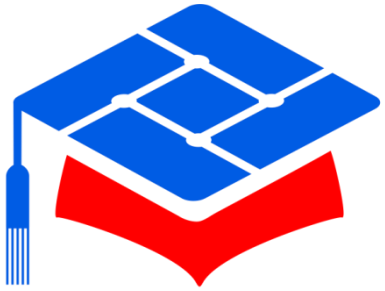

第六届

全国大学生集成电路创新创业大赛

报告类型*: RTL 模块设计说明文档

参赛杯赛*: 景嘉微杯

作品名称*: 一种应用于图形显示的 Upsampling IP

队伍编号*: CICC1215

团队名称*: 虹ヶ咲学园芯片设计同好会

RTL模块设计说明文档

——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 31 日

前言

本文档(RTL模块设计说明文档)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 31 日

目录

第一章　概要

上采样 IP 设计采用了模块化设计，主要分为三大部分：

1. Bicubic 上采样模块
2. 纹理分类模块
3. 自适应锐化模块

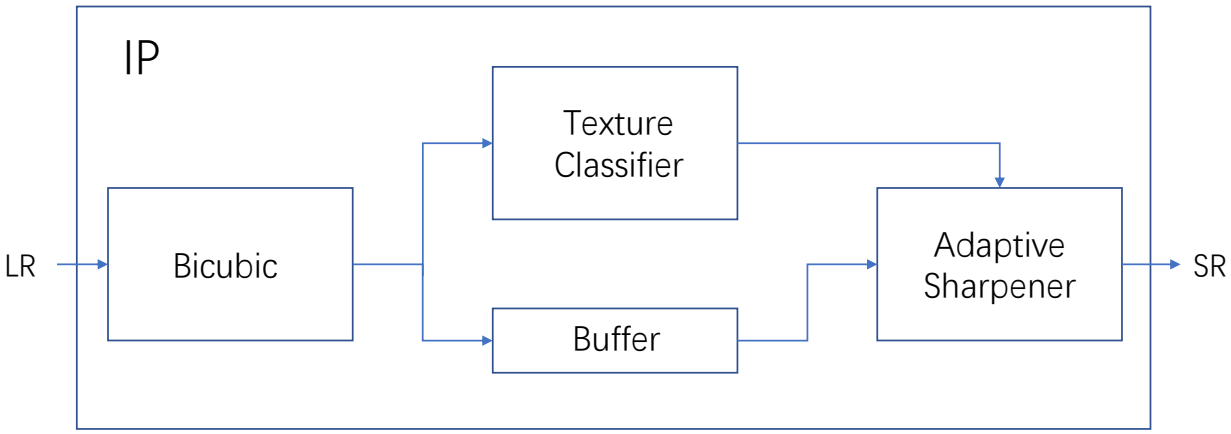

图 1.1: IP 概览

第二章　上采样模块

具体说明详见附件文档 APV21B-RTL Module Design。

第三章　纹理分类模块

纹理分类 IP 可提供对于单通道图像每个 5×5 或 3×3 图像块的纹理特征进行实时分类。包含了一个归一化高斯卷积核、一个标准拉普拉斯算子、一个 LBP 分类器、三个行缓冲模块、一个滤波器参数存储单元和一个控制单元。为了确保卷积操作后图像仍保持原始尺寸，行缓冲模块包含了图像填充处理操作与数据映射模块，由控制单元进行管理。

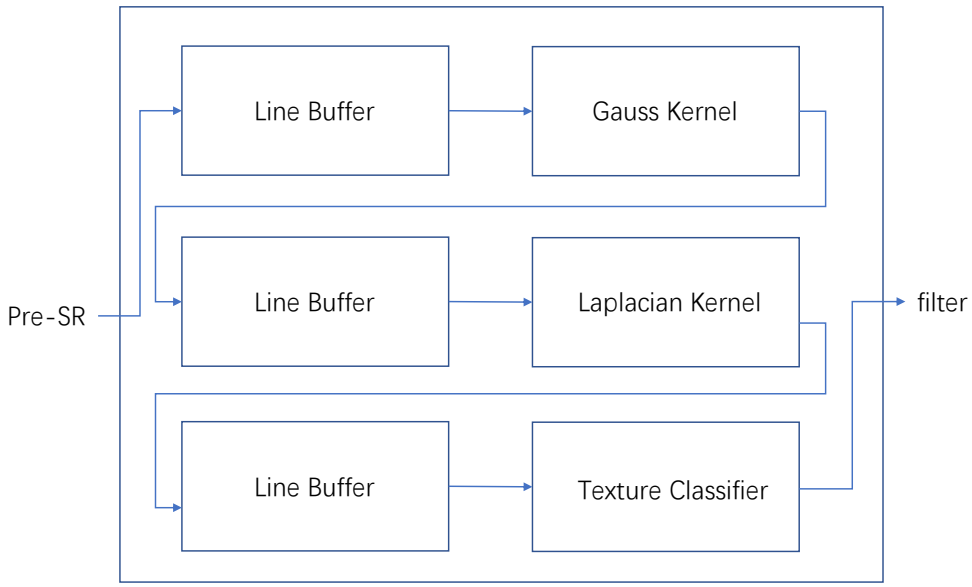

图 3.1: 纹理分类模块结构图

3.1　运算位宽与量化

为了节省片上资源的消耗，最大化提升 DSP 利用率，在本模块中，不同部分对于 DSP48E2 均有不同程度上的优化。其中，针对高斯滤波卷积，将核内权重进行 9 位无

符号量化，在保证高斯卷积核性能的同时充分利用了 DSP48E2 单元的乘法器。这种量化方法仅支持 8 位图像输入。另外，为了保证后续拉普拉斯滤波数据误差尽可能降低，我们将高斯滤波后的数据量化为 20 位，尽可能保留数据位宽，避免引入过大误差影响后续操作。在完成拉普拉斯算子卷积后生成的是 1 位图像输出至 LBP 分类器。

3.2　高斯卷积核

高斯卷积核包含了一个高斯系数寄存器单元、一组并行乘法器单元、一组并行的加法器单元和一组并行舍入单元。 每个像素进行高斯滤波卷积是以像素本身为中心，

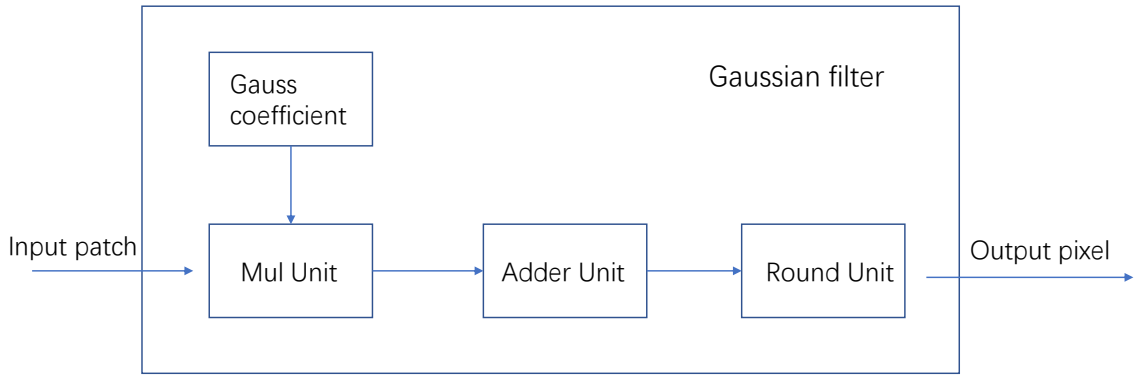

图 3.2: 高斯卷积核结构图

边缘 5×5 的像素块作为数据输入。每个像素块需要与对应的高斯系数进行相乘然后累和。该高斯卷积核 IP 每个时钟周期处理 4 个像素。

3.2.1　高斯系数寄存器单元

高斯滤波卷积核系数是由 σ 所决定，在运行过程中，σ 不会发生改变，所以该系数为常数。同时，5×5 卷积核系数与所在卷积核的位置有关，并意义对应，因此25 个系数只需要存储 6 个系数即可，其余均可通过对称性获取。

3.2.2　乘法器单元

并行乘法单元包含了 7 个 DSP48E2 用于实现 5×5 大小的卷积乘法操作。在两

5	4	3	4	5
4	2	1	2	4
3	1	0	1	3
4	2	1	2	4
5	4	3	4	5

图 3.3: 高斯卷积核数据矩阵数据分布

个时钟周期内可输出结果(一周期内也可实现结果输出，但为了提升 IP 最大时钟频率，输出端插入一级寄存器增大时序违例余量)

由于高斯系数具有对称性，当计算图像块对应的位置的时候，存在几对像素所相乘同一个高斯权重系数。例如上图 (0,0)、(0,4)、(4,0)、(4,4) 对应像素均是乘以同一个系数。因此，我们可以利用这一特性，并结合 DSP48E2 的大位宽乘法器，同时进行两个 8 位数据乘以一个 9 位权重，并保留完整位宽输出。具体设计分析可以查看 APV21B-RTL Module Design 中 Biubic RTL设计详细解释中的乘加单元(MA Unit)介绍。

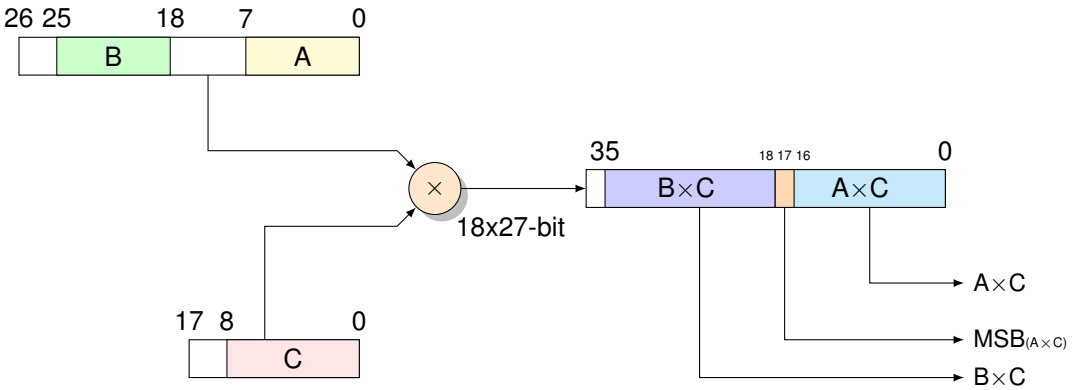

图 3.4: DSP乘法器单元结构图

3.2.3　累和单元

并行累和单元包含了 12 个 DSP48E2 用于实现 25 个数的累和操作。在六个时钟周

期后可输出结果。

$$result = \sum_{i=0}^{24} A_i \tag{3.1}$$

图 3.5 为一个 DSP48E2 内部架构图。

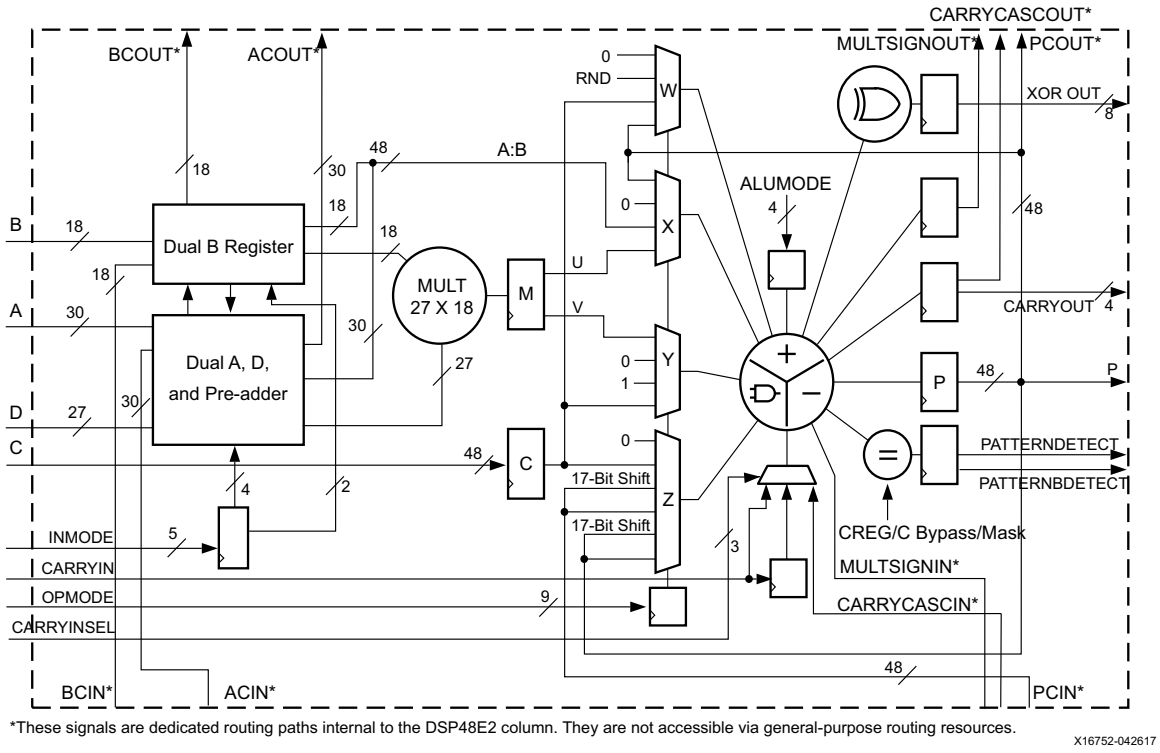

图 3.5: DSP内部结构图

我们可以看到，在 DSP48E2 内部，有两个可以实现加法的环节，第一个 A 与 B 进行乘法之前，有一个相对较小位宽的加法器，另外一个是在 A 与 B 乘法之后一个较大位宽的加法器。我们可以利用这两个加法器实现在一个 DSP48E2 内完成三个数累和，两个时钟周期后输出。

于是，我们可以基于这一个 3 输入 1 输出加法器搭建一个具有三级的 25 输入 1 输出的加法器单元，结果将会在六个时钟周期后输出。

其中需要注意的两个点，第一，我们充分利用了 DSP48E2 内的寄存器资源，以便提升该 IP 最大能达到的时钟频率；第二，我们尽可能的利用上了 DSP48E2 的位宽，保证数据运算时不会因为引入误差而影响后续运算。

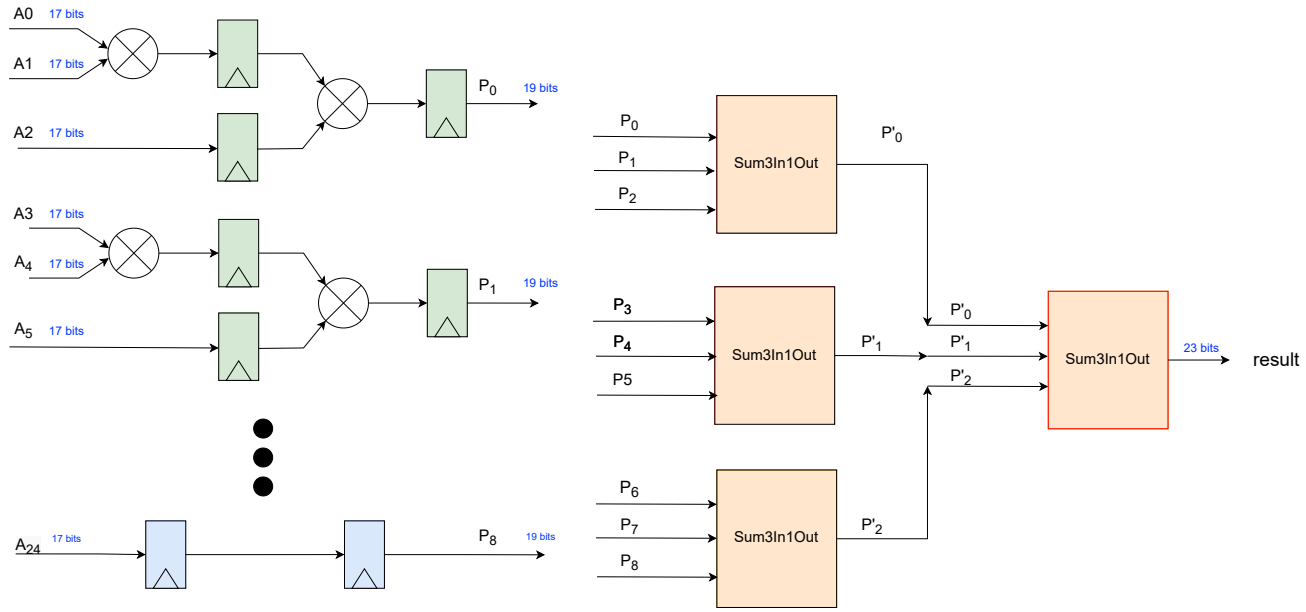

图 3.6: 累和模块

3.2.4　舍入单元

并行舍入单元是通过判断后截断位数的低一位是否为 1 进行简单的四舍五入运算，可以直接通过一个 DSP48E2 完成该操作。由于输入数据与权重数据均是经过量化的，定点位数进行四舍五入是简单的操作。舍入运算均会在两个时钟周期后输出结果。

3.3　拉普拉斯卷积核

拉普拉斯卷积核包含了一组并行的累和单元、一组并行比较单元。

在拉普拉斯卷积核中，我们使用 4 个 DSP48E2 进行 9 数累和运算。每个像素进行拉普拉斯滤波是以像素本身为中心，边缘 3×3 的像素块作为数据输入。每个像素亏需要与对应的拉普拉斯系数进行相乘然后累和。该拉普拉斯卷积核 IP 每个时钟周期处理 4 个像素。

由图 3.7 可以看到，拉普拉斯算子中心为 -8，边缘为 1 的权重分布。所以我们直接利用累和完成 9 数累和。其中中心像素我们直接对像素值的低位补 3 个 0 操作，同时将它所传入的 DSP48E2 的加法器设置为减法操作即可。该拉普拉斯卷积核可在四个时钟周期后输出结果。

需要注意的是，在高斯卷积与拉普拉斯卷积中均使用了 **3输入1输出** 累和模块，但

1	1	1
1	-8	1
1	1	1

图 3.7: 拉普拉斯卷积核数据

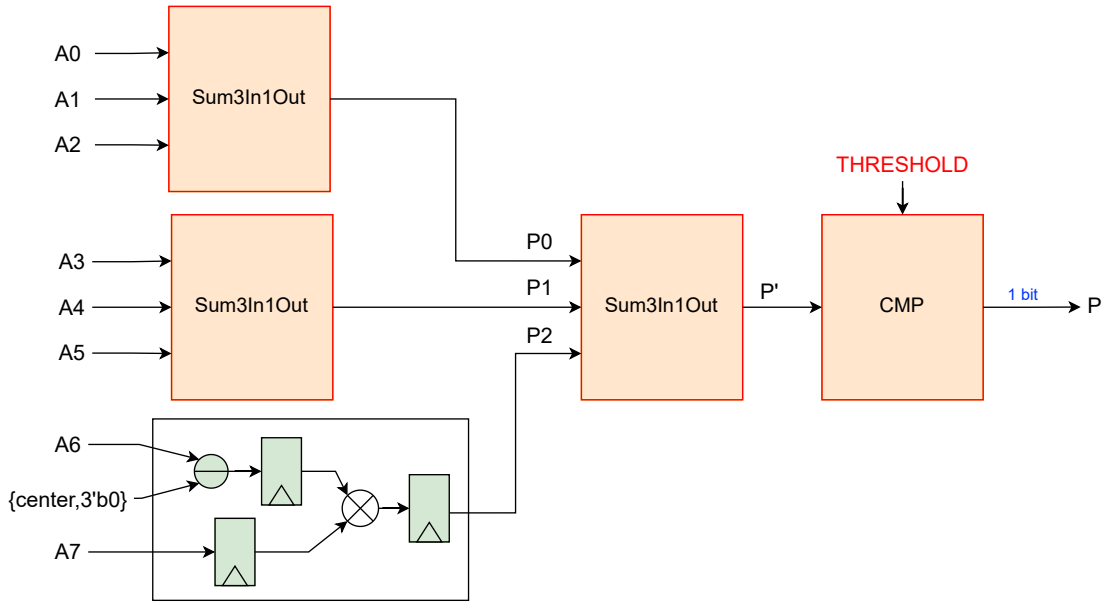

图 3.8: 拉普拉斯卷积

此时拉普拉斯使用时需要将输入数据转换为有符号数再进行运算。当数据位宽拓展时，需要注意高符号位选择。

3.4　纹理分类器

纹理分类器主要包含了四个部分：一个区域划分模块、一个统一编码模块、一个角度编码模块以及一个地址编码模块。 纹理分类单元每个时钟周期处理一个 5×5 经纹理检测后的二值图像块。经过一个时钟周期后输出纹理分类对应滤波器地址。

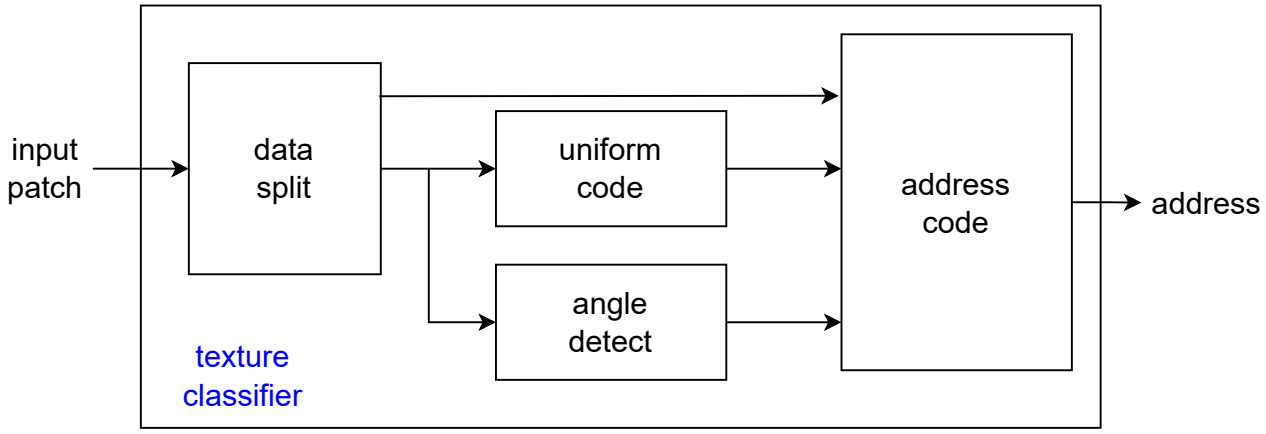

图 3.9: 纹理分类器

3.4.1　区域划分模块

区域划分模块主要将后续进行统一编码与角度编码的像素区域进行提取。

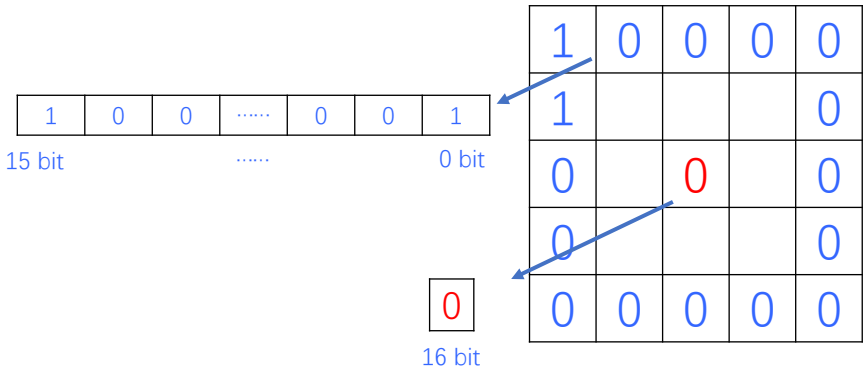

图 3.10: 区域划分模块

其中，对于边缘信息输出的 16 位数据按图像块顺时针提取，输出数据的第 0 位为像素块对应的(0,0)，第 1 位为像素块的(0,1)以此类推。

3.4.2　统一编码模块

统一编码模块用于对边缘区域的像素进行统一编码。我们可以将 16 位的边缘像素数据看作一个环形队列。然后对其按照顺时针的顺序进行边沿检测。可用公式表示为

式 3.2:

$$F = P_0 \oplus P_{15} + \sum_{i=0}^{14} P_i \oplus P_{i+1} \tag{3.2}$$

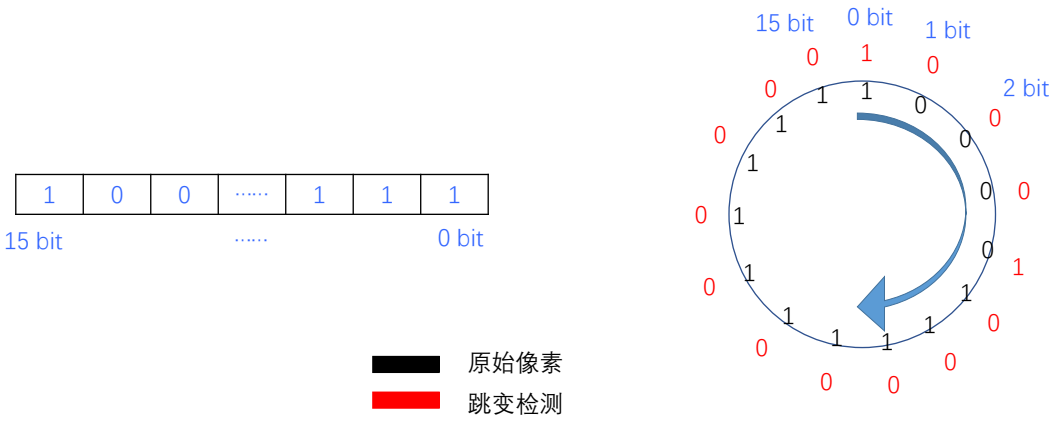

图 3.11: 边缘跳变检测

这里我们举一个简单的例子。黑色 0-1 数字代表了像素的数据，蓝色数字代表了该像素数据原本所在边缘数据的位置。每一位数据与它对应的下一位数据进行异或操作。异或结果使用了红色数字进行表示。其中 1 代表检测到边沿跳变，0 代表没有边沿跳变。

边缘像素数据异或运算后的结果通过计数 1 的个数，则为我们最终需要的跳变次数 F，编码为 0-16 中的偶数。

在进行边沿检测的时候，同时会统计边缘像素数据中1的个数 N。

当跳变次数 F 为 0 时，统一编码结果输出为 0；当跳变次数 F 为 2 时，统一编码结果输出为 N，其中 N 为整数 1-15；当跳变次数 $F \geq 4$ 时，统一编码结果输出为 16。

edge 1s	u	p
0/16	0	0
1∼15	2	1∼15
/	>=4	16

表 3.1: 跳变次数与编码

对于16个1比特统计个数，我们将其拆分为 4 个 **4输入1输出** 的查找表与 1 个 **4输**

入1输出 的小位宽加法器。由于这里进行的数据运算皆是小位宽，我们设计的加法器直接使用查找表进行实现。

3.4.3　角度编码模块

角度编码用于对边缘区域的像素进行角度编码。我们可以将 16 位的边缘像素数据看作一个环形队列。然后对其按顺时针的顺序进行上升沿检测。

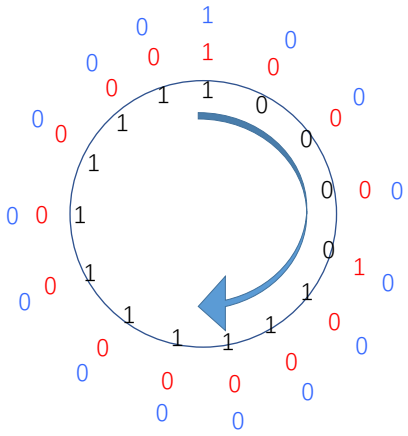

图 3.12: 角度编码

这里与统一编码模块的检测边缘思路一致，但在该模块中我们检测的是边缘数据下降沿跳变的位置。然后对于这 16 位的独热码进行提取 1 的位置，其结果为我们所需要的角度编码，结果为 0-15。

此处需要注意的是，由于我们角度编码产生的结果有效是建立在统一编码结果为 2 的基础，如果统一编码为其他结果，角度编码输出则为 0。同时，当数据经过统一编码后输出为 2 时，该数据进行角度编码的结果输出必然是独热码。

3.4.4　地址编码模块

地址编码模块通过接收来自统一编码模块、角度编码模块与二值像素块中心像素的信息，进行地址编码，用于查找该像素块纹理特征类别的滤波器地址。

由于统一编码的结果可以分为三种情况，第一种情况为特殊的全 0 或全 1，第二种情况为统一编码为 16，第三种情况为统一编码结果为 1-15。在第一种和第二种情况中，我们不考虑角度编码信息，默认置 0。结合中心像素信息，会产生四个对应的地

址。在第三种情况中，我们直接将中心像素信息、统一编码信息与角度信息拼接一起，输出为 9 位的地址。

$$\textbf{case1:}\quad t=\begin{cases}2'b10 & ,center=1\\ 2'b00 & ,center=0\end{cases}$$

$$\textbf{case2:}\quad t=\begin{cases}2'b11 & ,center=1\\ 2'b01 & ,center=0\end{cases}$$

$$where, addr=\{7'b0,t\}$$

$$\textbf{case3:}\quad addr=\{center,u,angle\}$$

注意，在统一编码的三种情况下，第一、二种情况占用了第三种情况的部分地址，但由于第三种情况不会出现编码为 0 的结果，所以在访问地址上并不冲突。

3.5　行缓冲模块

在纹理分类模块中，一共使用了三个行缓冲单元，其中包含两个五行缓冲单元以及一个三行缓冲单元。

每个行缓冲模块用于对输入数据进行缓冲，从而实现为后续计算同时输出三行数据；同时在缓冲时可以由控制信号进行像素填充操作。

缓冲模块的接口可以配置为 AXI4-Stream 接口，方便模块调用。注意，在纹理分类模块内部中，为了减少信号线扇出，仅保留数据总线与数据有效信号。

3.5.1　三行缓冲单元

三行缓冲单元由三个 FIFO 级联形成，读写信号由控制单元进行控制；输出数据连接至输出映射控制器，由控制单元控制，用于实现填充像素功能。

缓冲与运行

当图像的第一行数据流入 FIFO 时，第一行数据保留在第一行 FIFO 进行缓冲不输出；当第二行数据开始流入 FIFO 时，会流入第一行 FIFO，原本处于第一行 FIFO 的

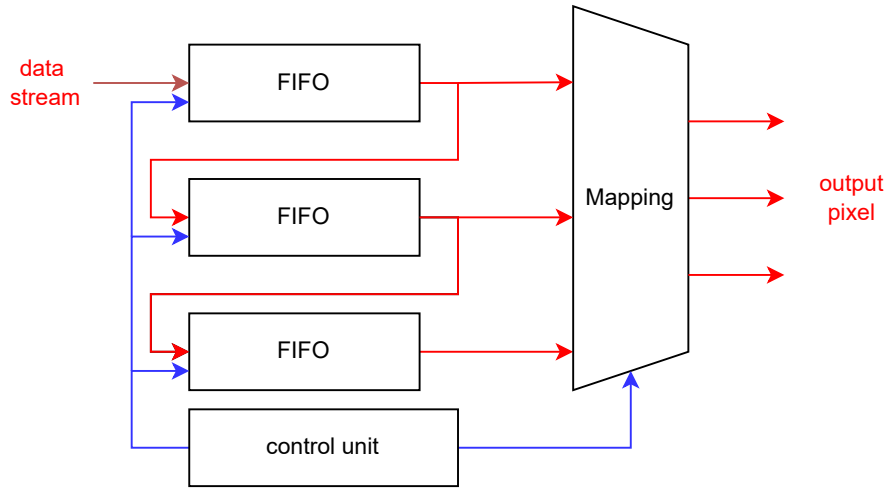

图 3.13: 三行数据缓冲

数据开始流向第二行 FIFO，此时仍处于缓冲模式；当第三行数据到来，会流入第一行 FIFO，原本在第一行 FIFO 的数据会流入第二行 FIFO，原本在第二行 FIFO 的数据会流入第三行 FIFO。同时，当第三行数据到来，由于算法上需要进行像素填充操作，所以可以开始进行缓冲数据流出用于后续运算；当最后一行流入的数据完全进入第三行 FIFO 时，该帧图像缓冲结束。

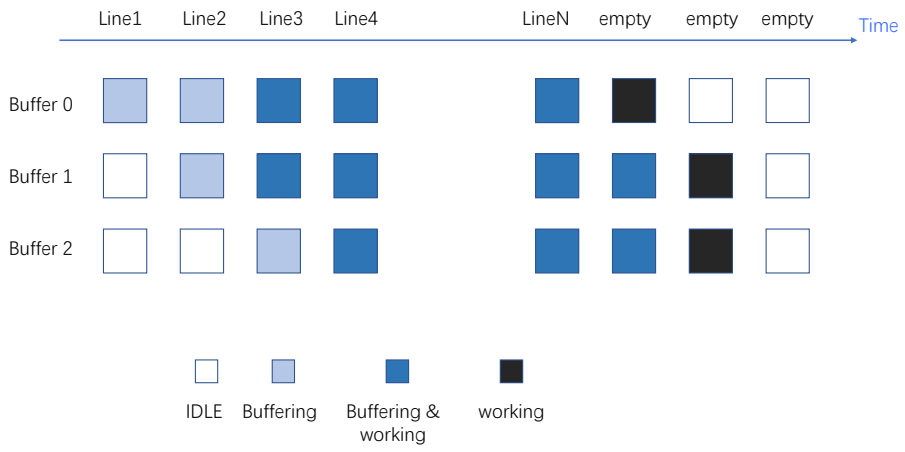

图 3.14: 三行缓冲运行

需要注意的是，刚开始运算像素行的数据仅由第一行 FIFO 与第二行 FIFO 提供，

具体表现为第一、二行 FIFO 输出数据有效，第三行 FIFO 输出数据为无效；当最后运算像素行的数据仅由第二行 FIFO 和第三行 FIFO 提供，具体表现为第二、三行 FIFO 输出数据有效，第一行 FIFO 输出数据无效。

输出映射控制

在输出控制映射模块中，主要完成像素行输入填充的映射以及列填充。具体映射规则由控制单元控制。

当只需要进行上边缘填充的时候，只需要将进入的两行像素中的上边缘像素行映射到另外一个输出端口中即可。下边缘填充同理。

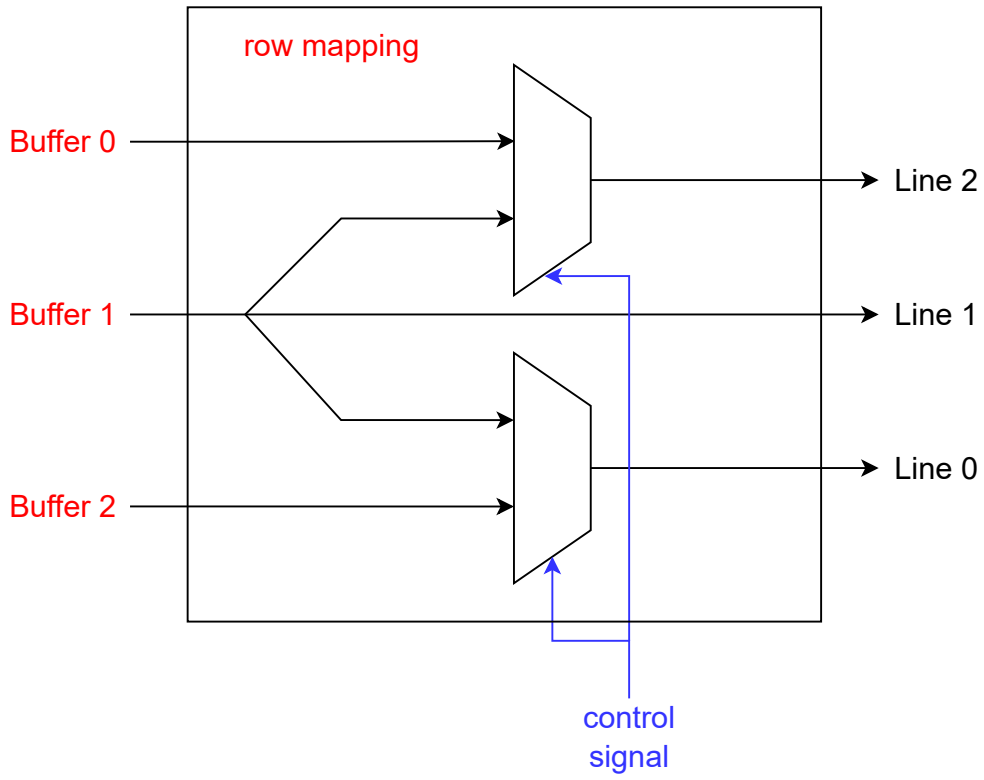

图 3.15: 行映射

控制单元

主要实现了对 FIFO 进行读写控制，以及输出映射的模式选择。其核心为一个计

数器单元，计数输入流有效数据个数。

需要注意的是，上述设计介绍均基于每时钟周期处理一个像素。实际上设计的模块运行在每时钟周期处理四个像素。 FIFO 每次缓冲从 1 个像素 N 比特数据变为个像素 $4N$ 比特输入，输出为四路像素阵列至对应的四个运算单元。在 FIFO 与输出映射单元数据通路间会插入一级寄存器，用于暂存后续需要使用到当前周期的像素。具体表现为将该寄存器数据经过多路选择模块进入输出映射单元。

3.5.2　五行缓冲单元

五行缓冲单元由五个 FIFO 级联形成，读写信号由控制单元进行控制；输出数据连接至输出映射控制器，由控制单元控制，用于实现填充像素功能。

具体设计细节可参考三行缓冲单元设计，五行缓冲单元为其拓展设计，设计思路与其一致。

3.6　自适应锐化模块

自适应锐化模块可进行对于不同 5×5 的图像块根据其纹理特征进行实时锐化。该 IP 包含了一个 5×5 卷积单元、一个滤波器参数存储单元以及一个行缓冲模块。为了确保卷积操作后仍使图像保持原始尺寸，行缓冲模块包含了图像填充处理操作与数据映射模块，由控制单元进行管理。滤波器存储单元用于存储不同纹理类型的锐化参数，为图像锐化卷积提供权重数据。

该自适应锐化模块每个时钟周期可处理 4 个像素。

3.6.1　运算位宽与量化

为了保证运算结果精度以及存储空间尽可能降低，最大化提升 DSP 和片上 RAM 利用率，在本模块中，对于 DSP48E2 和片上 RAM 的使用有进行特殊优化。其中，我们将滤波器参数进行 12 位有符号量化，在保证滤波器性能的同时充分降低了存储空间占用。另外，在 DSP 运算过程中，我们保留所有运算位宽，到结果输出时方进行截断操作。

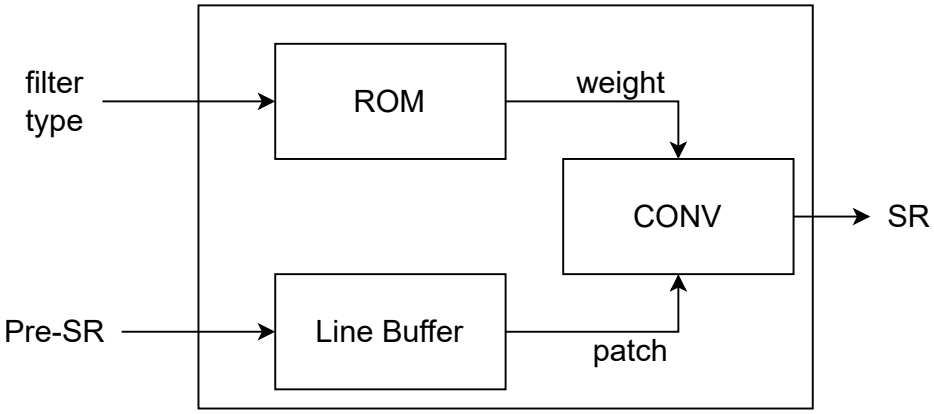

图 3.16: 自适应锐化模块

3.6.2　5×5 卷积单元

$5t \times 5$ 卷积单元包含了一组并行乘法器单元、一组并行累和单元、一组并行舍入单元以及一组输出限幅单元。 每个像素进行滤波卷积是以像素本身中心，大小为 5×5

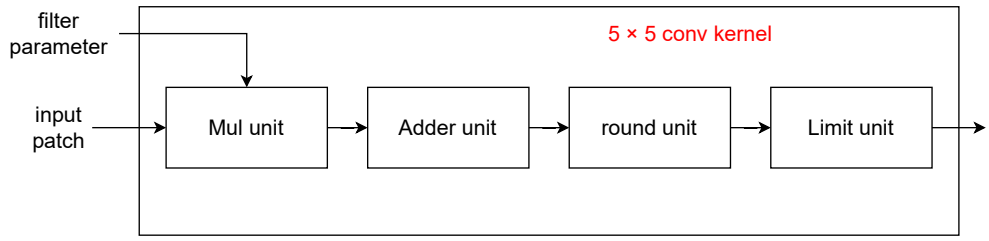

图 3.17: 卷积单元

的像素块作为数据输入，对应的 5×5 滤波器参数作为权重输入。每个像素块需要与对应的预学习的滤波器权重进行相乘然后累和。该卷积核 IP 每个时钟单元处理 4 个像素。

乘法器单元

并行乘法单元包含了 25 个 DSP48E2 用于实现 5×5 大小的卷积乘法操作。在两个时钟周期后可输出结果(一个时钟周期后也可以实现结果输出，但为了提升 IP 最大

时钟频率，输出端插入一级寄存器增大时序冗余)

需要注意的是，此处的自适应滤波器乘法单元与上面所介绍的 **高斯滤波器乘法单元** 有所区别，由于尽可能保留位宽，不能对 DSP48E2 进行进一步优化。

舍入单元

此处舍入单元与 **高斯滤波器舍入单元** 设计一致。

输出限幅单元

由于自适应锐化单元输出为 SR 图像像素，需要将像素数据进行限制在 8bit 内(0 - 255)。此限幅单元实现的功能与舍入单元进行耦合，利用 DSP48E2 特性，能同时实现。在两个时钟周期后输出结果

3.6.3　滤波器参数存储单元

滤波器参数存储单元用于存储 484 个 5×5 滤波器权重参数，每个参数被量化为 12 位有符号二进制数进行存储。为了满足卷积操作实时性要求，尽可能一次性将 25 个权重参数获取，我们将每个滤波器的参数拼接为一个 300 比特位宽二进制数进行存储。

在硬件实现上，我们可以选择 1 个 RAM18BE2 与 8 个 RAMB36E2 拼接而成，或者选择 5 个 URAM288 拼接而成。为了更好的利用 Xilinx UltraScale+ 架构，我们选择了后者进行实现，首先是因为其提供了更大位宽的选择，其次是因为我们可以利用其真双口读写的性质，有效减少多个运算单元访问存储权重参数问题，进而降低一半的权重存储成本。

需要注意的是，由于 URAM288 数据输出位宽为 72 比特，所以需要拼接 5 个 URAM288 才能满足 300 比特数据同时输出。对于第 5 个 URAM288 是浪费了大半的资源。可以通过将第五个 URAM288 替换成 RAMB18E2 进行优化，但为了方便设计以及存储架构的兼容性，暂时保留目前设计方案。

3.6.4　关于设计补充

由于该模块设计为每个时钟周期完成 4 像素运算操作，故需要使用 4 个 5×5 **卷积单元** 与 2 个 **滤波器参数存储单元**。

第四章　总结

具体硬件RTL设计已经详细介绍完毕，各模块参数性能请查看性能说明文档。