



Custom Intelligent Hardware (CiH) Laboratory

一个 10 位开源逐次比较型模数转换器

作者（排名不分先后）：
李沛哲
张久山

项目负责人：
陈迟晓 副研究员

Sponsorship/Grant:
国家重点研发计划

September 7, 2021



Revision History

August 19, 2021 – create the document v0.1
September 7, 2021 – last updated

Contents

1 RAIL 及设计流程介绍	1
1.1 RAIL 标准单元库介绍	1
1.2 基于 RAIL 的定制集成电路设计流程	1
2 设计实例——10 bit 65 MSPS SAR ADC	3
2.1 ADC 结构及其子电路设计	3
2.1.1 ADC 结构	3
2.1.2 棚压自举开关	3
2.1.3 比较器	8
2.1.4 DAC 阵列	9
2.1.5 压控延时单元 (VCDL)	12
2.2 ADC 版图设计	13
2.2.1 ADC Core 部分版图	13
2.2.2 ADC Full Chip 版图	14
3 ADC 后仿真结果	17

1 RAIL 及设计流程介绍

1.1 RAIL 标准单元库介绍

RAIL (Resilient Analog Instance language/Library) 是一款用于定制数模混合集成电路敏捷设计的标准单元库。RAIL 目前有 29 种共 51 个不同功能及尺寸的标准单元，可以支持 ADC 和模拟运算等电路的敏捷设计。有经验的开发人员使用 RAIL 及其设计流程仅需一个星期就可以设计出一个中速中高精度的 ADC，包括电路的前仿真、Full Chip 版图、及后仿真及 GDS 文件。

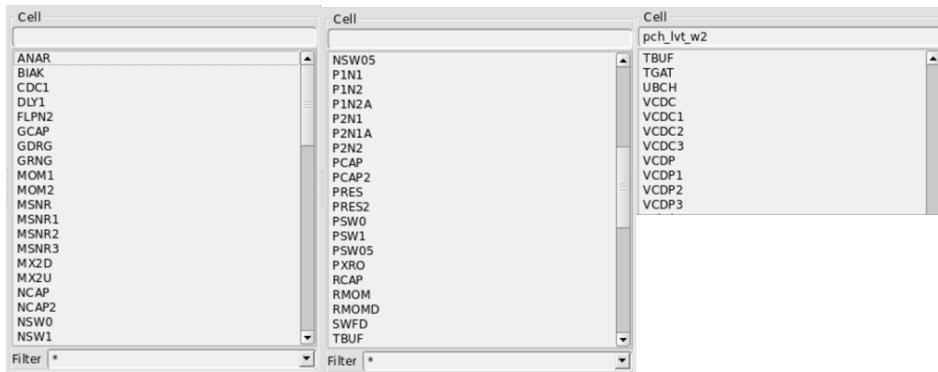


Fig. 1.1 RAIL65 库的标准单元

使用 RAIL 进行电路设计具有多种优势。

RAIL 的设计流程可以大大缩短设计周期。由于采用标准单元库设计，开发人员不需要具体调整每一个 MOS 管的宽长比，也不需要为标准单元专门绘制版图，因此电路的开发周期可以很短。

开发者可以使用 RAIL 搭建多种类型的电路。除了 RAIL 本身包含的 MOM 电容、数控电容、传输门开关、放电单元等，RAIL 目前拥有的标准单元支持开发者设计运放、偏置电路、比较器、栅压自举开关、电容 DAC 阵列以及压控延时单元等。这些电路搭配标准数字单元可以搭建出多种大规模的数模混合电路。

基于 RAIL 的设计较纯手工设计更容易布局布线，也更容易对电路进行修改。因为 RAIL 设计流程基于标准单元库，并且在绘制版图时使用类似于纯数字电路的方法，可以使用 ICC 进行自动布线，因此布线操作简化很多。如要对设计作出修改，只需要修改或新建标准单元，然后重新手动或自动布局，最后再自动布线即可。

由于 RAIL 的开源属性，任何人可以对 RAIL 的标准单元进行添加和修改，包括原理图和版图的修改，这使得开发者在开发定制电路的过程中根据自己需要构建电路单元。由于标准单元的复用属性，只需要对标准单元进行一次修改或添加就可以在电路设计中反复调用，不需要再重新绘制版图，这也使得 RAIL 可以在被众多开发者使用的过程中趋于完善。

1.2 基于 RAIL 的定制集成电路设计流程

RAIL 主要用于数模混合电路的设计，这里以 ADC 为例介绍基于 RAIL 的数模混合电路设计流程。设计流程图如下所示。

首先需要确定 ADC 的整体架构，即 ADC 需要哪些模拟电路以及 ADC 的什么样的数字逻辑。数字逻辑可以使用数字标准单元库搭建原理图，或用 VHDL 或 Verilog 生成原理图。模拟电路则使用 RAIL 中的标准单元进行原理图的搭建。在拥有模拟部分的原理图以及数字逻辑后，将这两部分合并成完整的 ADC 进行仿真，以验证 ADC 功能的正确性以及性能指标是否达到要求。

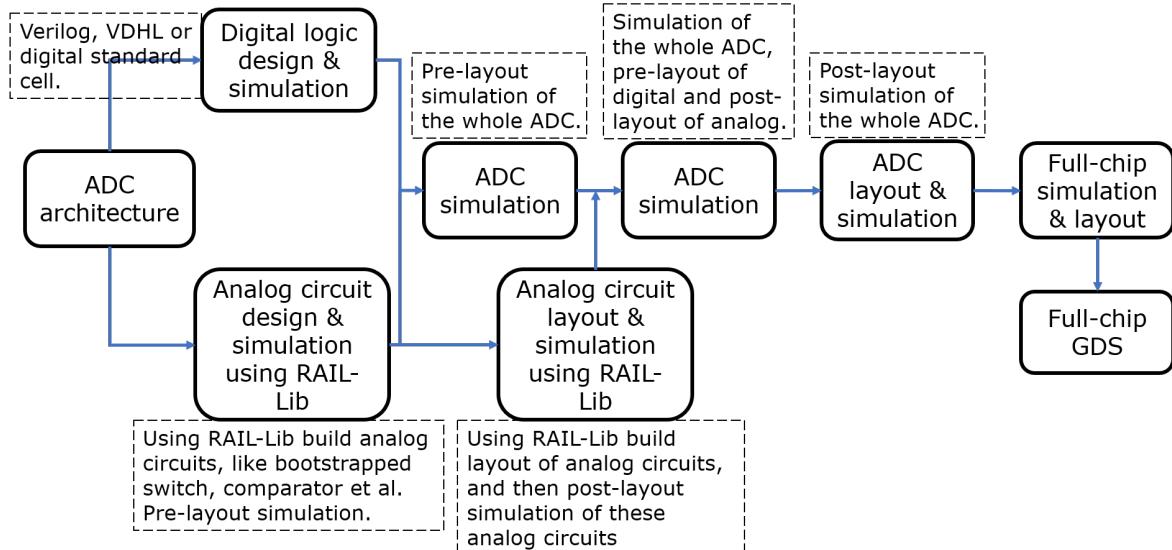


Fig. 1.2 基于 RAIL 的电路设计流程

在 ADC 的前仿真完成确保其性能指标达标后，开始绘制模拟电路的版图。ADC 的模拟电路一般是用 RAIL 的标准单元构建的，标准单元已经包含了对应的版图，不过在模拟器件内部仍需对标准单元进行布局和布线。在完成各个模拟电路版图的绘制并完成 LVS 与 DRC 的检查后，提取模拟电路模块的寄生参数将其放入 ADC 中对应的模拟模块中对 ADC 进行仿真，以验证此时 ADC 的功能和性能指标。

在确保 ADC 的功能和性能指标达标后，进行 ADC Core 部分的版图绘制。该步骤主要是对之前模拟单元的版图以及数字逻辑的版图进行布局和布线，一般包括采样开关、比较器、DAC 阵列以及数字电路等。在完成 ADC Core 部分的版图设计并完成 LVS 与 DRC 检查后，提取 ADC 的寄生参数进行后仿真，验证 ADC 的功能及性能指标。

ADC Core 的后仿真无问题后，进行芯片 Full Chip 的仿真及版图绘制。Full Chip 仿真需要将 ADC Core 与芯片的 pad 连接起来进行仿真，进行最后的验证。版图绘制则需要将 ADC Core 的版图与 pad 的版图放到一起进行布局和布线。值得注意的是在绘制 Full Chip 的版图时需要使用 RAIL 中的两个单元来规避一些 DRC 错误。

2 设计实例——10 bit 65 MSPS SAR ADC

2.1 ADC 结构及其子电路设计

2.1.1 ADC 结构

该 ADC 是一个 10bit 65MSPS 的 SAR ADC，结构参考 [1]，并在此基础上进行了一些改进，即增加了 1bit 的冗余 (redundancy)，以及采用了分裂电容 (split capacitor)。ADC 的主体结构如下两图所示，它包括采样开关 (这里使用栅压自举开关)、比较器、电容 DAC 阵列以及逻辑电路。除此之外还有一个压控延时 (VCDL) 单元来控制比较器的，以给数字逻辑和 DAC 建立以足够的时间。

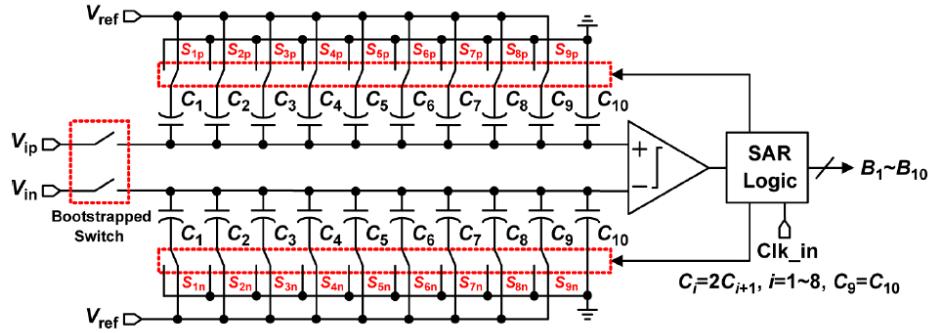


Fig. 2.3 ADC 结构 (论文)

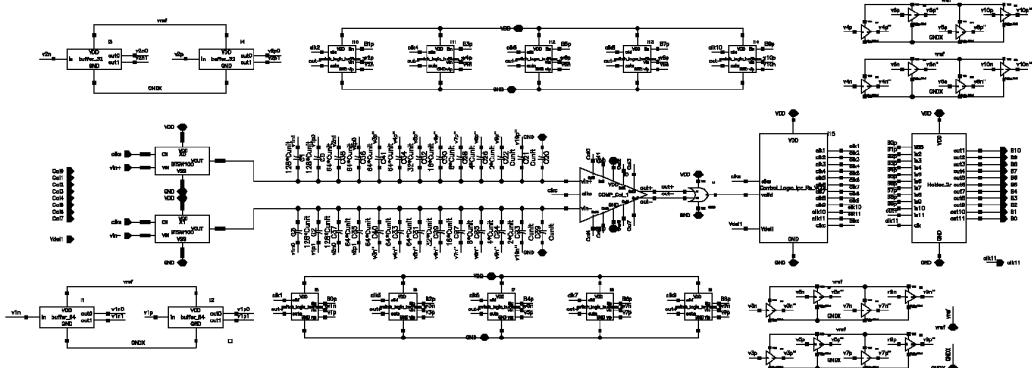


Fig. 2.4 ADC 结构 (原理图)

2.1.2 棚压自举开关

棚压自举开关的结构如下图所示。

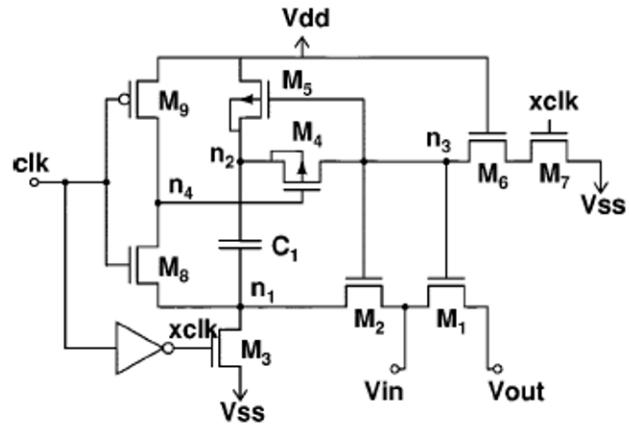


Fig. 2.5 棚压自举开关原理图

本设计中所用到的开关主要由 N、P 型传输管、MOM 电容、时钟反相器 (CKND) 和自定义类 (CDC) 构成，基于 RAIL65 库的开关电路结构及网表如下图所示。

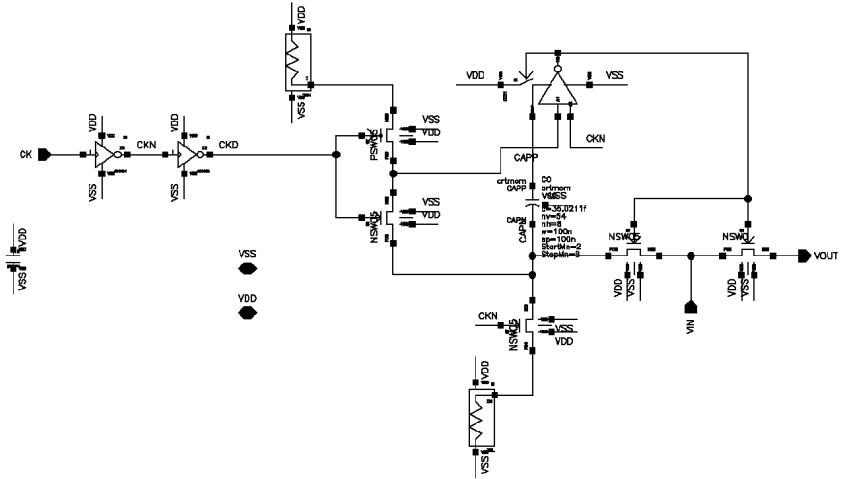


Fig. 2.6 基于 RAIL65 库的棚压自举开关电路

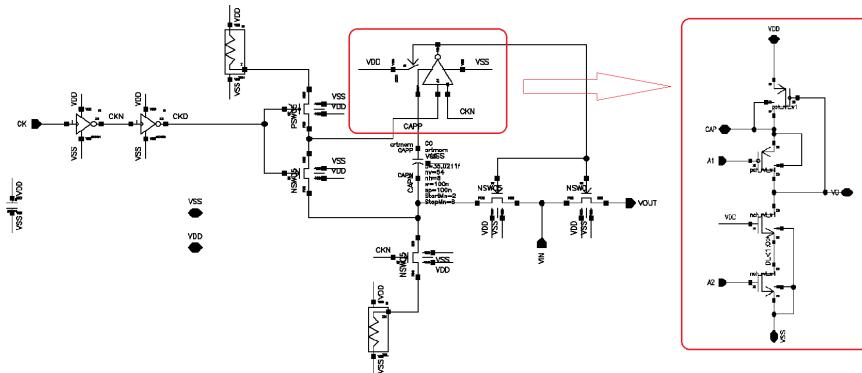


Fig. 2.7 棚压自举开关中的 CDC 器件

```

module BTSW100 (
    //inout VDD, VSS // Power Supply
    input CK,
    input VIN,
    output VOUT

);

wire VBST, UPIN;
wire CAPP, CAPN;
wire CKN, CKD;

NSW0 smp_sw0 (.GT(VBST), .POS(VOUT), .NEG(VIN));
NSW05 inp_sw0 (.GT(VBST), .NEG(VIN), .POS(CAPN));

CKND1 ck_drv (.I(CK), .ZN(CKN));
CKND0 ck_del (.I(CKN), .ZN(CKD));

NSW05 pdn_sw0 (.GT(CKN), .NEG(XVSS), .POS(CAPN));

CDC1 chinv0 (.CAP(CAPP), .A1(UPIN), .A2(CKN), .VO(VBST));

PSW05 pup_sw0 (.GT(CKD), .POS(UPIN), .NEG(XVDD));
NSW05 tin_sw0 (.GT(CKD), .NEG(UPIN), .POS(CAPN));

TIEH bvd0 (.Z(XVDD));
TIEL bvss0 (.ZN(XVSS));

DCAP4 axl0 ();

endmodule

```

Fig. 2.8 棚压自举开关网表

电路版图的绘制主要是通过在 ICC 中运行 TCL 脚本完成的，具体流程如下。

首先设置工作路径并配置库文件。部分代码如图所示，代码结构首先是设置工作路径、网表路径、GDS 输出路径，并确认当前顶层设计模型，其次是配置相关后端文件，主要包括工艺文件.tf、标准单元库文件、RC 寄生参数文件.tlplus、图层规划.map 文件、数据库.db 文件。

```

# The milkyway lib path and input/output path
set LIBR_PATH BTSW100_02
set RTL_PATH /home/ninezhang/workspace/icc/sarADC_chen/src/btsw100m_v1.v
set MODULE_NAME BTSW100
set VERSION v12
set GDS_PATH aprout/$MODULE_NAME$VERSION.gds

# The backend files provided by foundry
set TECHFILE_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/techfiles/tsmcn65_glmT2.tf
set REFLIB_PATH (/EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/cell_frame/tcbn65lp \
/home/ninezhang/workspace/MKW/MKW_RAIL/CEL_FRAM/rail65)
set TLUMAX_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/techfiles/tlplus/cln65lp_ip09m+alrdl_rcbest_top2.tlplus
set TLUMIN_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/techfiles/tlplus/cln65lp_ip09m+alrdl_rcworst_top2.tlplus
set TECH2ITF_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/techfiles/tlplus/star.map_9M
set STDCELL_DB_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Front_End/timing_power_noise/NLDm/tcbn65lp_200a
set RAILLIB_DB_PATH /home/xchen/workspace/lemon/digital/LC
set MAP_PATH /EDA/PDK/tsmCn65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/gdsout_6X2Z.map
set _host_options -max 16

set_tlu_plus_files -max_tluplus $TLUMAX_PATH \
-min_tluplus $TLUMIN_PATH \
-tech2itf_map $TECH2ITF_PATH

lappend search_path $STDCELL_DB_PATH
lappend search_path $RAILLIB_DB_PATH

# For simplicity we choose the typical case
set_app_var target_library "tcbn65lptc.db"
set_app_var link_library "tcbn65lptc.db rail65.db"

```

Fig. 2.9 开关版图的路径设置与配置文件设置

接下来导入电路网表。部分代码如图所示，代码功能主要是创建并打开 library，读取相应的网表文件。

```

create_mw_lib \
    -technology $TECHFILE_PATH \
    -mw_reference_library "$REFLIB_PATH" \
    -open $LIBR_PATH
read_verilog -top $MODULE_NAME -allow_black_box $RTL_PATH

```

Fig. 2.10 开关电路网表导入

最后是布局布线，它主要有以下几步。

规划并创建 floorplan，并添加物理约束如 core height、core width 等。

```

# The physical information set, including floorplan area and allowed routing metal
set TILE_HT 1.8
set TILE_WD 0.2
set CORE_ROW 5
set CORE_COL 71
set MOMCP_WD 21
set CORE_HT [expr {$TILE_HT * $CORE_ROW}]
set CORE_WD [expr {$TILE_WD * $CORE_COL}]
set TOP_RT_METAL M5

create_floorplan -control_type width_and_height \
    -core_width $CORE_WD \
    -core_height $CORE_HT

set_ignored_layers -min_routing M2 -max_routing_layer $TOP_RT_METAL

check_legality

```

Fig. 2.11 floorplan 规划

输入输出端口以及部分关键器件的位置规划。

```

set obj [get_terminal {"VIN"}]
set_attribute -quiet $obj layer M4
set_attribute -quiet $obj owner_port {VIN}
set_attribute -quiet $obj bbox {{2.90 0.00} {3.00 0.10}}

set_undoable_attribute [get_cells -all smp_sw0] origin {1.600 3.6}
set_undoable_attribute [get_cells -all smp_sw0] orientation {FS}
set_undoable_attribute [get_cells -all smp_sw0] is_fixed {1}

```

Fig. 2.12 端口及器件位置规划

部分器件放置完成后，剩余器件将由自动布局规划完成。

```

set_fp_placement_strategy -pin_routing_aware true
create_fp_placement -effort high

```

Fig. 2.13 器件自动布局

开关作为模拟器件，输入输出等关键模拟线需要手动绘制。

```

create_power_straps -direction horizontal -start_at 0.600 -nets {VIN} -layer M3 -width 0.32 -start_low_ends coordinate \
    -start_low_ends_coordinate 2.9 -start_high_ends coordinate -start_high_ends_coordinate 3.4 -extend_high_ends off -look_inside_std_cells
create_power_straps -direction vertical -start_at 3.000 -nets {VIN} -layer M4 -width 0.32 -start_low_ends coordinate -start_high_ends \
    coordinate -start_high_ends_coordinate 2.3 -extend_high_ends off -look_inside_std_cells
route_zrt_eco -nets {VIN}
route_zrt_detail

```

Fig. 2.14 手动布置模拟连线

部分模拟线绘制完成后，剩余连线将由自动连线完成。

布局布线完成后，为保证 VDD 和 VSS 整体相连，需要填充 FILLER（无逻辑功能，起填充连接作用）。

```
insert_stdcell_filler -cell_without_metal "FILL4 FILL2 FILL1" \
    -respect_keepout      -between_std_cells_only \
    -connect_to_power VDD -connect_to_ground VSS
```

Fig. 2.15 填充 FILLER

在栅压自举开关中所用到的电容为 MOM 电容，需要手动绘制它的版图，代码如下。

```
# MOM Cap generation
# P straps
create_power_straps -direction vertical -start_at $ORIGIN_X -nets $NODEL -layer M4 -width 0.32 -start_low_ends_coordinate -
    start_low_ends_coordinate $ORIGIN_Y -start_high_ends_coordinate -start_high_ends_coordinate $BOX_TOP -extend_low_ends off -extend_high_ends off
    -ignore_parallel_targets
Create_power_straps -direction vertical -start_at $ORIGIN_X -nets $NODEL -layer M6 -width 0.32 -start_low_ends_coordinate -
    start_low_ends_coordinate $ORIGIN_Y -start_high_ends_coordinate -start_high_ends_coordinate $BOX_TOP -extend_low_ends off -extend_high_ends off
    -ignore_parallel_targets
create_power_straps -direction horizontal -start_at [expr {$ORIGIN_Y+0.14}] -num_placement_strap $NHFIX -increment_x_or_y $PITCH -nets $NODEL
    -layer M5 -width 0.1 -start_low_ends_coordinate -start_low_ends_coordinate [expr {$ORIGIN_X-0.16}] -start_high_ends_coordinate -
    start_high_ends_coordinate $FIN_RGT -extend_low_ends off -extend_high_ends off -ignore_parallel_targets
create_power_straps -direction horizontal -start_at [expr {$ORIGIN_Y+0.16}] -num_placement_strap $NHFIX -increment_x_or_y $PITCH -nets $NODEL
    -layer M3 -width 0.1 -start_low_ends_coordinate -start_low_ends_coordinate [expr {$ORIGIN_X-0.16}] -start_high_ends_coordinate -
    start_high_ends_coordinate $FIN_RGT -extend_low_ends off -extend_high_ends off -ignore_parallel_targets
Create_power_straps -direction horizontal -start_at [expr {$ORIGIN_Y+0.34}] -num_placement_strap $NHFIX -increment_x_or_y $PITCH -nets $NODEL
    -layer M4 -width 0.1 -start_low_ends_coordinate -start_low_ends_coordinate [expr {$ORIGIN_X-0.16}] -start_high_ends_coordinate -
    start_high_ends_coordinate $FIN_RGT -extend_low_ends off -extend_high_ends off -ignore_parallel_targets
create_power_straps -direction horizontal -start_at [expr {$ORIGIN_Y+0.34}] -num_placement_strap $NHFIX -increment_x_or_y $PITCH -nets $NODEL
    -layer M6 -width 0.1 -start_low_ends_coordinate -start_low_ends_coordinate [expr {$ORIGIN_X-0.16}] -start_high_ends_coordinate -
    start_high_ends_coordinate $FIN_RGT -extend_low_ends off -extend_high_ends off -ignore_parallel_targets
```

Fig. 2.16 手动绘制 MOM 电容

最后一步是布置电源线。

```
derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
create_power_straps -direction vertical -start_at 0.200 -nets {VSS} -layer M2 -width 0.32 -start_low_ends_coordinate -start_high_ends
    coordinate -start_high_ends_coordinate 9.00 -extend_low_ends off -extend_high_ends off -look_inside_std_cells
create_power_straps -direction vertical -start_at 1.000 -nets {VDD} -layer M2 -width 0.32 -start_low_ends_coordinate -start_high_ends
    coordinate -start_high_ends_coordinate 9.00 -extend_low_ends off -extend_high_ends off -look_inside_std_cells
```

Fig. 2.17 布置电源线

版图绘制完成后，需要进行 LVS 检查，并输出版图的 GDS 文件。该步骤的 TCL 代码以及最终生成版图如图所示。

```
# Verify the placement and routing
save mw cel
verify_lvs

# Export the GDS-II file
set_write_stream_options -child_depth 0 -skip_ref_lib_cells \
    -map_layer $MAP_PATH
write_stream -format gds -cells $MODULE_NAME $GDS_PATH
write_verilog -no_tap_cells aprop/$MODULE_NAME$VERSION.lvs.v -pg -no_core_filler_cells
write_verilog -no_tap_cells aprop/$MODULE_NAME$VERSION.aprbhv.v -no_core_filler_cells -no_pg_pin_only_cells
close_mw_cel
```

Fig. 2.18 检查 LVS 并输出版图 GDS

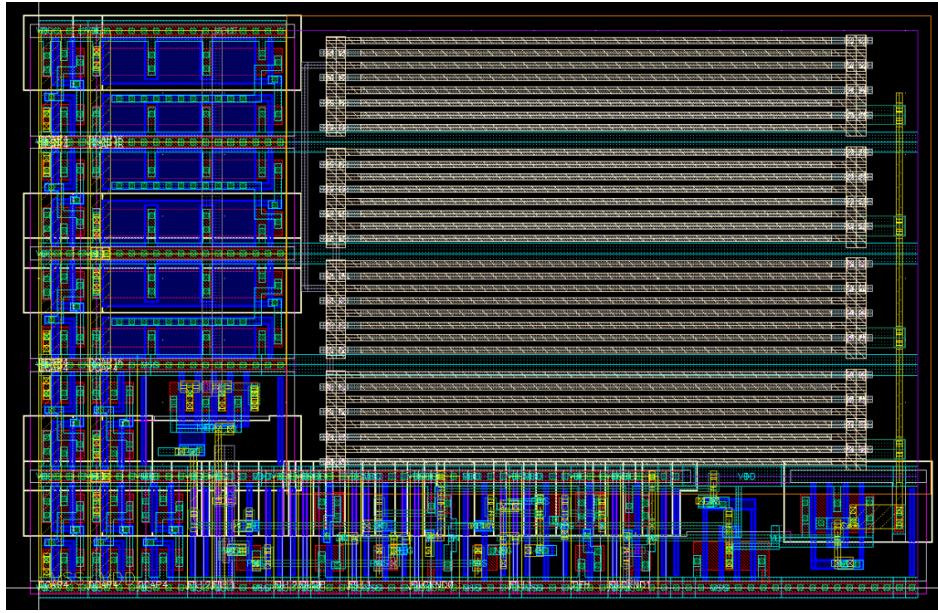


Fig. 2.19 采样开关版图

2.1.3 比较器

比较器结构如下图所示，它可以分为两部分——动态预放大器和锁存器。预放大器部分的差分结构单边是由两个 VCDC2 (voltage control discharge cell) 并联构成，当时钟信号为低电平时，放大器输出被预充电到高电平 VDD，时钟上升沿过程中，输出电压通过电压调制管放电，输入电压越大，放电速度越快，同时添加了 VCDP (voltage control discharge path) 用于加快放电速度。锁存器部分的单边结构类似于或非门，整体结构是由两个背靠背连接的 MSNR3 (mismatch enhanced NOR gate) 构成，通过比较放大器输出电压放电速度，产生比较结果。PCAP 用于校准比较器的 offset。

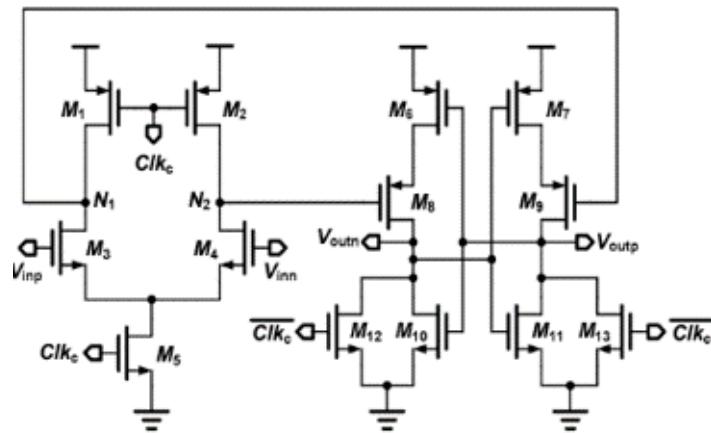


Fig. 2.20 比较器原理图

基于 RAIL65 库的比较器原理图如下图所示。

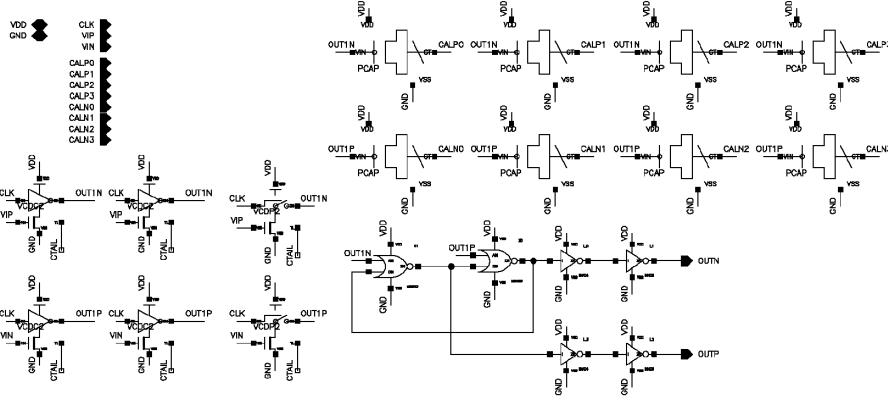


Fig. 2.21 基于 RAIL65 的比较器原理图

比较器的版图绘制流程和开关类似，主要分为四个步骤：设置工作路径并配置库文件、导入网表、布局布线、LVS 以及版图生成。值得注意得是，比较器作为一个差分的模拟电路，对电路的对称性要求非常严格，因此，电路中的连线全部需要手动绘制，避免工艺偏差以及不对称所带来的精度影响。部分连线代码和版图如下图所示。

```

create_power_straps -direction horizontal -start_at 0.810 -nets {VIP} -layer M3 -width 0.35 -start_low_ends_coordinate -
start_low_ends_coordinate 4.0 -start_high_ends_coordinate -start_high_ends_coordinate 9.0 -look_inside_std_cells
create_power_straps -direction vertical -start_at 5.40 -nets {VIP} -layer M2 -width 0.2 -start_low_ends_coordinate -start_high_ends
last_targets -look_inside_std_cells

create_power_straps -direction horizontal -start_at 0.810 -nets {VIN} -layer M3 -width 0.35 -start_low_ends_coordinate -
start_low_ends_coordinate 10.6 -start_high_ends_coordinate -start_high_ends_coordinate 15.6 -look_inside_std_cells
create_power_straps -direction vertical -start_at 14.2 -nets {VIN} -layer M2 -width 0.2 -start_low_ends_coordinate -start_high_ends
last_targets -look_inside_std_cells

```

Fig. 2.22 比较器部分布线代码

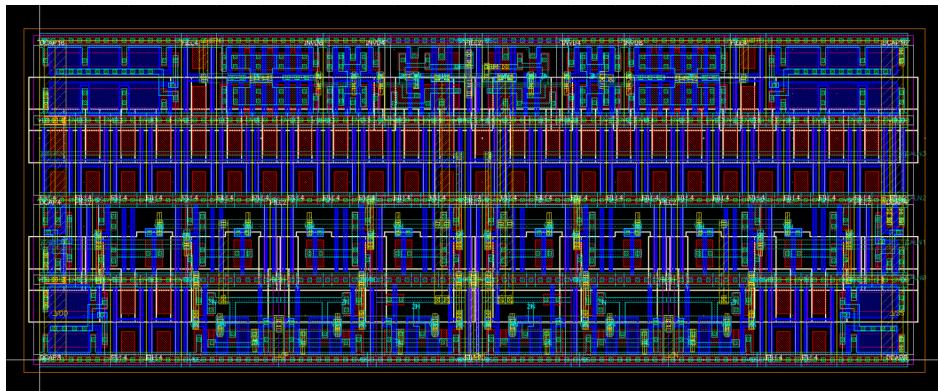


Fig. 2.23 比较器版图

2.1.4 DAC 阵列

电容阵列采用带有一位冗余的二进制结构，其值从 Cunit 到 256Cunit 递增，在 64Cunit 的位置处增加了一位冗余，且 128Cunit 和 256Cunit 的电容采用分立电容结构，用于降低共模电压的变化幅度。单位电容使用的是标准单元库的 MOM1 电容，其值约为 1fF。

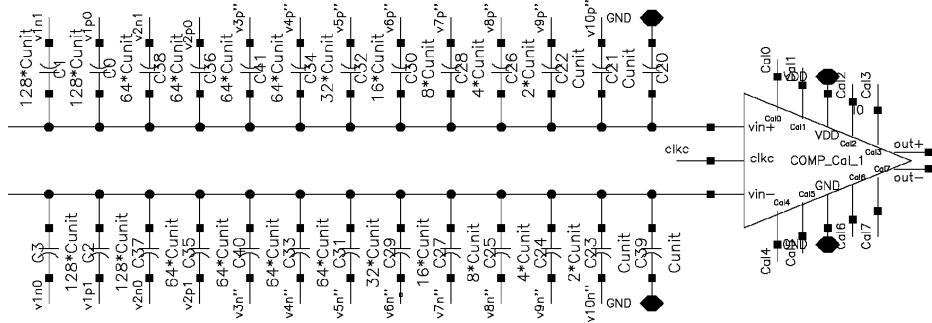


Fig. 2.24 电容阵列示意图

电容阵列的网表及部分连线代码以及最终版图如下图所示。单行电容由 16 个 MOM 电容排列，其值最高为 16Cunit，从 Cunit 到 256Cunit，一共需要 39 行电容排列，每行的电容由相应的反相器驱动，实现从参考电压 VREF 到 0 的切换。电容阵列的版图绘制流程也和开关类似：设置工作路径并配置库文件、导入网表、布局布线、LVS 以及版图生成。MOM 电容的版图如图所示，其由多层金属堆叠构成，容值主要为同层金属之间的寄生电容，有两个输入输出端口分别为 POS 端和 NEG 端，因此，在电容阵列的布线过程中，需要注意不同行、列电容的 POS 端、NEG 端的连接方式。

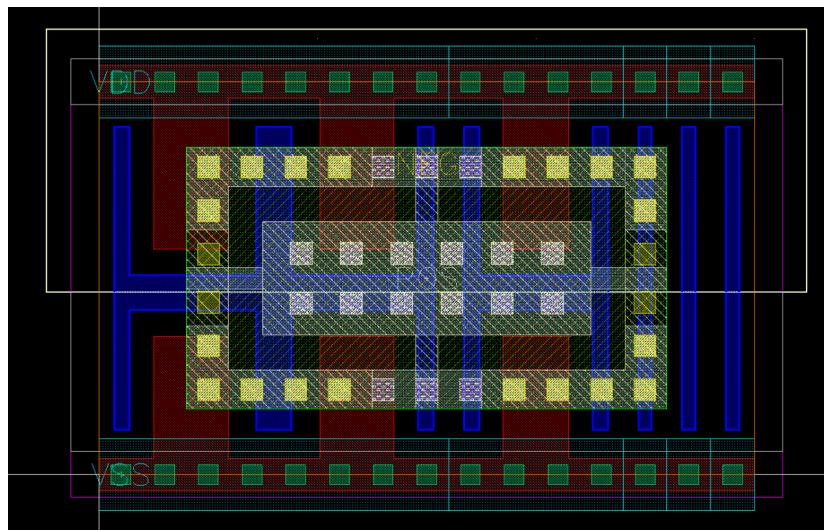


Fig. 2.25 单个 MOM 电容 (MOM1) 版图

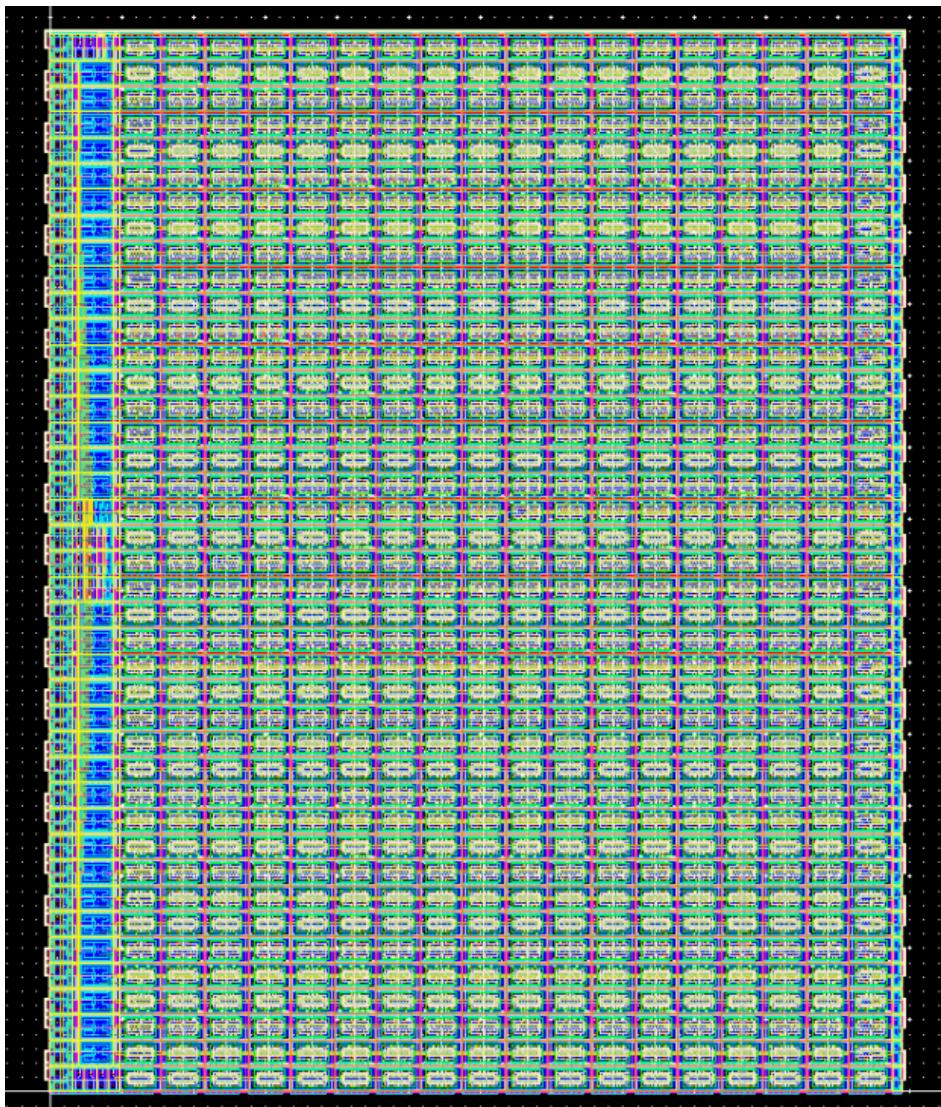


Fig. 2.26 电容阵列版图

```

module CDAC11b_H (
    input SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7, SW8, SW9, SW10, SW11,
    output CTOP
);
    SWCPDM row40 ();
    // cap 8
    SWCP08 row22 (.SW(SW3), .VTOP0(CTOP));
    // cap 4
    SWCP04 row19 (.SW(SW2), .VTOP0(CTOP));
    // cap 2
    SWCP02 row21 (.SW(SW1), .VTOP0(CTOP));
    // cap 1
    SWCP01 row20 (.SW(SW0), .VTOP0(CTOP));
    // cap 16
    SWCP16 row18 (.SW(SW4), .VTOP0(CTOP));
    // cap 32
    SWCP16 row17 (.SW(SW5), .VTOP0(CTOP));
    SWCP16 row23 (.SW(SW5), .VTOP0(CTOP));
    // cap 64
    SWCP16 row15 (.SW(SW6), .VTOP0(CTOP));
    SWCP16 row16 (.SW(SW6), .VTOP0(CTOP));
    SWCP16 row24 (.SW(SW6), .VTOP0(CTOP));
    SWCP16 row25 (.SW(SW6), .VTOP0(CTOP));
    // cap 128
    SWCP16_1 row11 (.SW(SW8), .VTOP0(CTOP));
    SWCP16_1 row12 (.SW(SW8), .VTOP0(CTOP));
    SWCP16_1 row28 (.SW(SW8), .VTOP0(CTOP));
    SWCP16_1 row29 (.SW(SW8), .VTOP0(CTOP));
    // cap 256
    SWCP16_1 row9 (.SW(SW9), .VTOP0(CTOP));
    SWCP16_1 row10 (.SW(SW9), .VTOP0(CTOP));
    SWCP16_1 row30 (.SW(SW9), .VTOP0(CTOP));
    SWCP16_1 row31 (.SW(SW9), .VTOP0(CTOP));
    // cap 512
    SWCP16_1 row5 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row6 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row7 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row8 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row32 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row33 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row34 (.SW(SW10), .VTOP0(CTOP));
    SWCP16_1 row35 (.SW(SW10), .VTOP0(CTOP));
    SWCPDL row0 (.VTOP1(CTOP));
endmodule

module SWCP10 [
    // inout VDD, VSS
    input SW;
    output VTOP0
]
    wire VBTM;
    wire FT00;
    INVDB8 DR08 (.I(SW), .ZN(VBTM));
    MOM1 r0c01 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c02 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c03 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c04 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c05 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c06 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c07 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r0c08 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c01 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c02 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c03 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c04 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c05 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c06 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c07 (.POS(VTOP0), .NEG(VBTM));
    MOM1 r1c08 (.POS(VTOP0), .NEG(VBTM));
    endmodule

```

Fig. 2.27 电容阵列布线

```

# cap 32
create_power_straps -direction horizontal -start_at 31.50 -nets {row17/VBTM} -layer M2 -width 0.2 -start_low_ends_coordinate -
start_low_ends_coordinate 4.60 -start_high_ends_coordinate -start_high_ends_coordinate 55.80
create_power_straps -direction horizontal -start_at 31.50 -nets {row17/XVSS} -layer M2 -width 0.2 -start_low_ends_coordinate -
start_low_ends_coordinate 57.30 -start_high_ends_coordinate -start_high_ends_coordinate 58.80

create_power_straps -direction horizontal -start_at 42.30 -nets {row23/VBTM} -layer M2 -width 0.2 -start_low_ends_coordinate -
start_low_ends_coordinate 4.60 -start_high_ends_coordinate -start_high_ends_coordinate 55.80
create_power_straps -direction horizontal -start_at 42.30 -nets {row23/XVSS} -layer M2 -width 0.2 -start_low_ends_coordinate -
start_low_ends_coordinate 57.30 -start_high_ends_coordinate -start_high_ends_coordinate 58.80

```

Fig. 2.28 电容阵列网表

2.1.5 压控延时单元 (VCDL)

压控延时单元的结构如图所示，其由 VCDC2 和 INVD1 的链式结构构成，输入电压越大，VCDC2 放电越快，则延时越小。下图为基于 RAIL65 库搭建的 VCDL。

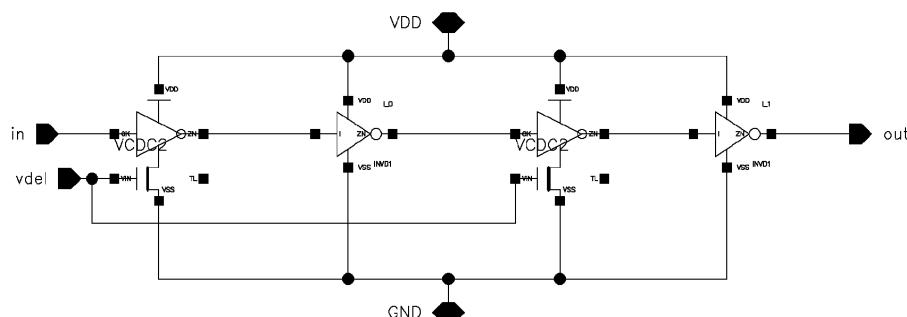


Fig. 2.29 基于 RAIL65 库的 VCDL 原理图

2.2 ADC 版图设计

2.2.1 ADC Core 部分版图

通过之前的对于 ICC 绘制版图的介绍，ADC core 的版图的绘制流程也可以分为四个步骤：设置工作路径并配置库文件、导入网表、布局布线、LVS 以及版图生成。布局布线是整个版图绘制流程中最重要的一环，通过分析 ADC core 的布局布线来介绍版图绘制流程。

布局布线流程首先需要对电容阵列、比较器、开关及时钟分频电路等模拟电路单元和数字逻辑控制单元进行布局规划。其次是对输入输出端口的位置布局；在完成了模拟电路单元的位置布局后，数字逻辑单元的布局是由 ICC 自动布局完成。器件布局完成后，由于使用了多组电源，因此需要对不同器件的电源电压进行划分，电容阵列的电源为参考电压源 VREFN 和 VREFP，比较器、开关及时钟分频电路的电源为模拟电压源 AVSS 和 AVDD，数字逻辑控制单元的电源为数字电压源 VDD 和 VSS，为保证电源的连接性，需要填充 DCAP 和 FILLER，填充完成后即可布置电源线。代码如图所示。

```
create_net -power AVDD
create_net -ground AVSS

derive_pg_connection -power_net AVDD -cells X0
derive_pg_connection -power_net AVDD -cells X1
derive_pg_connection -power_net AVDD -cells I0
derive_pg_connection -power_net AVDD -cells I_112
derive_pg_connection -power_net AVDD -cells I_113
derive_pg_connection -power_net AVDD -cells I_114
derive_pg_connection -power_net AVDD -cells I_115

derive_pg_connection -ground_net AVSS -cells X0
derive_pg_connection -ground_net AVSS -cells X1
derive_pg_connection -ground_net AVSS -cells I0
derive_pg_connection -ground_net AVSS -cells I_112
derive_pg_connection -ground_net AVSS -cells I_113
derive_pg_connection -ground_net AVSS -cells I_114
derive_pg_connection -ground_net AVSS -cells I_115

create_net -power VREFP
create_net -ground VREFN

derive_pg_connection -power_net VREFP -cells I_17
derive_pg_connection -power_net VREFP -cells I_18
derive_pg_connection -ground_net VREFN -cells I_17
derive_pg_connection -ground_net VREFN -cells I_18

derive_pg_connection -power_net VDD -power_pin VDD -ground_net VSS -ground_pin VSS
```

Fig. 2.30 ADC 电源线布置

参考电压源上需要并联去耦电容，去耦电容采用 DCAP 加 MOM 电容的形式，在 DCAP 上以 MOM 电容的形式依次添加不同层金属，以较小的面积实现较大的容值。本设计中的去耦电容容值约为 20pF。

```

create_power_straps -direction vertical -start_at 3.4 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFN} -layer M2 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off -look_inside_std_cells
create_power_straps -direction vertical -start_at 3.4 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFN} -layer M4 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off
create_power_straps -direction vertical -start_at 3.4 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFN} -layer M6 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off
create_power_straps -direction vertical -start_at 9.8 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFP} -layer M2 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off -look_inside_std_cells
create_power_straps -direction vertical -start_at 9.8 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFP} -layer M4 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off
create_power_straps -direction vertical -start_at 9.8 -num_placement_strap 2 -increment_x_or_y 12.8 -nets {VREFP} -layer M6 -width 4 -
start_low_ends_coordinate -start_low_ends_coordinate 0.20 -start_high_ends_coordinate -start_high_ends_coordinate 74.00 -extend_low_ends off -
extend_high_ends off

```

Fig. 2.31 去耦电容版图代码

布置完成电源线后，所有模拟器件之间的连接线作为模拟线需要手动绘制，数字连接线由自动布线完成。最终生成的 ADC Core 版图如下图所示。

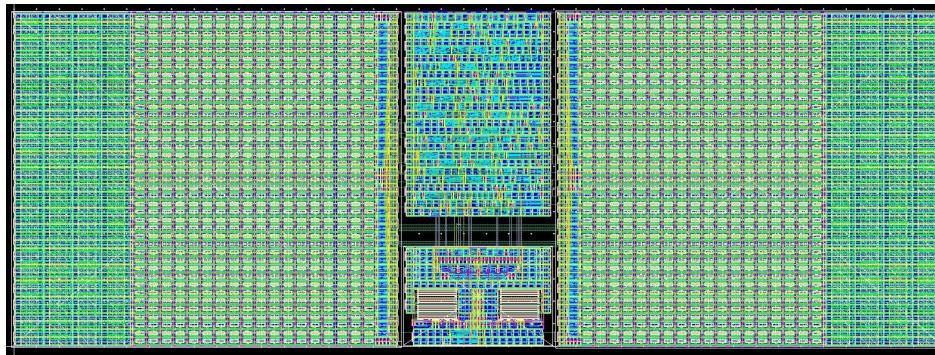


Fig. 2.32 ADC Core 部分版图

2.2.2 ADC Full Chip 版图

ADC Full Chip 的版图绘制主要是完成 PAD 的布局以及 PAD 与 ADC core 之间连接的布线。

绘制版图时首先要设置工作路径并配置库文件。相比较前文采样开关中所述的库文件配置，Full Chip 版图中使用了 PAD，因此需要相应的标准单元库文件 tphn65lpgv2od3_sl 和 tpbn65v。之后导入网表。创建并打开 library，读取相应的网表文件。配置 PAD 库文件的代码如下图所示。

```

set REFLIB_PATH {/EDA/PDK/tsmcN65/TSMCHOME/digital/Back_End/milkyway/tcbn65lp_200a/cell_frame/tcbn65lp \
/home/ninezhang/workspace/MKW/MKW_RAIL/CEL_FRAM/rail65 \
/EDA/PDK/tsmcN65/TSMCHOME/digital/Back_End/milkyway/tphn65lpgv2od3_sl_140a_mt_29lm/cell_frame/tphn65lpgv2od3_sl \
/EDA/PDK/tsmcN65/TSMCHOME/digital/Back_End/milkyway/tpbn65v_200a/cup/9m_6X1Z1U/cell_frame/tpbn65v}

```

Fig. 2.33 为 PAD 配置库文件

最后进行布局布线。PAD 主要分为模拟电源 PAD、2.5V 数字电源 PAD、1.2V 数字电源 PAD、模拟输入 PAD、数字信号输入 PAD 以及数字信号输出 PAD，除此之外还有模数隔离 PAD 和 CORNER。首先需要对众多 PAD 进行布局规划，注意在数字和模拟 PAD 之间需要添加数模隔离 PAD。芯片四周由 CORNER 构成，PAD 上需要依次添加不同方向的 bond 实现和外部的连接。PAD 的布局代码如下图所示。

```

set_undoable_attribute [get_cells -all I_8[1]] origin {200 506}
set_undoable_attribute [get_cells -all I_8[1]] orientation {S}
set_undoable_attribute [get_cells -all I_8[1]] is_fixed {1}
set_port_location PAD_B[0] -coordinate {0 0} -layer_name M7 -layer_area {176.5 419.185 198.5 423.185}

set_undoable_attribute [get_cells -all I_8[2]] origin {230 506}
set_undoable_attribute [get_cells -all I_8[2]] orientation {S}
set_undoable_attribute [get_cells -all I_8[2]] is_fixed {1}
set_port_location PAD_B[1] -coordinate {0 0} -layer_name M7 -layer_area {206.5 419.185 228.5 423.185}

create_cell {corner1 corner2} PCORNER_G
create_cell {corner3 corner4} PCORNERA_G
set_undoable_attribute [get_cells -all corner3] origin {0 0}
set_undoable_attribute [get_cells -all corner3] orientation {N}
set_undoable_attribute [get_cells -all corner3] is_fixed {1}
set_undoable_attribute [get_cells -all corner1] origin {0 506}
set_undoable_attribute [get_cells -all corner1] orientation {E}
set_undoable_attribute [get_cells -all corner1] is_fixed {1}
set_undoable_attribute [get_cells -all corner2] origin {680 506}
set_undoable_attribute [get_cells -all corner2] orientation {S}
set_undoable_attribute [get_cells -all corner2] is_fixed {1}
set_undoable_attribute [get_cells -all corner4] origin {680 0}
set_undoable_attribute [get_cells -all corner4] orientation {W}
set_undoable_attribute [get_cells -all corner4] is_fixed {1}

set_undoable_attribute [get_cells -all bond_b0] origin {200 506}
set_undoable_attribute [get_cells -all bond_b0] orientation {S}
set_undoable_attribute [get_cells -all bond_b0] is_fixed {1}

set_undoable_attribute [get_cells -all bond_b1] origin {230 506}
set_undoable_attribute [get_cells -all bond_b1] orientation {S}
set_undoable_attribute [get_cells -all bond_b1] is_fixed {1}

```

Fig. 2.34 PAD 布局代码

为了通过 Full Chip 的 DRC 并在一定程度上降低 ESD 的危害，需要在所有的模拟线上串联了一个电阻，这个电阻由 RAIL65 库里的 PRES 和 PRES2 构成，它们的版图如下图所示，通过在两边放置 PRES，中间放置 PRES2，即可得到一个具有一定阻值的 poly 电阻。为了使信号（或电源电流）不会衰减太多，需要这个电阻阻值尽可能小，电阻通过并联可以就可以得到任意小的电阻。

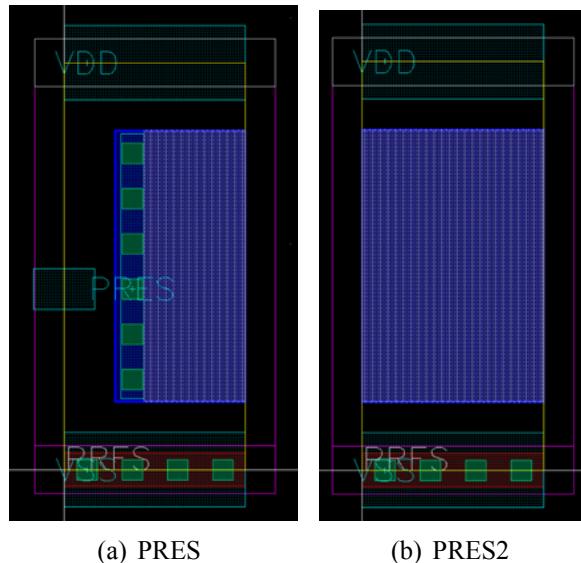


Fig. 2.35 PRES 和 PRES2 版图

布线需要考虑的重要因素是线阻大小，参考电压源、模拟电压源及数字电压源对于线阻有严格要求，需要按照要求绘制 PAD 和 core 之间的电源线，参考电压源通过使用多层金属布线来降低线阻，数字电压源通过使用电源环来实现供电以及降低线阻。在电源线的绘制过程中使用多组布线来减小趋肤效应带来的影响。部分数字线使用自动布线完成。版图绘制完成后，需要进行 LVS 及 DRC 的检查，在确保 LVS、DRC 及芯片的功能和性能指标无问题后即可提交整个芯片的 GDS 文件，整个芯片的设计完成。下图为整个 ADC 芯片（Full Chip）的版图。

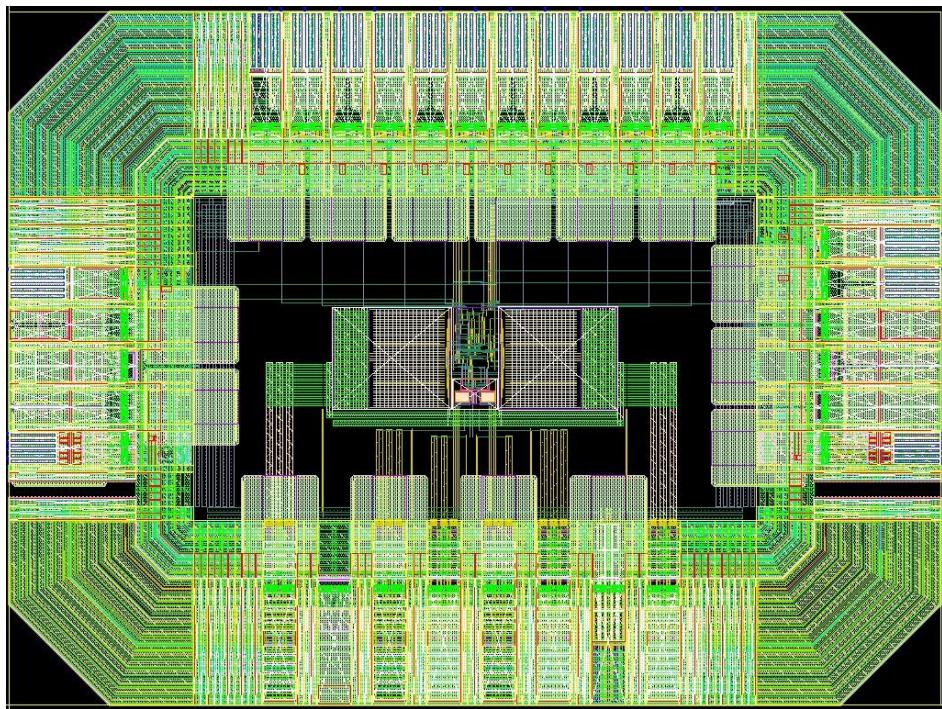
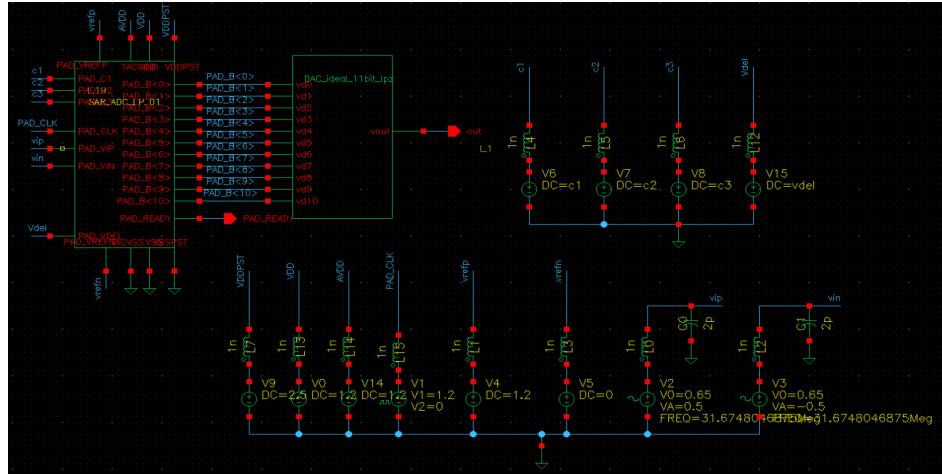


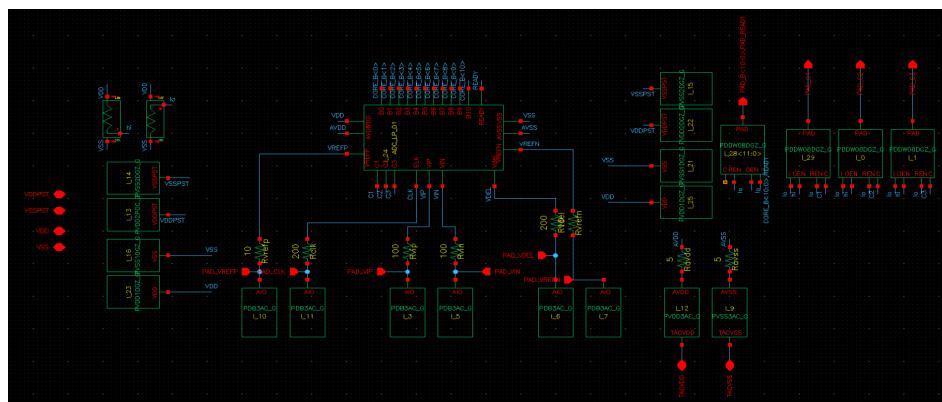
Fig. 2.36 ADC Full Chip 版图

3 ADC 后仿真结果

在完成 ADC Full Chip 的设计后，对电路进行最后阶段的仿真，仿真的 testbench 如下图。电源后接的电感用来模拟 bonding wire 的电感在 PAD 与 ADC Core 间添加的电阻为绘制完版图后计算得到的金属线阻。



(a) ADC Testbench (信号源)



(b) ADC Testbench (PAD 与 ADC Core)

Fig. 3.37 ADC Testbench

仿真结果如下图所示。设置 ADC 的采样频率为 65MHz, 给 ADC 一个(65M*499/1024) Hz 的正弦差分输入信号, 并对输出做 FFT, 可得 ADC 的频谱等 ADC 输出的时域波形、频谱及 ENOB 等如下图所示。(注意由于芯片数字信号 PAD 的电源电压是 2.5V)

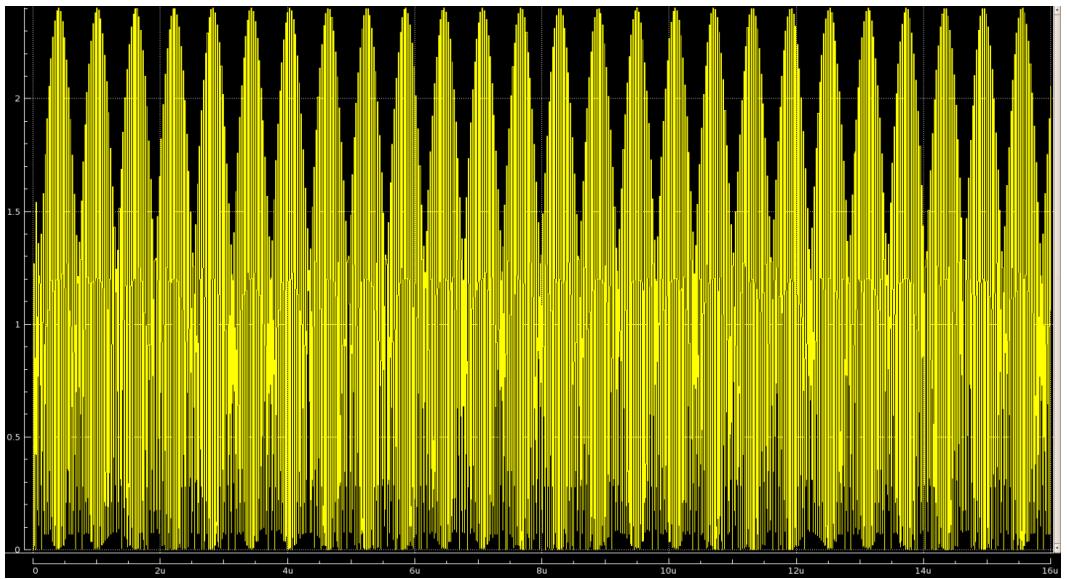


Fig. 3.38 ADC 输出信号时域波形

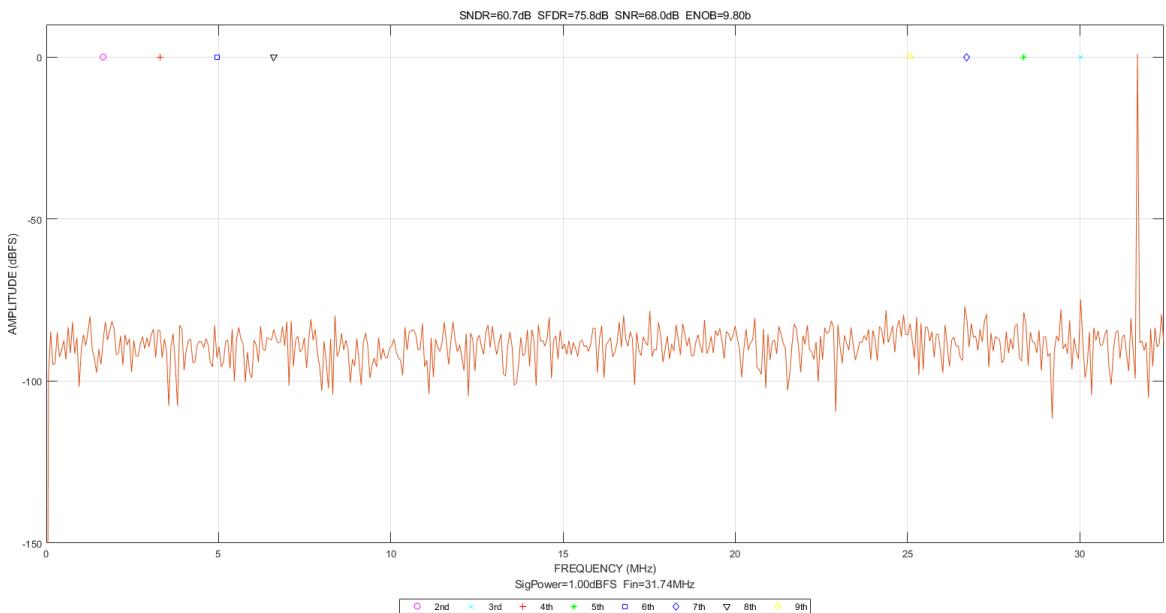


Fig. 3.39 ADC 输出信号频谱

下图为 ADC 的功耗分类，VDD 表示数字电路功耗，AVDD 表示模拟电路功耗，VREF 表示参考电压功耗。

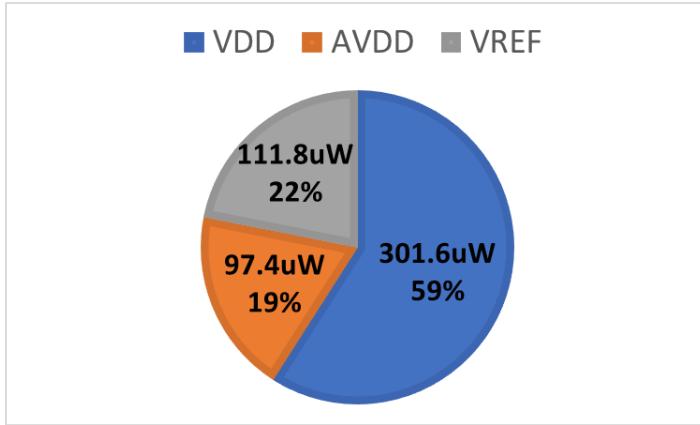


Fig. 3.40 ADC 功耗

ADC 采用 TSMC 65nm LP 工艺设计，输入电压范围理想值为 0-1.2V，由于非理想因素在 0.05V-1.15V。在 65MSPS 的采样频率下 ENOB 为 9.8bit，SFDR 为 75.8dB，总功耗为 511uW，Walden FoM 为 8.82fJ/Conv.-step，Schreier FoM 为 168.79dB。

Parameter	Specification
Processing	65nm LP
Sampling Frequency	65MSPS
ENOB/SNDR	9.8bit/60.7dB
SFDR	75.8dB
Power	511uW
FoM Walden	8.82fJ/Conv.-step
FoM Schreier	168.79dB

Table 3.1 ADC 性能参数

References

- [1] C.-C. Liu, S.-J. Chang, G.-Y. Huang, and Y.-Z. Lin, “A 10-bit 50-ms/s sar adc with a monotonic capacitor switching procedure,” *IEEE Journal of Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, 2010.