An Analysis of Multimedia Algorithms and Corresponding Hardware Architectural Support

K C Tang, Anthony S Fong, Angus K M Wu and Derek C W Pao
Department of Electronic Engineering
City University of Hong Kong
Tat Chee Ave, Kowloon, Hong Kong
E-mail: eefong@cityu.edu.hk

Abstract

Fast development of multimedia applications leads to new hardware requirements for achieving user acceptable performance throughput. The operations performed by a single multimedia application are very demanding on computing resources. Even though there are many multimedia enhanced hardware support features, the multimedia performance on these systems is not quite satisfactory. In this paper, we will describe several key multimedia algorithms and technologies, their computing resource requirements and the interaction with various hardware solutions.

Keywords: Multimedia, Computer Architecture, media-coprocessors, SIMD, MIMD, VLIW.

1. Introduction

The main reason for the less than satisfactory performance on processing multimedia data types is that multimedia data types are processed as conventional data, and processor registers in modern processor with multimedia processing capabilities [3,4] are used as temporary storage. In order to process large trunks of multimedia data, "move" instructions are used intensively to move multimedia data between memory and registers. The performance of this approach suffers from the large multimedia data block size which vastly exceeds a typical register size.

In order to develop an efficient multimedia system, the characteristics of various algorithms adopted in the multimedia applications must be fully understood. The algorithms used in multimedia applications can be categorized into sound codec, synthetic sound generation (e.g. MIDI), video sequence and image sequence codec (e.g. MPEG and JPEG), synthetic image, computer graphics, compression and decompression algorithms. These algorithms are of different nature, and therefore they have various computation requirements.

In this paper, we will briefly describe several multimedia algorithms in four main areas, namely computer graphics, video/audio, compression coding and saturated arithmetic. Then followed by the advantages and potential drawbacks for several existing hardware solution such as function specific processor, SIMD, MIMD and VLIW coprocessor. Finally, a comparison is drawn based on the multimedia application requirements together with suggestions on architectural alternatives.

2. Multimedia Algorithms and Key Technologies

The common algorithms used in computer animation are coordinate transformation, 3D object rendering, and texture mapping.

For the video/audio parts, MPEG video and audio codec are typical examples. The computation requirement for filtering and transform coding will be discussed.

Overflow has been a well known problem in most applications. If care has not been take to ensure the result do not overflow, data will be corrupted. In this paper, we will discuss the basic theory of saturated arithmetic, and also the pros and cons for both software and hardware saturated arithmetic implementation.

2.1 Computer Graphics

The most common operations are creation and modeling computer animation, changing the viewer coordinates, moving the 3D object from one place to another, and rotating the object by an arbitrary angle.

The above operations can be done through applying different transformation matrices on the 3D model. The general equation for these transformations is:
Where \( M \) is either a rotation, scaling, shearing or a composite transformation matrix in the form:

\[
M_{\text{comp}} = M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot \ldots \cdot M_j
\]

Each \( M_i \) is a rotational, scaling or shearing matrix [14].

The general form of composite transformation matrix \( M \) is:

\[
M = \begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_x \\
  r_{21} & r_{22} & r_{23} & t_y \\
  r_{31} & r_{32} & r_{33} & t_z \\
  0 & 0 & 0 & 1
\end{bmatrix}
\]

In order to move an object in a 3D world, each point defined by the object model must pass \( M \). Assuming a composite transformation matrix, \( M \) must be worked out before transformation. Nine multiplications and nine additions are required for a signal point of coordinate transformation.

After the transformation has been applied on the 3D model, projection should be carried out to project the 3D model onto a projection plane. This action is similar to taking a photo of the 3D model.

Simple projections such as orthogonal projection do not require any computation, since the object coordinate projected on the viewer plane is the original XY coordinate of the object in the world coordinate. However, orthogonal projections do not provide any ‘perspective’ for the viewer. Thus a perspective projection is more desirable. There are two possible perspective transformation according to the location of the projection plane [14]:

In Gouraud shading, intermediate shading information across an arbitrary polygon is being generated using linear interpolation and the key shading information can be obtained from the vertices. According to the figure, value of \( I_a, I_b \) and \( I_p \) are:

\[
 I_a = I_1 - (I_1 - I_2) \frac{y_1 - y_a}{y_1 - y_2} \\
 I_b = I_1 - (I_1 - I_3) \frac{y_1 - y_b}{y_1 - y_3} \\
 I_p = I_1 - (I_1 - I_a) \frac{x_b - x_p}{x_b - x_a}
\]
Another more realistic shading algorithm is Phong shading. This algorithm is based on generating shading information using the normal vector on the surfaces. An surface normal vector $\mathbf{N}_i$ can be obtained by interpolation of surface normal vector $\mathbf{N}_0$ and $\mathbf{N}_1$.

After a set of normal vector has been computed, any illumination model can be applied.

Most computer graphics algorithms are, in general, requires floating point representation due to having wider dynamic range and precision when compared with fixed point representation. For example, doing the perspective transformation, 3D object rotation and normal vector calculation in Phong shading, non integral number may result. This implies integer is not suitable for computer graphics applications. Thus floating point representation is used.

### 2.2 Video and Audio

The current trend in video and audio coding is to transform source data stream into another domain through transform coding such as discrete cosine transform (DCT). The aim of transform coding is to decorrelate the intra or inter frame error image content and to encode transform coefficients, rather than the original pels of the images [13]. The algorithm is to split input images into disjoint blocks of NxN pels b. The transformation can be represented as a matrix operation using an NxN transform matrix A. The NxN transform coefficient c can be obtained through linear forward transformation:

$$c = A b A^T$$

And the transform matrix A possess the property:

$$A^T = A$$

Thus the original b can be reconstructed by $A^T$:

$$b = A^T c A$$

The expression for a separable 2-D NxN DCT is:

$$F(u,v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos \left( \frac{\pi(2x+1)u}{2N} \right) \cos \left( \frac{\pi(2y+1)v}{2N} \right)$$

Where u, v, x, y equals to 0, 1, 2, ..., N1 and C(x) equals to one over root 2 if x is zero and one otherwise. The common value for N is 8 and 16. The above matrix equations demonstrate the close relation between MPEG video and audio algorithms and DCT, on which a large number of integer multiplications and additions are involved since the pixel values are represented using integer. Thus fast and high throughput integer multiplication and addition can boost video and audio performance by a noticeable amount.

The above computation requirement is also applicable on the upcoming MPEG-4 video standards. The main difference is that several video object planes exist in video sequence and the VOPs are shape coded before doing DCT.

Another new concept of video representation in MPEG-4 is embedded media player. Moreover, the multimedia application may have several program threads for decoding different video object planes (VOPs) [11]. Thus an object-oriented cross-platform solution is required to overcome the incompatibility problem. Java, designed as a platform independent language, was adopted as the platform of the future MPEG-4 developing platform.

However, Java is basically an executable source and requires an interpreter, Java Virtual Machine, to execute the Java executable bytecodes on the native operating system and processor. Various techniques have been explored to overcome slow execution of Java interpreter. One of these common techniques is called “Just in Time” compilation technique, which translates a single or group of Java bytecodes into machine instructions running a native operating system. Java. Object-oriented support may become another key multimedia application requirement in the near future.

### 2.3 Compression

A popular multimedia application is distributed video access/broadcast through the Internet. Over this information highway, however, the latency between successive packets is unpredictable and minimizing the number of information bits per image and audio signal implies more information can be delivered simultaneously.

According to the view of the end user, a successful interactive application is supposed to be real time, low latency, and is able to sustain high visual and audio quality. However, only limited bandwidth is available in most cases. In order to maintain high quality, short waiting time, and the decoding process can be started as soon as possible, entropy coding such as variable length coding has been widely used in performing compression for both video and audio data streams.
Multimedia data streams are usually entropy coded before or during transmission. Most common entropy coding algorithms require mapping data words into variable length bit strings after statistical analysis have been performed. Longer bit string will be assigned to the data pattern with higher entropy. This technique requires a lookup table for data word to bit stream mapping which can be predefined or building on the fly. The packing of the bit streams involve intensive shift and add/or operations.

2.4 Saturated Arithmetic
In video, audio and computer graphics, if the result after computation lies beyond a certain range, overflow will occur and this resulting value will be diverted from the expected value. This arithmetic warp around phenomenon will cause a sudden dark or bright pixel in video, an annoying pitch in the audio application and an object, color, and motion discontinuity in computer applications.

To remove such undesirable artifact, saturated arithmetic can be used. The basic idea of saturated arithmetic is to retain the result to a predefined maximum/minimum value even if overflow or underflow occur.

Saturated arithmetic can be implemented in either software or hardware. In the software approach, the operands have to be translate into representation with wider range before computation. The result is being checked against the predefined maximum/minimum value. The replacement criteria is the computed result fall outside of the expected range defined by maximum/minimum value.

In hardware implementation, one simple approach is introducing additional significant bit in the execution unit. The extended MSB can be served as an overflow indicator. When the result overflow, a certain predefined value will be put to the output of the execution unit instead of the computed result. This solution requires small increase in logic resources hence a small increase in silicon area. The software approach, however, requires more operation to do computation, which in general reduce performance.

3. Hardware Solutions

Many solutions have been introduced to increase multimedia performance. These solutions can be categorized into four different groups: Application specific chips(also known as function specific processor), multimedia enhanced microprocessor [3,4], MIMD parallel systems[7], and VLIW multimedia coprocessor [8,9,10]. Each of these solutions, however, has its own corresponding strengths and weaknesses[5,6] and will be discussed in the following sections. Knowing the behavior of the hardware architecture and the algorithms of multimedia applications can help exploring potential performance gain.

3.1 Function specific processor
The function specific processor system consists of a host general purpose processor and one or more function specific processor(s). Through the aids of device driver, the microprocessor governs all the activities of function specific processor. The job of the function specific chip is processor the data streams, acknowledge the host processor the data has been processed and wait for the host processor to get processed result and transferred to main memory if necessary.

This approach requires well-defined algorithms and can achieve a practical implementation at minimal cost and development time[5]. MPEG video decoding chips and 3D graphics coprocessor in PCs are typical examples [5]. The main drawback is the architecture has very limited flexibility and requires separated programming on device drivers in addition to the application programming. Both the function specific processor(s) and supporting programs quickly become obsolete when the standard changes.

3.2 SIMD General Purpose Processor
Many multimedia algorithms share the characteristics of low precision computation demanding such as motion estimation, discrete cosine transform and geometric transformation in computer graphics. Data level parallelism can be explored by introducing additional logic to partition a higher precision data path to handle multiple pieces of packed, lower precision data, with a single instruction such as Intel's MMX [4]. Hewlett Packard’s MAX and Sun Microsystems’ VIS [3] are typical examples of SIMD general purpose processor.

Within these architectures, for example, Intel’s MMX, the floating point pipeline was redesigned such that the pipeline is capable of processing and produce either eight 8bit data, four 16-bit data or two 32-bit data. All the instruction only support register direct mode, that is the instruction is in the format:

Two Addressing mode: OPCODE Ra, Rb
Three Addressing mode: OPCODE Ra, Rb, Rc

Where Ra, Rb and Rc are all register references.

Having only capability of exploring low precision data level parallelism, SIMD approach lacks supports for algorithms with data dependent decision trees[5] and a miss in predicting branch in the above general purpose
microprocessor results in flushing the pipeline and penalty cycles for fetching correct branch target instructions.

Another source of inefficiency is all the arithmetic operations are register based. To move data in main memory, registers are used and the result is sent back to main memory [2]. However, the size of data multimedia application process usually exceed the register size in an order of magnitude. Thus move instructions are used very frequently to move data from memory to register and vice versa. These multimedia data will not be reused in near future. For example, the pixel reconstructed from the MPEG video stream will be used for only 2 to 3 times for other frame reconstruction, but the number of pixel in a MPEG frame is in the order of several thousand, thus the pixel value may have to move between several times. This behavior locks up valuable processor registers and even hurt cache performance if these data has been encached.

3.3 MIMD Architecture

Task and data level parallelism can be achieved by implementing several programmable processing elements. To gain high performance improvements, the various processing elements are adapted to different coder subtasks in both the encoding and decoding processes [5,6]. This configuration leads to a heterogeneous architecture, consisting of multiple, programmable devices and function oriented blocks. Well defined low level tasks are mapped to specific, programmable or dedicated structures [5,6].

By using MIMD approach, the drawback of lacking data dependent decision support in SIMD architecture can be eliminated and linear arithmetic peak performance gain can be made by increasing the number of processing elements. Also, the memory latency can be hidden if there is sufficient programming threads to be executed [5]. Different tasks can be assigned to different processors and thus task level support can be achieved. However, this approach is costly, the size of memory and memory bandwidth must be increased to avoid starvation of the processing elements, and it also suffers from inter processing element communication problems similar to multiprocessor environment such as inter-processor data dependency and data coherency [5,6,7].

3.4 VLIW coprocessor

Besides task and data level parallelism, exploring instruction level parallelism offers a potential for significant performance improvements for programmable devices. Superscalar and very long instruction word (VLIW) architectures exploit these instruction level parallelism by issuing multiple independent instructions to different execution units. VLIW architecture is to have a single control unit to deliver a single fixed format for very long instruction word per cycle. Each instruction is composed of many independent instruction delivered to various execution units. The VLIW architecture itself does not provide mechanisms to resolve resource conflicts and the job was left to the compiler [5,6].

A VLIW media coprocessor and a host general purpose microprocessor can be integrated together to form a heterogeneous system [8,9,10]. While using a host processor to manage general operation and decision, tailor-made VLIW processors are used to handle low level multimedia algorithms. Thus task level parallelism can be explored.

The main drawback of this approach is instruction word incompatibility for the VLIW coprocessor and the host processor. Also, VLIW coprocessor mainly relies on the compiler to resolve any possible pipeline and resource conflicts with the aids of instruction optimization techniques such as software pipelining and trace scheduling. However, if intensive data dependency exists within the application, only lower than expected performance gain can be achieved since insufficient instruction level parallelism leads to pipeline stall and dead time. Thus the coprocessor has to wait for result computed by other unfinished instructions [5,6]. This approach also leads to separated programming on the host processor and the media coprocessor if an application needs to make use of the capability of the media coprocessor.

4. Comparison among the Architectures and Alternatives

All the above hardware solutions have their own strengths and weaknesses on different aspects in multimedia applications. The key nature of multimedia algorithms and possible alternatives are:

4.1 Massive low precision computation with saturated arithmetic support

In MPEG video and audio codec, DCT is usually adopted as fundamental algorithm to transform data streams into other domain after a statistical analysis on the data stream such as motion estimation and motion compensation has been carried out. These operations require huge fixed point addition and multiplication. In order to avoid visual artifact due to arithmetic overflow, saturated arithmetic should be performed. When comparing with software and hardware implementation, hardware approach is superior in terms of throughput and programming
complexity with sacrifice in small increase in silicon area.

4.2 Massive floating point computation
Most computer 3D models are represented in floating point number due to larger dynamic range that floating point number provides. Some transformation (e.g. Rotation, scaling and animation) may suffer from significant reduce in precision if fixed point representation is used. Multimedia application with synthetic images or 3D modeling requires huge amount of floating point multiplications and additions since most of the motion and change of view angle involve matrix multiplications. Fast floating point engine is adapted for matrix computation to boost computer graphics application performance.

4.3 Variable Bit Length Execution Engine
In entropy coding, the main goal is to map data word into variable length bit strings after doing statistical analysis. The packing of bit strings, however, requires intensive shift, addition, logical operation and checking the status of the buffer. This behavior usually occupies valuable execution units such as shifter, adder and logic unit. A dedicated variable bit engine with bit string indexing capability can be implemented to perform packing data words into bit streams and vice versa.

4.4 Possible Architecture Alternatives
In computer graphics applications, addition and multiplication are very common in coordinate transformation, modeling, rendering and etc. For multiplication, there is little to be done since pre-alignment and post-alignment are not necessary in most cases. However, floating point addition requires pre-alignment and post-normalization. The post-normalization can be combined with pre-alignment if the following operation applied on the result is also an addition. This approach can save significant amount of area and reduce the latency for a pipeline floating point adder. This approach is practical since the floating point datapath in state of the art microprocessors are 80 bits long, with 63 bit as mantissa and 16 bit for exponent and a sign bit.

5. Concluding Remarks
Individual approaches to multimedia computing have their own merits and disadvantages, when compared with others. The key for efficiency is to select the appropriate method for particular applications. Conventional computer architectures may not be ideal to process multimedia applications. A more comprehensive architecture such as extended descriptive computer may warrant in-depth investigation.

Reference