缓冲来暂存数据，其数据处理结构图如图 3.14 所示。在图像数据输入时，利用 RAM 暂存 4 行数据，同时每行有 4 个积存器存储单个像素值，从而形成一个 5×5 的数据处理窗口。

当前点被判断为坏点时，就要对该点进行补偿。补偿的方法有两种，当前点为亮点时，对周围的 8 个像素值取平均来补偿，但为了防止该点与周围场景差异过大，也可选择另一种方法，即将周围最亮点的值赋予补偿点。对于暗的补偿方法也是一样。

3.3.2 色彩插值

图 3.15 表示了一个色彩镶嵌滤波器阵列。X、Y 轴表示了每个感光器件的坐标。G、B 原色器件在偶数行交错排列，R、G 原色器件在奇数行交错排列。G 原色器件数是 R 和 B 原色器件数的两倍，因此它对一幅图像有很大影响。传统的彩色图像传感器用三片式的摄像方法，把自然界的图像分别用三色提取。从原理上讲，三片式彩色图像传感器可以获得很高的分辨率，但是它成本高，而且需要对红(R)、绿(G)、蓝(B)信号作重合调整，调整工艺复杂，同时还使体积增大。为实现单片彩色图像传感器，人们引入彩色滤色器阵列 CFA（Color Filter Array），在黑白 CMOS 图像传感器的基础上，增加彩色滤色结构和彩色信息处理模块就可以获取彩色图像。在 CMOS 图像传感器的像素上覆盖此规则的彩色滤色器阵列，就可以获得图像的彩色信息，再经过彩色信息处理，就可以获得色彩逼真的彩色

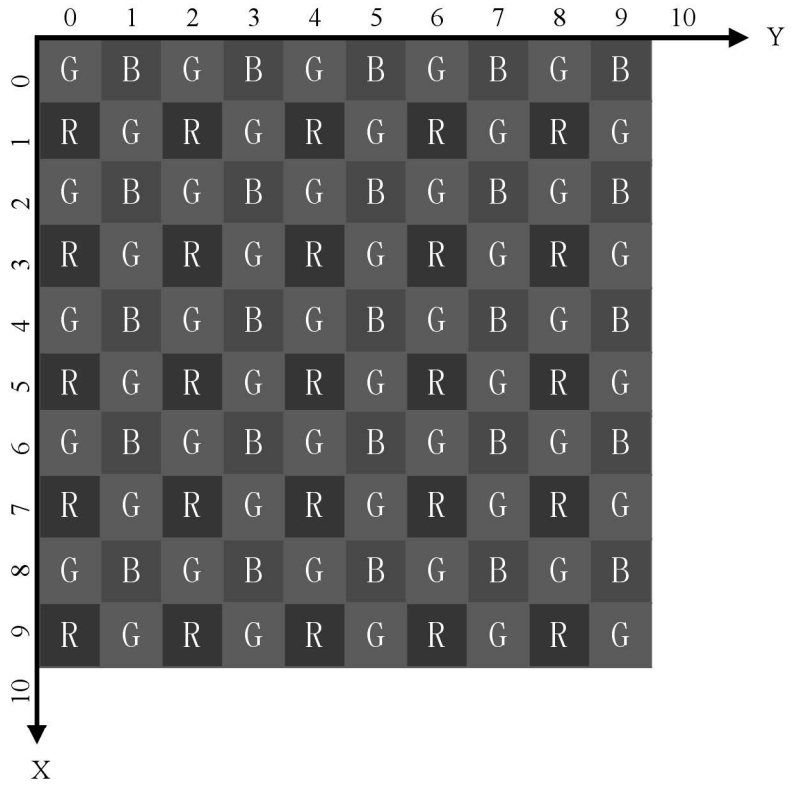

图 3.15 Bayer 型色彩滤波器阵列图

图像。

从原理上讲，采用 R、G、B 垂直条重复间置的栅状滤色器完全适合于 CMOS 彩色图像传感器。此时 CMOS 图像传感器对 R、G、B 分量采用完全相同的采样频率进行采样。由于彩色图像的 60%的亮度信息来源于绿色分量，人眼对亮度信息的变化远比对彩色信息的变化敏感，而且人眼对蓝色和红色图像分辨率低。因此，对三基色采用相同的采样频率显然是不合理的，从而出现了所谓的棋盘式结构的滤色器。典型的棋盘式滤色器是 Bayer 滤色器，如图 3.12 所示。这种滤色器的每一行上只有两种滤色单元：或者是 G、R，或者是 G、B。因此，整个滤色器上 G 光的采样单元数目是 R 光或 B 光的两倍。这种滤色器能很好地用在逐行扫描的方式实现逼真的彩色图像效果。由上面的滤色器结构可以知道，对某一个像素点，其只获得了三基色中的某一个值，其余两个值要从邻近像素插值得到[26]

由于只有一层 RGB 三原色镶嵌滤波器阵列,所以 CMOS 图像传感器的每一像素只有 RGB 三原色中一种颜色的感光器件。因此，预处理器必须从一像素最近邻像素得到该像素的其他两种颜色。插值可以用下式表示：

$$c(x,y)=\sum_{k}\sum_{l}c(x_k,y_l)h(|x-x_k|)h(|y-y_l|) \qquad (3.5)$$

其中 h 是由 c（x_k,y_l）决定的插值核，它是插值算法的最主要的部分。最简单的插值算法是邻域插值算法。由于需要插值的像素都有四个相邻的像素，这种方法将被插值像素的颜色分量赋予邻域中任何一个像素的颜色分量作为插值结果。而常用的是线性插值算法。这种插值方法考察被插值像素的邻域像素，用邻域中同色分量的平均值作为此像素的该色分量。本文在线性插值算法的基础上引入了像素的相关性。这个方法是通过研究邻域像素的相关性，在插值计算时消除领域像素颜色的相关性影响。由于绿色分量数量最多，其中包含的景象信息也比红色分量和蓝色分量多，因此可将绿色分量作为考察相关性的根据[27~28]。

3.3.2.1 双线性插值算法的结构

双线性插值算法由于其算法简单、重构图像效果较好、硬件代价小而被广泛采用。其基本结构是以一个3×3滤波器窗口为核心，根据R、G、B分量不同的相邻位置来重构缺失的另外两种颜色。

（1）R和B原色器件位置上的G由该颜色周围四个G求平均得到。

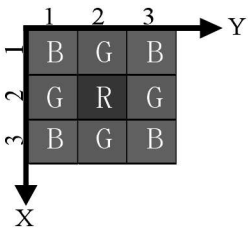

图 3.16 以 R 原色的位置情况

① 在 R 位置（奇数行偶数列），其 RGB 计算方法如下：

$$R(2,2) = R(2,2) \tag{3.6}$$

$$G(2,2) = (G(2,1) + G(2,3) + G(1,2) + G(3,2)) / 4 \tag{3.7}$$

$$B(2,2) = (B(1,1) + B(1,3) + B(3,1) + B(3,3))/ 4 \tag{3.8}$$

② 在 B 位置（偶数行奇数列），其 RGB 计算方法如下：

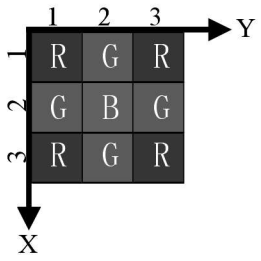

图 3.17 以 B 原色的位置情况

$$R(2,2) = (R(1,1) + R(1,3) + R(3,1) + R(3,3)) / 4 \tag{3.9}$$

$$G(2,2) = (G(2,1) + G(2,3) + G(1,2) + G(3,2)) / 4 \tag{3.10}$$

$$B(2,2) = B(2,2) \tag{3.11}$$

（2）对于 G 位置，由于 G 在不同的位置上有不同的近邻，要分两种情况：

① 对于 RG 行的中心点时，其 RGB 计算方法如下:

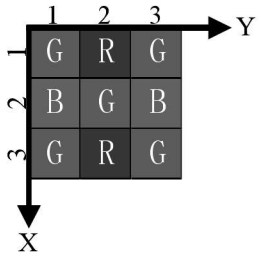

图 3.18 GB 行 G 原色的位置

$$R(2,2) =(R(1,2) + R(3,2)) / 2 \tag{3.12}$$

$$G(2,2) = G(2,2) \tag{3.13}$$

$$B(2,2) = (B(2,1) + B(2,3)) / 2 \tag{3.14}$$

② 对于 RG 行的中心点时，其 RGB 计算方法如下:

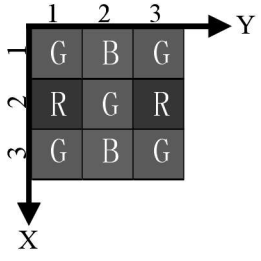

图 3.19 RG 行 G 原色的位置

$$R(2,2) = (R(1,2) + R(3,2)) / 2 \quad (3.15)$$

$$G(2,2) = G(2,2) \quad (3.16)$$

$$B(2,2) = (B(2,1) + B(2,3)) / 2 \quad (3.17)$$

3.3.2.2 本文提出的插值算法的结构

根据研究发现，引入相关性可以大大提高插值以后重建的图像质量，基于相关性插值的方法，本设计需要使用 5×5 的滤波器窗口，这样就要为了完成插值运算需要至少使用 4 行缓冲来保存图像数据，此数据流处理如上小节图 3.14 所示。本设计所用算法在一般的线性插值算法基础上得到各像素点另两种颜色分量的值，再在此像素值的基础上除去相关性的影响，使还原的彩色图像更加逼真[29~30]。

其算法如下：

（1）当 G 颜色成分位于窗口中心，左右为 R 颜色成分，上下为 B 颜色成分，此时插值情况如图 3.20 所示。

G_{11}	B_{12}	G_{13}	B_{14}	G_{15}
R_{21}	G_{22}	R_{23}	G_{24}	R_{25}
G_{31}	B_{32}	G_{33}	B_{34}	G_{35}
R_{41}	G_{42}	R_{43}	G_{44}	R_{45}
G_{51}	B_{52}	G_{53}	B_{54}	G_{55}

图 3.20 插值计算时 G 颜色成分位于窗口中心情况

这时插值算法为：

$$B_{3,3} = G_{3,3} + \frac{B_{3,2} + B_{3,4} - G_{3,2} - G_{3,4}}{2} \quad (3.18)$$

$$R_{3,3} = G_{3,3} + \frac{R_{2,3} + R_{4,3} - G_{2,3} - G_{4,3}}{2} \quad (3.19)$$

其中公式中 G 分量值为线性插值得到的初始值。

（2）当 R 颜色成分位于窗口中心，四周为 G 颜色成分，此时插值情况如图 3.21 所示。

R_{11}	G_{12}	R_{13}	G_{14}	R_{15}
G_{21}	B_{22}	G_{23}	B_{24}	G_{25}
R_{31}	G_{32}	R_{33}	G_{34}	R_{35}
G_{41}	B_{42}	G_{43}	B_{44}	G_{45}
R_{51}	G_{52}	R_{53}	G_{54}	R_{55}

图 3.21 插值计算时 R 颜色成分位于窗口中心情况

这时为了提高重建图像质量，滤波器窗口中所有的绿色成分都参与了插值运算，其插值算法如下：

$$G_{m,n}=\frac{G_{m,n-1}+G_{m,n+1}}{2}+\frac{-R_{m,n-2}+2R_{m,n}-R_{m,n+2}}{8} \tag{3.20}$$

$$R_{m,n}=G_{m,n}+\frac{R_{m-1,n-1}+R_{m+1,n-1}+R_{m-1,n+1}+R_{m+1,n+1}-(G_{m-1,n-1}+G_{m+1,n-1}+G_{m-1,n+1}+G_{m+1,n+1})}{4} \tag{3.21}$$

其他两种情况，插值算法与以上类似。插值计算仿真如图 3.22。

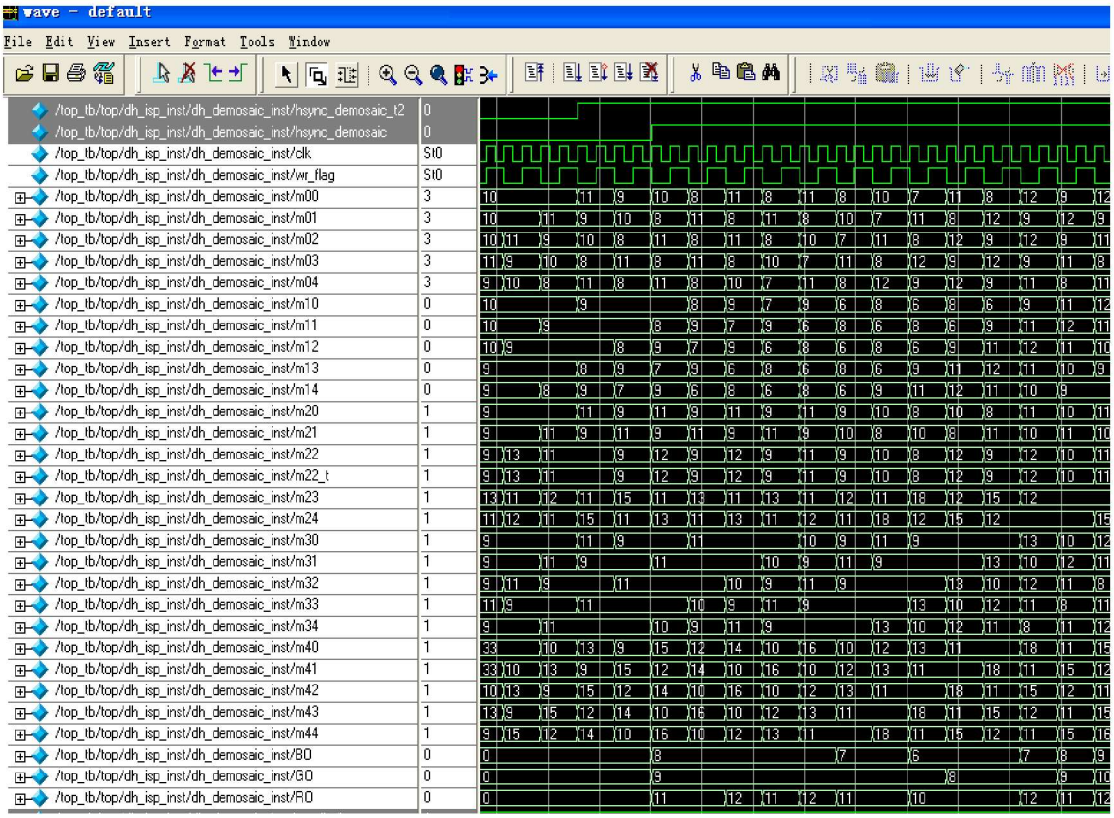

图 3.22 插值计算时序仿真