

数字集成电路课程设计报告



上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

课程设计名称:

姓 名: 贾鑫鹏

学 号: 519021911317

小 组 成 员: 贾鑫鹏 吴非 汪嘉杭 王玉麟

提 交 日 期:

目录

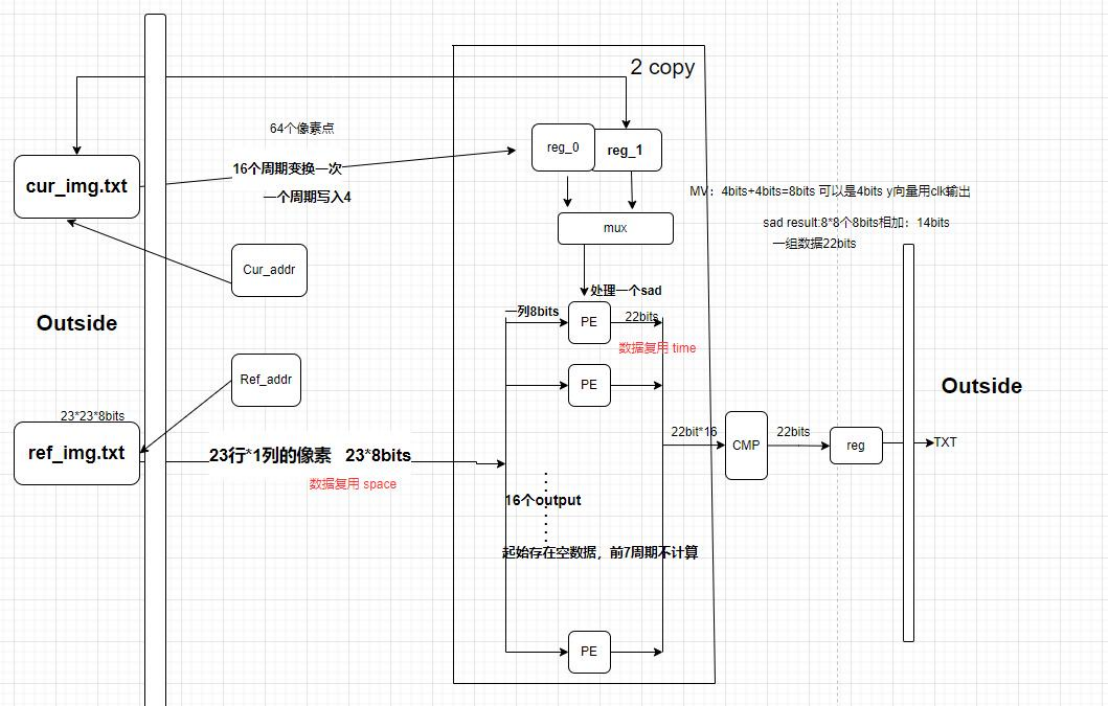
1. 设计规范简介
2. 电路性能分析与电路结构设计
3. 电路 RTL 模型与仿真验证
4. 电路逻辑综合策略与综合结果
5. 电路物理实现与结果分析
6. 任务分工与设计总结

1.课程设计规范简介

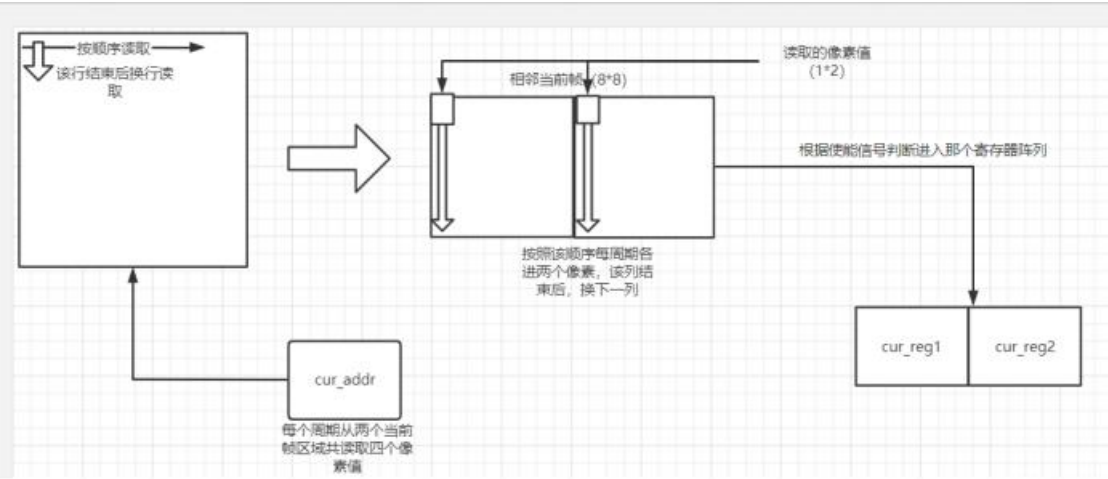
- 1) 每秒 60 帧 4K 视频 ($3840 \times 2160 @ 60\text{fps}$) 的实时处理能力
- 2) 采用全搜索 ME 算法、支持 8×8 块大小的 SAD 计算、搜索区间为 $[-7,8]$
- 3) 芯片设计工艺：华力 55nm 工艺
- 4) 评价指标：电路的实时处理能力、芯片的 PPA (Performance or frequency, Power, Area)、输入/输出数据的带宽及其利用效率

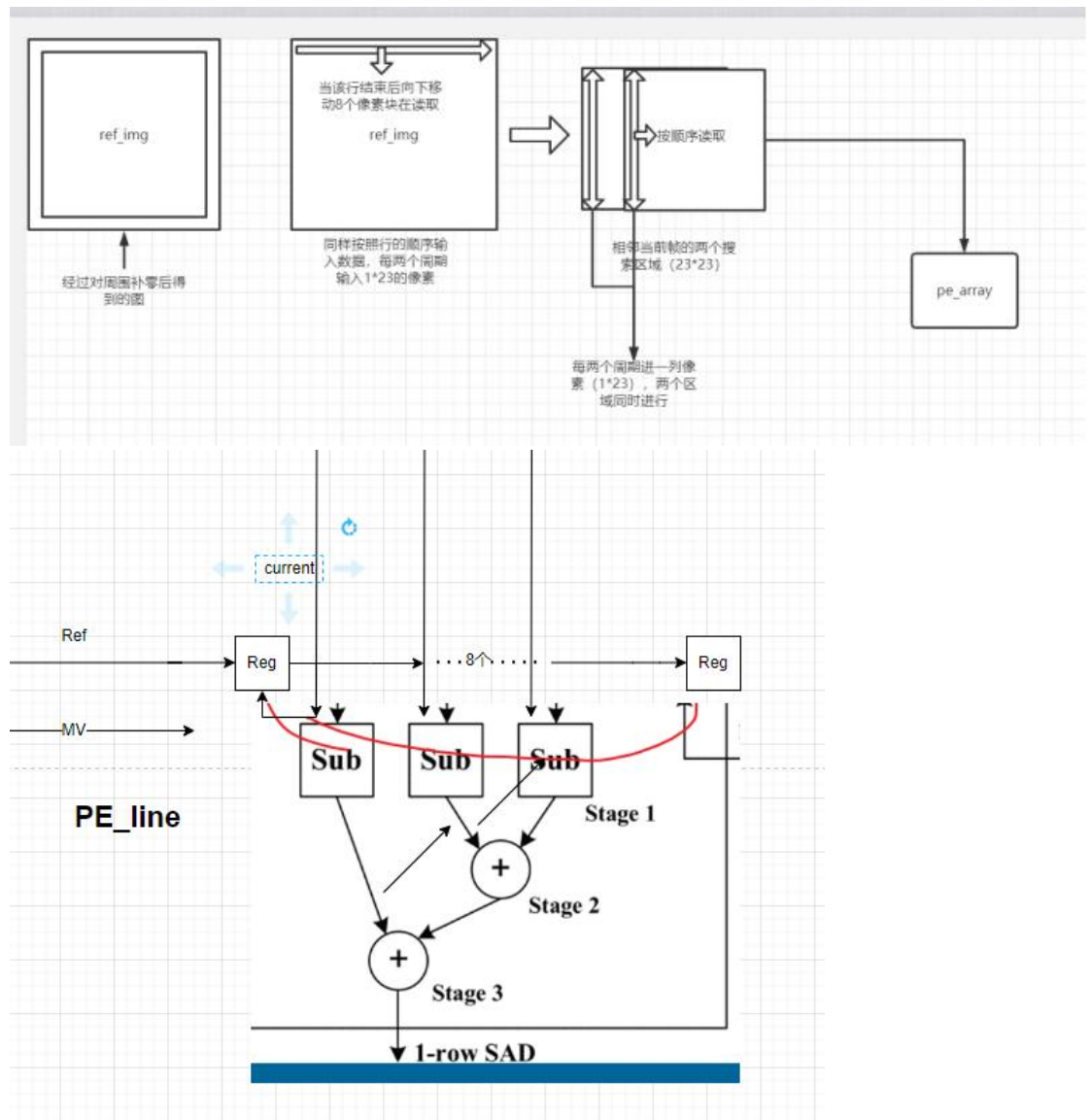
2.电路性能分析与电路结构设计

1.



如图，本架构的主要思想是每个周期计算一列 16 个 sad 块，然后更新参考帧数据，具体在 pe_line 图中的 8 个 reg。





虽然是参考帧的数据在不断流动，但是我们可以想像，这就好比当前块在搜索区域内滑动，这和卷积操作很相似，当前块为卷积核，不过不同之处在于我们一次计算一列内的所有值。

由于使用一套计算模块，当我们计算下一个当前块的时候会出现参考帧的数据接不上的情况，但是隔一个当前块刚好接上，但是由于需要，不能间隔输出，因此我们通过并行技术复制一份一样的模块，但是这样又会让性能过剩，为了减小发热，以及提高寿命，在一个周期停顿不计算，一个周期计算，这样性能不会改变。由于上述分析它和初始模块差了 16 个周期，因此也需要另一个 **cur_reg** 存储下一个当前块。

具体每个小模块：

cur_addr 和 **ref_addr** 产生需要的数据地址。

cur_reg: 存储当前块数据，有相对于当前 **sad** 二倍大小的空间（为了切换当前块的时候不停顿）。

pe,pe_line: 核心计算模块（同时也存储了参考帧数据）。

cmp: 比较一个周期来的 16 个 **sad** 值，选出最小的，并和全局的最小值比较，16 个周期后输出当前 **sad** 的最小值。

电路指标:

性能:

1/60 秒内需要计算 $3840 \times 2160 / 8 / 8 = 129600$ 个当前块的数据, 一个当前块需要 16 个周期, 200Mhz 带入计算出需要 0.010368s 计算一张图片, 计算 60 张需要 0.622s 小于 1s, 1s 内能计算约 96 张图片。

带宽:

```
module me(  
    input wire clk,  
    input wire rst,  
  
    input wire [15:0] cur_input0,  
    input wire [15:0] cur_input1,  
  
    input wire [3:0] ref0, input wire [3:0] ref1, input wire [3:0] ref2, input wire [3:0] ref3, input wire [3:0] ref4,  
    input wire [3:0] ref5, input wire [3:0] ref6, input wire [3:0] ref7, input wire [3:0] ref8,  
    input wire [3:0] ref9, input wire [3:0] ref10, input wire [3:0] ref11, input wire [3:0] ref12,  
    input wire [3:0] ref13, input wire [3:0] ref14, input wire [3:0] ref15, input wire [3:0] ref16,  
    input wire [3:0] ref17, input wire [3:0] ref18, input wire [3:0] ref19, input wire [3:0] ref20,  
    input wire [3:0] ref21, input wire [3:0] ref22,  
    output reg finish_flag,
```

由图可见, 输入带宽: $2 + 32 + 4 \times 23 = 126\text{bits}$

输出带宽:

```
output reg finish_flag,  
output reg ber,  
output reg [22:0] cur_ad,  
output reg [22:0] ref_ad,  
output reg [13:0] min_sad0,  
output reg [13:0] min_sad1, //这个只能是reg?  
  
output reg finish_a_cur0, output reg finish_a_cur1,  
output reg [3:0] mv_x0,  
output reg [3:0] mv_y0,  
output reg [3:0] mv_x1,  
output reg [3:0] mv_y1
```

$2 + 46 + 28 + 2 + 16 = 94\text{bits}$

利用效率:

输入: 参考帧数据一直输入, 当前帧的数据也只是在换行的时候停顿 32 个周期, 而一行内一共有 3840 个周期, 这很微小, 因此可以近似为 100%

输出效率:

每 16 个周期输出一次, 因此效率为 1/16, 当然也可以进行改进为近 100%, 每个周期输出所有数据的 16 份中的一份, 让后续模块整合 16 个周期的数据为完整的数据即可。

PPA:

功耗: 51.2346mW

面积: 芯片面积: 1646578nm^2

2.理论分析:

1s 内一共要处理的数据量: $1\text{s} \times 60 \times 3840 \times 2160 \times 8\text{bit}$
 $= 3981312000\text{bit}$, 一次 sad 计算处理 $8 \times 8 \times 8\text{bit} = 512\text{bit}$, 因此 1s 内一共要处理 7776000 个块

$$\frac{1}{2 \times 10^8 \text{ Hz}} = T \text{ (周电路周期)}$$

$$\frac{1 \text{ s}}{T} \cdot x = 7776000 \sqrt{16 \times 16}$$

$$x = 9.95 \approx 10 \text{ (每周期处理 10 个 sad)}$$

由此得到最少一个周期需要计算 10 个 sad 值，为了避免电路过于复杂，我们直接采取一个周期计算一系列的 sad 值，也就是 16 个 sad 值，这也就意味着性能会超出大约 60%。

同时根据全加器的延迟为 0.17ns，频率为 200mhz

$1 \times 10^9 / (2 \times 10^8 \times 0.17) = 29.5$ 级，因此全加器一个周期可以有 29.5 级而算一个 8bit 的加法，需要 8,4,2,1 共 4 级，一行 8 个点，加起来需要 3 级，一个 sad8 行，需要三级，一共 16 个 sad 相互比较，需要 4 级，一共需要 14 级远小于 29.5，满足要求。同时为了能够让频率更快，以及更好的区分计算 sad 值和比较 sad 值的功能，将其分成 pe 和 cmp 两个功能模块，两模块之间用 reg 分隔做流水线。

性能瓶颈：

根据上述的全加器级数，显然应该是数据从 pe 进到出 pe 的路径为关键路径，这是电路的性能瓶颈。

提升性能方法：

- 增加流水线级数
- 并行
- 采用更先进的架构（改进电路一个周期停顿一个周期计算的缺点）。

3.电路 RTL 模型与仿真验证

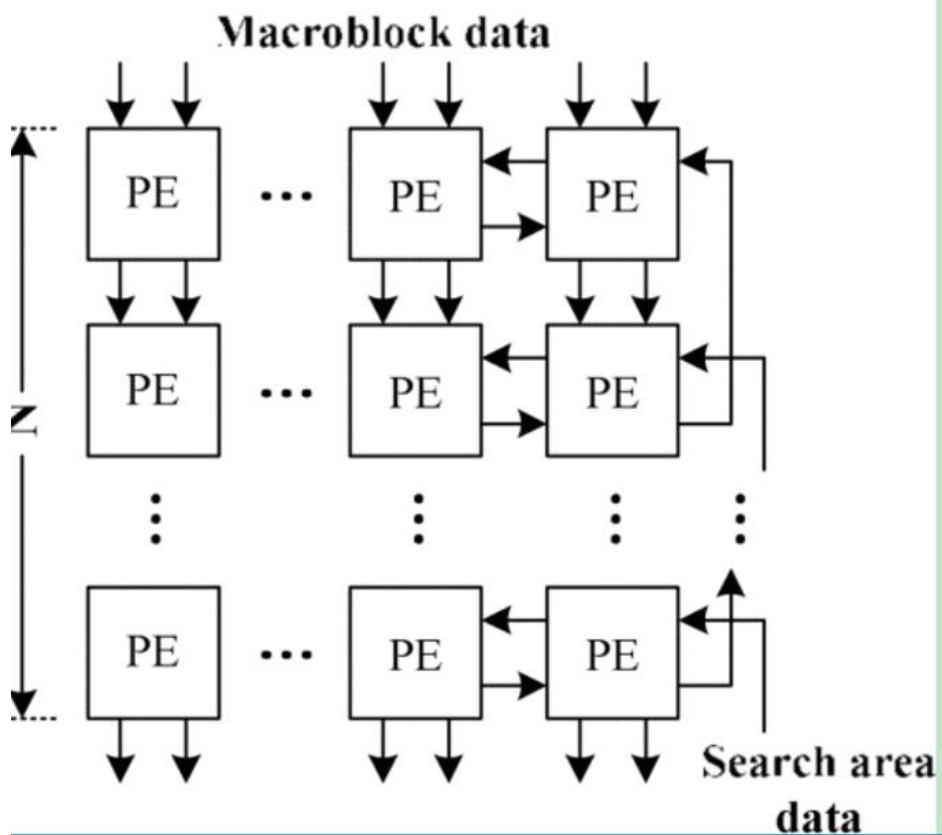
1. cur_addr.v 和 ref_addr.v:

根据电路架构功能输出地址给 me_tb.v, me_tb.v 返回数据给芯片。

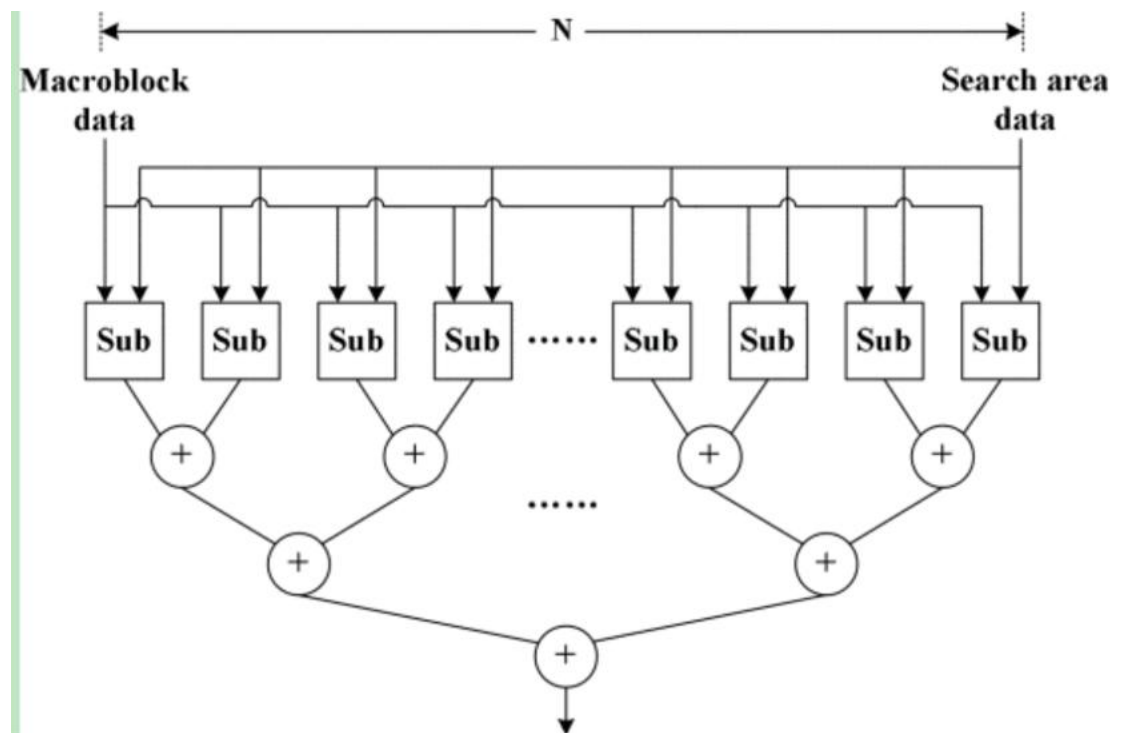
2. cur_reg.v:

由于当前块数据需要保存 16 个周期不变, 为了减小 io 数量, 提高性能, 使用了两个 cur_reg 模块来存储当前块数据, 之所以用两个是因为一个供给计算模块计算, 另一个同时更新下一个当前块的数据, 避免停顿问题。

3. pe_array.v,pe.v,pe_line.v 如二板块所描述, 电路核心计算模块, 采用混合架构 (收缩阵列架构和树结构混合)。
收缩阵列架构:



4. 树架构:



5. cmp.v:

在计算 16 个 sad 值的同时产生移动向量 mv_x 和 mv_y

6. delay.v:

当前块数据延迟，使能信号延迟

7. output_tb.v:

输出结果到文件

验证结果：

结果和参考结果对比，正确。

仿真性能：

由之前的性能分析，计算 60 张需要 0.622s 小于 1s，符合要求。

在该部分的工作：

1. 图像预处理: 电路整体以当前帧和搜索区域的一列为整体进行输入，所以在选择输入之前，需要对图像进行预处理，对于先前帧而言，需要对其周围[-7,8]范围内进行补零操作，在补零完成后以列的顺序存进寄存器中，而对于当前帧，为了方便对所需块的读取，按照 1*8 的

大小从左向右进行读入，即每次存储 8 个像素点，向右读取，直至本行读完，然后进行后换行操作。（该部分操作由脚本完成）

2. 读入数据并产生相应地址：由于整体架构的设计，本组并未采用 SRAM，所以在数据的读取与处理上，是由自己产生地址。对于当前帧而言，基于当前帧处理后的存储方式，只需要按照顺序读取固定数量像素点，即可输入当前帧的完整数据，同时由于架构的设计，需要两个周期输入 1×8 的像素值，所以存在标志位来判断当前周期读取像素的前八位或后八位；对于先前帧而言，在数据的输入上存在一定重复，所以无法直接按照顺序读取，基于对先前帧的预处理结果，需要两个参数来确定所需像素值的坐标，其中 x 向量决定所读像素所在的列数，而 y 向量决定像素所在的行的位置，对于该架构而言，每两个周期需要读入 1×23 的像素，同样存在标志位进行判断。在 testbench 中，将处理过的数据读入到 `mems` 中，并在相应周期对地址指向的像素值进行输出，并同未经处理过的原数据在关键部分进行对比，如换列后的值，换行后的值，图片最终的值等进行比较。得到结果与原图片像素一致，符合预期功能。

3. 地址产生模块的验证：

当前帧数据的存储与转换：对于架构而言，先前帧不需要进行长时间存储与保持，而当前帧数据在进入结构后需要仍保持 32 个周期不变化，所以需要两个寄存器阵列对其进行存储，当其中一个寄存器进行读取时，另一个寄存器进入保持状态，持续的将数据传输给后方 PE

模块，当达到 32 周期时输出的寄存器清空，同时开始存储即将用到的数据，另一个寄存器开始保持状态持续的对 PE 部分输出之前所存储的数据。在整个部分的难点是在达到 32 周期时寄存器状态的改变，开始时存储的数据会存在一个像素点的误差，在对使能信号调整后，使其满足了预期功能。对于该部分的验证，给其部分实现编好的数据，让其读取与输出，观察其波形即可。

4. 电路逻辑综合策略与综合结果

1. 逻辑综合流程

第一步：设置 lib 以及 search path

```
set search_path "$search_path ../rtl/idct ../scripts /home/student/Desktop/Work  
space/55nm ../work ../mem/asdrispkb1p64x16cm2sw0/tt1p2v25c"  
set target_library "hu55npkldut_tt1p0v25c.db HL55LPGP3VDS_SL_A01_P2_TT1D8V1D2  
V25C.db asdrispkb1p64x16cm2sw0_lib.db"  
set symbol_library "hu55npkldut.sdb"  
set link_library "* $target_library $symbol_library"
```

图 1

第二步：顶层文件的书写与例化，读取 RTL

```
analyze -format verilog ../rtl/idct/cmp.v  
analyze -format verilog ../rtl/idct/cur_addr.v  
analyze -format verilog ../rtl/idct/cur_reg.v  
analyze -format verilog ../rtl/idct/delay.v  
analyze -format verilog ../rtl/idct/ref_addr.v  
analyze -format verilog ../rtl/idct/sub.v  
analyze -format verilog ../rtl/idct/pe_line.v  
analyze -format verilog ../rtl/idct/pe.v  
analyze -format verilog ../rtl/idct/pe_array.v  
analyze -format verilog ../rtl/idct/me.v  
analyze -format verilog ../rtl/idct/me_chip.v  
elaborate me_chip
```

图 2

```
module me_chip (  
    input wire clk,  
    input wire rst,  
  
    input wire [15:0] cur_input0,  
    input wire [15:0] cur_input1,  
  
    output reg finish_flag,  
  
    input wire[3:0] ref0, input wire[3:0] ref1, input wire[3:0] ref2, input wire[3:0] ref3, input wire[3:0] ref4,  
    input wire[3:0] ref5, input wire[3:0] ref6, input wire[3:0] ref7, input wire[3:0] ref8,  
    input wire[3:0] ref9, input wire[3:0] ref10, input wire[3:0] ref11, input wire[3:0] ref12,  
    input wire[3:0] ref13, input wire[3:0] ref14, input wire[3:0] ref15, input wire[3:0] ref16,  
    input wire[3:0] ref17, input wire[3:0] ref18, input wire[3:0] ref19, input wire[3:0] ref20,  
    input wire[3:0] ref21, input wire[3:0] ref22,  
  
    output reg ber,  
    output reg [22:0] cur_ad,  
    output reg [22:0] ref_ad,  
    output reg [13:0] min_sad0,  
    output reg [13:0] min_sad1, // 这个只能是reg?  
  
    output reg finish_a_cur0, output reg finish_a_cur1,  
    output reg [3:0] mv_x0,  
    output reg [3:0] mv_y0,  
    output reg [3:0] mv_x1,  
    output reg [3:0] mv_y1  
);
```

图 3.me_chip 顶层文件

共包括 230 个 pad，其中 136 个为输入 pad，94 个为输出 pad。

第三步：设置约束

时钟周期为 5ns，其他参数按照周期等比例缩放。

```
create_clock -period 5 [get_ports clk]
set_clock_uncertainty 0.25 [get_clocks clk]
set_clock_transition 0.1 [get_clocks clk]

##### YBR #####
set_clock_latency -source 4 [get_clocks clk]
set_clock_latency 0.3 [get_clocks clk]
##### YBR #####

# 2.2
set_input_delay -max 2.5 -clock clk [remove_from_collection [all_inputs] [get_ports clk]]
# 2.3
set_output_delay -max 2.5 -clock clk [all_outputs]
```

第四步：compile

2. 逻辑综合结果

1)时序结果

Point	Incr	Path

clock clk (rise edge)	0.00	0.00
clock network delay (ideal)	4.30	4.30
inst_me/ber_reg/CK (SVL_FDPRBQ_1)	0.00 #	4.30 r
inst_me/ber_reg/Q (SVL_FDPRBQ_1)	0.23	4.53 r
inst_me/ber (me)	0.00	4.53 r
HPDWUW1416DGP_ber/PAD (HPDWUW1416DGP)	1.90	6.43 r
ber (out)	0.00	6.43 r
data arrival time		6.43
clock clk (rise edge)	5.00	5.00
clock network delay (ideal)	4.30	9.30
clock uncertainty	-0.25	9.05
output external delay	-2.50	6.55
data required time		6.55

data required time		6.55
data arrival time		-6.43

slack (MET)		0.12

1

2) 面积及资源使用结果

总面积为 1692353 平方微米，详细资源使用结果见下图。

```

Report : area
Design : me_chip
Version: 0-2018.06-SP1
Date   : Mon Apr 18 13:38:23 2022
*****

Library(s) Used:

    hu55npkldut_tt1p0v25c (File: /home/student/Desktop/Workspace/55nm/hu55npkldut_tt1p0v25c.db)
    HL55LPGP3VDS_SL_A01_TT1D8V1D2V25C (File: /home/student/Desktop/Workspace/55nm/
    HL55LPGP3VDS_SL_A01_P2_TT1D8V1D2V25C.db)

Number of ports:          229712
Number of nets:           505172
Number of cells:          244949
Number of combinational cells: 198285
Number of sequential cells: 38747
Number of macros/black boxes: 220
Number of buf/inv:        91004
Number of references:      4

Combinational area:       543527.865505
Buf/Inv area:             88170.317663
Noncombinational area:    301825.717290
Macro/Black Box area:     847000.000000
Net Interconnect area:    undefined (Wire load has zero net area)

Total cell area:          1692353.582795
Total area:               undefined
1

```

3) 功耗结果

除去 io_pad 的功耗，芯片的功耗为 50.7025mW

Power Group (%) Attrs	Internal Power	Switching Power	Leakage Power	Total Power
io_pad (81.32%)	18.4651	202.2242	2.5926e+03	220.6925
memory (0.00%)	0.0000	0.0000	0.0000	0.0000
black_box (0.00%)	0.0000	0.0000	0.0000	0.0000
clock_network (0.00%)	0.0000	0.0000	0.0000	0.0000
register (15.30%)	40.1817	1.2585	6.9610e+04	41.5111
sequential (0.00%)	4.6645e-05	5.5282e-06	39.3468	9.1519e-05
combinational (3.39%)	4.5880	4.4391	1.6418e+05	9.1914
Total	63.2349 mW	207.9219 mW	2.3642e+05 nW	271.3950

4) 设计违例

最小电容违例

min_capacitance

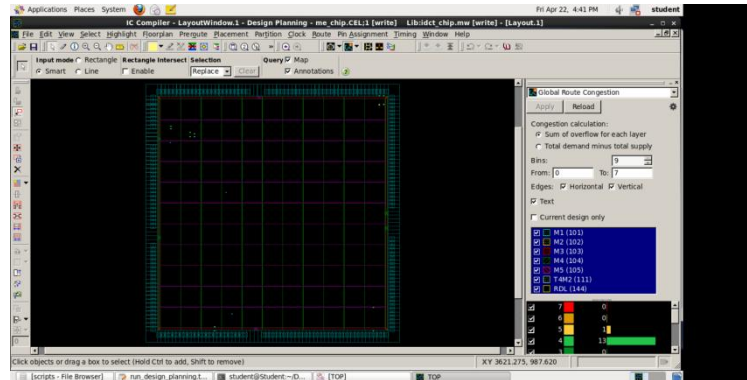
Net	Required Capacitance	Actual Capacitance	Slack	
clk	6.43	1.43	-5.00	(VIOLATED)
cur_input0[0]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[1]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[2]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[3]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[4]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[5]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[6]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[7]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[8]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[9]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[10]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[11]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[12]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[13]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[14]	6.43	1.43	-5.00	(VIOLATED)
cur_input0[15]	6.43	1.43	-5.00	(VIOLATED)
cur_input1[0]	6.43	1.43	-5.00	(VIOLATED)
cur_input1[1]	6.43	1.43	-5.00	(VIOLATED)
cur_input1[2]	6.43	1.43	-5.00	(VIOLATED)
cur_input1[3]	6.43	1.43	-5.00	(VIOLATED)

5. 电路物理实现与结果分析

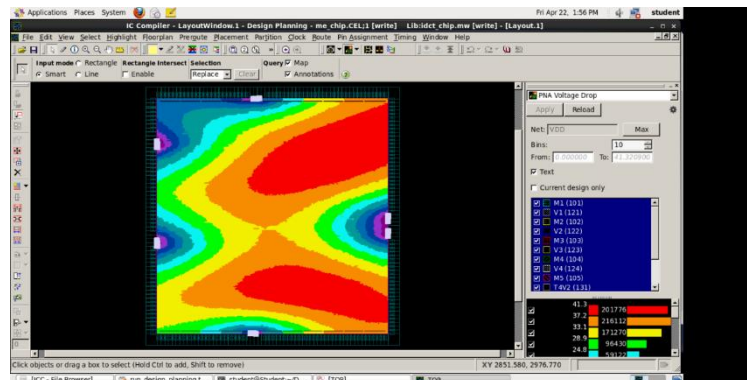
(1) ICC 设计电路包括，版图规划，布局，时钟树综合，布线

一下是各步骤结果：

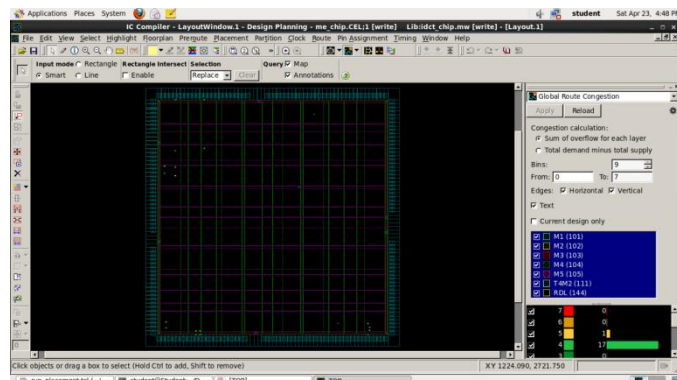
版图规划：芯片概貌图



压降

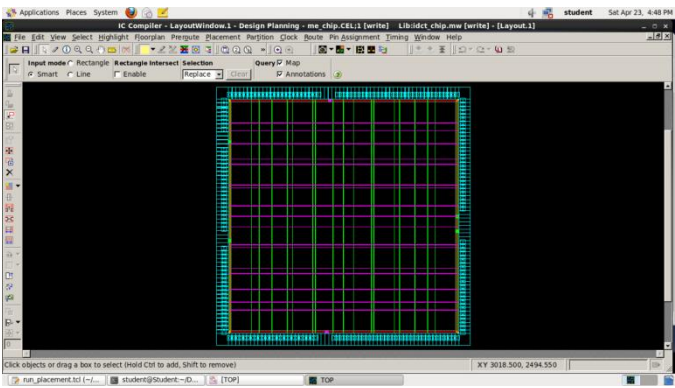


拥塞



Placement

芯片概貌图



时序结果

```
student@Student:~/Desktop/Workspace/IDCT_55nm_mem/ICC/work
File Edit View Search Terminal Help
inst_me/ref_addr/mult_48/FS_1/SUM_17_ (ref_addr_DW01_add_1)
0.00 7.48 r
inst_me/ref_addr/mult_48/PRODUCT_19_ (ref_addr_DW02_mult_0)
0.00 7.48 r
inst_me/ref_addr/U51/X (SVL_E02_S_2)
0.00 * 8.28 f
inst_me/ref_addr/ad1[19] (ref_addr)
0.00 8.28 f
inst_me/ref_ad_reg_19/_D (SVL_FDPRBQ_F_2)
0.16 * 8.44 f
data arrival time
8.44

clock clk (rise edge)
5.00 5.00
clock network delay (ideal)
4.30 9.30
clock uncertainty
-0.25 9.05
inst_me/ref_ad_reg_19/_CK (SVL_FDPRBQ_F_2)
0.00 9.05 r
library setup time
-0.25 8.80
data required time
8.80
-----
data required time
8.80
data arrival time
-8.44
-----
slack (MET)
0.36

1
icc_shell>
```

时钟树综合

时序结果

```
student@Student:~/Desktop/Workspace/IDCT_55nm_mem/ICC/work
File Edit View Search Terminal Help
inst_me/ref_addr/U58/X (SVL_AN2_0P5)
0.15 * 7.14 r
inst_me/ref_addr/U56/X (SVL_AN2_0P5)
0.12 * 7.26 r
inst_me/ref_addr/U54/X (SVL_AN2_0P5)
0.11 * 7.36 r
inst_me/ref_addr/U52/X (SVL_AN2_0P5)
0.12 * 7.48 r
inst_me/ref_addr/U51/X (SVL_E02_S_2)
0.72 * 8.20 f
inst_me/ref_addr/ad1[19] (ref_addr)
0.00 8.20 f
inst_me/ref_ad_reg_19/_D (SVL_FDPRBQ_F_2)
0.16 * 8.37 f
data arrival time
8.37

clock clk (rise edge)
5.00 5.00
clock network delay (ideal)
4.30 9.30
clock uncertainty
-0.20 9.10
inst_me/ref_ad_reg_19/_CK (SVL_FDPRBQ_F_2)
0.00 9.10 r
library setup time
-0.24 8.86
data required time
8.86
-----
data required time
8.86
data arrival time
-8.37
-----
slack (MET)
0.50

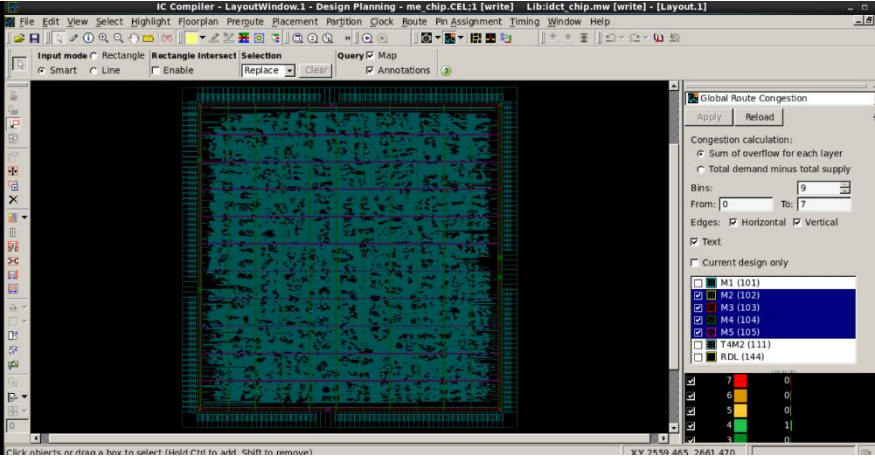
1
icc_shell>
```

布线

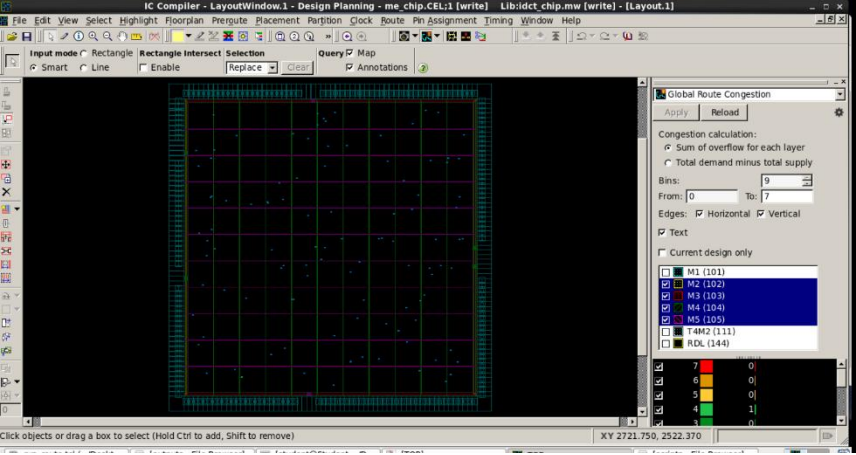
时序结果

File	Edit	View	Search	Terminal	Help
inst me/ref_addr/mult 48/U26/X (SVL E02 1)	0.10	6	6.07	r	
inst me/ref_addr/mult 48/U18/X (SVL A02 0PS)	0.12	6	6.20	r	
inst me/ref_addr/mult 48/U4 3/S (SVL A00F 1)	0.18	6	6.37	f	
inst me/ref_addr/mult 48/U65/X (SVL E02 1)	0.07	6	6.45	f	
inst me/ref_addr/mult 48/FS 1/A 12 (ref_addr_DW01_add 1)	0.00	6	6.45	f	
inst me/ref_addr/mult 48/FS 1/U59/X (SVL NR2 0PS)	0.11	6	6.56	r	
inst me/ref_addr/mult 48/FS 1/U02/X (SVL INV 1)	0.04	6	6.60	f	
inst me/ref_addr/mult 48/FS 1/U53/X (SVL A0121 0PS)	0.15	6	6.74	r	
inst me/ref_addr/mult 48/FS 1/U51/X (SVL A0121 0PS)	0.10	6	6.85	f	
inst me/ref_addr/mult 48/FS 1/U43/X (SVL A0121 0PS)	0.17	6	7.02	r	
inst me/ref_addr/mult 48/FS 1/U41/X (SVL A0121 0PS)	0.10	6	7.12	f	
inst me/ref_addr/mult 48/FS 1/U33/X (SVL A0121 0PS)	0.16	6	7.28	r	
inst me/ref_addr/mult 48/FS 1/U32/X (SVL D02 0PS)	0.14	6	7.41	r	
inst me/ref_addr/mult 48/FS 1/S0R 17 (ref_addr_DW01_add 1)	0.00	6	7.41	r	
inst me/ref_addr/mult 48/PRODUCT 19 (ref_addr_DW02_mult 0)	0.00	6	7.41	r	
inst me/ref_addr/U51/X (SVL E02 5 2)	0.61	c	8.03	f	
inst me/ref_addr/ad119 (ref_addr)	0.00	6	8.03	f	
inst me/ref_ad_reg 19 /D (SVL F0PRBQ F 2)	0.15	c	8.18	f	
data arrival time			8.18		
clock clk (rise edge)	5.00		5.00		
clock network delay (ideal)	4.30		9.30		
clock reconvergence pessimism	0.00		9.30		
clock uncertainty	-0.20		9.10		
inst me/ref_ad_reg 19 /CK (SVL F0PRBQ F 2)	0.00	6	9.10	r	
library setup time	-0.23		8.87		
data required time			8.87		
data arrival time			8.18		
slack (MET)			0.69		

Finish



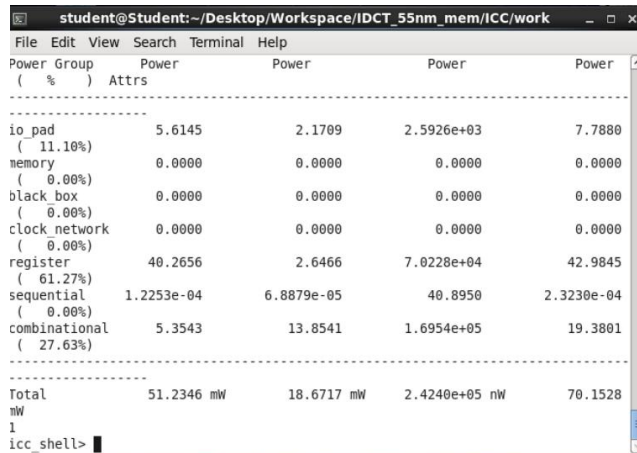
拥塞



面积结果

```
Area
-----
Combinational Area: 497640.067265
Noncombinational Area: 301937.997286
Buf/Inv Area: 44641.519311
Total Buffer Area: 15376.76
Total Inverter Area: 29264.76
Macro/Black Box Area: 847000.000000
Net Area: 0.000000
Net XLength : 3409100.50
Net YLength : 5150305.00
-----
Cell Area: 1646578.064551
Design Area: 1646578.064551
Net Length : 8559406.00
```

功耗结果



Power Group	Power	Power	Power	Power
(%)	Attrs			
io pad (11.10%)	5.6145	2.1709	2.5926e+03	7.7880
memory (0.00%)	0.0000	0.0000	0.0000	0.0000
black box (0.00%)	0.0000	0.0000	0.0000	0.0000
clock network (0.00%)	0.0000	0.0000	0.0000	0.0000
register (61.27%)	40.2656	2.6466	7.0228e+04	42.9845
sequential (0.00%)	1.2253e-04	6.8879e-05	40.8950	2.3230e-04
combinational (27.63%)	5.3543	13.8541	1.6954e+05	19.3801
Total	51.2346 mW	18.6717 mW	2.4240e+05 nW	70.1528 mW

1
icc_shell>

资源使用结果

```
Design Rules
-----
Total Number of Nets: 212172
Nets With Violations: 359
Max Trans Violations: 0
Max Cap Violations: 3
Max Fanout Violations: 356
-----

Hostname: Student

Compile CPU Statistics
-----
Resource Sharing: 0.00
Logic Optimization: 0.00
Mapping Optimization: 182.91
-----
Overall Compile Time: 197.09
Overall Compile Wall Clock Time: 199.59
-----
```

从上述结果可以看到，我们的 ICC 设计，总体来看效率很高，使用的 IO，面积，资源使用结果较多，在最初阶段存在一定的拥塞，但在经过优化后，拥塞得到了完美解决。压降最大为 $40\text{mV} < 50\text{mV}$ ，时序也均满足，符合要求。

6.任务分工与设计总结

贾鑫鹏:

参与架构设计的讨论，对数据的预处理。

RTL 代码中，地址产生模块 `cur_addr`, `ref_addr`，与当前帧存储模块 `cur_reg` 的编写与调试，`testbench` 部分内容。

后期代码运行中结果存在 bug，参与调试与解决。

问题:

Verilog 代码熟悉程度不高。

查资料熟悉代码内容，并对代码进行修改。

数据预处理

通过 `cpp` 脚本进行处理，在过程中出现了多次错误的处理

最终通过与原数据的多次对比，得到正确结果。

小模块中时序问题。

通过观察输出波形进行调试，改变使能信号与判断信号达到要求。

吴非:

电路架构图绘制和架构的设计

RTL 代码中 `pe`, `pe_line` 基本计算模块，`cmp` 比较器，输出到文件以及延迟模块的编写(包括相关模块的 `testbench`)

代码最后的整合和测试，以及后续逻辑，物理综合不符合后的修改。

问题:

Verilog 代码基本语法的熟悉:

问助教，上网查阅，同学讨论，编写 `testbench` 验证想法是否正确。

电路架构有功能 bug:

查阅资料，小组讨论修改架构和 RTL 代码。

代码调试:

换行后 32 个周期的停顿需要很准确的控制各个使能信号，通过查看波形调试，这部分所占用的时间甚至超过代码编写的时间。

前期讨论和进展不顺利:

定期督促和开会讨论。

王玉麟

参与电路架构的讨论

RTL 代码中，`cmp` 比较器，输出到文件及 `testbench`

物理综合

问题及解决思路

电路架构有功能 bug:

查阅资料，小组讨论修改架构和 RTL 代码

输入违例

输入时先经过 reg 再输入

电压压降，

先定义全部 pad 位置，调整电源 pad 位置

时序

调整 pad 位置，使布线距离更短

电路 skew 较大

优化时，设置调整参数

绕线偏差

实际 placement 存在误差，每次布局优化后，都需 route_zrt_global 重新评估

汪嘉杭

参与电路架构的讨论

前期 RTL 代码中的 mux 和 decoder，以及 testbench

pe 模块例化文件 pe_chip 和顶层文件的例化 me_chip 的编写

逻辑综合

问题及解决思路

时序不满足要求

通过分析时序报告来发现优化策略