," *IEEE Trans. Circuits Syst. II,* vol. 40, pp. 568–575, Sept. 1993.
[9] B. Wei and T. Meng "Programmable parallel Huffman decoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 175–178, Apr. 1995.
[10] D. S. Ma, J. F. Yang, and J. Y. Lee, "Programmable and parallel variable length decoder for video systems," *IEEE Trans. Consumer Electron.*, vol. 39, pp. 448–453, Aug. 1993.
[11] S. B. Choi and M. H. Lee, "High speed pattern matching for a fast Huffman decoder," *IEEE Trans. Consumer Electron.,* vol. 41, pp. 97–103, Feb. 1995.
[12] S. F. Chang and D. G. Messerschmitt, "Designing high-throughput VLC decoder Part I—Concurrent VLSI architectures," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 2, pp. 187–196, June 1992.
[13] S. H. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "Design of low power variable length decoder using fine grain nonuniform table partitioning," in *IEEE Int. Symp. Circuits Syst.*, June 1997, pp. 2156–2159.
[14] R. Hashemian, "Design and hardware construction of a high speed and memory efficient Huffman decoding," in *IEEE Int. Conf. Consumer Electron.,* 1994, pp. 74–75.
[15] S. H. Cho, T. Xanthopoulos, and A. P. Chandrakasan, "An ultra low power variable length decoder for MPEG-2 exploiting codeword distribution," in *IEEE Custom Integrated Circuits Conf.,* May 1998, pp. 177–180.



Seong Hwan Cho received the B.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1995, the M.S. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, in 1997, and is currently working toward the Ph.D. degree at MIT.

His research interests include low power very large scale integration (VLSI), radio frequency (RF) communication circuits, and systems for wireless sensors.

Thucydides Xanthopoulos (M'91) received the S.B. and S.M. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1992 and 1995, respectively, and is currently working toward the Ph.D. degree on data-driven signal processing systems for video applications at MIT.

His research interests include low power VLSI design, embedded digital signal processing (DSP) architectures and computer-aided design (CAD) for low power.

Mr. Xanthopoulos is a member of the Association for Computing Machinery (ACM). He was awarded a National Semiconductor Graduate Fellowship in 1996.

Anantha P. Chandrakasan (S'87–M'85) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been with the Massachusetts Institute of Technology, Cambridge, where he is currently an Associate Professor of electrical engineering and computer science.

He has co-authored *Low Power Digital CMOS Design* (Norwell, MA: Kluwer, 1995) and co-edited *Low Power CMOS Design* (Piscataway, NJ: IEEE Press, 1998). His research interests include the ultralow power implementation of custom and programmable DSP's, distributed wireless sensors, multimedia devices, emerging technologies, and CAD tools for VLSI.

Dr. Chandrakasan served on the Technical Program Committee of various conferences, including ISSCC, VLSI Circuits Symposium, DAC, ISLPED, and ICCD. He served as a technical program co-chair for the 1997 International Symposium on Low-Power Electronics and Design (ISLPED), VLSI Design'98, and the 1998 IEEE Workshop on Signal Processing Systems, and as a general co-chair of the 1998 ISLPED and the 1998 IEEE Computer Society Annual Workshop on VLSI. He is currently the signal processing subcommittee chair for ISSCC'99. He is an associate editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He is a member of the Design and Implementation of Signal Processing Systems (DISPS) Technical Committee (TC) of the Signal Processing Society. He held the Analog Devices Career Development Chair from 1994 to 1997. He received the 1995 NSF Career Development Award, the 1995 IBM Faculty Development Award, the 1996 and 1997 National Semiconductor Faculty Development Awards, and the IEEE Communications Society 1993 Best Tutorial Paper Award.