芯设实验报告

4*4脉动阵列设计

PB20000328 杨博涵

设计背景

介绍

脉动阵列（systolic array）是一种由计算单元（PE）构成的阵列结构，其工作方式和过程犹如人体血液循环系统的工作方式和过程。在这种阵列结构中，数据按预先确定的“流水”方式在阵列的处理单元间有节奏地“流动”。在数据流动的过程中，所有的处理单元**同时并行**地对流经它的数据进行处理，因而它可以达到很高的并行处理速度。同时，**预先确定的数据流动模式**使数据从流进处理单元阵列到流出处理单元阵列的过程中完成所有对它应该做的处理，无需再重新输入这些数据，且只有阵列的“边界”处理单元与外界进行通信，由此实现在不增加阵列输入、输出带宽的条件下，提高阵列整体的处理速度。由于阵列和处理单元的结构简单、规则一致，可达到很高的模块化程度，非常适合超大规模集成电路的设计和制造。

由此可见，脉动阵列极其适合于具有固定计算模式、单操作简单重复的任务。本次实验便是利用脉动阵列加速矩阵乘法。

脉动阵列的优势

脉动阵列发明于1982年H. T. Kung的论文"Why systolic architectures?"，其中作者给出了三个理由：

- **Simple and regular design**

脉动阵列简单而规整的布局极大的降低了设计成本，Google公司正是利用这个特点在数月内就基于脉动阵列完成了TPU的设计。

- **Concurrency and communication**

所有PE同时运行的特点提高了吞吐率，并且数据在其中流动完成一个个操作的过程无需从DRAM中额外取数据，极大利用了数据重用性，减少了带宽需求。一边流入数据，成品就从另一边流出来，无需额外缓存。

- **Balancing computation with I/O**

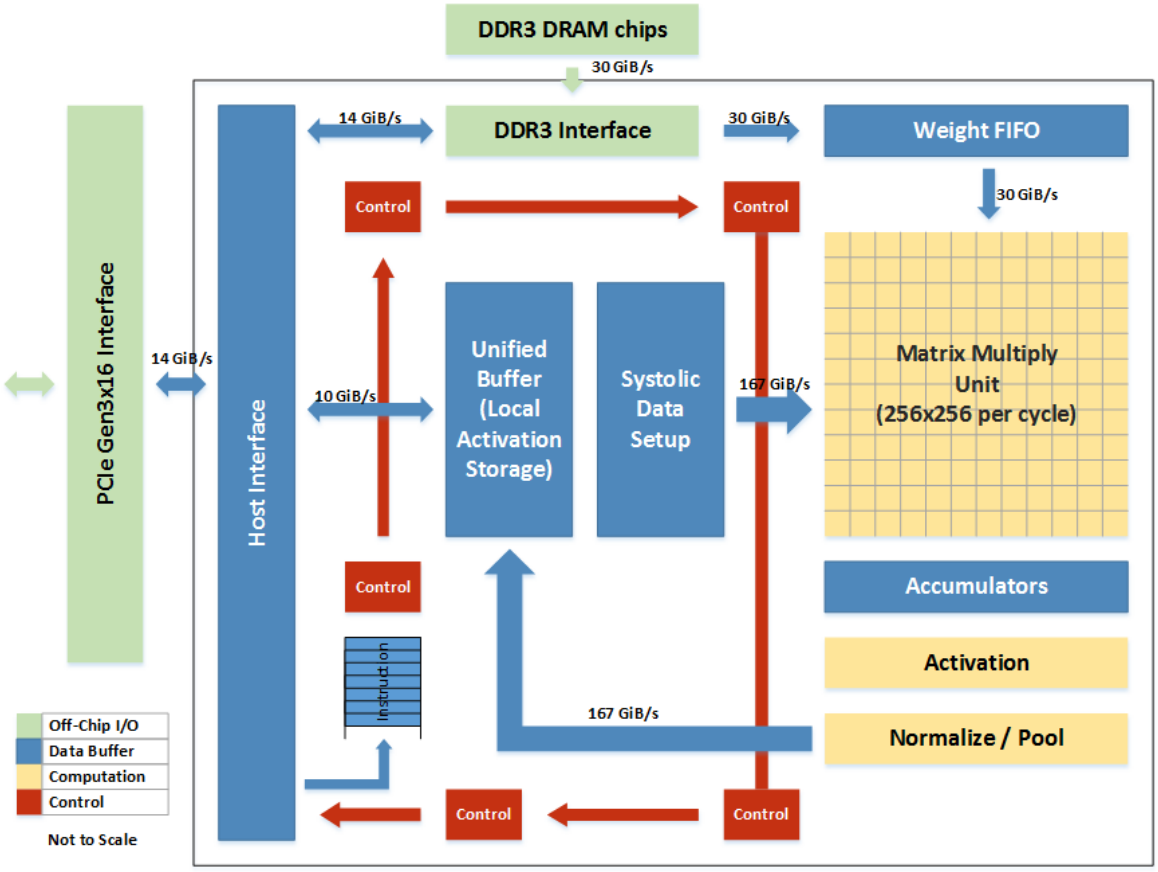
传统的计算系统的模型中，数据存取的带宽往往大大低于数据处理的速度。因此，整个系统的处理能力很大程度受限于访存的能力，这个问题也是多年来计算机体系结构研究的重要课题之一，可以说是推动处理器和存储器设计的一大动力。而脉动架构用了一个很简单的方法：**让数据尽量在处理单元中多流动一会儿**，巧妙的平衡了这个问题。

- **电路设计上的优势**

PE只能向相邻的PE发送数据，避免了全局广播和扇入（fan-in），减少了单管上的负载，获得快速的响应时间。

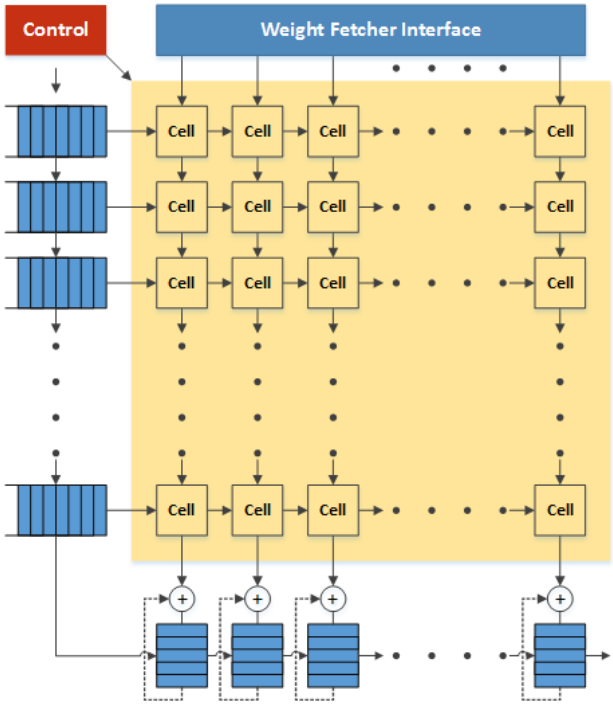
脉动阵列的工业应用

第44届ISCA会议上，Google提出了用于数据中心服务器端进行神经网络推理加速的张量处理器TPU，其相比于服务器端的CPU与GPU速度快了近15-30倍。其中的MMU作为运算核心则是基于脉动阵列打造的。



设计方案

本次实验，我根据脉动阵列的原理，设计了一个4*4的矩阵乘法加速阵列，大概的架构图如下所示。

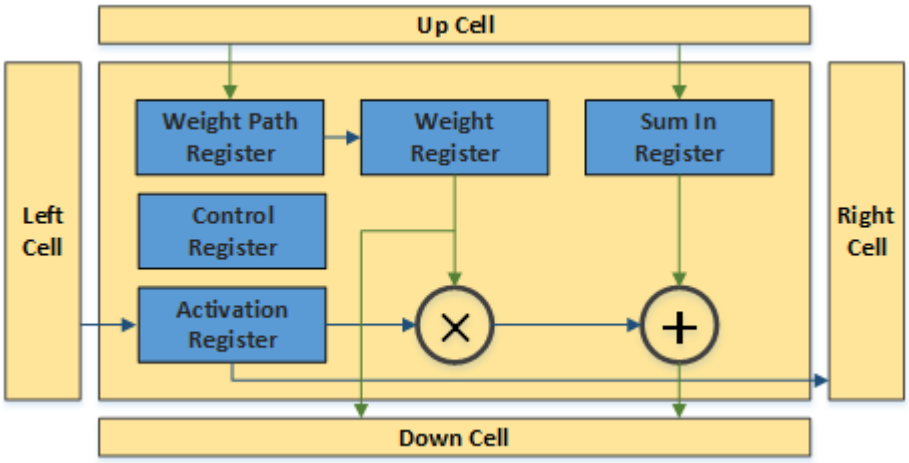


输入的两个矩阵分别为A与B，A从左向右流动，B从上往下流动，乘积保留在cell当中，以供下个继续作为操作数。等到所有的数据全部计算完毕后，再将数据打拍取出。

同时，要实现正确的矩阵运算，数据进入脉动阵列需要调整好形式，并且按照一定的顺序。这就**需要对原始的矩阵进行一些reformat**，我增加了数据缓存区，用于规范输入数据的形状。

每一个cell由MAC和寄存器构成，寄存器包括A寄存器、B寄存器、输出寄存器，每个时钟计算output=A*B+output。

下图是一个示意图，但是采用的方案不是我用的Output Stationary方法，所以仅供参考。



参数文档

DMA+准备区规格：

1. 准备区有AB两个，大小为7*4的移位寄存器阵列
2. DMA一次顺序访存一行4*2个数，然后交由准备区后进行下一次访存，一切由控制器控制
3. 要求一次对每个矩阵的每一单元同时load一个元素，则带宽是 8unit/clk
4. A矩阵的物理存储排布是关于竖直轴对称，B矩阵的物理存储排布是关于水平轴对称
5. 在ram里最好是A列优先，B行优先，这样是连续取址的

PE规格：

1. 采用A、B矩阵流入，输出固定的方法，在最后提取输出到缓存区
2. 8bitMAC
3. 全部采用流水化，短连线的方式

实现方法

verilog源代码由array.v、controller.v、pe.v、shift_buffer.v、systolic_array_wrapper.v组成

array.v

实现二维阵列的拓扑结构，连接各个例化的PE

controller.v

控制器代码为标准三段式FSM，由四个状态组成：

- **STATE_IDLE**

脉动阵列闲置状态

- **STATE_LOAD**

数据装载状态，由en信号从IDLE态转换过来

- **STATE_COMPUTE_PUMP**

数据注入阵列状态，等装载完毕后自动开始

- **STATE_COMPUTE_OUT**

结果输出状态，节拍输出最终结果。

pe.v

乘加器MAC的实现

shift_buffer.v

移位寄存器的实现，包括数据排序控制器

systolic_array_wrapper.v

top文件

约束参数

时序约束

Clk_period = 40ns

在这个周期下，芯片满足setup约束。同时，Hold约束也通过添加buffer成功满足，没有时序违例，全部met。

```
set TOP_MODULE              systolic_array

set unit                    1

set jitter                  [expr 0.10*$unit]
set skew                    [expr 0.05*$unit]
set trans                   [expr 0.50*$unit]

set input_delay_percent     0.3
set output_delay_percent    0.3
set operating_condition     WCCOM

set DRIVE_CELL              "BUFFD1BWP7T"
set DRIVE_PIN               "Z"
set PIN_LOAD                [expr 32*[load_of tcb018gbwp7twc_ccs/BUFFD1BWP7T/I]]

set PAD_TRANS               0.5
set PAD_LOAD                50

set INPUT_CLK_LIST          [get_ports clk_p]
set INPUT_RST_LIST          [get_ports rstn_p]

set CLK_PERIOD              [expr 40*$unit]
```

floorplan && pns

采用固定宽高比的方式完成布局规划

```
create_floorplan \
        -control_type       aspect_ratio \
        -core_aspect_ratio  1.0 \
        -core_utilization   0.4 \
        -left_io2core       25 \
        -bottom_io2core     25 \
        -right_io2core      25 \
        -top_io2core        25 \
        -start_first_row
}
```

同时，调整offset直至电源网不被vdd/vss pad阻挡

```
set_power_ring_strategy \
    ring1 \
    -core \
    -nets       "${MW_POWER_NET} ${MW_GROUND_NET}" \
    -template   ./script/tpns_ring.tpl:ring(11,11)

compile_power_plan -ring


# Specify strategy for top mesh.
# template : mesh(wh, wv, oh, ov, ph, pv, nh, nv)
set_power_plan_strategy \
    mesh1 \
    -core \
    -nets       "${MW_POWER_NET} ${MW_GROUND_NET}" \
    -template   ./script/tpns_mesh.tpl:mesh(2,2,30,38,30,42,40,40) \
    -extension  {{stop: outermost_ring}}
```

芯片接口

芯片共有20个pin，32个pad

- Clk_p, rstn_p : 常规pin
- En_p : 使能信号
- Ack_p : 工作完成信号
- P_shift_in : 输入数据（8bit）
- P_shift_out : 输出数据（8bit）

```
module systolic_array //top模块
(
    input clk_p,
    input rstn_p,
    input en_p, //协处理器启动信号

    input [7:0] p_shift_in,

    output [7:0] p_shift_out,
    output ack_p //示意结束

);
```

仿真结果

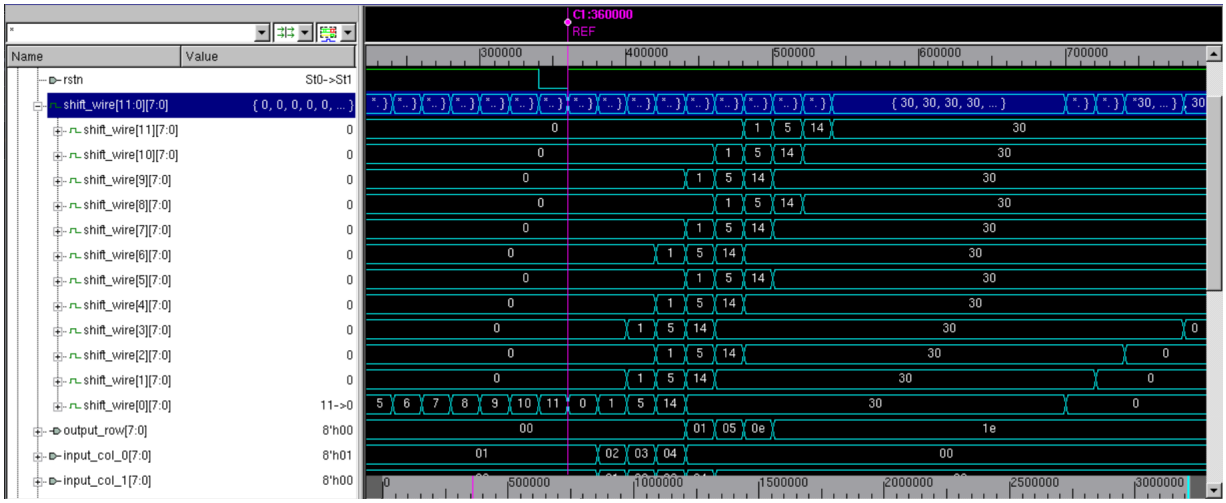
仿真了若干输入结果，均为正确，下面举例三种

- A = 1 2 3 4
  1 2 3 4
  1 2 3 4
  1 2 3 4
  B = 1 1 1 1
  2 2 2 2
  3 3 3 3
  4 4 4 4
  C = A*B =

  30 30 30 30
  30 30 30 30
  30 30 30 30
  30 30 30 30

下图为仿真结果。T=40~120为load阶段，数据在装载进移位寄存器。T=120~440为注入阶段，共注入了(440-120)/20 = 16个时钟周期，从阵列左上角进入直到最后一个数据离开右下角。T=440~500为输出阶段，将4*4的矩阵C输出，结果正确。

```
T=    0, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=   40, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=   60, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=1, state_compute_pump= 0, state_compute_out=0
T=   80, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=2, state_compute_pump= 0, state_compute_out=0
T=  100, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=3, state_compute_pump= 0, state_compute_out=0
T=  120, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  140, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 1, state_compute_out=0
T=  160, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 2, state_compute_out=0
T=  180, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 3, state_compute_out=0
T=  200, out0=  1, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 4, state_compute_out=0
T=  220, out0=  5, out1=   1, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 5, state_compute_out=0
T=  240, out0= 14, out1=   5, out2=  1, out3=  0, state=2, state_shift=0, state_compute_pump= 6, state_compute_out=0
T=  260, out0= 30, out1=  14, out2=  5, out3=  1, state=2, state_shift=0, state_compute_pump= 7, state_compute_out=0
T=  280, out0= 30, out1=  30, out2= 14, out3=  5, state=2, state_shift=0, state_compute_pump= 8, state_compute_out=0
T=  300, out0= 30, out1=  30, out2= 30, out3= 14, state=2, state_shift=0, state_compute_pump= 9, state_compute_out=0
T=  320, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=10, state_compute_out=0
T=  340, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=11, state_compute_out=0
T=  360, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=12, state_compute_out=0
T=  380, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=13, state_compute_out=0
T=  400, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=14, state_compute_out=0
T=  420, out0= 30, out1=  30, out2= 30, out3= 30, state=2, state_shift=0, state_compute_pump=15, state_compute_out=0
T=  440, out0= 30, out1=  30, out2= 30, out3= 30, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  460, out0= 30, out1=  30, out2= 30, out3= 30, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=1
T=  480, out0= 30, out1=  30, out2= 30, out3= 30, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=2
T=  500, out0= 30, out1=  30, out2= 30, out3= 30, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=3
T=  520, out0= 30, out1=  30, out2= 30, out3= 30, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  540, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  640, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=0, state_compute_pump= 0, state_compute_out=0
```



- D = 1 1 1 1
  1 0 0 1
  1 0 0 1

1 1 1 1

E = 12 14 16 18

20 22 24 26

28 30 32 34

36 38 40 42

F = 96 104 112 120

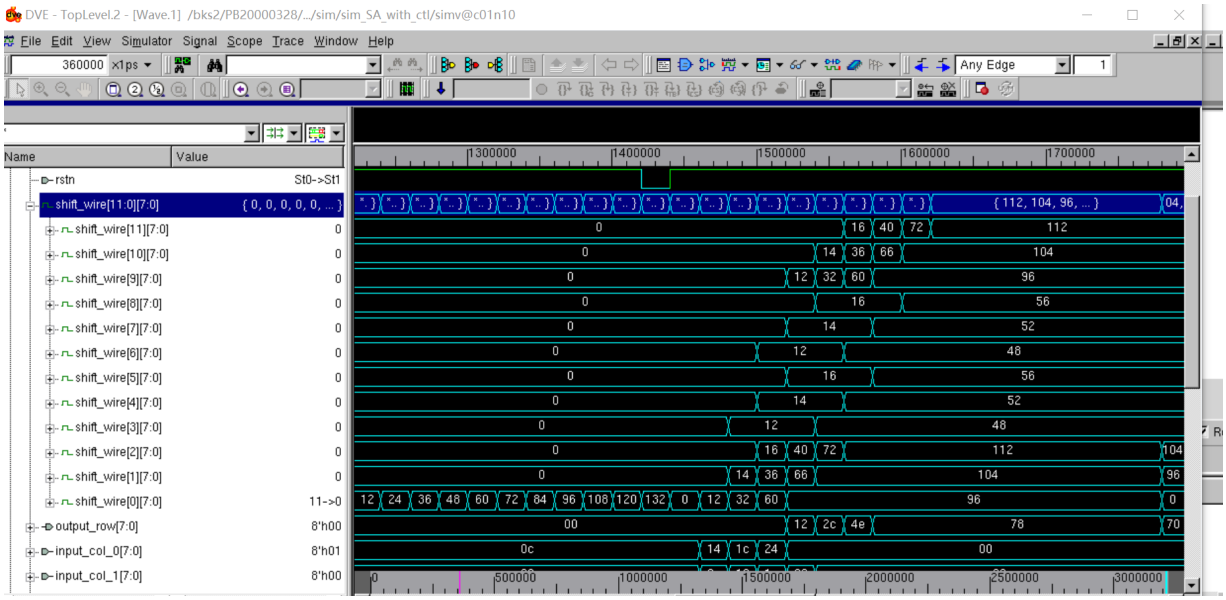
48 52 56 60

48 52 56 60

96 104 112 120

T=1040~1120为输出阶段，将4*4的矩阵F输出，结果正确。

```
T=  540, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  640, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  660, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=1, state_compute_pump= 0, state_compute_out=0
T=  680, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=2, state_compute_pump= 0, state_compute_out=0
T=  700, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=3, state_compute_pump= 0, state_compute_out=0
T=  720, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 0, state_compute_out=0
T=  740, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 1, state_compute_out=0
T=  760, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 2, state_compute_out=0
T=  780, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 3, state_compute_out=0
T=  800, out0= 18, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 4, state_compute_out=0
T=  820, out0= 44, out1=  18, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 5, state_compute_out=0
T=  840, out0= 78, out1=  18, out2= 18, out3=  0, state=2, state_shift=0, state_compute_pump= 6, state_compute_out=0
T=  860, out0=120, out1=  18, out2= 18, out3= 18, state=2, state_shift=0, state_compute_pump= 7, state_compute_out=0
T=  880, out0=120, out1=  60, out2= 18, out3= 44, state=2, state_shift=0, state_compute_pump= 8, state_compute_out=0
T=  900, out0=120, out1=  60, out2= 60, out3= 78, state=2, state_shift=0, state_compute_pump= 9, state_compute_out=0
T=  920, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=10, state_compute_out=0
T=  940, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=11, state_compute_out=0
T=  960, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=12, state_compute_out=0
T=  980, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=13, state_compute_out=0
T= 1000, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=14, state_compute_out=0
T= 1020, out0=120, out1=  60, out2= 60, out3=120, state=2, state_shift=0, state_compute_pump=15, state_compute_out=0
T= 1040, out0=120, out1=  60, out2= 60, out3=120, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1060, out0=120, out1=  60, out2= 60, out3=120, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=1
T= 1080, out0=112, out1=  56, out2= 56, out3=112, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=2
T= 1100, out0=104, out1=  52, out2= 52, out3=104, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=3
T= 1120, out0= 96, out1=  48, out2= 48, out3= 96, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1140, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1240, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=0, state_compute_pump= 0, state_compute_out=0
```



- G = 0 3 6 9

  12 15 18 21

  24 27 30 33

  36 39 42 45

  H = 2 0 0 1

  0 2 1 0

  0 1 2 0

  1 0 0 2

I = 9 12 15 18
45 48 51 54
81 84 87 90
117 120 123 126

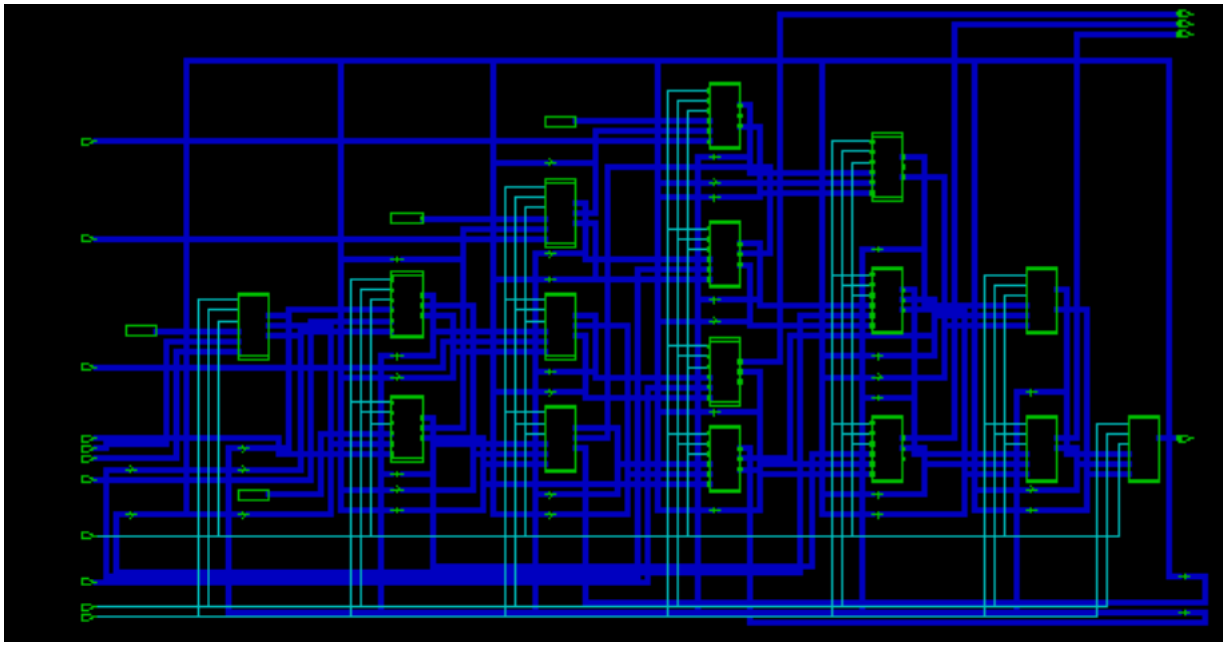
T=1660~1720为输出阶段，将4*4的矩阵I输出，结果正确。

```
T= 1140, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1240, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1260, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=1, state_compute_pump= 0, state_compute_out=0
T= 1280, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=2, state_compute_pump= 0, state_compute_out=0
T= 1300, out0=  0, out1=   0, out2=  0, out3=  0, state=1, state_shift=3, state_compute_pump= 0, state_compute_out=0
T= 1320, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1340, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 1, state_compute_out=0
T= 1360, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 2, state_compute_out=0
T= 1380, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 3, state_compute_out=0
T= 1400, out0=  0, out1=   0, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 4, state_compute_out=0
T= 1420, out0=  0, out1=  12, out2=  0, out3=  0, state=2, state_shift=0, state_compute_pump= 5, state_compute_out=0
T= 1440, out0=  0, out1=  12, out2= 24, out3=  0, state=2, state_shift=0, state_compute_pump= 6, state_compute_out=0
T= 1460, out0= 18, out1=  12, out2= 24, out3= 36, state=2, state_shift=0, state_compute_pump= 7, state_compute_out=0
T= 1480, out0= 18, out1=  54, out2= 24, out3= 36, state=2, state_shift=0, state_compute_pump= 8, state_compute_out=0
T= 1500, out0= 18, out1=  54, out2= 90, out3= 36, state=2, state_shift=0, state_compute_pump= 9, state_compute_out=0
T= 1520, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=10, state_compute_out=0
T= 1540, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=11, state_compute_out=0
T= 1560, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=12, state_compute_out=0
T= 1580, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=13, state_compute_out=0
T= 1600, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=14, state_compute_out=0
T= 1620, out0= 18, out1=  54, out2= 90, out3=126, state=2, state_shift=0, state_compute_pump=15, state_compute_out=0
T= 1640, out0= 18, out1=  54, out2= 90, out3=126, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1660, out0= 18, out1=  54, out2= 90, out3=126, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=1
T= 1680, out0= 15, out1=  51, out2= 87, out3=123, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=2
T= 1700, out0= 12, out1=  48, out2= 84, out3=120, state=3, state_shift=0, state_compute_pump= 0, state_compute_out=3
T= 1720, out0=  9, out1=  45, out2= 81, out3=117, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
T= 1740, out0=  0, out1=   0, out2=  0, out3=  0, state=0, state_shift=0, state_compute_pump= 0, state_compute_out=0
$finish called from file "systolic_array_sim_with_control.v", line 330.
```

后续步骤

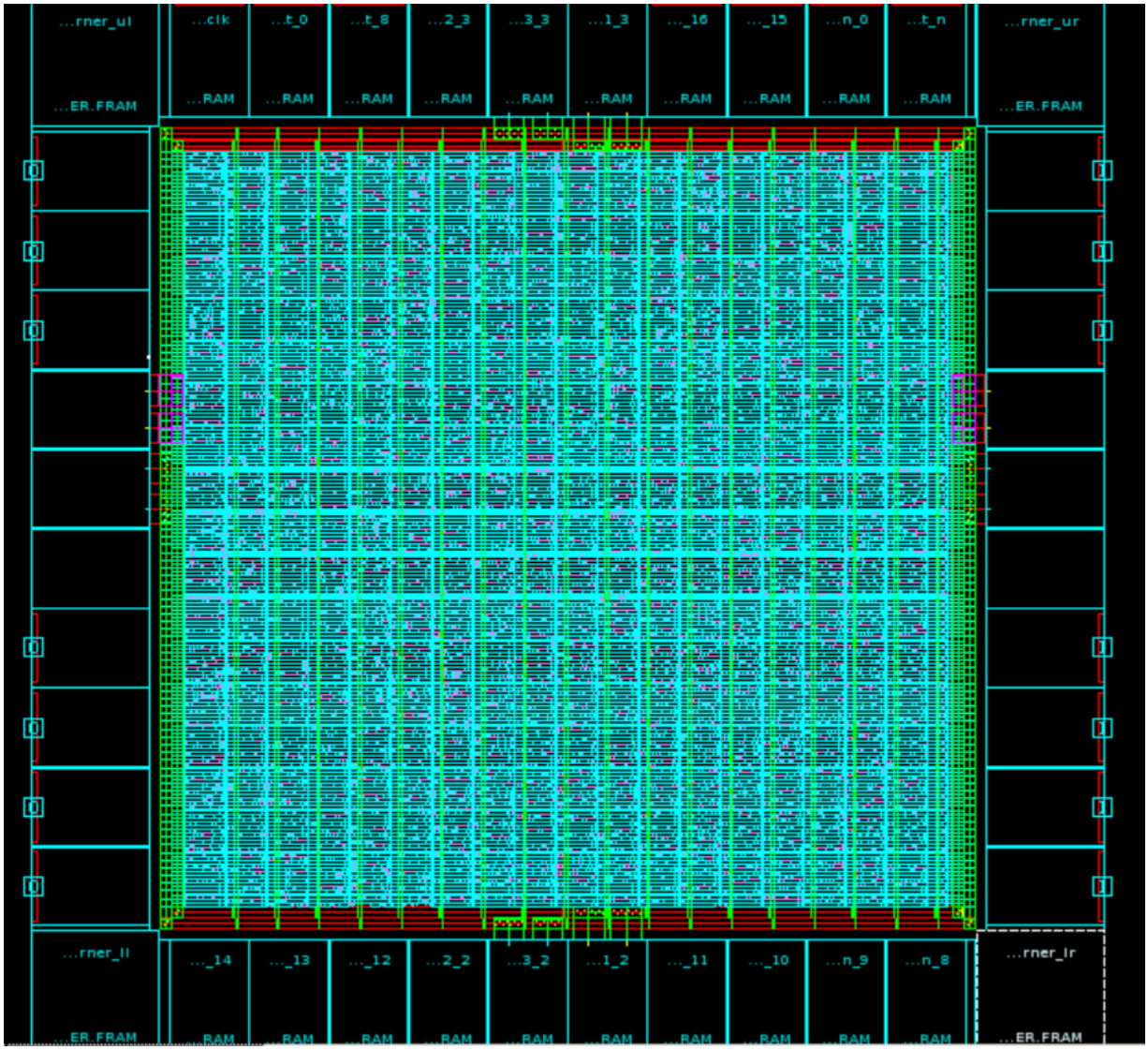
综合

systolic array (4x4) schematic：



布局布线

目前经过各项脚本参数（IOpad布局，电源网参数）的设置，最新版图已通过时序、drc和lvs检验。



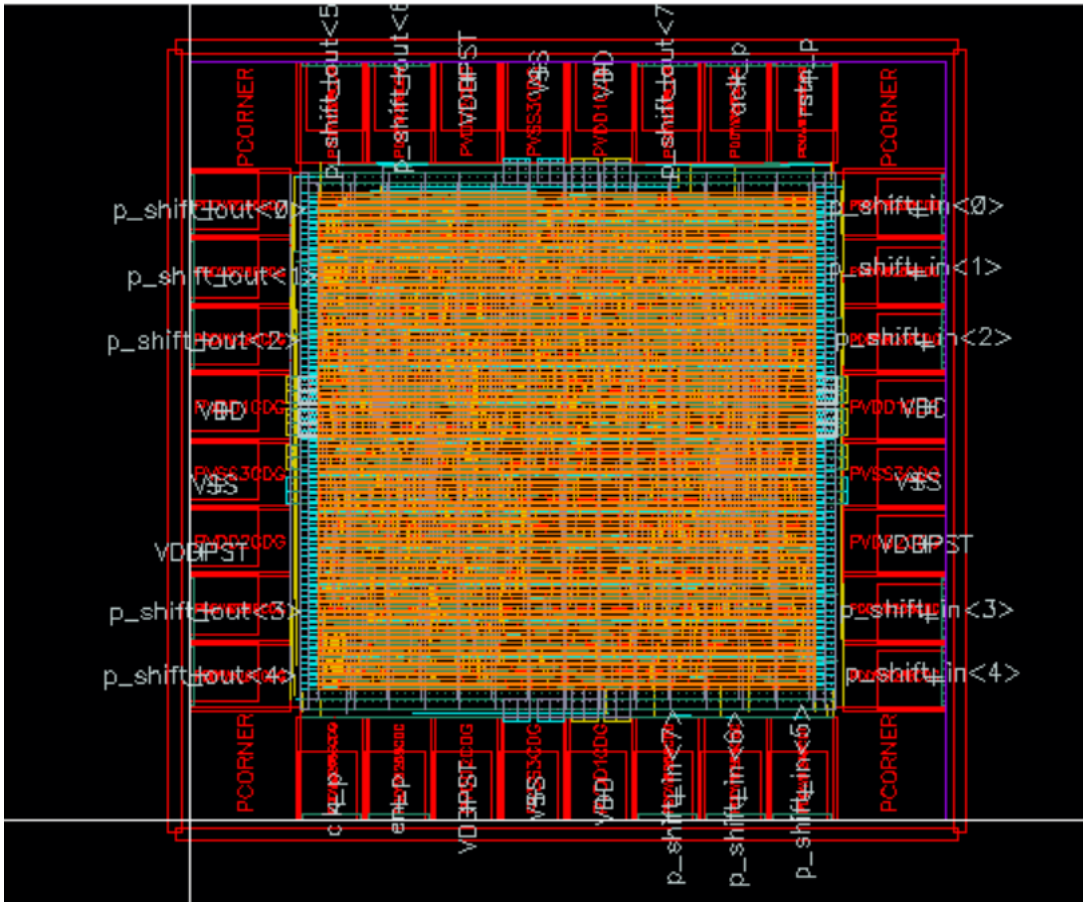
生成了相应的面积报告如下

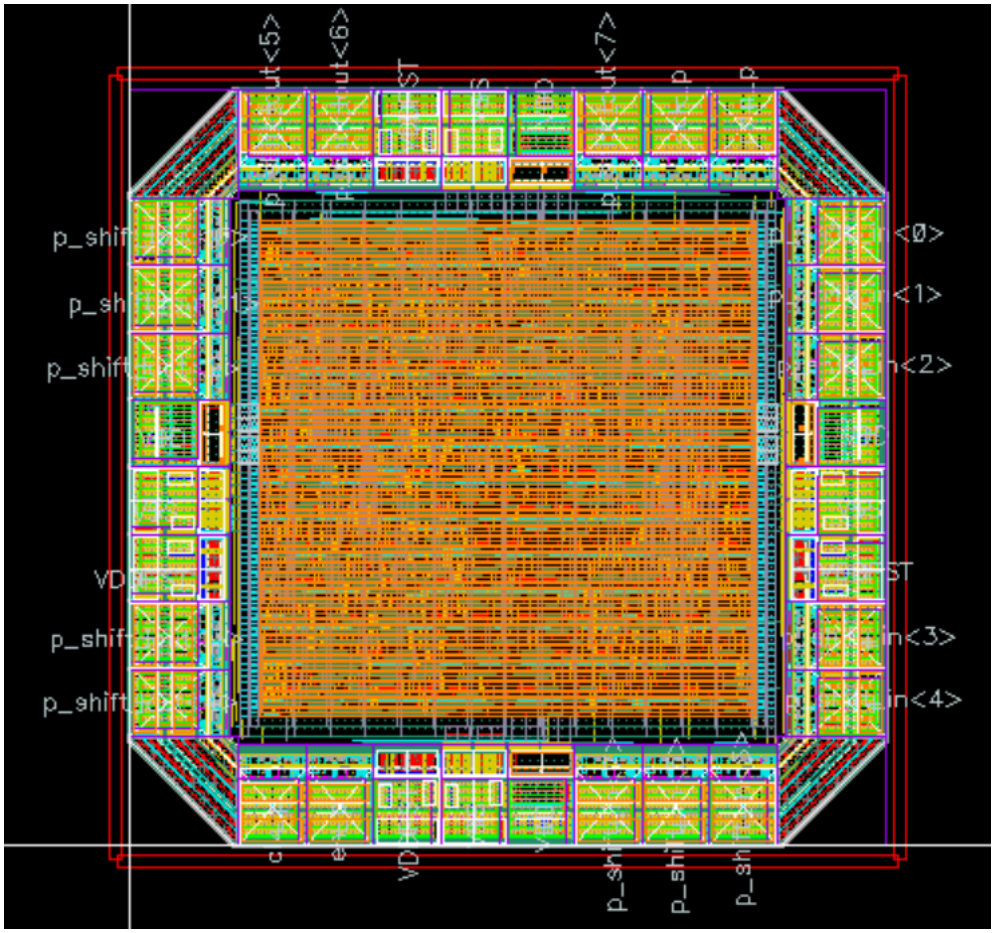
```
****************************** P&R Summary ********************************
Date : Wed Feb  8 16:23:32 2023
Machine Host Name: c01n10
Working Directory: /bks2/PB20000328/ic_design/apr
Library Name:      systolic_array_APR
Cell Name:         outputs_icc.CEL;1
Design Statistics:
    Number of Module Cells:        13209
    Number of Pins:                51924
    Number of IO Pad Cells:        32
    Number of IO Pins:             20
    Number of Nets:                4394
    Average Pins Per Net (Signal): 3.14834

Chip Utilization:
    Total Std Cell Area:           86220.87
    Total Blockage Area:           14995.41
    Total Pad Cell Area:           374800.00
    Core Size:     width 607.04, height 599.76; area 364078.31
    Pad Core Size: width 677.04, height 671.00; area 454293.84
    Chip Size:     width 917.04, height 911.00; area 835423.44
    Std cells utilization:         24.70%
    Cell/Core Ratio:               23.68%
    Cell/Chip Ratio:               55.18%
    Number of Cell Rows:            153
```

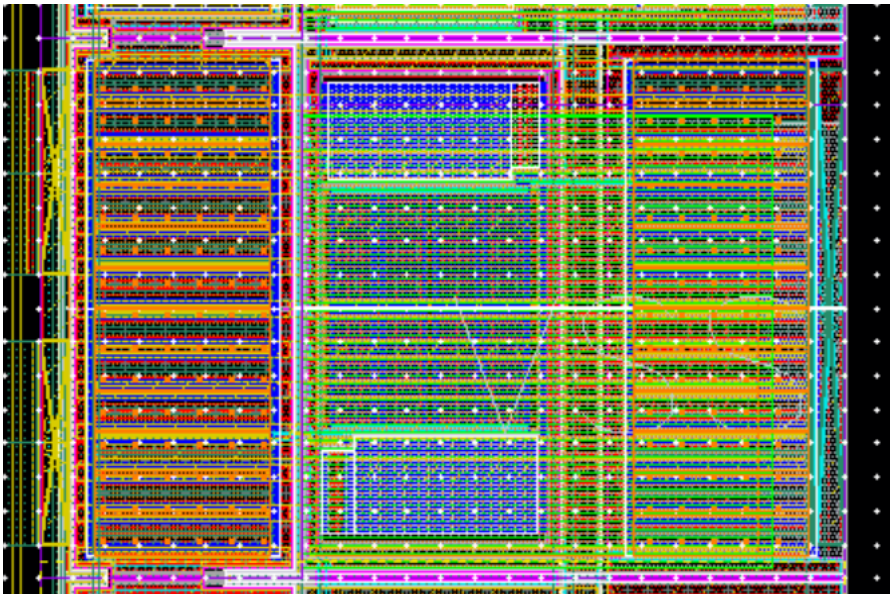
release阶段

icc gds生成后，需要在cadence virtuoso上完成bonding pad和guard ring的手工版图添加，并在mentor calibre上进行DRC, ant, LVS验证，已全部通过

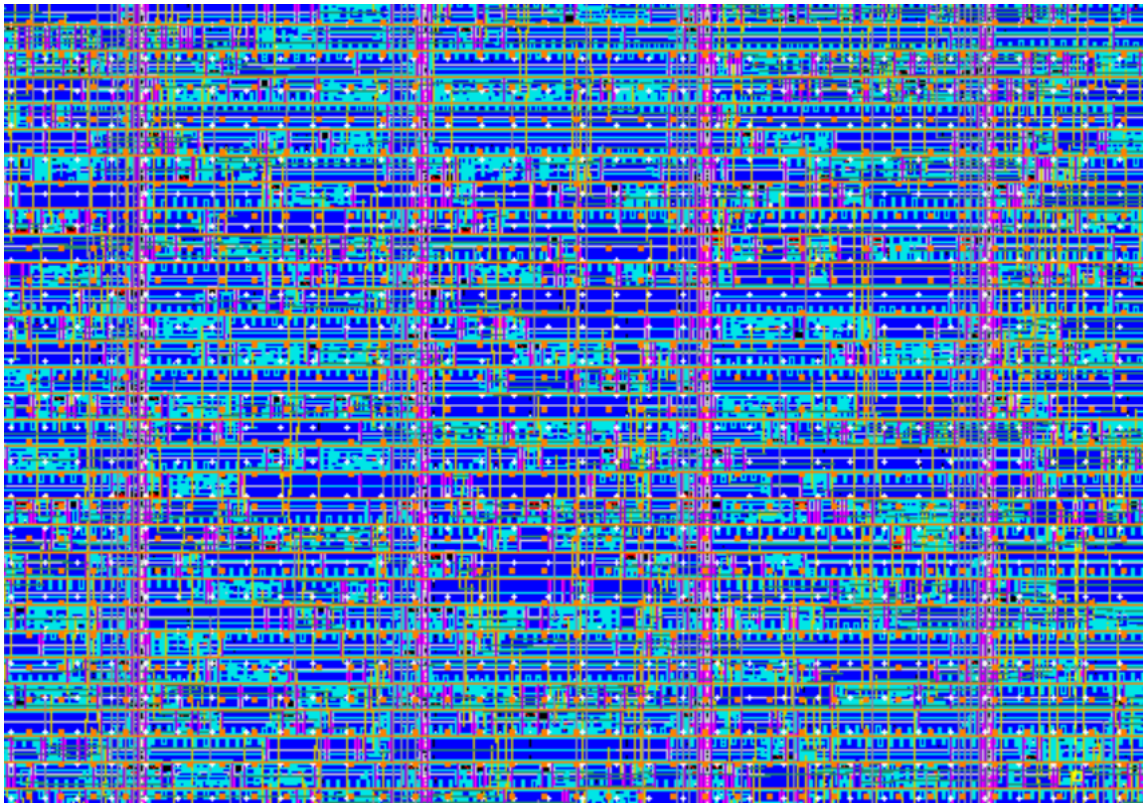



out<5>
out<6>
out<7>
p_shif
<0>
p_sh
<1>
p_shif
<2>
VD
ST
p_shif
in<3>
p_shif
in<4>
VD
p_sh
p_sh
p_sh

pad:



inside:



LVS：

实验已完成，layout已提交。