Associative Processors for Video Coding Applications

Subhash Balam and Dan Schonfeld, *Senior Member, IEEE*

***Abstract*—Associative processors are single-instruction multiple-data-based architectures that provide huge parallelism which can be very useful for processing video in real time. This paper describes architectural requirements for video and shows that associative processors are ideally suited for applications involving video. Some routines like discrete cosine transforms (DCTs) lack required parallelism and their implementation becomes critical for the overall performance of the processor for video. We present three different approaches for computing two-dimensional DCT on associative processors and discuss their implementation details. Many methods like conditional execution of DCT, have been suggested in literature for power efficiency. In this paper, we discuss integration of these techniques in our implementation. Finally, performance of Associative processors is compared with that of RISC and DSP processor and show that associative processors are better suited for processing video both in terms of speed and power.**

***Index Terms*—Associative processors, content-addressable memory (CAM), discrete cosine transform (DCT), video.**

I. Introduction

RECENT advances in communication have shown significant improvement in multimedia applications involving video. With wireless communication standards like Wifi and 3G wireless providing enough bandwidth for video, there has been a growing demand for hand-held devices capable of encoding and decoding video in real-time. The multimedia platforms used in such devices must provide enough computational power to support video within the real-time constraints.

Video compression algorithms exploit the spatial and temporal redundancies to achieve compression. Discrete cosine transform reduces the spatial redundancy while motion compensation reduces temporal redundancy in a video sequence. Though the routines in video coding are computationally intensive, there is large amount of data parallelism involved in the video coding routines. Associative processors exploit this parallelism for processing video in real time.

Block diagram of a video coding algorithm is shown in Fig. 1. Typically in video coding standards like H.263 and MPEG4 simple profile,individual frames of a Video sequence are divided into Macro blocks of 16×16 pixels which are further divided into blocks of 8×8 pixels. These blocks of video data are transformed using discrete cosine transform (DCT). DCT reduces the spatial redundancy by representing the data in fewer coefficients. There are four different type of DCT. Type II DCT is most popularly used in image and video coding standards as it approximates Karhaunan–Loeve transform for first-order Markov process with high correlation.

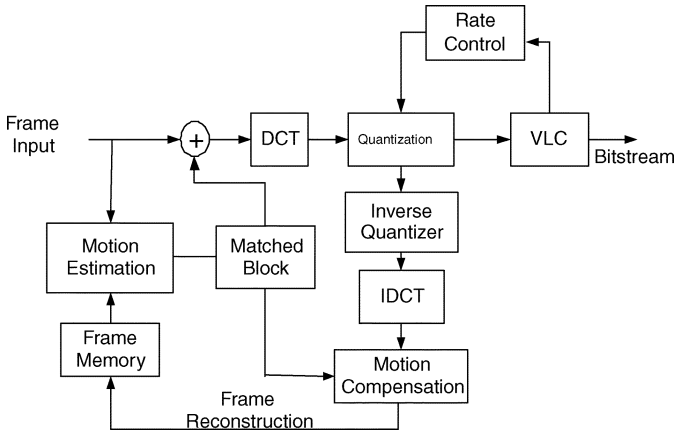

Fig. 1. Block diagram of encoder.

Motion estimation of the macroblocks reduces the temporal redundancy. In this step, a macroblock is compared with the pixels of previous image to find a match. The motion estimation is performed up to subpixel resolution for better quality. Sum of absolute difference (SAD) is most commonly used criteria because of its simplicity in implementation. The residue is then coded using DCT.

DCT coefficients are further quantized to achieve further compression. Quantization is a lossy stage and it usually retains low frequency components in DCT domain. During quantization, most of the high frequency coefficients are reduced to zero. Finally, quantized DCT coefficients and motion vectors from motion estimation are entropy coded to get the compressed bit stream.

In this paper, we first study the computational complexity and available parallelism in the video coding algorithm. We look at various approaches for fast video coding. Then we describe the architecture of a content-addressable memory (CAM)-based associative processor and how it can be efficient in video coding. Parallelism available in performing DCT is limited and depends on the implementation. This paper describes three different approaches for implementing DCT on associative processor.

II. Architectural Requirements for Fast Video Coding

The computational requirements of video processing are a direct result of high data rates associated with a video sequence. Table I shows the percentage of processor usage for QCIF video stream at 15 frames/sec on a RISC processor. The analysis shown is using three step search motion estimation algorithm including motion estimation at half-pixel resolution. Chen DCT algorithm proposed in [1] was used in implementation.

Manuscript received August 16, 2004; revised August 24, 2005. This paper was recommended by Associate Editor R. Chandramouli.

S. Balam is with Qualcomm Inc., San Diego, CA 92122 USA (e-mail: bsubhash@qualcomm.com).

D. Schonfeld is with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607-7053 USA (e-mail: dans@uic.edu).

Digital Object Identifier 10.1109/TCSVT.2005.858696

TABLE I
CPU LOAD FOR ENCODER FUNCTIONS

Encoder Function	CPU Load(in percentage)
Motion Estimation	5.5
SAD Calculation	71
DCT/IDCT	10
Q/IQ	2.2
VLC	1.1
Motion Compensation	2.2
Miscelleneous	6.67

A. Approaches for Fast Video Coding

Motion estimation algorithms follow a predefined sequence to find a matching block in the previous image. The search procedure usually reduces the number of search locations which reduce the number of arithmetic operations to a large extent. Two-dimensional (2-D) DCT can be performed with one-dimensional 1-D DCTS to reduce the order of multiplication. Algorithms for fast DCT are described in [1], [2], and [3]. In addition to using these optimized algorithms for motion estimation and DCT, video coding using following approaches are power efficient schemes that reduces the number of operations performed on an average.

Early Exit From Motion Estimation: The motion estimation can be suspended once the minimum matching criteria is reached. In test cases like Mother and Daughter, News Reader, most the background remains constant. By using early exit criteria, only 25%–27% of the macroblocks proceed to second SAD computation.

DCT Computation Criteria: DCT is computed only if the matching criteria was above a predefined threshold. The matching criteria used for motion estimation gives enough information about amount the content in the residue. If the content in the residue is significantly less, often the DCT coefficients will be quantized to zero.

Simulations by Liao and Wolfe in [4] show that available Instruction level Parallelism on video processing varies from 32–1000.To meet the real-time requirements of video, it is important that this parallelism has to be exploited. It is therefore important that to study the parallelism available in individual blocks to find an appropriate architecture suitable for video.

Motion estimation, quantization, inverse quantization and motion compensation involve stages in which single operation is performed in multiple data. This operation on the data is largely independent and can be performed in parallel. Available parallelism in DCT is implementation dependent. Unlike other functional blocks of the encoder, fast coding DCT algorithms involve multiplications in which the operands are result of previous operation.

Requirements for fast video processing can be summarized as follows.

1) Single-instruction multiple-data (SIMD) instruction set for simultaneous data processing. SIMD instruction exploits the available data parallelism in video coding. SIMD instructions also help to overcome slow clock rates of the processor for processing large amount of data.
2) Lower data traffic by providing large number of registers. Video is a high bandwidth application and involves huge amount of data. Fewer registers would lead to frequent data traffic from external memory to the register to store intermediate data.
3) Must possess a multiply and accumulation unit for faster computation of DCT. DCT is only routine in video coding that requires a multiplier unit. Lack of multiplier unit would reduce overall performance of the application platform for video coding.

SIMD instruction set is necessary to overcome huge number of operations involved in video. Associative processors provide the required parallelism that can be very effective in performing Motion estimation, however it lacks a multiplier unit resulting in complex DCT implementation.

III. ARCHITECTURE OF ASSOCIATIVE PROCESSING ARRAY (APA)

Associative processor consists of an array of content addressable memory bits. All arithmetic operations are performed on the processor core using a series of compare and write operations. The processor core consists of mask, pattern and a tag register. In the compare operation, the mask register sets up series of bits that will be used for the comparison. Pattern register sets the bits against which the comparison is to be performed. The result of the comparison is stored in tag register for each row. In the write operation, the tag register sets up the rows in which the write operation is performed. The bits to be in each row are chosen from the bits set in the mask register and the bits are modified as set up in pattern register. Unlike the RISC architecture, bits in associative processors are addressed by association and, hence, the name associative processors. Data I/O is performed using a region machine that operates in parallel to the operations performed on associative array. Shift operation on the tag register is useful to move data inside the core. Instruction decoder dispatches the instructions to the associative processor and the serial processor. The serial processor performs simple arithmetic operations involving fewer data.

Arithmetic operations were initially performed by implementing a truth table for addition and subtraction. The performance of the associative processing array (APA) is enhanced by performing following modifications to the architecture.

1) By using multiple tag registers to hold multiple results of compare operation.
2) A full bit adder across tag registers, removing the delay involved in truth-table approach.
3) A parallel set mask and compare/write instruction effectively reducing the number of cycles to half the original value.

The architecture of the APA processor and its associated registers are depicted in Figs. 2 and 3, respectively.

A. APA Advantage

The main advantage of an associative processor is its simplicity. It provides massive scale of parallelism due to which effective number to cycles for each arithmetic values is reduced to

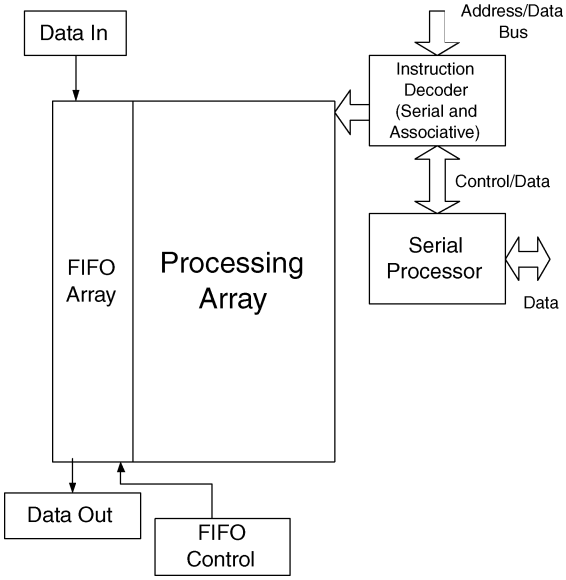

Fig. 2. Architecture of APA.

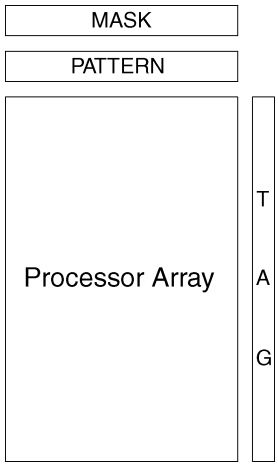

Fig. 3. Processing array and registers of associative processor.

fractional values. An associative processor performs an 8 bit addition in 25 cycles. For an array of size 512, the effective number of cycles to nearly 0.05.

Advantages of using associative processors for video coding can summarized as follows.

1) SAD values for each macroblock can be computed in less than 135 cycles. The number of cycles further reduce depending on the number of SADs performed in parallel. On a load-store model RISC/DSP architecture with SIMD instruction set for four parallel additions, SAD operation requires minimum of 300 cycles for each macroblock.
2) APA architecture handles high data rate in video through the region machine that operates in parallel to the operations inside the APA core.
3) The parallelism advantage can be extended to motion compensation and quantization routines. Quantization per macro block would require less that 180 cycles and motion compensation round 35 cycles.
4) Operations in associative processors are performed to required bit resolution. In RISC architectures the bit resolution is always fixed to 16 bit or 32 bit.These are many redundant operations that contribute for increase in power.

Associative processor provides a platform that handles all the requirements of video processing efficiently. However, lack of a multiplier unit and limited amount of parallelism makes implementation of DCT more expensive and more critical for performance of Associative processor. A simple implementation of DCT could be computing each coefficient as a projection operation on individual basis of DCT. This can be accomplished by performing eight parallel multiplications followed by a pyramidal addition (see Fig. 6). Disadvantage of this approach is that DCT basis has to be loaded into the array, which means 64 coefficients of required bit resolutions have to be loaded before computing DCT. Due to this additional data, the number of bits available for temporary storage during multiplication operation is reduced. This results in a higher traffic which is eight times the amount of video data itself. In the following section we describe three different approaches for performing 2-D DCT on associative processors with lower data traffic and more efficient in implementation.

IV. 2-D Transform on Associative Processors

Different approaches discussed in this section can be extended to other transforms also. The performance explained in this section is based on implementation on an APA of 512 rows and 160 columns with 32 columns used specifically for first-in first-out (FIFO) implementation for data transfer.

A. Multiple DCT Computation

Simple approach to overcome the lack of parallelism in performing DCT is to perform multiple DCTs simultaneously in each row of the processor array. In an array of 512, 256 1-D DCT/IDCTs can be performed in parallel. The reduction in the number of DCTs compared to rows is because of the buffer requirement for computing multiple stages of DCT to 12 bit resolution that is required for standard compliance. Y. Shain, A. Akerib, and R. Adar in [5] show that such an implementation on IDCT can be performed in 32 cycles on a 2-K array. However, the implementation performed was using truth table approach.

The enhanced APA, integrated with adder/subtractor in the tag registers results in rapid decrease in the cycles. Simulations shows that 1-D DCT can be performed in 3826 cycles. The row column transpose is particularly more expensive in this method. Each bit of 8×8 block is transposed using shift operations on tag registers. Using the parallel mask and compare/write operations the transpose operation can be completed in 2000 cycles. Another set of 1-D DCT is performed as done initially. Due to increase in range of values after 1-D DCT, the net increase in cycles for second set of 1-D DCT is 4248 cycles. 2-D DCT can be implemented in nearly 9800 cycles. Since 32 2-D DCTs are performed in parallel, the number of cycles per DCT is around 325 cycles. These results on based on implementation of fast DCT proposed in [2].

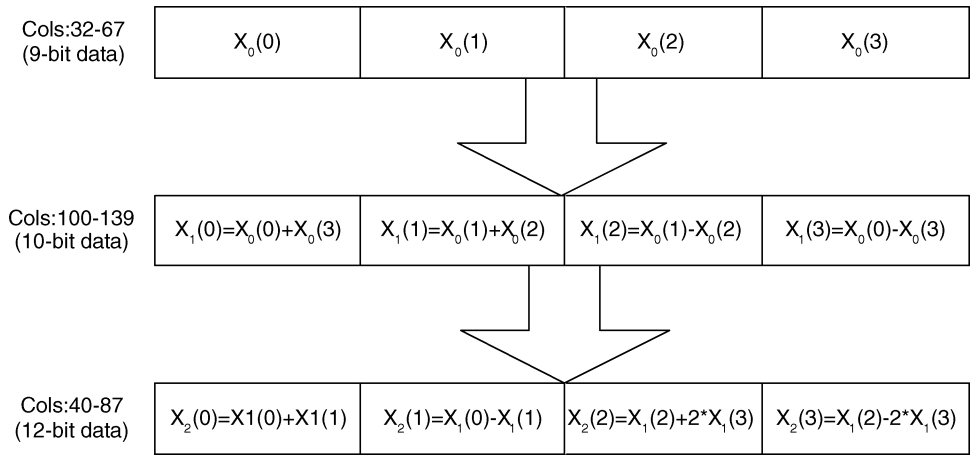

Fig. 4. Integer transform on APA.

Multiple DCT computation approach is used to perform 64 2-D-DCTs (at reduced resolution) and 32 2-D inverse DCT (IDCTs) in parallel. The main drawback of this method is maintaining state of the encoder. The state of encoder is required to implement conditional DCT execution for block matching cases. This method also requires additional memory and higher traffic as the processor memory has to be flushed before proceeding to next stage of execution.

1) Integer Transform on APA: Integer transform explained in [6] is adopted in H264 video coding standard and version 10 of MPEG4 coding standard. The proposed transform completely removes the mismatch that appears in IDCT implementation in previous coding standards. The paper also describes a fast algorithm for implementing the integer transform.

The inherent parallelism in APA can be very useful in implementing the fast coding technique presented in [6] as it uses only shift and add operation.The fast coding technique proposed is depicted in Fig. 4 and uses eight additions in each of the two stages. Due to the positional control, shift operation can be completely avoided in APA. Since the transform operates on 4×4 data,transpose can be implemented in less than 500 cycles. The 2-D integer transform can be implemented in 1050 cycles. On an array of 512 rows 128, such transforms can be computed in parallel, effectively bringing net cycles to less than 12 cycles.

B. Pyramid Computation of DCT

The drawbacks of computing DCT using the previous approach can be avoided by using pyramidal approach for computing DCT. The idea is based on optimized DCT matrix proposed by Pan in [7]. The optimized DCT matrix reduces the number of adders at each bit resolution and computes the resultant DCT using shift and add approach. Similar ideas were proposed in [8] and [9].The current approach works on similar idea but the implementation is based on modified DCT array.

DCT in pyramidal approach is computed by performing a series of additions and shifts as required in the bit positions of coefficients of each basis vector. This method of performing arithmetic at bit level is called distributed arithmetic.Each DCT coefficient can be computed by the (1). A_k are the coefficients from 1-D DCT basis vector an x_k are input data

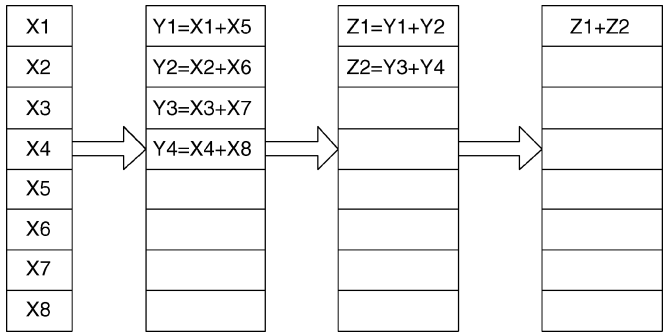

Fig. 5. Pyramidal addition.

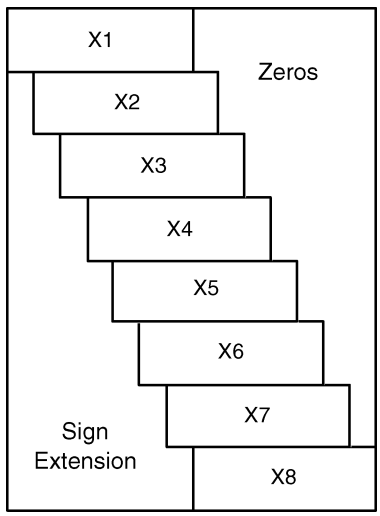

Fig. 6. Storage of the computed values at each bit resolution.

$$y_i = \sum_{k=1}^{8} A_k^i x_k. \tag{1}$$

A_k can be represented in 2s complement form as given below

$$A_k^i = -b_{k0}^i + \sum_{j=1}^{t} 2^{-j} b_{kj}^i \tag{2}$$

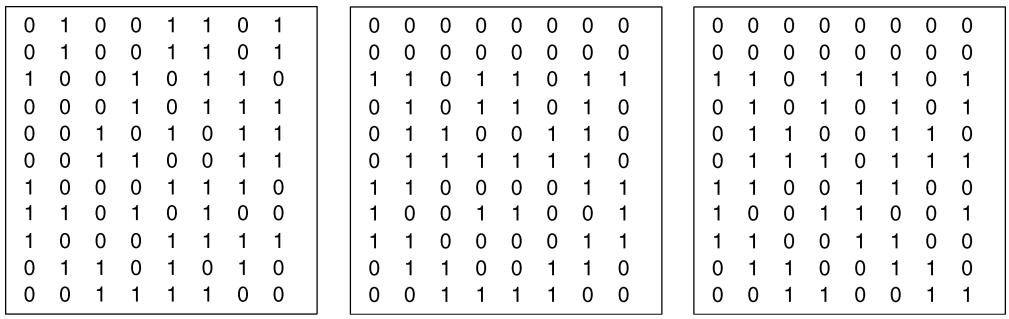

Fig. 7. Bit patterns for computing fifth coefficient.

substituting the 2s complement value of A_k^i, y_i can be computed as follows

$$y_i = \sum_{k=1}^{8}(-b_{k0}^i * x_k) + \sum_{k=1}^{8}\sum_{j=1}^{t}(2^{-j}b_{kj}^i) * x_k \tag{3}$$

$$y_i = \sum_{j=1}^{t} 2^{-j} X_j \tag{4}$$

where X_j represents the summation at each bit resolution. The resultant pyramidal addition and storage of the summation values at bit positions consistent with the shift operations is depicted in Figs. 5 and 6, respectively. This computation however is very inefficient and results in huge number of operations which can consume nearly 12 000 cycles for 1-D DCT alone.

The first level of optimization that can be done is by performing a absolute addition on coefficients A_k and moving the sign to the sample values. This operation is represented in

$$y_i = \sum_{k=1}^{8} \mathrm{abs}(A_k^i)(\mathrm{sign}(A_k^i) * x_k) \tag{5}$$

substituting $z_k^i = \mathrm{sign}(A_k^i) * x_k$ and $C_k^i = \mathrm{abs}(A_k^i)$ the equation can be written as follows:

$$y_i = \sum_{k=1}^{8} C_k^i * z_k^i. \tag{6}$$

This reduces the need for performing summations at two most-significant bit positions. This directly results in reduction of nearly 600 cycles per bit resolution.

Further optimization in the computations is performed by re-arranging the lower four positions of the samples. This results in a bit structure that can be efficiently implemented in APA. The bit structure thus derived is shown in Fig. 7 for computing the fifth DCT coefficient. The bit patterns for each DCT coefficient after re-arrangement is shown in Fig. 8.

Another main advantage of the this optimization is the reuse of the summation values previously computed. Table II shows the number of cycles involved in computation for each bit resolution after rearrangement of the structure. The number of cycles without the rearrangement would be more than twice the value with re-arrangement.

One of the main approaches for reducing computations is by detecting noncoded block after motion estimation. References [10] and [11] exclusively use early mechanism for computing power efficient DCT. Pyramidal DCT method can be coupled with this method of conditional computation of DCT to get significant reduction in the average MIPS for highly correlated video sequences.

Row–column transpose in pyramidal approach can be completely avoided. This is because the results of 1-D DCT are arranged in single column and the values for computing DCT in the 2-Ddimension are placed at an offset of eight.The pyramidal computation in the first dimension are performed at 4,2,1 shifts and in the second dimension they are performed at 32,16,8 shifts. For performing DCT in second dimension the rows have to be rearranged to match the bit reordering requirement in second dimension. This arrangement is shown in Fig. 9. The transpose operation is inherently performed using the selection pattern while performing compare operations. The resulting distortion for the same bit resolution is depicted in Fig. 10; whereas distortion values for DCT computed at different resolutions are provided in Table III.

This method of performing DCT can further reduce average MIPS by varying bit resolution for computing DCT. To avoid severe detoriation in the reconstructed image quality, bit resolution can varied depending on the quantization. Since the quantization introduces distortion, the additional distortion introduced by varying bit resolution can be made transparent by computing bit resolution of DCT as a function of quantizer value. In the current implementation, SAD thresholds for detecting not coded block were adaptively computed depending on number of not coded blocks that were previously undetected. Table IV summarizes the results that were observed using this approach.

Data presented in the Table IV is consistent with the observation that for sequences with high correlation like Claire and Mother and Daughter, most of the macroblocks are skipped and, hence, the quantizer remains lower because of which bit resolution remains higher. In high motion sequences like Foreman, there is significant amount of data in the residue and fewer skipped blocks. Quantizer value remains high and, hence, lower average bit resolution. Note that in implementation the bit resolutions are computed at frame level and not macro block level. The results would be much better when implemented at macro block level.

The resultant DCT values can be analyzed at varying bit resolution to optimize quantization operation. In cases where the quantizer is the power of two the quantization process can be

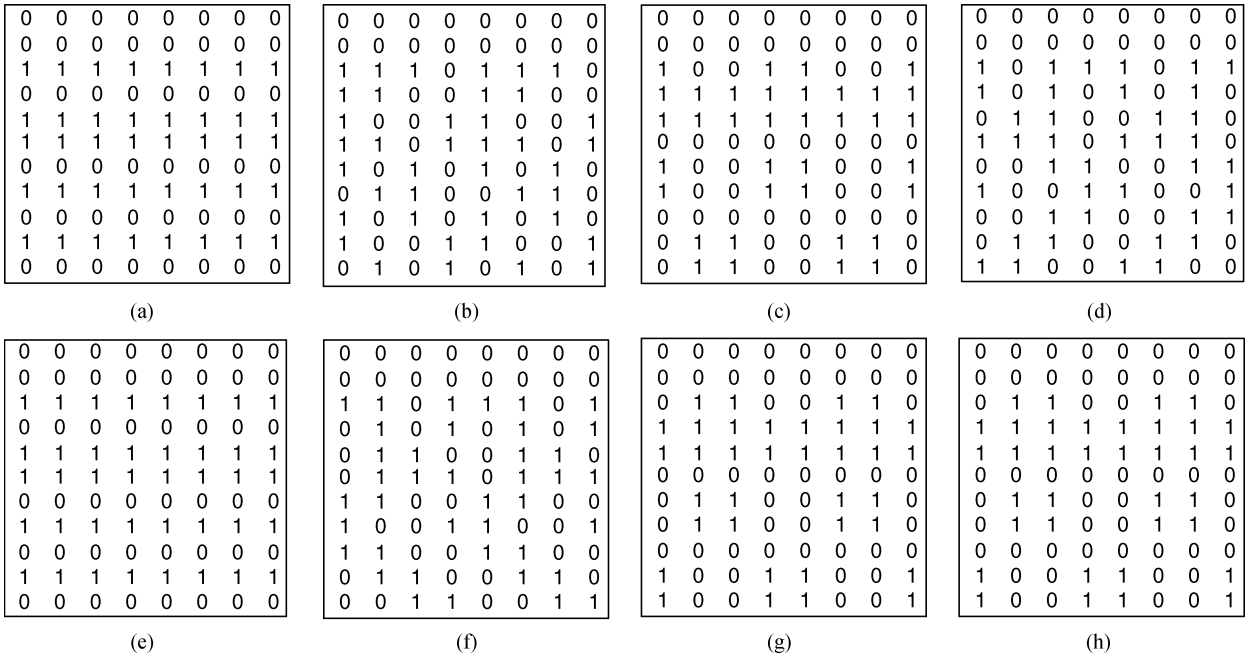

Fig. 8. Bit patterns in rearranged matrix with each coefficient represented in column.

TABLE II
CYCLES FOR COMPUTING DCT AT EACH BIT RESOLUTION

Bit Resolution	Cycles for first Dimension	Cycles for second Dimension
6	1913	2313
7	2197	2643
8	2393	2878
9	2489	2994
10	2611	3158
11	2787	3158

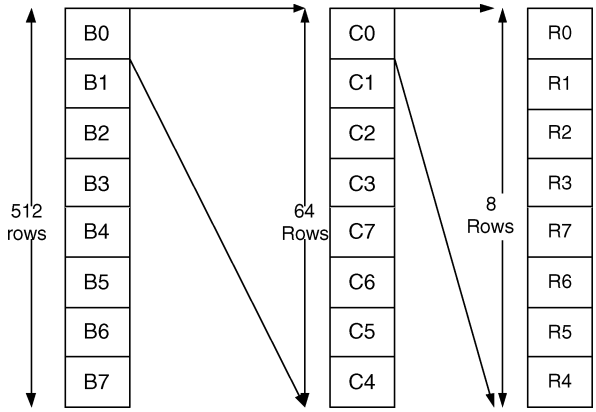

Fig. 9. Data setup for 2-D DCT.

completely avoided. For other cases quantization can be performed at improved resolution after compensating for bits at powers of two. For example, if the quantization value is 18, the quantization can be performed for 18/16 as 4 bits can be compensated at bit positions itself.

C. Distributed Arithmetic Using Look-Up Table (LUT)

As the resolution of video sequence increases, it is observed the average number of SAD calculations per macroblock also reduce. It means at higher resolutions DCT will remain a bottle neck for video coding. So it would be useful to explore a possibility of modifying the architecture for speeding the computation of the DCT. The main drawback in the associative processor is lack of multiplier unit.

The current technique performs distributed arithmetic on bit slices of the input data and not on DCT basis as performed in pyramidal approach. The main advantage of this technique is that the number of cycles required to compute inner product depends on the number of bits in the input data. To demonstrate this idea consider (1) which gives each DCT coefficient y_i for inputs x_k. word. If x_k is represented in the 2's complement as

$$x_k = -c_{k0} + \sum_{i=1}^{t} 2^{-i} c_{ki} \tag{7}$$

by substituting this value and evaluating for y_i, the equation can be simplified as

$$y_i = \left(\sum_{k=1}^{8} -c_{k0} + \sum_{i=1}^{t} 2^{-i} \sum_{k=1}^{8} c_{kt} \right) A_k. \tag{8}$$

In this operation the values of x_k are normalized to one. The values of each summation operation depends on the nth bit of x. This summation can be precomputed and stored in ROM. During computation of y_k, values from precomputed buffer can be restored and added until all digits of x are used. In this process, the number of values in the ROM depends on number of product terms in the summation operation. For N values, the number of precomputed values would be 2^N. Stanley A. White

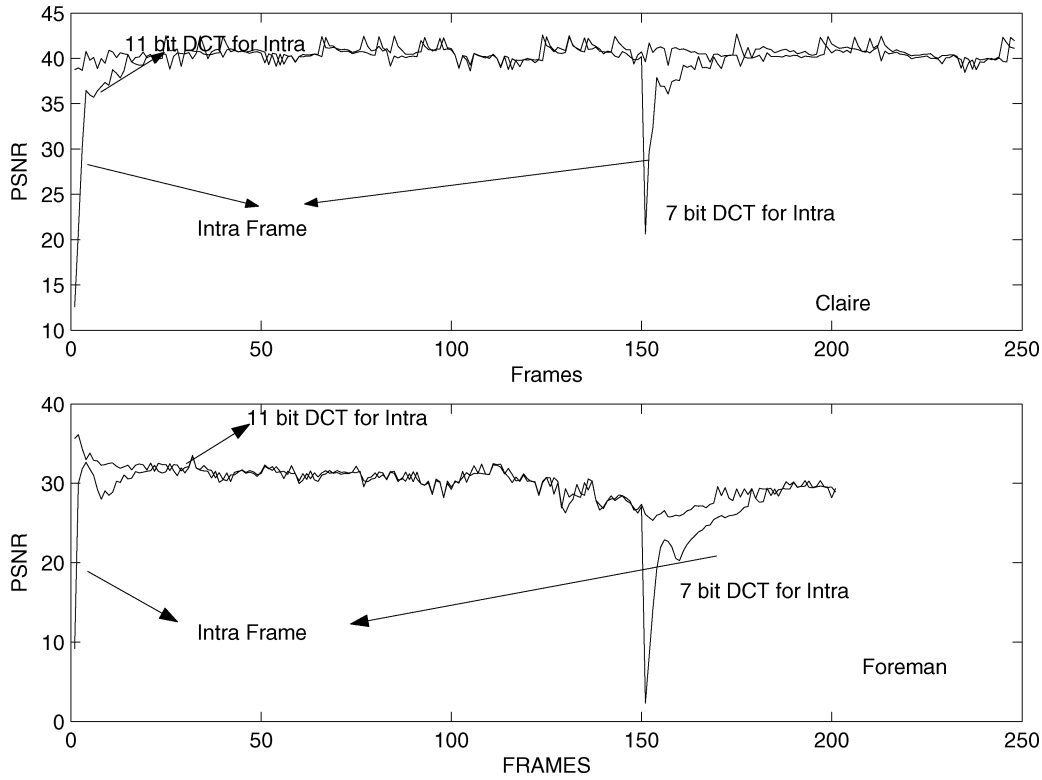

Fig. 10. PSNR plot for same bit resolution in inter/intra-coding. (1) Claire. (2) Foreman.

TABLE III
PSNR WITH DIFFERENT DCT RESOLUTION FOR INTER MBS

Video Sequence	Inter Resolution	PSNR Y Data	PSNR Cb Data	PSNR Cr Data
Claire	6	38.29	39.48	40.97
	7	39.53	40.89	42.37
	8	40.39	41.34	42.92
	9	40.68	41.51	43.10
	10	40.73	41.53	43.12
	11	40.73	41.53	43.12
Foreman	6	27.75	35.89	35.96
	7	25.61	36.12	36.40
	8	29.75	36.05	36.47
	9	30.29	36.44	36.94
	10	30.54	36.81	37.41
	11	30.60	36.89	37.52
salesman	6	34.41	39.96	40.22
	7	35.58	40.27	40.54
	8	36.43	40.57	40.70
	9	36.62	40.61	40.84
	10	36.98	40.83	41.03
	11	37.07	40.90	41.14

in [12] describes various methods and tradeoffs that are possible to reduce the ROM size.

Block diagram of the proposed architecture is shown in Fig. 11. The individual bit slices from data are extracted using the mask/pattern instructions. The resulting tag register is used as index to the LUT that contains precomputed table for all the combinations of the values in the tag. For this operation the tag register is divided into slices of 8 bits each. On a 512 array processors, tag is divided in 64 8-bit partitions, which are used to access LUTs. The value from the LUT is then added to the shifted version of the accumulator for each DCT coefficient. This operation takes place in the array of accumulators shown in Fig. 12.

Row–column transpose is performed by a switching mechanism involved in moving the contents of the accumulators to the processor array. A switching network in the return path rearranges the bits to perform this row–column transpose. This approach of computing DCT in the associative array speeds up the computation of DCT. Simulations show that such a mechanism would be able to compute DCT in less than 120 cycles.

The DCT computation on this enhanced architecture (APA with LUT) happens in pipeline stages. The pipeline involves extracting individual bit slices from the input array, memory access, memory read followed by summation with shift from previously accumulated value.

This technique of performing DCT can however be expensive in terms of size and memory required. The amount of complexity involved in additional control and increase in number of instructions for each operation can be complicated. Further investigation of this technique is required for switching between DCT and IDCT computation. The switching mechanism for transpose in this mechanism is only for 8×8 matrix and this restricts the flexibility in the implementing other 2-D transforms.

TABLE IV
PSNR FOR SEQUENCES WITH VARIABLE INTER RESOLUTION WITH ADAPTIVE THRESHOLDING

Video Sequence		Detected Not Coded MBs	Undetected Not Coded MBs	Total MBs	AVG BIT RESOL	PSNR Y	PSNR Cb	PSNR Cr
Claire	Reference	0	15398	24354	12	41.53	41.62	43.61
	Adap Algo	18646	145	24354	9	40.05	41.40	42.83
Foreman	Reference	0	5653	19701	12	30.40	36.75	37.43
	Adapt Algo	10391	1256	19701	6.9	27.05	37.40	38.83
Mother And Daughter	Reference	0	21413	47223	12	35.72	39.72	39.91
	Adapt Algo	37526	203	47223	8.3	33.41	39.62	39.55

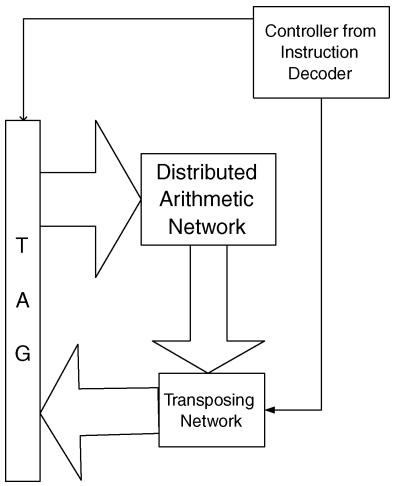

Fig. 11. Block diagram of distributed arithmetic in APA.

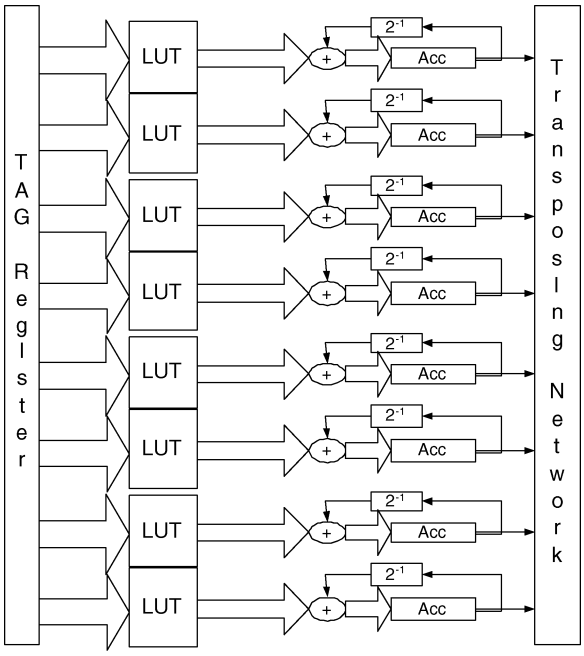

Fig. 12. LUT approach for performing distributed arithmetic.

D. Discussion

Three methods proposed for implementing a 2-D transform are based on three different approaches and can be useful for different applications. In the multiple DCT computation method, the effective number of cycles for computing DCT is very less. This approach complicates the implementation and control as 32 IDCT/DCTs are computed in parallel. With reduced resolution in multiplication 64 DCTs can be computed in parallel. With sufficient memory and added delay, selective DCT can be used implemented. Assuming only 60% of the video data to be coded, the effective number of cycles would be even lesser. This approach can be useful in streaming applications and in implementing transcoders and transraters where the timing constraint is relaxed. The main drawback however remains is the increase in traffic. Video coding using the multiple-DCT approach happens in stages. The data in the array has to flushed at each stage before proceeding to the next stage resulting in increase in data traffic.

TABLE V
DATA TRAFFIC IN VIDEO CODING

Video Block	Traffic in APA (in Million)	Traffic in RISC (in Million)
Motion Estimation	1.23	2.53
Half-pel ME	0	1.7
DCT/IDCT	1.14	2.28
Quant/InvQuant	0	1.14

The second approach takes nearly 750 cycles for computing each block of DCT, its effectiveness depends on the amount of correlation across frames. The main advantage of this method is the reduction in the amount of traffic. The data traffic involved in DCT-quantization-inverse quantization and IDCT is completely removed. Even though multiple DCT method does not consume additional cycles for this data movement(as it is performed in parallel), the power involved in this transfer is completely reduced in this technique. This feature makes this approach extremely power efficient for real-time video coding, particularly in embedded mobile devices.

Finally, the LUT approach combines the advantages of previous two techniques. For video at high resolution DCT would remain a bottle neck. This approach overcomes this bottle neck by performing DCT in 120 cycles and IDCT in 170 cycles. However the increase in size and additional power consumption in this technique is yet to be investigated.

V. ADVANTAGES OF APA OVER RISC/DSP BASED ARCHITECTURES

The associative processor provides a suitable platform for real-time video coding. The region machine works in parallel with the execution cycles, thus overcoming the delay involved in high data rates of video. The associative memory can be viewed as large number of registers which can reduce the

TABLE VI
PERFORMANCE OF RISC, DSP, AND APA FOR MPEG4 VIDEO

Video Block	RISC (Mil cyc/sec)	DSP (Mil cyc/sec)	APA(multiple DCT) mil cyc/sec	APA(LUT based) (Mil cyc/sec)
Motion Estimation	125	15.5	4.75	4.75
DCT/IDCT	11	7	2.9	1.2
Quant/InvQuant	2.5	1.14	0.6	0.6
Total (incl misc)	150	39	15	9

amount of traffic in traditional RISC based architectures. Direct result of reduction in traffic is the reduction in power associated with memory access. Most of the data in associative processors is reused in each of the computational stages. Table V shows the amount of data traffic that is reduced by using APA when compared to RISC and a dual-mac DSP. Comparison of performance of APA with that of RISC and DSP in Table VI is based on the assumption the motion estimation is performed on each macroblock until the final stage of three step search and that DCT/IDCT were performed subsequently. The overall performance of the architecture is implementation dependent.

VI. FUTURE WORK

There is more work on instruction coding. As the requirements of embedded processing increase, it is important that associative processors must be used to process other media such as audio (MP3 and AAC). The performance of associative processors for audio coding is yet to be determined. As the audio coding standards like AAC use overlapped transforms like MDCT, APA can be a good platform for audio coding as well.

Associative processors are clocked at 100 MHz which might limit its performance while coding video at HDTV rates. Another important issue that can be importance is delay involved in context switch when multiple applications are executed in parallel. Since it involves a huge increase in data traffic, the processor array has to partitioned for different applications to avoid the data flush in the array. However the feasibility of such a mechanism is still to be studied.

VII. CONCLUSION

APA provides an ideal platform for coding video in real-time. Motion estimation which is generally a bottle neck in RISC based processors is efficiently implementable in APA. Unlike RISC architectures, each operation is performed at desired bit resolution thereby reducing the number a number of redundant operations.

In this paper, we have presented three different approaches for implementing DCT. The first approach which computes multiple DCTs has a performance better that reported in other architectures. DCT computation by pyramidal approach is computationally expensive, but it provides a mechanism of computing DCT at multiple resolutions.By using the adaptive thresholds technique, the effective number of cycles can be minimized to a large extent. The LUT speeds up APA and can be used to implement any finite-impulse response (FIR) like operation. One of the main factors affecting power consumption is traffic involved in the data. APA reduces the data traffic because the processing array provides enough buffer to avoid redundant traffic caused due to fewer registers.

A trend in recent video coding standards indicates a decrease in the block size for better compression. Implementation of H264 integer transform clearly shows that APA is more efficient processing blocks of smaller sizes.

Associative processors are relatively new and their capabilities are yet to be explore. Associative architectures have shown significant improvement over RISC based architectures both in terms of power and performance. In an embedded platform involving video coding, video remains most intensive application. Though associative processors are very efficient in processing video, their success depends on how flexible they are to overcome concurrency issues involved in an embedded system capable of handling multiple applications.

REFERENCES

[1] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-23, no. 9, pp. 1004–1009, Sep. 1977.

[2] C. Loeffer, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. IEEE Conf. Acoust., Speech Signal Process.*, Glasgow, U.K., May 1989, pp. 988–991.

[3] B. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-32, no. 12, pp. 1004–1009, Dec. 1984.

[4] L. L. Heng and A. Wolfe, "Available parallelism in video applications," in *Proc. 13th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Research Triangle Park, NC, Dec. 1997, pp. 321–329.

[5] A. Y. Shain and R. Adar, "Associative architecture for fast DCT," in *Proc. Int. Conf. Acoustics, Speech Signal Process.*, Seattle, WA, May 1998, pp. 3109–3112.

[6] H. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, and M. Sun, "Low-complexity transform and quantization in h.264/avc," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 598–603, Jul. 2003.

[7] W. Pan, "A fast 2-D DCT algorithm via distributed arithmetic optimization," in *Proc. Int. Conf. Image Process.*, Vancouver, BC, Canada, Sep. 2000, pp. 114–117.

[8] W. Pan, A. Shams, and M. A. Bayoumi, "Neda: A new distributed arithmetic architecture and its application to one dimensional discrete cosine transform," in *Proc. IEEE Int. Workshop Signal Process. Syst.*, Taipei, Taiwan, Oct. 1999, pp. 159–168.

[9] T. Tran, "The binDCT: Fast multiplierless approximation of the DCT," *IEEE Signal Process. Lett.*, vol. 7, no. 6, pp. 141–143, Jun. 2000.

[10] I. Richardson and Y. Zhao, "Adaptive algorithms for variable-complexity video coding," in *Proc. Int. Conf. Image Process.*, Thessaloniki, Greece, Oct. 2001, pp. 457–460.

[11] N. August and D. Ha, "Low power design of DCT and IDCT for low bit rate video codecs," *IEEE Trans. Multimedia*, vol. 6, no. 6, pp. 414–422, Jun. 2004.

[12] S. White, "Applications of distributed arithmetic to digital signal processing:A tutorial review," *IEEE Acoust., Speech, Signal. Mag.*, vol. 3, pp. 4–16, Jul. 1989.

Subhash Balam received the B.S. degree in electrical and electronics communication from Birla Institute of Technology and Science, Pilani, India, in July 2000 and the M.S. degree from the University of Illinois, Chicago, in 2004.

He worked for Motorola India Electronics Private Limited as Software Engineer until August 2002. He then joined the University of Illinois at Chicago for graduate studies where he has worked on computer architectures for video under Dr. Schonfeld's supervision. He is currently with Qualcomm Inc., San Diego, CA, where he works as System Design Engineer.

Dan Schonfeld (M'90–SM'05) was born in Westchester, PA, on June 11, 1964. He received the B.S. degree in Electrical engineering and computer science from the University of California, Berkeley, and the M.S. and Ph.D. degrees in electrical and computer engineering from The Johns Hopkins University, Baltimore, MD, in 1986, 1988, and 1990, respectively.

In August 1990, he joined the Department of Electrical Engineering and Computer Science at the University of Illinois, Chicago, where he is currently an Associate Professor in the Departments of Electrical and Computer Engineering, Computer Science, and Bioengineering, and Co-Director of the Multimedia Communications Laboratory (MCL) and member of the Signal and Image Research Laboratory (SIRL). He has authored over 60 technical papers in various journals and conferences. He has served as consultant and technical standards committee member in the areas of multimedia compression, storage, retrieval, communications, and networks. He has previously served as President of Multimedia Systems Corp. and provided consulting and technical services to various corporations including AOL Time Warner, Chicago Merchantile Exchange, Dell Computer Corporation, Getco Corporation, EarthLink, Fish & Richardson, IBM, Jones Day, Latham & Watkins, Mirror Image Internet, Motorola, Multimedia Systems Corporation, nCUBE, NeoMagic, Nixon & Vanderhye, PrairieComm, Teledyne Systems, Touchtunes Music, Xcelera, and 24/7 Media. His current research interests are in signal, image, and video processing; video communications; video retrieval; video networks; image analysis and computer vision; pattern recognition; and genomic signal processing.

Dr. Shonfeld has served as an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING on nonlinear filtering as well as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING on multidimensional signal processing and multimedia signal processing. He was a member of the organizing committees of the IEEE International Conference on Image Processing and IEEE Workshop on Nonlinear Signal and Image Processing. He was the plenary speaker at the INPT/ASME International Conference on Communications, Signals, and Systems.