数字集成电路课程设计报告

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

课程设计名称：H.266 运动向量估计电路

姓　　名：冯洪

学　　号：519030910130

小 组 成 员：朱恒毅、杨凯杰、王清训
冯洪、林羽

提 交 日 期：2022/05/04

目录

1. 课程设计规范简介

1.1. ME(运动估计)简介

运动估计(Motion Estimation，简称 ME)是　之一。

大多数视频序列中，相邻图像内容非常相似，仅是一些运动物体的位置发生了变化。 因此，对于运动物体，可以通过仅对其运动信息编码来减少编码量。ME 就是指提取当前图像运动信息的过程。基于块的运动表示法（将图像分成大小不同的像素块，只要块大小选择合适，则各个块的运动形式可以看成是统一的）兼顾了运动估计精度和复杂度，是历代视频编码国际标准的核心技术。

由于时间上的相关性，视频每一帧图像与相邻的若干帧图像之间存在一定的信息冗余量，运动估计的目的便是找出冗余的信息，从而将冗余的信息消除，这样就能够使得传输与存储的信息大幅度的降低，提升压缩率。

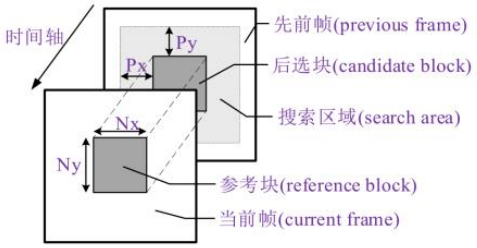
1.2. 设计指标及应用要求

1) 每秒 60 帧 4K 视频（3840×2160@60fps）的实时处理能力
2) 采用全搜索 ME 算法、支持 8×8 块大小的 SAD 计算、搜索区间为[-7,8]
3) 芯片设计工艺：华力 55nm 工艺
4) 评价指标：电路的实时处理能力、芯片的 PPA（Performance or frequency, Power, Area）、输入/输出数据的带宽及其利用效率。

1.3. 基于块匹配的 ME 算法

1.3.1. 算法思路

基于块匹配的运动估计算法思路如图 1 所示。当前时刻的视频帧称为当前帧，上一时刻的视频帧称为先前帧。在当前帧中的某个参考块，可以用先前帧中的某个候选块以及二者之间的坐标差（运动向量）来表示。



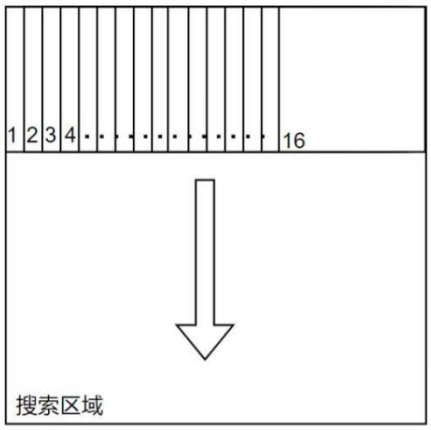
运动估计块匹配示意图

在进行运动估计时，首先在先前帧中，找到和参考块具有相同坐标的一个候选块。 其次，在该候选块附近确定搜索区域。以该候选块左上角为坐标原点，水平方向搜索范围为[-7,8]，垂直方向搜索范围为[-7,8]，则左上角横坐标在-7~8 范围内，纵坐标在-7~8 范围内的块称作候选块。由水平方向搜索范围以及垂直方向搜索范围构成的区间为搜索区间。之后，需要在搜索区间内寻找和参考块匹配程度最高的块，并以该块右上角坐标为运动向量(Motion Vector，简称 MV)。

1.3.2. 搜索方式及匹配准则

课程设计搜索方式采用全搜索块匹配算法(FSBM)，其通过对搜索区域进行穷尽式匹配比较来找到最佳的搜索匹配块。虽然 FSBM 算法计算量巨大，但由于其匹配效果好而在视频点播、数字电视等对视频图象质量要求很高的领域得到广泛的应用。

本课程设计利用并行处理，运算单元能同时完成 16 个匹配块的 sad 结果运算。运算单元的处理顺序如图 2。



运算单元处理顺序

在搜索区域内进行匹配时，匹配准则为参考块和候选块之间的像素亮度值的累计绝 对误差。每个块的大小为 8×8，搜索区域为[-7, 8]（水平、垂直方向均为[-7, 8]）。 令 x(i, j)表示当前帧在坐标（i, j）处的像素亮度值，y(i+m, j+n)表示先前帧在坐标（i+m, j+n）处的像素亮度值，则二者之间的向量为（m, n），且二者之

间的像素亮度值的累计绝对误差（SAD）可表示为：

$$\boldsymbol{SAD(m,n)=\sum_{i=0}^{7}\sum_{i=0}^{7}|x(i,j)-y(i+m,j+n)|} \qquad \boldsymbol{-7\le m,n\le 8}$$

在搜索区域内找到所有候选块的对应 SAD 值，其中对应最小 SAD 值的（m, n）即为该参考块的运动向量(Motion Vector)，即

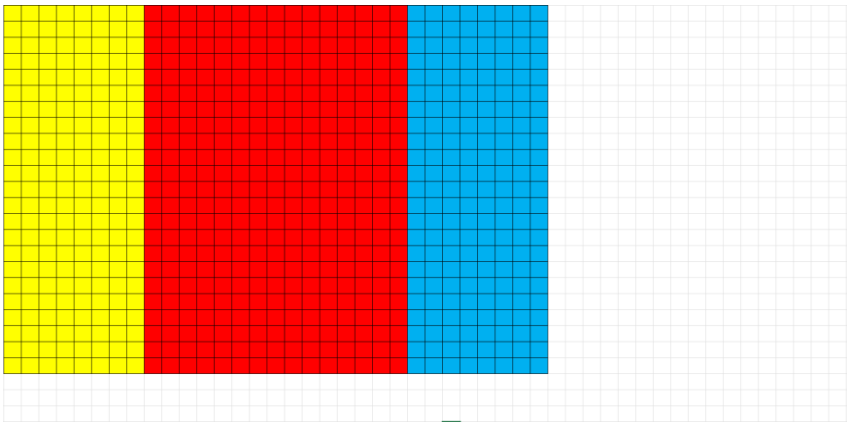
$$MV=\arg\{min\,SAD(m,n)\} \qquad \boldsymbol{-7\le m,n\le 8}$$

1.3.3. 数据复用

在进行全搜索匹配时，可以通过数据复用来提升 ME 电路的效率。

对于一帧 4k 图片，按照每块 8×8，搜索区间为[-7,8]，则可以划分为 480×270 个搜索区域，即 480 列、270 行。

为减小面积开销，本次课程设计仅实现了相邻搜索区域的数据复用，且仅对搜索区域的行进行了数据复用。采用如图 3 的数据复用方式。



数据复用方式

在某行中，红色和黄色区域是上一块搜索区域，红色和蓝色是下一块搜索区域，其中红色部分的数据可以进行数据复用。

2. 电路性能分析与电路结构设计

2.1. 电路性能分析

2.1.1. 工作频率

将电路工作频率预设为200MHz，后续性能分析都将在此基础上进行。

2.1.2. 实时处理能力

一帧画面中存在$(3840 \times 2160)/(8 \times 8) = 129600$个完全独立的$8 \times 8$当前帧参考块，每个参考块的搜索区间内存在$(7 + 8 + 8 - 8 + 1)^2 = 256$个候选块，假设每个SAD计算单元在一个时钟周期内可以完成一个SAD值的计算，于是有$(129600 \times 256 \times 60)/200\text{M} \cong 10$，即至少需要10个SAD计算单元同时工作才能满足实时处理要求。$\lfloor 256/10 \rfloor = 25$，即取最少10个SAD计算单元时，需要在25个时钟周期内完成关于一个参考块的所有计算，即计算出该参考块与所有候选块之间的SAD值并确定最佳候选块。

综上，确定使用16个SAD计算单元，且处理周期为25个时钟周期，以此满足4K@60fps视频的实时处理要求。

2.1.3. 吞吐率

若将吞吐率定义为单位时间内完成的工作量，并用计算出的SAD值表征工作量，则根据每处理周期内计算出256个SAD值，有吞吐率为：$(256 \times 200\text{M})/25 = 2.048 \times 10^9$。

2.1.4. 峰值带宽

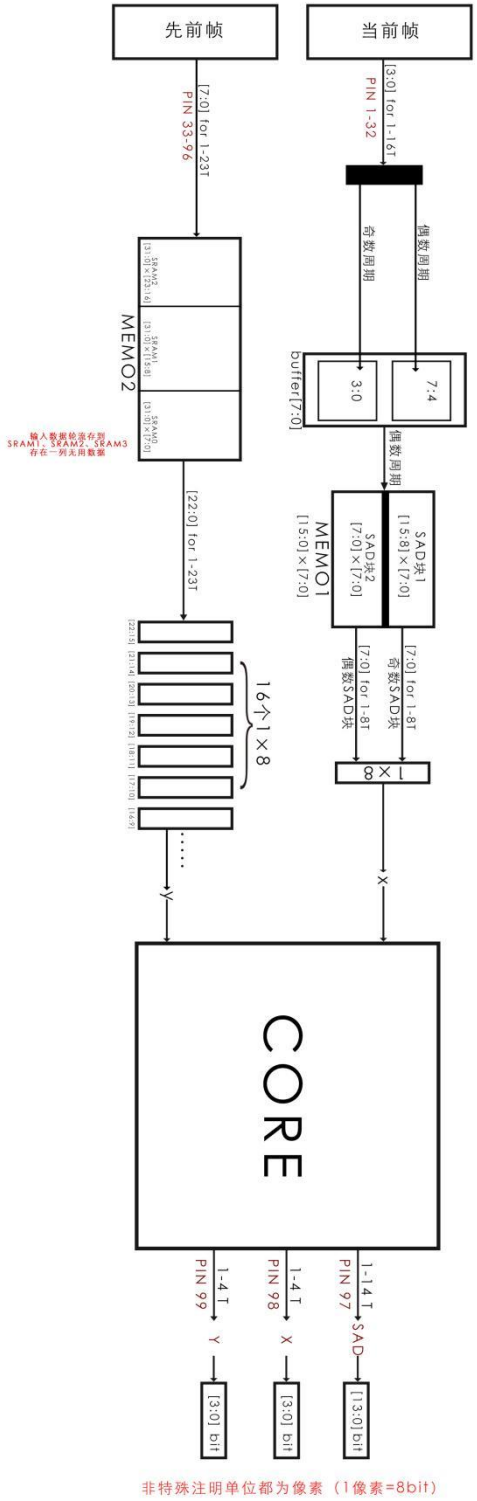
从I口向SRAM写入数据时，对于先前帧，每周期输入8B数据，对于当前帧，写入数据时每周期输入4B数据；通过O口向外输出信号时，最佳候选块的相关信号共有22b数据，输出最佳候选块坐标信号时存在每周期2b数据的峰值。由于存在先前帧和当前帧数据同时输入且结果数据同时输出的时钟周期，故峰值带宽为12.25B。

2.1.5. 平均带宽

从I口向SRAM写入数据时，对于先前帧，一个处理周期里前23个时钟周期每周期写入8B数据，其输入带宽利用率为23/25，对于当前帧，一个处理周期里前16个时钟周期每周期写入4B数据，其输入带宽利用率为16/25；通过O口向外输出信号时，在14个时钟周期内共输出14b的最佳候选块SAD值数据，在4个时钟周期内共输出8b的最佳候选块坐标数据，输出带宽利用率为8/25。总的平均带宽为（$23 \times 8 + 16 \times 4 + 2.75$）$/25 = 10.03$B。

2.2. 电路硬件结构与说明

2.2.1. 电路整体结构

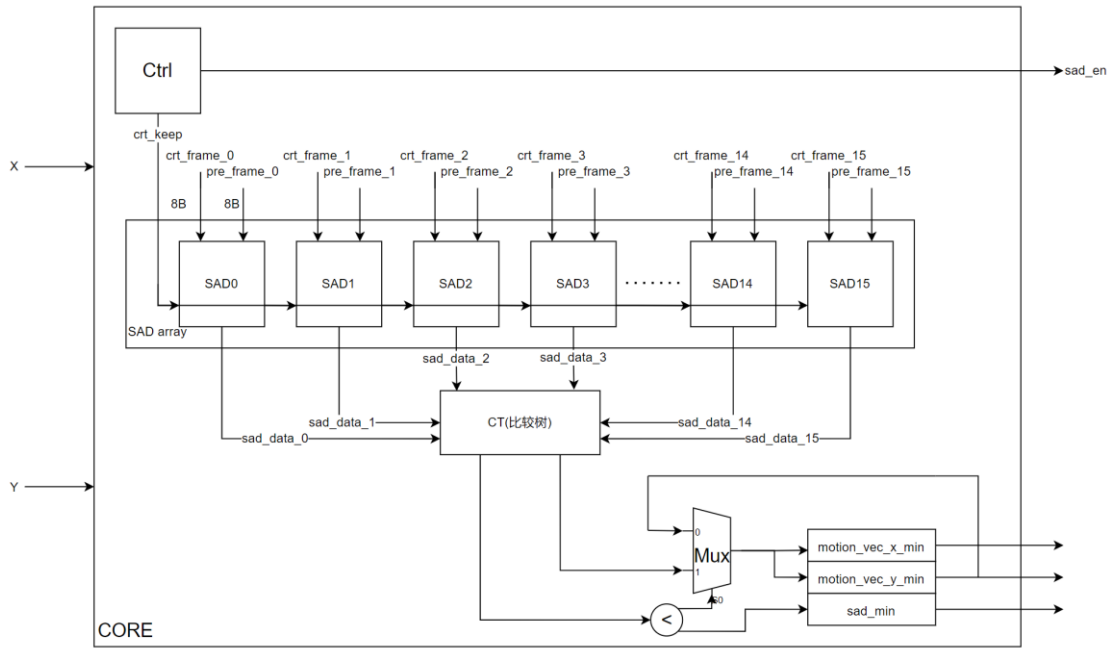


电路整体架构

电路整体架构分为计算核心（Core）和输入输出缓存（IO Buffer）两部分，前者进行SAD值的计算，搜索最佳候选块并输出运动向量以及最小SAD值；后者分别对输入文件中的先前帧和当前帧数据进行组织，以特定形式为计算核心提供计算所需数据，同时组织输出数据。

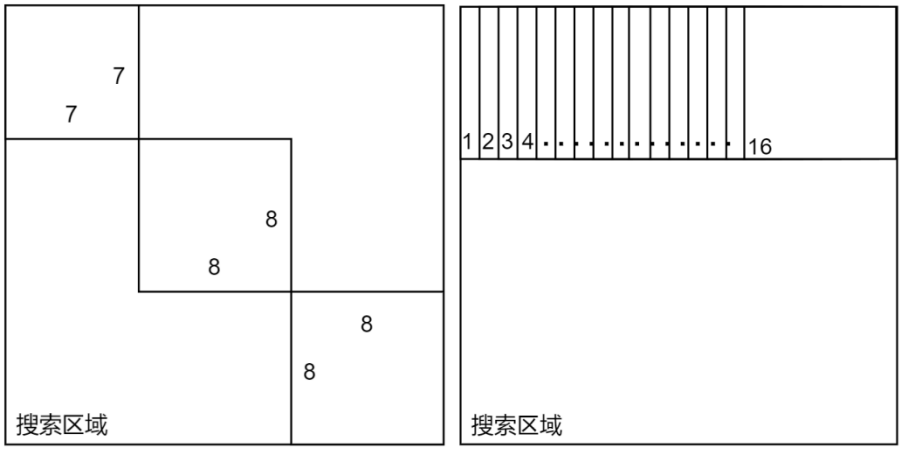
上图包含电路整体以及输入输出缓存架构，未显示计算核心内部细节。当前帧的输入占用 32 个 IO 接口，每次读入 4 个像素的数据，并创建一个 8bit 的 buffer 用于暂时存放不完整的当前帧像素信号。当 buffer 中的像素信号完整后，将其存储于 MEMO1 中，之后作为信号 X 输入计算核心；先前帧的输入占用 64 个 IO 接口，每次读入 8 个像素的数据，存储与 3 个 SRAM 组成的 MEMO2 中，并通过 SRAM 的交替实现先前帧像素数据的复用，之后组织为 16 个 64 位先前帧数据作为 Y 输入计算核心；最后将计算核心运算所得最佳候选块的 SAD 值和运动向量通过 3 个 IO 接口分别串行输出。

2.2.2. 计算核心总体架构



计算核心架构

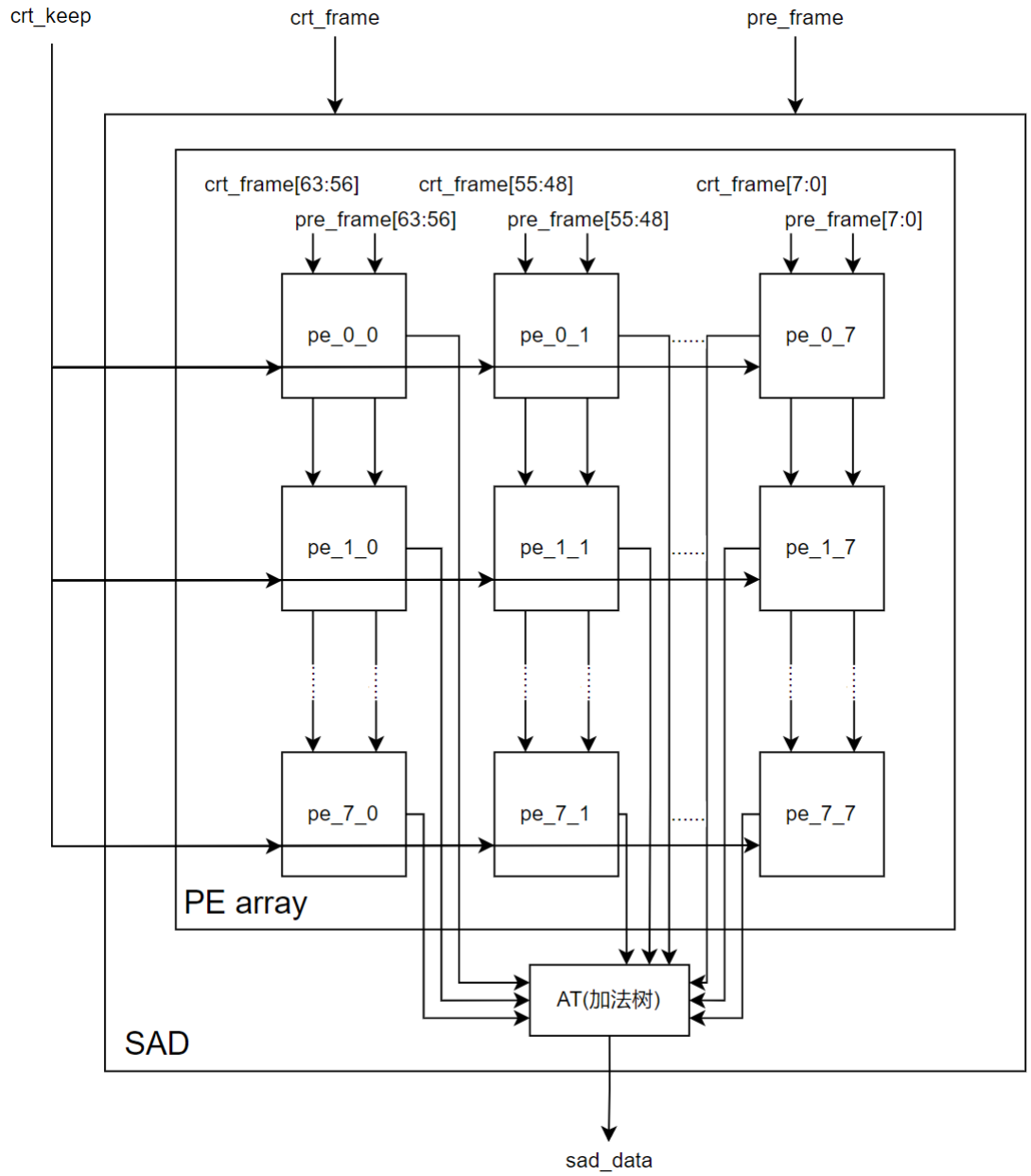
图2为计算核心的总体架构，16个相同的SAD计算单元在控制单元（Ctrl）的指示下进行内部数据传输或SAD值的计算，完成计算后将结果（sad_data）输入比较树（Compare Tree，简称CT）进行比较并输出最小的SAD值及其对应候选块的横纵坐标。



搜索区域示意图

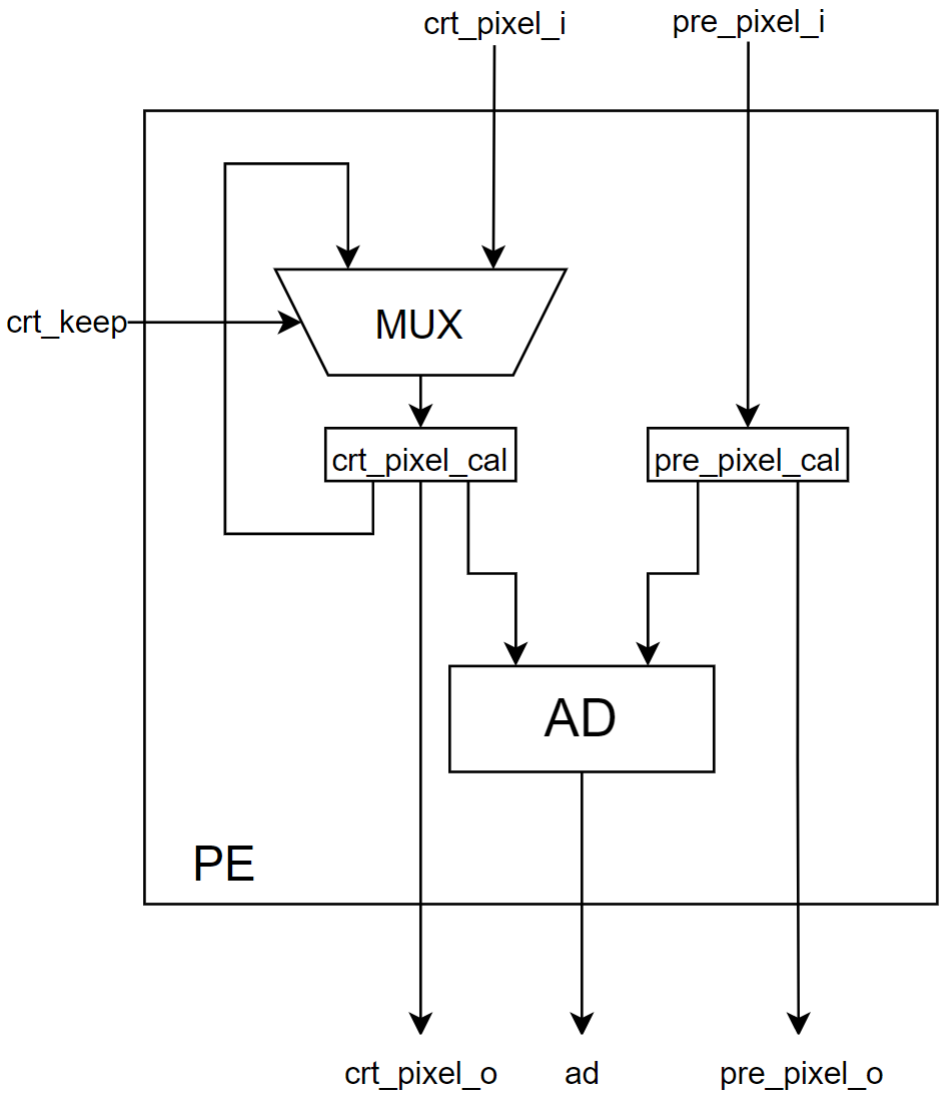
见上图，可知搜索区域在横向上存在16个8×8的候选块，与SAD计算单元数目匹配，因此在计算完“一行”的SAD值后，通过特殊的数据流动形式，将SAD计算单元在形式上整体下移，即可完成下“一行”SAD值的计算，下移15次后即可完成整个搜索区域的计算。

2.2.3. SAD计算单元架构



SAD计算单元架构

SAD计算单元的内部结构如上图所示，由64个处理单元（Process Element，简称PE）和加法树（Adder Tree 简称AT）组成。处理单元在形式上构成8×8的阵列，各列之间相互独立，列内部在crt_keep信号的控制下进行像素数据的传递与AD值（当前帧与先前帧相应点位像素信号差的绝对值）的计算，计算结束后经由加法树进行加和得到SAD值进行输出，并将SAD计算单元进行形式上的下移，计算下一个SAD值。



处理单元架构

图6显示了处理单元的结构，寄存器crt_pixel_cal和pre_pixel_cal分别用于存放当前帧和先前帧对应点位的像素信号，前者在 crt_keep 信号的指示下进行数据传递或锁存，并在锁存后通过AD计算单元判断大小后做差得到差的绝对值并输出。

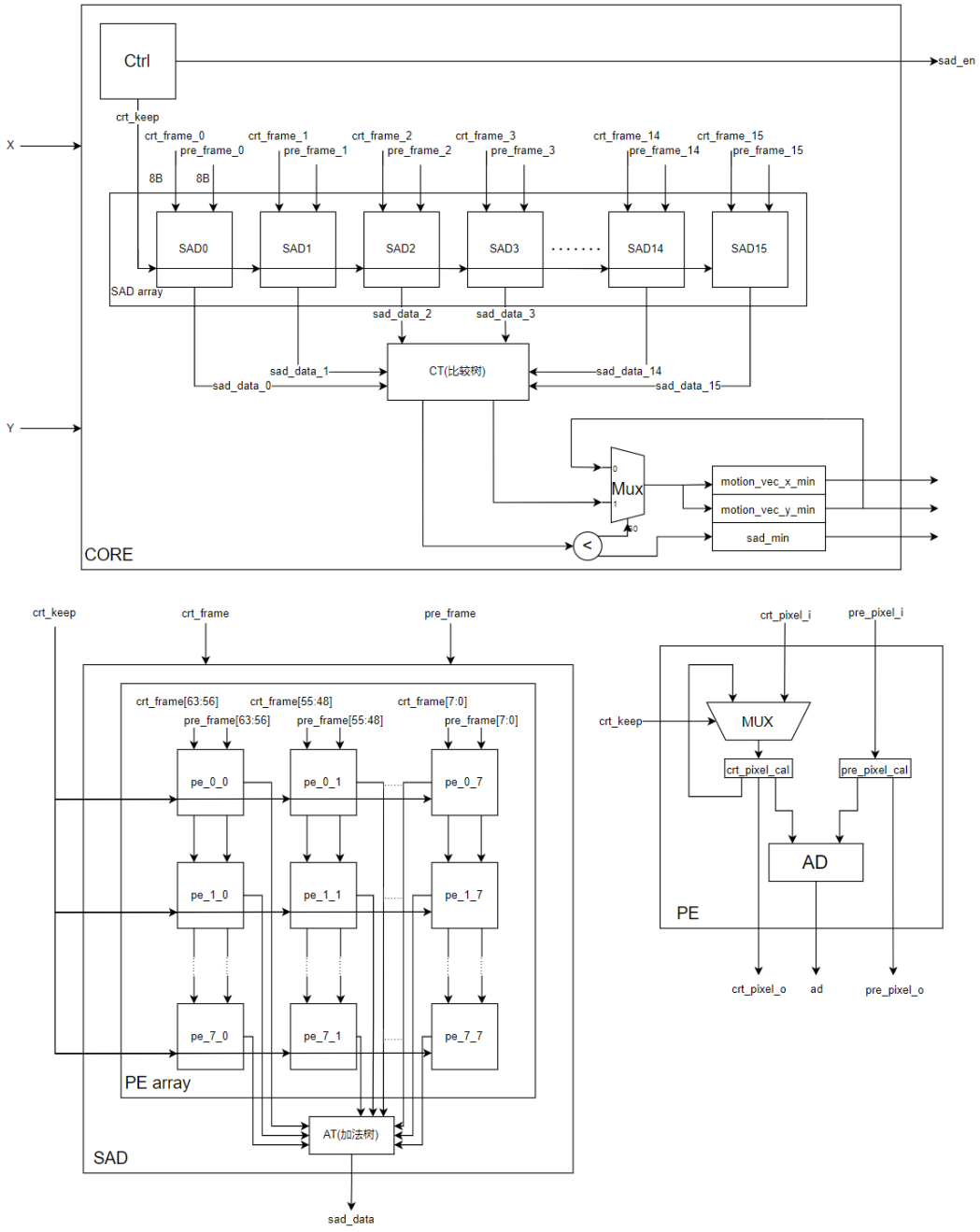
3. 电路 RTL 模型与仿真验证

3.1. RTL 模型

3.1.1. 总体设计

总体上模型由两个主要部分构成，一部分是计算核心，另一部分是 IO 与核心之间数据传输的 buffer。具体的架构图见前文电路结构设计部分。总体的数据流动是从 INPUT 口读入数据，输入数据形式包括 32bit 和 64bit 两种，32bit 是当前帧数据，64bit 是先前帧数据，读入方式类似于 FIFO 的方式，chip 对外不会传递地址值，只向外传递是否要读取数据。数据进入 chip 之后先存入 buffer 中，然后从 buffer 按既定规则给计算核心数据，经过计算得到最小 SAD 值和对应的 x、y 坐标值之后输入到输出模块，然后对结果进行一定处理之后 SAD、vec_x、vec_y 分别通过 OUTPUT 口串行输出到片外。

3.1.2. CORE 架构



CORE 运行方式概括来讲就是：以 25 个时钟周期为一个处理周期，前 8 个周期每个周期输入一个 64 位的当前帧数据 X，同时赋值给 crt_frame_0 至 crt_frame_15，经过内部的数据传递后，SAD 计算单元将计算所需的当前帧像素数据锁存；前 23 个周期每个周期输入 16 个 64 位先前帧数据的 Y，分别依次赋值给 pre_frame_0 至 pre_frame_15，前 8 个周期结束后 SAD 计算单元内也存储了用于计算的先前帧像素数据，于是在第 9 个周期就可以开始进行 SAD 值的计算，由于先前帧数据在前 23 个周期内持续输入并向下传递，于是所有候选块第一行

的像素数据会被丢弃而搜索区域下一行的像素数据会进入处理单元中，因此可以认为 SAD 计算单元在形式上进行了下移，并在下一周期完成下一行候选块 SAD 值的计算，于是在 23 个周期内可以完成一个参考块所有 SAD 值的计算。然后经过 2 个周期完成运算之后，将 sad_min、运动向量 vec_x_min 和 vec_y_min 输出。CORE 顶层 RTL 实现在 core.v 中

CORE 中并行设计了 16 个 SAD 运算单元，对于当前帧的每一个 8×8 像素块需要进行 256 次 SAD 运算，通过上述数据传递方式在一个处理周期内完成一个 8×8 当前帧像素块的运算。16 个 SAD 运算单元外接比较树，对每个周期算出的 16 个 SAD 值进行比较得到最小值，比较完成后再与上一次的最小值比较，得到至此为止的最小值和对应运动向量，完成所有候选块的计算之后，输出 sad_min 和运动向量 vec_x、vec_y。SAD 运算单元的 RTL 实现在 sad.v 中，比较树的 RTL 实现在 compare_tree.v 中

每个 SAD 由 64 个处理单元以及加法树组成，由于 SAD 运算单元每次要处理 8×8 个像素，因此每个处理单元负责处理 1 个像素，处理单元内部的寄存器 crt_pixel_cal和pre_pixel_cal分别用于存放当前帧和先前帧对应点位的像素信号，前者会在控制信号crt_keep的指示下选择向下流动或锁存，实现当前帧像素信号在前8个时钟周期内的向下传递，后者持续接收输入信号并向外输出存放的像素信号，实现先前帧像素信号的持续向下传递。最后在crt_pixel_cal 信号锁存后通过AD计算单元判断大小后做差得到差的绝对值并输入加法树进行求和得到 SAD 值。处理单元的实现在 pe.v 中

3.1.3. IO buffer

IO buffer 部分主要由三个模块组成：当前帧输入、先前帧输入以及结果输出。接下来分别详细介绍各个模块的设计和实现。

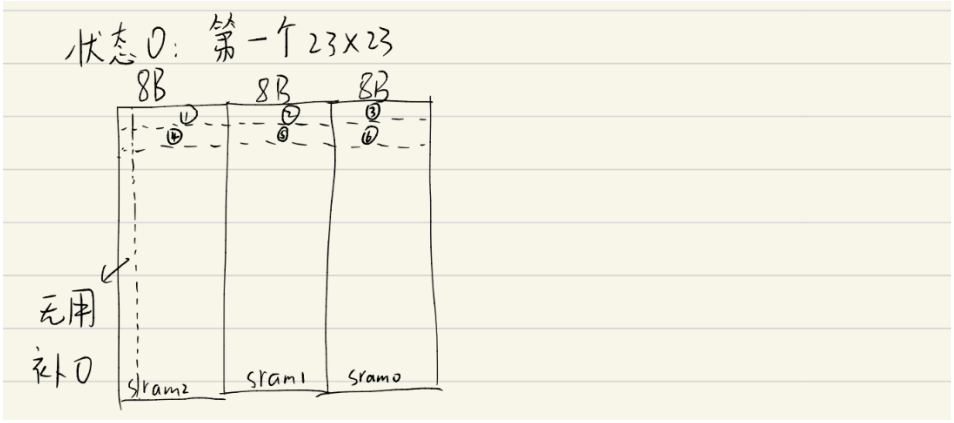
3.1.3.1. 当前帧输入

当前帧的输入占用了 32 个 IO 接口，每次读入四个像素的数据。对于当前帧的读取顺序是自上而下一行一行自左至右地读 8×8 的像素块，比如第一行就是 8×3840 个像素，共有 480 个 8×8 的像素块。由于每个周期只能读入 32 个像素，因此每个 1×8 的像素行要经过 2 个周期才能读入，因此创建一个 8bit 的 buffer，每两个周期 buffer 向 MEMO1 发送一个完整的 1×8，也就是经过 16 个周期才能将完整的 8×8 像素块存在 MEMO1 中。在每一行的开始需要经过一个准备阶段（主要是为先前帧服务），准备阶段中，当前帧模块将此行第一个 8×8 的像素块存在 MEMO1 中，为后续运算作准备，当核心开始运算之后每一个 25T 的运算周期的前 16 个周期都会传入下一个 8×8 像素块，因此 MEMO1 大小为 16×8，可存储两个 8×8 像素块。在每个运算周期的前 8 个周期，MEMO1 向核

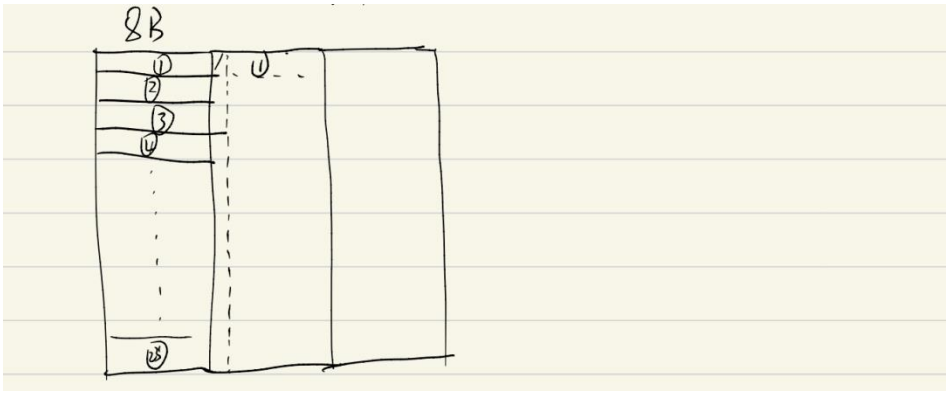
心每周期传 8 个像素的数据。当前帧输入模块的 RTL 实现在 cur.v 中

3.1.3.2. 先前帧输入

先前帧输入占用了 64 个 IO 接口，每次读入 8 个像素的数据，先前帧读取顺序：以 23 行像素为 1 行，每往下 1 行就下移 8 个像素，比如第一行是前 23 行像素，第二行就是 9-31 行像素。每一行从左往右读取，第一次读取 23×23 的像素块，之后每次读取 23×8 的像素块，即可以复用上一个 23×23 像素块的后 15 列。然后行的切换时由于没有利用复用，因此需要一个准备时间，在准备时间中，要花 69 个周期读入 23×24 的像素块，之后的每个 25T 的运算周期的前 23 个周期读入 23×8 的像素块。在准备阶段读入 23×24 的像素块的原因在于 MEMO2 的结构。MEMO2 由 3 个 24×8 的 SRAM 构成，因此为了方便第一次就直接读入 23×24 的像素块，但是第一列的数据是无效的，利用一个指示变量标记哪一个 SRAM 中的第一列无效。每一行启动之后每一次新的 23×8 像素块的依次存入 SRAM2、SRAM1、SRAM0 中。



第一个 23×23 以 1×8 像素行为单位传送顺序



后续 23×8 以 1×8 像素行为单位传送顺序

MEMO2 的输出和输入是同时进行的，每个周期会读入 8×1 像素行，加上复用，这样就使得 23×1 的像素行可以输出，达到了核心要求的每个周期传 23 个

像素（后续做处理得到 16 个 8×1 像素行）的要求，每个 25 T 的运算周期的前 23 个周期都在给核心传数据。先前帧输入模块的 RTL 实现在 pre_frame_buffer.v 中

3.1.3.3. 结果输出

结果的输出逻辑上相对简单，当核心给到计算完成的信号之后，使用寄存器存住核心传过来的 sad_min 和运动向量 vec_x、vec_y,然后将运动向量减 7 得到[-7,8]范围的运动向量，然后将最小 SAD 值和运动向量串行输出到片外，由于较大的 SAD 也仅仅只需要 14bit 的寄存器，也就是说只要 14 个周期就能传到片外，远远小于 25，所以 IO 接口数 3 个即可。其中还存在一个小的优化，由于运动向量存在正负的问题，因此需要 4 位寄存器来存储，但是由于 8 无法用 4 位存储（会变成-8），为了避免因为一个值额外增加一位，通过 testbench 中检测到-8 自动识别为 8 来解决这个问题。结果输出模块的 RTL 实现在 result.v 中

当前帧和先前帧的输入模块、结果输出模块和计算核心整合在一起通过一个顶层文件调用，当前帧除了第一块之外是否输入由先前帧的输出信号控制，通过顶层文件将所有模块的时序对准，正确完成功能，此外由于核心需要先前帧给 16 个 8×1 像素行，因此顶层文件还将先前帧模块输出的 23×1 的像素行切分为[23:16]、[22:15]……[7:0]的 16 个 8×1 像素行。顶层文件为 top.v

3.2. 电路验证方法

首先对于各个模块（当前帧输入、先前帧输入、计算核心）首先各自编写 testbench 验证功能的正确性，比如当前帧和先前帧，准备了一些输入数据，然后检验输出数据是否与输入数据相同，计算核心则是准备了第一个 SAD 块的当前帧和先前帧像素数据，然后计算出最小 SAD 和运动向量后和期望结果作比较来验证运算正确性，结果输出模块因为是一个非主要模块，未单独测试。模块单独测试没有涉及到模块之间时序上是否对准的问题，也无法验证最终结果的正确性，因此在模块单独测试之后，编写了顶层文件，然后编写了 testbench 对整体系统功能正确性进行验证。Testbench 的验证方式是将所给的 4k 数据从文件中读出作为系统的输入，然后将从结果输出模块中输出的结果写入 txt 文件然后与期望输出结果作比较验证是否正确，另外，通过 reset 置 0 后再置 1 观察输出验证 reset 功能的正确性。

3.3. 验证平台搭建

由于各模块单独验证的 testbench 较简易且作用比较小，因此这里只介绍整体 testbench 的搭建过程。

1. 老师提供的 4K 数据文件的数据顺序与我们设计的顺序有所不同，因此首先利用 python 进行脚本的生成，将输入数据改为我们期望的顺序，并且改为二进制且当前帧改为 32bit 一行先前帧改为 64bit 一行，脚本见 cur.txt 和 ref.txt
2. 设置所需的一些寄存器变量，设置时钟频率为 200MHz，读写文件初始化，reset，然后调用 ME 模块。
3. 根据先前帧输入模块和当前帧输入模块所给的读数据信号控制是否从 4K 文件中读数据
4. 设置一个计数器，当 ME 模块给 testbench 输出信号时开始计数，因为 sad 为 14 位，因此 count 计数到 14 后置 0
5. 当 ME 模块给 testbench 输出信号时，sad 和运动向量开始一位一位的接收，当满 4 位时运动向量停止接收并写到文件中，其中当接收到-8 时转化成 8 写到文件中。当满 14 位时，sad 停止接收并写到文件中。
6. 运行 10000ns 之后，reset 拉低，然后再拉高，写文件继续不变，读文件从头开始，验证 reset 是否到达系统重新开始运行的功能。
7. 测试文件为 tb.v

3.4. 验证结果

3.4.1. 功能点验证

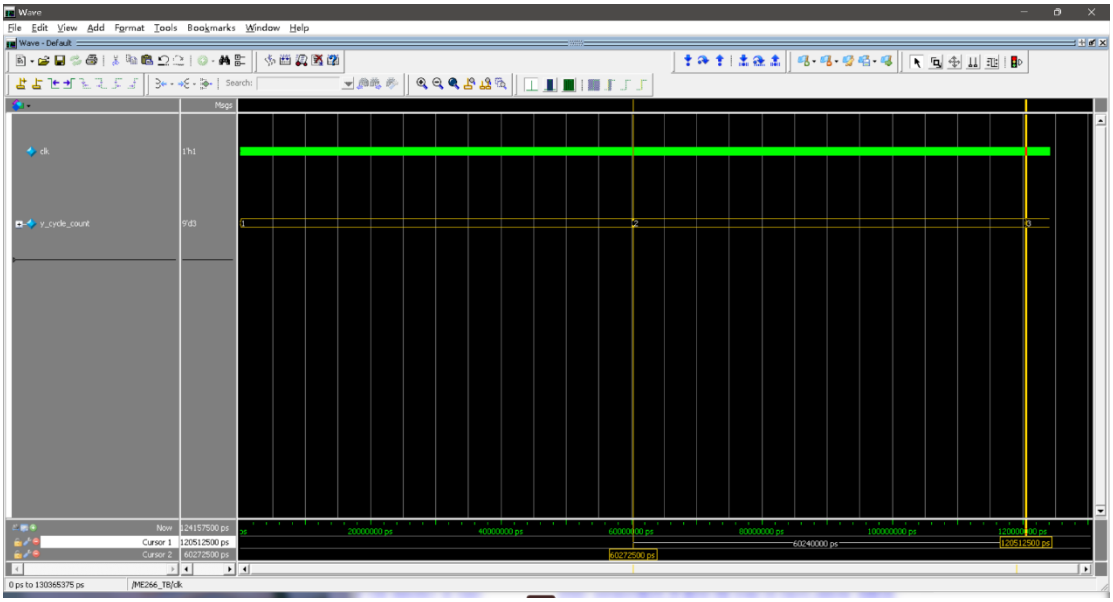
ME 的功能其实比较单一，就是计算一帧 4K 画面的 sad 以及得到运动向量，其功能的正确性全部来源于计算结果或者说输出结果的正确性上。因此我们尽可能多的验证 SAD 计算值的正确。对于我们的设计而言，最需要关心的 SAD 值是第一个（第一个错说明计算方式或者数据流整体存在问题或者仅仅准备阶段的处理存在问题），第二三个（如果错说明准备阶段之后的处理存在问题），以及从第二行开始每一行的开头（如果错说明换行时的操作存在问题），我们仿真失败的点也主要集中在这几个地方。除了 ME 模块功能的正确性验证之外，对 reset 的重启功能也进行了验证，所有的验证结果输出到了 sad_out.txt、x_out.txt、y_out.txt 三个文件中

3.4.2. 测试向量

虽然前文提到了几个比较重要容易出错的功能点，但是在做仿真验证时，测试向量仍是使用了老师提供的 4K 文件中的所有数据外加一次 reset，因此在验证的严谨性这方面不存在问题，我们验证了所能够验证的所有数据，将结果与老师提供的期望结果对比，结果完全一致，说明了我们功能的正确性。

3.4.3. 仿真性能

a. 时钟频率达到 200MHz
b. 当前帧输入带宽为 32，先前帧输入带宽为 64，输出带宽为 1，加上输入输出的控制信号仅输入输出共用 IO 接口 102 个
c. 当前帧输入带宽利用效率为 16/25，先前帧输入带宽利用率为 23/25，输出带宽利用率为 8/25（由于输出带宽很小所以并未实现很高的利用率）
d. 实时性：如下图在 200M 的时钟频率下，每一行的处理时间是 0.06024ms，共 270 行，即每一帧的处理耗时 16.2648ms，即 1s 可以处理 **61.48 帧**，达到了实时性的要求，在后续综合中，始终周期提高到了 5.08ns，经换算在这种条件下，1s 可以处理 **60.51 帧**，仍然满足实时性要求



4. 电路逻辑综合策略与综合结果

4.1. 逻辑综合流程

4.1.1. 环境设置

在综合前我们需要添加逻辑综合所需的 link library、target library、symbol library 库文件和 RTL code 中使用的 SRAM 的库文件，以及他们对应的搜索路径。同时将 SRAM 库的.db 文件拷贝至搜索路径下。

```
#Library Setup
set search_path "$search_path ../rtl/idct ../scripts /home/student/Desktop/Workspace/55nm ../work ../mem/asdrlspkb1p64x16cm2sw0/tt1p2v25c"
set target_library   "hu55npkldut_tt1p0v25c.db HL55LPGP3VDS_SL_A01_P2_TT1D8V1D2V25C.db asdrlspkb1p64x16cm2sw0_lib.db"
set symbol_library   "hu55npkldut.sdb"
set link_library     "* $target_library $symbol_library pre_sram.db cur_sram.db"
```

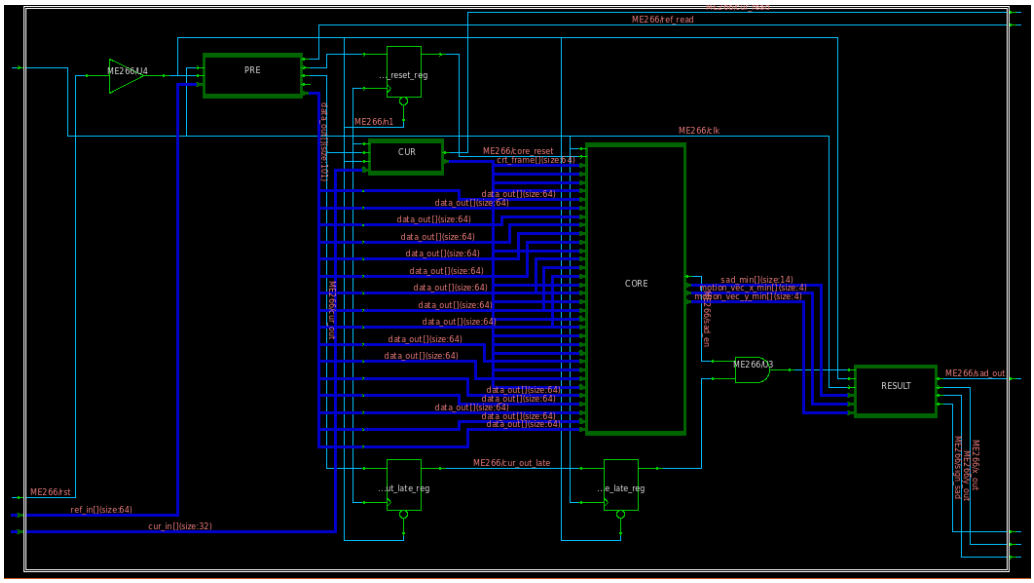
4.1.2. 门级网表

在生成门级网表前，我们需要对 RTL code 顶层文件进行 I/P PAD 的封装，即在 me266_chip.v 中例化顶层文件和所有的 I/O PAD（调用 HPDWUW1416DGP 库）。

完成封装文件的编写后，我们需要将所有的代码模块拷贝至对应 trl 代码路径下，并在 dc 中通过 analyze 函数读入文件。应当注意的是，各模块应该按照从底层到顶层的顺序读入。按照模块调用顺序，我们依次读入当前帧和先前帧的输入 buffer、运算模块、输出 buffer、顶层文件和封装文件。

```
analyze -format verilog ../rtl/me266/cur.v
analyze -format verilog ../rtl/me266/pre_frame_buffer.v
analyze -format verilog ../rtl/me266/pe.v
analyze -format verilog ../rtl/me266/sad.v
analyze -format verilog ../rtl/me266/compare_tree.v
analyze -format verilog ../rtl/me266/core.v
analyze -format verilog ../rtl/me266/result.v
analyze -format verilog ../rtl/me266/top.v
analyze -format verilog ../rtl/me266/me266_chip.v
elaborate me266_chip
```

完成以上步骤后，通过 elaborate 函数得到门级网表。由于逻辑综合对代码规范更为严格，对于网表生成中出现的带宽 mismatch、端口悬空、无用寄存器等报错需要对 RTL code 进行修改。最终生成的门级网表可视化如下：



4.1.3. 约束添加

在这个环节我们需要为逻辑综合添加时钟、环境、设计规则等方面的约束。

在时钟约束上，我们以根据设定的 200MHz 时钟频率对如下参数进行了设置。其中 clock period 原为 5ns，后序在物理设计环节调整为 5.08ns。

clock period	5.08
clock transition	0.1
clock uncertainty	0.25
clock latency（source）	4
clock latency	2
input delay（max）	2.5
output delay（max）	2.5

在环境约束上，我们对 wire load model、input driver 和 output capacitance 进行了如下约束。

set_wire_load_mode segmented
set auto_wire_load_selection true
set_driving_cell -lib_cell HPDWUW1416DGP -pin PAD [all_inputs]
set_load [load_of HL55LPGP3VDS_SL_A01_TT1D8V1D2V25C/HPDWUW1416DGP/PAD] [all_outputs]

在设计规则上，我们设置了最大 transition 和 fanout。

max transition	3.0
max fanout	32

此外，我们还将 rstn 和 clk 信号设置为理想网络、将 I/O PAD 设置为 don't touch 格式。

```
set_ideal_network {rst}
set_ideal_network {clk}
set_dont_touch [get_cells "HPDWUW1416DGP*"] true
set_fix_multiple_port_nets -all -buffer_constants
```

最后，我们通过 compile 命令将网表与工艺节点进行映射，逻辑综合完成。

4.2. 综合结果

4.2.1. 时序结果

通过 report timing 命令我们可以得到综合后的时序报告。报告显示，关键路径位于 core 运算模块中，与预期相符。Data required time 为 10.79ns，data arrival time 为 10.79ns，裕度为 0，满足综合要求。

```
clock clk (rise edge)                                    5.08       5.08
clock network delay (ideal)                              6.00      11.08
clock uncertainty                                       -0.25      10.83
ME266/CORE/sad_min_reg[10]/CK (SVL_FDPSBQ_1)             0.00      10.83 r
library setup time                                      -0.04      10.79
data required time                                                 10.79
--------------------------------------------------------------------------
data required time                                                 10.79
data arrival time                                                 -10.79
--------------------------------------------------------------------------
slack (MET)                                                        0.00
```

4.2.2. 面积结果

通过 report area 命令我们可以得到综合后的面积报告。逻辑综合无法给出 Net Interconnect 的面积，除此之外芯片的 Total cell area 为 917576。

```
Combinational area:             249619.993078
Buf/Inv area:                    30118.199150
Noncombinational area:          149239.718976
Macro/Black Box area:           518717.009766
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                917576.721819
Total area:                 undefined
```

4.2.3. 功耗结果

通过 report power 命令我们可以得到综合后的功耗报告。其中 Cell Internal Power 为 52.8891 mW， 占比 21%；Net Switching Power 为 197.7956 mW ，占比 79%；Total Dynamic Power 为 250.6846 mW 伴随 119.4131 uW 的 Cell Leakage Power，占比极小。Switching power 由于 I/O PAD 面积较大难以优化 ，我们试图通过减小 IO buffer 内部的寄存器数量来优化 Internal Power，但效果并不明显。

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	16.0369	197.7241	1.2076e+03	213.7629	(85.23%)	
memory	18.1527	1.0406e-02	8.6537e+03	18.1718	(7.25%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	18.6649	1.6577e-02	2.7646e+04	18.7078	(7.46%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	3.5703e-02	4.3893e-02	8.1904e+04	0.1615	(0.06%)	
Total	52.8903 mW	197.7950 mW	1.1941e+05 nW	250.8040 mW		

4.2.4. 资源使用结果

资源使用主要包括端口、节点、组合逻辑单元、时序逻辑单元、buf/inv 等的使用量，我们可以在面积报告中查询使用情况。

```
Number of ports:                    110355
Number of nets:                     226990
Number of cells:                    102707
Number of combinational cells:       82095
Number of sequential cells:          16989
Number of macros/black boxes:          108
Number of buf/inv:                   32789
Number of references:                    4
```

4.2.5. 设计违例

通过 report constraint all violators 命令我们可以得到综合后的设计违例报告。报告中主要提示了部分路径过短，未达到最小延迟的要求。该部分违例并不影响逻辑综合的时序报告，且将在物理设计中进行修正。

```
****************************************
Report : constraint
        -all_violators
Design : me266_chip
Version: O-2018.06-SP1
Date   : Wed Apr 13 11:40:23 2022
****************************************


   min_delay/hold ('clk' group)

                            Required        Actual
   Endpoint                 Path Delay     Path Delay        Slack
   -------------------------------------------------------------
   ME266/CUR/dua/DA[0]          6.41           6.15 f        -0.25  (VIOLATED)
   ME266/CUR/dua/DA[1]          6.41           6.15 f        -0.25  (VIOLATED)
   ME266/CUR/dua/DA[2]          6.41           6.15 f        -0.25  (VIOLATED)
   ME266/CUR/dua/DA[3]          6.41           6.15 f        -0.25  (VIOLATED)
   ME266/CUR/dua/DA[4]          6.41           6.15 f        -0.25  (VIOLATED)
```

4.2.6. 设计检查

通过 design check 命令我们可以得到综合后的检查报告。报告中未有报错，

```
   ****************************************
   check_design summary:
   Version:     O-2018.06-SP1
   Date:        Wed Apr 13 23:15:10 2022
   ****************************************

                      Name                                                    Total
   ------------------------------------------------------------------------------------
   Inputs/Outputs                                                            7084
       Multiply driven inputs (LINT-6)                                         98
       Unconnected ports (LINT-28)                                           6986

   Cells                                                                      649
       Nets connected to multiple pins on same cell (LINT-33)                 649
   ------------------------------------------------------------------------------------
```

代表逻辑综合成功。报告中指出了一些 warning，下面对其进行分析：

其一，input PAD 方向可能错误。原因：一方面，IO 中的 PAD 同时具有 input/ouput 两个方向的属性，因为对输入来说用作 input，对输出来说用作 output。由于有 output 的属性，工具认为可能有问题。另一方面，添加 set_driving_cell -lib_cell HPDWUW1416DGP -pin PAD [all_inputs]约束时，driving cell 上的 design rule 加到了 input port 上，（例如 driving cell 自己可能有 max cap 之类的 rule）input 应该是输入，但除了 input port 的驱动，还有个 output pin PAD 也连着它。输入是不应该有多驱动的，所以有 warning。该 warning 与设计无关。

其二，pe 和 sad 模块部分例化存在未连接端口。原因：模块中使用移位寄存器，其最后一位用于抛出数据不做使用，并非设计错误。

其三，存在结点连接了 core 模块中的多个 pin。原因：为了减小芯片内部带宽，先前帧 buffer 对 core 的输出从 1 多个端口一一对应多个计算模块改为一个端口分别输出到不同计算模块。故先前帧一对多输出是内部带宽优化的结果，非

设计错误。

5. 电路物理实现与结果分析

5.1. Initialization

根据 ICC 仿真的要求，首先建立 link library、target library 这些逻辑资源，并 link 到 physical library，随后导入 DC 中生成的.v 文件与.sdc 约束。

5.2. Floorplan

```
create_floorplan -core_utilization 0.8 -left_io2core 30.0 -
bottom_io2core 30.0 -right_io2core 30.0 -top_io2core 30.0
```

```
insert_pad_filler -cell {HPFILLER5 HPFILLER1 HPFILLER05
HPFILLER01 HPFILLER0005}
```

建立芯片版图，放置 IO 引脚，并填充 filler，由于芯片为 IO 受限型芯片，因此初始 uitilization 的设置并不重要。

```
move_objects -to {150 1070} [get_cells ME266/PRE/sram2]
```

```
move_objects -to {150 1010} [get_cells ME266/PRE/sram1]
```

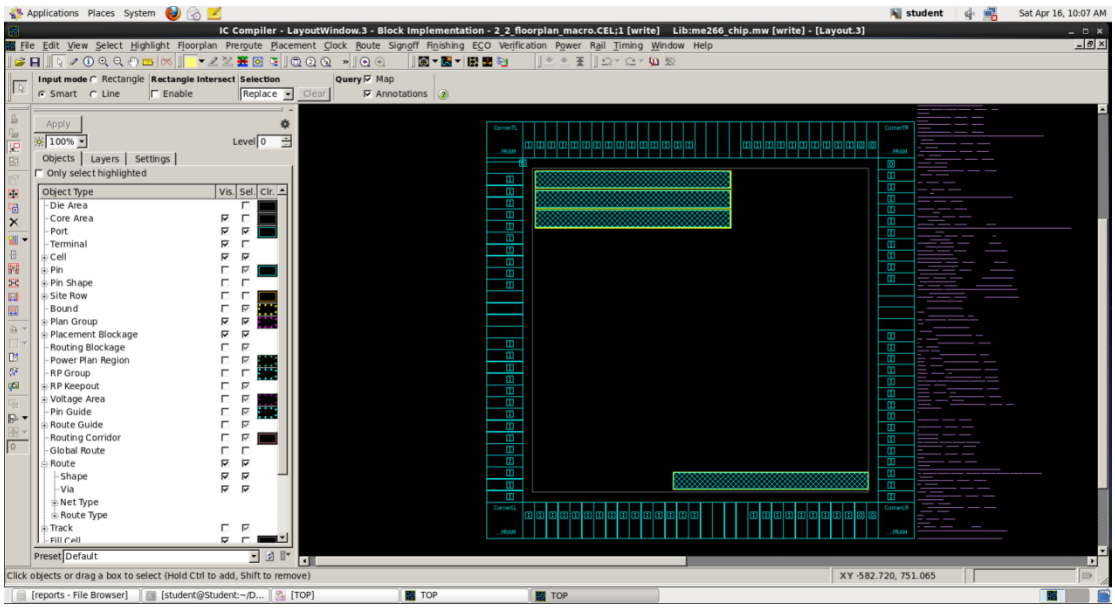
```
move_objects -to {150 950} [get_cells ME266/PRE/sram0]
```

```
move_objects -to {570 150} [get_cells ME266/CUR/dua]
```

合理放置 marco 元件，在本芯片中为四块 sram：其中 3 块用于缓存先前帧数据，按照 IO 口的排布，将其放在芯片左上角位置；剩下一块缓存当前帧数据，将其放在芯片右下角位置。

```
create_placement_blockage -coordinate {{147.000 946.400}
{747.000 1121.400}} -name placement_blockage_0 -type hard
```

注意，这里为了防止 EDA 将 standard cell 布置在 sram 的缝隙中，导致跨 sram 的连线具有不合理的长度，难以收敛，需要为先前帧 sram 划定一块禁止布置的区域，芯片概貌如下图所示：

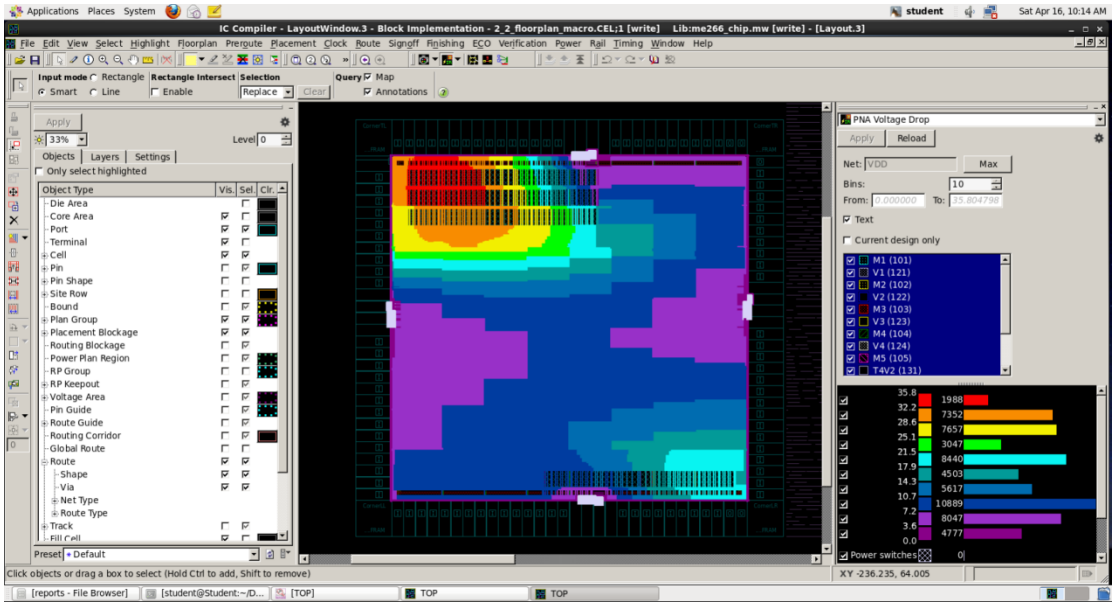


preroute_instances

preroute_standard_cells -fill_empty_rows -remove_floating_pieces

analyze_fp_rail -nets {VDD VSS} -voltage_supply 1.0 -pad_masters {HPVDD1DGP HPVSS1DGP} -analyze_power

定义电源电压为 1.0V，通过预绕线对电压降进行分析，电压降分布，如下图所示：



可以发现最大电压降集中在 sram 块部分，为 35.8mV，不超过 VDD 的 5%，可以认为电压降不会影响到芯片正常工作。

结束 floorplan，报告时序如下所示，最大违例为 1.72s，经过路线查询，发现是有几个小扇出节点有较大的线电容所致，因此暂时忽略，在后续布线时修复。

```
报告文件：floorplan.timing
Startpoint: ME266/CORE/sad_10/sad_data_reg_1_
                    (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ME266/CORE/motion_vec_x_min_reg_1_
                  (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                    Incr          Path
  ---------------------------------------------------------------------------
  clock clk (rise edge)                                    0.00          0.00
  clock network delay (ideal)                              6.00          6.00
  ME266/CORE/sad_10/sad_data_reg_1_/CK (SVL_FDPRBQ_1)      0.00 #        6.00 r
  …
  ME266/CORE/motion_vec_x_min_reg_1_/D (SVL_FDPRB_1)       0.00 *       12.49 r
  data arrival time                                                     12.49

  clock clk (rise edge)                                    5.08          5.08
  clock network delay (ideal)                              6.00         11.08
  clock uncertainty                                       -0.25         10.83
  ME266/CORE/motion_vec_x_min_reg_1_/CK (SVL_FDPRB_1)      0.00         10.83 r
  library setup time                                      -0.06         10.77
  data required time                                                    10.77
  ---------------------------------------------------------------------------
  data required time                                                    10.77
  data arrival time                                                    -12.49
  ---------------------------------------------------------------------------
  slack (VIOLATED)                                                      -1.72
```

5.3. Placement

place_opt

place_opt -effort high

route_zrt_global

place_opt_feasibility -skip initial_placement

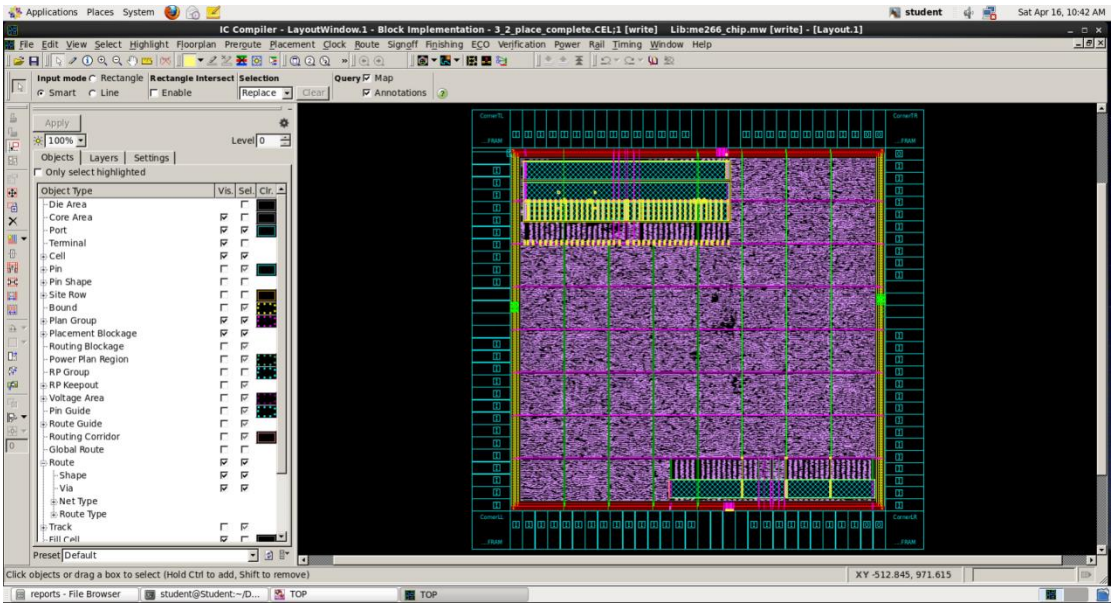
route_zrt_global

利用三次渐进的布局优化指令，逐步收束时序，最终可以达到如下所示的时序，仅违例 0.02ns，完全可以认为能在布线时解决违例。

报告文件：placement.timing

```
Startpoint: ME266/CORE/sad_10/pe_02/crt_pixel_cal_reg_0_
                    (rising edge-triggered flip-flop clocked by clk)
  Endpoint: ME266/CORE/sad_10/sad_data_reg_11_
                  (rising edge-triggered flip-flop clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                                    Incr          Path
  ---------------------------------------------------------------------------
  clock clk (rise edge)                                    0.00          0.00
  clock network delay (ideal)                              6.00          6.00
  ME266/CORE/sad_10/pe_02/crt_pixel_cal_reg_0_/CK (SVL_FDPHRBQ_1)
  …
  ME266/CORE/sad_10/sad_data_reg_11_/D (SVL_FDPRBQ_1)      0.00 *        10.76 r
  data arrival time                                                      10.76

  clock clk (rise edge)                                    5.08          5.08
  clock network delay (ideal)                              6.00         11.08
  clock uncertainty                                       -0.25         10.83
  ME266/CORE/sad_10/sad_data_reg_11_/CK (SVL_FDPRBQ_1)
                                                           0.00         10.83 r
  library setup time                                      -0.09         10.74
  data required time                                                     10.74
  ---------------------------------------------------------------------------
  data required time                                                     10.74
  data arrival time                                                     -10.76
  ---------------------------------------------------------------------------
  slack (VIOLATED)                                                       -0.02
```

placement 完成后的芯片概貌如下图所示：



5.4. CTS

set_clock_tree_options -target_skew 0.15

set_clock_uncertainty 0.1 [all_clocks]

在 DC 的基础上进一步收束约束，渐进的约束有助于设计收敛。

set_clock_tree_options -max_fanout 32

避免扇出过大导致时钟树违例。

clock_opt -only_cts -no_clock_route -update_clock_latency

查看时序，此时建立时间都收敛而保持时间有违例存在，因此针对保持时间进行优化：

set_fix_hold [all_clocks]

extract_rc

clock_opt -only_psyn -no_clock_route

对时钟进行布线：

route_zrt_group -all_clock_nets

报告所有约束违例，可以发现尽管在建立时间和保持时间上有所违例，但违

例程度并不大，期望可以在布线阶段解决。

报告文件：cts_cons.rpt

max_delay/setup ('clk' group)

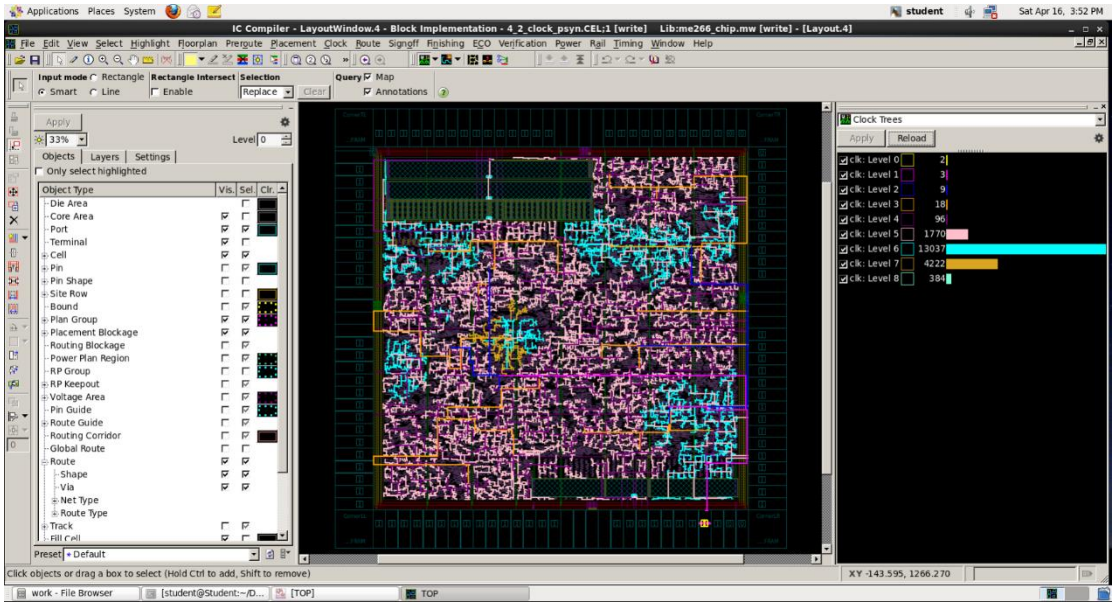
Endpoint	Required Path Delay	Actual Path Delay	Slack	
ME266/CORE/sad_14/sad_data_reg_13_/D	10.22	10.30 r	-0.08	(VIOLATED)

....

min_delay/hold ('clk' group)

Endpoint	Required Path Delay	Actual Path Delay	Slack	
ME266/CORE/sad_1/pe_13/pre_pixel_cal_reg_5_/D	5.51	5.41 f	-0.10	(VIOLATED)

完成后的时钟树如下图所示：



5.5. Routing

route_opt -initial_route_only

`route_opt -incr -effort high`

`route_zrt_eco`

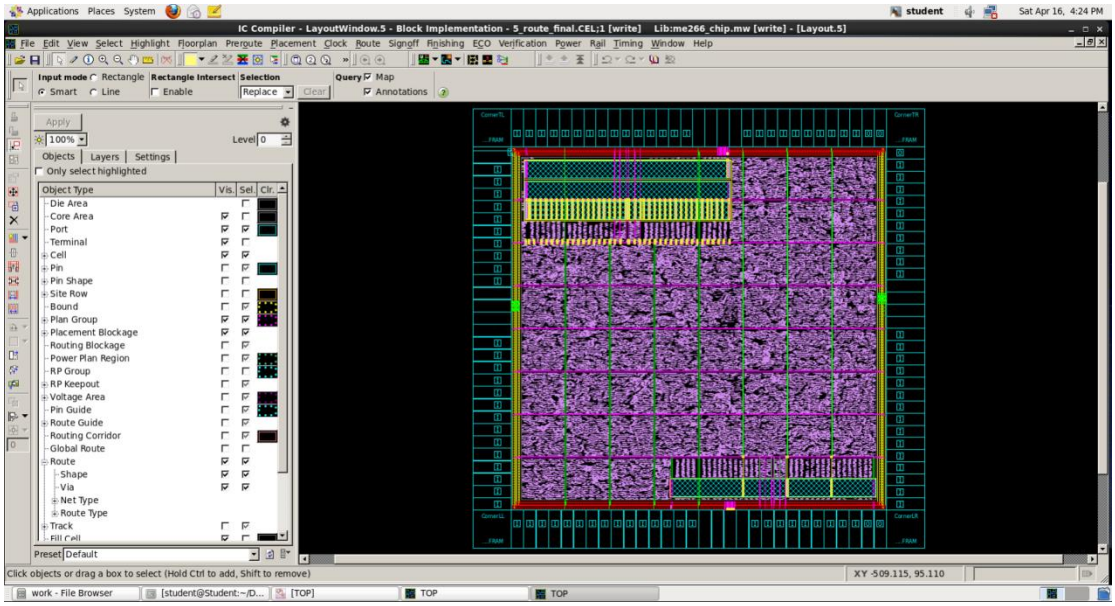
`route_opt -incr -only_hold_time`

`route_opt -incr -effort high`

`route_zrt_eco`

经过测试，采用上述的渐进布线优化指令，通过四次 route_opt，初步布线、强优化布线、优化保持时间、再次强优化，成功使约束收敛到如下的效果：

芯片概貌如下图所示：



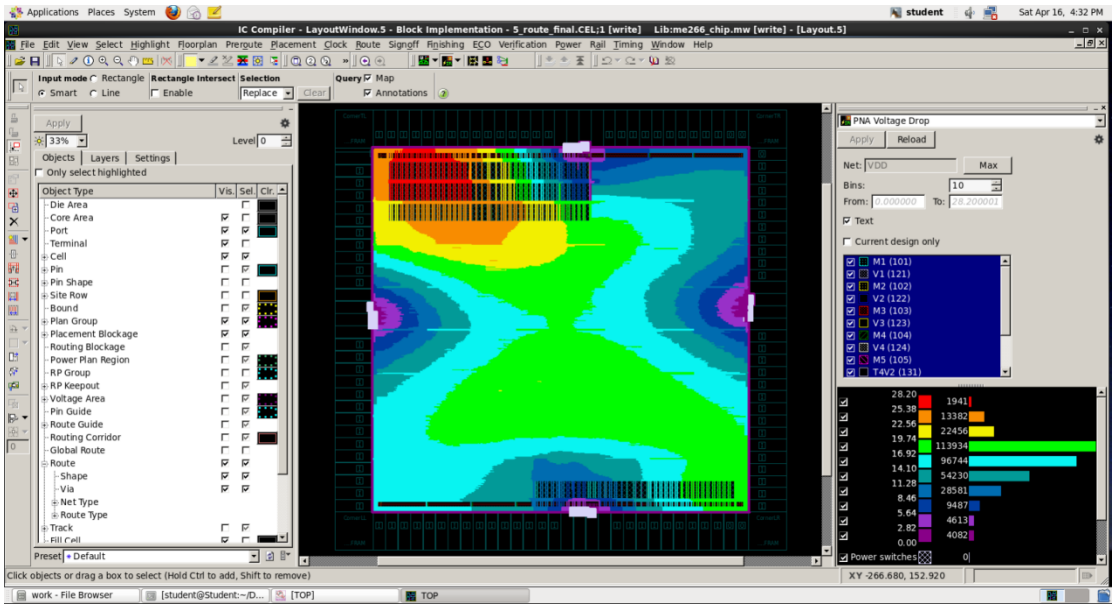
芯片面积报告如下所示：

```
Number of ports:                          104
Number of nets:                           321
Number of cells:                          217
Number of combinational cells:            112
Number of sequential cells:                 0
Number of macros/black boxes:             104
Number of buf/inv:                          2
Number of references:                       5

Combinational area:             247295.153240
Buf/Inv area:                    28292.319203
Noncombinational area:          149198.558986
Macro/Black Box area:           518717.009766
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                915210.721992
Total area:                 undefined
```

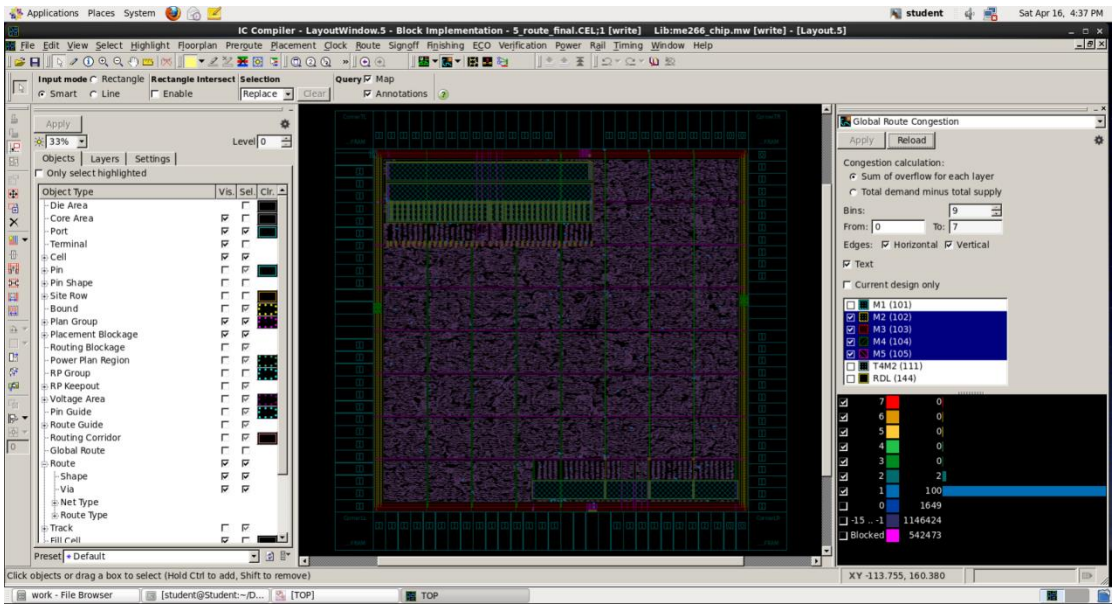
芯片功耗报告如下所示：

```
Global Operating Voltage = 1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  38.1910 mW   (82%)
  Net Switching Power  =   8.4591 mW   (18%)
                         ---------
Total Dynamic Power    =  46.6501 mW  (100%)

Cell Leakage Power     = 120.2197 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.6972           0.9825        1.2076e+03             1.6809  (   3.59%)
memory            18.1475        4.7708e-02        8.6537e+03            18.2038  (  38.92%)
black_box          0.0000           0.0000           0.0000             0.0000  (   0.00%)
clock_network      0.6030           6.9759        1.6840e+03             7.5806  (  16.21%)
register          18.7138        7.9901e-02        2.7600e+04            18.8215  (  40.24%)
sequential         0.0000           0.0000           0.0000             0.0000  (   0.00%)
combinational  2.9308e-02           0.3731        8.1074e+04             0.4835  (   1.03%)
--------------------------------------------------------------------------------------------------
Total             38.1908 mW        8.4591 mW     1.2022e+05 nW         46.7703 mW
```

芯片电压降热力图如下所示，注意到布线完成后电压降甚至从 35.8mV 降低到了 28.2mV：



芯片拥塞情况如下图所示，没有严重拥塞的出现：



可以发现，已经没有了建立时间和保持时间的违例，时序成功收敛。

报告文件：route_final_cons.rpt

```
****************************************
Report : constraint
         -all_violators
Design : me266_chip
Version: O-2018.06-SP1
Date   : Fri Apr  8 22:22:19 2022
****************************************

    Parasitic source       : LPE
    Parasitic mode         : RealRC
    Extraction mode        : MIN_MAX
    Extraction derating : 25/25/25

  max_fanout

                                  Required         Actual
  Net                             Fanout           Fanout           Slack
  -----------------------------------------------------------------
  ME266/net_clk_G2B4I66 (dont_touch)
                                  32.00            38.00            -6.00    (VIOLATED)
  ME266/CUR/net_clk_G2B4I566 (dont_touch)
                                  32.00            33.00            -1.00    (VIOLATED)
```

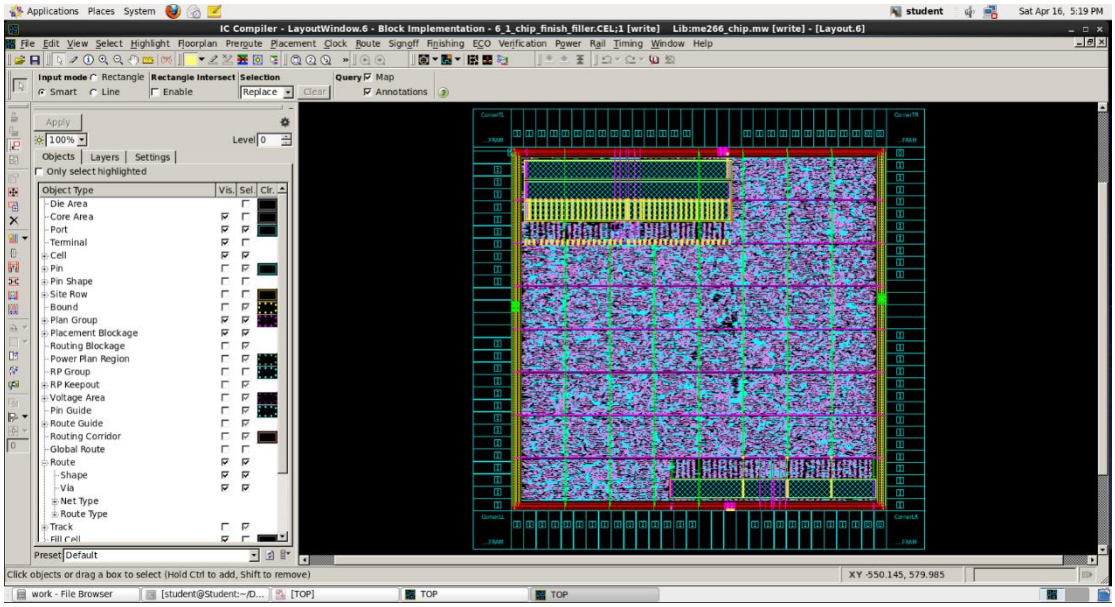
```
---------------------------------------------------------------
Total                          2                      -7.00

min_capacitance

                                Required          Actual
Net                             Capacitance       Capacitance          Slack
---------------------------------------------------------------
clk (dont_touch)                6.43              1.43               -5.00    (VIOLATED)
cur_in[0]                       6.43              1.43               -5.00    (VIOLATED)
…
cur_in[31]                      6.43              1.43               -5.00    (VIOLATED)
ref_in[0]                       6.43              1.43               -5.00    (VIOLATED)
…
ref_in[63]                      6.43              1.43               -5.00    (VIOLATED)
rst                             6.43              1.43               -5.00    (VIOLATED)
cur_read                        6.43              2.85               -3.57    (VIOLATED)
ref_read                        6.43              2.85               -3.57    (VIOLATED)
sad_out                         6.43              2.85               -3.57    (VIOLATED)
sign_sad                        6.43              2.85               -3.57    (VIOLATED)
y_out                           6.43              2.85               -3.57    (VIOLATED)
x_out (dont_touch)              6.43              2.87               -3.56    (VIOLATED)

---------------------------------------------------------------
Total                          104                    -511.42

max_area

                                Required          Actual
Design                          Area              Area                 Slack
---------------------------------------------------------------
me266_chip                      0.00              915210.75          -915210.75

(VIOLATED)
```

上述设计违例分为三个部分，首先是时钟存在两个最大扇出的违例，但违例程度不大，并且不影响到时序的收敛；其次是 IO 口的最小电容，这都是更后端考虑的问题，本课程中不做处理；最后的最大面积违例是因为对于 IO 受限的本

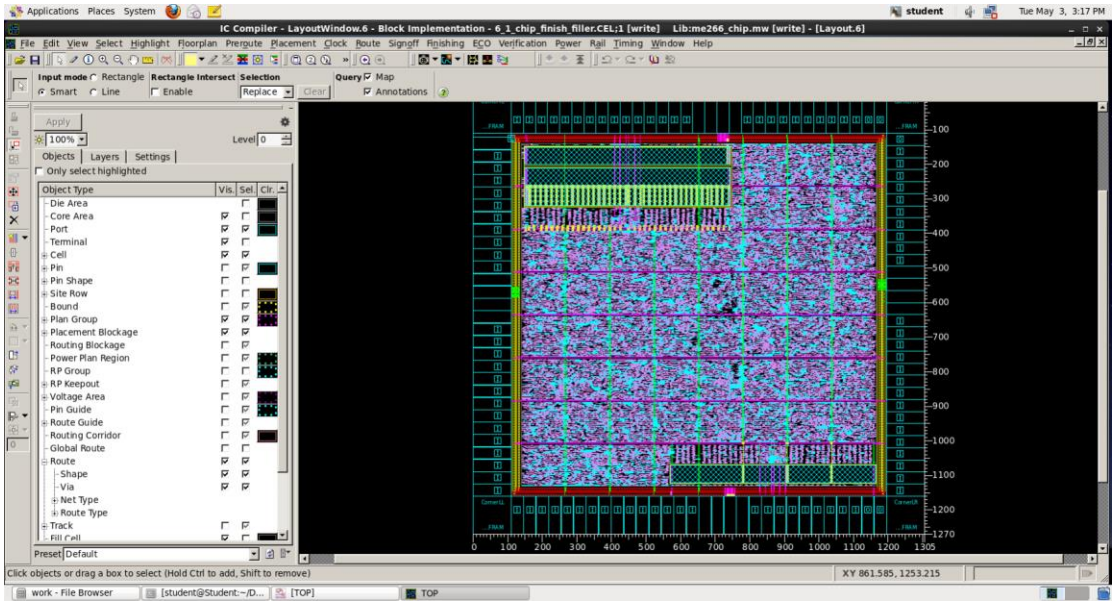
芯片没有对最小面积进行限定，默认是 0，此违例可以忽略。

5.6. Finishing

插入 filler 后，芯片概貌如下所示：



利用 ruler 工具，量得芯片包括 IO pad 的整体面积如下所示，为 1305*1270：



至此，本课程涉及的全部数字芯片电路设计到此完成。

6. 任务分工与设计总结

6.1. 任务分工

1. 朱恒毅（组长）：参与了从前端设计到后端仿真的全部过程，作为组长全程跟踪小组进度；主要负责了 core 的时序确定，负责了 ICC 的全流程。
2. 王清训（副组长）：参与电路性能分析、电路结构设计、电路 RTL 模型和仿真验证；主要负责了 IObuffer 部分的功能设计、当前帧输入、结果输出模块的 RTL 代码以及 RTL 整体的仿真验证。
3. 冯洪（组员）：参与了电路性能分析、电路结构设计、电路 RTL 模型与仿真验证；主要负责了先前帧图像数据传输部分的 RTL 代码设计及其仿真验证。
4. 林羽（组员）：参与了 IOBuffer 的设计、RTL 的设计；主要负责了 RTL 顶层和测试模块的编写与验证仿真，负责了 DC 的全流程。
5. 杨凯杰（组员）：参与了电路结构设计、电路 RTL 模型设计及其仿真验证；主要负责了 core 的 RTL 模型设计、编写及其仿真验证。

6.2. 设计总结

6.2.1. 先前帧复用问题

问题描述：最开始的先前帧数据传输的 RTL 打算以“S”形的顺序进行数据传输，能实现搜索区域行与行之间的数据复用，但仿真验证结果出现问题。发现此模型的 RTL 代码逻辑较为复杂，修改多次代码仍未解决。

解决方案：将数据传输顺序按一行一行进行传输，简化传输逻辑，同时也能减小 SRAM 的面积开销。

6.2.2. Core 数据复用问题

问题描述：为减小芯片面积，需要降低芯片内部传输带宽，具体到计算核心部分的设计中即如何降低输入输出缓存（IO Buffer）与计算核心之间的传输带宽，或者说如何实现计算核心内部的数据复用。

解决方案：设计处理单元时，在其内部设置两个寄存器分别存储用于计算的当前帧和先前帧像素数据，并为当前帧寄存器设置锁存机制。在设计 SAD 计算单元时，将 64 个处理单元每 8 个上下连接形成形式上的 8 列，使得像素数据可以在处理单元之间传递，在一个处理周期的前 8 个周期结束后，所有处理单元中都存在相应的当前帧像素数据并进行锁存。由于先前帧像素信号持续输入与向下传

递，因此可以认为 SAD 计算单元在形式上进行了下移，从而实现了像素数据的复用。相比于为每个处理单元都输入像素数据，该方案的传输带宽明显降低，仅为前者的1/8。

6.2.3. Sram 带宽匹配问题

问题描述：在当前帧输入模块的 RTL 代码编写过程中，遇到的主要问题是由于带宽是 4bit 因此在每接收两次数据才能往 SRAM 中写一次。

解决方案：通过增加了一个 8bit 的中间缓存接收数据并每两个周期传给 SRAM 一次数据来解决了这个问题。

6.2.4. 测试数据输入错位问题

问题描述: 在 RTL 模型设计初期，由于先前帧与当前帧带宽不同，故在一个周期内 tb 需要读取不同行数的输入数据。此时由于两个模块的读数使能方式不统一、控制传递有一个周期的延时且$fscanf 函数在初次启动时不能读取数据等问题，使得先前帧和当前帧在数据输入时存在复杂的错位。

解决方案：确定 buffer 模块对 tb 使能的方式，增加两个 I/O 口, 使能信号为 1 时读取数据并提前拉高；同时优化脚本，补全位差，将先前帧和当前帧输入数据文档调整为 16bit 或 64bit 一行且符合读数顺序。使得先前帧和当前帧每次使能时只需读取一行数据，简化读数方式，对齐时序。

6.2.5. ICC marco cell 排布问题

问题描述：在开始的时候仅仅将四个 sram 堆放在一起，导致了 sram 之间、sram 与边界之间出现了大量的缝隙，排布进缝隙的 standard cell 引出的长导线带来巨大的延迟难以时序收敛。

解决方案：通过咨询助教，了解到 sram 的排布一方面需要贴近相应的 IO，另一方面需要避免缝隙，最后通过查询版图和 sram 的尺寸，敲定了合理的排布范围，并划定 block 区域，最终实现了合理排布。