实现函数说明文档

——第六届全国大学生集成电路创新创业大赛景嘉微杯初赛提交文档

队名：虹ヶ咲学园芯片设计同好会

成员：黄金源 邓立唯 林明锋

2022 年 5 月 27 日

前言

本文档仅作为虹ヶ咲学园芯片设计同好会（成员：黄金源、邓立唯、林明锋）参加第六届全国大学生集成电路创新创业大赛景嘉微杯赛初赛提交文档供评委评分使用。

虹ヶ咲学园芯片设计同好会

2022 年 5 月 27 日

目录

第一章　基本介绍

1.1　整体流程

1.2　数据结构

1.2.1　图片格式

1. **BitmapHead:**包含bmp文件信息头各个信息相对于文件头的偏移量

```
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;
```

2. **Bitmap:**包含bmp图像的文件信息：文件名、文件信息头结构体指针和图像数据指针

```
typedef struct {
```

```
  char* fileName;
  BitmapHead *head;
  ImageMat *image;
}Bitmap;
```

3. **ImageMat：** 包含图像的宽度、高度和数据所在的内存首地址

```
typedef struct {
  int width;
  int height;
  uint8_t *pData;
}ImageMat;
```

1.2.2　量化

1. **qint16_t:**带量化位数的16位变量类型，real表示变量真实值，Qn表示该变量需要量化的位数

```
typedef struct {
  int16_t real;
  uint8_t Qn;
}qint16_t;
```

2. **qint32_t：** 带量化位数的32位变量类型，real表示变量真实值，Qn表示该变量需要量化的位数

```
typedef struct {
  int32_t real;
  uint8_t Qn;
}qint32_t;
```

3. **qImageMat：** 带量化位数的图像数据存储格式

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

```
3    int height;
4    qint16_t* pData;
5 }qImageMat;
```

4. **QuantizeError：**量化运算成功或失败标志

```
1 typedef enum {
2   OK = 0,
3   QN_ALARGE,
4   QN_BLARGE
5 }QuantizeError;
```

5. **qCompareRes：**量化比较结果

```
1 typedef enum {
2   Q_EQ = 0,
3   Q_GR,       // >
4   Q_LT        // <
5 }qCompareRes;
```

第二章　实现函数细节

2.1　超分辨率实现

1. **双三次插值**

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

参数:

bitmap: 要进行双三次插值的图像

pBitmap: Bitmap类型的指针变量，用于存储fileName的文件信息

返回值:

无返回值

作用:

对输入图像进行双三次插值，提高图像分辨率

2. **上采样——多线程实现**

```
void upsampling_pthread_x16(ImageMat* src, ImageMat** dst,
    qint16_t** weight);
```

参数:

src: 要进行采样的图像

dst: 上采样后得到的图像

weight: 带有量化位数的权重

返回值:

无返回值

作用：

对输入图像进行双三次插值，提高图像分辨率

说明：

在本次设计中，需要先根据patch_edge进行图像分类，再将图像的patch 和 patch_edge一一对应，执行相应计算。本次设计中，需要处理4k分辨率图像，采用顺序执行的方式非常耗费时间，因此该函数采用了并行计算的方式，先将待处理图像分为4×4共计16个部分，创建16个线程并行计算，等待16个线程返回后再执行下一步计算，大大提高了运行速率。

2.2　锐化

2.3　其余组件

2.3.1　内存管理

```
void freeqWeightSpace(qint16_t ** weight, int32_t type);
```

2.3.2　图片读取与格式转换

1. **ReadBitmap**——读取bmp格式图片

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

参数:

fileName: 要读取的bmp格式图片路径

pBitmap: Bitmap类型的指针变量，用于存储fileName的文件信息

返回值：

BitmapReadError 类型的枚举变量，用于表示读取文件是否成功

作用：

用于读取bmp格式图片

2. **WriteBitmap**——写bmp格式图片

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

参数:

fileName: 要写入的bmp格式图像路径

bitmap: Bitmap类型的指针变量，用于存储fileName的文件信息

返回值:

BitmapWriteError 类型的枚举变量，用于表示写入文件是否成功

作用:

用于写bmp格式图像

3. **ImageMatRGBtoYUV**——将RGB编码图片转换为YUV编码图片

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

参数:

mat: ImageMat类型的结构体指针，用于存储图像的长度、宽度和数据信息

返回值:

无返回值

作用:

将RGB编码图片转换为YUV编码图像

4. **ImageMatYUVtoRGB**——将YUV编码图片转换为RGB编码图片

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

参数:

mat: ImageMat类型的结构体指针，用于存储图像的长度、宽度和数据信息

返回值：

无返回值

作用：

将YUV编码图像转换为RGB编码图像

5. **ImageMatRGBtoYUV**——提取YUV编码格式中Y通道数据

```
ImageMat* splitYChannel(ImageMat* mat);
```

参数:

mat: ImageMat类型的结构体指针，用于存储图像的长度、宽度和数据信息

返回值：

ImageMat 类型指针，存储图片的长度、宽度和YUV编码格式中Y通道数据

作用：

提取YUV编码格式中Y通道数据

6. **mergeYChannel2Image**——将YUV编码格式中Y通道数据合并到图像中

```
void mergeYChannel2Image(ImageMat* img, ImageMat* im_y);
```

参数:

mat: 要合并的目标图像，ImageMat类型的结构体指针，用于存储图像的长度、宽度和数据信息

im_y：ImageMat 类型指针，存储要合并图像的长度、宽度和Y通道数据信息

返回值：

无返回值

作用：

将YUV编码格式中Y通道数据合并到图像中

7. **DestoryImageMat**——释放图像内存空间

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

参数:

mat: ImageMat类型的结构体指针，用于存储图像的长度、宽度和数据信息

返回值:

无返回值

作用:

内部调用free 释放申请的图像内存

第三章　最终效果

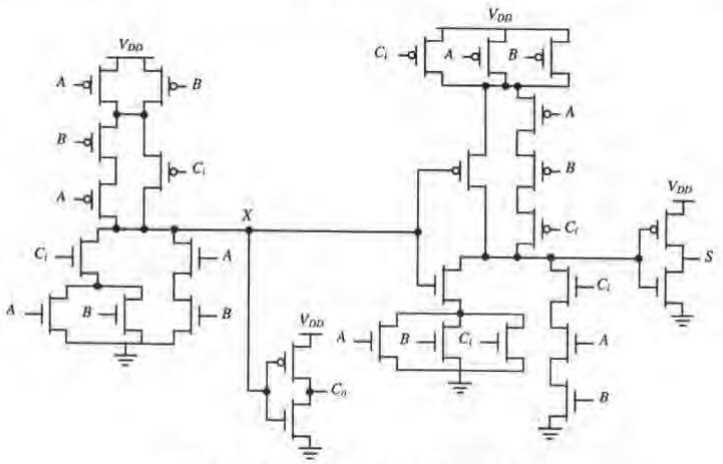

图 11.4　利用互补静态 CMOS 实现的全加器

参考文献

[1] 作者. 文献[M]. 地点:出版社,年份.

[2] 作者. 文献[M]. 地点:出版社,年份.