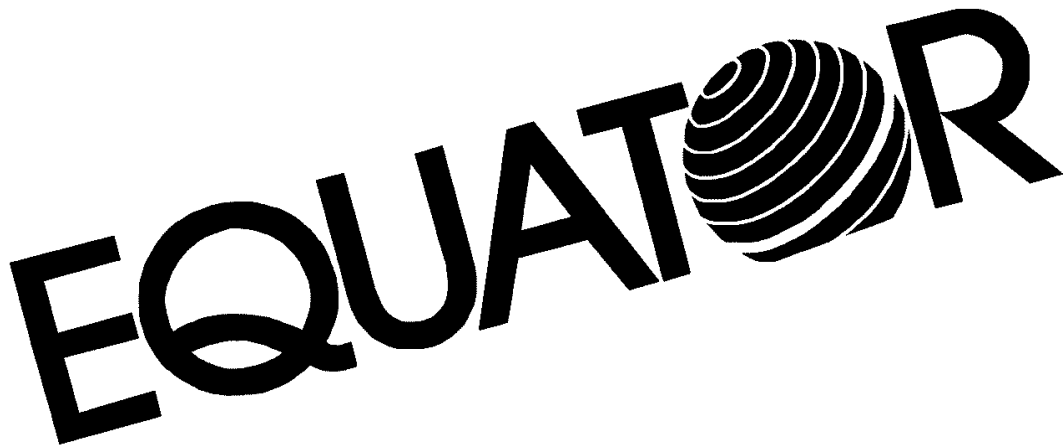

Equator Software Reference

Datasheet: MAP-CA DSP MPEG-4 Video Decoder

Equator Technologies, Inc.

May 11, 2001

Document Number: SWR.DS.M4VD.2001.05.11

The logo for Equator Technologies, featuring the word "EQUATOR" in a bold, sans-serif font, followed by a stylized globe icon composed of horizontal lines, and the letter "R" in the same font. The entire logo is tilted upwards from left to right.

EQUATOR

Equator Software Reference Datasheet: MAP-CA DSP MPEG-4 Video Decoder

May 11, 2001

Copyright © 2001 Equator Technologies, Inc.

Equator makes no warranty for the use of its products, assumes no responsibility for any errors which may appear in this document, and makes no commitment to update the information contained herein. Equator reserves the right to change or discontinue this product at any time, without notice. There are no express or implied licenses granted hereunder to design or fabricate any integrated circuits based on information in this document.

The following are trademarks of Equator Technologies, Inc., and may be used to identify Equator products only: Equator, MAP, MAP1000, MAP1000A, MAP-CA, MAP Series, Broadband Signal Processor, BSP, FIRtree, DataStreamer, iMMediaC, iMMediaTools, Media Intrinsic, VersaPort, SofTV, StingRay, Equator Around, and the Equator Around logo. Other product and company names contained herein may be trademarks of their respective owners.

The MAP-CA digital signal processor was jointly developed by Equator Technologies, Inc., and Hitachi, Ltd.

MAP-CA DSP MPEG-4 video decoder at a glance

The Equator Technologies, Inc. MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package provides an example implementation of MPEG-4 video image decoding on the MAP-CA digital signal processor. This software can be used by developers as a reference for creating their own MPEG-4 decoders for the MAP-CA DSP.

Features

The MAP-CA DSP MPEG-4 video decoder provides a number of important features for software developers:

- implementation as a well-documented set of C source code modules that utilize MAP-CA DSP Media Intrinsic C extensions
- inclusion of highly optimized, re-usable software modules, designed to allow users of the MAP-CA DSP to quickly adopt these modules to their own code base
- examples of efficient and parallel use of resources on the MAP-CA DSP, including
 - DataStreamer DMA controller
 - VLx coprocessor
 - Media Intrinsic C extensions
 - data cache and instruction cache

Benefits

The MAP-CA DSP MPEG-4 video decoder provides easily adaptable software modules for software developers. The benefits include

- substantial savings in development costs
- greatly improved time to market

Performance

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code is intended to provide a basis for customer development of MPEG-4 decoder software, and has not yet been fully optimized for performance, nor does it contain comprehensive error handling or other production-level code routines. As such, it does not demonstrate the full potential performance of optimized code running on the MAP-CA DSP.

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

Processor support

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package fully supports the MAP-CA digital signal processor.

Development platform support

The MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package fully supports the Shark series of evaluation and development platforms from Equator Technologies.

Functionality

The MAP-CA DSP MPEG-4 video decoder passes all MPEG-4 standard conformance streams. This decoder supports 4:2:0 coded video format and Simple Profile at Simple Level (SP@SL) video streams. The decoder performs DC prediction and AC prediction. The current version of this decoder does not support error checking.

Ordering information

To order the MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package, use part number 098-0208-10.

For more information, contact your nearest Equator sale office:

Corporate Headquarters

Equator Technologies, Inc.
1300 White Oaks Road
Campbell, CA 95008
Phone: (408) 369-5200
FAX: (408) 371-9106
Email: info@equator.com
URL: <http://www.equator.com>

Western North America

Equator Technologies, Inc.
19782 MacArthur Blvd, Suite 210
Irvine, California 92612
Phone: (949) 260-0974
FAX: (949) 263-0926
Email: info@equator.com

Eastern North America

Equator Technologies, Inc.
571 West Lake Avenue, Suite 12
Bay Head, New Jersey 08742
Phone: (732) 892-5221
FAX: (732) 892-5234
Email: info@equator.com

Japan

Equator Japan KK
Harmony Wing 5F
1-32-4 Hon-cho, Nakano-ku
Tokyo, Japan 164-0012
Phone + 81-3-5354-7530
FAX: +81-3-5354-7540
Email: info@equator.com

Europe

Equator Europe
Les Algorithmes
Batiment Aristote A 06410 Biot 2000 Route Des Lucioles
Sophia Antipolis, France
Phone: +33 (0) 492944716
FAX: +33 (0) 492944717
Email: info@equator.com

To locate your local Equator sales office, or for further information, go to <http://www.equator.com>

Type style conventions

With the exception of section and subsection headings, the formatting of text in this document follows the following conventions:

Normal descriptive text is presented in Times New Roman font.

Italicized Times New Roman text is used for document titles.

Underlined Times New Roman text is used for emphasis in normal descriptive text.

Any input or output text for any computer program is presented in Courier New font. This includes source code, command-line text, and program output.

Italicized Courier New text is used for any portion of a path, including individual file names.

Courier New text is used for any placeholder for a set of text input or output items for a program.

Table of contents

MAP-CA DSP MPEG-4 video decoder at a glance	1
Features	1
Benefits.....	1
Performance	1
Processor support	2
Development platform support.....	2
Functionality	2
Ordering information.....	2
Type style conventions	3
<u>Chapter 1</u> Overview of the MAP-CA DSP	7
VLIW core CPU.....	7
Pipelining of VLIW core instructions.....	8
Register files	9
Video and graphics coprocessors.....	9
Variable Length Encoder/Decoder (VLx)	9
Video Filter (VF)	9
DataStreamer DMA controller	9
I/O devices	11
Pipelined nature of decoding on the MAP-CA DSP	11
<u>Chapter 2</u> MPEG-4 video decoder.....	13
Overview of the MAP-CA DSP MPEG-4 video decoder.....	13
Variable length decoding.....	13
VLIW core processing.....	14
Data flow	14
Instruction cache management	15
Data cache management.....	15
Video display	15
Synchronization	16
Current implementation limitations.....	16
Example bitstreams and sample application.....	16
Accuracy.....	17
Performance	17
I-frame performance	17

P-frame performance	18
MPEG-4 video decoder API.....	19
Ordering information.....	20

Chapter 1

Overview of the MAP-CA DSP

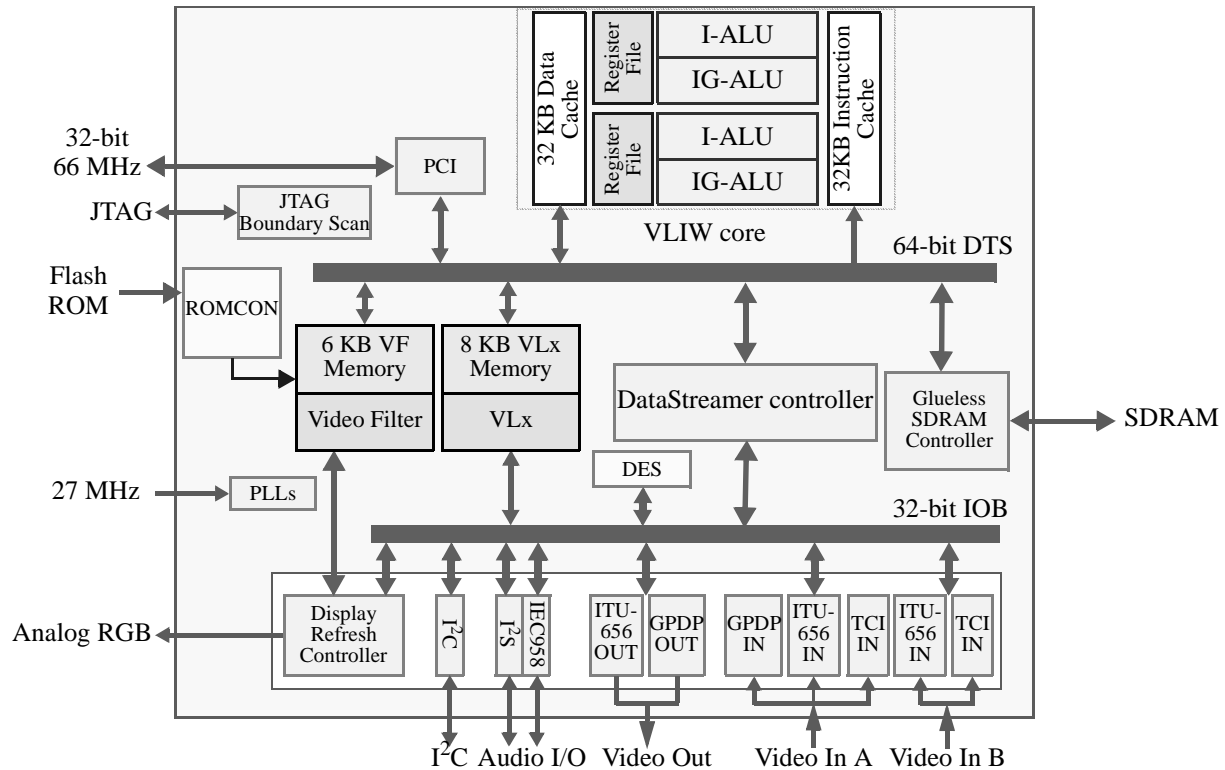


Figure 1-1: MAP-CA DSP block diagram

The MAP-CA digital signal processor is a high-performance, fully programmable processor developed for various multimedia applications that require real-time processing of video, audio, and image data. The MAP-CA DSP is capable of executing up to 30 billion operations per second on 8-bit data. The MAP-CA DSP also integrates several peripheral devices on the chip, such as input and output ports for real-time video and audio data.

1.1 VLIW core CPU

Much of the computational power of the MAP-CA DSP comes from its VLIW (very long instruction word) core central processing unit. In addition to offering high performance on intensive numeric and multi-dimensional matrix operations found in video and signal processing applications, the VLIW core CPU offers a high level of programmability and supports serial and irregular code found in control and data-driven functions. This joint ability to handle serial processing as well as parallel processing avoids the need for a separate host processor for control functions, thus making the MAP-CA DSP a cost-effective solution for consumer and embedded systems.

The VLIW core architecture is a fusion of general-purpose microprocessor and DSP paradigms. Like most modern RISC microprocessors, the VLIW core uses a traditional load/store architecture so that all operations are performed on the data available in the registers. Also integrated on the chip are a 32-kilobyte, 4-way set-associative data cache with true least-recently-used (LRU) cache replacement policy; a 32-kilobyte, 2-way set-associative instruction cache; and full virtual memory support with multiple translation look-aside buffers (TLB). On the other hand, like most modern DSPs, the VLIW core supports a wide array of partitioned and application-specific operations, which can be very efficient when used to operate on multimedia data such as image bitmaps and audio samples.

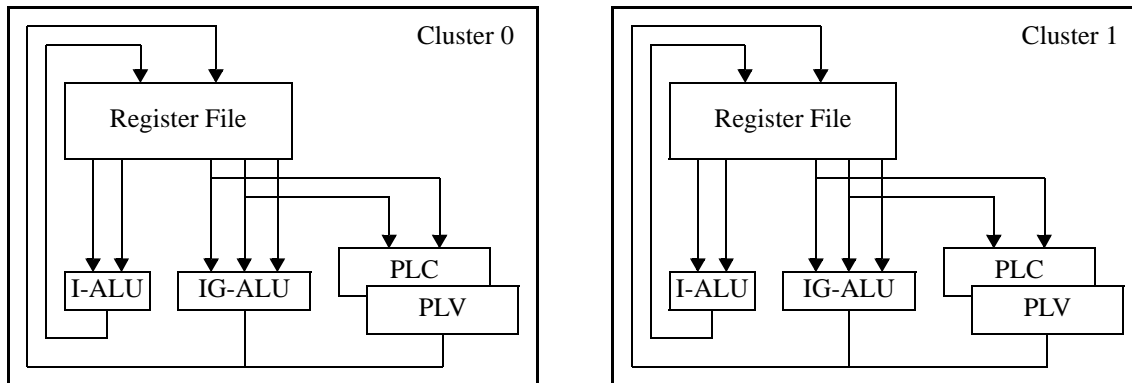


Figure 1-2: Simplified block diagram of two execution clusters in the MAP-CA DSP

The MAP-CA DSP's VLIW core consists of two execution clusters, as shown above. Each cluster contains an integer arithmetic and logic unit (I-ALU), an integer and graphics arithmetic and logic unit (IG-ALU), a register file consisting of thirty-two general purpose 64-bit registers, and a set of predicate and special registers. The I-ALU and IG-ALU on a cluster operate concurrently and independently of one another. They are controlled by separate opcode fields in the instruction word but share a register file and program counter.

The I-ALU executes 32-bit integer arithmetic operations, 32- and 64-bit load and store operations, and branch operations. This unit is primarily used for memory accesses, program flow control, and address calculations. The IG-ALU has a 64-bit adder and shifter, which can be partitioned into 8, 16, 32, or 64 bits for SIMD (single instruction, multiple data) operations. For example, a byte-wise partitioned subtract operation performs eight subtractions, each on an 8-bit partition. In other words, when a partitioned operation is executed, the same base operation (e.g., addition or subtraction) is applied to each partition independently. We thus get eight 8-bit results packed into a 64-bit value for byte-wise partitioned operations.

The IG-ALU is also capable of performing 128-bit integer MAD (mean absolute difference) operations as well as 128-bit vector sum operations. This allows the sum of absolute differences and inner product operations to each be computed with a single instruction, greatly improving the performance of video compression and decompression applications. One IG-ALU can execute eight 16-bit fixed-point MAD operations each cycle.

1.1.1 Pipelining of VLIW core instructions

Because of the way the execution stages are pipelined in the hardware, each execution unit is capable of independently issuing one instruction per cycle. Most integer instructions take only one cycle to complete, but partitioned instructions take multiple cycles for the result to become available and be written to the destination register. However, since a new instruction can be issued on every cycle, even

though the one previously issued has not been completed, the effective number of execution cycles per pipelined instruction can always be one.

1.1.2 Register files

The register file on each cluster consists of thirty-two 64-bit general-purpose registers. Since the I-ALU and IG-ALU on each cluster are connected to the same register file, each register can be accessed as a single 64-bit quantity or a pair of 32-bit quantities. This unique feature is very useful in transposing a 2-dimensional matrix, as discussed in later sections.

1.2 Video and graphics coprocessors

The MAP-CA DSP integrates on-chip coprocessors to remove from the VLIW core tasks that do not make efficient use of the VLIW core's wide data path. One such coprocessor on the MAP-CA DSP is the Variable Length Encoder/Decoder (VLx), a 16-bit microprocessor specifically designed for variable-length decoding and encoding. The Video Filter (VF) is a coprocessor integrated on the MAP-CA DSP to handle the scaling of images at the pixel clock rate.

1.2.1 Variable Length Encoder/Decoder (VLx)

The Variable Length Encoder/Decoder (VLx) is a 16-bit RISC microprocessor, with an instruction set optimized for bit-serial processing. The VLx's dedicated RAM (VLmem) consists of two 2-kilobyte banks for data and a 4-kilobyte bank for instructions.

The primary use of the VLx is the decoding and encoding of variable-length codes (VLCs). The bit-serial nature of the coded data and data dependencies in the control flow of video coding algorithms do not make efficient use of the highly parallel VLIW core. Such algorithms can be implemented more efficiently on serial processors like the VLx, which runs concurrently with the VLIW core. By using the VLx for sequential bit-parsing algorithms, the VLIW core is free to process more computationally intensive parallel code.

1.2.2 Video Filter (VF)

The Video Filter (VF) can scale an image to an arbitrary size using a separable convolution filter with five horizontal and three vertical taps. The VF can take images in YUV color format with 4:2:0 or 4:2:2 chroma sampling and produce output images with 4:4:4 chroma sampling to the Display Refresh Controller.

1.3 DataStreamer DMA controller

In several signal and image processing algorithms, the same sequence of operations is repeatedly performed on subsets of a large set of data. The access pattern of the data is typically sequential and known beforehand. Although most processors with cache memory allow a small subset of the data to be brought into the cache, this scheme does not fit well in signal and image processing algorithms. In the cache memory architecture, copying of the data from external memory to cache memory is initiated only after the data are found to not exist in the cache memory (cache miss). It takes several cycles to copy the

data from memory to cache, and during this time the processor is usually idle and cannot execute other instructions. In other words, on cache-based architectures, the data transfer from external memory to the cache does not overlap with data processing.

Since data access patterns are often regular and well known beforehand, many DSPs have employed a DMA (direct memory access) engine that can transfer a data set between external memory and the cache in parallel with the processing of a previous data set. For example, the Texas Instruments TMS320C80 contains the Transfer Controller, which can transfer data without the processor's intervention. Only the transfer parameters — such as the starting address and the byte to transfer — are set up by the processor at the beginning of the processing of a data set; once the Transfer Controller starts transferring the data, the processor is not involved in individual data transfers.

One of the disadvantages of this sort of DMA engine is that the on-chip memory must have a separate address space from the external memory, because the source and destination addresses specified by the transfer parameters must be unique. This necessitates that the on-chip memory be used only as a scratchpad memory rather than as cache memory.

The MAP-CA DSP provides the best characteristics of both approaches with the DataStreamer DMA controller, a sophisticated 64-channel DMA controller with 8 kilobytes of dedicated on-chip buffer memory. All 64 channels can be allocated at the same time; the DataStreamer DMA controller uses a priority-based scheme to schedule which channel will be active at any given time. In contrast with previous solutions using DMA engines, cache memory does not need to have a separate address space.

A DataStreamer DMA controller transfer is specified as a memory-to-memory copy, with the cache coherency mode specifying different types of transfer. If the source address is specified to be coherent and the destination address is specified to be non-coherent, then data will be read from the cache (assuming the data is already in the cache) and data will be written directly to external memory. Similarly, if the source address is non-coherent and the destination address is coherent, then data will be read directly from external memory and written to the cache. See the figure below for an illustration of the latter example.

Each memory-to-memory transfer requires two channels: one source channel and one destination channel. A buffer allocated from the 8-kilobyte buffer memory is used to hold the data temporarily between the source and destination channels. The source and destination channels, along with the buffer, constitute a data flow called a 'path'. For transfers to or from an I/O device, only one channel is specified, since the other channel is connected directly to the I/O device.

The transfer parameters — such as the starting address and the transfer size — are specified in a data structure called a 'descriptor'. The descriptors are allocated in memory; the DataStreamer DMA controller reads one descriptor at a time as it transfers data. Multiple descriptors can be chained, one pointing to another, so that transfers of data from disjoint memory locations or with different geometry can be performed without interrupting the VLIW core.

Unmodified C code will run through the normal cache mechanism.

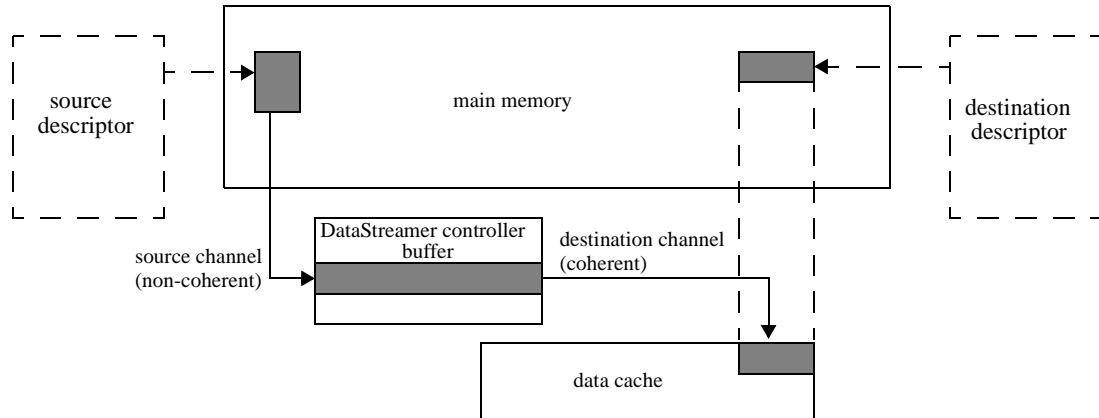


Figure 1-3: Transferring data from memory to the data cache using the DataStreamer DMA controller

1.4 I/O devices

The MAP-CA DSP integrates several I/O devices, including the Transport Channel Interface (TCI) to receive MPEG transport or program streams, a PCI bus interface for connection to another MAP-CA DSP or host system, and an I²C bus interface. For the display of images on a monitor, the MAP-CA DSP also has a Display Refresh Controller (DRC) that can accept image data in various formats and can multiplex between multiple image streams. The display timing is completely programmable in the DRC, supporting a range from interlaced NTSC scans to high-resolution progressive scans with up to 1280×1024 pixels. The output from the DRC can be sent to the internal DAC (digital-to-analog converter) for display on a high-resolution PC monitor or to an external NTSC encoder chip for display on an NTSC monitor.

1.5 Pipelined nature of decoding on the MAP-CA DSP

The VLIW core, coprocessors, DataStreamer DMA controller, and Display Refresh Controller operate in a pipelined fashion. For example, while the VLx coprocessor is decoding a macroblock, the VLIW core can be processing the previous macroblock. Also, while the Display Refresh Controller is scanning out a frame, the VLx and VLIW core can be decoding the next frame.

Chapter 2 MPEG-4 video decoder

2.1 Overview of the MAP-CA DSP MPEG-4 video decoder

MPEG-4 decoding on the MAP-CA DSP utilizes several functional blocks in parallel. First, the variable-length codes (VLCs) in the input bitstream are decoded by the VLx coprocessor. The decoded information is then passed to the VLIW core, which performs the pixel-processing operations — such as the inverse quantisation, inverse discrete cosine transform (IDCT), and half-pel interpolation — to reconstruct the coded picture. The reconstructed picture can be displayed on an NTSC monitor by the Display Refresh Controller (DRC) and an external NTSC encoder chip. Data transfers between the VLx, the VLIW core, the DRC, and memory are handled by the DataStreamer DMA controller so that the processing units do not wait for data to arrive from memory.

2.1.1 Variable length decoding

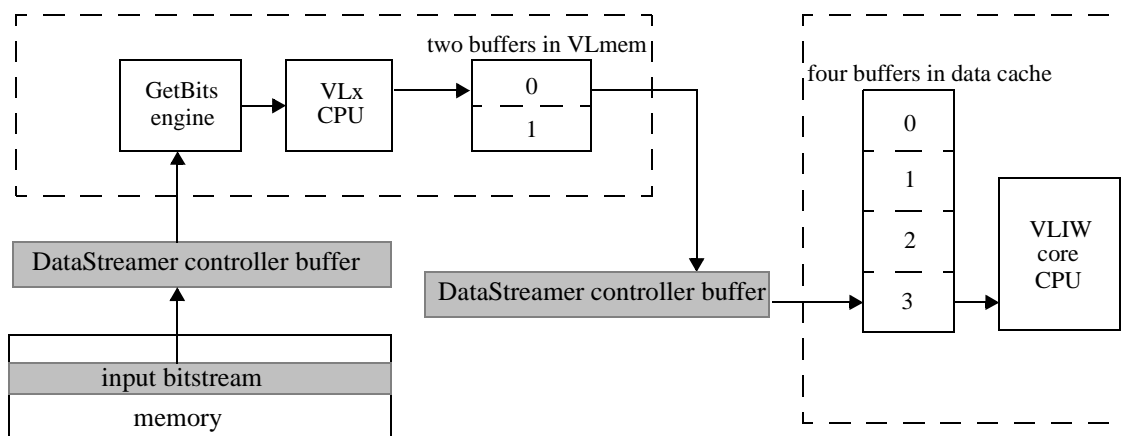


Figure 2-1: Transferring DCT coefficient blocks from the VLx to the VLIW core

The input bitstream arrives at the VLx via a DataStreamer DMA controller buffer. For a standalone configuration, the entire input bitstream is usually loaded into the memory by the host PC. In this case, the starting address is set to the beginning location of the bitstream in the memory, and the transfer size is set to the size of the bitstream. The DataStreamer DMA controller can loop the descriptor, so the bitstream can be repeated indefinitely if desired. For a streaming configuration where a small portion at a time of the input bitstream is passed from the host PC or TCI, the transfer descriptors specify a set of circular buffers in memory.

The VLx receives data from the DataStreamer DMA controller through the GetBits engine (GB). The VLx can request an arbitrary number of bits (up to 16) from the GetBits engine, which supplies the data from its 112-bit input buffer. When the GetBits engine runs out of bits, it reads the next 32 bits of data from the DataStreamer DMA controller buffer.

The VLx has three stages of processing and uses a dedicated region in its coprocessor memory to store the data in each stage. The first stage decodes all of the header symbols. The decoded information is stored in the header information buffer. The VLx then decodes the macroblock symbols and stores them into a macroblock information buffer. The third stage produces DCT coefficients in 8×8 matrix form.

Each macroblock in the MPEG-4 bitstream can contain up to six 8×8 blocks of DCT coefficients. The VLIW core allocates four circular buffers in the data cache to receive the DCT coefficients from the VLx, as shown in the figure above. The DataStreamer DMA controller source channel reads one macroblock of data from one of two buffers in the coprocessor memory and the destination channel stores them to one of four buffers in the data cache. After the transfer of a macroblock is complete, the source and destination channels advance to their next buffer, waiting for a command from VLx to transfer the next macroblock.

The transfer of macroblocks from the coprocessor memory to the data cache is done while the VLIW core is processing the previous macroblock. During the transfer, the VLIW core does not need to check for the arrival of each and every block. When the VLIW core is ready to process the next macroblock, it checks to see if all of the blocks from a macroblock have been transferred.

2.1.2 VLIW core processing

The VLIW core performs all of the pixel-processing operations. Using the header and macroblock information and the DCT coefficients from VLx, the VLIW core performs the inverse quantisation, inverse DCT, motion compensation, half-pixel interpolation, and pixel additions. The VLIW core also uses the DataStreamer DMA controller to load the pixel data from the reference pictures and to store the reconstructed picture.

2.1.2.1 Data flow

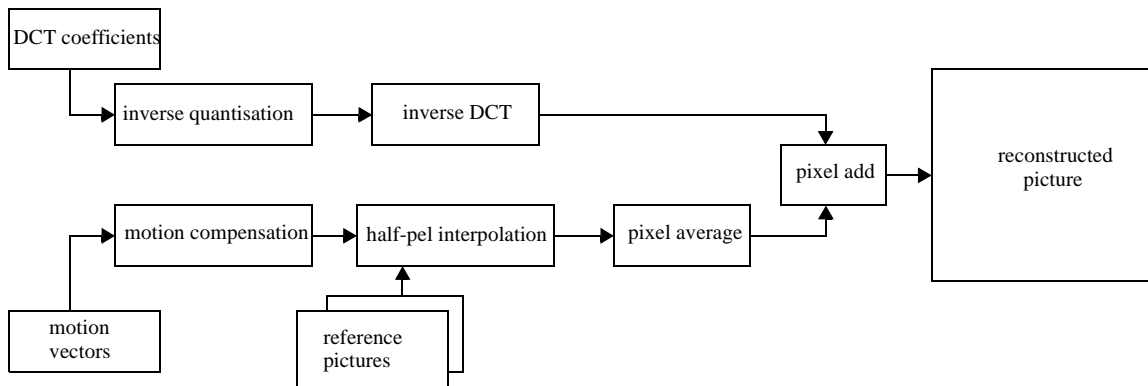


Figure 2-2: Data flow paths in VLIW core processing.

The VLIW core operates on a macroblock unit; this means that a complete set of operations is performed on a macroblock before the next macroblock is processed. There are two paths of data flow in the VLIW core, as shown above. The first path involves the inverse quantisation followed by the inverse DCT. This path produces six 8×8 arrays of 16-bit values, representing the reconstructed macroblock of the motion-compensated residual values. The second path performs motion compensation on the macroblock of reference pictures using the half-pel motion vectors decoded by VLx. This path also involves half-pel interpolation and bidirectional interpolation on previous and future reference macroblocks. The results from the two paths are then combined in the pixel addition stage, producing the reconstructed pixels of a macroblock.

2.1.2.2 Instruction cache management

The MAP-CA DSP MPEG-4 video decoder software has been designed to maximize performance by minimizing instruction cache misses during execution. The MAP-CA DSP MPEG-4 video decoder has been designed to break up the decoding algorithm along frame processing lines, using separate software routines for decoding each type of frame picture: I, P, and B. The routines for these frame types are designed so that each routine can fit into the 32K instruction cache. Alignment of functions is forced to 16K to ensure that the instruction cache is fully utilized.

Instruction cache misses are further reduced through the use of inline expansion of functions. Inlining can increase program performance by removing function call overhead and revealing additional opportunities for parallel scheduling of operations and other optimizations. All of the function calls inside the frame processing routines are inlined.

2.1.2.3 Data cache management

The MAP-CA DSP MPEG-4 video decoder software has been designed to maximize performance by minimizing data cache misses during execution. This is accomplished through the use of a specially designed C language structure, `DCACHE_MAP`, that contains a linear map of all locally used memory. This structure allows the MAP-CA DSP MPEG-4 video decoder to use no global variables; all of the shared variables used in the program are contained in the `DCACHE_MAP` structure.

2.1.3 Video display

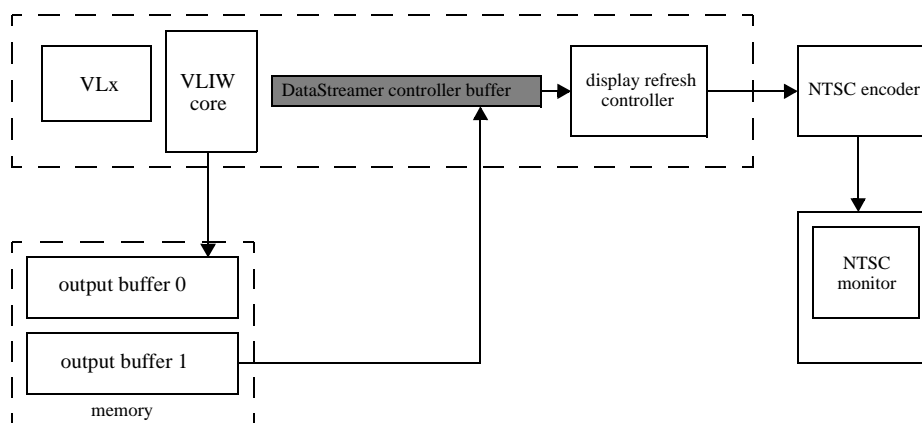


Figure 2-3: Data flow in displaying an image frame to an NTSC monitor through the DRC

The MAP-CA DSP MPEG-4 video decoder supports 4:2:0 coded video output.

An image frame to be displayed on a screen is transferred from the memory to the Display Refresh Controller (DRC) via the DataStreamer DMA controller, as shown above. Since the destination of the DataStreamer DMA controller transfer is an I/O device, only the source channel needs to be set up for this path. When the output device is an interlaced NTSC monitor, the source channel descriptor must be set up to read the scan lines in an interlaced order: all of the active scan lines in the first field are transferred first, followed by all of the active scan lines in the second field. The DRC generates the horizontal and vertical timing information and sends the formatted digital video data conforming to the ITU-R BT.656 standard to an external NTSC encoder chip.

It is also possible for the DRC to route the pixel data to an integrated digital-to-analog converter (DAC) on the MAP-CA DSP for display on a PC monitor. Only the progressive-scan timings are supported in

this mode. The Video Filter can be used to do the necessary conversion from the interlaced source image to a progressive format.

2.1.4 Synchronization

For the correct display of images, a new frame should be displayed only after the all of the scan lines in the current frame have been scanned out. A frame is read from the output buffer to the DRC via a DataStreamer DMA controller buffer. Associated with this buffer is a source channel descriptor that gives the starting address and size of the frame to be transferred. The starting address of a frame is read only at the beginning of the transfer of the frame. Therefore, as long as the transfer of a frame has begun, the starting address of the associated source channel descriptor can be changed without affecting the transfer of that frame. Before queuing a new frame, the core checks to see that the transfer of the previous frame has begun. If the transfer of the previous frame has already begun, the starting address of the source channel descriptor is modified to point to the next frame to be scanned out. This descriptor chains to itself (loops) at the end of the transfer of a frame to begin the transfer of the next frame — the one now pointed to by the new starting address. This scheme ensures that the transfer of a frame to the DRC will be completed before the transfer of the next frame begins, preventing a frame from being prematurely overwritten on the display by a new frame.

It is also necessary to synchronize decoding of frames in the VLIW core with the display. As the DataStreamer DMA controller is passing data from one output buffer to the DRC, the VLIW core is writing data to the other output buffer. Since the MAP-CA DSP MPEG-4 video decoder can decode a frame faster than real-time, it is necessary to synchronize decoding of frames in the VLIW core with the display to prevent the output buffer for the frame currently being scanned out from being overwritten by a new frame. The code running on the VLIW core incorporates a mechanism for waiting for the previous frame to be completely scanned out before the VLIW overwrites the output buffer from which that frame was read. A wait function checks for the current transfer address in the DataStreamer DMA controller; before starting to decode a new frame, the VLIW core calls this function to make sure that the buffer it will decode into is not currently being displayed. If the buffer is currently being displayed, the function will not return until the all of the lines in the frame have been scanned out.

2.2 Current implementation limitations

The current version of the MAP-CA DSP MPEG-4 video decoder passes all MPEG-4 standard conformance streams. This decoder supports 4:2:0 coded video format and Simple Profile at Simple Level. This includes AC prediction and short and long headers.

The current version of this decoder does not support error checking and does not function under the beta version of the VxWorks operating system.

2.3 Example bitstreams and sample application

The MPEG-4 video decoder library is shipped with example bitstreams and a sample application ready to run — once built from the source code — on any MAP-CA DSP platform.

2.4 Accuracy

In order to achieve consistent picture quality across different decoder implementations and technologies, the MPEG-4 standard uses the IEEE standard specification 1180-1990 to limit the inaccuracy of the IDCT outputs.

The MAP-CA DSP MPEG-4 video decoder IDCT implementation meets the IEEE specification in all tests. The overall mean error and mean square error are -0.00007 and 0.00816, respectively. The allowable limits are 0.0015 and 0.02, respectively. The largest mean error and mean square error in any of the 8×8 locations are 0.0029 and 0.0098, respectively, well within the allowed values of 0.015 and 0.06, respectively. The peak error at any of the 64 locations is one in magnitude, also meeting the specification.

2.5 Performance

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

2.5.1 I-frame performance

The first I-frame of the *coastguard.m4v* bitstream (included with the MAP-CA DSP MPEG-4 video decoder) was processed for this performance evaluation. The following table presents cycles needed for each sub-algorithm to process the frame:

IntraDCpred	203,515 cycles
IntraACpred	231,332 cycles
UpdateIntraACpred	154,662 cycles
Iquant	394,667 cycles
idct	255,886 cycles
PixelAddI	158,351 cycles
OutputData	73,017 cycles
PadBorder	18,342 cycles
Miscellaneous	1,631,111 cycles
Total	3,120,883 cycles

The “Miscellaneous” row of this table encompasses all tasks not specifically mentioned in the sub-algorithm performance breakdown (e.g., waiting for data from the VLx).

2.5.2 P-frame performance

Of the first 40 frames of the *coastguard.m4v* bitstream, the largest P-frame (the 27th frame overall, the 26th P-frame) was processed for this performance evaluation. The following table presents cycles needed for each sub-algorithm to process the frame:

ProcSkippedP	not needed for this frame
PadBorder	not needed for this frame
ComputeMV	87,760
LoadRefP	94,602
UpdateIntraACpred	304,503
IntraDCpred	not needed for this frame
IntraACpred	not needed for this frame
Iquant	334,587
idct	234,128
InterpP	166,146
PixelAddP	219,146
PixelAddI	not needed for this frame
OutputData	64,373
PadBorder	15,958
Miscellaneous	1,382,952
Total	2,904,155

The “Miscellaneous” row of this table encompasses all tasks not specifically mentioned in the sub-algorithm performance breakdown (e.g., waiting for data from the VLx).

The MAP-CA DSP is capable of decoding two Simple Profile at Simple Level (SP@SL) MPEG-4 video streams in real time, with sufficient cycles left to perform other tasks, such as the audio and system decoding.

2.6 MPEG-4 video decoder API

The MAP-CA DSP MPEG-4 video decoder is written with an application program interface (API) that aids in the management of caches and buffers, the setting up of the DataStreamer DMA controller and VLx, initialization of data items, and various other low-level tasks.

The MAP-CA DSP MPEG-4 video decoder API uses the `Mpeg4VdecStatus` structure to communicate between API elements. The pointers `frameJustDecoded`, `frameToQueueForDisplay`, `frameToDecode`, `frameToDecodeIntoNext` point to the internal frame buffers of the decoder and are rotated with the API calls. The `dcachePtr` points to the structure containing all local memory references. The `decodeValid` field is set if the decoder determines a new picture exists in the stream. The `frame_no` field is the frame number of the frame to be decoded next. The `result` field is reserved for error flags.

```
typedef struct
{
    unsigned char *frameJustDecoded;
    unsigned char *frameToQueueForDisplay;
    unsigned char *frameToDecode;
    unsigned char *frameToDecodeIntoNext;
    // unsigned char *BframeToDecodeIntoNext; // For PB-mode
    // unsigned char *BframeToQueueForDisplay; // For PB-mode display
    void *dcachePtr;
    int decodeValid;
    int pb_mode;
    int frame_no;
    SCODE result;
} MPEG4VdecStatus;
```

```
void
MPEG4VideoDecodeOpen(
    void * bitstreamAddr,
    unsigned int bitstreamNumFrames,
    unsigned int bitstreamLength,
    MPEG4VdecStatus *mpeg4VdecStatus);
```

This call does the following.

1. Initializes the data cache pointer to allow for data cache management.
2. Initializes the frame buffer pointers for the double buffering of the input and output frame buffers.
3. Initializes various arrays, tables, and data values.
4. Sets up the DataStreamer DMA controller.
5. Sets up the GetBits data for the VLx.
6. Loads and starts the VLx program.
7. Initializes the `Mpeg4VdecStatus` structure.

```
void
MPEG4VideoDecodeFrame(MPEG4VdecStatus *mpeg4VdecStatus);
```

This call does the following.

1. Decodes an I, P, or B frame based on the next picture header in the bitstream.
2. Outputs the decoded frame into a frame buffer in SDRAM.
3. Rotates the output frame buffer's pointer.
4. Modifies `Mpeg4VdecStatus` appropriately.

Ordering information

To order the MAP-CA DSP MPEG-4 Decoder Media Library Reference Source Code Software Package, use part number 098-0208-10.

For more information, contact your nearest Equator sale office:

Corporate Headquarters

Equator Technologies, Inc.
1300 White Oaks Road
Campbell, CA 95008
Phone: (408) 369-5200
FAX: (408) 371-9106
Email: info@equator.com
URL: <http://www.equator.com>

Western North America

Equator Technologies, Inc.
19782 MacArthur Blvd, Suite 210
Irvine, California 92612
Phone: (949) 260-0974
FAX: (949) 263-0926
Email: info@equator.com

Eastern North America

Equator Technologies, Inc.
571 West Lake Avenue, Suite 12
Bay Head, New Jersey 08742
Phone: (732) 892-5221
FAX: (732) 892-5234
Email: info@equator.com

Japan

Equator Japan KK
Harmony Wing 5F
1-32-4 Hon-cho, Nakano-ku
Tokyo, Japan 164-0012
Phone + 81-3-5354-7530
FAX: +81-3-5354-7540
Email: info@equator.com

Europe

Equator Europe
Les Algorithmes
Batiment Aristote A 06410 Biot 2000 Route Des Lucioles
Sophia Antipolis, France
Phone: +33 (0) 492944716
FAX: +33 (0) 492944717
Email: info@equator.com

To locate your local Equator sales office, or for further information, go to <http://www.equator.com>