APV21B - Software Reference Functions

Deng LiWei
Nijigasaki IC Design Club

May 31, 2022

INTRODUCTION

The APV21B Real-time Video 16X Bicubic Super-resolution core is a soft IP core. It provides fully real-time 16X Bicubic interpolation video super-resolution, and its high performance design allows it to support video output resolutions in excess of 4K 60FPS.

The APV21B is compatibled with the AXI4-Stream Video protocol as described in the **Video IP: AXI Feature Adoption** section of the *Vivado AXI Reference Guide* (Xilinx Inc. UG1037) and **AXI4-Stream Signaling Interface** section of the *AXI4-Stream Video IP and System Design Guide* (Xilinx Inc. UG934).

This document is intended to describe the implementation and details of the functions of the software reference model for this IP. The complete technical documentation can be found in the user manual of this IP.

1 Overview

This IP has a software reference model in C, which fully compatible with hardware computing.

The software reference model includes an image matrix library, a bitmap library and a Bicubic processing program.

The software has been compiled under Linux with gcc.

2 Image Matrix Library

Image matrix library provides the functions to operate pixel matrix in the memory by coordinate. The Image Matrix Library of the APV21B's software reference model supports storage 32-bpp raw pixels in the memory, and read/write the pixels by coordinate. The library also support color space conversion (RGB to Gray/RGB to YUV/YUV to RGB) for full image. The library provides functions to create (malloc) new memory space for a new size-specificed image.

2.1 Structure

```
typedef struct
{
        int width;
        int height;

        uint8_t *pData;
}ImageMat;
```

The Image Matrix Library provides a structure of image matrix, it is an instance of an actionable object of this function library.

Table 1: Fields in the *ImageMat* Structure

Field	Type	Description
width	int	Width (pixels) of image
height	int	Height (pixels) of image
pData	uint8_t*	Pointer to pixel matrix memory space

2.2 Functions

2.2.1 Create

```
ImageMat* NewImageMat(int width, int height);
```

Create a new image map instance by specificed width and height.

Table 2: Parameters of the *NewImageMat* Function

Parameter	**Type**	**Description**
width	int	Width (pixels) of image
height	int	Height (pixels) of image

2.2.2 Dispose

```
void DestoryImageMat(ImageMat* mat);
```

Dispose and destory the memory space of the pixel matrix.

Table 3: Parameters of the *DestoryImageMat* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix

2.2.3 Get Pixel Data

```
uint32_t GetPixel(ImageMat* mat, int x, int y);
```

Get the pixel data in 32-bpp at specificed coordinate.

Table 4: Parameters of the *GetPixel* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix
x	int	X-coordinate (0 to WIDTH - 1) of image
y	int	Y-coordinate (0 to HEIGHT - 1) of image

2.2.4 Set Pixel Data

```
void SetPixel(ImageMat* mat, int x, int y, uint32_t color);
```

Set the pixel data in 32-bpp at specificed coordinate.

Table 5: Parameters of the *SetPixel* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix
x	int	X-coordinate (0 to WIDTH - 1) of image
y	int	Y-coordinate (0 to HEIGHT - 1) of image
color	uint32_t	32-bpp color to set

2.2.5 Color Space Convert (RGB to YUV)

```
void ImageMatRGBtoYUV(ImageMat* mat);
```

Convert the color space of full image from RGB to YUV, alpha channel reserved.

Table 6: Parameters of the *ImageMatRGBtoYUV* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix

2.2.6 Color Space Convert (YUV to RGB)

```
void ImageMatYUVtoRGB(ImageMat* mat);
```

Convert the color space of full image from YUV to RGB, alpha channel reserved.

Table 7: Parameters of the *ImageMatYUVtoRGB* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix

2.2.7 Color Space Convert (RGB to Gary)

```
void ImageMatRGBtoGray(ImageMat* mat);
```

Convert the color space of full image from RGB to Grayscale, alpha channel reserved.

Table 8: Parameters of the *ImageMatRGBtoGray* Function

Parameter	**Type**	**Description**
mat	ImageMat *	Instance pointer of a image matrix

2.2.8 Copy the Matrix

```
uint32_t CopyMat(ImageMat* dest, ImageMat* src);
```

Copy the full image matrix to a new image matrix.

Table 9: Parameters of the *CopyMat* Function

Parameter	**Type**	**Description**
dest	ImageMat *	Instance pointer of destination image matrix
src	ImageMat *	Instance pointer of source image matrix

3 Bitmap Library

Bitmap library is for creating, reading, modifying Windows standard bitmaps. The Bitmap Library of the APV21B's software reference model supports operation on 24-bpp and 32-bpp Windows uncompressed standard bitmaps. The functions of this library is listed as follows.

1. Read the bitmap file to an Image Matrix;
2. Create a new bitmap with an Image Matrix;
3. Write a bitmap object to a file.

3.1 Structure

3.1.1 Bitmap Head

```
typedef struct
{
        uint32_t biSize;
        uint32_t biWidth;
        uint32_t biHeight;
        uint16_t biPlanes;
        uint16_t biBitCount;
        uint32_t biCompression;
        uint32_t biSizeImage;
        uint32_t biXPelsPerMeter;
        uint32_t biYPelsPerMeter;
        uint32_t biClrUsed;
        uint32_t biClrImportant;
}BitmapHead;
```

The bitmap information head structure of a bitmap file. This strcuture is same as the strcuture of the Windows standard bitmap file.

Table 10: Fields in the *BitmapHead* Structure

Field	Type	Description
biSize	uint32_t	Header size in bytes, usually 40
biWidth	uint32_t	Image width in pixels
biHeight	uint32_t	Image height in pixels
biPlanes	uint16_t	Color planes, usually 1
biBitCount	uint16_t	Bits per Pixel, 24 or 32
biCompression	uint32_t	Compression algorithm, for no compression, 0
biSizeImage	uint32_t	Image size in bytes
biXPelsPerMeter	uint32_t	Horizontal resolution (pixels/m)
biYPelsPerMeter	uint32_t	Vertical resolution (pixels/m)

Continued on next page

Table 10: Fields in the *BitmapHead* Structure (Continued)

biClrImportant	uint32_t	Used on for indexed image

3.1.2 Bitmap Instance

```
typedef struct
{
        char* fileName;
        BitmapHead *head;
        ImageMat *image;
}Bitmap;
```

The bitmap instance. All operations of bitmap are on it. It includes a image matrix and head informations. An optional file name can be assigned.

Table 11: Fields in the *Bitmap* Structure

Field	**Type**	**Description**
fileName	char *	File name string (Optional)
head	BitmapHead *	Pointer to head structure of the bitmap
image	ImageMat *	Pointer to pixel matrix instance of the bitmap

3.2 Enumerations

3.2.1 Bitmap Read Error

```
typedef enum
{
        BMPR_ERR_OK,
        BMPR_ERR_FILE,
        BMPR_ERR_FILE_FORMAT,
        BMPR_ERR_BPP_FORMAT,
        BMPR_ERR_BMP_COMPRESSED
}BitmapReadError;
```

Error enumerations when reading a bitmap file.

Table 12: Items in the *BitmapReadError* Enumeration

Item	**Description**
BMPR_ERR_OK	The file is sucessfully opened and read
BMPR_ERR_FILE	The file can not be opened. Maybe the it is not exist or cause by premission problems
BMPR_ERR_FILE_FORMAT	The file to read is not a standard Bitmap file and the program can not analyze it

Continued on next page

Table 12: Items in the *BitmapReadError* Enumeration (Continued)

BMPR_ERR_BPP_FORMAT	The bit per pixel field of the bitmap file is not 24 or 32, which supported by the library
BMPR_ERR_BMP_COMPRESSED	The bitmap file is compressed, which unsupported by the library

3.2.2 Bitmap Write Error

```
typedef enum
{
        BMPW_ERR_OK,
        BMPW_ERR_NULL_POINTER,
        BMPW_ERR_FILE
}BitmapWriteError;
```

Error enumerations when writing a bitmap file.

Table 13: Items in the *BitmapWriteError* Enumeration

Item	**Description**
BMPW_ERR_OK	The file is sucessfully created and wrote
BMPW_ERR_NULL_POINTER	The bitmap instance to write is a null pointer
BMPW_ERR_FILE	The file can not be created. Maybe the it is cause by premission problems

3.3 Functions

3.3.1 Create (32-bpp)

```
void NewBitmap32bpp(Bitmap** pBitmap, int width, int height,
                const char* fileName);
```

Create a new 32-bpp black bitmap (all pixels 00000000H) with specificed size, and a optional file name for it.

Table 14: Parameters of the *NewBitmap32bpp* Function

Parameter	**Type**	**Description**
pBitmap	Bitmap **	The pointer of a bitmap instance pointer, the instance can be null
width	int	Width (pixels) of the image to create
height	int	Height (pixels) of the image to create
fileName	const char*	File name of the image to create (Optional)

3.3.2 Create (24-bpp)

```
void NewBitmap24bpp(Bitmap** pBitmap, int width, int height,
                    const char* fileName);
```

Create a new 24-bpp black bitmap (all pixels 000000H) with specificed size, and a optional file name for it.

Table 15: Parameters of the *NewBitmap24bpp* Function

Parameter	**Type**	**Description**
pBitmap	Bitmap **	The pointer of a bitmap instance pointer, the instance can be null
width	int	Width (pixels) of the image to create
height	int	Height (pixels) of the image to create
fileName	const char*	File name of the image to create (Optional)

3.3.3 Dispose

```
void DestoryBitmap(Bitmap* bitmap);
```

Dispose and destory a bitmap instance, free all memory space of the pixel matrix.

Table 16: Parameters of the *DestoryBitmap* Function

Parameter	**Type**	**Description**
bitmap	Bitmap *	The pointer of a bitmap instance

3.3.4 Read from File

```
BitmapReadError ReadBitmap(const char* fileName, Bitmap** bitmap);
```

Read a bitmap file to a bitmap instance.

Table 17: Parameters of the *ReadBitmap* Function

Parameter	**Type**	**Description**
fileName	const char*	File name of the image to read
bitmap	Bitmap **	The pointer of a bitmap instance pointer, the instance can be null
RETURN	BitmapReadError	Status of the reading operation

3.3.5 Write to File

```
BitmapWriteError WriteBitmap(Bitmap* bitmap, const char* fileName);
```

Write a bitmap instance to a bitmap file.

Table 18: Parameters of the *WriteBitmap* Function

Parameter	**Type**	**Description**
bitmap	Bitmap *	The pointer of a bitmap instance
fileName	const char*	File name of the image to write (create/overwrite)
RETURN	BitmapWriteError	Status of the writing operation

4 Bicubic Processing Program

The Bicubic Processing provides the exact same quantized Bicubic image interpolation calculation function as the hardware implementation. Its quantization coefficients can be adjusted. It also provides raw floating point calculation functions for the evaluation of quantization effects. The output of these programs is able to verify the results of the hardware output in the same quantization mode.

4.1 Macros

4.1.1 Quantization Switch

USE_FIXED Use quantized fixed for computation.

4.1.2 Quantization Width

FIXED_Q The left-shift bits of quantization.

4.2 Functions

4.2.1 Bicubic Coefficient

```
double Bicubic(double x)
```

Calc the Bicubic coefficient by distance X.

Table 19: Parameters of the *Bicubic* Function

Parameter	Type	Description
x	double	Distance
RETURN	double	Bicubic Coefficient

4.2.2 Bicubic LUT Pre-calculation (Non-quantization)

```
void precalculateLUT()
```

Compute all bicubic coefficients for any conditions into a LUT. No quantization method used.

4.2.3 Bicubic Interpolation (Non-quantization)

```
void bicubic_interp(Bitmap* bitmap, Bitmap** pOutput)
```

Bicubic interpolation super-resolution on a bitmap image. No quantization method used.

Table 20: Parameters of the *bicubic_interp* Function

Parameter	Type	Description
bitmap	Bitmap*	The pointer of input bitmap instance

Continued on next page

Table 20: Parameters of the *bicubic_interp* Function (Continued)

pOutput	Bitmap**	The pointer of a bitmap instance pointer, the instance can be null

4.2.4 Bicubic LUT Pre-calculation (With quantization)

```
void precalculateLUTFiexd()
```

Compute all bicubic coefficients for any conditions into a LUT. Quantization method used.

4.2.5 Bicubic Interpolation (With quantization)

```
void bicubic_interp_fixed(Bitmap* bitmap, Bitmap** pOutput)
```

Bicubic interpolation super-resolution on a bitmap image. Quantization method used.

Table 21: Parameters of the *bicubic_interp_fixed* Function

Parameter	**Type**	**Description**
bitmap	Bitmap*	The pointer of input bitmap instance
pOutput	Bitmap**	The pointer of a bitmap instance pointer, the instance can be null

NOTICE OF DISCLAIMER

The information disclosed to you hereunder (the “Materials”) is provided solely for the selection and use of Nijigasaki IC Design Club products. Tothe maximum extent permitted by applicable law: (1) Materials are made available “AS IS” and with all faults, Nijigasaki IC Design Club hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOTLIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULARPURPOSE; and (2) Nijigasaki IC Design Club shall not be liable (whether in contract or tort, including negligence, or under any other theory ofliability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (includingyour use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including lossof data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if suchdamage or loss was reasonably foreseeable or Nijigasaki IC Design Club had been advised of the possibility of the same. Nijigasaki IC Design Club assumes noobligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to productspecifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. IP cores may be subject to warranty and support terms contained in a license issued toyou by Nijigasaki IC Design Club. Nijigasaki IC Design Club products are not designed or intended to be fail-safe or for use in any plication requiring fail-safeperformance; you assume sole risk and liability for use of Nijigasaki IC Design Club products in Critical.