JAGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard

MARIO KOVAC AND N. RANGANATHAN, SENIOR MEMBER, IEEE

In this paper, we describe a fully pipelined single chip VLSI architecture for implementing the JPEG baseline image compression standard. The architecture exploits the principles of pipelining and parallelism to the maximum extent in order to obtain high speed and throughput. The architecture for discrete cosine transform and the entropy encoder are based on efficient algorithms designed for high speed VLSI implementation. The entire architecture can be implemented on a single VLSI chip to yield a clock rate of about 100 MHz which would allow an input rate of 30 frames per second for 1024 × 1024 color images.

***Keywords*—data compression, DCT, JPEG, parallel processing, VLSI**

I. INTRODUCTION

Data compression is the reduction or elimination of redundancy in data representation in order to achieve savings in storage and communication costs. Data compression techniques can be broadly classified into two categories: lossless and lossy schemes. In lossless methods, the exact original data can be recovered while in lossy schemes a close approximation of the original data can be obtained. The lossless methods are also called entropy coding schemes since there is no loss of information content during the process of compression. Lossless methods are used for text compression and image compression in certain environments such as medical imaging where no loss of information is tolerated and typically the compression ratio is around 3:1. Lossy compression methods are commonly applied in image and audio compression and depending upon the fidelity required compression ratios of even up to 100:1 can be obtained. Digital images require an enormous amount of space for storage. For example, a color image with a resolution of 1024 × 1024 picture elements (pixels) with 24 b per pixel would require 3.15 M bytes in uncompressed form. At a video rate of 30 frames per second, this requires a data rate of 94 M bytes per second. With the recent advances in video applications such as video teleconferencing, HDTV, home entertainment systems, interactive visualization and multimedia, there is an increasing demand for even higher bandwidth computing and communication systems. Very high speed implementation of efficient image compression techniques will significantly help in meeting that challenge.

In recent years, a working group known as Joint Photographic Expert Group (JPEG) consisting of three international standard organizations, International Telegraph and Telephone Consultative Committee (CCITT), International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC), has defined an international standard for coding and compression of continuous-tone still images. This standard is commonly referred to as the JPEG standard. The primary aim of the JPEG standard is to propose an image compression algorithm that would be application independent and aid VLSI implementation of data compression [9].

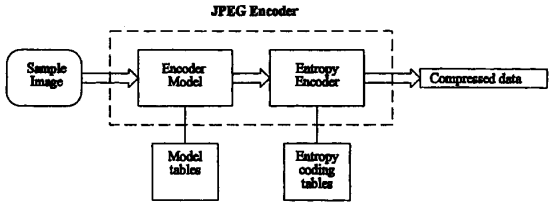


Fig. 1. JPEG Encoder.

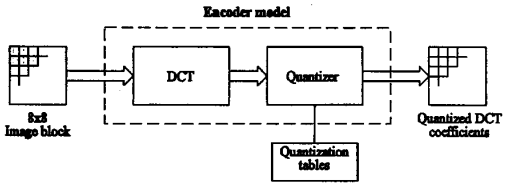


Fig. 2. JPEG baseline encoder model.

Manuscript received February 22, 1994; revised July 10, 1994.. M. Kovac's work was supported by a Fullbright scholarship and N. Ranganathan's work is supported in part by Enterprise Florida Innovation Partnership FTRI Fund (formerly Florida High Technology and Industry Council).

M. Kovac is with the Faculty of Electrical Engineering, University of Zagreb, 41000 Zagreb, Croatia.

N. Ranganathan is with the Center for Microelectronics Research, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620 USA.

IEEE Log Number 9407659.

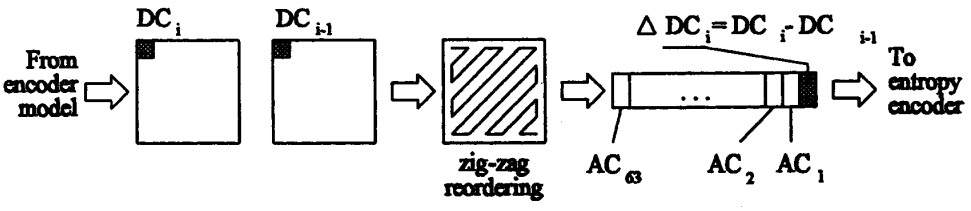


Fig. 3. Reordering of DCT output.

In this paper, we propose an efficient single chip VLSI architecture for implementing the JPEG baseline compression standard algorithm. The architecture fully exploits the principles of pipelining and parallelism to achieve high speed and throughput. The JPEG baseline algorithm consists mainly of two parts: 1) Discrete Cosine Transform (DCT) computation and 2) Entropy encoding. The hardware architecture for DCT is based on a modification of the algorithm proposed in [21] for reduction in computation. The entropy encoding part consists of runlength encoding followed by Huffman encoding. The architecture for entropy encoding is based on a hardware algorithm designed to yield maximum throughput and clock speed.

The paper is organized as follows. An outline of the JPEG compression standard is given in Section II. Section III describes briefly related work on the implementation of the JPEG standard. The proposed system architecture is discussed in Section IV. The design of specific algorithms and their mapping onto efficient hardware architectures for each component of the JPEG standard are described in Sections V and VI. Conclusions are provided in Section VII.

II. Outline of the JPEG Compression Standard

The basic model for the JPEG encoder is shown in Fig. 1. The encoder model transforms the input image into an abstract representation more suitable for further processing. The encoder model may require parameters stored in some model tables for achieving this transformation. The entropy encoder is a compression procedure which converts the output of the encoder model into a compressed form. Also, the entropy encoder may use tables for storing the entropy codes. Four distinct coding processes were derived based on the above described JPEG model: 1) baseline process, 2) extended DCT-based process, 3) lossless process, and 4) hierarchical process.

The baseline and the extended processes are also known as DCT-based processes since they use DCT within the encoder model. The lossless process uses prediction based methods within the encoder model. The hierarchical process uses the encoder model from the extended process or the lossless process. The baseline process uses Huffman codes for entropy encoding while the other three processes use either Huffman or arithmetic. Since the focus of this paper is on VLSI implementation of the baseline process, we describe the baseline process in detail in the rest of this section. For a complete overview of the JPEG standard and the various processes, the reader is referred to [7]–[10].

The encoder model for the baseline process is shown in Fig. 2. The input image is divided into nonoverlapping blocks of 8 × 8 pixels and input to the baseline encoder. The pixel values are converted from unsigned integer format to signed integer format and DCT computation is performed on each block. DCT transforms the pixel data into a block of spatial frequencies that are called the DCT coefficients. Since the pixels in the 8 × 8 neighborhood typically have small variations in gray levels, the output of DCT will result in most of the block energy being stored in the lower spatial frequencies. On the other hand, the higher frequencies will have values equal to or close to zero and hence, can be ignored during encoding without significantly affecting the image quality. The selection of frequencies based on which frequencies are more important and which ones are less important can affect the quality of the final image. JPEG allows for this by letting the user predefine the quantization tables used in the quantization step that follows the DCT computation. The selection of quantization values is critical since it affects both the compression efficiency and the reconstructed image quality.

The block of DCT coefficients output by the encoder model is rearranged into one dimensional data using zigzag reordering as shown in Fig. 3. The location (0, 0) of each block i contains the DC coefficient for the block represented as DC_i. This DC coefficient is replaced by the value ΔDC_i which is the difference between the DC coefficients of block i and block $i-1$. Since the pixels of adjacent blocks are likely to have similar average energy levels only the difference between the current and previous DC coefficients is used, which is commonly known as differential pulse code modulation (DPCM) technique. It should be noted that the high frequency coefficients that are more likely to be zeroes get grouped at the end of the one dimensional data due to the zigzag reordering.

The entropy encoder details are shown in Fig. 4. The entropy encoder uses variable length encoding based on a statistical model in order to encode the rearranged DCT coefficients. In the entropy encoder the quantized DCT coefficients are converted into a stream of [runlength count, category] pairs. For each pair, there is a corresponding variable length Huffman code which will be used by the Huffman encoder to perform the compression. The Huffman codes are stored in a table. A detailed description of the

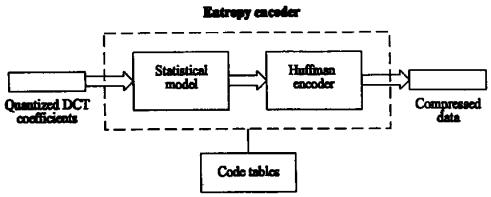


Fig. 4. JPEG baseline entropy encoder.

various steps in the entropy encoder is given later in Section VI.

In order to achieve better compression results, very often, input images are transformed to a different color space (or color coordinates) representation before being input to the encoder. Although the JPEG algorithm is unaffected by the color, since it processes each color independently, it has been shown that by changing the color space, the compression ratio can be significantly improved. This is due to the perception of the human visual system and the less perfect characteristics of the display devices. One of the most appropriate color spaces for the JPEG algorithm has been shown to be YCbCr, where Y is the luminance component and Cb and Cr are the two chrominance components. Since the luminance component carries much more information compared to the chrominance components, JPEG allows different tables to be used during compression. For additional information on how color could affect compression the reader is referred to [1]–[6].

III. Related Work

The JPEG baseline compression standard uses DCT and the Huffman entropy coding method for achieving compression. Due to the wide spectrum of applications in which DCT is used, several researchers have worked on this topic resulting in a vast amount of literature. Similarly, there exist different software and hardware approaches towards the implementation of Huffman coding. Since it is difficult to cover the entire work in the literature, we only provide a brief outline and pointers to some of the important contributions in this section.

It should be noted that two-dimensional DCT computation can be implemented as a sequence of two one-dimensional DCT's which is commonly referred to as the separability property. This approach is simpler to implement in hardware. It was shown by Haralick [24] that the DCT of N points can be computed using two N-point FFT's by exploiting the symmetry of the inputs. Later, Tseng and Miller [25] showed that the DCT can be obtained more efficiently by just computing the real part of the first N coefficients of the 2N-point DFT. The computation of 8-point DCT needed for JPEG can be replaced by 16-point DFT computation followed by scaling. An optimum form for 16-point DFT was developed by Winograd [27]. Arai, Agui, and Nakagima adapted Winograd's solution for 8-point DCT reducing the computation by using the symmetry property [21]. The hardware implementation of one-dimensional scaled DCT in our proposed architecture is based on the algorithm by Arai *et al.* [21]. Their computational flowgraph requires 5 multiplications, 29 additions and 16 two's complement operations (referred as multiplications by –1 by Arai *et al.* [21]). In the next section, we describe a modification to this algorithm that reduces the number of two's complement operations required from 16 to 12. Due to the extensive use of DCT in various applications that demand real time processing, numerous VLSI chips have been designed and built by both university and industry. For a compiled list of the different VLSI chips and their performance, the reader is referred to [20].

A class of VLSI architectures has been proposed for data transformation of tree based codes including the Huffman codes in [44]. Their algorithms use the principle of propagation of a token in a reversed binary tree constructed from the original Huffman codes. Thus the algorithms map to tree based architectures. Several other architectures have been proposed in the literature for implementing static compression techniques in [45], [46]. The codes are fixed prior to the implementation and can not be changed later on which is a disadvantage with the static schemes. A few other VLSI architectures for implementing VLC coders using sequential and concurrent VLSI models are described in [47]–[50]. In the architecture described in this paper the Huffman codes are stored in RAM modules so that the codes can be changed depending on the application.

Recently, a few special purpose VLSI chips implementing the JPEG baseline compression standard have been built and successfully commercialized. The Intel's i750 video processor [39], [40] consists of two chips, the 82 750PB pixel processor and the 82 750DB display processor. The pixel processor can be programmed to implement the JPEG compression standard. The C-CUBE CL550 is a single chip processor for JPEG image compression and decompression [38]. The core of the chip is a compression/decompression unit which consists of the FDCT/IDCT, the quantizer, the run-length encoder/decoder and the Huffman encoder/decoder. The chip can operate at up to 35 MHz. The chip can draw data at rates up to 17.5 million pixels per second and produce compressed data at a rate of approximately 2 million bytes per second. Since the entropy encoder in the chip operates at a slower speed than the DCT module a FIFO buffer is used between the two modules to avoid overflow during compression. Whenever the amount of data in the buffer reaches a certain level a delay signal is generated which stalls the DCT computation as well as the data input to the system. LSI Logic announced a chipset for JPEG compression that consists of L64735 DCT processor, L64745 JPEG coder and L74765 color and raster-block converter [41]. The chipset operates at maximum rate of 35 MHz and processes still image data at up to 30 million bytes per second. LSI Logic s JPEG chipset is described in [51]. In July 1993, LSI Logic announced a single chip JPEG coprocessor L64702 designed for graphics and video applications in personal computers, engineering workstations and laser printers [42]. The chip is capable

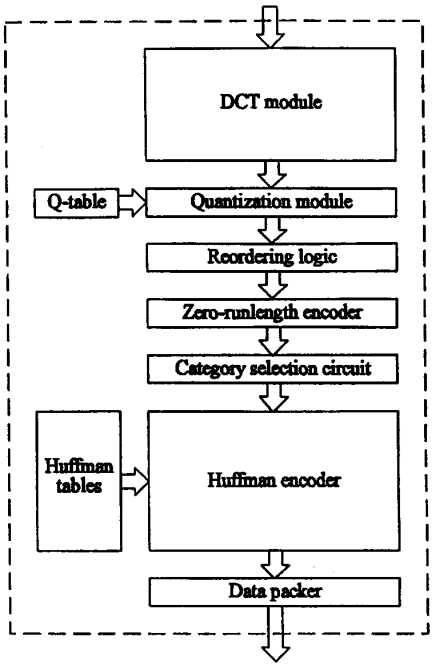


Fig. 5. JAGUAR architecture.

of compressing and decompressing data at rates up to 8.25 million bytes per second with an operating frequency of 33 MHz.

The fact that there does not exist any paper in the literature that describes the complete architecture for implementing the JPEG standard was the initial motivation for our work. From the information available on the few commercial chips described above, it was clear that we can achieve much better speeds by designing a linear static pipeline architecture with no global communication or global control logic. Such an architecture is advantageous in that higher clock speeds can be easily obtained by decreasing the granularity of processing in each stage. In other words, the clock period can be reduced by subdividing the critical delay path into smaller slices or stages.

In this paper, we propose a fully pipelined VLSI architecture for implementing the JPEG baseline compression standard. The architecture does not require any global communication or global control logic. Thus, the entire architecture can be sliced into thin stages resulting in a small clock period. The architecture for DCT and for category selection and Huffman coding in the entropy encoder are based on the efficient algorithm that lead to high speed VLSI implementation. With the architecture proposed in this paper, it is possible to obtain data compression rates of 100 million bytes per second or more.

IV. JAGUAR: JPEG VLSI Architecture

The system architecture of JAGUAR, the JPEG baseline compression chip is shown in Fig. 5. The entire architecture

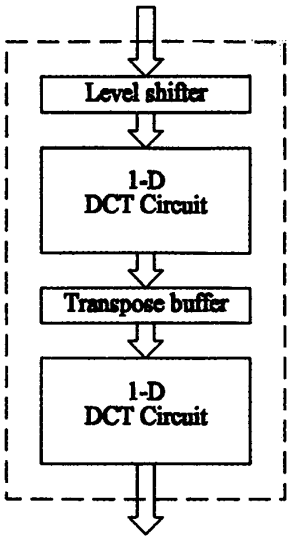


Fig. 6. DCT module.

is organized as a linear multistage pipeline in order to achieve high throughput. The hardware organization shown in Fig. 5 reflects the sequence of computation in the JPEG baseline process. The architecture consists of: 1) Encoder model and 2) Entropy encoder. The encoder model consists of DCT module, quantization module and reordering logic. The entropy encoder consists of several modules such as zero-runlength encoder, category selection circuit, Huffman encoder and data packer. The image to be compressed is input to the architecture at the rate of one pixel per clock cycle. The input data is processed by the various modules in a linear fashion where each module itself is organized internally as a multistage linear pipe. The compressed data is output by the system at a variable rate depending on the amount of compression achieved. The design of each module is described in detail in the rest of the section.

V. Encoder Model

The encoder model consists of: 1) DCT module, 2) quantization module, and 3) zigzag reordering buffer. These modules are described in the rest of this section.

A. DCT Module

The DCT module shown in Fig. 6 consists of a level shifter, two DCT circuits and a transpose buffer. As mentioned in Section II, the scaled two-dimensional DCT computation can be separated into two one-dimensional DCT operations and each one-dimensional DCT can be implemented by using modified DFT. The first DCT computation is performed row-wise and the second DCT computation is performed column-wise.

The main features of the proposed DCT module is as follows. The DCT module is based on an algorithm that reduces the number of two s complement operations from

16 to 12. The basic goal was to arrive at a design that is a multistage linear static pipeline with a small clock period. It was found that the 16-b multiplier required for the DCT computation can be reduced to a 10 × 13-b multiplier by actually examining the values of the weights and the coefficients specific for JPEG implementation. This was further verified through extensive simulations. Since the research on DCT circuits is rich and mature, the description of the DCT module design is limited only to cover the requirements for JPEG implementation.

B. DCT Algorithm

The algorithm proposed by Arai *et al.* [21] requires 5 multiplications, 29 additions and 16 two's complement additions. The algorithm is modified here in order to reduce the number of two's complement operations from 16 to 12, besides the same number of multiplications and additions. The modified algorithm is stated below:

Step 1:

$$b_0 = a_0 + a_7; \quad b_1 = a_1 + a_6; \quad b_2 = a_2 - a_4;$$
$$b_3 = a_1 - a_6; \quad b_4 = a_2 + a_5; \quad b_5 = a_3 + a_4;$$
$$b_6 = a_2 - a_5; \quad b_7 = a_0 - a_7;$$

Step 2:

$$c_0 = b_0 + b_5; \quad c_1 = b_1 - b_4;$$
$$c_2 = b_2 + b_6; \quad c_3 = b_1 + b_4;$$
$$c_4 = b_0 - b_5; \quad c_5 = b_3 + b_7; \quad c_6 = b_3 + b_6; \quad c_7 = b_7;$$

Step 3:

$$d_0 = c_0 + c_3; \quad d_1 = c_0 - c_3; \quad d_2 = c_2 \quad d_3 = c_1 + c_4;$$
$$d_4 = c_2 - c_5; \quad d_5 = c_4; \quad d_6 = c_5; \quad d_7 = c_6;$$
$$d_8 = c_7;$$

Step 4:

$$e_0 = d_0; \quad e_1 = d_1; \quad e_2 = m3 * d_2; \quad e_3 = m1 * d_7;$$
$$e_4 = m4 * d_6 \quad e_5 = d_5; \quad e_6 = m1 * d_3; \quad e_7 = m2 * d_4;$$
$$e_8 = d_8;$$

Step 5:

$$f_0 = e_0; \quad f_1 = e_1; \quad f_2 = e_5 + e_6;$$
$$f_3 = e_5 - e_6; \quad f_4 = e_3 + e_8; \quad f_5 = e_8 - e_3;$$
$$f_6 = e_2 + e_7; \quad f_7 = e_4 + e_7;$$

Step 6:

$$S_0 = f_0; \quad S_1 = f_4 + f_7; \quad S_2 = f_2; \quad S_3 = f_5 - f_6;$$
$$S_4 = f_1; \quad S_5 = f_5 + f_6; \quad S_6 = f_3; \quad S_7 = f_4 - f_7;$$

where:

a_i input elements $(0 \le i \le 7)$

S_i scaled DFT coefficients $(0 \le i \le 7)$

m_i fixed multipliers: $m1 = \cos(4\pi/16)$; $m2 = \cos(6\pi/16)$; $m3 = \cos(2\pi 16) - \cos(6\pi 16)$; $m4 = \cos(2\pi 16) + \cos(6\pi 16)$.

C. DCT Circuit Architecture

The circuit architecture for DCT computation is shown in Fig. 7. The circuit consists of six partitions as shown in the figure. Each partition contains a register set (RS) and an arithmetic unit with some associated control logic. Each register set consists of two columns of eight registers each except for the columns RS-d and RS-e which have nine registers per column. The circuit accepts one pixel per clock cycle and the entire processing is performed as a linear pipe. The algorithm described in Section V–B is mapped directly onto the architecture such that each step in the algorithm corresponds to a partition in the architecture. When the left column of register set RS-a is filled with eight data elements, the entire column is copied onto the corresponding registers in the right column. As the adder logic performs the computations as in Step 1 of the algorithm, the left column keeps receiving new input data. A similar process occurs in each of the partitions simultaneously. It takes eight clock cycles to complete all the computations in Step 1 which is the same number of cycles needed for filling up the left column. The adders are single stage units while the 13 × 10-b multiplier is a six stage Wallace tree multiplier. It will be later seen that the worst case critical path for the entire chip is the 14-b addition in the DCT computation. The DCT circuit has a latency of 59 clock cycles computed as 9 cycles per stage for all the stages except for the stage with the multiplier that requires 14 cycles. The same module is replicated for column-wise DCT computation. It should be noted that a VLSI architecture for computing DCT with a latency of less than 100 cycles has been proposed in [52].

D. Transpose Buffer

The transpose buffer is shown in Fig. 8. The buffer consists of an 8 × 8 array of register pairs organized as shown in figure. The data is input to the transpose buffer in row-wise fashion until all the 64 registers are loaded. The data in those registers are copied in parallel onto the corresponding adjacent registers which are connected in column-wise fashion. While the data is being read out from the column registers, the row registers will keep receiving further data from the DCT module. Thus, the output of row-wise DCT computation is transposed for column-wise DCT computation. The transpose buffer has a latency of 64 clock cycles.

E. Quantization Module

The quantization module is shown in Fig. 9. It consists of a RAM to store the quantization table and a 16-b multiplier. The output of DCT needs to be scaled which is done by multiplying each coefficient with the predefined scaling factor [7]. The quantization step in the JPEG algorithm involves multiplying the output of DCT with a set of predefined values from a quantization table. Since both the above steps involve multiplication the two steps are merged into a single multiplication step by suitably combining the scaling and the quantization parameters. The latency of the

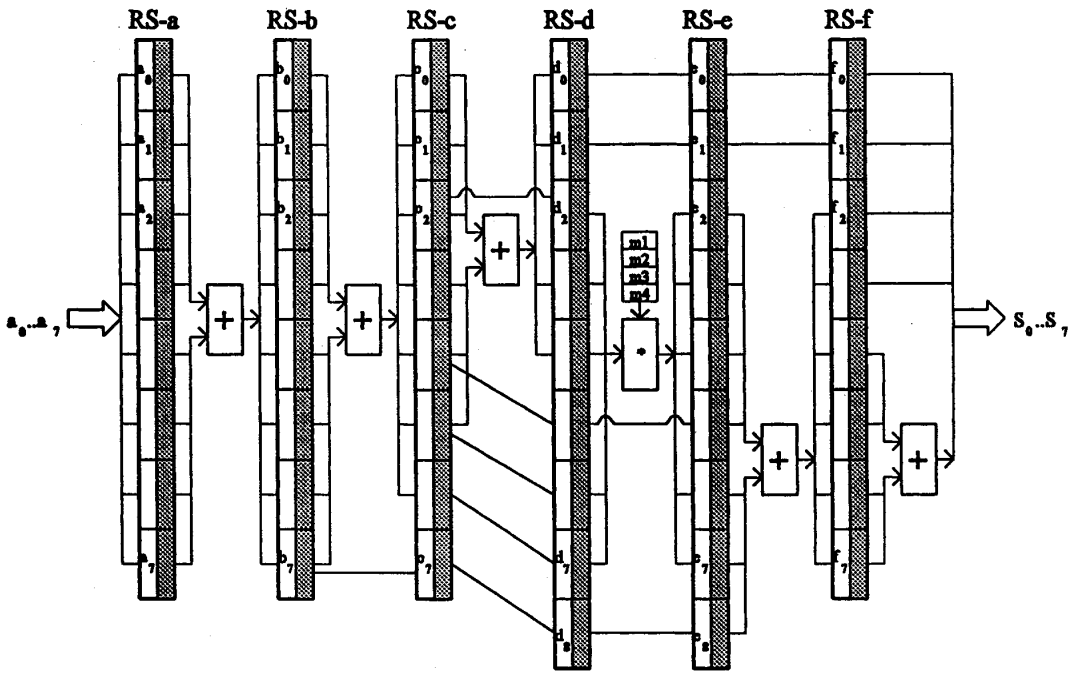


Fig. 7. One dimensional DCT circuit.

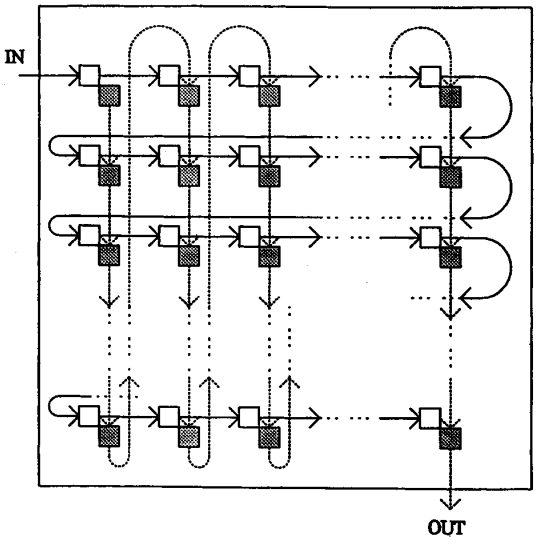


Fig. 8. Transpose buffer.

quantization module is six clock cycles which equals the number of stages in the multiplier.

F. Zigzag Reordering Buffer

Each block of data that is output by the quantization module needs to be reordered in a zigzag fashion before being forwarded to the entropy encoder. This reordering is achieved by using an 8 × 8 array of register pairs organized in a fashion similar to the transpose buffer.

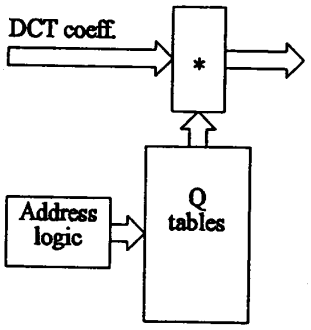


Fig. 9. Quantization module.

VI. Entropy Encoder

The function of the entropy encoder is to code the quantized coefficients from the encoder model using variable length encoding. The architecture of the entropy encoder is shown in Fig. 10. As can be seen in the figure, the entropy encoder consists of 1) zero-runlength coder, 2) category selection circuit, 3) strip logic, 4) Huffman encoder, and 5) data packer. Each block of quantized pixel data consists of one DC coefficient followed by 63 AC coefficients.

The main intention behind the design of the entropy encoder is to achieve a linear pipe with a small clock period

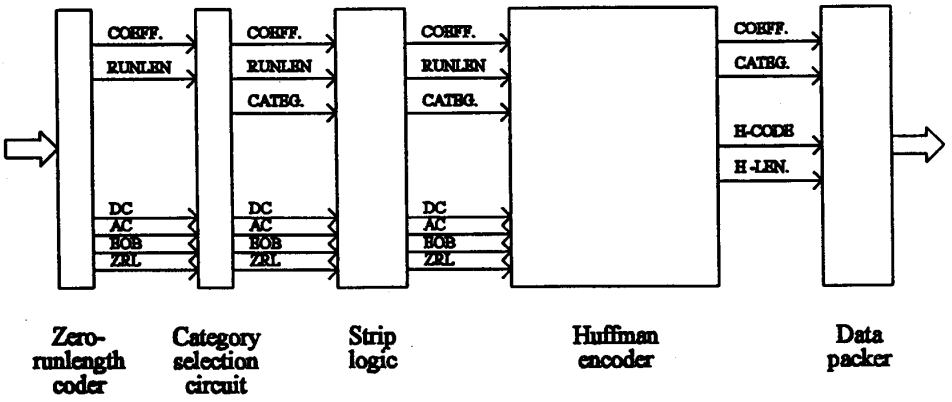


Fig. 10. Entropy encoding logic.

for each stage. In the JPEG chip described in [38], the entropy encoder operates at a slower speed than the DCT module which in turn decreases the overall compression rate possible. In the proposed design, the entropy encoder consumes input data at the same rate as the DCT module. This is achieved by carefully designing the data packer which produces output at a variable rate depending on the amount of compression.

The various steps of the entropy encoder algorithm are briefly outlined as follows. The first step is to calculate ΔDC which is the difference between the current DC coefficient and the DC coefficient of the previous block. Also, the JPEG algorithm requires that the DC/AC coefficients are decremented by one if the sign of the coefficient is negative. The next step is to extract the zero-runlength count from the stream of the AC coefficients within that block. The block data is thus converted into a stream of AC coefficients with an associated count value indicating the number of zeros preceding that coefficient. The runlength count is represented as a 4-b field. When the runlength is greater than 16, two special symbols, ZRL and EOB are used to code the data depending on certain conditions. A zero-runlength symbol ZRL (represented in JPEG as F/0) is inserted within the data whenever a runlength of 16 zeros is encountered. The end-of-block symbol EOB (represented in JPEG as 0/0) is inserted whenever it is detected that the rest of the AC coefficients until the end of the block are zeros. A 4-b status field is generated corresponding to each coefficient which indicates if the data being output is a DC or AC coefficient, ZRL or EOB symbol. The above steps are performed within the zero-runlength coder.

Within the category selection circuit, each DC and AC coefficient is associated with a corresponding category depending on the magnitude of the coefficient. The definition of categories as defined by the JPEG standard is shown in Table 1. Each element in the stream of data coming out of the category selection unit consists of coefficients, the corresponding category, the runlength count and the four-bit status. It should be noted that the data stream still contains all 64 coefficients including the streaks of zero coefficients which have been encoded as zero runlength counts. Also if an EOB symbol follows one or more ZRL symbols within the data stream the ZRL symbols are redundant and must be stripped off the data stream. The above functions are performed within the strip logic.

Table 1 JPEG Category Definitions [7]

Cat.	DC difference	AC coeff
0	0	
1	-1, 1	-1, 1
2	-3, -2, 2, 3	-3, -2, 2, 3
3	-7···-4, 4···7	-7···-4, 4···7
4	-15···-8, 8···15	-15···-8, 8···15
5	-31···-16, 16···31	-31···-16, 16···31
6	-63···-32, 32···63	-63···-32, 32···63
7	-127···-64, 64···127	-127···-64, 64···127
8	-255···-128, 128···255	-255···-128, 128···255
9	-511···-256, 256···511	-511···-256, 256···511
10	-1023···-512, 512···1023	-1023···-512, 512···1023
11	-2047···-1024, 1024···2047	

During the next step, each data element consisting of ⟨AC/DC coefficient, runlength count, category, status⟩ output by the strip logic is converted into a corresponding element: ⟨AC/DC coefficient, category, Huffman code, Huffman code length⟩. The Huffman code is selected based on the runlength count, category and status fields. The set of Huffman codes are prestored in a table and can be changed depending on the application. The category and the Huffman code length fields are used in the data packer unit to pack the variable length compressed data (comprised of DC/AC coefficient and the Huffman code) into a stream of fixed length compressed data units to be output by the compression chip. The implementation

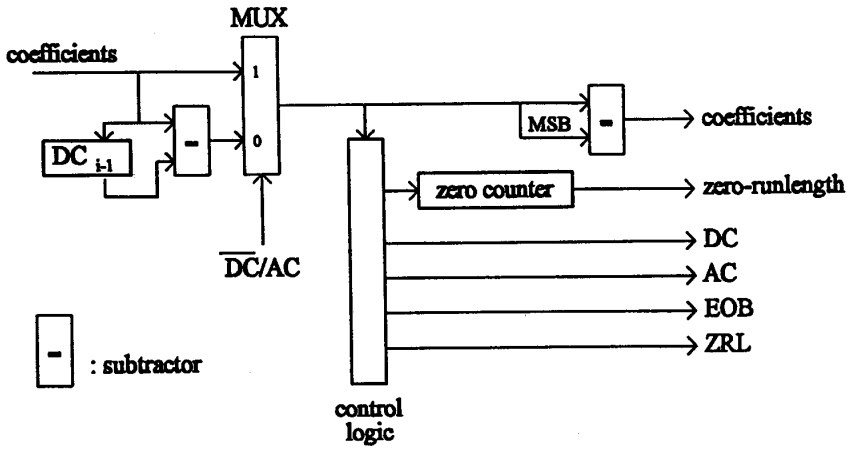


Fig. 11. Zero-runlength coder.

of each module within the entropy encoder architecture is described below.

A. Zero-Runlength Coder

The zero-runlength coder module performs the functions as described in the earlier part of this section. The module consists of three stages and thus a latency of three cycles. The first stage consists of logic for computing ΔDC while the second stage derives the runlength count and the third stage is used for decrementing negative coefficients. The various stages of the zero-runlength coder are shown in Fig. 11.

B. Category Selection Circuit

The category selection is defined in the JPEG compression standard as shown in Table 1. A straightforward implementation of category selection would require storing the ranges in memory and comparing the input data with those prestored values which requires complex address decoding and control logic. However, the table memory can be avoided and the entire category selection can be achieved with a simple combinational circuit. This circuit operates like an encoder which converts the given coefficient into the corresponding category in a single clock cycle. The circuit is given in Fig 12. However, it should be noted that the negative coefficients must be decremented by one before applying the conversion logic as per the JPEG standard.

C. Strip Logic

The strip logic shown in Fig. 13 consists of four stages. Each stage has three registers to hold the coefficient, runlength count and category fields corresponding to a data element output by the category selection circuit and a set of one-bit registers to hold the corresponding status. The status bits are decoded and used to strip the zero-valued coefficients and also to strip off the ZRL symbols that precede an EOB symbol. It should be noted that there could be a maximum of three ZRL symbols preceding an EOB symbol. The strip logic acts as a four stage buffer through which the compressed data elements after the removal of zero coefficients travel before being forwarded to the Huffman encoder. The valid bit signal is set to high whenever valid data is being output by the strip logic for Huffman encoding. It should be noted that the ZRL bit needs to be reset whenever a ZRL symbol has been deleted from the data stream.

D. Huffman Encoder Module

The Huffman encoder module consists of Huffman code tables stored in random access memory modules and logic for replacing the category, runlength count pairs with the corresponding Huffman codes. Although the size of the DC coefficient code table is small, the code table storage for AC coefficients is relatively large. In order to keep the clock period small the memory for the code tables is organized as a set of five RAM modules arranged in a linear pipeline fashion. The idea is to reduce the access time by keeping the memory size small. The table is accessed by using the runlength, category pair for addressing. The input data passes through each of the five stages and depending on the address the corresponding Huffman code and the code length are output. The hardware organization is shown in Fig. 14 which is self explanatory.

E. Data Packer

The data packer unit shown in Fig. 15 is used to convert variable length compressed data into fixed length compressed data stream. The logic consists of registers A and B, two left-shift units, two multiplexers and control logic which includes two registers A-length and B-length. The data packer works as follows. The Huffman code is first loaded into register A left justified. Depending on the length of the Huffman code, the coefficient is loaded through a multiplexer into register A, bit-aligned with the Huffman code. It should be noted that the total length of the Huffman code and the coefficient cannot exceed 26 bits and the

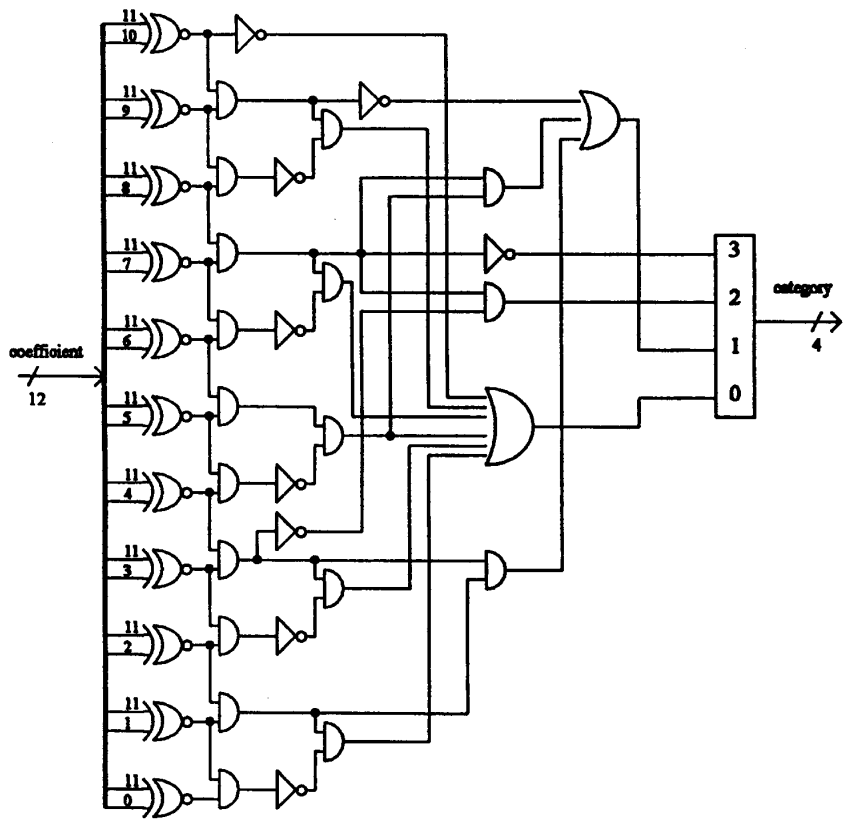


Fig. 12. Category selection circuit.

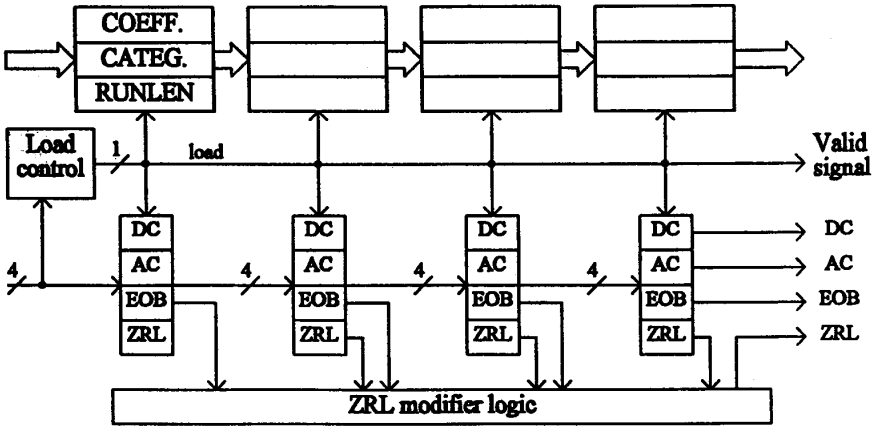


Fig. 13. Strip logic.

length information is loaded into the A-length register. The data in register A is loaded through a left shifter unit and a multiplexer into register B. The loading of new data into register B is controlled using the shifter and the multiplexer which are in turn controlled by the values in the A-length and the B-length registers. Whenever register B has more than 32 b of information which is indicated by the B-length register value a 32-b compressed data is output. A similar

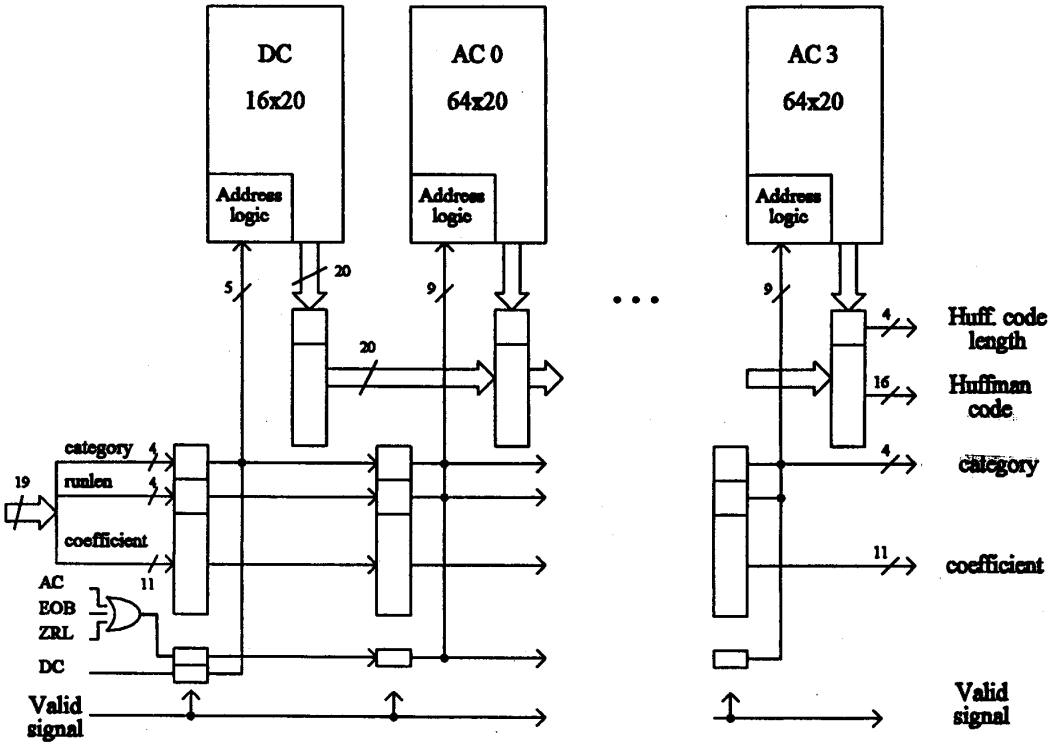


Fig. 14. Huffman encoder.

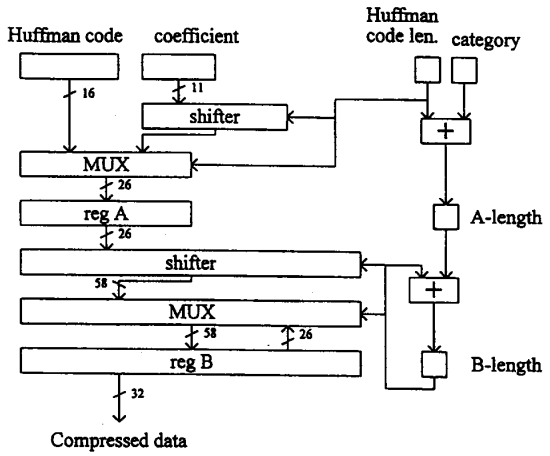


Fig. 15. Data packer.

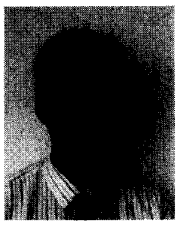
logic can be used for byte-stuffing required by the JPEG standard.

VII. Conclusions

The proposed VLSI architecture was simulated and verified for functional correctness using Verilog and Veritime. The entire architecture is organized as a linear multistage pipeline. Thus the clock period can be reduced by increasing the level of fine grain parallelism. Since the entire architecture requires only a few major components such as three 13 × 10-b multipliers, a few adders and small random access memory modules, the architecture can be realized as a single VLSI chip. A prototype VLSI chip implementing a part of the proposed architecture (for the one-dimensional DCT) designed using 2-μm CMOS technology and the Cadence design tools has been submitted to MOSIS for fabrication. The circuit was fit on a 6.8 mm × 6.9 mm MOSIS standard frame. While the implementation of the entire architecture would be expensive to fabricate, it was decided to prototype the DCT portion of the circuit that contains the worst case delay critical path for the proposed architecture. The details of the prototype implementation are omitted since the chip is currently still under fabrication. Based on the prototype chip implementation, it is estimated that the entire JPEG chip can be implemented on a silicon area of 12 mm × 14 mm. For the architecture described in this paper, the critical path of the chip depends on a 14-b adder circuit which can be easily achieved with a 10 ns clock period using 1 μm CMOS process. It should be noted that the 14-b addition can be done in two clock cycles with the use of two 7-b adders organized in two stages if a faster clock is required. Thus the proposed chip can function with an operating frequency of 100 MHz. This would allow an input rate of 100 million pixels per second leading to a rate of 30 frames per second for 1024 × 1024 color images.

REFERENCES

[1] R. W. G. Hunt, *Measuring Color*. New York: Halsted, 1987.
[2] A. N. Netravali and B. G.Haskell, *Digital Pictures, Representation and Compression*. New York: Plenum, 1988.
[3] J. D. Foley *et al., Computer Graphics*. New York: Addison Wesley, 1992.
[4] R. Salmon and M. Slater, *Computer Graphics*. New York: Addison-Wesley, 1987.
[5] D. R. Clark, *Computers for Imagemaking*. Oxford, UK: Pergamon.
[6] D. Travis, *Effective Color Displays*. San Diego: Academic, 1991.
[7] ISO/IEC, Int. Standard DIS 10918, "Digital compression and coding of continuous-tone still images. "
[8] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1993.
[9] G. K. Wallace, "The JPEG still picture compression standard," CACM, vol. 34, no. 4, pp. 31–44, 1991.
[10] A. Leger, T. Omachi, and G. K. Wallace, "JPEG still picture compression algorithm," *Optical Engineering*, vol. 30, no. 7, pp. 947–954, 1991.
[11] M. Rabbani and P. W. Jones, *Digital Image Compression Techniques*. Washington: SPIE Press, 1991.
[12] H. Lohscheller and U. Franke, "Colour picture coding—Algorithm optimization and technical realization," *Frequenz*, vol. 41, no. 11/12, pp. 291–299, 1987.
[13] M. L. Liou and T. R. Hsing, "An overview for video signal processing," *IEEE Symp. Circuits and Systems*, Philadelphia, PA, pp. 208–212, May 1987.
[14] J. L. Mitchell and W. B. Pennebaker, "Envolving JPEG color data compression standards," *Standards for Electronic Imaging Systems*, SPIE, vol. CR37, pp. 68–97, 1991.
[15] G. Wallace, R. Vivian, and H. Poulsen, "Subjective testing results for still picture compression algorithms for international standardization," *Proc. IEEE Global Telecomm. Conf.*, Hollywood, pp. 1022–1027, Nov/Dec. 1988.
[16] A. Leger, J. L. Mitchell, and Y. Yamazaki, "Still picture compression algorithms evaluated for international standardization," *Proc. IEEE Global Telecomm. Conf.*, Hollywood, pp. 1028–1032, Nov./Dec. 1988.
[17] G. P. Hudson, H. Yasuda, and I. Sebestyen, "The international standardization of a still picture compression technique," *Proc. IEEE Global Telecomm. Conf.*, Hollywood, pp. 1016–1021, Nov./Dec. 1988.
[18] P. H. Ang, P. A. Ruetz, and D. Auld, "Video compression makes big gains," *IEEE Spectrum*, pp. 16–19, Oct. 1991.
[19] A. K. Jain, "Image data compression: A review," *Proc. IEEE*, vol. 69, no. 3, pp. 349–389, 1981.
[20] K. R. Rao and P. Yip, *Discrete Cosine Transform*. San Diego, CA: Academic, 1990.
[21] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E71, no. 11, pp. 1095–1097, 1988.
[22] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing*. New York: Addison Wesley, 1993.
[23] H. Lohscheller, "Vision adapted progressive image transmission," *Signal Processing II*, H. W. Schussler, Ed. Amsterdam: North-Holland, 1983, pp. 191–194.
[24] R. M. Haralick, "A storage efficent way to implement the discrete cosine transform," *IEEE Trans. Comp.*, vol. C-25, pp. 764–765, July 1976.
[25] B. D. Tseng and W. C. Miller, "On computing discrete cosine transform," *IEEE Trans. Comp.* vol. C-27, no. 10, pp. 966–968, 1978.
[26] H. F. Silverman, "An introduction to programming the winograd fourier transform algorithm (WFTA)," *IEEE Trans. ASSP*, vol. ASSP-25, no. 2, pp. 152–165, 1977.
[27] S. Winograd, "On computing the discrete fouriere transform," *Mathematics of Computation*, vol. 32, no. 141, pp. 175–199, 1978.
[28] "IEEE standard specifications for the implementations of 8 × 8 inverse discrete cosine transform," *IEEE Std*, pp. 1180-1990.
[29] E. Linzer and E. Feig, "New scaled DCT algorithms for fused multiply/add architectures," *ICASSP*, Toronto, pp. 2201–2204, May 1991.
[30] E. Feig, "A fast scaled-DCT algorithm," *Proc. SPIE*, Santa Clara, CA, vol. 1224, pp. 2–13, Feb. 1990.
[31] H. A. Peterson, H. Peng, J. H. Morgan, and W. B. Pennabaker, "Quantization of color image components in the DCT domain," *Proc. SPIE*, pp. 210–222, Feb. 1991.
[32] M. Vetterly and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol. 6, pp. 267–278, 1984.
[33] W. Chen, C. H. Smith, and S. C. Fralick, "A fast computational agorithm for the discrete cosine transform," *IEEE Trans. Communications*, vol. COM-25, no. 9, pp. 1004–1009, 1977.
[34] P. Duhamel and C. Guillemot, "Polynomial transform computation of the 2-D DCT," *Proc. ICASSP*, Albuquerque, pp. 1515–1518, Apr. 1990.
[35] A. Lightenberg and J. H. O'Neill, "A single chip solution for an 8 × 8 two dimensional DCT," *Proc. IEEE Symp. Circuits and Systems*, Philadelphia, pp. 1128–1131, May 1987.
[36] L. McMillan and Lee Westover, "A forward mapping realization of the inverse discrete cosine transform," *Proc. Data Compression Conference*, Snowbird, pp. 219–228, Mar. 1992.
[37] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comp.*, vol. C-23, pp. 90–93, 1974.
[38] *CL550 Users Manual*, C-Cube Microsystems, Milpitas, CA, 1992.
[39] *82750PB Pixel Processor databook*, INTEL, Santa Clara, CA, Oct. 1993.
[40] *82750DB Display Processor databook*, INTEL, Santa Clara, CA, Sept. 1993.
[41] *JPEG Chipset Technical Manual*, LSI Logic, Milpitas, CA, Jan. 1993.
[42] *L64702 JPEG Coprocessor Technical Manual*, LSI Logic, Milpitas, CA, July 1993.
[43] N. Ranganathan, *VLSI Algorithms and Architectures*, IEEE Comp. Soc. Press, Los Alamitos, 1993.
[44] A. Mukherjee, N. Ranganathan, and M. Bassiouni, "Efficient VLSI designs for data transformations of tree-based codes," *IEEE Trans. Cir. Sys.*, vol. 38, pp. 306–314, Mar. 1991.
[45] A. Mukherjee, N. Ranganathan, J. W. Flieder, and T. Acharya, "MARVLE: A VLSI chip for data compression using tree-based codes," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 203–214, June 1993.
[46] M. T. Sun, "VLSI architecture and implementation of high speed entropy decoder," *Proc. ISCS*, pp. 200–202, 1991.
[47] M. T. Sun, K. M. Yang, and K. H. Tzou, "A high speed programmable VLSI for decoding variable length codes," *Proc. SPIE*, vol. 1153, Aug. 1989.
[48] S. F. Chang and D. G. Messerschmitt, "Designing high throughput VLC decoder: Part I—Concurrent VLSI architectures," *IEEE Trans. Circ. and Sys. for Video Tech.*, June 1992.
[49] J. L. Sicre and A. Leger, "Silicon complexity of VLC decoder vs. Q-coder," *CCITT* Feb. 1989.
[50] M. T. Sun and S. M. Lei, "A parallel VLC decoder for advanced television applications," *Proc 3rd Int. Workshop on HDTV*, Aug. 1989.
[51] D. A. Luthi, P. Tong, and P.A. Ruetz, "A video-rate JPEG chip set," *IEEE 1992 Custom Integrated Circuits Conf.*, May 1992.
[52] M. Maruyama *et al.*, "VLSI architecture and implementation of a multi-function, forward/inverse discrete cosine transform processor," *SPIE* vol. 1360, Visual Communications and Image Processing '90, pp. 410–417, 1990.



Mario Kovac received the B.S. and M.S. degrees in computer science and engineering from the Faculty of Electrical Engineering, University of Zagreb, Croatia, in 1988 and 1991, respectively, where he is working towards the Ph.D. degree.

He has been on the faculty of the University of Zagreb since 1989 and is currently holding a scientific assistant position. During 1990, 1991, and 1993, he was a visiting Research Scholar at the University of South Florida, Tampa. His research interests include computer architecture, parallel processing, VLSI and implementation of algorithms and architectures in hardware (on both PCB and chip level).

N. Ranganathan (Senior Member, IEEE) was born in Tiruvaiyaru, India, in 1961. He received the B.E. (honors) degree in electrical and electronics engineering from the Regional Engineering College, Tiruchirapalli, University of Madras, India, in 1983 and the Ph.D. degree in Computer Science from the University of Central Florida, Orlando, FL, in 1988.

He is currently an Associate Professor in the Department of Computer Science and Engineering and the Center for Microelectronics Research at the University of South Florida, Tampa. His teaching and research interests include VLSI design and hardware algorithms, computer architecture and parallel processing. He is currently involved in the design and implementation of VLSI architectures for computer vision, image processing, pattern recognition, databases, data compression, and signal processing applications.

Dr. Ranganathan is a member of the IEEE Computer Society, the IEEE Computer Society Technical Committee on VLSI, the ACM and the VLSI Society of India. He served as the Program Co-Chair for VLSI Design '94 and the General Co-Chair for VLSI Design '95. He is also on the Program Committees of ICCD, ICPP'95 and IPPS '95, IEEE SPDP ('95), and ICHP ('95). He is the Program Chair for the IEEE Computer Society Annual Workshop on VLSI, April 1995, to be held in Clearwater, FL. He serves on the editorial boards of Pattern Recognition and VLSI Design. He is the guest editor of a special issue of International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) to be published in 1995. He is the editor of a two-volume series on VLSI Algorithms and Architectures published by IEEECS Press in June 1993.