Application Report
SPRAB83–December 2009

Smart Codec Features in TMS320DM365

Naveen Srinivasamurthy, Mahant Siddaramanna and Ritesh Rajore *MMCodecs*

ABSTRACT

You will significantly enhance video encoder quality by incorporating external input/feedback from the system. Additionally, the robustness of the video codec will also be vastly improved. The video codec can utilize such information to react quickly and adapt to changing dynamics of the input video signal. This greatly enhances the end-user experience as the video encoder can prioritize the more important spatial and temporal parts of the video signal. This application report presents features that have been incorporated in the TMS320DM365 H.264 codec, which makes use of external input/feedback from the system to improve the overall end-user video quality. Specifically, face information is obtained from the DM365 video processing front end (VPFE) and these regions are represented with higher fidelity. Scene change information is detected using a low-resolution encoder, which is used by a high-resolution encoder to encode the scene change frames as I-slices.

Contents

List of Figures

1 Introduction

Excellent video quality is currently achievable on the DM365 device for a wide variety of resolutions. This has been achieved by a very efficient codec implementation, that takes the device capabilities into account. Within the device constraints, the video encoder efficiently implements the correct video codec tools that provide maximal quality gains.

However, there are application scenarios that can be further exploited to significantly improve the overall video quality while remaining within the device capabilities. One such scenario is video conferencing. It is well known that, perceptually, the face regions are the most important when compared to background regions. The face detection capability in the DM365 VPFE can be used to detect faces; the video encoder can make use of this information to represent the face regions with higher fidelity (i.e., introduce less distortion).

Scene changes introduce special challenges to the encoder. Without a priori knowledge about a scene change, the encoder will encode the frame as a P-slice. Here, most of the macroblocks (MBs) are encoded as intra MBs. However, in the DM365, the number of intra prediction estimation (IPE) modes used during I-slice encoding is significantly more than that used during P-slice encoding. Therefore, a priori knowledge of scene changes is beneficial, as then the scene change frame(s) can be encoded more efficiently as an I-slice utilizing more IPE modes, greatly improving the perceptual quality of the video frame.

2 Smart Codec Features

This section describes the different smart codec features currently deployed in the DM365 H.264 video encoder in detail.

2.1 Region of Interest (ROI) and Region of Non-Interest (RONI) Coding

There exist many applications where particular spatial and temporal regions of the video signal are of more interest to you than other areas. Examples are:

- In video conferencing applications, the viewer pays more attention to the face regions when compared to other regions
- In security applications, areas of potential activity (e.g., doors, windows) are more important

These important regions or the regions where the viewer pays more attention, are called regions of interest (ROI). In such scenarios it is important that the ROI areas are reproduced as reliable as possible since they contribute significantly towards the overall quality and perception of the video. When encoding a video signal, the video encoder prioritizes the ROI areas and encodes them at higher fidelity when compared to non-ROI areas. This is achieved by assigning higher number of bits to the ROI areas when compared to non-ROI areas. Similarly, if regions of non-interest (RONI) are known, bits can be stolen from the RONI areas to enable higher fidelity for other areas of the video.

2.1.1 Face Detection

In the DM365, face detection capability is available as part of the VPFE. It can detect up to 10 faces along with a confidence level. The detected face regions are treated as regions of interest (ROI) and the video encoder represents the face regions with higher fidelity.

2.1.2 User-Supplied Regions of Interest

You can specify certain regions of the video as more important, i.e., regions of interest. Similarly, you can also specify certain regions of the video as non-interest. Then, the video encoder represents the regions of interest with higher fidelity and steals bits from the regions of non-interest to improve the video quality in other parts of the video.

2.1.3 Codec Interface

The region of interest information is passed to the codec xDM interface [1]. The xDM2 understands rectangle coordinates. The following interface is implemented with extended xDM2.0.

```
#define MAX_ROI    5             /* Max ROI supported by codec */

/**
 * @brief      Enumeration for ROI classification, this list can increase
 */

typedef enum {
FACE_OBJECT = 0,
BACKGROUND_OBJECT = 1,
  FOREGROUND_OBJECT = 2,
  DEFAULT_OBJECT = 3,
  PRIVACY_MASK = 4
} ROI_type;


/**
 * @brief      2-dimensional point
 */
typedef struct XDM_Point {
    XDAS_Int32 x;
    XDAS_Int32 y;
} XDM_Point;

/**
 * @brief      Rectangle
 */
typedef struct XDM_Rect {
    XDM_Point topLeft;
    XDM_Point bottomRight;
} XDM_Rect;


/*
 * @brief      Region of interest structure
 */
typedef struct ROI_Interface
{
 XDM_Rect listROI [MAX_ROI];           /* list of ROI passed to codec */
 ROI_type roiType [MAX_ROI];           /* Type classification of ROI */

   XDAS_Int32 numOfROI;                /* Number of Region of interest passed to codec */
   XDAS_Int32 roiPriority [MAX_ROI];   /* Priority of ROI */
 } ROI_Interface;
```

2.1.3.1 Description

- **XDM_Point:** This structure is used to represent a point. It contains X and Y co-ordinate points.
- **XDM_Rect:** This structure is used to define a rectangle that is used for ROI. It contains top left and bottom right co-ordinates of the rectangle.
- **MAX_ROI:** Maximum number of ROI supported by the codec.
- **ROI_Interface:** This is used to define the details of ROI. The structure consists of the following elements:
  - **numOfROI:** Number of ROI limited by MAX_ROI
  - **listROI:** Listing the co-ordinates for the given ROI
  - **roiPriority:** Priority of ROI. A higher positive value means that more importance is given to the ROI compared to other regions. In other words, it determines the number of bits given to ROI; a lesser value of priority means that the importance of the region is less and codec can give lesser bits to that region. Recommended values are 1, 2 and 4.
  - **roiType:** This field specifies the type of ROI. Codec may take some special action depending on the type of ROI.
- **ROI_Interface** will be part of the extended inArgs of codec XDM interface/CE interface.

2.1.4 Example Usage

To provide the end user with flexibility of enabling/disabling ROI coding, an extended dynamic parameter enableROI is provided. enableROI can take only two values:

- 0 (disable ROI coding)
- 1 (enable ROI coding)

Two possibilities exist for ROI coding:

- The VPFE module can be used to detect faces. The detected faces can be treated as ROIs. In this case, ROI type should be set as FACE_OBJECT.
- You can specify co-ordinates of ROI. Here ROI type should be set as FOREGROUND_OBJECT.

In both the cases, the application should populate the fields of ROI_Interface (extended inArgs) accordingly and set enableROI flag to realize ROI coding.

NOTE:

- If any value other than 0 and 1 is used for enableROI, then encoder returns error.
- If FixedQp is used for encoding, ROI gets automatically disabled.
- If the number of ROI being input is greater than MAX_ROI, encoder returns error.
- Currently there is no support for different quality factor for different ROIs. So quality factor of the first ROI is used for all the specified ROIs.
- Quality factor is taken through roiPriority field. Recommended values are 1, 2 and 4.

ROI can be of any type as mentioned in ROI_type. If the ROI is detected as FACE_OBJECT, then a guard band is added around it. For all other ROI types, no guard band is added.

Example Settings for User-Specified ROI:

```
dynamicparams.enableROI = 1;           // set enableROI
inargs.roiParameters.numOfROI = 1;     // Number of ROIs
/* Top left co-ordinates*/
inargs.roiParameters.listROI[0].topLeft.x = 80;
inargs.roiParameters.listROI[0].topLeft.y = 128;
/* Bottom Right co-ordinates*/
inargs.roiParameters.listROI[0].bottomRight.x = 96;
inargs.roiParameters.listROI[0].bottomRight.y = 160;
inargs.roiParameters.roiPriority[0] = 4;                    // ROI priority
inargs.roiParameters.roiType[0] = FOREGROUND_OBJECT;        // ROI type
```

2.2 Pseudo-Multi-Pass Encoding

It is well known that utilizing multi-pass encoding significantly improves video quality. However, real-time encoding precludes the use of true multi-pass encoding. Instead, pseudo-multi-pass encoding can be used to achieve most of the benefits of multi-pass encoding. In pseudo-multi-pass encoding, initially, a low resolution of the video frame is encoded and statistics from this low-resolution encoding are used when encoding the higher resolution of the video frame. Specifically, in surveillance use cases, low-resolution encoding for monitoring of the same source is followed by higher resolution encoding for storage purposes. In this scenario, pseudo-multi-pass encoding comes at no extra cost. In other scenarios, the size of the low resolution can be kept small to ensure that the processing overhead is minimized; for these cases CIF resolution low-resolution encoding is recommended. Note that in order to meet the real-time constraints, both the low- and high-resolution encoders must complete within the capture time between successive video frames.

The low-resolution encoding can generate the following statistics that can be passed to high-resolution encoding:

- Number of intra MBs in the frame
- Number of inter MBs in the frame
- Number of bits consumed by the frame
- Number of bits consumed by each MB

- Average QP value used for the frame
- Motion vectors for each MB
- Macroblock coding mode for each MB

2.2.1 Scene Change Detection

Upon completion of encoding, it is relatively straight-forward to determine scene changes. When scene changes occur, motion estimation obviously will not work and most of the MBs of the video frame will be encoded as intra MBs. This information can be used by the low-resolution encoder to trigger a scene change in the high-resolution encoder. When a scene change is triggered, the video frame is encoded as an I-slice.

2.2.2 Pseudo-Multi-Pass Encoding Setup

A pseudo-multi-pass encoding setup is shown in Figure 1. The input sequence is resized and encoded by the low-resolution encoder. The high-resolution encoder encodes the high-resolution input sequence. The two encoders are run sequentially for every frame captured: first the low-resolution encoder and then the high-resolution encoder. This operation is shown in Figure 2. After capture of frame N, it is resized and fed as input for encoding to the low-resolution encoder. The low-resolution encoder generates metaData as specified in Section 2.2.4 for every frame. This generated metaData is stored in DDR memory. The details of the information stored are indicated in Section 2.2.4. After completion of the low-resolution encoding, the high-resolution encoder is fed the high resolution frame as input. The metaData generated by the low-resolution encoder is consumed by the high-resolution encoder while encoding the high resolution frame. Note that for real-time operation, both low-resolution and high-resolution encoding have to complete within the capture time between successive frame captures, i.e., low- and high-resolution encoding for frame N have to complete before frame N+1 is available for encoding.

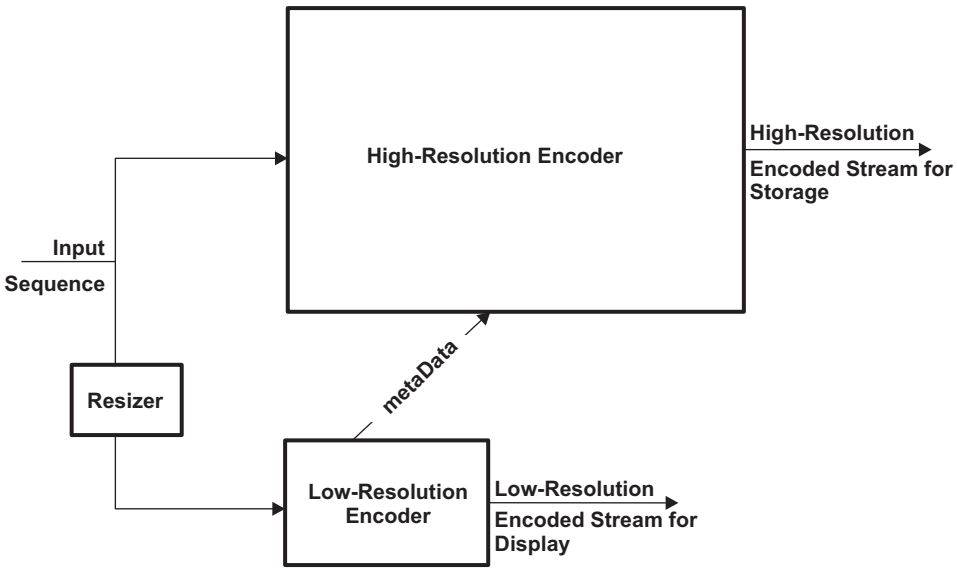

Figure 1. Pseudo-Multi-Pass Encoding

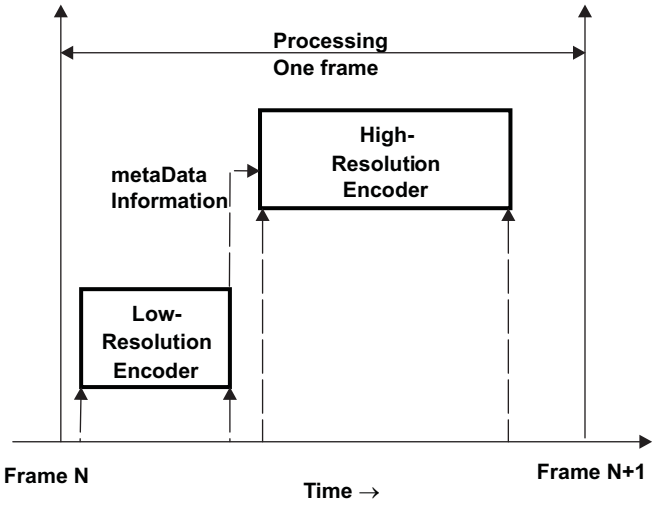

Figure 2. Sequential Operation of the Low- and High-Resolution Encoders

2.2.3 Example Use Cases

2.2.3.1 Dual stream IPNC

In IP Network Camera (IPNC) applications, multiple streams can be encoded. A low-resolution stream is encoded for display purposes, while a high-resolution stream is encoded for storage/archiving purpose. In this scenario, metaData from the low-resolution stream can be used when encoding the high-resolution stream. Here pseudo-multi-pass encoding is achievable at negligible overheads because the application itself necessitates encoding of the input stream at two different resolutions. This is shown in Figure 1.

2.2.3.2 Storage Applications

A high-quality stream is encoded and stored for archival purposes. This is typical in security and surveillance applications. This application necessitates that the stored video be of excellent quality as it could potentially be used for legal purposes. Scene change information derived from a low-resolution stream (e.g., CIF) is of great benefit in this scenario. The same setup as in Figure 1 is used. Here the encoded low-resolution stream is discarded.

2.2.4 Codec interface

```
/*
 * H264VENC_DynamicParams
 * This structure defines the run time parameters for all H264VENC objects
 */
typedef struct IH264VENC_DynamicParams
{
/*
 *        Definition of all the
 *        existing parameters.
 */
   XDAS_Int32 metaDataGenerateConsume;  // Flag to enable/disable metaData Consume
                                        // and metaData generation
} IH264VENC_DynamicParams;

/**
 *           @brief       MVSAD structure
 *           @remark      Structure containing buffer description of SAD value and Motion
Vectors
 */

 typedef  struct MVSAD_Interface
 {
    XDAS_UInt32  sad;
    XDAS_UInt16  mvX;
    XDAS_UInt16  mvY;
 } MVSAD_Interface ;

/**
 *          @brief         MBRow info
 *          @remark        Structure containing buffer description of MB row related Parameters.
 */
typedef  struct MBRowinfo
{
   XDAS_UInt32 gmvVert;              // GMV information per ROW.
} MBRowinfo;

/**
 *          @brief         MB info
 *          @remark        Structure containing buffer description of MB related Parameters.
 */

typedef  struct MBinfo
{
   XDAS_UInt16  numBitsMB;           // Number of bits to encode MB
   XDAS_UInt8  mbCodingMode;         // MB coding mode Inter or Intra
   XDAS_UInt8  mbQP;                 // QP of MB
} MBinfo;

/**
 *          @brief        Frame info
 *          @remark       Structure containing buffer description of frame related  Parameters.
 */

typedef  struct FrameInfo_Interface
{

   XDAS_UInt16  width;               // Width of the Frame in pixels.
   XDAS_UInt16  height;                  // Height of the Frame in pixels.
   XDAS_UInt32  sceneChangeFlag;     // Flag to indicate scene change.
   XDAS_UInt32  bitsPerFrame;        // Number of Bits used to encode frame.
   XDAS_UInt32  frameRate;           // Frame rate per second.
   XDAS_UInt32  bitRate;             // Target Bit rate in bps.
   XDAS_Int32   *mvSADpointer;       // Pointer containing address of MVSAD of
                                        all the MBs in a frame.
   XDAS_Int32   *mbComplexity;       // Pointer containing address of MB
                                        information of all the MBs  in a frame.
   XDAS_Int32   *gmvPointerVert;     // Pointer containing vertical GMV values
                                        per row
} FrameInfo_Interface;
```

2.2.4.1 Description

- **metaDataGenerateConsume:** Flag to enable/disable metaData consume and generation. It can take the following values as per different use cases.
  - **0:** metaData is not used
  - **1:** Generate metaData in current instance. In this case, the encoder generates the metaData and stores the frame related information in the frameInfo_Interface structure.
  - **2:** Consume metaData in current instance. If the flag value is 2, then the current encoder instance uses the metaData generated by the other encoder to improve/customize the encoding operation.
- **MVSAD_Interface:** The SAD value and motion vectors (X and Y direction) for each macroblock are stored in this structure format when mvSADoutFlag is enabled.
- **MBinfo:** This structure is used to store MB information. It contains the following elements.
  - **numBitsMB:** Number of bits to encode MB
  - **mbCodingMode:** MB coding mode Inter or Intra
  - **mbQP:** QP of MB
- **FrameInfo_Interface:** This structure is used to store frame information. It contains the following elements.
  - **width:** Width of the frame in pixels
  - **height:** Height of the frame in pixels
  - **sceneChangeFlag:** Flag to indicate scene change
  - **bitsPerFrame:** Number of bits used to encode frame
  - **frameRate:** Frame rate per second
  - **bitRate:** Target bit rate in bps
  - **mvSADpointer:** Pointer containing address of MVSAD
  - **mbComplexity:** Pointer containing address of MB information of all the MBs in a frame
  - **gmvPointer:** Pointer containing GMV info per row

2.2.5 Example Usage

To provide you with the flexibility to generate/consume metaData information, an extended dynamic parameter *metaDataGenerateConsume* is provided. It can take only three values:

- 0 (no metaData generated or consumed)
- 1 (Generate metaData)
- 2 (Consume metaData)

In case of the low-resolution encoder:

metaDataGenerateConsume is set to 1.

In case of the high-resolution encoder:

metaDataGenerateConsume is set to 2.

When the mvSADflag is enabled, the MV and SAD values of the stream are given out to the application [1] . When *metaDataGenerateConsume* is set to 1 and mvSADflag is also enabled, one extra buffer is required to store MV and SAD values. The address of this buffer is stored in the *mvSADpointer*, which is the part of the structure *FrameInfo_Interface* mentioned in the Section 2.2.4. If mvSADflag is disabled, the *mvSADpointer* pointer points to NULL.

Example Settings for Low-Resolution Encoder

In this case, the application requests for buffers that are used to pass frame-level information from the low-resolution encoder to the high-resolution encoder.

```
dynamicparams.metaDataGenerateConsume = 1   // generate metaData
If(metaDataGenerateConsume = 1)  /* Low resolution encoder */
{
       // Buffer request for Maximum number of MBs in a frame
       sStatus->videncStatus.bufInfo.minOutBufSize[2] = (uiSize >> 4) * 4;
       // Buffer request for storing GMV information per row.
       sStatus->videncStatus.bufInfo.minOutBufSize[3] = uiExtHeight * 4;
       // Buffer request to store frame level information.
       sStatus->videncStatus.bufInfo.minOutBufSize[4] =sizeof(FrameInfo_Interface) ;
}
```

Where *uiSize* is the maximum number of pixels in a frame and *uiExtHeight* is the height of the frame in pixels.

Addresses of these buffers are passed to the encoding function where variables of the structure mentioned in the Section 2.2.4 are updated if metaDataGenerateConsume is enabled.

Example Settings for High-Resolution Encoder

In this case, the application passes the address of the structure of frame level information updated by low-resolution encoder. The high-resolution encoder makes use of frame level information as and when it is required.

```
dynamicparams.metaDataGenerateConsume = 2   // consume metaData

If(metaDataGenerateConsume = 2) /* Will be used in High resolution encoder */
{
       /*
        * Use pointer given by low resolution encoder to
        * fetch frame level information from DDR memory
        * and take appropriate decisions.
        */
}
```

NOTE:

- When setting *dynamicparams.metaDataGenerateConsume = 2* for the high-resolution encoder, the low-resolution encoder must be run with *dynamicparams.metaDataGenerateConsume = 1* or else severe quality degradation occurs.
- The usage of the generated metaData is internal to the codec by the high-resolution encoder; no further input is required from you.

3 References

1. *H.264 High Profile Decoder on DM365 User's Guide* (SPRUEV0)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265