DPI Projects

- Reading and writing MBP file -
- Reading and writing PNG file -
- Running Convolution with BMP file -

2020

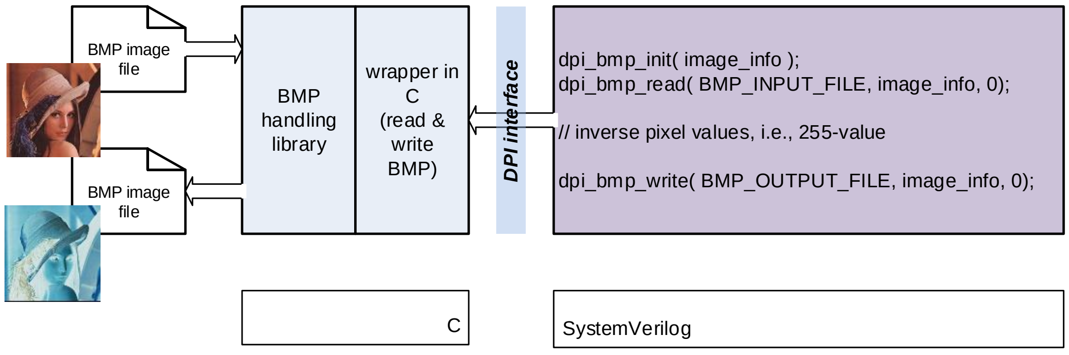
Ando Ki, Ph.D.

adki@future-ds.com

Table of contents

Table of contents

Reading and writing BMP file



```
dpi_bmp_init( image_info );
dpi_bmp_read( BMP_INPUT_FILE, image_info, 0);

// inverse pixel values, i.e., 255-value

dpi_bmp_write( BMP_OUTPUT_FILE, image_info, 0);
```

Reading and writing BMP file

```
// c/dpi_bmp_wrapper.c
#include “bmp_handle.h"
#include “dpi_bmp_wrapper.h”

int dpi_bmp_init( image_info_t *image_info)
{ .... }
int dpi_bmp_read( char *file_name
                 , image_info_t *image_info
                 , int upsidedown )
{  ...... }
int dpi_bmp_write( char *file_name
                 , image_info_t *image_info
                 , int upsidedown )
{  ...... }
```

```
// verilog/tester.sv
module tester
( .... );
    ....
    `include “bmp_handle.sv”  // see the next slide
     ....
    initial begin
       ... ...
       ret = dpi_bmp_verbose_set( 0 );
       ret = dpi_bmp_init( image_info );
       ret = dpi_bmp_read( BMP_INPUT_FILE, image_info, 0);
       .... inverse pixel values ....
       ret = dpi_bmp_write( BMP_OUTPUT_FILE, image_info, 0);
    end
endmodule
```

Reading and writing BMP file: bmp_handle.sv

```
... ...

typedef struct {
      BITMAPFILEHEADER header      ; // Bitmap header
      BITMAPINFOHEADER info        ; // Bitmap information
      int    unsigned ImageWidth  ; // width in pixel
      int    unsigned ImageHeight ; // height in pixel
      int    unsigned BitsPerPixel; // bits per pixel
      int    unsigned BytesPerLine; //
      int    unsigned SkipPerLine ; //
      int    unsigned DibSize     ; // DIB header size in bytes
      int    unsigned ClrSize     ; // Color table size in bytes
      int    unsigned ImageSize   ; // ImageSize in Byte including skip (in bytes)
      chandle         pDibHdr     ; // DIB (device independent bitmap)
      chandle         pColor      ; //
      chandle         pBitMap     ; // pixels (in BMP format, i.e., BGR – B comes first)
} image_info_t;
```

Reading and writing BMP file: bmp_handle.sv

```
import "DPI-C" function int
dpi_bmp_init( inout image_info_t image_info );

import "DPI-C" function int
dpi_bmp_read( input string file_name, inout image_info_t image_info, input int upsidedown );

import "DPI-C" function int
dpi_bmp_write ( input string file_name, inout image_info_t image_info, input int upsidedown );

import "DPI-C" function int
dpi_bmp_get_pixels( output byte pixel[], inout image_info_t image_info );

import "DPI-C" function int
dpi_bmp_put_pixels( input byte pixel[], inout image_info_t image_info );

import "DPI-C" function int
dpi_bmp_wrapup ( inout image_info_t image_info );
```

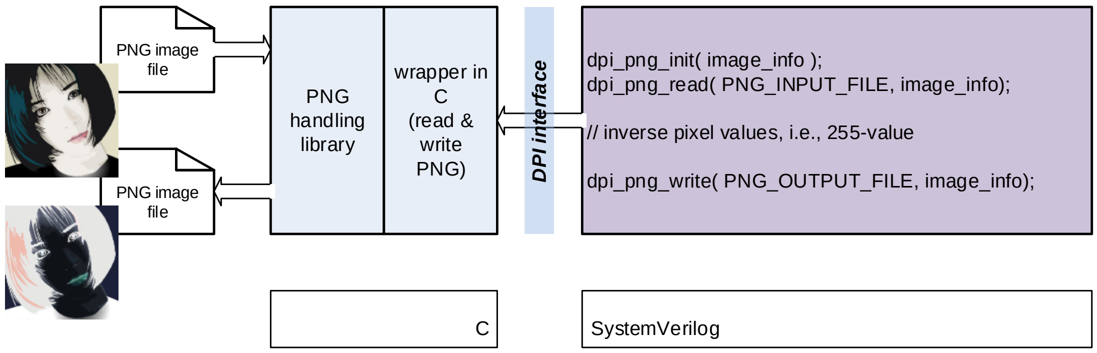
Reading and writing BMP file

- Go to the example directory
  - $ cd ...../codes/prj_bmp/xsim
- Run ‘make’
  - (do not forget to setup environment for simulation)
  - $ make

```
$ cd ...../codes/prj_bmp/xsim
$ set_vivado
$ make
$ display result.bmp
```

Table of contents

Reading and writing PNG file



```
dpi_png_init( image_info );
dpi_png_read( PNG_INPUT_FILE, image_info);

// inverse pixel values, i.e., 255-value

dpi_png_write( PNG_OUTPUT_FILE, image_info);
```

Reading and writing PNG file

```
// c/dpi_png_wrapper.c
#include "stb_image.h"
#include "dpi_png_wrapper.h"

int dpi_png_init( image_info_t *image_info)
{ .... }
int dpi_png_read( char *file_name
                 , image_info_t *image_info)
{  ...... }
int dpi_png_write( char *file_name
                 , image_info_t *image_info)
{  ...... }
```

```
// verilog/tester.sv
module tester
( .... );
    ....
    `include "png_handle.sv"  // see the next slide
     ....
    initial begin
      ... ...
      ret = dpi_png_verbose_set( 0 );
      ret = dpi_png_init( image_info );
      ret = dpi_png_read( PNG_INPUT_FILE, image_info);
      .... inverse pixel values ....
      ret = dpi_bmp_write( PNG_OUTPUT_FILE, image_info);
    end
endendmodule
```

Reading and writing PNG file: bmp_handle.sv

```
... ...

typedef struct {
      int    unsigned ImageWidth  ; // width in pixel
      int    unsigned ImageHeight ; // height in pixel
      int    unsigned BytesPerPixel; // bytes per pixel
      int    unsigned BytesPerLine; //
      int    unsigned SkipPerLine ; //
      int    unsigned ImageSize   ; // ImageSize in Byte including skip (in bytes)
      chandle         pBitMap     ; // pixels (in RGB format, i.e., R comes first)
} image_info_t;
```

Reading and writing BMP file: bmp_handle.sv

```
import "DPI-C" function int
dpi_png_init( inout image_info_t image_info );

import "DPI-C" function int
dpi_png_read( input string file_name, inout image_info_t image_info );

import "DPI-C" function int
dpi_png_write ( input string file_name, inout image_info_t image_info );

import "DPI-C" function int
dpi_png_get_pixels( output byte pixel[], inout image_info_t image_info );

import "DPI-C" function int
dpi_png_put_pixels( input byte pixel[], inout image_info_t image_info );

import "DPI-C" function int
dpi_png_wrapup ( inout image_info_t image_info );
```

Reading and writing BMP file

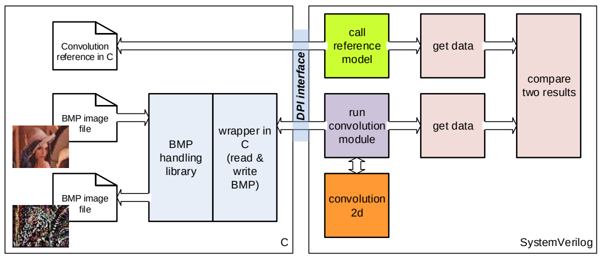
- Go to the example directory
  - $ cd ...../codes/prj_png/xsim
- Run ‘make’
  - (do not forget to setup environment for simulation)
  - $ make

```
$ cd ...../codes/prj_png/xsim
$ set_vivado
$ make
$ display result.png
```

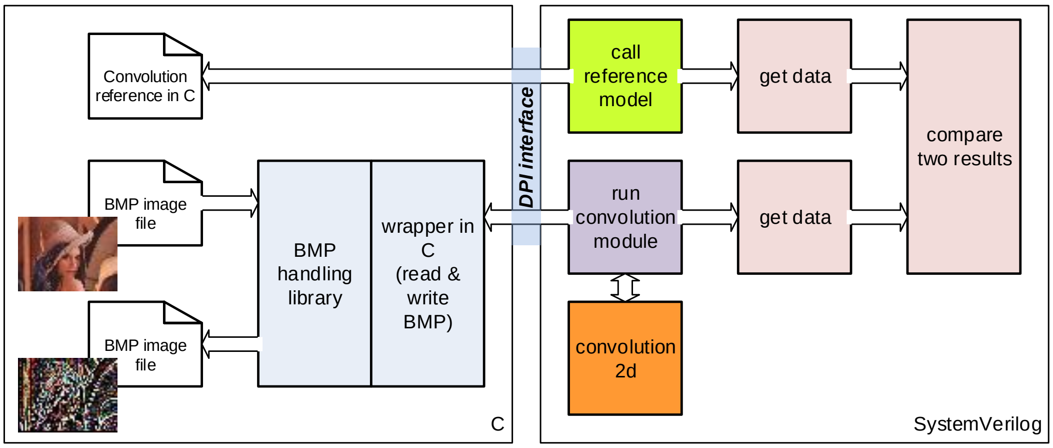
Table of contents

Verification plan

- Use BMP image file as input data
  - refer to following GitHub page
  - https://github.com/adki/BmpHandle
- Use reference C code or behavioral Verilog code to check result
  - Use VPI or DPI to interface C with Verilog
- Choose kernel
  - Laplacian edge detector



Verification plan



Issues

- How to get gray scale bitmap data to be used as single channel input of convolution.
  - Use color space conversion and take luminance: RGB to YCbCr/YUV
- How to deal with 32-bit floating point value after normalizing pixel data.
  - C routines can be used for reference model in which gray image and normalization can be done.