

Real Time MPEG-I and MPEG-II Compression with the Alpha Microprocessors

Joseph Chou
Strategic Marketing Manager
Digital Semiconductor

Introduction — What is Visual Computing

Visual Computing is defined in the following four areas:

1. Creating and accessing of 3D graphics images for
 - VRML (inter/intranet)
 - Animation (Film, studio)
 - Business presentation (Sales/marketing)
 - Architecture work through (Training)
 - Learning (Education)
 - MCAD (Productivity)
 - Games (Entertainment)
2. Creation and playback of DVD/DC ROM motion video for
 - Web A/V streaming (Inter/intranet)
 - Business presentation (Sales/marketing)
 - Movies and games (Entertainment)
 - Architecture work through (Training)
 - Interactive books (Education)
 - Video editing (Production, Pleasure)
3. Enhancement of
 - Communication through video conferencing and 3D whiteboard
 - 3D games with streaming motion video background
4. Create realistic photographic through
 - Image processing

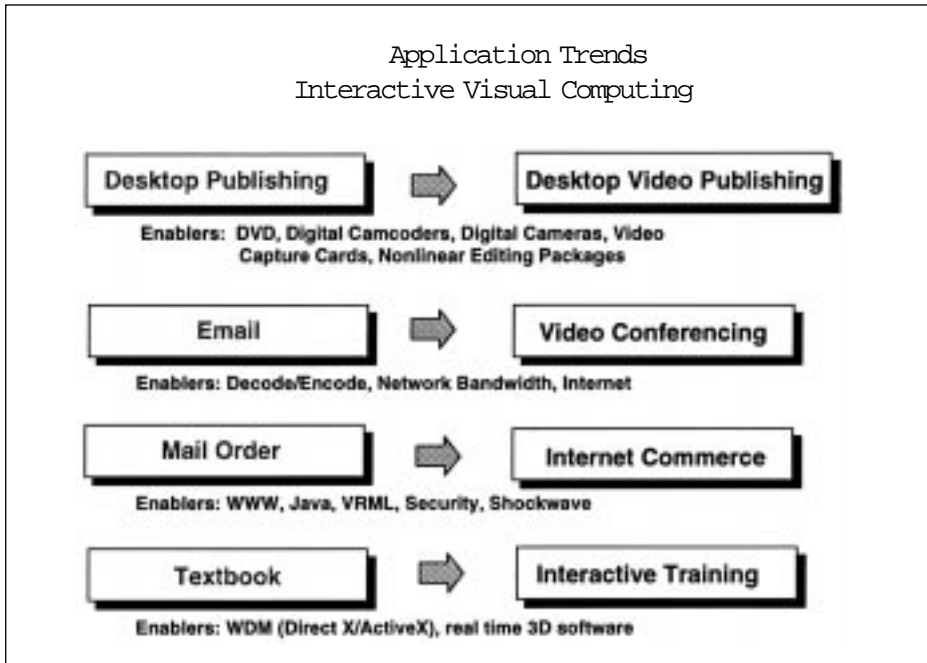
The need for visual applications continues to drive PC performance innovation. High-end visual applications such as 3-D animation, image processing, video conferencing, and video editing are moving to the Windows NT market because of continuing lower software package prices and increasing PC user demand.

As videocentric applications become more pervasive over the next several years, requirements for the efficient mixing of video and traditional applications will grow. Video and image types will replace text as the standard means of communication, presentation, training, and record keeping.

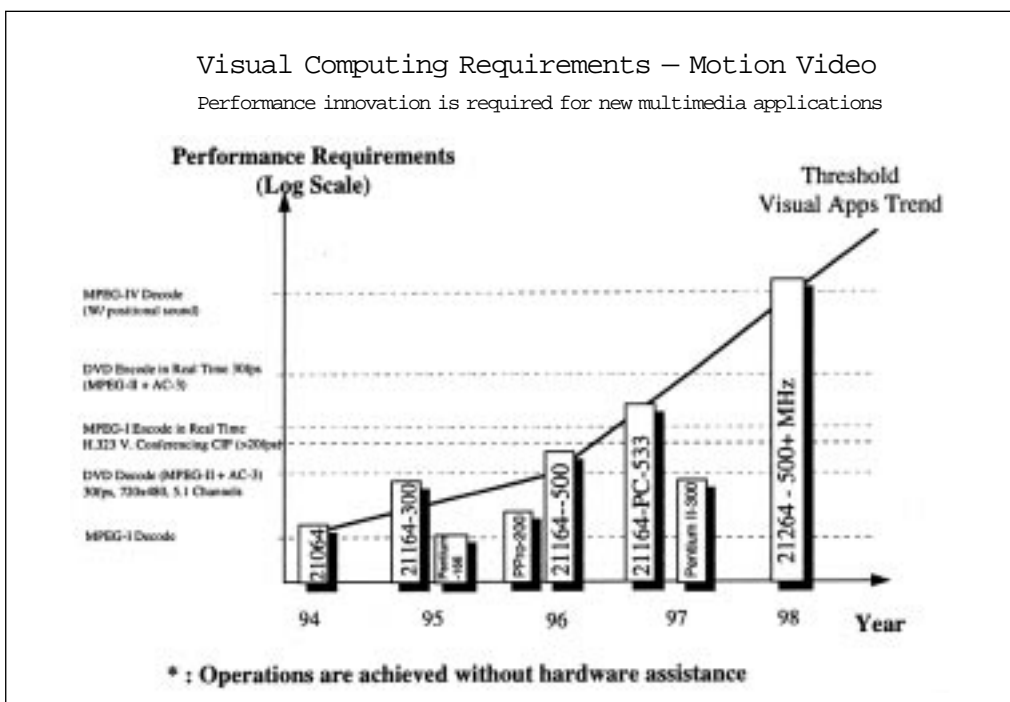
This paper will focus on motion video applications for MPEG-I real time authoring with Alpha 21164PC and MPEG-II real time authoring with Alpha 21264 microprocessors.

Present and future application trends in interactive visual computing can be compared as follows:

Application Trends Interactive Visual Computing



Unlike the past, it is currently a trend for emerging compute-intensive multimedia applications to use host-based implementation with high-performance microprocessors to cross the barriers of threshold applications without hardware assistance or with little hardware assistance until the multimedia technology is mature. A threshold application is defined as requiring a tremendous amount of CPU computing power, and only those CPUs with enough host power crossing the performance requirement barriers can handle this kind of application without hardware assistance. Because there is usually a lag of low-cost hardware assistance solutions, the benefits of using host-based multimedia for emerging multimedia applications are time-to-market and system cost savings. To gain back CPU host power for other applications, the hardware assistance solution is viable in a well-balanced system when the specific application technology is mature and the assistance chips reach the economics of scale production.



The most performance-demanding portion of the video operation is real-time video encode. It is the compute-intensive motion estimate operation that makes real-time 352x240 CIF MPEG-I and 720x480 CCIR.601 MPEG-II video encode threshold applications currently not implementable on today's processors. The goal of video encode is to compress motion video information for storage and communication while maintaining its quality for playback. The performance requirements of doing video encode vary between 4X and 20X of video decode depending on the algorithms used. DVD (MPEG-II) and CD (MPEG-I) compression and decompression are highly asymmetric media operations. Since the video will be played back many times in on low-cost hardware, the computational requirements are heavily skewed towards encoding, leaving the decode requirements relatively simple. Motion Video Instructions (MVI) on Alpha 21164PC provide users with the first system capable of real-time CD/ROM authoring as well as DVD playback while Alpha 21264 provides real time DVD authoring capability. Some limited number of microprocessors today can demonstrate MPEG-I and MPEG-II decode but none of them are capable of doing real-time MPEG-I encode or DVD encode without hardware assistance.

Alpha's architecture today is the leader in multimedia applications including 3-D animation graphics, MPEG-I and -II video playback, speech recognition, music synthesis, image processing, video conferencing, and more. Adding MVI technology to the Alpha architecture further extends its capability to compress high-quality real-time motion video CD and DVD data and high-quality real-time video conferencing. MVI technology is a significant enhancement to the Alpha 64-bit architecture since it was introduced in 1992. Alpha microprocessors enabled with MVI technology will deliver superior performance to emerging visual computing applications that demand more CPU performance.

MVI technology will enable host-based video encode in Windows NT PC systems using Alpha microprocessors. It supports Single Instruction, Multiple Data (SIMD) type operations and provides Pack and Unpack instructions for pixel merge and de-merge parallel operations. It also provides clamping instructions for saturation control in pixel operations. The most important instruction in MVI is the pixel error instruction. The pixel error instruction replaces about 23 of x86 instructions required for motion estimate. In a traditional host-based video encode algorithm, the MPEG-II motion estimate calculation occupies more than 61% of the 21164-400 CPU resources, and non-Alpha microprocessors have a worse percentage rate. It is this situation that prevents most CPUs from doing a quality host-based video encode. The MVI is designed to solve this problem. With pixel error instruction, the CPU utilization on motion estimate can be reduced to 15% from 61%. This will free up 46% of CPU host power for other concurrent applications, such as AC-3 encode and multitasking. The pixel error instruction provides three-level SIMD type operations. It can manipulate eight pixel operations at same time by calculating the sum of absolute difference value between referenced macroblock and searched macroblock. The grand totals of all pixel errors between the two macroblocks are compared in a macroblock search. When the most similar macroblock is found with smallest grand sum of pixel errors, the associated motion vector will be stored in coded video stream and the pixel error will be further encoded by DCT and Huffman coding. The compression time will be approximately 20X faster when using the revolutionary pixel error instruction for the predicted pictures (P-pictures) and bidirectional pictures (B-pictures). Since more than 85%-90% of the video pictures consist of P- and B-pictures, MVI becomes critical to host-based motion video applications.

The key attributes of motion video compression applications are:

- Small integer data type (8 bit video pixels)
- Intensive summation operations of absolute values of parallel subtractions
- Highly parallel operations

The highlights of MVI technology are:

- SIMD (Single Instruction, Multiple Data)
- 32 MVI 64-bit registers (shared with integer registers)

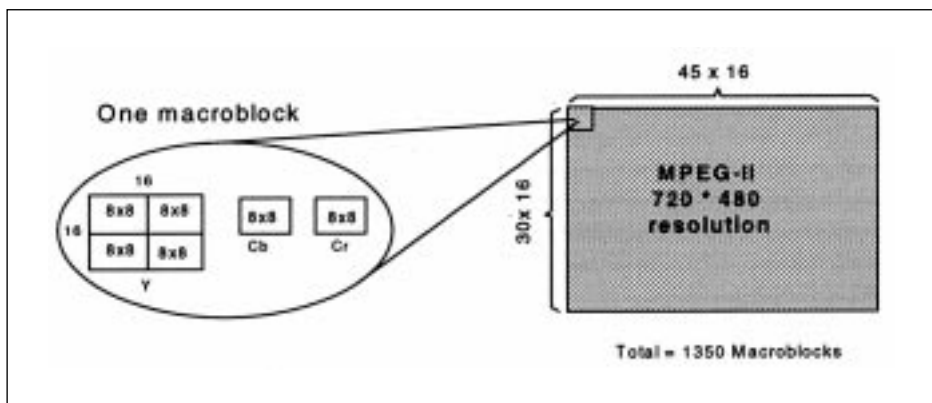
- Concurrent MVI and FPU operations
- 13 new instructions

The basis of MVI technology is SIMD. SIMD allows multiple arithmetic operations to be processed in a single instruction in a single clock. This technology reduces a huge number of motion estimate calculations and enables CPU host based MPEG-I and MPEG-II compressions. MVI technology is integrated into Alpha architecture in such a way as to maintain backward compatibility with Windows NT environments. MVI instruction details are explained in the Sample Problem and Solution section.

Picture Basics and Compression Techniques

Macroblocks and Blocks

A macroblock consists of a 16-pixel by 16-line set of Y luminance components and the corresponding 8-pixel by 8-line of Cb and Cr chrominance components. An MPEG-II 720x480 picture contains 1350 macroblocks as shown in the following figure. A block is an 8-pixel by 8-line set of luminance components or a chrominance component. A macroblock consists of four luminance blocks and two chrominance blocks. An 8x8 block is the unit that is used for Discrete Cosine Transform (DCT) in I-pictures (intra pictures).



Picture Types

There are three types of pictures: intra pictures, predicted pictures, and bidirectional pictures.

Intra Pictures (I-pictures)

I-pictures are coded using only the information in a single video frame. The DCT transform is applied to each of the 8x8 pixel blocks individually to transform the spatial domain to the frequency domain to reduce the amount of data. The transform exploits the spatial redundancy of the pixels by converting them to a set of independent coefficients. After DCT transformation, the lowest frequencies tend to have larger coefficients and the highest frequency tends to have a zero coefficient. The high-frequency coefficients can afford to be dropped because the human eye lacks the ability to detect high-frequency change. Because the DCT is unitary, the maximum value of the DCT coefficient is limited to a factor of 8 times the original value. In addition to the original 8-bit input value, three additional bits are needed to hold the precision of intermediate results.

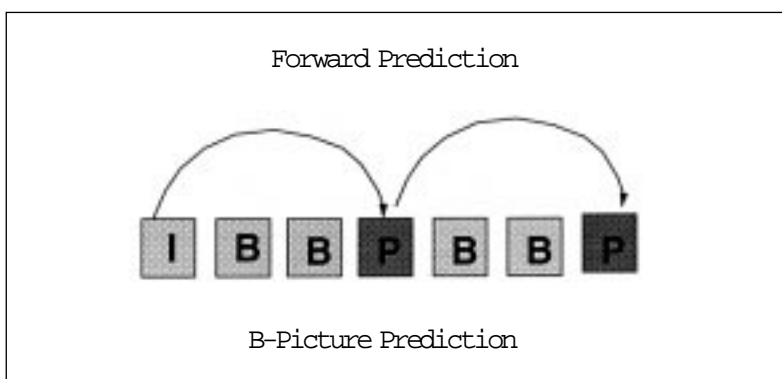
The resultant DCT coefficients are quantized to form a quantization matrix. This process reduces the amount of data by allowing the high-energy, low-frequency coefficients to be coded more accurately with a great number of bits and the low-energy, high-frequency coefficients are coded with few or zero bits. The coefficients in the quantization

matrix are then coded by Run-Length and Huffman coding. Due to the low-pass characteristic of DCT and quantization, many zeros are generated. Run-Length then scans the 2-D quantization matrix in a zigzag pattern to create a 1-D sequence. Each nonzero coefficient has a pair of pointers indicating the coefficient value and the number of zeros between itself and the previous nonzero coefficients. Each pair of pointers is assigned a variable-length code from a Huffman lookup table. The most likely combinations get a code with few bits and the less likely ones get codes with more bits, thus further reducing the total amount of data. Due to the spatial limit, the I-picture compression rate is moderate compared to P- and B-pictures. I-pictures serve as reference pictures for random access in the picture stream and are used to generate P- and B-pictures.

Redundancy	Encode Methods
Spatial	DCT
Temporal	Motion Compensation Prediction & Interpolation
Coding	Run length and Huffman coding

Predicted Pictures (P-pictures) and Bidirectional Pictures (B-pictures)

The P- and B-pictures are where MPEG derives its maximum compression efficiency from pixel error search of Motion Compensation (MC). MC is heavily used to exploit temporal redundancy for P- and B-pictures. Since pictures are usually closely related, often a picture can be retained as a translation of a prior picture. The motion estimate technique reduces a huge amount of data and achieves maximum compression efficiency. P-pictures are coded through forward prediction with respect to the nearest previous I- or P-picture. A search is conducted in the I- or P-picture to find the macroblock that best matches the macroblock under consideration in the P-picture. The difference between the two macroblocks is the pixel error (or prediction error). The offset of the macroblock under consideration is the motion vector. This error can be coded using DCT and quantization techniques. The compression efficiency and the quality of the reconstructed motion video depend on the accuracy of motion estimate. P-pictures use motion compensation; therefore, the compression ratio is higher than for I-picture. P-pictures serve as reference pictures for P- and B-pictures and can propagate coding errors.



$$\text{Pixel error} = E(dx, dy) = \sum_{i=0}^{15} \sum_{j=0}^{15} | r(x_0 + dx + i, y_0 + dy + j) - n(x_0 + i, y_0 + j) | \quad (1)$$

$$\text{Prediction error} = E(x_0, y_0) = \text{Min } E(dx, dy) \quad (2)$$

$$\text{Motion vector} = \text{the } (dx, dy) \text{ of } E(x_0, y_0) \quad (3)$$

Where :

- x_0 = X position of the macroblock
- y_0 = Y position of the macroblock
- dx = x offset from X position
- dy = y offset from Y position
- r = pixel from the reference frame
- n = pixel from the new frame

Example: P- and B-pictures Search

This example assumes that the full search window range is +/- 8 pixels for a MPEG-II picture. The overall performance requirement is calculated as follows:

Total subtractions and loads required for a macroblock search:

$$(16 \times 16) \text{ pixels} \times 16 \times 16 \text{ dx, dy boundary} \times 3 = 192\text{K (without MMX)} \quad \textcircled{1}$$

$$(2 \times 16) \text{ pixels} \times 16 \times 16 \text{ dx, dy boundary} \times 3 = 24\text{K (with MMX)} \quad \textcircled{1'}$$

Total absolute value operations for a macroblock search:

$$(16 \times 16) \times 16 \times 16 = 64\text{K (with or without MMX)} \quad \textcircled{2}$$

Total calculations for summations:

$$(16 \times 16 - 1) \times 16 \times 16 = 64\text{K (with or without MMX)} \quad \textcircled{3}$$

Thus total calculations for a macroblock (①+②+③ or ①'+②+③):

$$= 192\text{K} + 64\text{K} + 64\text{K} = 320\text{K (without MMX)} \quad \textcircled{4}$$

$$\text{or } = 24\text{K} + 64\text{K} + 64\text{K} = 152\text{K (with MMX)} \quad \textcircled{4'}$$

There are 1350 macroblocks in an MPEG-II 720x480 picture. Out of 1350 macroblocks, 146 macroblocks are on the border and 1204 macroblock blocks are in the inner region. Calculations required for inner region macroblocks:

$$1204 * 320\text{K} = 376\text{M calculations (without MMX)} \quad \textcircled{5}$$

$$\text{or } 1204 * 152\text{K} = 179\text{M calculations (with MMX)} \quad \textcircled{5'}$$

Calculations required for macroblocks on the border:

$$142 * 160\text{K} + 4 * 80\text{K} \cong 23\text{M calculations (without MMX)} \quad \textcircled{6}$$

$$\text{or } 142 * 76\text{K} + 4 * 38\text{K} \cong 11\text{M calculations (with MMX)} \quad \textcircled{6'}$$

Total calculations for an 8-pixel distance search:

$$= 376\text{M} + 23\text{M} = 399 \text{ Million calculations (without MMX)} \quad \textcircled{7}$$

$$\text{or } = 179\text{M} + 11\text{M} = 190 \text{ Million calculations or Clocks (with MMX)} \quad \textcircled{7'}$$

Performance requirement:

$$= 399\text{M} \times 28 = \mathbf{10.9} \text{ Billion instructions per second (with MMX)} \quad \textcircled{8'}$$

$$\text{or } = 190\text{M} \times 28 = \mathbf{5.2} \text{ Billion instructions per second (with MMX)} \quad \textcircled{8}$$

A normal picture sequence requires 28 frames per second for P- and B-pictures. The worst-case performance requirement as shown in the result of equation ⑧' is more than 4 billion instructions per second with MMX technology. This prevents today's PC systems from doing host-based MPEG-II encode, even with MMX technology. The next section explains how to resolve this issue.

Solution: MVI Instructions

Macroblock Search Using MVI Instructions

Using the same macroblock search assumption in the previous section, that is, +/- 8 pixel search windows, the performance requirement of Alpha microprocessors enabled with MVI is calculated as follows:

$$\begin{array}{rcl}
 (2 \times 16) & \times & 16 \times 16 \times 3 = 24\text{K} \quad \textcircled{1} \\
 \text{Macroblock} & \text{Search window} & \text{loads \& SIMD} \\
 \text{pixels} & \text{dx, dy boundary} &
 \end{array}$$

There are 1350 macroblocks in an MPEG-II 720x480 picture. Out of 1350 macroblocks, 146 macroblocks are on the border and 1204 macroblock blocks are in the inner region.

$$\begin{array}{l}
 \text{Calculations required for inner region macroblocks:} \\
 1204 * 24\text{K} = 28.2\text{M calculations} \quad \textcircled{1} \\
 \text{Calculations required for macroblocks on the border:} \\
 142 * 12\text{K} + 4 * 6\text{K} (1.7\text{M calculations} \quad \textcircled{1} \\
 \text{Total calculations for an 8-pixel distance search } (\textcircled{1} + \textcircled{1}): \\
 = 28.2\text{M} + 1.7\text{M} = 29.9 \text{ million calculations} \\
 \text{The total calculations required for 28 P and B frames per second are:} \\
 30\text{M} * 28 = \mathbf{837 \text{ million calculations/sec}} \quad \textcircled{1}
 \end{array}$$

Compared to a Pentium-II MMX enabled processor, Alpha 21264 with MVI technology has a motion estimate performance advantage factor of over 4500%. See the comparison in the following table.

Performance		MVI		Pentium-II 300 MHz		Alpha 21264-500
		W/O MMX	With MMX	W/O MMX	With MMX	With MVI
MC Calculation Requirements for 720*480 @ 30fps		10.9 billions/sec	5.2 billions/sec			837 millions/sec
Motion Estimate Performance	No MHz Adj.	0.48X	1X			6.2X
	With MHz & Architecture Adj.*	0.48X	1X			21X

* : 21264 has four integer units for motion estimate

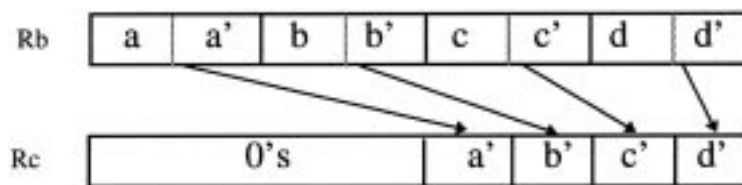
MVI Instruction Set Summary

The instructions in the table are grouped by categories of functions.

Category	Instructions	Description
Motion Compensation	PERR Ra, Rb, Rc	Pixel Error. Absolute value of difference between each of the bytes in Ra and Rb is calculated. The sum of the resulting bytes is written to Rc.
Data Conversion	UNPKBL Rb, Rc UNPKBW Rb, Rc PKLB Rb,Rc PKWB Rb,Rc	Unpack Bytes to Longwords Unpack Bytes to Words Pack Longwords to Bytes Pack Words to Bytes
Data Clamping	MINUB8 Ra,Rb or lit,Rc MINSB8 Ra,Rb or lit,Rc MINUW4 Ra,Rb or lit,Rc MINSW4 Ra,Rb or lit,Rc MAXUB8 Ra,Rb or lit,Rc MAXSB8 Ra,Rb or lit,Rc MAXUW4 Ra,Rb or lit,Rc MAXSW4 Ra,Rb or lit,Rc	Vector Unsigned Byte Minimum Vector Signed Byte Minimum Vector Unsigned Word Minimum Vector Signed Word Minimum Vector Unsigned Byte Maximum Vector Signed Byte Maximum Vector Unsigned Word Maximum Vector Signed Word Maximum

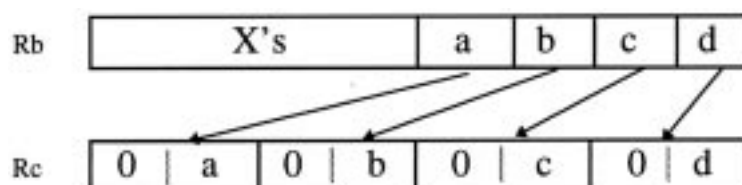
Examples of MVI Instructions

This section will describe briefly three examples of MVI instructions. The following example shows a pack instruction—**PKWB Rb, Rc**. It takes four 16-bit values from Rb and packs them into four 8-bit values in lower four bytes of Rc, performing transactions if one of the 16-bit source values cannot fit into an 8-bit result. The pack instructions are used to restore data after SIMD operations. The other instruction in this group is **PKLB Rb, Rc**. It takes two double words of Rb and packs them into the lower two bytes of Rc, performing transactions if necessary.



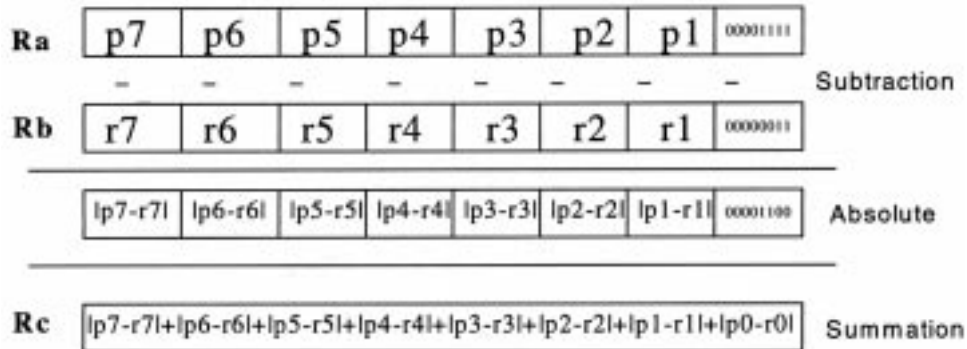
Pack

The unpack instructions perform the opposite operations of the pack instructions. The following example shows an unpack instruction—**UNPKBW Rb, Rc**. It takes four 8-bit values from the lower four components of Rb and unpacks them into four 16-bit values in Rc, performing zero bit insertion to extend 8-bit values to 16-bit values.



Unpack

The following instruction is the core of MVI technology (PERR Ra,Rb,Rc. This instruction performs three levels of SIMD operations in every pipelined clock tick in next generation Alpha: parallel subtractions, parallel absolute value calculations, and parallel summations. It takes eight byte values from Ra and eight byte values from Rb, and performs byte-level subtractions. The result of each byte is manipulated with absolute value operations. The absolute value of each byte is then added together and stored in Rc.



Application Performance Comparison

High-Quality Video Conferencing: H.323 LAN/Internet with MVI

The following example shows the video-conferencing capabilities of current Alpha processors and future MVI-enabled Alpha processors. The example is based on the H.320 standard with 176x144 full-screen QCIF video at 30 frames per second and G.728 LD-CELP audio. The comparison indicates that MVI technology improves video quality, and more important, it reduces CPU utilization to 50% from 100%. The 50% CPU resource saving can be used for other concurrent multitasks in video conferencing, such as 3-D graphics whiteboard, to improve productivity.

MVI	Alpha 21164-400MHz Without MVI	Alpha 21164PC-533MHz With MVI
Encode Process		
Video		
Encode	17%	15%
Motion Estimate	61%*	15% (Full Quality)
Decode	12%	10%
Audio	10%	9%
CPU Utilization	100%	49%

* Compromised quality

DVD Encode Performance Estimate

DVD consists of MPEG and audio data. The high-end specification of the DVD standard is MPEG-II video and AC-3 audio. The video resolution used for comparison is 720x480x30 fps CCIR.601 MP@ML (Main Profile at Main Level) standard. The following example shows huge CPU utilization improvement with MVI. More important, it helps host-based multimedia implementation cross the threshold. The DVD encode requires more than 4X the performance of today's best processor. With MVI technology and next generation Alpha microprocessor implementation, real-time DVD encode will occur without hardware assistance. Motion estimate requires more than 3X of current resources without MVI. With MVI, the requirement is reduced to approximately 25%, which makes CPU utilization under 100%. This utilization rate makes real-time encode possible.

MVI Encode Process	Alpha 21264 MVI disabled	Alpha 21264 MVI enabled
Motion Estimate	~360%	~25%
Picture Coding	16%	16%
DCT/IDCT	8%	8%
Run Length	8%	8%
Huffman Coding	7%	7%
AC-3 Encode	30%	30%
CPU utilization	~429%	~94%

Conclusions

The arrival of Windows NT visual computing drives PC platform innovations. 3-D graphics, image processing, and motion video applications will be focused on the mainstream desktop PC and become pervasive. The Alpha microprocessor has leading performance in visual computing because of its floating-point and integer computing power. Due to its floating-point weakness, an X86 CPU with MMX does not provide performance benefits to 3-D graphics in a mainstream PC system. MMX does not resolve 3-D graphics geometry and set-up bottlenecks that heavily rely on floating-point performance. With software rendering, MMX may help improve 3-D performance in very low-end PC systems (under \$1500) that do not have a 3-D graphics chip. When the 3-D graphics chips become pervasive in 1998, MMX benefits to low-end systems will disappear. Due to its lack of performance on integer operations, an X86 CPU with MMX will provide marginal and compromised benefits to video playback. MMX will improve IDCT performance with IEEE noncompliant quality because of its 16-bit precision limitation. IDCT represents 33% of video playback calculations. A high-quality, IEEE-compliant

MPEG video playback system requires more than MMX's 16-bit IDCT transform. Overall, MMX only provides marginal performance benefits to niche applications such as very low-end 3-D graphics, image processing, and nonstandard video playback. Alpha will continue to raise end user's performance expectations for emerging threshold multimedia applications, while supporting the balanced system concept to maximize performance/price ratio when hardware assist becomes cost competitive.

Apps \ CPU	Pentium-II W/ MMX 300MHz	Alpha 21164PC W/ MVI 533MHz
3D graphics	✗ 300K polygons/s (LinPack 100x100)	✓ > 720K polygons/s (LinPack 100x100)
Low end 3D graphics (S/W rendering)	✗ ¹ MMX	✓
Video conferencing	✗ ² MMX H.323 352x288 QCIF, 10 fps	✓ ^{MVI} H.323 LAN/H.320 ISDN 352x288 CIF, >20 fps
DVD playback (MPEG-II Video + AC-3 Audio)	✗ ³ MMX Needs H/W assistance	✓ Fully host based
DVD authoring (MPEG-II Video + AC-3 Audio)	✗ Needs H/W assistance	✓ ^{MVI-21264} Fully host based
MPEG-I A/V playback	✗ ⁴ MMX	✓
MPEG-I A/V authoring	✗ Needs H/W assistance	✓ ^{MVI} Host based 5.1 channels
Image processing	✓ ^{MMX}	✓
Concurrent video conferencing & 3D graphics	✗ Needs H/W assistance	✓

✗ : No benefit or marginal benefit ✓ : beneficial

Author's Contact Details

Joseph Chou
 Strategic Marketing Manager
 Digital Semiconductor
 77 Reed Road, HLO2-3/M07
 Hudson, MA01749
 Tel: (508) 568-5892
 E-mail: jchou@mail.dec.com