Video Compression – An Introduction

The increasing demand to incorporate video data into telecommunications services, the corporate environment, the entertainment industry, and even at home has made digital video technology a necessity. A problem, however, is that still image and digital video data rates are very large, typically in the range of 150Mbits/sec. Data rates of this magnitude would consume a lot of the bandwidth, storage and computing resources in the typical personal computer. For this reason, Video Compression standards have been developed to eliminate picture redundancy, allowing video information to be transmitted and stored in a compact and efficient manner.

1. Video Compression Standards

During the '80s and '90s, Discrete Cosine Transform (DCT) based compression algorithms and international standards were developed to alleviate storage and bandwidth limitations imposed by digital still image and motion video applications.

Today there are three DCT-based standards that are widely used and accepted worldwide:

- JPEG (Joint Photographic Experts Group)
- H.261 (Video codec for audiovisual services)
- MPEG (Motion Picture Experts Group)

Each of these standards is well suited for particular applications: JPEG for still image compression, H.261 for video conferencing, and MPEG for high-quality, multimedia systems.

2. Video Compression Processing Functions

As mentioned earlier, the JPEG, H.261, and MPEG video compression standards are all based on the DCT. In addtion to being DCT-based, many processing functions and compression principles are common to these standards.

The basic compression scheme for all three standards can be summarized as follows: divide the picture into 8x8 blocks, determine relevant picture information, discard redundant or insignificant information, and encode relevant picture information with the least number of bits.

Common functions to all three standards are:

- DCT
- Zig-Zag Scanning
- Quantization
- Entropy Coding
- Motion Estimation

2.1 DCT & Zig-Zag Scanning

The Discrete Cosine Transform is closely related to the Discrete Fourier Transform (FFT) and, as such, allows data to be represented in terms of its frequency components. Similarly, in image processing applications the two dimensional (2D) DCT maps a picture or a picture segment into its 2D frequency components.

For video compression applications, since the variations in the block tend to be low, the great majority of these transformations result in a more compact representation of the block. The block energy is packed into the corresponding lower frequency bins.

The DCT component at coordinates (0,0) is referred to as the DC bin. All other components are referred to as AC bins.

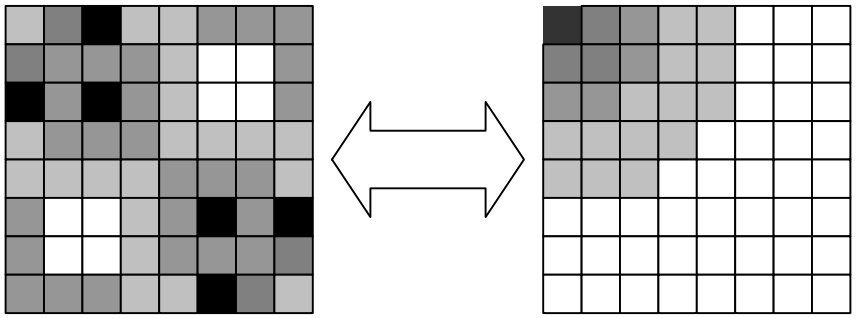

Figure 1. The DCT operation.
(a) Original picture. (b) Corresponding DCT mapping of (a)

Since the mapping is from lower to higher frequencies in the horizontal and vertical directions, zig-zag scanning of the resulting 2D frequency bins clusters packets of picture information from low to high frequencies into a 1D stream of bins.

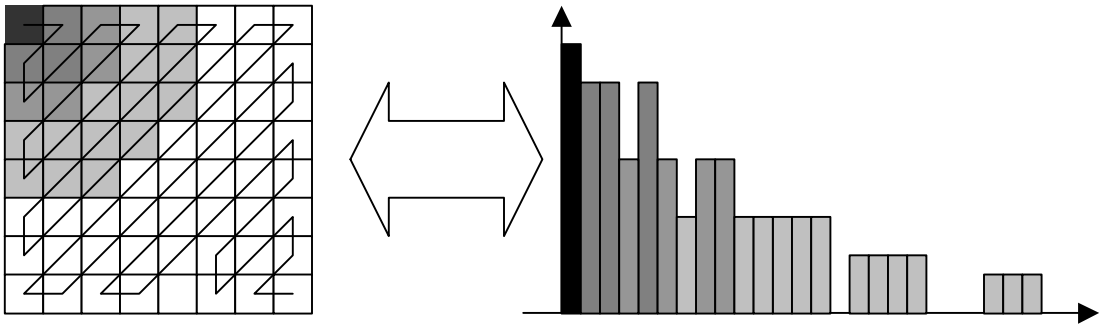

Figure 2. Zig-zag scanning

2.2 Quantization

Quantization is the primary source of data loss in DCT based image compression algorithms. Quantization reduces the amount of information required to represent the frequency bins by converting amplitudes that fall in certain ranges to one in a set of quantization levels. For simplicity, all the standard image compression algorithms use linear quantization where the step size quantization levels are constant.

Quantization in the frequency domain has many advantages over directly quantizing the pixel values. Quantization of the pixel values results in a visual artifact called "contour" distortion where small changes in amplitude in a gradient area cause step-sized changes in the reconstructed amplitude. Except for the DC bin, quantization error for each of the frequency bins average out to zero over the 8 x 8 block.

2.3 Entropy Coding

Entropy coding is a loss-less compression scheme based on statistical properties of the picture or the stream of information to be compressed. Although entropy coding is implemented slightly different in each of the standards, the basic “entropy coding” scheme consists of encoding the most frequently occurring patterns with the least number of bits. In, this manner, data can be compressed by an additional factor of 3 or 4.

Entropy coding for video compression applications is a two step process: Zero Run-Length Coding (RLC) and Huffman coding.

RLC data is an intermediate symbolic representation of the quantized bins which utilizes a pair of numbers. The first number represents the number of consecutive zeros while the second number represents the value between zero-run lengths. For instance the RLC code (5,8) represents the sequence (0,0,0,0,0,8) of numbers.

Huffman coding assigns a variable length code to the RLC data, producing variable length bitstream data. This requires Huffman tables which can be pre-computed based on statistical properties of the image (as it is in JPEG) or can be pre-determined if a default table is to be used (as it is in H.261 and MPEG). In either case, the same table is used to decode the bitstream data.

As mentioned above, frequently occurring RLC patterns are coded with the least number of bits. At this point the digital stream, which is a representation of the picture, has no specific boundaries or fixed length. This information can now be stored or appropriately prepared for transmission.

2.4 Motion Estimation

In general, successive pictures in a motion video sequence tend to be highly correlated, that is, the pictures change slightly over a small period of time. This implies that the arithmetic difference between these pictures is small. For this reason, compression ratios for motion video sequences may be increased by encoding the arithmetic difference between two or more successive frames.

In contrast, objects that are in motion increase the arithmetic difference between frames which in turn implies that more bits are required to encode the sequence. To address this issue, motion estimation is utilized to determine the displacement of an object

Motion estimation is the process by which elements in a picture are best correlated to elements in other pictures (ahead or behind) by the estimated amount of motion. The amount of motion is encapsulated in the motion vector. Forward motion vectors refer to correlation with previous pictures. Backward motion vectors refer to correlation with future pictures.

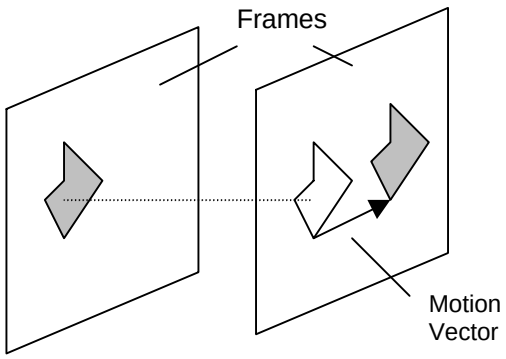

Figure 3. Motion Estimation.

An efficient motion estimation algorithm increases frame correlation, which in turn minimizes pixel arithmetic difference. Resulting in not only higher compression ratios but also in higher quality decoded video sequences. Motion estimation is an extremely computationally intensive operation difficult to implement in real-time. For this reason, a variety of motion estimation algorithms have been implemented by the industry.

3. The JPEG Compression Algorithm

The JPEG algorithm was designed to efficiently compress continuous-tone still images. In addition to its use in still image compression, JPEG has also been adapted for use with motion video sequences. This adaptation (commonly called motion-JPEG) uses the JPEG algorithm to individually compress each frame in a motion video sequence.

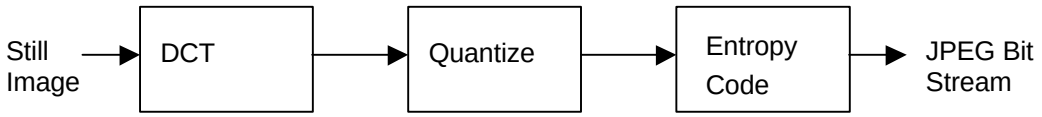

Figure 4. JPEG Encoding

Each color component of a still image is treated as a separate gray-scale picture by JPEG. Although JPEG allows any color component separation, images are usually separated into Red, Green, Blue (RGB) or Luminance (Y), with Blue and Red color differences (U=B-Y, V=R-Y).

Separation into YUV color components allows the algorithm to take advantage of the human eye's lower sensitivity to color information. The U and V color components are commonly recorded at a lower bandwidth and sub-sampled to one-half in the horizontal dimension (called 4:2:2), or one-half in both the horizontal and vertical (called 4:2:0).

JPEG partitions each color component picture into 8x8 pixel blocks of image samples. An 8x8 DCT is applied to each block. For quantization, JPEG uses quantization matrices. JPEG allows a different quantization matrix to be specified for each color component.

Using quantization matrices allow each frequency bin to be quantized to a different step size. Generally the lower frequency components are quantized to a small step size and the high frequency components to a large step size. This takes advantage of the fact that the human eye is less sensitive to high frequency visual noise, but is more sensitive to lower frequency noise, manifesting itself in blocking artifacts.

Modification of the quantization matrices is the primary method for controlling the quality and compression ratio in JPEG. Although the quantization step size for any one of the frequency components can be modified individually, a more common technique is to scale all the elements of the matrices together.

The final stage of compression is the zig-zag scanning and entropy coding. Although JPEG allows for two different types of entropy coders, nearly all JPEG implementations use the Huffman coding option. The JPEG standard allows for the use of user-definable Huffman coding tables. To decompress a JPEG image each operation is performed in reverse order.

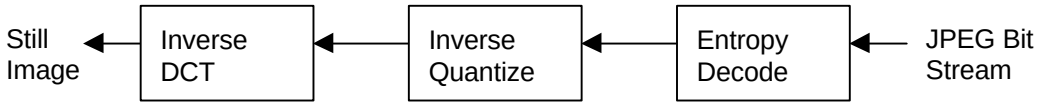

Figure 5. JPEG Decoding

4. The H.261 Compression Algorithm

Video conferencing and video telephony are the intended applications for the H.261 compression algorithm. For these applications, representation of limited motion video (taking heads) is a key component.

To allow for low-cost implementations, H.261 fixes many of the system parameters. Only the YUV color component separation with the 4:2:0 sampling ratio is allowed by the standard. In addition, H.261 allows for only two frame sizes, CIF (352x288) and QCIF (176x144).

As with the JPEG standard, each color component picture is partitioned into 8x8 pixel blocks of picture samples. Instead of coding each block separately, H.261 groups 4 Y blocks, 1 U block, and 1 V block together into a unit called a macroblock. The macroblock is the basic unit for compression.

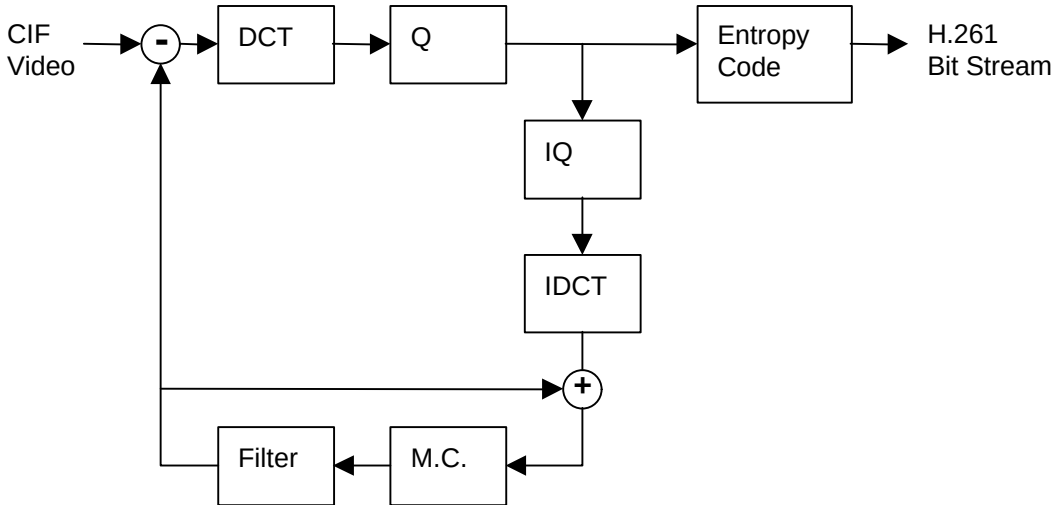

Figure 6. H.261 Encoder

To compress each macroblock, the H.261 standard allows the compressor to select from several compression options. The H.261 standard only specifies the decoding of each of the compression options. The method used to select the options is not standardized. This allows vendors to differentiate their products by providing methods with different cost-quality tradeoffs. A typical method used to compress H.261 is described below.

First, motion estimation is performed on each macroblock. Since objects in the frame may be moving in different directions, each macroblock is allowed to have a different motion vector. The motion vector is used as a displacement vector to fetch a macroblock from the preceding frame to be used as a prediction. Motion estimation in H.261 is only performed relative to the preceding frame, and on full-pixel offsets up to a maximum of +/-15 in the horizontal and vertical directions. To improve the prediction, H.261 allows for an optional loop-filter to be applied to the prediction on a macroblock basis.

Next, a decision must be made to code either the arithmetic difference between the offset prediction macroblock and the current macroblock or to code the current macroblock from scratch. Since the arithmetic difference is usually small, coding the arithmetic difference results in higher compression.

An 8x8 DCT is applied to each block in either the arithmetic difference macroblock or the current macroblock. Instead of quantization matrices, H.261 uses one quantization scale for all frequency bins. Since the DC bin is the most important, it is separately quantized to a fixed 8 bit scale. Adjustment of the quantization scale on a per macroblock basis is the primary method for controlling the quality and compression ratio in H.261.

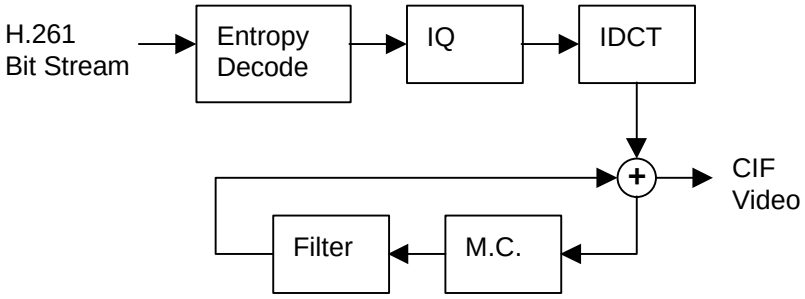

Figure 7. H.261 Decoder

The final stage of compression is the zig-zag scanning, run-length encoding and entropy coding. Unlike JPEG, H.261 specifies fixed Huffman coding tables for entropy coding.

To decompress an H.261 frame inverse operations are performed in reverse order. Motion estimation is not necessary since the motion vectors are embedded in the compressed bitstream. The H.261 de-compressor simply applies the motion vector offset to retrieve the prediction, if necessary.

5. The MPEG Compression Algorithm

MPEG compression algorithms were developed to address the need for higher quality pictures and increased system flexibility, which are required by multi-media systems. Since it was developed later, MPEG was able to leverage the efforts behind the development of both the JPEG and H.261 algorithms.

As with H.261, only the YUV color component separation with the 4:2:0 sampling ratio is allowed by the MPEG standard. Unlike H.261, the frame size is not fixed although a 352x240 frame size is typically used. MPEG adopted the macroblock of H.261 (4 Y blocks, 1 U block, and 1 V block) as the basic unit for compression. To compress each macroblock, the MPEG standard allows the compressor to select from several compression options.

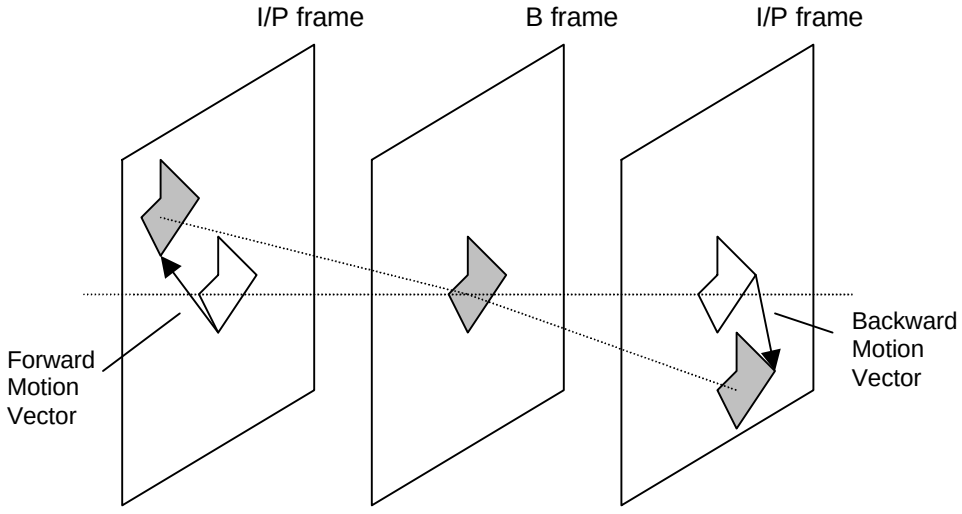

Figure 8.: MPEG Motion Compensation

There are many more options available under the MPEG standard than under H.261. As with H.26l, the MPEG standard only specifies the decoding of each of the compression options. The method used to select the options is not standardized, allowing vendors to differentiate their products by providing methods with different cost-quality trade-offs. A typical method used to compress MPEG is described below.

First, motion estimation is performed on each macroblock. In addition to motion estimation from just the preceding frame, MPEG allows for prediction from frames in the past or future or a combination of a past and future frame (with restrictions).

Since objects in the frame may not be moving steadily from frame to frame, each macroblock is allowed to have up to two motion vectors (one relative to a past frame and another relative to a future frame). Note that to allow for predictions from future frames, the extra frames must be buffered and the sequence coded out-of-order.

Motion estimation is also allowed over a greater range (up to +/- 1023) and with half-pixel resolution. The loop-filter of H.261 is not included in MPEG because the half-pixel resolution motion vectors serve the same purpose.

Next, a four-way decision must be made. MPEG allows the prediction formed from the arithmetic difference between the current macroblock and an offset macroblock from a past frame, future frame, an average between past and future frame, to be coded; or to code the current macroblock from scratch. A different decision can be made for each macroblock subject to the restrictions that follow.

Key frames (called Intra or I frames) which do not allow any predicted macroblocks are coded periodically to allow for random access into the video stream. Forward predicted frames (called P frames) allow macroblocks predicted from past P frames or I frames or macroblocks coded from scratch. I frames and P frames are used as past and future frames for Bi-directional predicted frames (called B frames). B frames allow for all four types of macroblocks.

An 8x8 DCT is applied to each block in either the arithmetic difference or current macroblock. MPEG uses both matrices (like JPEG) and a scale factor (like H.261) for quantization. Since the DC bin is the most important, it is quantized to a fixed 8 bit scale.

Since the visual effects of frequency bin quantization are different for predicted and current blocks, MPEG allows for two matrices (one for each type). Typically, the matrices are set once for a picture sequence and the quantization scale is adjusted to control the compression ratio.

The final stage of compression is the zig-zag scanning, run-length encoding and entropy coding. Like H.261, MPEG specifies fixed Huffman coding tables for entropy coding.

To decompress an MPEG frame each operation is performed in reverse except for motion estimation. Since the motion vectors and the decision are embedded in the compressed bit-stream, the MPEG de-compressor just needs to apply the motion vector offsets to retrieve the prediction from the past and/or future frames if necessary.

6. VIDEO*FLOW*™ *Architecture*

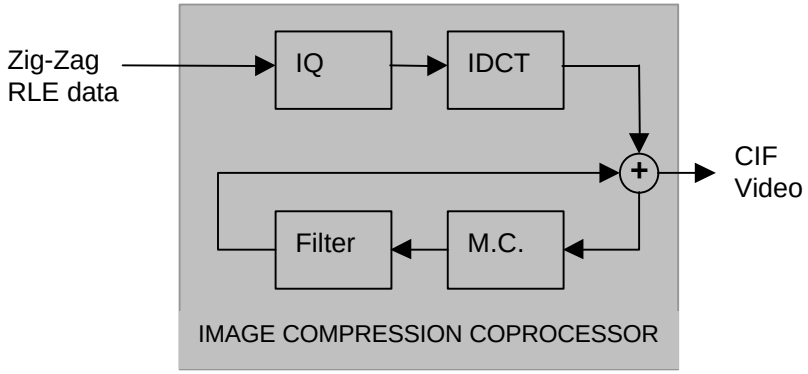

Figure 9. ICC Decoding Architecture

The VIDEO*FLOW*™ architecture, developed by Array Microsystems, exploits the fact that the MPEG-1, H.261 and JPEG compression standards have many algorithmic operations in common. The Image Compression Coprocessor (ICC) performs all common compression operations, including DCT, quantization, and zig-zag processing. The ICC has multiple parallel processing units highly optimized to perform these and other common operations.

The Motion Estimation Coprocessor (MEC) performs motion estimation for video encoding applications. The MEC includes a high-speed block matcher which estimates motion by searching an area in video memory, looking for the closest match to the target b1ock. Controlled by an embedded RISC processor, the MEC can be programmed to perform a broad range of motion estimation algorithms.

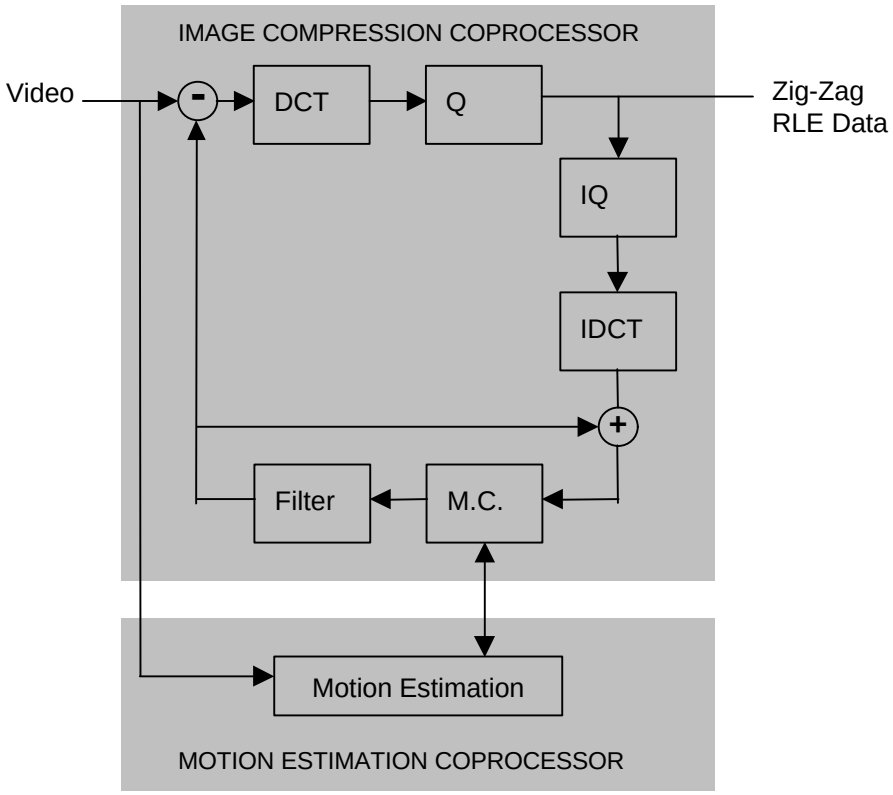

Figure 10. ICC~MEC Encoding Architecture

The VIDEO*FLOW*™ technology is designed to allow the highest possible flexibility while providing ease of development. A large number of techniques for increasing picture quality, for a given compression level, are easily implemented by the chip. For example, adaptive quantization and clipping control are two techniques included in the VIDEO*FLOW*™ chipset that have been found to he the useful in increasing compression and improving image quality. A high degree of flexibility is also provided in implementing motion estimation algorithms.

Software support tools, including a graphical programming environment and chipset simulator, dramatically reduce development complexity and time-to-market. With a strong emphasis on low cost and easy to use development tools, Array Microsystems Inc. enables video compression application development to a broader range of customers.

