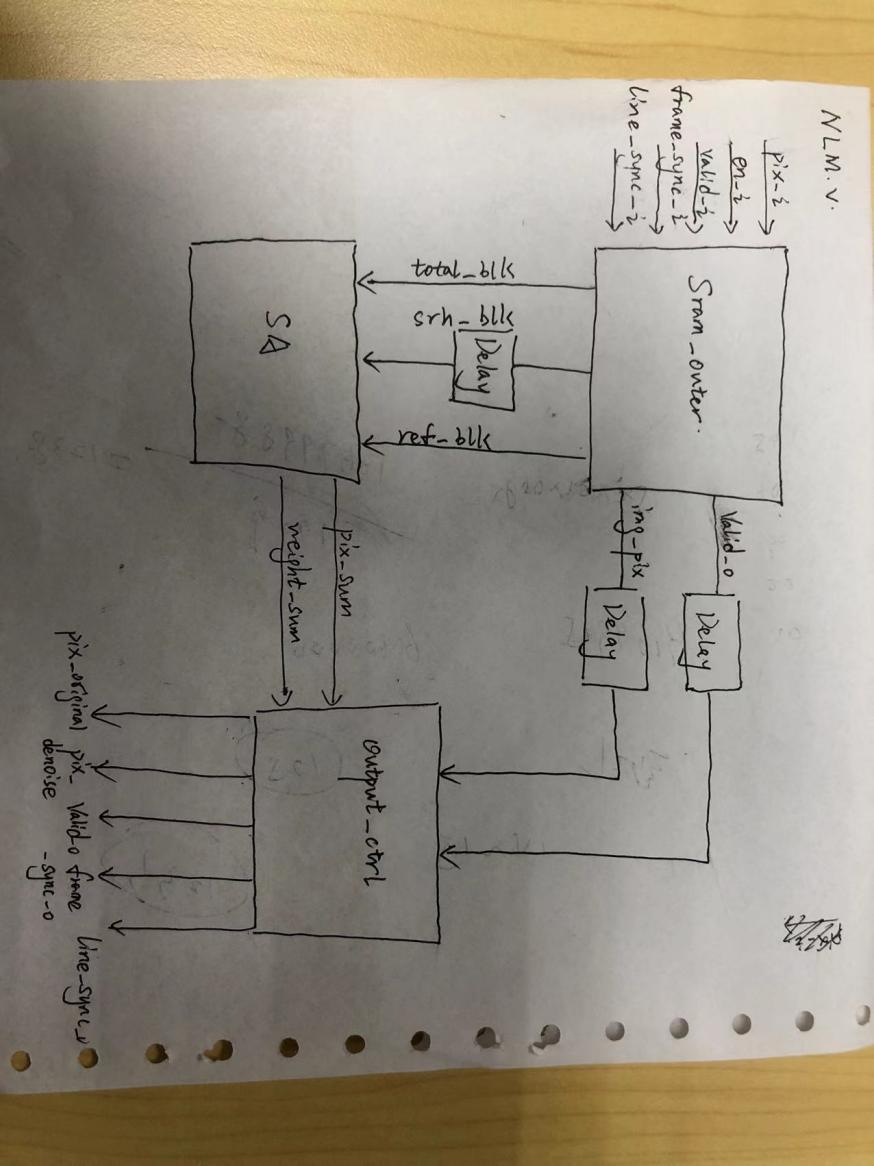
自顶向下模块功能描述

1. 顶层模块：NLM.v

输入：行场同步信号、数据有效信号、模块使能、原像素流、

输出：行场同步信号、数据有效信号、原像素流、去噪像素流

顶层模块主要负责连接已有的模块，搭建数据通路，顶层框图如下：



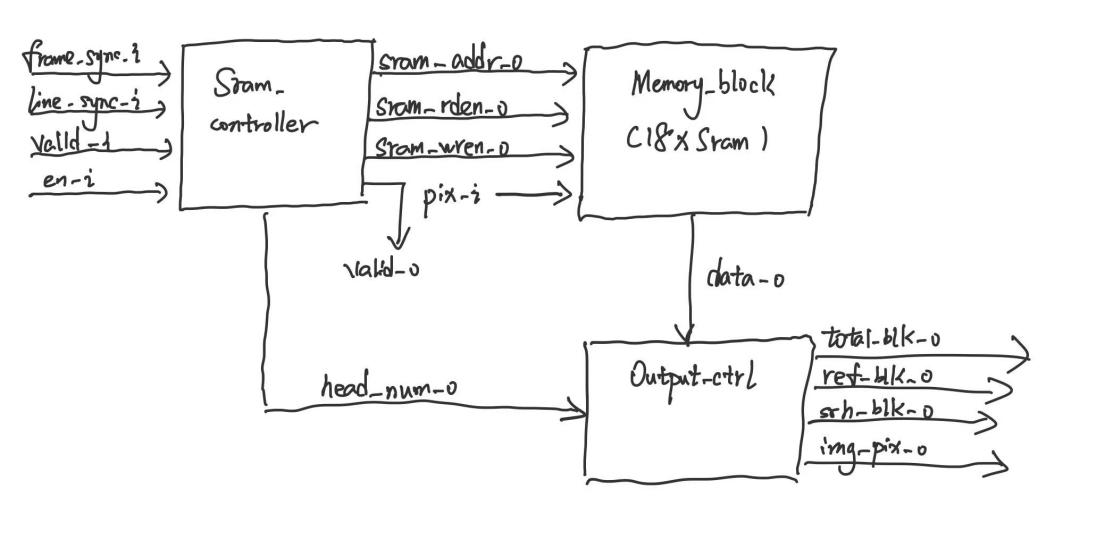
其中，Sram\_outer的输出valid\_o以及img\_pix进行延时的目的，是将原像素流与脉动阵列输出的pix\_sum和weight\_sum对齐，便于进行计算。Srh\_blk进行延时的原因是将图像块中心像素点与PE1输出的权重对齐。

1. Sram外围电路：Sram\_outer.v

输入：行场同步信号、数据有效信号、模块使能、原像素流

输出：合适时序的原像素流、SA计算所需的三个列输出(srh\_blk\ref\_blk\total\_blk)

Sram外围电路的主要功能是根据行场同步信号与输入像素流，对像素进行存储，并按列输出，同时生产时序合适的原像素流，其中包含三个模块，本模块框图如下：

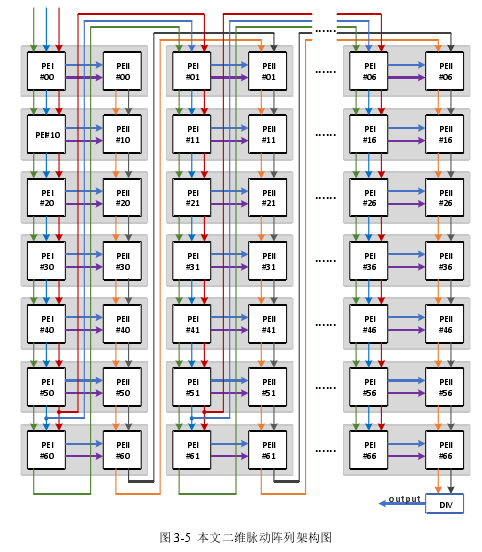


1. 脉动阵列：SA.v

输入：三列数据

输出：像素加权和、权重和（除了正在处理的中心点以外）

NLM算法的主要计算在此完成，框图与下图类似，唯一的区别在于，下一列第一个PE的srh\_blk(蓝色）和total\_blk（红色）来自于上一列的倒数第三个PE的输出（因为raw图需要隔列取搜索窗口像素点）。



1. output\_ctrl模块

输入：像素加权和、权重和、与前者对齐的原像素流与数据有效信号

输出：去噪流(pix\_denoise)、与去噪流对齐的原像素流(pix\_original)，以及后级需要的行场同步与数据有效信号。

该模块总共三步：

第一步，由于像素流已经与权重和对齐，可以直接通过简单的组合逻辑获取中心像素点的权重以及加权和。将pix\_sum以及为满足cordic除法器要求而左移12位（像素位宽）的weight\_sum送入cordic触发器，获取去噪结果。

第二步，将输入的原像素流及其数据有效信号和cordic除法器的结果对齐（由于像素位宽是12位，所以用了12个cordic迭代单元，也理应用12个寄存器进行延时）。

第三步，创建控制输出的状态机。产生驱动状态机的行列计数信号，对最后输出的像素流进行计数。总共四个状态，IDLE为空闲状态，FRONT为前八行原像素输出状态，NORMAL状态为中间行，前后8个像素点为原像素值，其余为去噪结果。POST状态为后八行原像素输出状态。

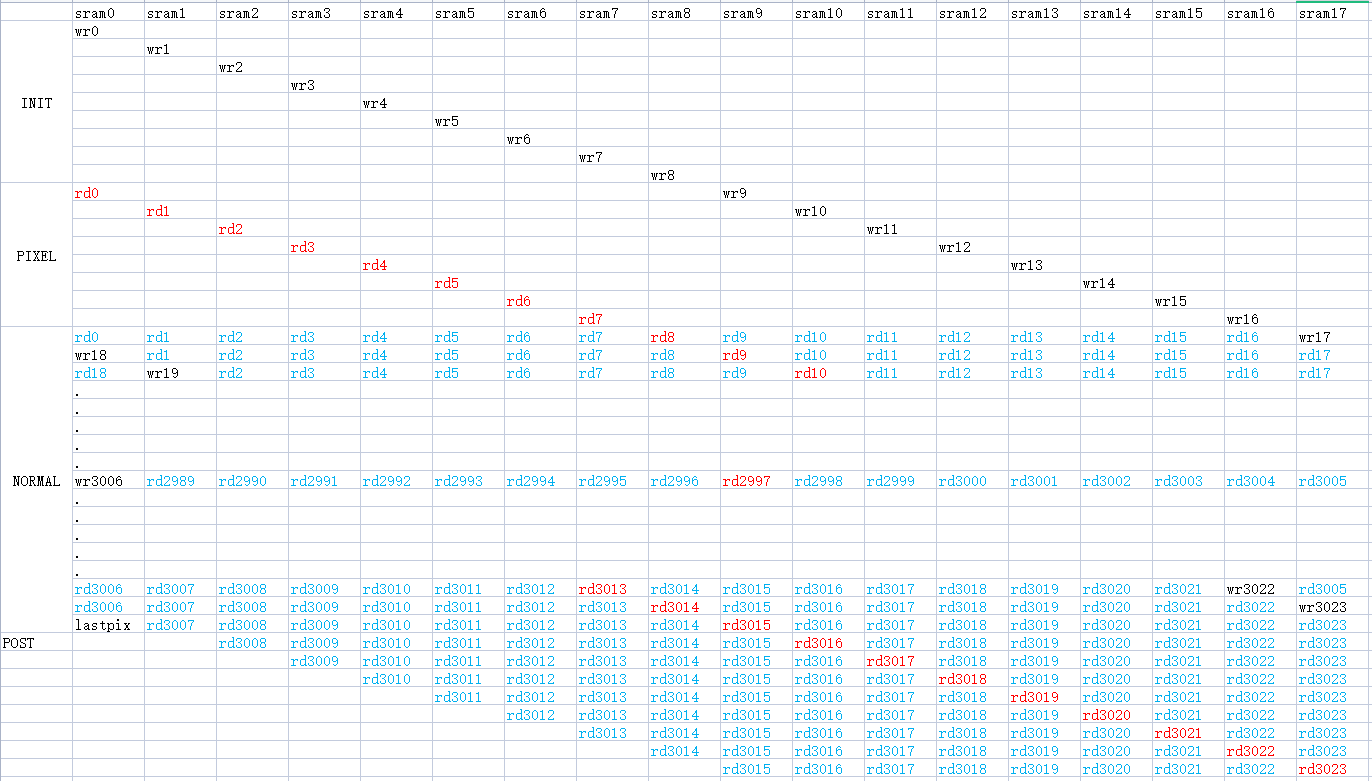
根据状态，最后可以用组合逻辑获取去噪像素流。

1. Sram控制电路：Sram\_controller.v

输入：数据有效信号、行场同步信号、模块使能

输出：Sram读写使能、Sram读写地址、数据有效信号、当前读取的列首地址(head\_num\_o)

该模块主要功能是根据行场同步信号，生成SRAM的控制信号。其具体时序及状态机如下图所示。INIT是前9行的初始化状态，此时只会进行写操作。PIXEL是原像素流读取状态，会对之前已经缓存的行按顺序进行读操作，最终使得输出的原像素流与最终去噪像素流之间仅仅存在一个脉动阵列的延时。NORMAL状态开始正常读写，为后续计算提供列数据。POST状态是在像素写完之后，继续循环移位读使能信号，以获取剩余的原像素流。



1. 行缓存：Memory\_block.v

输入：sram读写使能信号（18位）、sram读写地址信号

输出：1 \* 18的一列数据

该模块用18片SRAM对数据进行缓存，并将地址上的18个数据提取出来，组成一个大位宽的信号传给下一级。

1. SRAM输出模块：Memory\_output.v

输入：行缓存传来的一列数据、sram控制电路传来的列头地址

输出：三列输出、原像素流输出

行缓存传来的1\*18个数据中，只有17个是有效的，我们可以根据列头地址提取出这17个数据组成一列，经过7个寄存器可以同时输出有效的3列数据，且可以完整地输出原像素流，结构与华科的论文类似，减去了padding的部分。

1. 脉动阵列基本单元：PE.v、PE1.v、PE2.v

这一部分的硬件与华科的论文基本完全相同，唯一的区别在于，由于搜索窗口内像素点需要隔行取，所以每一行的PE对传入的列数据都需要隔开一行取，srh\_bit\_o（搜索窗口像素点）的选取也是同样的方式。此外，由于增加了参数的可配置性，采用了加法树来实现累加器。

1. 加法树：AdderTree.v

并行加法树模块需要配置数据位宽以及数据长度，即可对传入的length个数据进行累加。参考：<https://blog.csdn.net/MmikerR/article/details/106469344>

1. cordic除法迭代单元：cordic.v

完全采用华科论文中的结构。

可以通过vscode重新编译：

iverilog -o ./build/a.out ./src/tb\_NLM.v ./src/Sram\_controller.v ./src/sram.v ./src/Memory\_block.v ./src/Memory\_output.v ./src/Sram\_outer.v ./src/AdderTree.v ./src/NLM.v ./src/PE.v ./src/PE1.v ./src/PE2.v ./src/SA.v ./src/output\_ctrl.v .\src\cordic\_divider.v