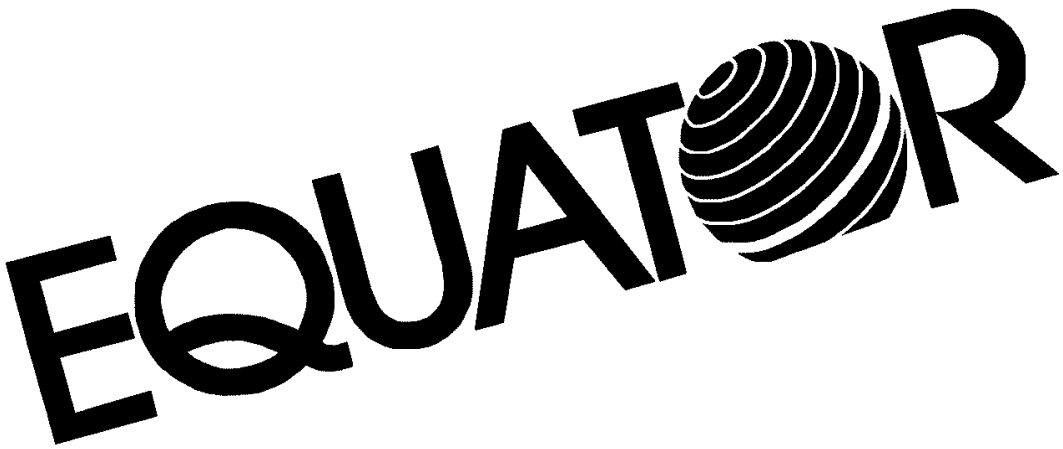
Equator Software Reference

Datasheet: MAP-CA DSP H.263+ Video Encoder

Equator Technologies, Inc.

June 25, 2001

Document Number: SWR.DS.HPVE.2001.06.25



Equator Software Reference
Datasheet: MAP-CA DSP H.263+ Video Encoder

June 25, 2001

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

MAP-CA DSP H.263+ video encoder at a glance

The Equator Technologies, Inc. MAP-CA DSP H.263+ Encoder Media Library Product Source Code Software Package provides an implementation of H.263 video image encoding on the MAP-CA digital signal processor. This package can be used as a turnkey solution to H.263+ video encoder needs or as a reference for the development of H.263 encoders for the MAP-CA DSP.

Features

The MAP-CA DSP H.263+ video encoder 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 Intrinsics 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 MAP-CA DSP on-chip resources, including
    - DataStreamer DMA controller
    - VLx coprocessor
    - Media Intrinsics C extensions
    - data cache and instruction cache

Benefits

The MAP-CA DSP H.263+ video encoder 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 H.263+ Encoder Media Library Product Source Code has been optimized for performance.

The MAP-CA DSP H.263+ video encoder can encode a frame in less than 2.1 million cycles. This translates to just over 60 million cycles required for encoding of an H.263 CIF (Common Intermediate Format) video stream.

The MAP-CA DSP H.263+ video encoder can encode four H.263 streams in real time, with sufficient cycles left to perform other tasks, such as audio and system encoding.

Processor support

The MAP-CA DSP H.263+ Encoder Media Library Product Source Code Software Package fully supports the MAP-CA digital signal processor.

Development platform support

The MAP-CA DSP H.263 Encoder Media Library Product Source Code Software Package fully supports the Shark and StingRay evaluation and development boards from Equator Technologies.

Functionality

The current version of the MAP-CA DSP H.263+ video encoder supports encoding of P and I type frames. The encoder supports the GOB (group of blocks) layer and both INTER and INTRA type macroblocks. The encoder provides rate control.

The MAP-CA DSP H.263+ video encoder uses motion estimation with a –16/+15 full search in the horizontal direction and a +/-8 binary search in the vertical direction. Each step of the binary search is a half search, meaning that every other pixel location is searched. Half-pixel motion estimation is performed in three directions. The encoder contains “early out” code, which exits the search as soon as minimum criteria are met.

The MAP-CA DSP H.263+ video encoder uses the VLx for generating the output bitstream. The bitstream is placed in a buffer in SDRAM as specified by the calling application. The buffer size must be evenly divisible by 2048 bytes. The current implementation requires input images of size 720 × 480 in 4:2:2 format. The sub-QCIF, QCIF, CIF (Common Intermediate Format), and 4CIF source formats are supported. The current version of the H.263 encoder requires recompilation in order to change the size of frame that is encoded.

The MAP-CA DSP H.263+ video encoder implements all elements and functionality of ITU-T H.263, plus Annexes D, J, L, and T.

Ordering information

To order the MAP-CA DSP H.263+ Encoder Media Library Product Source Code Software Package, use part number 098-0201-20.

For more information, contact your nearest Equator sales 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.

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

Table of contents

Chapter 1 Overview of the MAP-CA DSP

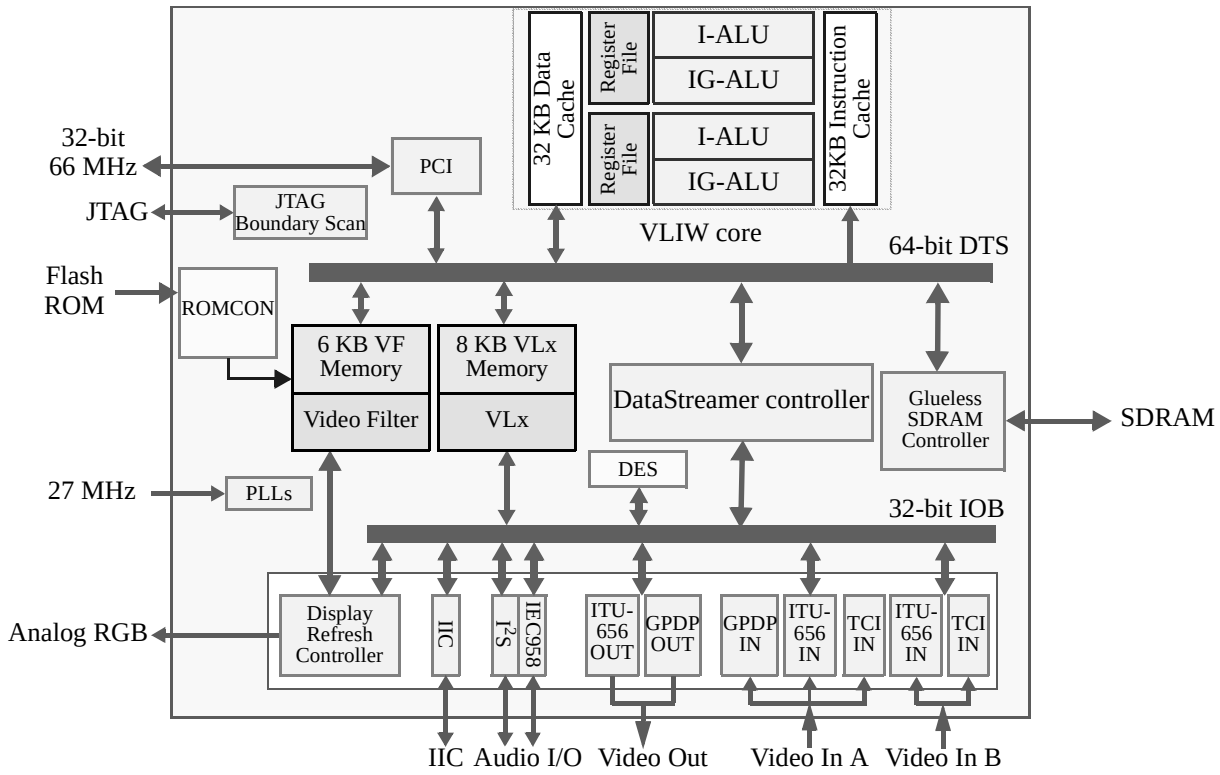


Figure 1-1: MAP-CA DSP block diagram

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.

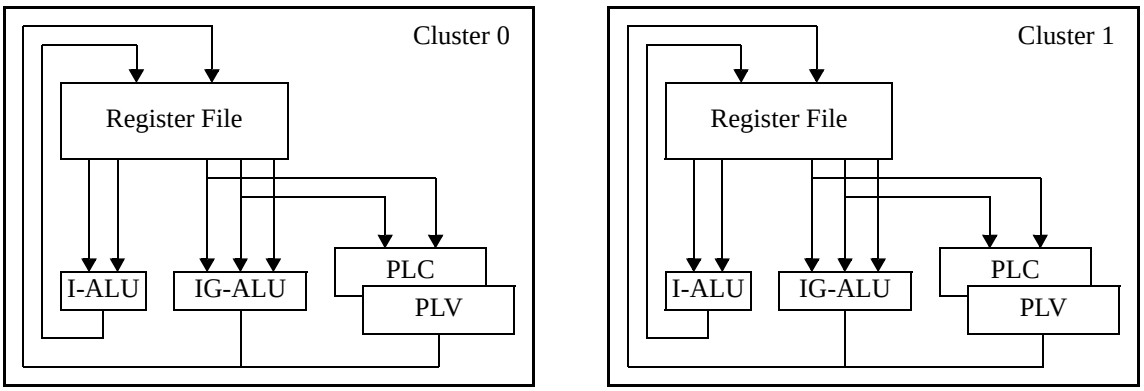


Figure 1-2: Simplified block diagram of two execution clusters in the MAP-CA DSP

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 other. 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.

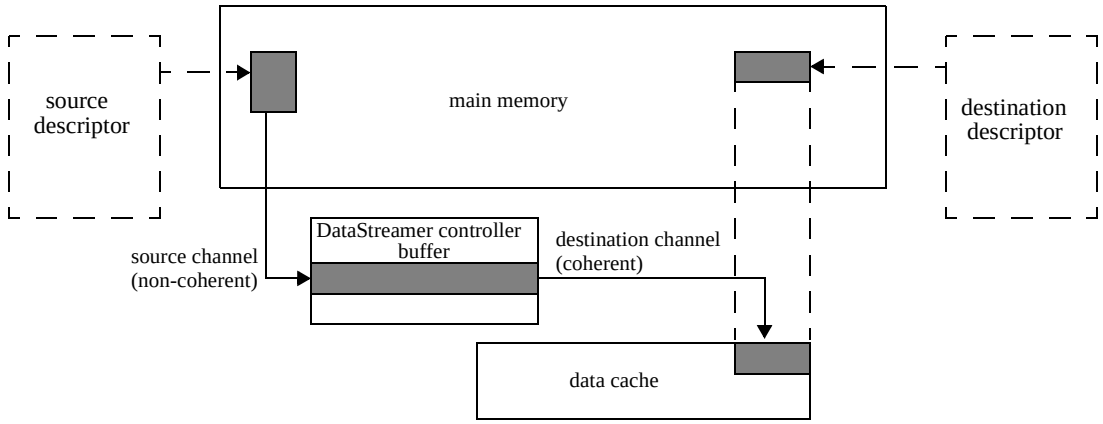


Figure 1-3: Transferring data from memory to the data cache using the DataStreamer DMA controller

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 IIC serial I/O 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.

ITU-T H.263 supports progressive scans only, so the interlaced scan functionality of the MAP-CA DSP is not exercised by the H.263 coding applications.

1.5 Pipelined nature of encoding and 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 H.263+ video encoder

2.1 H.263+ video encoder application overview

The standard defined by ITU-T H.263 is used for compression of moving pictures at low bit rates. The basic configuration of the video source coding algorithm is based on the ITU-T H.261 recommendation. This video coding algorithm uses inter-picture prediction to take advantage of temporal redundancy; the algorithm then uses transform coding of the remaining signal to reduce spatial redundancy.

An H.263 encoder must contain within it the functionality of an H.263 decoder, as well. Encoded bitstreams are decoded to produce reconstructed pictures that are used in computing motion vectors and, optionally, for display to a monitor at the encoding site.

The MAP-CA DSP H.263+ video encoder utilizes several functional blocks in parallel. First, the VLIW core performs the pixel operations such as motion estimation, discrete cosine transform (DCT), and quantization. Then the quantized information is passed to the Variable Length Encoder/Decoder (VLx) coprocessor. The VLx coprocessor creates the encoded bitstream, a stream of variable-length codes. The VLIW core then performs inverse quantization, inverse discrete cosine transform (IDCT), and half-pixel interpolation to reconstruct the encoded picture. Reconstructed pictures are then optionally displayed on an NTSC monitor by the DRC and an external NTSC encoder chip. Data transfers between the VLIW core, DRC, VLx, 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 VLIW core processing

The VLIW core performs all of the pixel operations. The VLIW core uses the DataStreamer DMA controller to read the pixel data from the input picture and from the reference picture. The VLIW core performs the motion estimation, DCT, and quantization. The DataStreamer DMA controller transfers the quantized information and macroblock header information to VLx memory (VLmem). The quantized information is double buffered to allow the core to continue reconstructing the reference pictures while the DataStreamer DMA controller transfers the data. The VLIW core performs the inverse quantization, inverse DCT, motion compensation, half-pixel interpolation, and pixel additions necessary to reconstruct the reference pictures.

2.1.2 Variable-length encoding

The DataStreamer DMA controller passes the quantized information and macroblock header data to the VLx. In turn, the VLx outputs the data through the GetBits engine. The VLx can output an arbitrary number of bits (up to 16) to the GetBits engine, which supplies the data to the DataStreamer DMA controller buffer.

VLx processing occurs in several stages. First, the VLx processes picture header information. The VLIW core uses PIO writes to pass picture header parameters to VLx memory (VLmem). The core then sends a command to the VLx to splice out picture header values to the output buffer. The VLIW core uses PIO writes to pass Group of Blocks (GOB) header parameters to VLmem. The core then sends a command to the VLx to splice out GOB header values to the output buffer. The VLx also processes the DCT coefficients and performs variable-length coding of the output macroblock information bitstream.

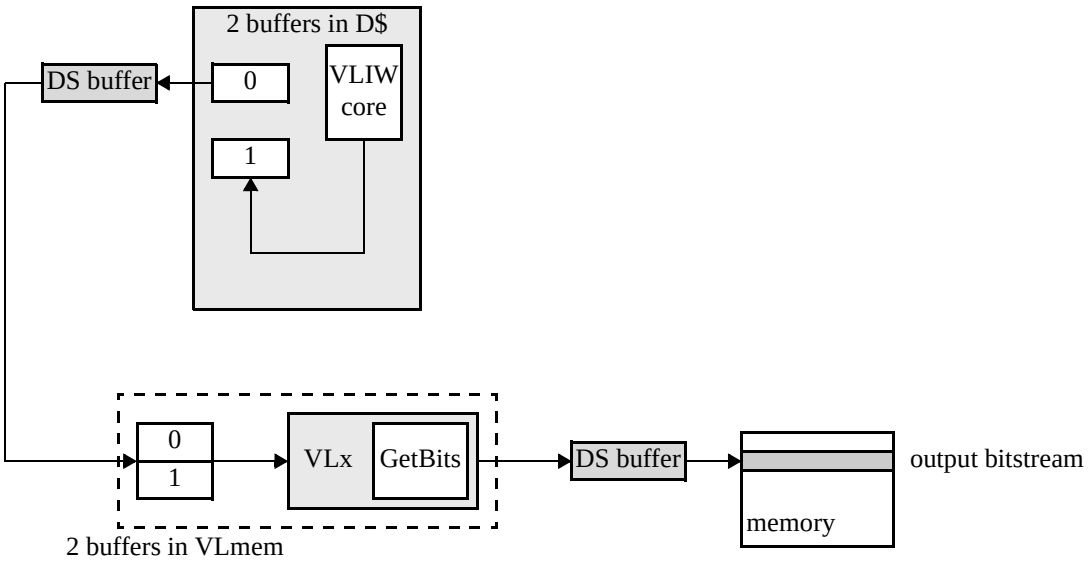


Figure 2-1: Transferring DCT coefficients from data cache (D$) to VLmem

2.1.3 Reconstructed data flow.

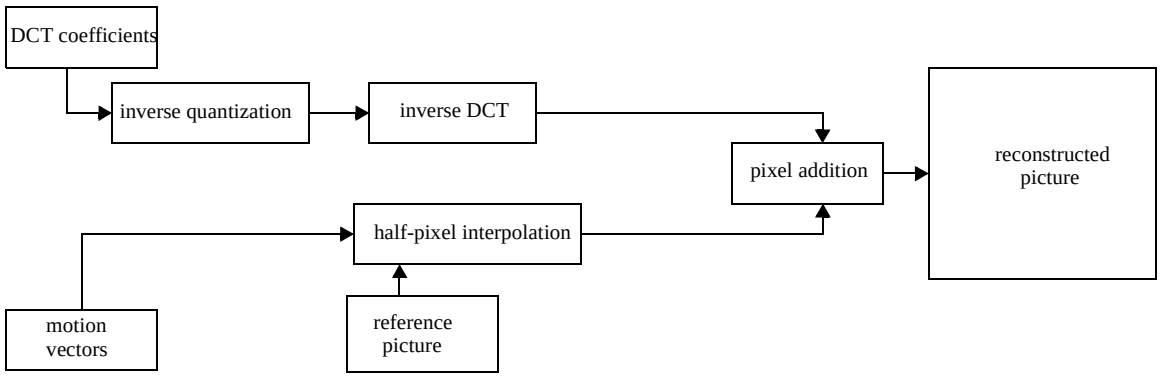


Figure 2-2: Data flow paths in VLIW core processing of reconstructed picture

The VLIW core operates on a macroblock unit; 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 reconstruction processing. The first path involves the inverse quantization 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 the motion compensation on the macroblock of reference pictures using the half-pixel motion vectors. This path also involves the half-pixel interpolation on previous reference macroblocks. The results from the two paths are then combined in the pixel addition stage, producing the reconstructed pixels of a macroblock

2.1.4 Video display

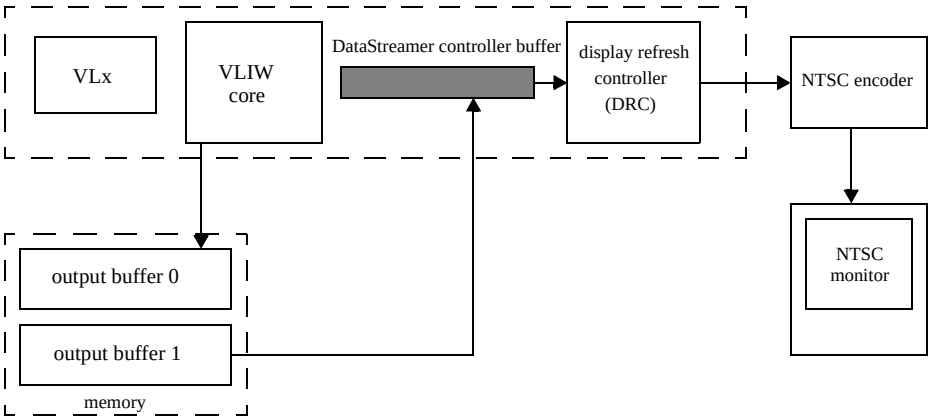


Figure 2-3: Data flow in displaying an image frame to an NTSC monitor through the DRC

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 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.

2.1.5 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 H.263+ video encoder 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.

2.1.6 Global coding methods

2.1.6.1 Instruction cache management

The MAP-CA DSP H.263+ video encoder software has been designed to maximize performance by minimizing instruction cache (I-cache) misses during execution.

The H.263+ video encoder algorithm is structured to use separate software routines for encoding each type of picture: I and P. The routines for these picture types are designed so that each routine can fit into the MAP-CA DSP's 32-kilobyte instruction cache. Alignment of functions is forced to 16Kb, ensuring full utilization of the instruction cache.

Instruction cache misses are further reduced through the use of inline expansion of functions. This ensures that multiple functions in a routine will not have the same I-cache address — which would cause I-cache misses. 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 `EncodePictureI` and `EncodePictureP` routines are inlined.

2.1.6.2 Data cache management

The MAP-CA DSP H.263 encoder software has been designed to maximize performance by minimizing data cache misses. This is accomplished in part through the use of `DCACHE_MAP`, a 48-kilobyte C language structure that contains a linear map of all locally used memory. The H.263+ video encoder program uses no global variables; all of the shared variables used in the program are contained in the `DCACHE_MAP` structure.

2.2 Memory usage

SDRAM is consumed by

- 31 kilobytes of instructions (up to 50 kilobytes with annexes),
- bitstream output buffers of size determined by user (minimum size of 2 kilobytes, typical size on the order of 1 megabyte),
- 1.4 megabytes of double-buffered input buffers (for 4CIF camera input), and
- 48 kilobytes for the `DCACHE_MAP` structure.

2.3 Latency

Operating with source images in Common Intermediate Format (CIF), the current MAP-CA DSP H.263+ video encoder takes 33 msec to capture a full picture frame and 7 msec processing time to encode a single stream. The worst-case encoder delay (latency) for encoding four streams would therefore be 33 msec for capture plus 4 x 7 msec for processing. The capture time of 33 msec can be cut in half by modifying the current frame capture driver software to start encoding after the first of the two fields has been received, with the second field not used. This reduces the total encoder delay for a single stream to 16.5 msec + 7 msec = 23.5 msec

2.4 Accuracy

In order to achieve consistent picture quality across different implementations and technologies, ITU-T H.263, Annex A uses the IEEE standard specification 1180-1990 to limit the inaccuracy of inverse discrete cosine transform (IDCT) output.

The MAP-CA DSP H.263+ video encoder 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. 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 1 in magnitude, also meeting the specification.

2.5 Current implementation features and limitations

The current version of the MAP-CA DSP H.263+ video encoder supports encoding of P frames and I frames.

The current version of the H.263 encoder supports the GOB (group of blocks) layer and both INTER and INTRA type macroblocks.

The sub-QCIF, QCIF, CIF (Common Intermediate Format), and 4CIF source formats are supported. 4CIF source format is only supported without the GOB (group of blocks) header.

The current version of the H.263 encoder provides rate control.

The current version of the H.263 encoder uses motion estimation with a –16/+15 full search in the horizontal direction and a +/-8 binary search in the vertical direction. Each step of this binary search is a half search, meaning that every other pixel location is searched. Half-pixel motion estimation is performed in three directions. The encoder contains “early out” code, which causes the motion search to terminate as soon as minimum match criteria are met.

The current version of the H.263 encoder uses the VLx for generating the output bitstream. The bitstream is placed in a buffer in SDRAM as specified by the calling application. The buffer size must be evenly divisible by 2048 bytes. The current implementation requires input images of size 720×480 in 4:2:2 format.

The current version of the H.263 encoder requires recompilation in order to change the size of frame that is encoded.

The MAP-CA DSP H.263+ video encoder implements all elements and functionality of ITU-T H.263, plus Annexes D, J, L, and T.

2.6 Performance

In this section, performance information is provided for macroblock loop processing in the I-frame and P-frame control data flow. The first 80 frames of a representative video input sequence (Paris) were used in the performance measurement reported on below. The number of consecutive P-frames in the sequence was four; thus, there were 16 I-frames and 64 P-frames used in the performance measurement. The tables below present the median number of clock cycles per frame on a routine-by-routine basis.

Overhead due to performance measurement can result in cache misses that artificially increase the number of clock cycles above the number achieved by an actual video encoding application. The numbers in the tables below should therefore be viewed as approximations.

2.6.1 I-picture encoding performance

Performance information, measured in clock cycles, is provided for macroblock loop processing in the I-frame control data flow. Quantization was set to level 6 for this measurement.

The total number of cycles for an I-frame is less than 2.1 million cycles, which includes data caching and measurement overhead not itemized in the table below.

routine	cycles/1000
`InputMB`	41
`Convert422to420`	221
`DCT`	342
`Quantize`	196
`SendVlxMB`	60
`Dequant`	185
`IDCT`	260
`ReconImage`	43
`OutputMB`	71

Table 2-1: I-picture encoding performance

2.6.2 P-picture encoding performance

Performance information, measured in clock cycles, is provided for macroblock loop processing in the P-frame control data flow. Quantization was set to level 6 for this measurement.

The total number of cycles for an P-frame is less than 2.1 million cycles, which includes data caching and measurement overhead not itemized in the table below.

routines	cycles/1000
`InputMB`	42
`PframeConvert422to420`	147
`LoadRefMB`	46
`MotionEst`	350
`PredP`	156
`DCT`	340
`Quant`	203
`SendVlxMB`	101
`Dequant`	185
`IDCT`	139
`PredReconP`	253
`OutputMB`	74

Table 2-2: P-picture encoding performance

2.7 Example bitstreams and sample application

The H.263+ video encoder 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.

2.8 H.263+ video encoder API

The MAP-CA DSP H.263+ video encoder 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 H.263+ video encoder API uses the `H263VencStatus` structure to communicate between API elements.

```
typedef struct
{
   unsigned char        *frameToQueueForDisplay;
   unsigned char        *frameToDecodeIntoNext;
   unsigned char        *frameToCapture;
   unsigned char        *frameToEncode;
   unsigned char        *FFprevAddr;
   int                  sourceFormat;
   int                  currentAddr;
   unsigned char        *null_input;
   unsigned char        *output_buf;
   unsigned char        *buffersForDS;
   int                  output_buf_size;
   int                  totalQP;
   int                  totalBits;
   int                  bitRate;
   int                  targetBits;
   unsigned char        *dcacheMemory;
   int                  pels;
   int                  lines;
   int                  e_lines;
   int                  e_pels;
   int                  freeze_request;
   int                  freeze_release;
   unsigned int         QI;
   unsigned int         QP;
   unsigned int         P_rate;
   unsigned int         pcount;
   int                  PTYPE;
   PVLX_BIN             pVlxBin;
} H263VencStatus;
```

The pointer `frameToQueueForDisplay` points to the output buffer where the reference frame will be stored. The pointer `frameToDecodeIntoNext` points to the other buffer of the output frame double buffer set. The pointer `frameToEncode` points to the input buffer holding the input frame. The pointer `frameToCapture` points to the other buffer of the input frame double buffer set. The calling application must provide the encoder with these pointers.

`FFprevAddr` is a pointer to the ending address of the previous frame or the beginning of the current frame. See usage with `currentAddr`, defined below.

`sourceFormat` indicates the source format with which the image is to be encoded. The source formats supported are sub-QCIF, QCIF, CIF (Common Intermediate Format), and 4CIF. The values of `sourceFormat` corresponding to these formats are 1, 2, 3, and 4 respectively (as described in section 5.1.3 of ITU-T H.263).

`currentAddr` indicates the address to which the DataStreamer DMA controller is currently writing output. In conjunction with output_buf, this value can be used to determine how much of the output buffer has been consumed. `FFprevAddr` can be used in conjunction with `currentAddr` can be used to copy the currently encoded frame to another destination.

`null_input` is a pointer to a 64-byte buffer that is used to set up the VLx GetBits engine.

`output_buf` is a pointer provided by the application that dictates the starting address of the encoder's circular outbut buffer.

`output_buf_size` indicates the size of the output buffer, in bytes. This must be divisible by 2048.

`buffersForDS` is a pointer used by the encoder library's caching mechanisms.

`totalQP` is the sum of all the `QP` values on a macroblock basis. Hence, the average QUANT for the encoded picture is the value of `totalQP` divided by the number of macroblocks in the picture.

`totalBits` indicates the number of bytes encoded into the bitstream for the encoded picture. This field is not the number of bits. The semantics or naming of this field may be changed in future versions of this encoder.

`bitRate` indicates the number of encoded bits per second.

`targetBits` indicates the number of encoded bits per picture.

`dcacheMemory` is a pointer to memory allocated by the application for use in the library.

`pels` and `lines` indicate the width and height of the source image to be encoded in terms of pels (picture elements) and lines. (For example, an image encoded with CIF source format is 352 pels by 288 lines). `e_pels` and `e_lines` indicate the width and height of the memory space used for reconstruction of the image. The parameters are primarily used for configuring video output display.

`freeze_request` and `freeze_release` are parameters used with Annex L.

`QI` is the QUANT value for I-frames.

`QP` is the QUANT value for P-frames.

`P_rate` is the number of sequential P-frames to encode.

`pcount` is a counter for the number of frames that have been encoded.

`PTYPE` is the type of frame encoded. The value is 0 for I-frame, 1 for P-frame.

`pVlxBin` is a pointer to memory that will contain VLx code. This is used for context switching.

To support context switching, the encoder application manages persistent memory used by the library. This is accomplished by means of manipulation by the application of the following pointer elements of the `H263VencStatus` structure:

```
frameToQueueForDisplay
frameToDecodeIntoNext
null_input
buffersForDS
dcacheMemory
pVlxBin
```

These pointers should not be considered for direct use by the application designer. Instead, the top-level API call `VencStatusEncodeOpen` should be used by the application designer for manipulation of memory allocation. The application designer is free to use the following pointer elements of the `H263VencStatus` structure when allocating memory:

```
frameToCapture
frameToEncode
output_buf
```

`VencStatusEncodeOpen` also allocates memory for these pointers.

```
void VencStatusEncodeOpen(H263VencStatus *h263VencStatus,
  int targetBits,
  int QI,
  int QP,
  int P_rate,
  int firstTime
)
```

`firstTime` is used for context switching. If the VLx has not been initialized by another encoder or decoder, `firstTime` should be set to the value 1; else, it should be set to 0.

The other parameters for this call are as described above, in the discussion of `h263VencStatus`.

The call

1. allocates all memory for the library,
2. sets up the DataStremer DMA controller,
3. sets up the VLx GetBits engine,
4. loads and starts the VLx program, and/or
5. handles proper initialization in context switching mode.

```
H263EncodeFrame(H263VencStatus *h263VencStatus)
```

This call

1. encodes an I or P frame based upon the configuration set by `VencStatusEncodeOpen` and
2. flushes the GetBits output buffer.

```
H263EncodeFrameTask(H263VencStatus *h263VencStatus)
```

This call is the context switching version of `H263EncodeFrame`. This call

1. encodes an I or P frame based upon the configuration set by `VencStatusEncodeOpen` and
2. flushes the GetBits output buffer.

```
VencStatusPostEncoding(H263VencStatus *h263VencStatus)
```

This call should always follow an encode frame (with or without context switching). This call rotates the input and output frame buffers.

```
H263VideoEncodeClose(H263VencStatus *h263VencStatus)
```

This call

1. flushes the GetBits output buffer and
2. stops the VLx and closes the DataStreamer DMA controller buffer channels.

Ordering information

To order the MAP-CA DSP H.263+ Encoder Media Library Product Source Code Software Package, use part number 098-0201-20.

For more information, contact your nearest Equator sales 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