Texas Instruments

H.261 Implementation on the TMS320C80 DSP

Application Report

1997

Digital Signal Processing Solutions

Texas Instruments

Printed in U.S.A., June 1997

SPRA161

H.261 Implementation on the TMS320C80 DSP

Application Report

SPRA161
June 1997

TEXAS INSTRUMENTS

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Figures

H.261 Implementation on the TMS320C80 DSP

ABSTRACT

This report describes the coding requirements, techniques, and decisions which must be made to utilize the TMS320C80 DSP as an integrated services digital network (ISDN) video-system manager and provides an overview on how the processor handles video signal in the ISDN narrow-band format in conformance with the International Telecommunications Union (ITU)–T H.261 Recommendation.

1 Introduction

The development of video encoding/decoding transmission standards by the International Telecommunications Union (ITU) has resulted in a series of recommendations which attempt to specify practices and protocols for various service types.

For video-signal management using narrow-band ISDN service, this has led to development of the H.261 Recommendation. The H.261 Recommendation, "*Video Codec For Audiovisual Services at $p \times 64$ kbits/s*" supports and defines coder/decoder (codec) protocols for transmission bit rates of up to P $\times$ 64 kbps, where P is between 1 and 30, which results in a maximum throughput of up to 1.92 Mbps.

H.261 precedes the Joint Photographic Experts Group (JPEG) and Motion Picture Experts Group (MPEG) formats and specifically defines only the signal decoder mechanism. By necessity, however, all encoder schemes must be compatible with any decoder used to process the coded data streams. Therefore, the P $\times$ 64 Kbps standard is actually a series of recommendations which describe transmission items such as:

- H.221 frame structure
- H.230 frame synchronous control
- H.242 communications between audio-visual terminals
- H.320 systems terminal equipment
- H.261 video codec systems

Both H.261 and JPEG codecs use discrete cosine transform (DCT) and variable length codes (VLC) techniques. JPEG processes incoming picture frames independently, using intraframe DCT while the H.261 recommended using a block-based, motion-compensating scheme.

Similar to other video-encoding/-decoding methods, H.261 uses picture data in previous frames to predict the image blocks in the current frame. Therefore, only differences of a small magnitude between the displaced previous block and the current block are transmitted rather than entire picture blocks as in the JPEG standards.

Several characteristics and design considerations are relevant to the H.261:

1. H.261 defines only the decoder. Encoders, which are not explicitly specified by the standard, are expected to be compatible with any well-defined decoder.
2. H.261 is designed for real-time communications and to reduce encoding delays uses the closest previous frame for motion-picture-sequence coding.
3. H.261 tries to balance the hardware complexities between the encoder and the decoder since they are both needed for real-time videophone applications. Other coding schemes, such as vector quantization (VQ) may have a rather simple decoder but must have a more complex encoder.
4. H.261 compromises between coding performance, real-time requirements, implementation complexities, and system robustness. Motion-compensated DCT coding is a mature standard.
5. The H.261 final coding structures and parameters are tuned more toward low bit-rate data transmission applications. Selection of coding structures and coding parameters is more critical to codec performance at very low bit-rates. At low bit-rates, data is transmitted at a slower pace and any discrepancies in reception are more able to disrupt reception of data. At higher bit rates less-than-optimal parameter values do not affect CODEC performance as much.

This report provides basic information on how the H.261 is implemented, and explains how the TMS320C80 implementation is accomplished. While current implementations conform with the H.261 standard, many parameters that affect the quality of the picture are not defined by the H.261 recommendations. This report gives some of the encoding/decoding details so designers can understand and enhance the code to best fit specific applications.

2 Basic Overview of the H.261 Recommendation

H.261 specifies a set of protocols that every compressed-video bitstream must follow and a set of operations that every standard, compatible decoder must be able to perform. The actual hardware codec implementation and the encoder structure can vary greatly from one design to another. The data structure of the encoder/decoder and the requirements of the video bitstream also are described. The video bitstream contains the picture layer, group-of-blocks layer, macroblock layer, and the block layer (with the highest layer having its own header, followed by a number of lower layers).

- ***Picture size:*** The only two picture formats that are allowed by the H.261 at the present time are the common-intermediate format (CIF) and quarter-common-intermediate format (QCIF). The CIF picture size is 352 pixels (pels) per line by 288 lines while the QCIF is 176 pels per line by 144 lines. The QCIF picture size is half as wide and half as tall as the CIF picture.
- ***Color-space:*** The 4:1:1 format is used. The picture color is made of three components: the luminance signal Y and the color-difference information signals C_R and C_B. The C_R signal and the C_B signal are each subsampled at half the rate of the Y-signal in both the horizontal and vertical direction. For every $2 \times 2 = 4$ Y samples, there is one sample of each for C_R and C_B. The bit size of each Y, C_R, and C_B sample is 8.
- ***Picture Hierarchy****:* Picture frames are partitioned into 8 line by 8 pel image blocks square. Macroblocks (MB) are made of four Y blocks, one C_R block, and one C_B block at the same location as shown in Figure 1. A group of blocks (GOB) is made of 33 MBs. Figure 1 shows these relationships while Figure 2 shows how the video bitstream is separated into different layers .

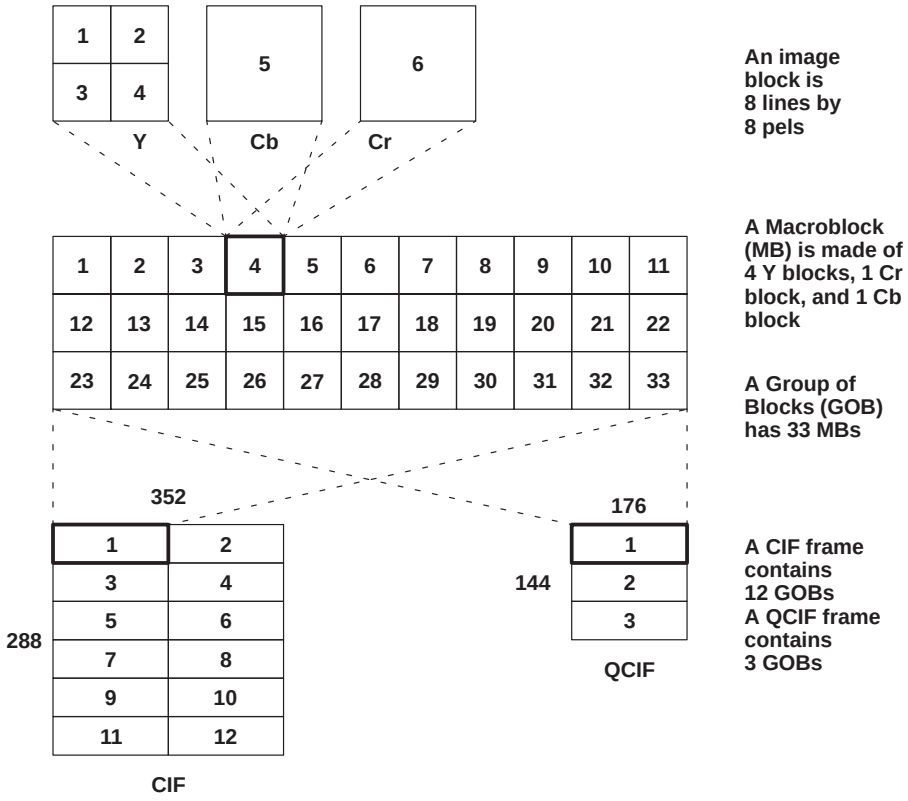

Figure 1. Recommendation H.261 Image Format and Hierarchy

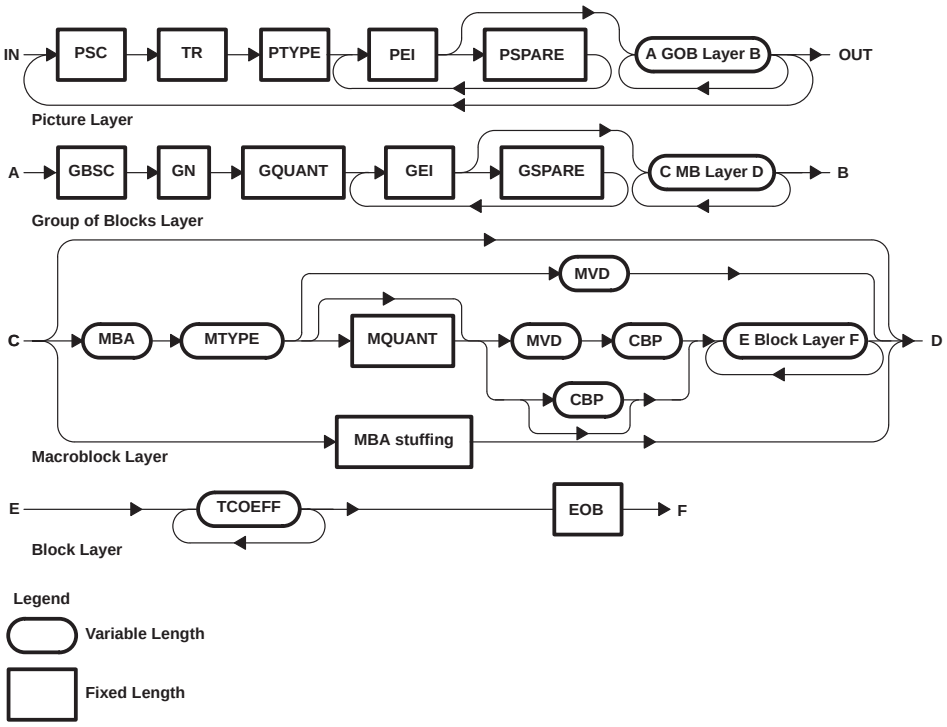

Figure 2. Video Multiplex Coder Syntax Diagram

- Picture layer: Data for each picture consists of a picture header followed by data for a GOB. The data stream is a compressed videostream which contains:
  - Picture start code (PSC), which is a fixed 20-bit pattern
  - Temporal reference (TR), which is a 5-bit input-frame number, which can have 32 possible values that indicate the number of dropped frames.
  - Type information, (PTYPE) which is a 6-bit field described as:
    - bit 1 – Split-screen indicator, defined as 0 for off, 1 for on
    - bit 2 – Document-camera indicator, defined as 0 for off, 1 for on
    - bit 3 – Freeze-picture release, defined as 0 for off, 1 for on
    - bit 4 – Source format, defined as 0 for QCIF, 1 for CIF
    - bit 5 – Optional still-image mode HI_RES, defined as 0 for on, 1 for off
    - bit 6 – Spare
  - Optional spare field (PEI). If set to 1 indicates that a 9-bit value appears in the PSPARE field, if set to 0, no data follows in the PSPARE field.

  - Spare-Information field (PSPARE). Currently encoders must not insert PSPARE until specified by ITU and must be designed to discard PSPARE if PEI is set to 1. This allows ITU to specify backward-compatible additions to PSPARE.

- GOB layer: Each picture is divided into GOBs, each of which is one-twelfth of the CIF or one-third of the QCIF picture area. A GOB relates to 176 pels by 48 lines of Y and the spatially corresponding 88 pels by 24 lines for each C_B and C_R.
- A GOB header contains the following:
  - Group of blocks start code (GBSC), a 16-bit pattern
  - Group number (GN), a 4-bit GOB address
  - Quantizer (GQUANT) information such as the initial-step size normalized to the range 1 to 31. At the start of a GOB, the quantization value QUANT is set to GQUANT.
  - Optional spare field (GEI). If set to 1 indicates the presence of a following data field designated as GSPARE.
  - A spare information field (GSPARE). Currently encoders must not insert GSPARE until specified by the ITU and must be designed to discard GSPARE if GEI is set to 1. This allows the ITU to specify backware-compatibie additions to GSPARE.
- Macroblock (MB) layer: Each GOB layer is divided into 33 macroblocks (see Figure 1). A macroblock relates to 16 pels by 16 lines of Y and the spatially corresponding 8 pels by 8 lines for each C_R and C_B.
  - A variable-length codeword macroblock address (MBA) indicating the position of a macroblock within a group of blocks. The first block transmitted in MBA is the absolute address and, for subsequent macroblocks, the MBA is the difference between the absolute address of the current macroblock and the last transmitted macroblock. When macroblocks are skipped, the value for the MBA equals one plus the number of skipped macroblocks preceding the current macroblock in the GOB. Macroblocks that contain no information are not transmitted.
  - Macroblock type (MTYPE). Variable-length codewords giving information about the macroblock and which data elements are present. These elements are of the following type: intra-, inter-, inter- with motion compensation (MC), or inter- with MC and a filter. There are ten types in total (see Table 1).

  - Quantizer (MQUANT). The normalized quantizer step size is used until the next MQUANT or GQUANT. If MQUANT is received, the quantization value QUANT is set to MQUANT. The value is from 1 to 31.
  - Motion vector data (MVD), is differential-displacement vector data. MVD is included in all macroblocks and is obtained from the macroblock vector by subtracting the vector of the previous macroblock. For this calculation the vector of the preceding macroblock is considered to be 0 in the following three situations:
    - Evaluating MVD data for the macroblocks 1, 12, and 23
    - Evaluating MVD for macroblocks in which MBA does not represent a difference of 1
    - MTYPE of the previous macroblock was not motion compensated
    - Limited to ± 15 Y pels for both the horizontal and vertical components.
    - Only one MVD per macroblock is indicated by MTYPE.

- Coded block pattern (CBP) is a variable length field that is present if indicated by MTYPE. The codeword gives a pattern number signifying those blocks in the macroblock for which at least one transform coefficient is transmitted. The pattern number for the CBP is given by:

  $$32 \bullet P_1 + 16 \bullet P_2 + 8 \bullet P_3 + 4 \bullet P_4 + 2 \bullet P_5 + P_6$$

  where $P_n = 1$ if any coefficient is present for block *n*, else 0.
- Block layer. A macroblock is composed of four luminance blocks and one each of the two color difference blocks. Data for a block consists of codewords for transform coefficients followed by an end-of-block (EOB) marker. The order of transmission is the four Y luminance data items followed by block 5, the C_B value, and block 6 which is the C_R value.
  - Transform coefficients (TCOEFFs) consist of quantized-transform coefficients, followed by the EOB symbol. Transform-coefficient data is always present for all six blocks in a macroblock when MTYPE indicates INTRA. In other cases, the MTYPE and the CBP signal which blocks have coefficient data transmitted for them. The quantized-transform coefficients are transmitted sequentially according to the sequence set in the standard.
  - The most commonly occurring combinations of successive zeros (RUN) and the following value (LEVEL) are encoded with variable length codes. Other combinations of (RUN, LEVEL) are encoded with a 20-bit word consisting of 6-bit ESCAPE, 6-bit RUN, and 8-bit LEVEL. For the variable-length encoding scheme, there are two

code tables. The first is used to transmit LEVEL in INTER, INTER+MC, and INTER+MC+FIL, and the second for all other LEVELs except the first one in the INTRA blocks, which is a fixed-length code of 8 bits.

2.1 Transform, Quantization, and Run-Length Encoding

The Y-, C_R-, and C_B- sampled signals are each represented by 8 bits (1 to 254). The value to be transformed is represented by 9 bits (–256 to +255) because, during inter-frame coding, negative values can be generated. During the encoding phase, if the transform coefficient (TCOEFF) is sent for the picture block, the 8 x 8 block that contains the 9-bit values processed by a two-dimensional DCT, which generates an 8 x 8 transformed coefficient. The coefficient ranges in value from –2048 to 0 and 0 to +2047. For the intra-discrete-cosine transformed coefficient (the one at the upper left corner during Intra-frame coding), the range is from 0 to +2047.

These values are passed to the quantizer, which generates 8 x 8 values between –128 to+127, called LEVELs. The intra-DC coefficient is linearly quantized with a fixed-step size of 8. All other coefficients are quantized based on a QUANT value, which changes from macroblock to macroblock.

The QUANT value can be set by GQUANT or MQUANT. The quantizer reduces the precision of the data sample and includes a “dead zone” close to zero, which forces most small coefficients to zero.

After the quantization process, most high-spatial-frequency coefficients are zero. Therefore, when the data is being zig-zag scanned in the order as shown in Figure 3, most of the non-zero runs are concentrated at the beginning. The number of successive zeroes between two non-zero coefficients is called a RUN. The Huffman-coding scheme uses a shorter code to represent more likely occurring combinations of the RUN and LEVEL pair, but unlike the JPEG coding, the Huffman-code table is fixed. The Huffman-coded pairs are called variable-length codes. The other less likely occurring pairs are coded with a 20-bit fixed-length codes.

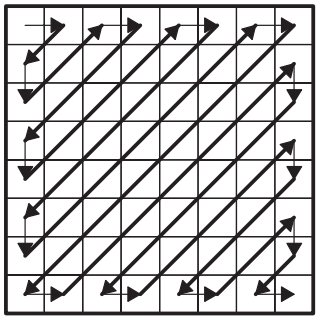

Figure 3. H.261 Recommended Zig-Zag Scanning Procedures

3 Applying the H.261 Recommendations

It is important to understand that the H.261 is an evolving standard. When this standard was adopted in 1985, the technology advances currently being implemented were anticipated but not entirely implemented by the standard.

3.1 What is Not Specified in the Recommendation

The H.261 essentially defines only the signal decoder. The signal encoder is not completely specified by the H.261 standard but is expected to be compatible with the decoder. This encoder allows various implementations to be available on the market, from low-end and low-cost implementations to high-performance and high-cost implementations, but at the same time, also allows these implementations to be compatible.

Frame rate is defined by the National Television Systems Committee (NTSC) at 29.97 frames per second (fps). The actual frame rate can be less, since an encoder is allowed to drop frames. This usually happens if the encoded bitstream is not sent out fast enough. When frames are dropped, the temporal reference (TR) value indicates how many frames have been dropped. The criteria used to determine when to drop frames are not defined in the standard.

Encoder and decoder buffers are used to delay the bitstream. The size of buffers affects the amount of transmission delay. Exactly how to implement the encoder and decoder buffers items such as buffer sizes, buffer thresholds, and either fixed or adaptive thresholds, is not defined in the standard.

H.261 does not define how to select coding mode. Coding mode decisions is not defined in the standard. There are 10 macroblock types (MTYPEs) defined by the recommendation, but how to decide *which* MTYPE to be used is *not* defined. For example, choosing between inter-frame and intra-frame coding and choosing which type and usage criteria of loop filters, and whether motion compensation detection should be used or not, respectively, are undefined by the recommendation.

The H.261 recommendation also does not define the quantization value (MQUANT or GQUANT) or how a specific quantizer value affects the number of bits sent per frame. For larger quantization values, more coefficients are zeroed, resulting in fewer RUN and LEVEL pairs being sent. An effective encoder adaptively adjusts the quantization values based on the image content and available channel bandwidth.

H.261 also does not specify how motion vectors are to be obtained. So, if motion compensation is used, the choices of which displacement-estimating algorithm to use is left open to the designer. Using block matching is a popular scheme, but many other block-motion-estimation algorithms exist. Good motion estimation algorithms require a large amount of processor power so the algorithm must be chosen carefully.

A common component of a video information signal is noise. Signal pre-processing is frequently used to reduce coding noise as a component of the video information. Post-processing of the signal may further reduce the induced artifacts, such as blockiness, that can be inadvertently be introduced during data compression. H.261 does not specify the type or amount of pre- and post-processing required. These items are usually accomplished by using various types of spatial and temporal filters. The encoder can be designed to adjust the filter parameters adaptively, based on the available channel bandwidth.

4 Implementation on the TMS320C80 Processor

H.261 can use any of ten major types of coding modes shown in Table 1. Another major concern in processing video information is the motion estimation algorithm selected. This aspect of the video-processing task is highly time consuming, so selection of the algorithm is critical to ensure maximum efficiency and throughput. These activities are controlled by the master processor and four parallel processors on-board the TMS320C80 chip.

4.1 Major Types of Coding Modes

The video bitstream contains both the picture and picture quality data values which are macroblock type, motion-vector data, quantizer, and the transform coefficients. Out of all these video bitstream information values, those that actually contain the picture data and ultimately affect the picture quality are MTYPE, MVD, QUANT, and TCOEFF. Ten types of coded macroblocks are possible as indicated by the value MTYPE; however, there are only three major types. Table 1 shows how these three major types are implemented:

- Intra-frame coded where only the original pixels are transform-coded.
- Inter-frame coded with motion vector only. The motion vector is sent and the decoder uses the last reconstructed frame and the received motion vector to rebuild the new MB.
- Inter-frame coded with motion vector and coded differences. The decoder uses the previously reconstructed frame and the received motion vector and also uses the received transform-coded pixel differences to rebuild a new macroblock.

Table 1. TMS320C80 Implementation Of Different MTYPEs

PREDICTION	MQUANT	MVD	CBP	TCOEFF	TMS320C80 IMPLEMENTATION
Intra–				x	Intra–
Intra–	x			x	Intra–
Inter–			x	x	Inter– w/ coded diff (MV = 0)
Inter–	x		x	x	Inter– w/ coded diff (MV = 0)
Inter + MC		x			Inter–MV only
Inter + MC		x	x	x	Inter–w/ coded diff
Inter + MC	x	x	x	x	Inter– w/ coded diff
Inter + MC + FIL		x			Inter– MV only
Inter + MC + FIL		x	x	x	Inter– w/ coded diff & filter
Inter + MC + FIL	x	x	x	x	Inter– w/ coded diff & filter

Figure 4 shows the data-flow diagram for TMS320C80 encode/decode. During intra-frame coding, the block is discrete cosine transformed, quantized, zig-zag scanned, and run-length encoded. The encoded bitstream and coding-mode decisions are sent to the buffer. From the H.261, the encoder contains an inverse quantizer and inverse discrete cosine transform (IDCT) function to reconstruct a frame if motion compensation is to be done on the next frame. The decoder does an inverse quantization and IDCT to generate the picture.

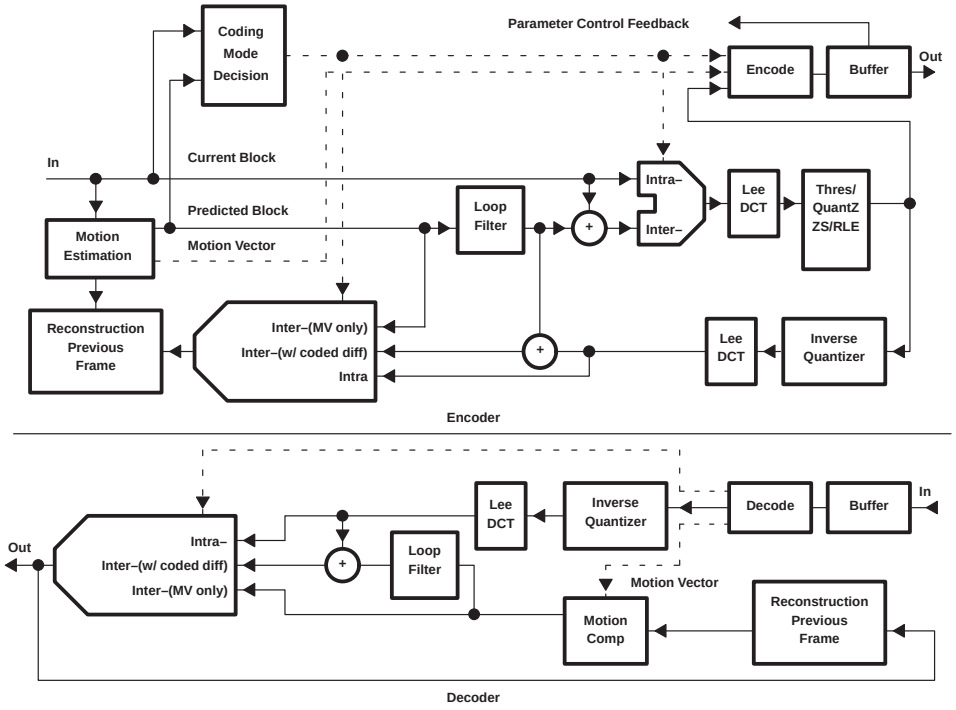

Figure 4. TMS320C80 H.261 Recommendation Compression and Decompression Codec Used in the TMS320C80

During inter-frame coding using the motion vector only, no transformed coefficients are sent and only a motion vector is sent. The picture-block-to-picture-block differentiation is represented by a motion vector. The motion estimator uses the previous reconstructed frame to compare with the current block to determine the motion vector. The motion estimator also applies the motion vector to the previous reconstructed frame to regenerate a predicted block which represents the reconstructed frame. The decoder then uses the motion vector to motion-compensate the previously reconstructed frame to generate the picture. This is similar to the predictor used in pulse-coded-modulation speech coding where the motion estimator in the encoder uses the motion vector to build the reconstructed frame rather than just copying the present frame to the reconstructed frame buffer to simulate the performance of the decoder. The encoder is doing the same thing as the decoder and any accumulated errors in the decoder are detected and are present in the encoder. This ensures that the decoder image has low distortion.

During inter-frame coding using motion vectors and coded differences, differences between the current block and a predicted block are discrete-cosine transformed, quantized, zig-zag scanned, and run-length encoded. Together with the motion vector, the differences are sent to the buffer. The decoder uses the motion vector to motion compensate for the predicted block and adds the result to the inverse quantized, IDCT coefficients to regenerate the picture. If MTYPE specifies no motion vector, then the motion vector is regarded as having a zero displacement. At the same time, the encoder is rebuilding the reconstructed present frame using the same scheme as the decoder.

A loop filter keeps the differences between a motion-compensated picture block and the current block small. Any coded differences are smaller and less information is required to be sent, reducing the transmission bit rate.

4.2 Coding Mode Decisions in the TMS320C80

The TMS320C80 video processor selects which coding mode to use per block by evaluating several parameters. First, any sampled value variations that occur within a block are measured and the sum of absolute differences (SAD) between the block average and the individual samples is evaluated. This produces the value INTRA_SAD. The encoder evaluates the SAD of the present block and the previously analyzed block to create the INTER_SAD value. A coding mode decision is made by a process called STEPA, where it is determined if any motion compensation is required and whether or not motion compensation data is to be sent.

A threshold value is compared with the SAD00 value to establish the INTER_SAD at zero-motion vector. With the INTER_SAD value less than the threshold value, no motion compensation is required.

STEPA first computes the INTER_SAD value for a motion-vector value of zero to determine if any motion estimation is necessary. If motion is required to be performed, STEPA also determines whether motion-compensated data needs to be sent. If the sum of absolute difference is still too high, it computes the INTRA_SAD and determines whether intra-frame coding is used. The actual coding uses a value which represents the mean absolute difference (MAD).

- INTRA_SAD
  - A measure of the variation of sample values within a block
  - Sum of absolute differences between the block average and the individual samples in the block
- INTER_SAD
  - A measure of differences between the current block and the previous block
  - Sum of absolute difference between the current block samples and the block samples from the previous reconstructed frame
- SAD00
  - INTER_SAD with zero displacement
- SATMC
  - INTER_SAD with a non-zero motion vector

Figure 5 is a pseudocode example that demonstrates how to decide on which coding mode to use. This psuedocode gives some basic insight into the decision-making process. The actual code is much more complex.

```
/* If the INTER_SAD with no MV is already below a threshold */
If SAD00 < 100
          /* No motion estimation task is necessary */
          /* Block will most likely not be coded at all */
Else
          /* Need to do some motion estimation */
/* If motion estimation is to be performed, check if the motion
    compensated INTER_SAD is quite so much less than the INTER_SAD with
    no motion compensation */
If SATMC + 100 < SAT00
          /* Motion compensated data need to be send */
Else
          /* No motion compensation required*/
/* If the INTRA_SAD is so much less than the selected INTER_SAD */
INTER_SAD = SATMC or SAT00
If INTRA_SAD + 500 < INTER_SAD
         /* Use Intra-Frame Coding */
Else
         /* Use Inter Frame Coding */
```

Figure 5. TMS320C80 H.261 Coding Mode Decision

4.3 Motion Estimation

H.261 motion estimation is one of the most time-consuming tasks. Current software releases support two different motion-estimation search algorithms, which result in a one-at-a-time search and a three-step search. Figure 5 demonstrates the actions of these two search algorithms as well as an exhaustive search algorithm and an XY search algorithm. In either case, the sum of the absolute differences between the current block and the displaced previously reconstructed block is used as a merit factor.

The three-step search searches the origin and displacements of ±4 first to find the best general area. It then refines the search around a new origin by searching displacements of ±2. The final step searches displacements of ±1. The current software release limits the maximum displacement to ±7 instead of ±15 as specified in H.261.

The one-at-a-time search uses either the origin or the displacement from the previous MB as the origin of the search. It then refines the search by searching displacements of ±1 until all neighboring blocks show a higher SAD. The current software release limits the maximum number of searches to 25.

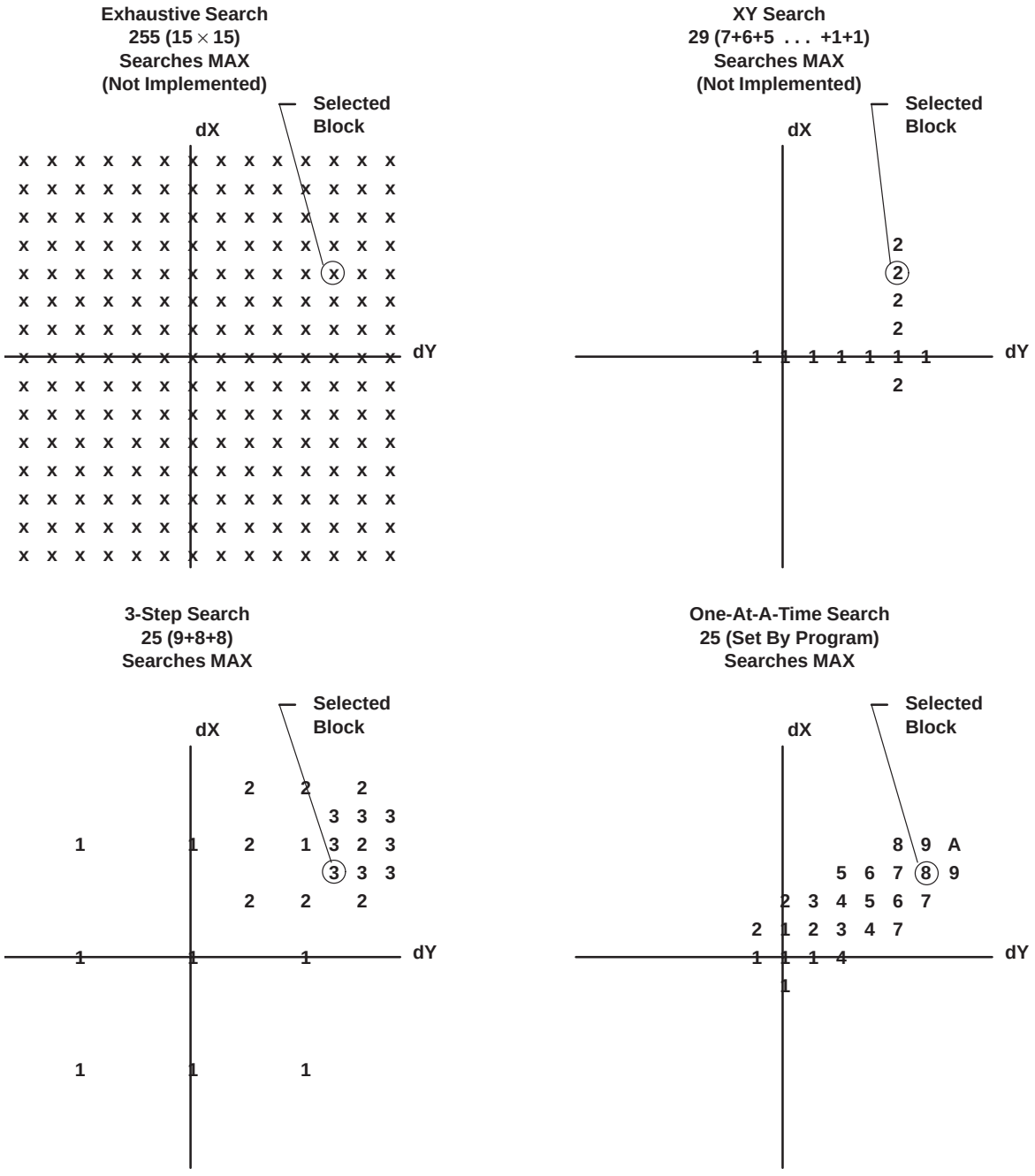

Figure 6. TMS320C80 H.261 Recommended Motion-Estimation-Search Algorithms

4.4 Bit-Rate Control

The encoder uses adaptive quantization and frame dropping to control the bit rate generated from the encoded picture frame.

4.5 Adaptive Quantization

The QUANT value is incremented or decremented based on how many bits are generated from the previous frame. As the QUANT value is increased, more LEVELs (digitally-quantized-transform coefficients) become zero and the number of bits generated should be less. If QUANT is decreased, then more non-zero LEVELs are generated and the number of bits generated should be more and the picture quality should improve.

4.6 Frame Dropping

If the number of bits not transmitted reaches a threshold and data is about to overflow the buffer, then it is necessary to drop a frame.

4.7 Multitasking on the TMS320C80

The TMS320C80 has one master processor (MP), four parallel processors (PP0 to PP3), a transfer controller (TC), and a video controller (VC). The MP and PP3 are used to implement various other tasks required by the H.320 recommendation and other recommendations such as H.221, H.242, and G.728. The remaining three parallel processors, PP0 – PP2, are used solely for the H.261 video encoder/decoder implementation. PP0, PP1, and PP2 are called PP_BLOCK0. Each communicates with one another and with the master processor via the multitasking executive and command buffer interface.

During the encoding, STEPA makes coding-mode decisions and calculates the INTRA_SAD for each MB. In STEPA, all three PPs of PP_BLOCK0 work in parallel to perform the motion-estimation task. In STEPB, the three PPs perform different tasks. PP0 performs the loop filtering, image difference, and DCT. PP1 performs the thresholding, quantization, zig-zag scanning, and inverse quantization. PP2 performs the IDCT and block reconstruction.

The decoding phase has one step and the three PPs perform different tasks. PP0 performs bit-stream parsing and acts as the client for the other two server PPs, PP1, and PP2. The server PPs perform the rest of the decoding tasks such as IDCT, motion compensation, and loop filtering.

5 Using the H.261 Code

The current H.320 code was intended for the VisionPoint project which has been terminated. There are only a few interface functions on which to run H.261 code, which is located in the H320\H261 directory. The \H320\SHARE and H320\UTIL directories also must be included because there are utility functions to be used by the H.261 code. When using the code on the software development board (SDB), keep the \H320\DRV directory to allow an H.261 loopback demo on the SDB using a camera and a display.

5.1 Initialization Code

The steps necessary to exercise the H.261 recommended code are as follows:

1. Initialize the H.261 buffer functions:

```
BufferInit();
BufferInstallMalloc (MemAlloc,MemFree);
```

2. Initialize the H.261 FEC:

```
H261FecInit();
```

3. Create the encoder and decoder task

```
taskIdH261Enc = TaskCreate(TASKID_H261_ENC,H261_Encoder,NULL,7,4096);
taskIdH261Dec = TaskCreate(TASKID_H261_DEC,H261_Decoder,NULL,8,4096);
```

4. Create a timer function

   The encoder and decoder tasks directly interface with the video capture and display drivers. The tasks can be started by a timer at every frame, so a timer function must be created:

```
taskIdTimer = TaskCreate(–1,TimeMgr,NULL,18,4096);
```

5. Resume tasks

   These tasks must be resumed using the TaskResume functions. There are just two major functions that get the bitstream from the H.261 encoder and put the bitstream to the H.261 decoder.

```
H261FecGetEncodedBuffer (bitrate);
/* Get encoded bitstream from encoder */

H261FecDecodeBuffer (dbuffer, bitrate);
/* Decode the dbuffer bitstream */
```

5.2 Loopback Program

The following shows how a program does a loopback by reading an encoded buffer and placing the buffer value in the decoder buffer. Much of the programming detail is omitted in this sample to simplify the concept.

```
for(;;) {
   TaskWaitSema (h221TxSemaId); /* Wait for timer signal */
      if ((dbuffer = H261FecDecGetEmptyBuffer()) != NULL) {
      ebuffer = H261FecGetEncodedBuffer (bitrate);
      memcpy(dbuffer,ebuffer,(bitrate+7)>>3);
      H261FecEncReclaimBuffer(ebuffer);
      H261FecDecodeBuffer(dbuffer,bitrate);
   }
}
```

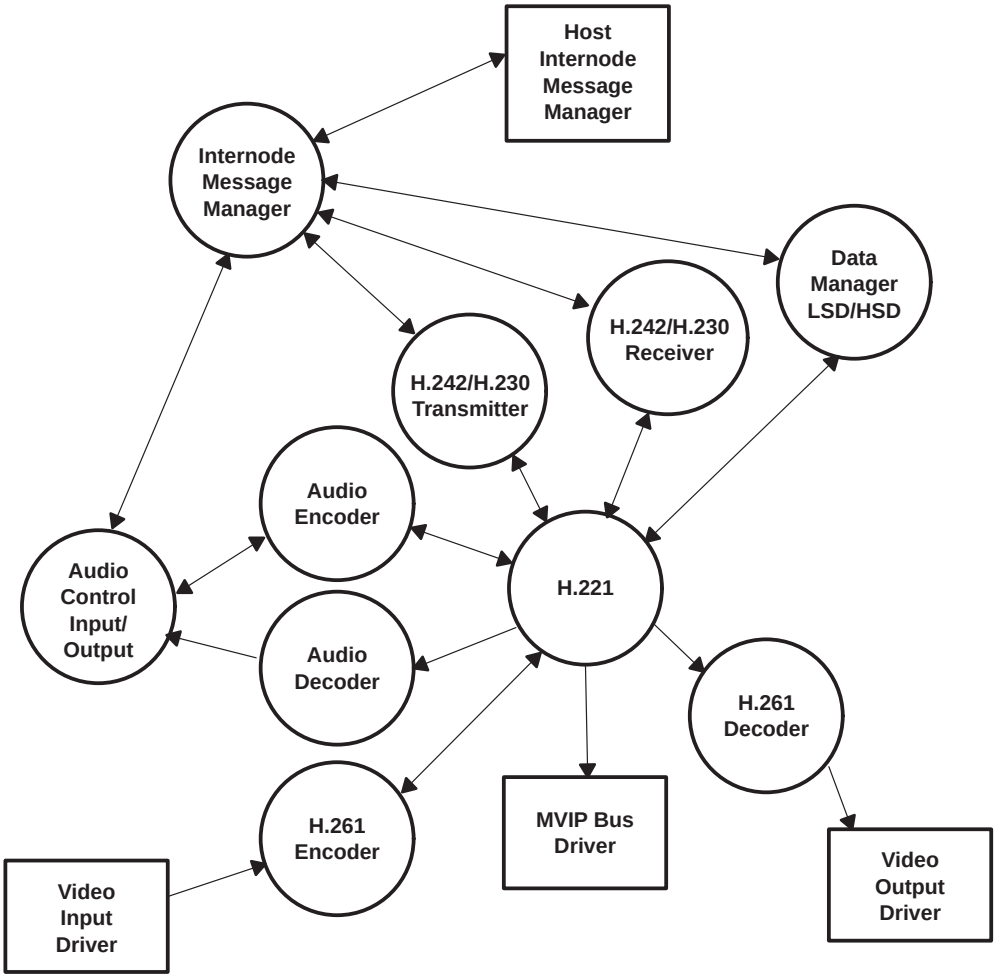

Figure 7. TMS320C80 H.261 Tasking Model

6 Conclusion

With a 40-MHz TMS320C80, CIF resolution, and data transmission at a frame rate of 30 fps, the loading of the PPs is almost up to 100 percent in a typical video-conferencing session. The encoder loads about 60 percent while the decoder loads about 30 percent of all the PPs in PP_BLOCK0. With a 60-MHz TMS320C82 coming out soon, Texas Instruments (TI™)[1] is planning to implement the H.261 recommendation on the TMS320C82.

H.261 does not explicitly specify a standard encoder but many basic operational elements are strongly constrained by it. Most other crucial elements are still open to manipulation by the design engineer. A few examples are:

- The coding-mode decision
- Motion-estimation algorithms
- Pre- and post-processing
- Quantization
- Frame dropping
- Encoder-and-decoder buffer sizes
- Loop-filtering methods; etc.

Improvements can be made to current software design of the video encoder/decoder are improvements to the motion estimation and adaptive quantization algorithms. In fact, TI's H.263 implementation on the C82 has an improved motion-estimation routine that reduces bit rates of typical videoconferencing session by half while essentially maintaining the same picture quality. A new rate-control algorithm has been developed.[4]

With various video encoder/decoder software implementations, designers of videophone systems can progressively improve encoder/decoder performance without significant additional future major hardware redesigns.

1. TI is a trademark of Texas Instruments Incorporated.

7 References

1. ITU-T Recommendation H.261 (1993): “Video Codec for Audiovisual Services at Px64 kbits”.
2. Jeremiah Golston, “TMS320C80 H.320 Software User’s Guide”, Release 1.1, Texas Instruments, Oct 1995.
3. Arun N. Netravali and Barry G. Haskell, “Digital Pictures: Representation, Compression, and Standards”, Second Edition, AT&T Bell Laboratories, 1995.
4. Jennifer L. H. Webb, “Rate Control for H.261 Video Coding Through Quantization Step Size Update and Selective Coding of Coefficients”, Technical Activity Report, Texas Instruments, June 1996.

Appendix A Glossary

address	Program code memory location or data-storage location
ANSI	American National Standards Institute
ANSI C	A version of the C programming language
BRI	Basic-rate service on ISDN
buffer	An intermediate storage space
CBP	Coded block pattern
CD	Coded differences
CIF	Common intermediate format (352 pixels × 288 lines)
CODEC	Coder/Decoder or Compression/Decompression
DCT	Discrete-cosine transform
DSP	Digital signal processor
EOB	End of block
FLC	Fixed-length code
fps	Frames per second (fps)
GBSC	Group of blocks start code
GN	Group number
GOB	Group of blocks
GQUANT	A 5-bit quantizer
IB	Image block (8 pixels x 8 lines)
IDCT	Inverse direct cosine transform
interframe coding w/motion vector (MV)	Only the motion vector is transmitted
interframe w/MV and CD	Uses previously reconstructed frames and MV and CD
intraframe coding	Only the original pixels are transformed
ISDN	Integrated services digital network
JPEG	Joint Photographic Experts Group format
Loop Filter	Keeps signal levels minimized
MAD	Mean absolute difference
MB	Macroblock
MBA	Macroblock address
MP	Master processor
MPEG	Motion Picture Experts Group format
MQUANT	5-bit quantizer
MTYPE	Macroblock type
MV	Motion vector

MVD	Motion-vector data
NTSC	National Television Systems Committee Standard
PAL	Phase alternating line
PCM	Pulse-coded modulation
PP	Parallel processor
PSC	Picture-start code
QCIF	Quarter CIF format (176 pixels by 144 lines)
RUN	Number of zeroes between two non-zero coefficients
SAD	Sum absolute differences
SDB	Software development board
TC	Transfer controller
TCOEFF	Transform coefficients
TR	Temporal reference
VC	Vector quantization
VC	Video controller
VLC	Variable-length coding
VQ	Vector quantization