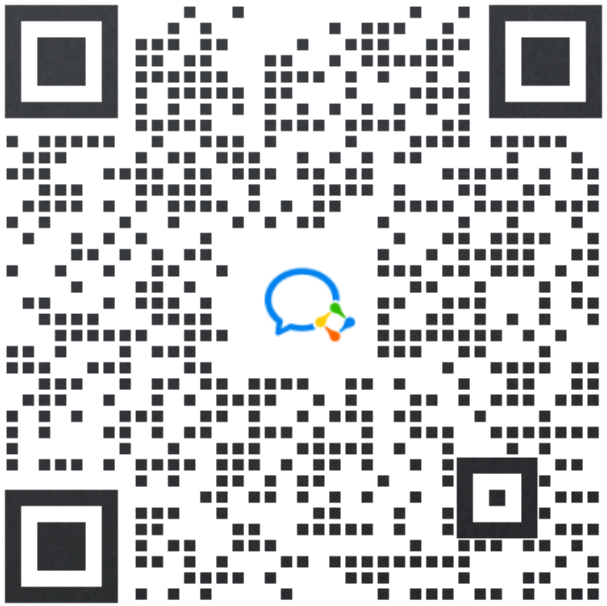
iEDA-Tutorial

iEDA开源交流群

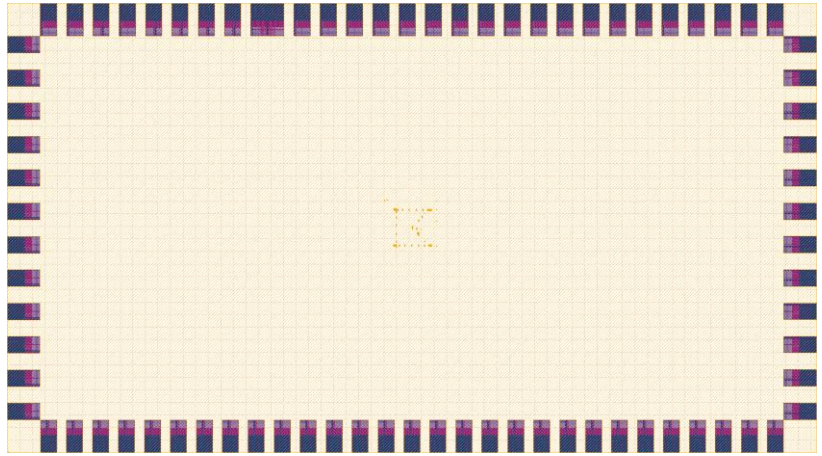


iEDA课题组

2023/7/31

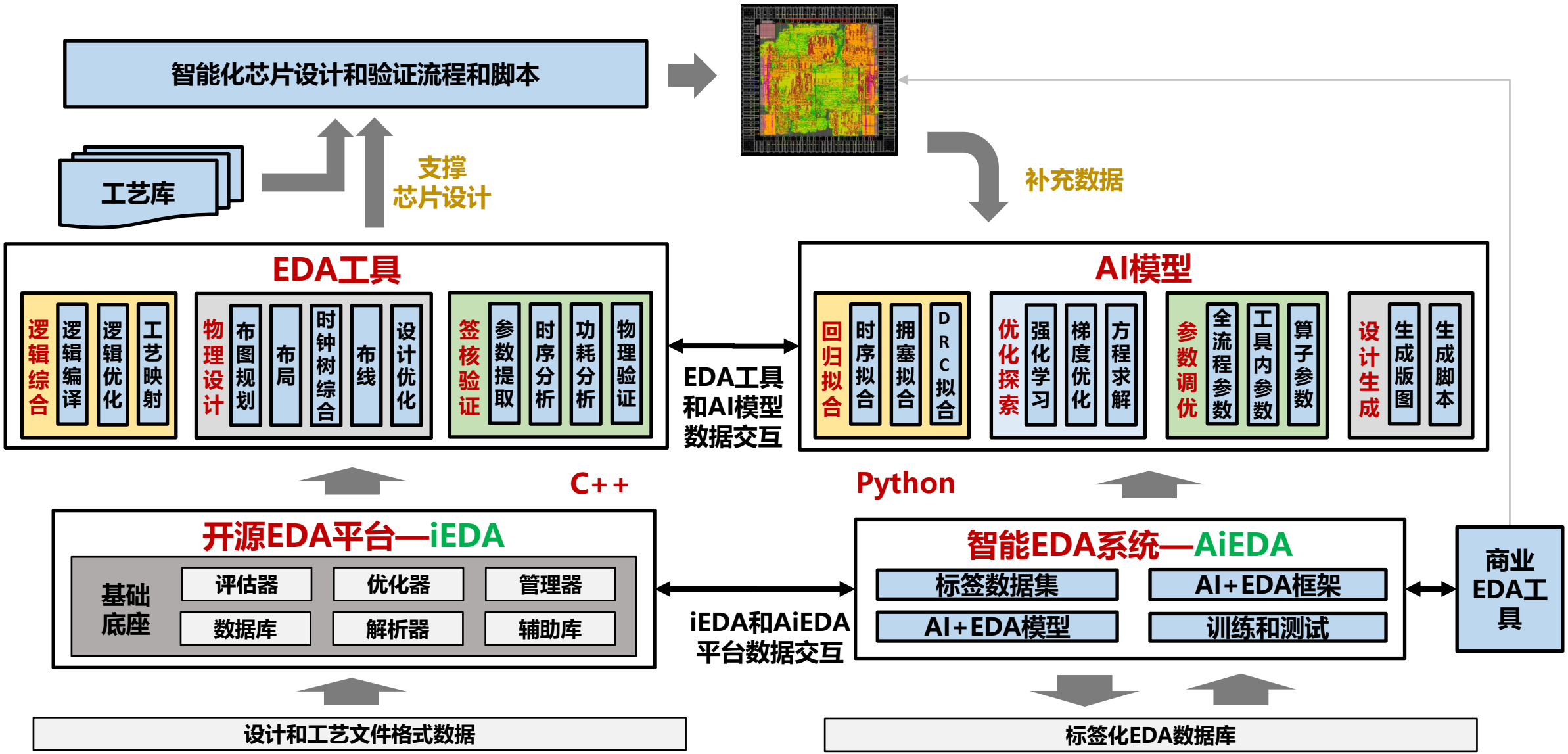
关于iEDA

- 关于 **iEDA** 中的 “**i**”
  - **含义 1: Infrastructure**
  - **含义 2: Intelligent**
- **目标**
  - **构建开源数字EDA基础设施**
  - **探索新的、有价值的EDA研发方法论和技术**
  - **实现高质量和高性能的EDA工具**
- **开源链接：**
  - **Gitee: https://gitee.com/oscc-project/iEDA**
  - **GitHub: https://github.com/OSCC-Project/iEDA**
  - **打开Gitee或GitHub搜索 OSCC/iEDA 即可**

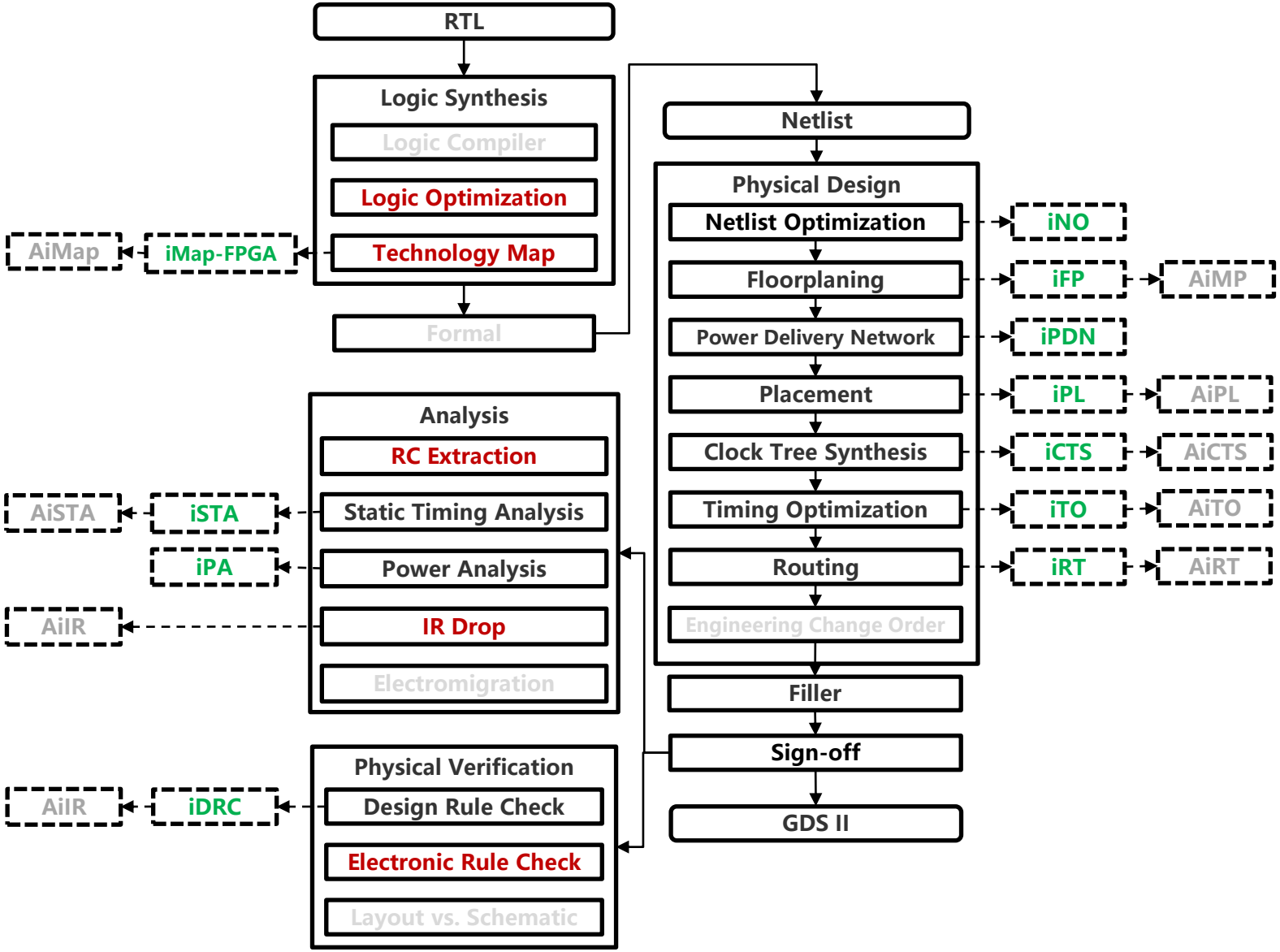


开源不是目的而是方式

iEDA和AiEDA


智能化芯片设计和验证流程和脚本
工艺库
支撑
芯片设计
补充数据
EDA工具
逻辑综合
逻辑编译
逻辑优化
工艺映射
物理设计
布图规划
布局
时钟树综合
布线
设计优化
签核验证
参数提取
时序分析
功耗分析
物理验证
EDA工具
和AI模型
数据交互
AI模型
回归拟合
时序拟合
拥塞拟合
DRC拟合
优化探索
强化学习
梯度优化
方程求解
参数调优
全流程参数
工具内参数
算子参数
设计生成
生成版图
生成脚本
C++
Python
开源EDA平台—iEDA
基础底座
评估器
优化器
管理器
数据库
解析器
辅助库
iEDA和AiEDA
平台数据交互
智能EDA系统—AiEDA
标签数据集
AI+EDA框架
AI+EDA模型
训练和测试
商业EDA工具
设计和工艺文件格式数据
标签化EDA数据库

iEDA工具和AI模型



iEDA Tutorial

- 第一期（2023年6月）：时序(iSTA)和功耗(iPW)分析工具及技术
- **第二期（2023年7月）：布图(iFP)/布局(iPL)和时序优化(iTO)工具及技术**
- 第三期（2023年8月）：iEDA基础底座和AiEDA框架系统介绍
- 第四期（2023年9月）：时钟树综合(iCTS)和布线(iRT)工具及技术
- 第五期（2023年10月）：工艺映射(iMap)和验证工具及技术介绍
- 第六期（2023年11月）：AiEDA模型研究进展介绍

iEDA-Tutorial 第二期

布图(iFP)、布局(iPL)和时序优化(iTO)工具介绍

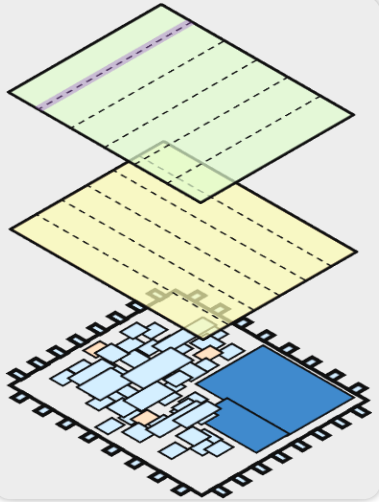
黄增荣、陈仕健、邱奕杭、黄富兴、吴鸿熙

2023/7/31

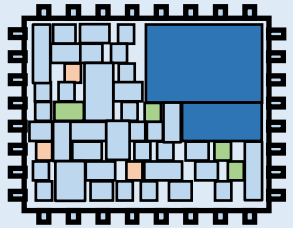
设计过程——物理设计

- System Specifications
- Architectural Design
- Functional Design
- Logic Synthesis
- Tech Mapping
- **Physical Design**
- Physical Verification
- Sign Off
- Layout Processing
- Fabrication
- Package and Test

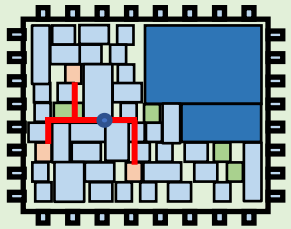
Netlist



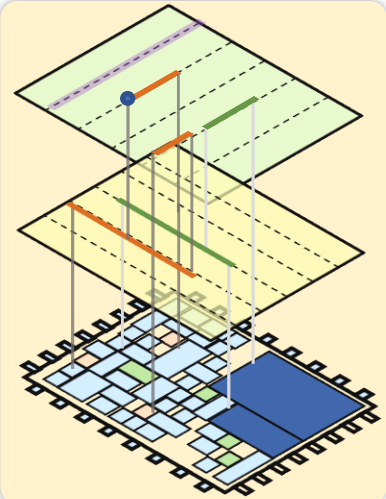
Floorplan：确定Die和Core大小，确定IO位置，规划Macro的摆放，设计电源线，放置物理单元。



Placement：摆放标准单元至合法的位置，插入填充单元，进行时序优化，使得设计目标最优。

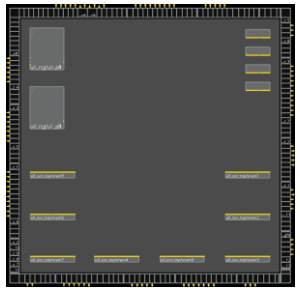


CTS：设计时钟网络，连接每个FF的时钟pin。实现时钟同步，功耗低等目标。

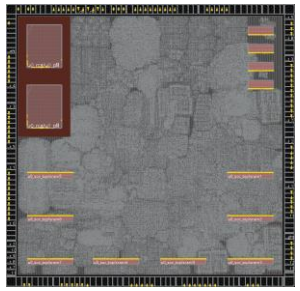


Routing：将连接单元pin的Net在规定的金属层的Track上实体化，使得线长短，时序功耗低，设计规则满足。

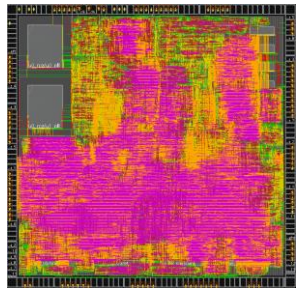
GDS II



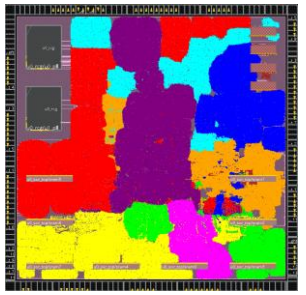
布图规划



布局

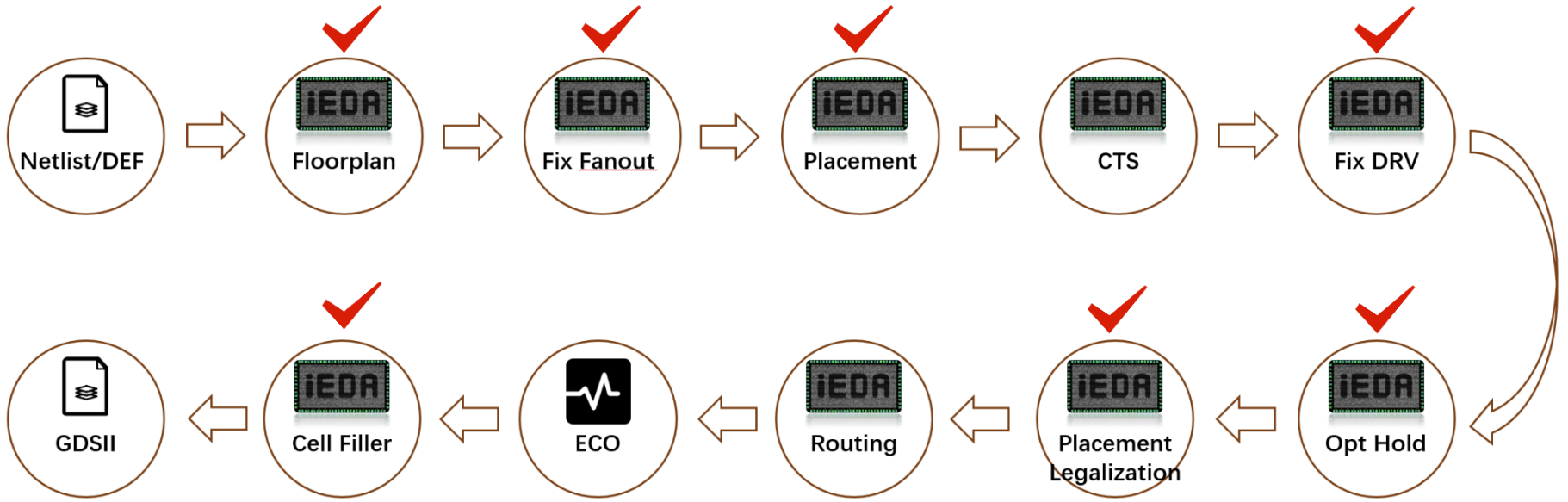


布线



版图

会议主题


Netlist/DEF
Floorplan
Fix Fanout
Placement
CTS
Fix DRV
Opt Hold
Placement Legalization
Routing
ECO
Cell Filler
GDSII


开源EDA原型系统环境
工艺库
设计文件
服务器
开源EDA原型系统
辅助工具

iEDA Tutorial 第二期议程

- **Part 1 iEDA-iFP/iPDN工具架构、特性与使用（黄增荣）**

- Part 2 iEDA-iMP 关键技术（黄富兴）

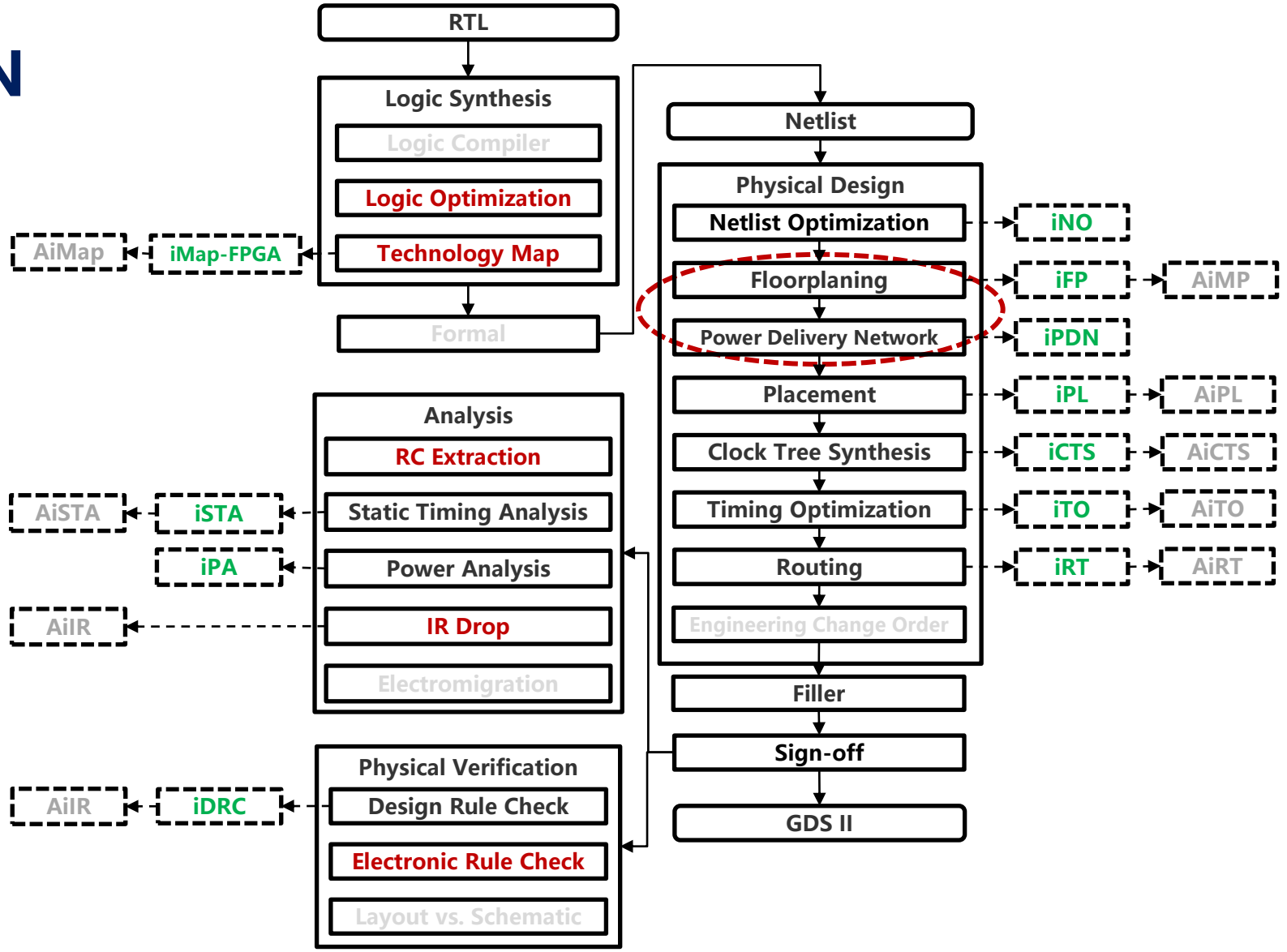
- Part 3 iEDA-iPL 问题介绍、架构、使用与规划 （陈仕健）

- Part 4 iEDA-iPL 关键技术 （邱奕杭）

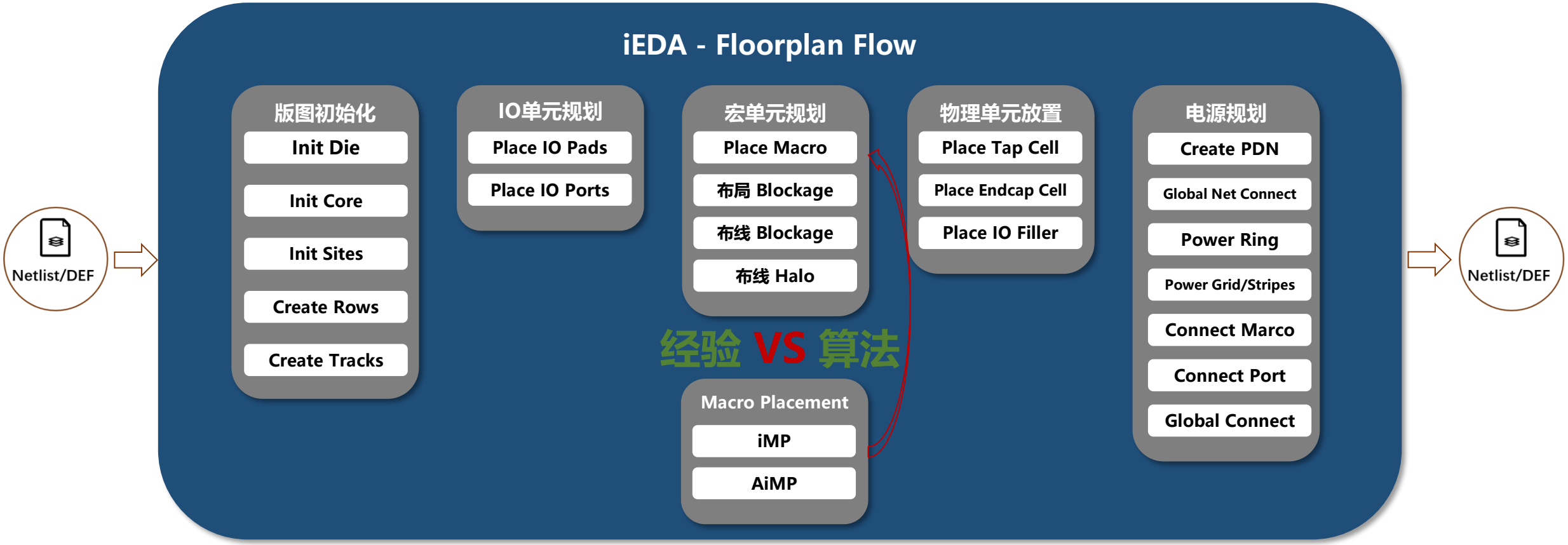
- Part 5 iEDA-iTO工具架构、特性与关键技术 （吴鸿熙）

iEDA - Floorplan概述

- **iEDA – iFP/iPDN**
  - 目标
    - 确定芯片面积
    - 确保时序收敛
    - 保证芯片稳定
    - 满足布线要求
  - 内容
    - 版图初始化
    - I/O单元规划
    - 宏单元规划
    - 物理单元放置
    - 电源规划



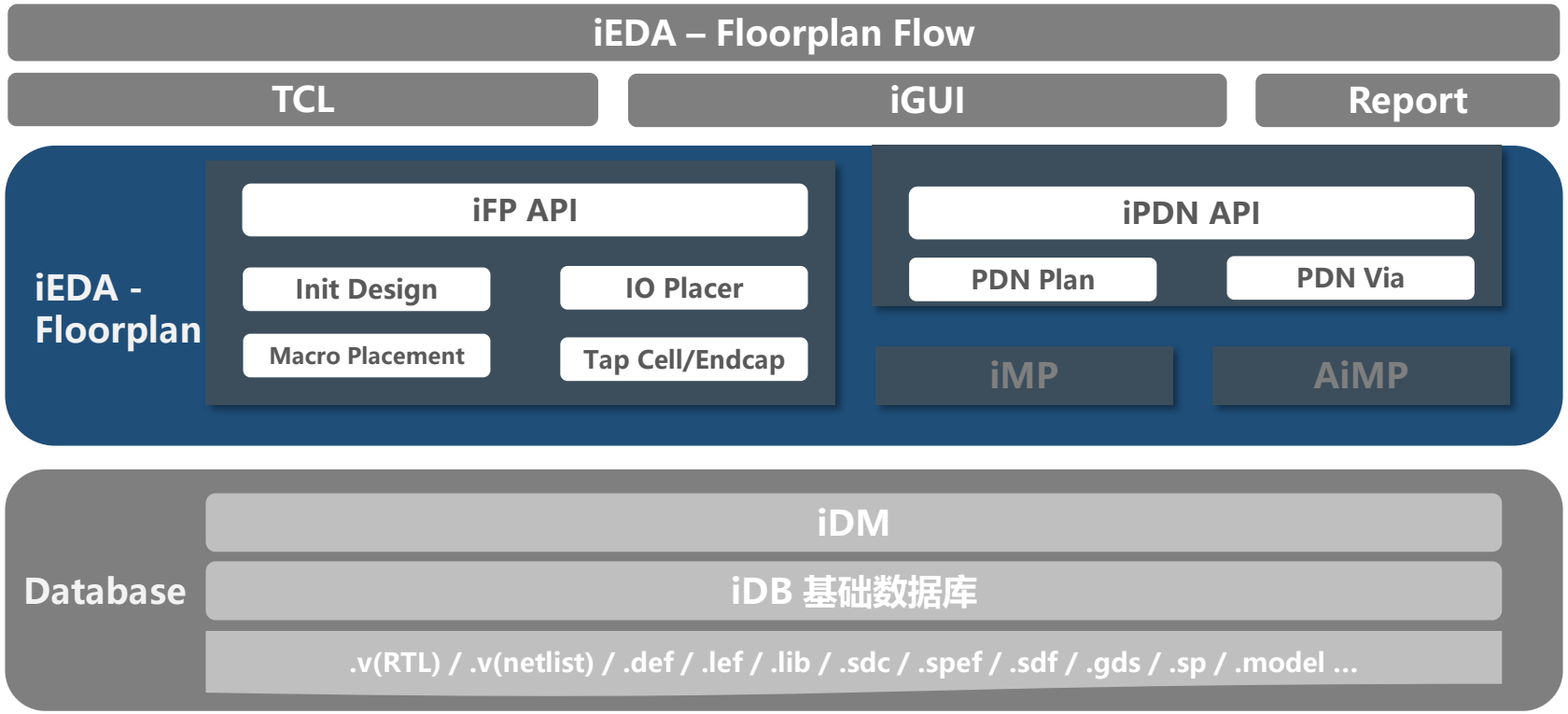
iEDA – Floorplan 整体流程



iFP / iPDN 软件框架图

iFP – 代码结构

```
.
├── api
│   ├── CMakeLists.txt
│   ├── ifp_api.cpp
│   └── ifp_api.h
├── CMakeLists.txt
├── source
│   ├── CMakeLists.txt
│   ├── config
│   │   └── CMakeLists.txt
│   ├── data
│   │   ├── CMakeLists.txt
│   │   ├── ifp_enum.cpp
│   │   ├── ifp_enum.h
│   │   ├── ifp_interval.cpp
│   │   └── ifp_interval.h
│   ├── module
│   │   ├── CMakeLists.txt
│   │   ├── endcap_cell
│   │   │   └── CMakeLists.txt
│   │   ├── init_design
│   │   │   ├── CMakeLists.txt
│   │   │   ├── init_design.cpp
│   │   │   └── init_design.h
│   │   ├── io_placer
│   │   │   ├── CMakeLists.txt
│   │   │   ├── io_placer.cpp
│   │   │   └── io_placer.h
│   │   ├── io_router
│   │   │   └── CMakeLists.txt
│   │   ├── macro_placer
│   │   │   └── CMakeLists.txt
│   │   └── tap_cell
│   │       ├── CMakeLists.txt
│   │       ├── tapcell.cpp
│   │       └── tapcell.h
│   └── utility
│       └── CMakeLists.txt
└── test
    └── CMakeLists.txt
```



iPDN – 代码结构

```
.
├── api
│   ├── CMakeLists.txt
│   ├── ipdn_api.cpp
│   └── ipdn_api.h
├── CMakeLists.txt
├── source
│   ├── CMakeLists.txt
│   ├── config
│   │   └── CMakeLists.txt
│   ├── data
│   │   ├── CMakeLists.txt
│   │   ├── ipdn_basic.cpp
│   │   ├── ipdn_basic.h
│   │   ├── ipdn_enum.cpp
│   │   └── ipdn_enum.h
│   ├── module
│   │   ├── CMakeLists.txt
│   │   ├── pdn_plan
│   │   │   ├── CMakeLists.txt
│   │   │   ├── pdn_cut_stripe.cpp
│   │   │   ├── pdn_cut_stripe.h
│   │   │   ├── pdn_plan.cpp
│   │   │   ├── pdn_plan.h
│   │   │   └── pdn_plan_macro.cpp
│   │   ├── pdn_router
│   │   │   └── CMakeLists.txt
│   │   ├── pdn_sim
│   │   │   └── CMakeLists.txt
│   │   └── pdn_via
│   │       ├── CMakeLists.txt
│   │       ├── pdn_via.cpp
│   │       └── pdn_via.h
│   ├── solver
│   │   └── CMakeLists.txt
│   └── utility
│       └── CMakeLists.txt
└── test
    └── CMakeLists.txt
```

功能：版图初始化

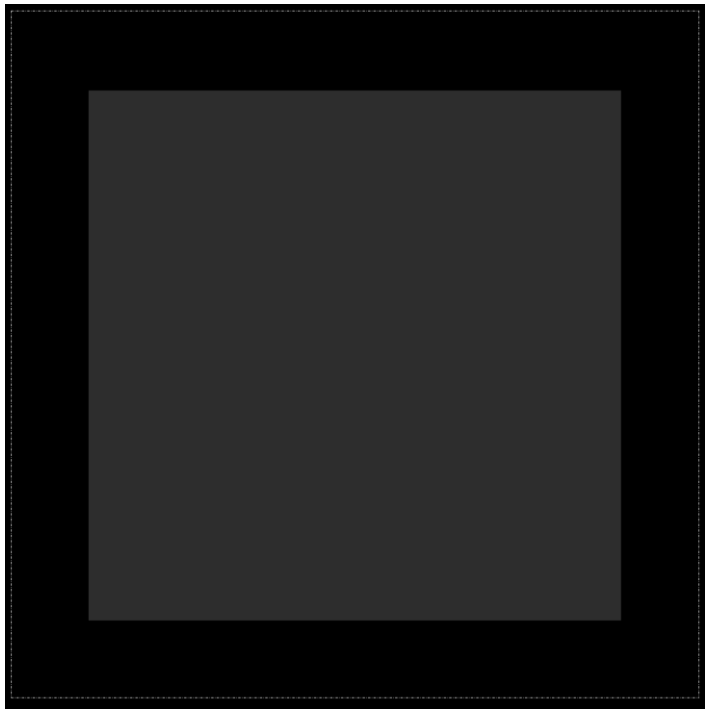
- 版图初始化
  - 加载设计相关文件
  - 确定Sites（IO Site、Core Site、Corner Site）
  - 确定面积、利用率
  - 创建Row
  - 创建布线轨道

- 脚本

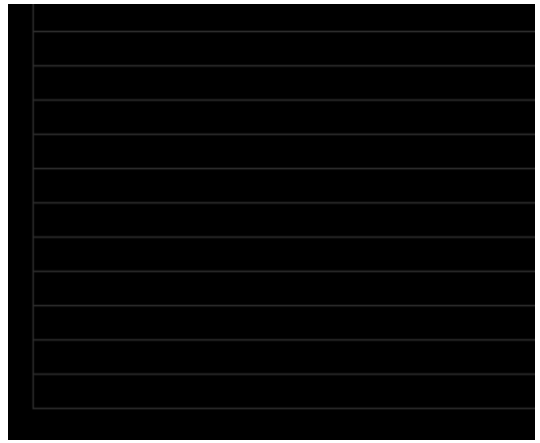
```
#============================================================
##   init floorplan
#============================================================
set DIE_AREA "0.0    0.0   1499.96   1500"
set CORE_AREA "169.96 170.0 1330 1325.6"
set PLACE_SITE core
set IO_SITE pad
set CORNER_SITE corner

init_floorplan \
   -die_area $DIE_AREA \
   -core_area $CORE_AREA \
   -core_site $PLACE_SITE \
   -io_site $IO_SITE \
   -corner_site $CORNER_SITE

source ./script/iFP_script/module/gern_tracks.tcl
```

```
gern_track -layer M1 -x_start 140 -x_step 280 -y_start 200 -y_step 200
gern_track -layer M2 -x_start 120 -x_step 200 -y_start 200 -y_step 200
gern_track -layer M3 -x_start 120 -x_step 200 -y_start 200 -y_step 200
gern_track -layer M4 -x_start 120 -x_step 200 -y_start 200 -y_step 200
```

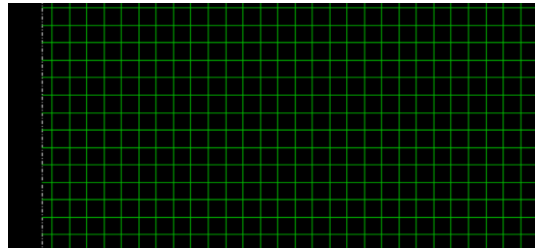
- 版图



版图初始化



Rows



Tracks（Prefer& Non-Prefer）

- 输出数据

```
DIEAREA ( 0 0 ) ( 2999920 3000000 ) ;
```

```
ROW CORE_ROW_0 core 668080 2649400 N DO 7114 BY 1 STEP 280 0
 ;
ROW CORE_ROW_1 core 668080 2647600 FS DO 7114 BY 1 STEP 280 0
 ;
ROW CORE_ROW_2 core 668080 2645800 N DO 7114 BY 1 STEP 280 0
 ;
ROW CORE_ROW_3 core 668080 2644000 FS DO 7114 BY 1 STEP 280 0
```

```
TRACKS X 6920 DO 333 STEP 9000 LAYER AP ;
TRACKS Y 7000 DO 333 STEP 9000 LAYER AP ;
TRACKS Y 800 DO 1875 STEP 1600 LAYER M9 ;
TRACKS X 2320 DO 1874 STEP 1600 LAYER M9 ;
TRACKS X 2320 DO 1874 STEP 1600 LAYER M8 ;
```

功能：IO单元规划

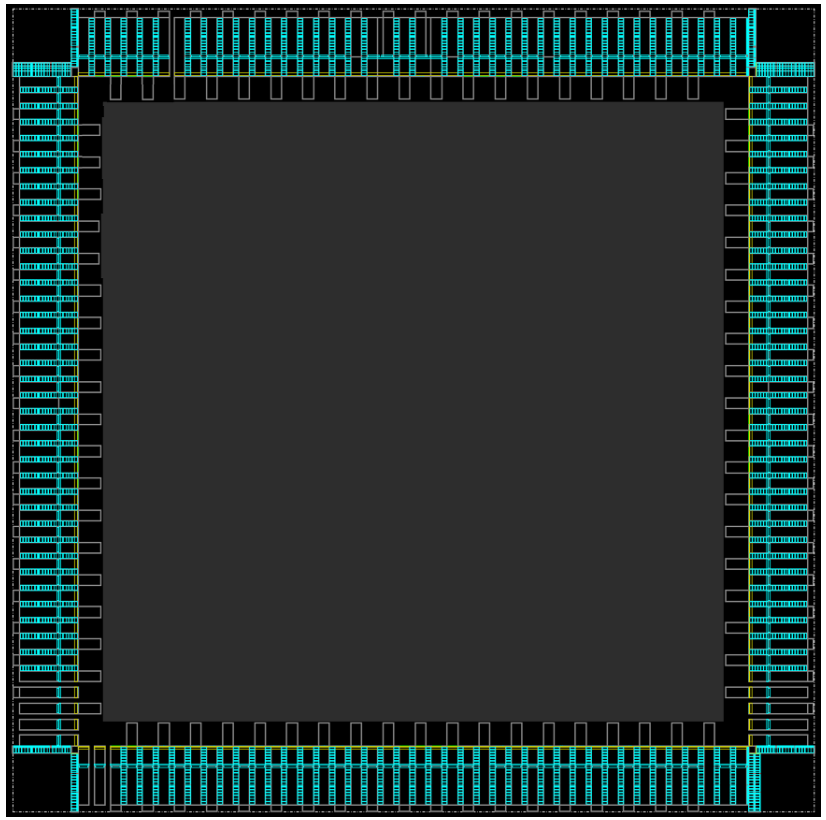
- IO单元规划
  - Pad Placement
  - Port Placement（选用）

- 脚本

```
place_instance -inst_name u0_spi_cs_1_PAD                    -llx 2586000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_spi_clk_PAD                     -llx 2466000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_core_dip2_PAD                   -llx 2346000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_core_dip0_PAD                   -llx 2226000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_rst_PAD                         -llx 2106000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vddio_top_PAD                      -llx 1986000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vdd_dummy_top_PAD                  -llx 1866000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vss_dummy_top_PAD                  -llx 1746000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vddio_top_2_PAD                    -llx 1626000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vddio_top_1_PAD                    -llx 1266000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_clk_cfg_5_PAD                   -llx 1146000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_clk_cfg_3_PAD                   -llx 1026000   -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_clk_cfg_1_PAD                   -llx 906000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_pll_cfg_0_PAD                   -llx 786000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_pll_cfg_2_PAD                   -llx 666000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vssio_top_2_PAD                    -llx 546000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vdd_dummy_top_1_PAD                -llx 426000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name vss_dummy_top_1_PAD                -llx 306000    -lly 2819900   -orient R180  -cellmaster PAD60GU
place_instance -inst_name u0_core_dip1_PAD                   -llx 2286000   -lly 2661320   -orient R180  -cellmaster PAD60NU
place_instance -inst_name u0_core_dip3_PAD                   -llx 2406000   -lly 2661320   -orient R180  -cellmaster PAD60NU
place_instance -inst_name u0_spi_cs_0_PAD                    -llx 2526000   -lly 2661320   -orient R180  -cellmaster PAD60NU
place_instance -inst_name u0_clk4div_out_PAD                 -llx 2166000   -lly 2661320   -orient R180  -cellmaster PAD60NU
```

```
place_port -pin_name osc_25m_in_pad -offset_x 69500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name osc_25m_out_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name osc_100m_in_pad -offset_x 69500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name osc_100m_out_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name sys_rst_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name clk_sel_25m_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name clk_sel_100m_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_port -pin_name clk4div_out_pad -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
```

- 版图



- 输出数据

```
- u0_sdram_ba1_o_pad_PAD PAD60GU + FIXED ( 2046000 2680 ) N
 ;
- u0_sdram_addr12_o_pad_PAD PAD60GU + FIXED ( 1926000 2680 ) N
 ;
- u0_sdram_addr10_o_pad_PAD PAD60GU + FIXED ( 1806000 2680 ) N
```

```
PINS 104 ;
 - chiplink_rx_clk_pad + NET chiplink_rx_clk_pad + DIRECTION INPUT  + USE SIGNAL
 + LAYER ALPA ( -31000 -30000 ) ( 31000 30000 ) + FIXED ( 2399500 4462160 ) N
;
 - chiplink_rx_data0_pad + NET chiplink_rx_data0_pad + DIRECTION INPUT  + USE SIGNAL
 + LAYER ALPA ( -31000 -30000 ) ( 31000 30000 ) + FIXED ( 2466500 4462160 ) N
;
 - chiplink_rx_data1_pad + NET chiplink_rx_data1_pad + DIRECTION INPUT  + USE SIGNAL
 + LAYER ALPA ( -31000 -30000 ) ( 31000 30000 ) + FIXED ( 2533500 4462160 ) N
```

功能：宏单元规划

- 宏单元规划
  - 宏单元放置
  - 增加Blockage（布局、布线）
  - 增加Halo（布线）
  - PDN后清除布线障碍
- 脚本

```
create_inst -inst_name esd_pll_left                                    -coord_x 289810 -coord_y 2415620    -orient R90 -type NETLIST  -cellmaster PCLAMPC_H_G -status FIXED
create_inst -inst_name esd_pll_top                                     -coord_x 390070 -coord_y 2655100    -orient R0  -type NETLIST  -cellmaster PCLAMPC_V_G -status FIXED
place_instance -inst_name u0_rcg/u0_pll                                   -llx 359920     -lly 2344000    -orient MX    -cellmaster PLLTS28HPMLAINT
place_instance -inst_name u0_soc_top/u0_ysyx_210539/icache/Ram_bw/ram     -llx 596540     -lly 370000     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/icache/Ram_bw_1/ram   -llx 521000     -lly 370000     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/icache/Ram_bw_2/ram   -llx 445460     -lly 370000     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/icache/Ram_bw_3/ram   -llx 369920     -lly 370000     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw/ram     -llx 445460     -lly 977180     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw_1/ram   -llx 445460     -lly 1584360    -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
place_instance -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw_2/ram   -llx 369920     -lly 977180     -orient R0    -cellmaster TS5N28HPCPLVTA64X128M2FW
```

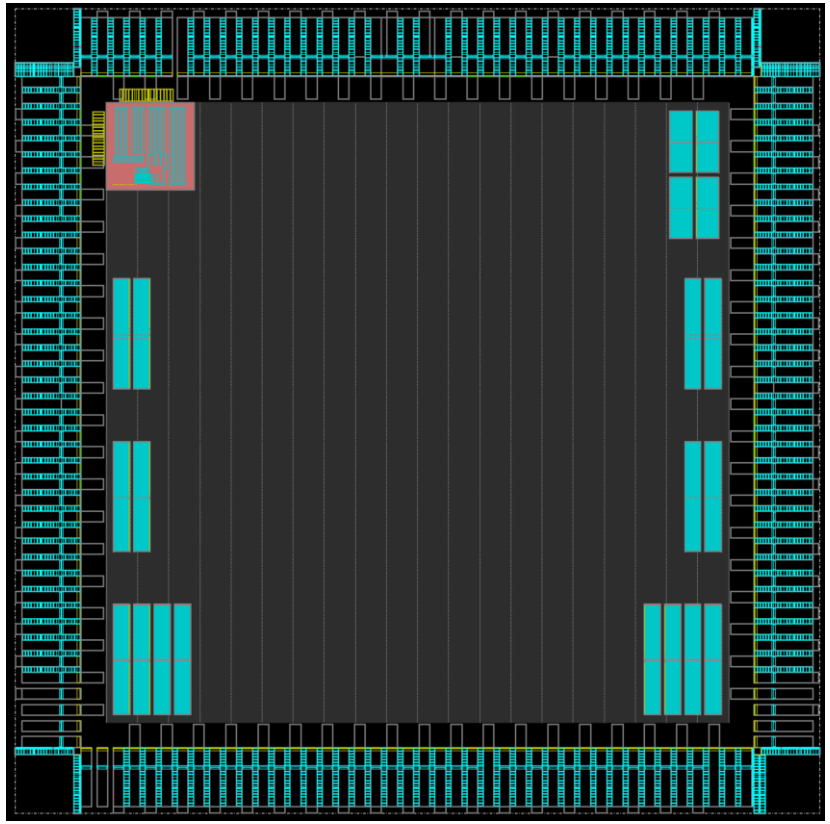
```
# blockage
add_placement_blockage -box "339920 2324000 667920 2652000"
add_placement_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer11 -distance "2000 2000 2000 2000"
add_placement_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer12 -distance "2000 2000 2000 2000"
add_placement_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer21 -distance "2000 2000 2000 2000"
add_placement_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer22 -distance "2000 2000 2000 2000"
add_placement_halo -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw/ram -distance "2000 2000 2000 2000"
add_placement_halo -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw_1/ram -distance "2000 2000 2000 2000"
```

```
add_routing_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer11 -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer12 -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer21 -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_vga_ctrl/vga/buffer22 -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw/ram -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw_1/ram -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
add_routing_halo -inst_name u0_soc_top/u0_ysyx_210539/dcache/Ram_bw_2/ram -layer "M1 M2 M3 M4 M5 M6 M7" -distance "2000 2000 2000 2000"
```

```
add_routing_blockage -layer "M1 M2 M3 M4 M5 M6 M7 M8 M9 AP" -box "339920 2324000 667920 2652000"
```

```
clear_blockage -type routing
```

- 版图



- 输出数据

```
  - u0_soc_top/u0_ysyx_210539/icache/Ram_bw_1/ram TS5N28HPCPLVTA64X128M2FW  + FIXED ( 521000 370000 ) N
  ;
```

```
BLOCKAGES 21 ;
    - PLACEMENT RECT ( 339920 2324000 ) ( 667920 2649600 ) ;
    - PLACEMENT RECT ( 2539710 2142740 ) ( 2622380 2372160 ) ;
    - PLACEMENT RECT ( 2441040 2142740 ) ( 2523710 2372160 ) ;
    - PLACEMENT RECT ( 2539710 2388160 ) ( 2622380 2617580 ) ;
    - PLACEMENT RECT ( 2441040 2388160 ) ( 2523710 2617580 ) ;
```

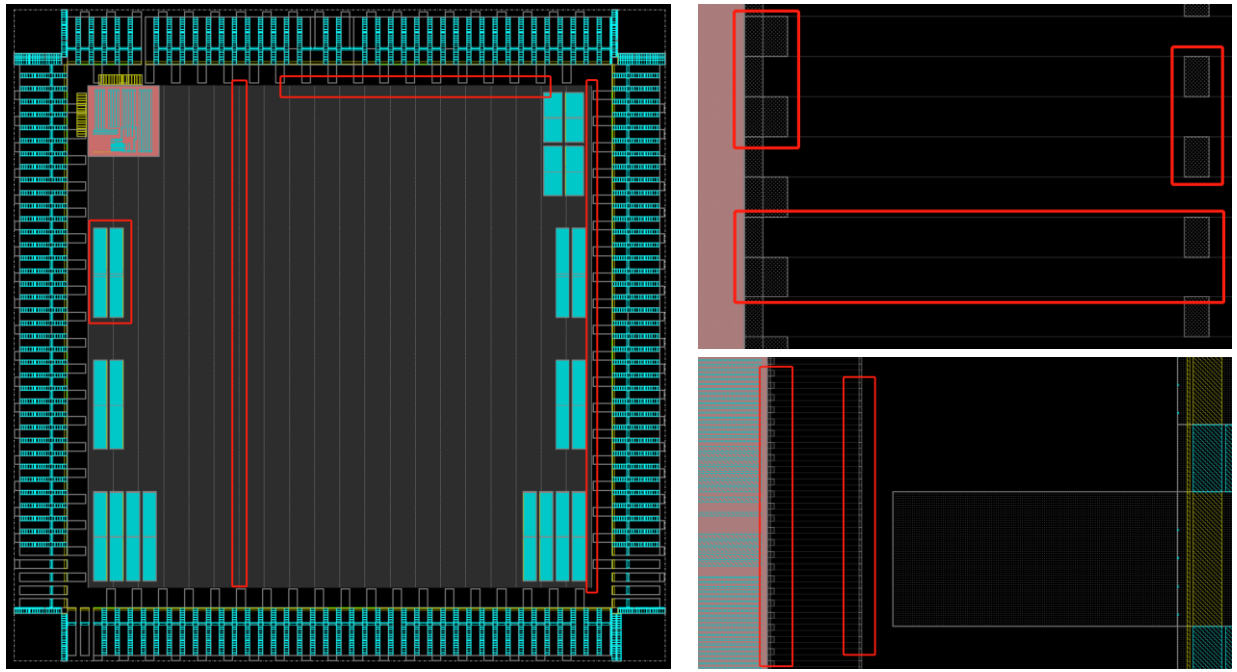
功能：物理单元放置

- 物理单元放置
  - Endcap
    - 边界处、宏单元周围
    - 闩锁效应、消除不对称性
  - Tapcell
    - 为所有标准单元的N阱和衬底提供偏置电源

- 脚本

```
#=============================================================
##   Tap Cell
#=============================================================
tapcell \
   -tapcell TAPCELLBWP40P140 \
   -distance 58 \
   -endcap BOUNDARY_LEFTBWP40P140
```

- 版图



- 输出数据

```
- PHY_0 TAPCELLBWP40P140 + SOURCE DIST + FIXED ( 340760 338400 ) FS
  ;
- PHY_1 TAPCELLBWP40P140 + SOURCE DIST + FIXED ( 571760 338400 ) FS
  ;
- PHY_2 TAPCELLBWP40P140 + SOURCE DIST + FIXED ( 803600 338400 ) FS
  ;
- PHY_3 TAPCELLBWP40P140 + SOURCE DIST + FIXED ( 1035440 338400 ) FS
  ;
```

```
- ENDCAP_0 BOUNDARY_LEFTBWP40P140 + SOURCE DIST + FIXED ( 339920 338400 ) FS
  ;
- ENDCAP_1 BOUNDARY_LEFTBWP40P140 + SOURCE DIST + FIXED ( 2659160 338400 ) FS
  ;
```

功能：电源规划

- 电源规划
  - 创建电源网络
  - Global Connect
  - 构建宏单元电源网络
  - 构建标准单元电源网络
  - 创建高层电源网络
  - 构建电源通孔
  - 连接宏单元
  - 连接电源IO

- 脚本

```
add_pdn_io -net_name VDD   -direction INOUT -is_power 1
add_pdn_io -net_name VSS   -direction INOUT -is_power 0
```

```
global_net_connect -net_name VDD     -instance_pin_name VDD       -is_power 1
global_net_connect -net_name VDD     -instance_pin_name VDDA      -is_power 1
global_net_connect -net_name VSS     -instance_pin_name VSS       -is_power 0
```

```
place_pdn_port -pin_name VDD -io_cell_name vdd_top -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_pdn_port -pin_name VDD -io_cell_name vdd_top_1 -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
place_pdn_port -pin_name VDD -io_cell_name vdd_top_2 -offset_x 2500 -offset_y 63000 -width 62000 -height 60000 -layer ALPA
```

```
create_grid -layer_name M1 -net_name_power VDD -net_name_ground VSS -width 0.15
create_grid -layer_name M2 -net_name_power VDD -net_name_ground VSS -width 0.15
```

```
create_stripe -layer_name M7 -net_name_power VDD -net_name_ground VSS -width 0.45 -pitch 10 -offset 2.925
create_stripe -layer_name M8 -net_name_power VDD -net_name_ground VSS -width 4 -pitch 9.6 -offset 0.5
```

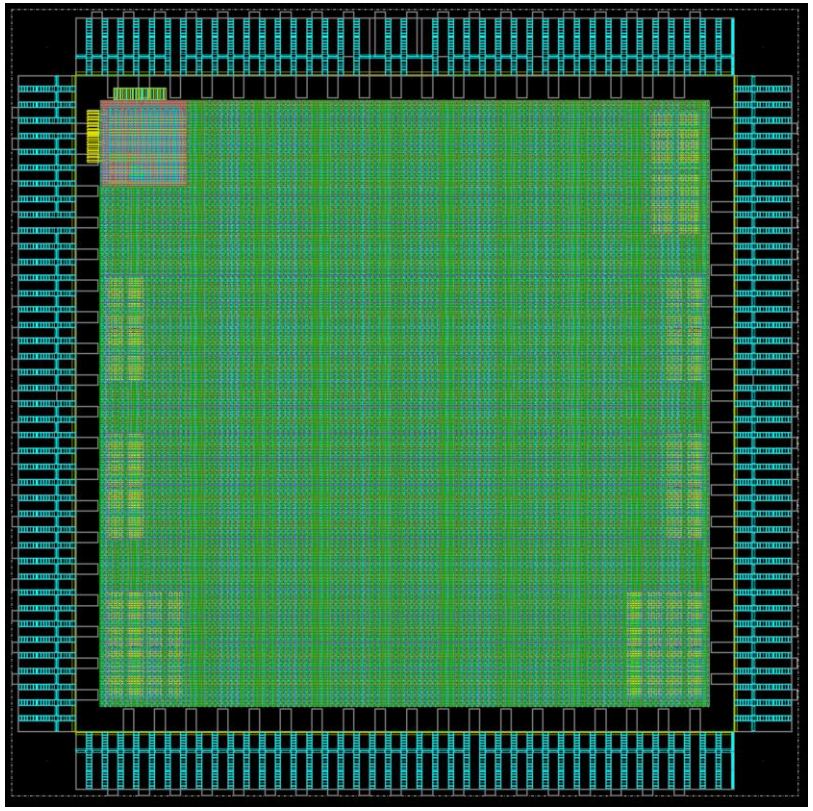
```
connect_two_layer -layers [concat $connect2 $connect3 $connect4 $connect5 $connect6]
```

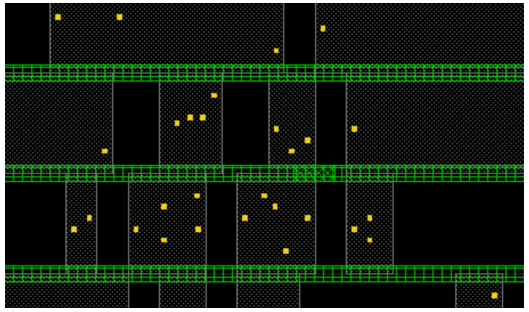
```
connect_macro_pdn -pin_layer "M4" -pdn_layer "M7" -power_pins "VDD" -ground_pins "VSS" -orient "R0 R180 MX MY"
```

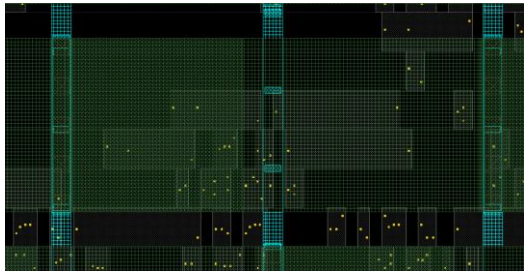
```
connect_io_pin_to_pdn -point_list "938 4450 938 4297" -layer ALPA
connect_io_pin_to_pdn -point_list "1006 4450 1006 4337" -layer ALPA
```

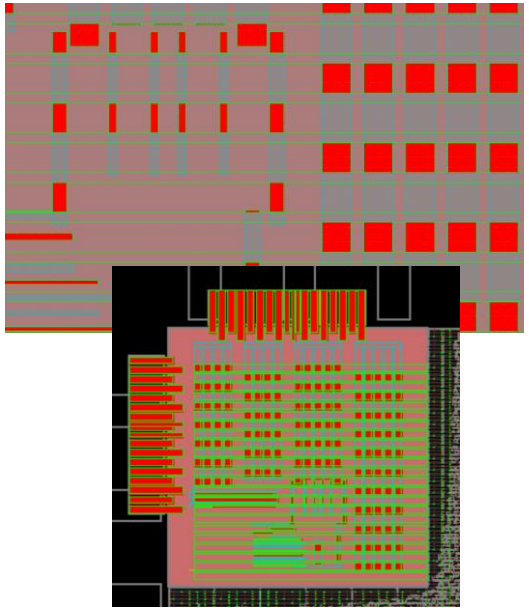
```
connect_pdn_stripe -point_list "1011.995 4331.72 1011.995 4373.04" -net_name VDD -layer METAL8
connect_pdn_stripe -point_list "1131.995 4331.72 1131.995 4373.04" -net_name VDD -layer METAL8
```

- 版图









- 输出数据

```
SPECIALNETS 5 ;
- VDD ( * VDD ) ( * VDDA ) ( * VDDESD )
  + USE POWER
    + ROUTED M1 300 + SHAPE FOLLOWPIN ( 339920 338400 ) ( 2660000 * )
      NEW M1 300 + SHAPE FOLLOWPIN ( 339920 342000 ) ( 2660000 * )
      NEW M1 300 + SHAPE FOLLOWPIN ( 339920 345600 ) ( 2660000 * )
      NEW M1 300 + SHAPE FOLLOWPIN ( 339920 349200 ) ( 2660000 * )
      NEW M1 300 + SHAPE FOLLOWPIN ( 339920 352800 ) ( 2660000 * )
      NEW M1 300 + SHAPE FOLLOWPIN ( 339920 356400 ) ( 2660000 * )
```

iFP：TCL命令集

商业工具	iEDA	功能说明
floorPlan	init_floorplan	初始化版图信息，配置包含DIE面积、CORE面积、SITE信息
	gern_track	生成版图Track信息
placeInstance	place_instance	单元放置，如果单元不存在，则先由指定cellmaster创建单元，再进行单元放置
addInst	create_inst	创建单元
createPlaceBlockage	add_placement_blockage	设置布局Blockage
	add_placement_halo	设置指定宏单元的布局Halo，其中单元位置必须已经放置在版图中
addRoutingHalo	add_routing_halo	设置指定宏单元的布线Halo
createRouteBlk	add_routing_blockage	设置布线Blockage
addIoFiller	place_io_filler	摆放IOFiller，支持四个边自动填充
addWellTap	tapcell	放置tapcell以及endcap
	clear_blockage	清除blockage

iPDN：TCL命令集

商业工具	iEDA	功能说明
	add_pdn_io	为电源Net添加IO Pin
globalNetConnect	global_net_connect	创建电源网络
createPhysicalPin	place_pdn_port	放置IO Pin，且位置由指定的IO Cell及配置决定
sroute	create_grid	生成标准单元供电网络
addStripe	create_stripe	生成标准单元条形电源线
editPowerVia	connect_two_layer	连接指定的两层的电源网络
	connect_macro_pdn	连接宏单元的电源线
add_shape	add_segment_stripe	为指定电源网络增加电源线；如果设置了point_begin和point_end，增加电源线的同时，在point_end坐标点打上通孔

iFP/iPDN报告解读

- fp_db.rpt（floorplan后的版图数据Summary）

```
#####################################################################
Summary
+-----------------------+-------------------------------------------+
| Module                | Value                                     |
+-----------------------+-------------------------------------------+
| DIE Area ( um^2 )     | 2249940.000000 = 1499.960000 * 1500.000000 |
| DIE Usage             | 0.166554                                  |
| CORE Area ( um^2 )    | 1340542.224000 = 1160.040000 * 1155.600000 |
| CORE Usage            | 0.279541                                  |
|                       |                                           |
| Number - Site         | 9                                         |
| Number - Row          | 1284                                      |
| Number - Track        | 20                                        |
| Number - Layer        | 32                                        |
| Number - Routing Layer | 10                                       |
| Number - Cut Layer    | 10                                        |
| Number - GCell Grid   | 0                                         |
| Number - Cell Master  | 16314                                     |
| Number - Via Rule     | 516                                       |
|                       |                                           |
| Number - IO Pin       | 110                                       |
| Number - Instance     | 306234                                    |
| Number - Blockage     | 21                                        |
| Number - Filler       | 0                                         |
| Number - Net          | 311863                                    |
| Number - Special Net  | 5                                         |
+-----------------------+-------------------------------------------+
```

iFP/iPDN报告解读

- fp_db.rpt（Instance统计）

Summary - Instance

Type	Number	Number Ratio	Area	Area Ratio
All Instances	306234	1	1540250545312	1
Netlist	283484	0.92571	1498943713312	0.973182
Physical	22750	0.0742896	41306832000	0.0268183
Timing	0	0	0	0
Core	305518	0.997662	990416448000	0.643023
Core - logic	282768	0.923372	949109616000	0.616205
Pad	540	0.00176336	63126315520	0.0409844
Block	23	7.5106e-05	22214492576	0.0144226
Endcap	0	0	0	0
Cover	153	0.000499618	464493289216	0.30157
Ring	0	0	0	0

iFP/iPDN报告解读

- fp_db.rpt（Layer信息统计）

Summary - Layer

Layer	Net - Wire Length	Net - Wire Number	Net - Via Number	Net - Patch Number	Special Net - Wire Length	Special Net - Wire Number	Special Net - Via Number
NW	0	0	0	0	0	0	0
PW	0	0	0	0	0	0	0
PO	0	0	0	0	0	0	0
CO	0	0	0	0	0	0	0
VTUL_N	0	0	0	0	0	0	0
VTUL_P	0	0	0	0	0	0	0
VTL_N	0	0	0	0	0	0	0
VTL_P	0	0	0	0	0	0	0
VTH_N	0	0	0	0	0	0	0
VTH_P	0	0	0	0	0	0	0
VTUH_N	0	0	0	0	0	0	0
VTUH_P	0	0	0	0	0	0	0
M1	0	0	0	0	2662333020	5447	0
VIA1	0	0	0	0	0	0	0
M2	0	0	0	0	2662333020	5447	0
VIA2	0	0	0	0	0	0	129652
M3	0	0	0	0	0	0	0
VIA3	0	0	0	0	0	0	129652
M4	0	0	0	0	0	0	0
VIA4	0	0	0	0	0	0	132074
M5	0	0	0	0	0	0	0
VIA5	0	0	0	0	0	0	132074
M6	0	0	0	0	0	0	0
VIA6	0	0	0	0	0	0	132074
M7	0	0	0	0	481072540	278	0
VIA7	0	0	0	0	0	0	24736
M8	0	0	0	0	547659280	241	0
VIA8	0	0	0	0	0	0	56926
M9	0	0	0	0	558927200	241	0
RV	0	0	0	0	0	0	2722
AP	0	0	0	0	52377840	23	0
OVERLAP	0	0	0	0	0	0	0

iFP/iPDN报告解读

- fp_db.rpt（Pin信息统计）

Summary - Pin Distribution

Pin Number	Net Number	Net Ratio	Instance Number	Instance Ratio
0	23575	0.075594	22903	0.074789
1	32	0.000103	54	0.000176
2	190758	0.611672	55603	0.181570
3	56737	0.181929	84767	0.276805
4	17285	0.055425	41624	0.135922
5	9805	0.031440	99808	0.325921
6	2252	0.007221	1351	0.004412
7	1042	0.003341	0	0.000000
8	775	0.002485	101	0.000330
9	1387	0.004447	2	0.000007
10	439	0.001408	0	0.000000
11	217	0.000696	0	0.000000
12	488	0.001565	0	0.000000
13	257	0.000824	0	0.000000
14	128	0.000410	0	0.000000
15	208	0.000667	0	0.000000
16	108	0.000346	0	0.000000
17	519	0.001664	0	0.000000
18	294	0.000943	0	0.000000
19	110	0.000353	0	0.000000
20	98	0.000314	0	0.000000
21	132	0.000423	0	0.000000
22	231	0.000741	0	0.000000
23	1064	0.003412	0	0.000000
24	66	0.000212	0	0.000000
25	330	0.001058	0	0.000000
26	119	0.000382	0	0.000000
27	64	0.000205	0	0.000000
28	100	0.000321	0	0.000000
29	105	0.000337	0	0.000000
30	58	0.000186	0	0.000000
31	62	0.000199	0	0.000000
32	125	0.000401	0	0.000000
>= 32	2893	0.009277	21	0.000069

总结

- **iFP和iPDN提供布图规划相关的TCL接口和Flow**
- **依赖工程师经验**
- **未来计划**
  - 集成 iMP
  - 集成 AiMP
  - 丰富GUI交互
  - 辅助设计（Auto Macro PDN、Auto Pins等）

iEDA Tutorial 第二期议程

- Part 1 iEDA-iFP/iPDN工具架构、特性与使用（黄增荣）

- **Part 2 iEDA-iMP 关键技术（黄富兴）**

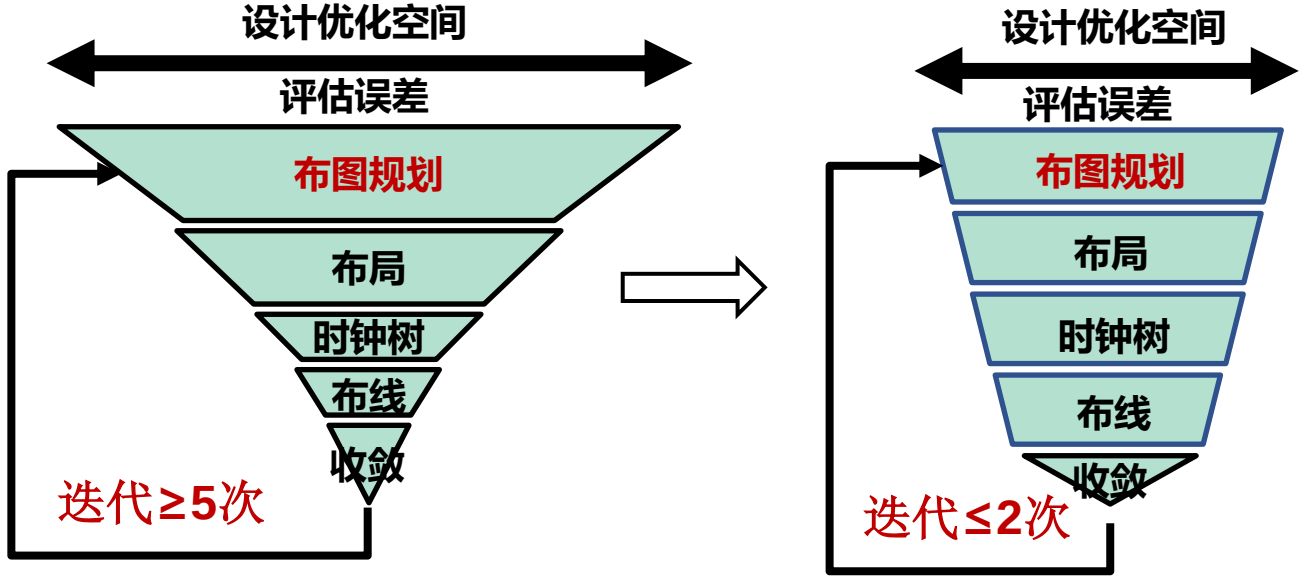
- Part 3 iEDA-iPL 问题介绍、架构、使用与规划 （陈仕健）

- Part 4 iEDA-iPL 关键技术 （邱奕杭）

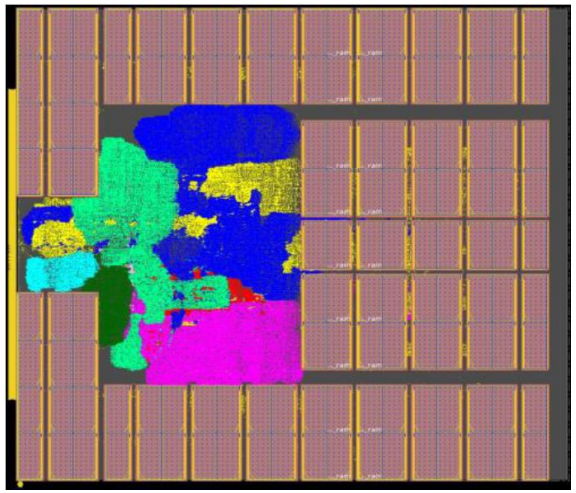
- Part 5 iEDA-iTO工具架构、特性与关键技术 （吴鸿熙）

宏单元布局自动化工具-iMP

- 宏单元：Intellectual Property(IP)核，例如：RAM，ROM。
- 特点：宏单元数量远小于标准单元，面积远大于标准单元，总面积占比超过一半。
- 布图规划：IO布局、宏单元布局、电源规划、填充物插入。
- 挑战：宏单元布局属于布局实现最开始步骤，设计优化空间大，但是指标评估误差大；解空间大，目标和约束多，探索极其困难；传统芯片宏单元布局设计以工程师摆放为主。
- 价值：辅助工程师设计，减少设计时间；优化更多目标，提高宏单元布局质量。



优化空间大，评估误差大

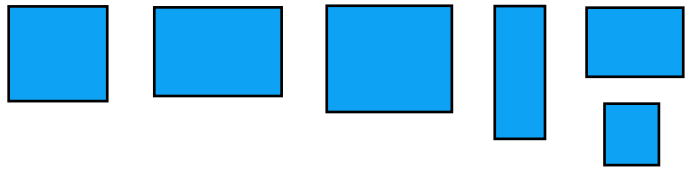


人工耗时数周

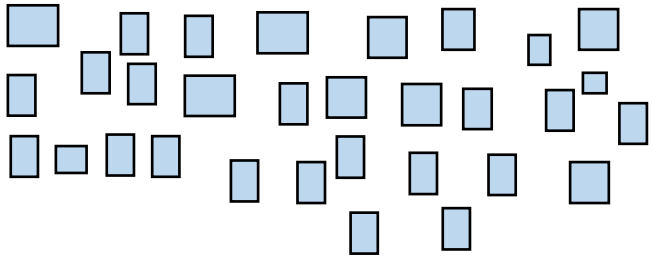
布图&宏单元布局：研究问题

- **问题（Problem）**
- 输入:
  - 宏单元集合，
  - 标准单元集合，
  - 网表。
- 目标:
  - 放置在固定边框内，
  - 缩短线长，
  - 最小化面积，
  - 最小化时序，
  - 可布线性。

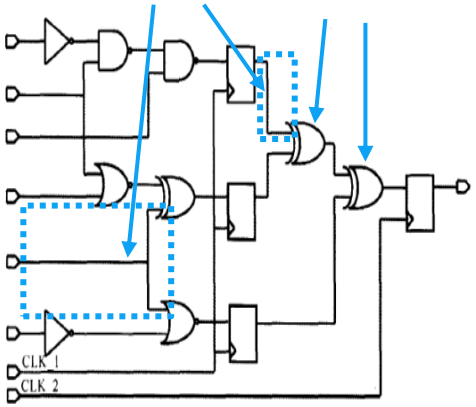
宏单元



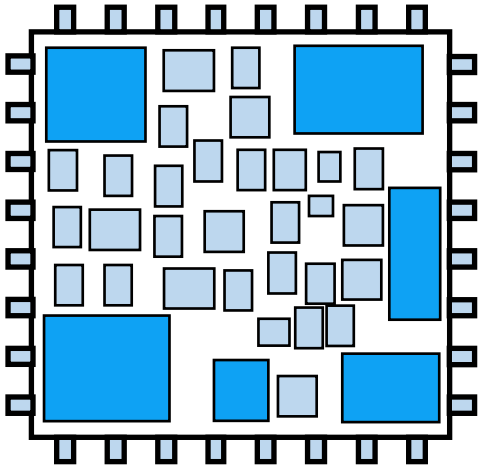
标准单元



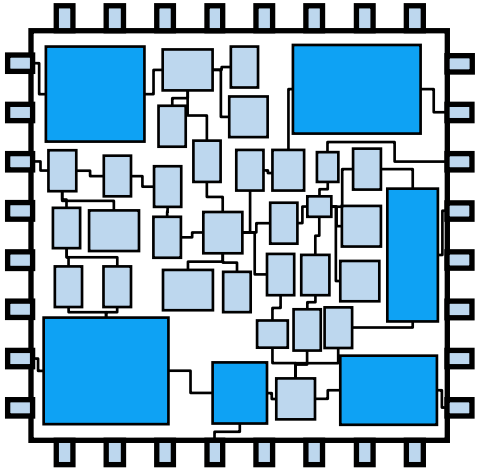
线网　逻辑单元



网表



放置在布局区域内



最小化线长，时序，
提高可布线性

01 研究问题

02 研究内容

03 未来展望

布图&宏单元布局：研究内容

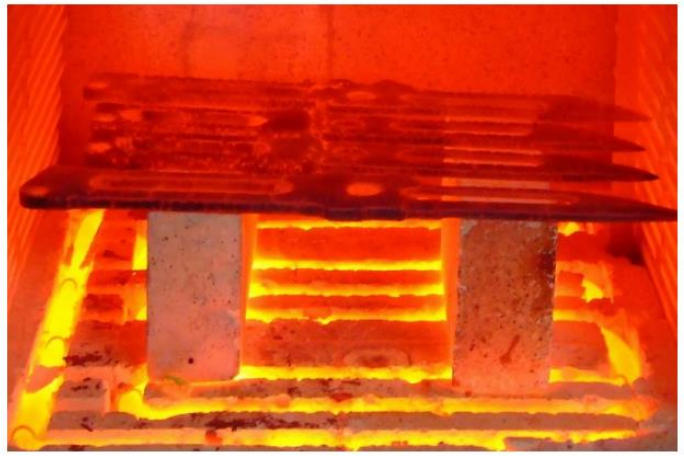
- **方法（How）**
  - 模拟退火：一种数值优化技术，用于在因空间过大而普通搜索方法无法取得结果的情况下搜索解决方案。
  - 基于多级框架的贪心算法：贪心算法是一种常用的求解最优化问题的方法。它的基本思想是通过每一步的局部最优解来构建全局最优解。
  - 解析法: 建模成数学模型，使用最优化方法（nesterov, CG）计算出宏单元的位置。
  - AI：强化学习

模拟退火方法

- **模拟退火方法是一种常用的全局优化算法，其思想来源于固体材料的冷却过程中晶格缺陷的退火过程。模拟退火方法通过模拟固体材料的退火过程，寻找最优解。**
- **在模拟退火方法中，首先需要定义一个能量函数，即优化问题的目标函数。然后，在一个初始解（温度较高）的邻域中进行随机扰动，计算扰动后的解的能量，根据一定的概率接受扰动解作为新的当前解。接着，逐渐降低温度，控制概率接受新解的程度，并在每个温度下重复上述过程。通过不断的迭代过程，模拟退火方法可以逐渐搜索到全局最优解。**

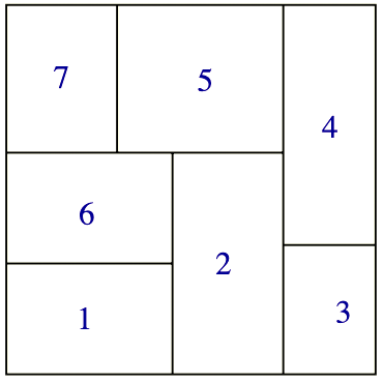
模拟退火方法的基本步骤如下：

1. 初始化：确定初始解和初始温度。
2. 生成新解：在当前解的邻域中进行随机扰动，得到新的解。
3. 计算能量差：计算新解与当前解之间的能量差或距离差。
4. 判断接受条件：根据一定的概率，判断是否接受新解作为当前解。概率通常依赖于能量差和当前温度。
5. 降低温度：温度下降的速度决定了新解被接受的几率，通常采用指数函数使温度逐渐降低。
6. 终止条件：根据设定的终止条件判断是否结束算法。终止条件可以是达到最大迭代次数、温度下降到一定程度或找到满足需求的解。



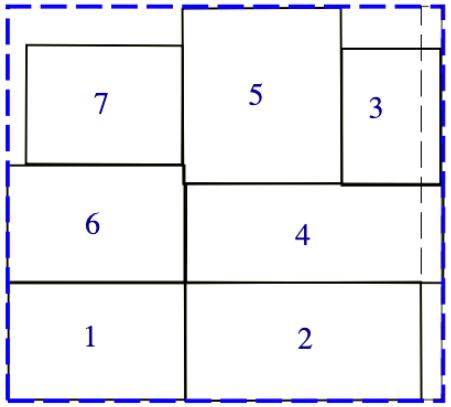
模拟退火方法

- 如何用模拟退火方法解决宏单元布局问题?
  - 模拟退火方法的关键:
    - 定义解-表达式
    - 产生新解-扰动
    - 能量函数-评估



E_1= 16H2V75VH34HV

表达式



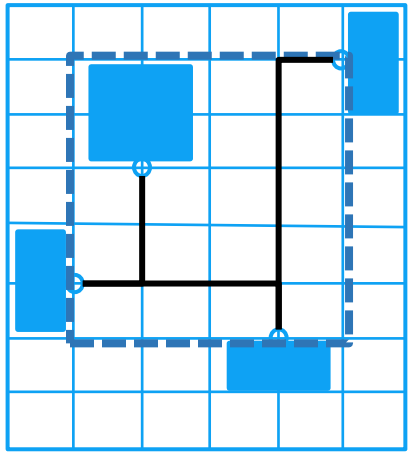
E_2 = 16H7H24H53VHV

产生新解

$$\Delta Cost = Cost(E_2) - Cost(E_1)$$

$$cost(E) = \lambda WL(E) + Area(E)$$

- Area: area of the smallest rectangle
- WL : overall wirelength
- λ: user-specified parameter



$$WL_{HPWL}(x,y) = \sum_{e \in E}\left(\max_{m_i,m_j \in e}|x_i - x_j| + \max_{m_i,m_j \in e}|y_i - y_j|\right)$$

半周长线长

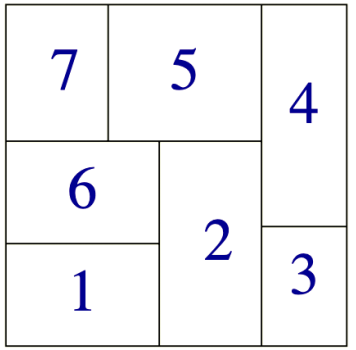
评估

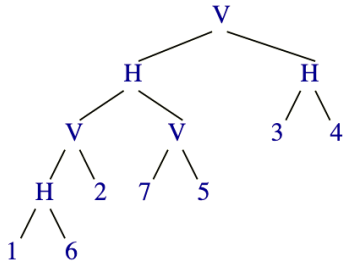
Floorplan表达式发展历程

- Slicing Tree[1] - 1986
- Sequence pair[2] -1995
- O-tree[3] -1999
- B*-tree[4] -2000
- Corner Block List[5] -2000
- Corner Sequence [6] -2003
- Twin Binary Sequence [7] -2003
- TCG, TCG-S[8] -2004
- ACG[9] -2004

Floorplan Representation

Slicing Tree



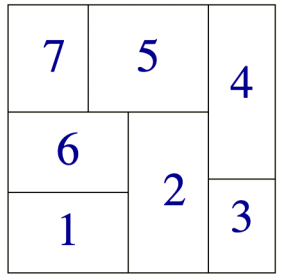


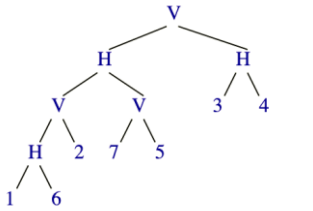
E = 16H2V75VH34HV

[1] D. Wong and C. Liu, “A new algorithm for floorplan design,” in Proc. Des. Autom. Conf. (DAC), 1986, pp. 101–107.
[2] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, “Rectangle- packing-based module placement,” in Proc. Int. Conf. Comput.-Aided Design (ICCAD), 1995, pp. 472–479.
[3] P.-N.Guo,C.-K.Cheng,andT.Yoshimura,“An O-tree representation of non-slicing floorplan and its applications,” in Proc. Des. Autom. Conf. (DAC), 1999, pp. 268–273.
[4] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, “B*-trees: A new representation for non-slicing floorplans,” in Proc. Des. Autom. Conf. (DAC), 2000, pp. 485–463.
[5] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, “Corner block list: an effective and efficient topological representation of non-slicing floorplan,” in Proc. Int. Conf. Comput.-Aided Design (ICCAD), 2000, pp. 8–12.
[6] J.-M. Lin, Y.-W. Chang, and S.-P. Lin, “Corner sequence - a P-admissible floorplan representation with a worst case linear-time packing scheme,” IEEE Trans. Very Large Scale Integr VLSI Syst., vol. 11, no. 4, pp. 679–686, 2003.
[7] E. F. Young, C. C. Chu, and Z. C. Shen, “Twin binary sequences: A nonredundant representation for general nonslicing floorplan,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 22, no. 4, pp. 457–469, 2003.
[8] J.-M. Lin and Y.-W. Chang, “TCG-S: orthogonal coupling of P*- admissible representations for general floorplans,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 23, no. 6, pp. 968–980, 2004.
[9] H. Zhou and J. Wang, “ACG-adjacent constraint graph for general floorplans,” in Proc. Int. Conf. Comput. Aided Design (ICCAD), 2004, pp. 572– 575.

版图表达式

- **切片树[1]（Slicing tree)**
  - **表达式**





E = 16H2V75VH34HV

  - **扰动**
  - – M1 (Operand Swap): Swap two adjacent operands.
  - – M2 (Chain Invert): Complement some chain (V = H, H = V).
  - – M3 (Operator/Operand Swap): Swap two adjacent operand and operator.

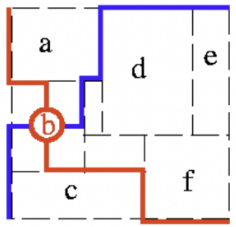
  - **Packing：逆波兰式，后缀表示法-O(n)**
  - **解空间：$O(n!2^{2.6n}/n1.5)$**

优点：简单易实现，解空间小,packing效率快
缺点：只能表示slicing 结构。

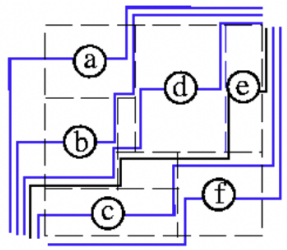
[1]D. Wong and C. Liu, “A new algorithm for floorplan design,” in *Proc. Des. Autom. Conf. (DAC)*, 1986, pp. 101–107.

版图表达式

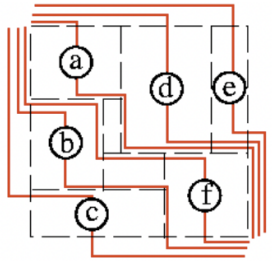
- **序列对[1]（Sequence pair)**
  - **表达式**



Loci of module b



Positive loci: abdecf



Negative loci: cbfade

(Γ_+, Γ_-) = (abdecf, cbfade)

$\mathcal{M}^{aa}(x) = \{x' \mid x' \text{ is after } x \text{ in both } \Gamma_+ \text{ and } \Gamma_-\}$,

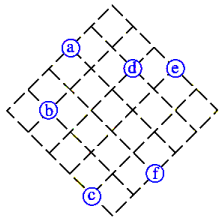
$\mathcal{M}^{bb}(x) = \{x' \mid x' \text{ is before } x \text{ in both } \Gamma_+ \text{ and } \Gamma_-\}$,

$\mathcal{M}^{ba}(x) = \{x' \mid x' \text{ is before } x \text{ in } \Gamma_+ \text{ and after } x \text{ in } \Gamma_-\}$,

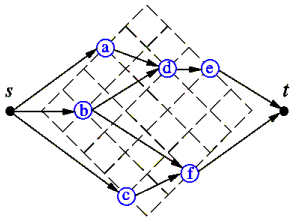
$\mathcal{M}^{ab}(x) = \{x' \mid x' \text{ is after } x \text{ in } \Gamma_+ \text{ and before } x \text{ in } \Gamma_-\}$.

Theorem 2 :
Let **Gridding**(Π) = (Γ_+, Γ_-). If $x' \in \mathcal{M}^{aa}(x)$, then x' is right to x in Π.
The claim holds replacing the pair of words ("$\mathcal{M}^{aa}$" and "right to") with any of ("$\mathcal{M}^{bb}$" and "left to"), ("$\mathcal{M}^{ba}$" and "above"), and ("$\mathcal{M}^{ab}$" and "below").
Before the proof, an example is shown. In Fig.3, modules d, e, f are in $\mathcal{M}^{aa}(b)$, and they are right to b in Π.

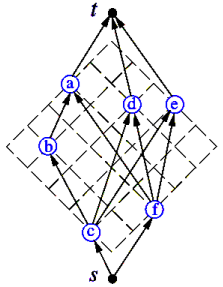
  - **Packing:**
    - **基于约束图:**



Packing for sequence pair: (abdecf, cbfade)



Horizontal constraint graph (Transitive edges are not shown)



Vertical constraint graph (Transitive edges are not shown)

拓扑排序 – $O(n^2) = O(n+e)$, where $o(e) = o(n^2)$

优点：可以表示任合版图，pack效率可接受
缺点：解空间大，packing算法复杂

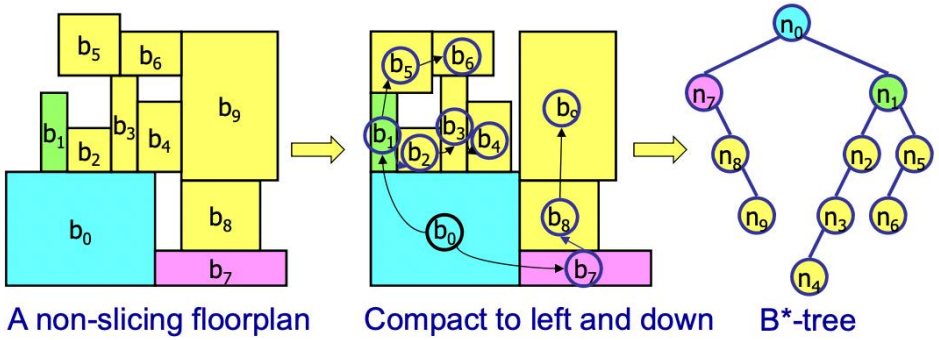
    - **基于最长公共子序列[2] -O(nlogn)**
  - **解空间：$O(n!)^2$**

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle- packing-based module placement," in Proc. Int. Conf. Comput.-Aided Design (ICCAD), 1995, pp. 472–479.

[2] Xiaoping Tang, Ruiqi Tian and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 12, pp. 1406-1413, Dec. 2001, doi: 10.1109/43.969434.

版图表达式

- B*-tree[1]
  - 表达式



  - Packing：
    - x坐标：
      - 左子节点：右侧相邻方块中最低的块（xj = xi + wi）。
      - 右子节点：上方第一个具有相同x坐标的块（xj = xi）。
    - y坐标：
      - 左子节点：具有相同y坐标的上方第一个块（yj = yi）。
      - 右子节点：右侧相邻方块中最低的块（yj = yi + hi）。
  - 解空间： $O(n!2^{2n}/n1.5)$

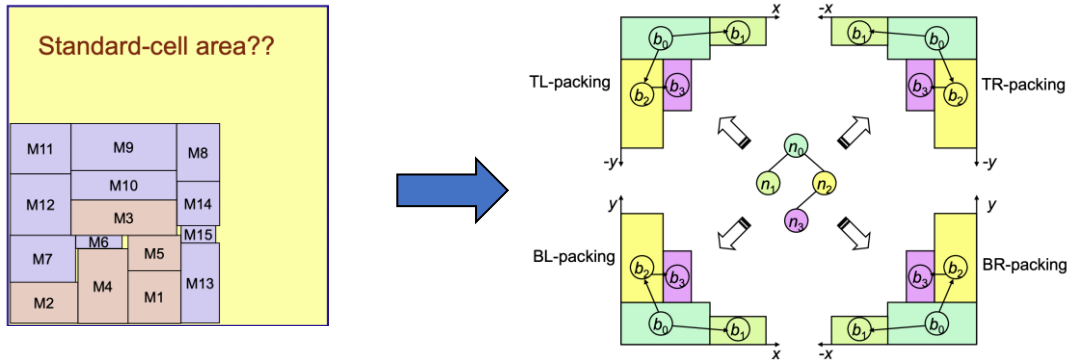
– O(n)

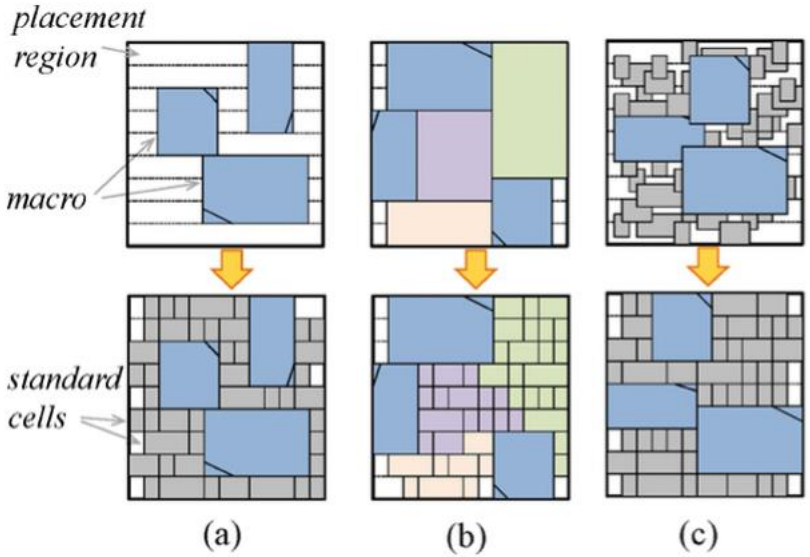
优点：简单易实现，解空间小,packing效率快，可以表示non-slicing版图
缺点：只能有compackted结构

[1] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, “B*-trees: A new representation for non-slicing floorplans,” in Proc. Des. Autom. Conf. (DAC), 2000, pp. 485–463.

宏单元布局的应用

- 如何处理宏单元与标准单元之间的关系?
  - MP-Tree[1]:

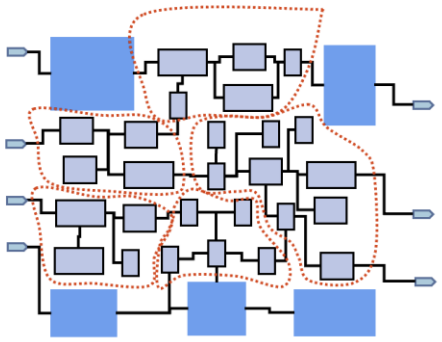




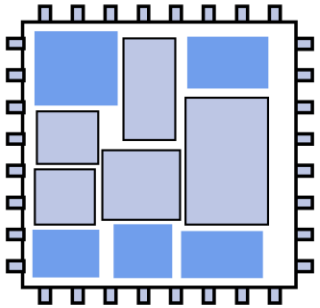
Mixed-Size Placement

(a) Two – stage
(b) Floorplanner guide
(c) One - stage

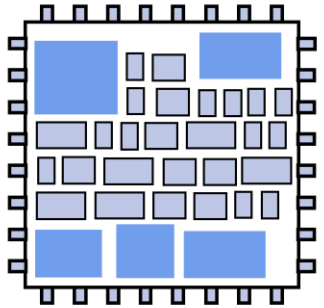
  - 划分:



划分



宏单元布局



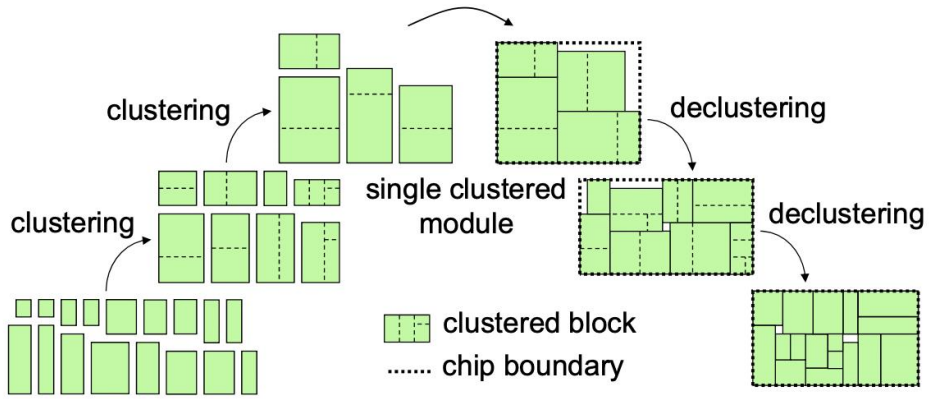
标准单元布局

[1] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and D. Liu, “MP-trees: A packing-based macro placement algorithm for modern mixed-size designs,” IEEE Trans. Comput.-Aided Des., vol. 27, no. 9, pp. 1621– 1634, Sep. 2008.

基于多级框架的贪心方法发展历程

为了应对大规模宏单元布局需求，基于多级框架的贪心方案被提出：

名称	框架类型	子问题方法	年份
MB*-tree[1]	自底向上聚类	基于B*-tree的模拟退火	2003
Capo[2]	自顶向下超图划分	基于SP的模拟退火	2004
IMF[3]	自顶向下超图2划分	基于B*-tree的模拟退火	2008
Defer[4]	自顶向下超图2划分	基于GST的贪心枚举	2010
Qinfer[5]	自顶向下超图2划分	拟牛顿合法化	2021



[1] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, “Multilevel floorplanning/placement for large-scale modules using B*-trees,” in Proc. Des. Autom. Conf. (DAC), 2003, pp. 812–817.
[2] S. Adya, S. Chaturvedi, J. Roy, D. Papa, and I. Markov, “Unification of partitioning, placement and floorplanning,” in Proc. Int. Conf. Comput.- Aided Design (ICCAD), 2004, pp. 550–557.
[3] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, “A new multilevel framework for large-scale interconnect-driven floorplanning,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 27, no. 2, pp. 286–294, 2008.
[4] J. Z. Yan and C. Chu, “Defer: Deferred decision making enabled fixed-outline floorplanning algorithm,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 29, no. 3, pp. 367–381, 2010.
[5] P. Ji, K. He, Z. Wang, Y. Jin, and J. Wu, “A quasi-newton-based floorplanner for fixed-outline floorplanning,” Comp. Oper. Res., vol. 129, p. 105225, 2021.

采用多级框架的方法

- **MB*-tree[1]**
  - **自底向上聚类**
    - Dead space: The area utilization for clustering two modules m_i and m_j can be measured by the resulting dead space s_{ij}, representing the unused area after clustering m_i and m_j. Let s_{tot} denote the dead space in the final floorplan P. We have $s_{tot} = A_{tot} - \sum_{m_i \in M} A_i$, where A_i denotes the area of module m_i and A_{tot} the area of the final enclosing rectangle of P. Since $\sum_{m_i \in M} A_i$ is a constant, minimizing A_{tot} is equivalent to minimizing the dead space s_{tot}.
    - Connectivity density: Let the connectivity c_{ij} denote the number of nets between two modules m_i and m_j. The *connectivity denisty* d_{ij} between two (primitive or cluster) modules m_i and m_j is given by

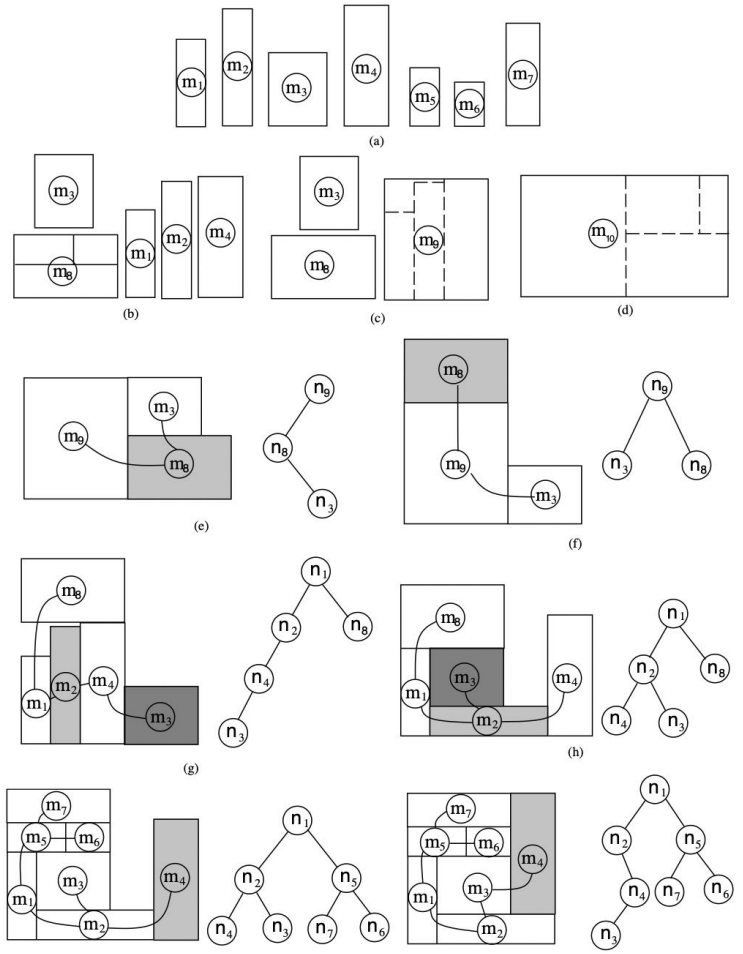
$$d_{ij} = c_{ij}/(n_i + n_j), \quad (1)$$

where n_i (n_j) denotes the number of primitive modules in m_i (m_j). Often a bigger cluster implies a larger number of connections. The connectivity density considers not only the connectivity but also the sizes of clusters between two modules to avoid possible biases.

$$\phi(\{m_i, m_j\}) = \alpha \hat{s}_{ij} + \frac{\beta K}{\hat{d}_{ij}},$$

  - **自顶向下解聚类**

    基于B*-tree的模拟退火[2]



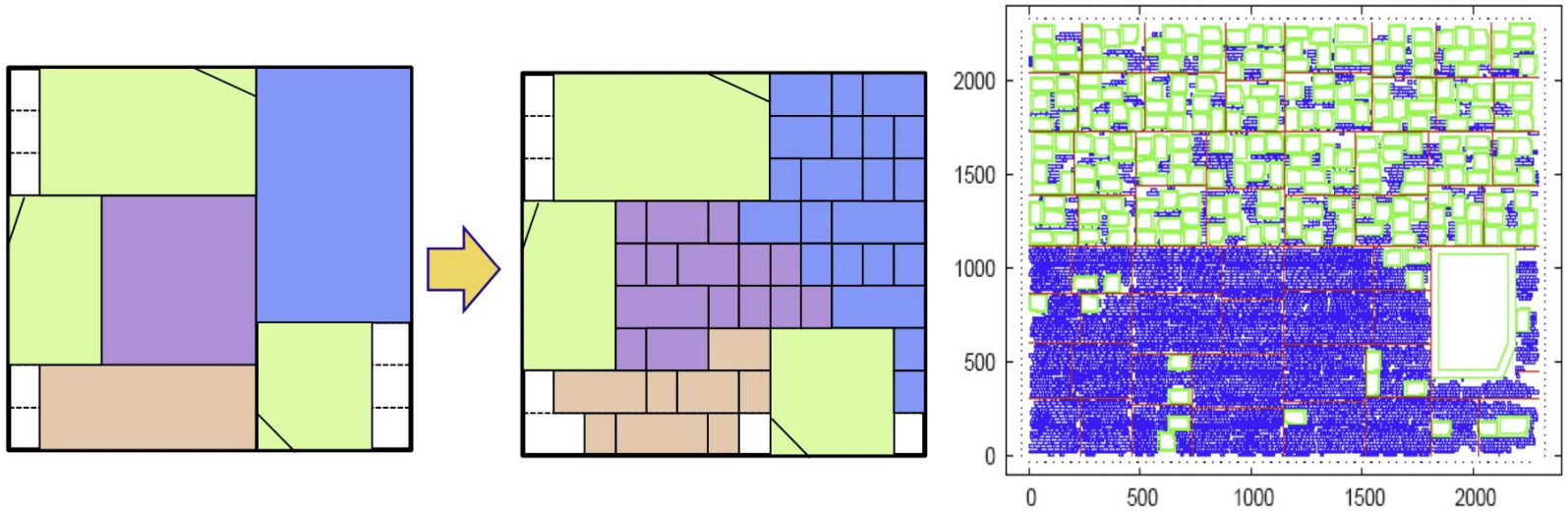
[1] H.-C. Lee, Y.-W. Chang, J.-M. Hsu, and H. Yang, “Multilevel floorplanning/placement for large-scale modules using B*-trees,” in Proc. Des. Autom. Conf. (DAC), 2003, pp. 812–817.

[2] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, “B*-trees: A new representation for non-slicing floorplans,” in Proc. Des. Autom. Conf. (DAC), 2000, pp. 485–463.

采用多级框架的方法

- **Capo[1]**

  首先使用最小切割划分，将原始电路切割成多个分区。同时，芯片区域被分割成小块的Bin。然后，采用Sequence pair基于模拟退火的版图规划[2]，在其相关的分区内对每个分区执行Fixed-outline floorplanning。



[1] S. Adya, S. Chaturvedi, J. Roy, D. Papa, and I. Markov, “Unification of partitioning, placement and floorplanning,” in Proc. Int. Conf. Comput.- Aided Design (ICCAD), 2004, pp. 550–557.
[2] S. Adya and I. Markov, “Fixed-outline floorplanning: enabling hierar- chical design,” IEEE Trans. Very Large Scale Integr VLSI Syst., vol. 11, no. 6, pp. 1120–1135, 2003.

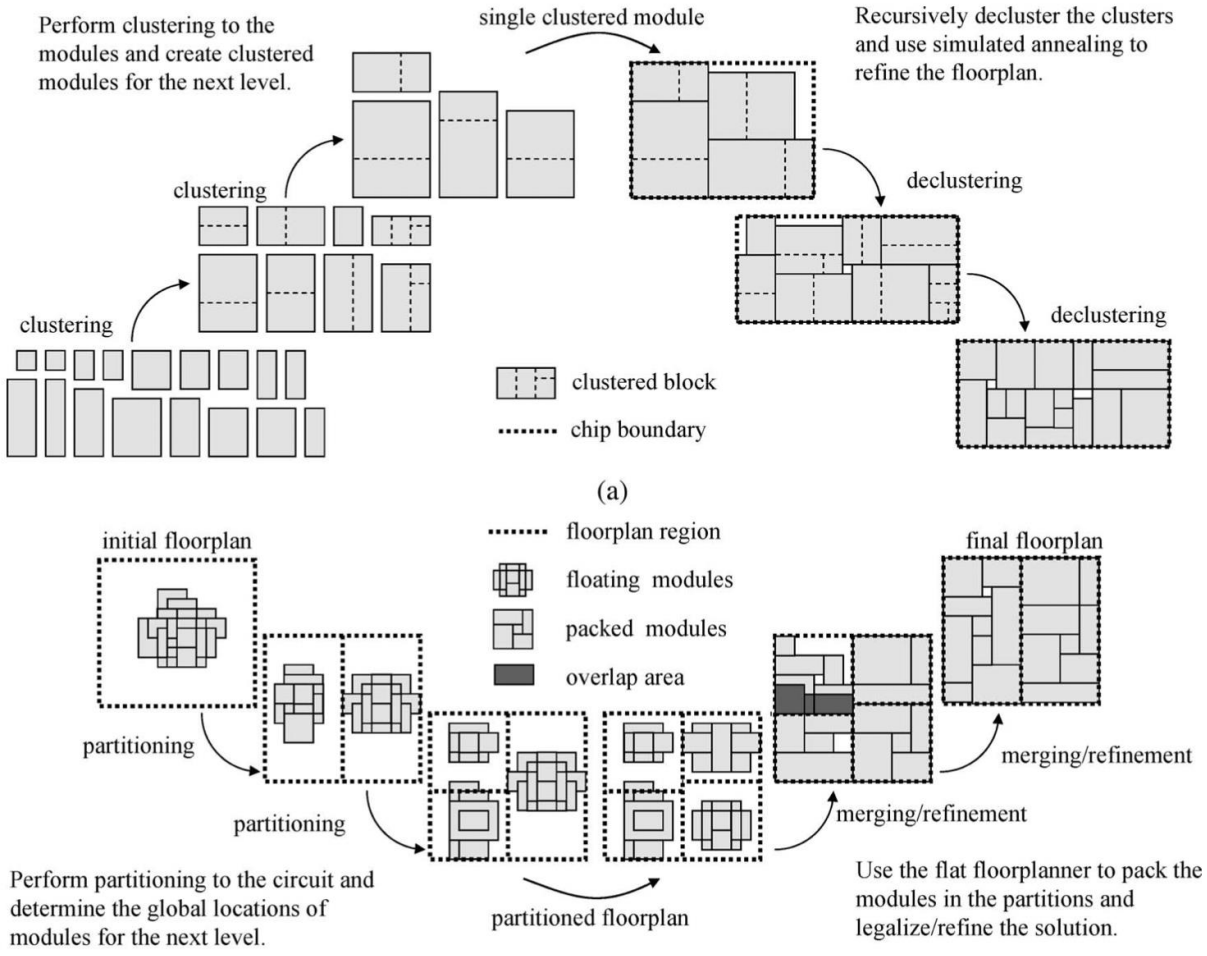
采用多级框架的方法

- **IMF[1]**
  - **自顶向下划分**

Interconnect-Driven Multilevel Floorplanning (IMF)
Input: Modules, nets, I/O pads, fixed-die parameters. Output: A feasible floorplan within the fixed-die with wirelength (HPWL) being minimized.
1. Initialize the data structures and the chip dimension; 2. Set all modules at the center of the floorplan region; 3. **The Partitioning Stage**: 4. **until** every region has fewer than n_{max} modules 5. Choose a partition; 6. Create a hypergraph model; 7. Bipartition the hypergraph; 8. Move modules to the new sub-regions; 9. Generate the partitioned floorplan; 10. **The Merging Stage**: 11. **while** there exists more than one region 12. Choose two neighboring regions; 13. Apply fixed-outline floorplanning; 14. Merge two regions; 15. Generate the final floorplan; 16. **return** the final floorplan;

  - **自底向上解聚类**

基于B*-tree的模拟退火[2]



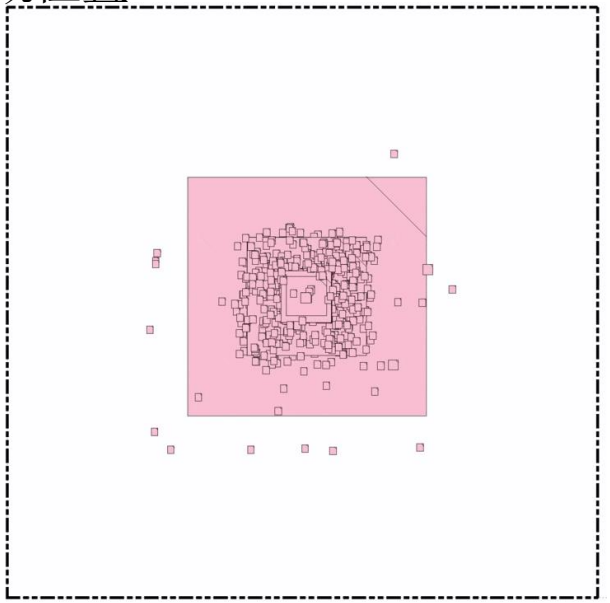
[1] T.-C. Chen, Y.-W. Chang, and S.-C. Lin, “A new multilevel framework for large-scale interconnect-driven floorplanning,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 27, no. 2, pp. 286–294, 2008.

[2] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, “B*-trees: A new representation for non-slicing floorplans,” in Proc. Des. Autom. Conf. (DAC), 2000, pp. 485–463.

基于解析法Floorplanning发展历程

- 使用解析法的Floorplanning一般分为两个阶段。
  - 全局规划阶段，将布局问题建模成数学模型，使用最优化方法计算大概位置。
  - 合法化阶段，使用启发式方法，确定最终满足约束的位置。
- 解析法的发展历程：

名称	模型	合法化	年份
Analytical [1]	bell-shaped smoothing and Density control	function pl2sp() in Parquet-4 [7]	2006
AR [2]	Attractor-Repeller model	SOCP based on relative position matrix	2008
UFO [3]	Push-Pull model	SOCP based on constraint graph	2011
F-FM [4]	placement model based on NTUplace	gradual partition based on position and area Slicing-Tree	2014
Ref. [5]	placement model based on NTUplace	SAINT[8]	2021
PeF[6]	Poisson’ s Equation Base density model	Constraint Graph based legalization	2023

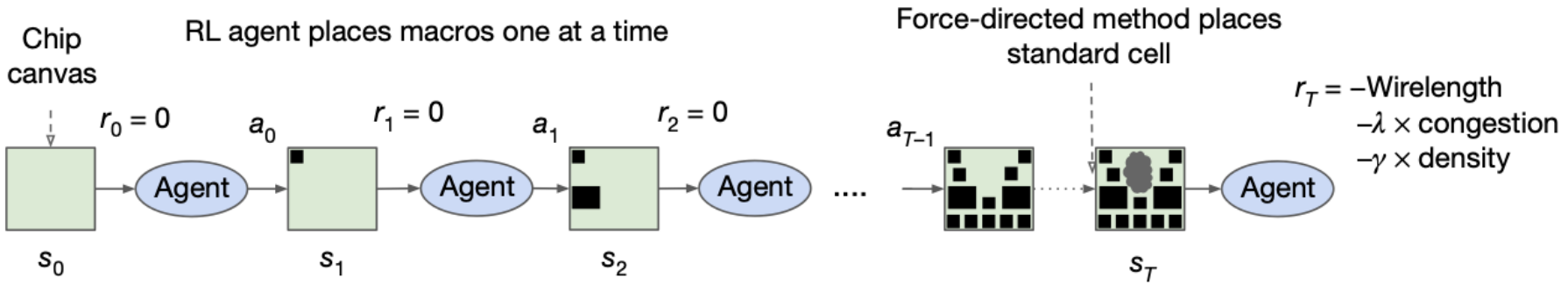


[1] Y. Zhan, Y. Feng, and S. S. Sapatnekar, “A fixed-die floorplanning algorithm using an analytical approach,” in Proc. Asia South Pac. Des. Autom. Conf. (ASP-DAC), 2006, pp. 771–776.
[2] C. Luo, M. F. Anjos, and A. Vannelli, “Large-scale fixed-outline floorplanning design using convex optimization techniques,” in Proc. Asia South Pac. Des. Autom. Conf. (ASP-DAC), 2008, pp. 198–203.
[3] J.-M. Lin and Z.-X. Hung, “UFO: Unified convex optimization algo- rithms for fixed-outline floorplanning considering pre-placed modules,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 30, no. 7, pp. 1034–1044, 2011.
[4] J.-M.LinandJ.-H.Wu,“F-FM:Fixed-outlinefloorplanningmethodolo- gy for mixed-size modules considering voltage-island constraint,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 33, no. 11, pp. 1681–1692, 2014.
[5] J.-M. Lin, T.-T. Chen, H.-Y. Hsieh, Y.-T. Shyu, Y.-J. Chang, and J.-M. Lu, “Thermal-aware fixed-outline floorplanning using analytical models with thermal-force modulation,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 29, no. 5, pp. 985–997, 2021.
[6] X. Li, K. Peng, F. Huang, and W. Zhu, “PeF: Poisson’s equation based large-scale fixed-outline floorplanning,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 42, no. 6, pp. 2002–2015, 2023
[7] S. Adya and I. Markov, “Fixed-outline floorplanning: enabling hierar- chical design,” IEEE Trans. Very Large Scale Integr VLSI Syst., vol. 11, no. 6, pp. 1120–1135, 2003.
[8] J.-M. Lin, P.-Y. Chiu, and Y.-F. Chang, “SAINT: Handling module folding and alignment in fixed-outline floorplans for 3d ics,” in Proc. Int. Conf. Comput.-Aided Design (ICCAD), 2016, pp. 1–7.
[9] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, “NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints,” IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 27, no. 7, pp. 1228–1240, 2008.

基于AI方法宏单元布局

- CT [1] -2021

谷歌提出了一种基于强化学习的宏单元布局方法, 他们将宏单元布局过程视作一种马尔可夫决策过程(MDP)，设计了一个强化学习智能体，在每一个时间步给出一个宏单元的摆放位置，直到所有的宏单元摆放完成。此后，进行快速地布局布线对布图结果进行评估，并计算奖励值反馈给强化学习智能体。神经模型使用了图神经网络作为主干网络，以提取芯片网表中包含的拓扑信息。经过数小时的训练，智能体产生的布图结果在部分评价指标上可以接近或超过商业EDA工具以及专家手工摆放的结果。



[1] MIRHOSEINI A, GOLDIE A, YAZGAN M, et al. A graph placement methodology for fast chip design[J]. Nature, 2021, 594(7862): 207-212.

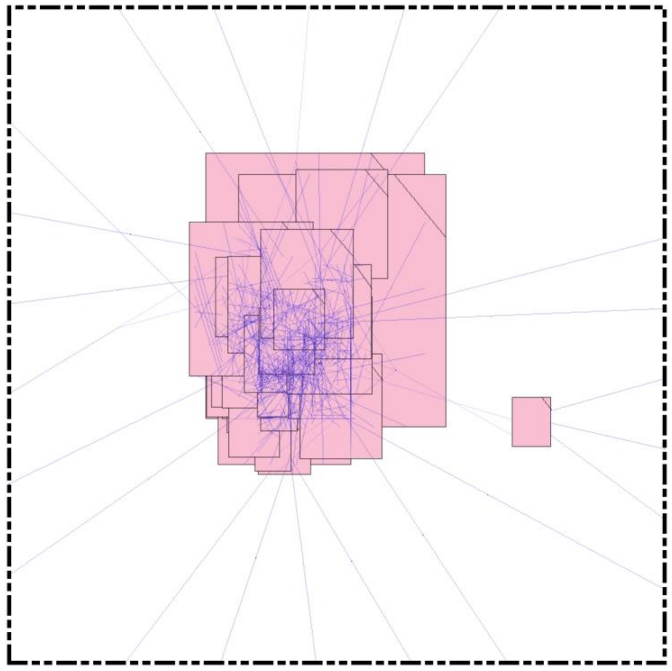
01 研究问题

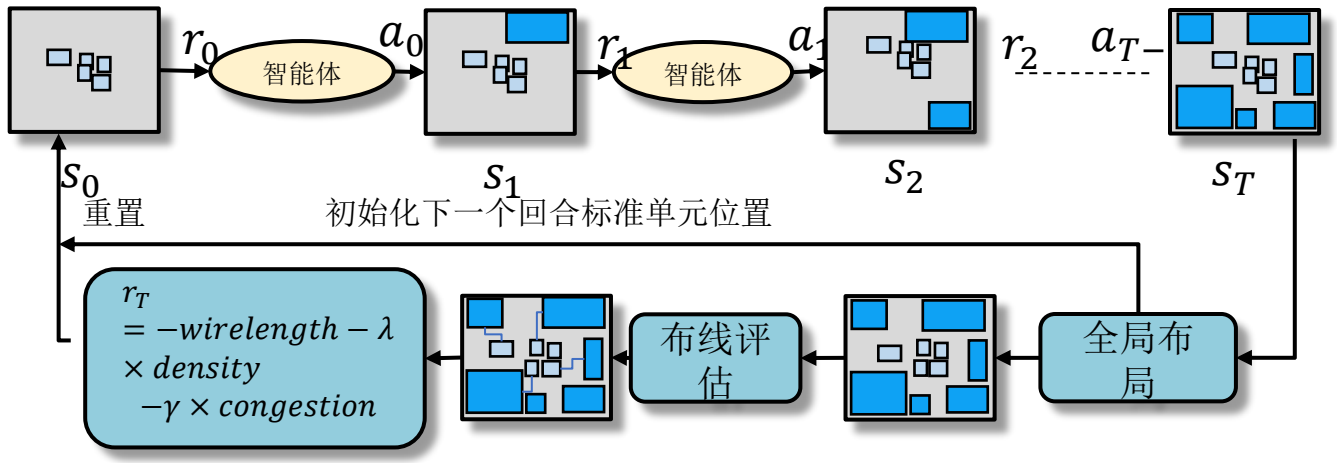
02 研究内容

03 未来展望

未来展望

- 解析法：“Handling Orientation and Aspect Ratio of Modules in Electrostatics-based Large Scale Fixed-Outline Floorplanning”，ICCAD-2023. (accepted)
- AI方法：强化学习，建模成马尔可夫决策过程。





马尔可夫决策过程

总结

- iMP现有的工具
  - 基于序列对的模拟退火
  - 基于B*-tree的模拟退火
- 未来计划
  - 解析法
  - 整合AI方法

iEDA Tutorial 第二期议程

- Part 1 iEDA-iFP/iPDN工具架构、特性与使用（黄增荣）

- Part 2 iEDA-iMP 关键技术（黄富兴）

- **Part 3 iEDA-iPL 问题介绍、架构、使用与规划 （陈仕健）**

- Part 4 iEDA-iPL 关键技术 （邱奕杭）

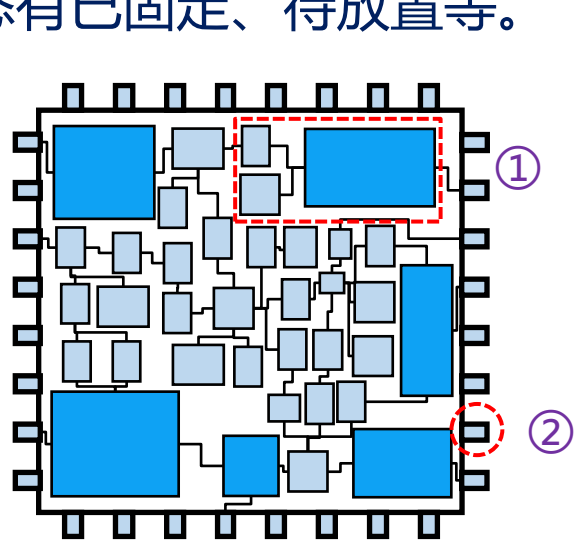
- Part 5 iEDA-iTO工具架构、特性与关键技术 （吴鸿熙）

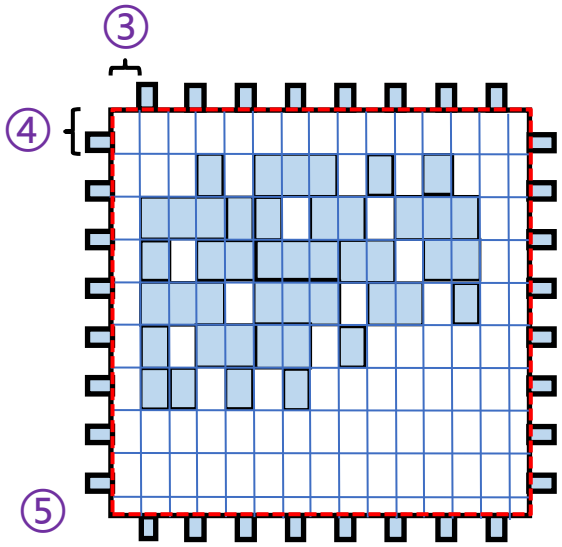
布局问题

- **布局问题介绍**

给定有连接线关系（线网）的单元集合以及布局区域，布局对单元进行放置。一般布局目标是最小化线网总线长/时序/功耗，要求单元满足合法性（在布局区域内、对齐Row/Site、单元相互不重叠等）的规定。

- 线网：单条线网可连接多个Pin点（Pin点位置在单元上或者是在IO处）；线网走线方向只有横纵方向。
- 单元：形状通常是矩形。单元类型有宏单元、标准单元（时序单元、逻辑单元）等；单元状态有已固定、待放置等。



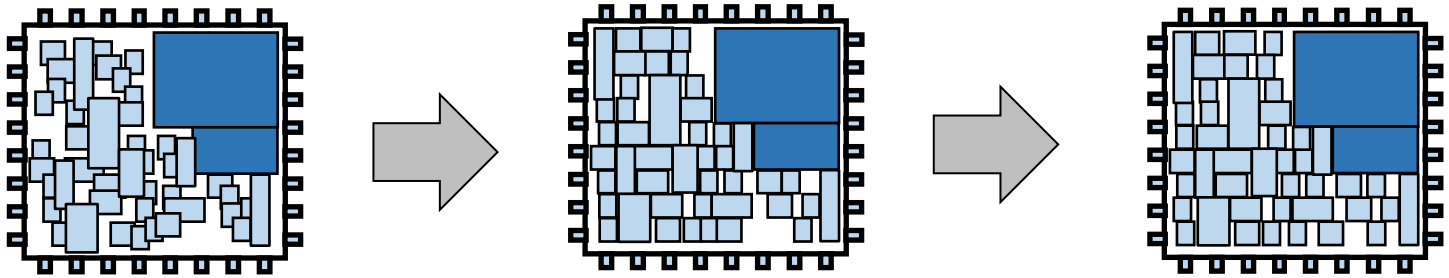


① 单元及其线网
② IO pin
③ Site（布局最小网格）
④ Row（包含若干个Site）
⑤ 布局区域

布局问题

- **布局划分三个子问题**

一步到位求解大规模布局问题是NP难的，因此一般划分为三个阶段：全局布局、合法化和详细布局。



全局布局：单元扩散至合适的位置，忽略单元重叠

合法化：把单元放到行上，消除单元之间重叠

详细布局：局部调整单元，使设计目标更优化

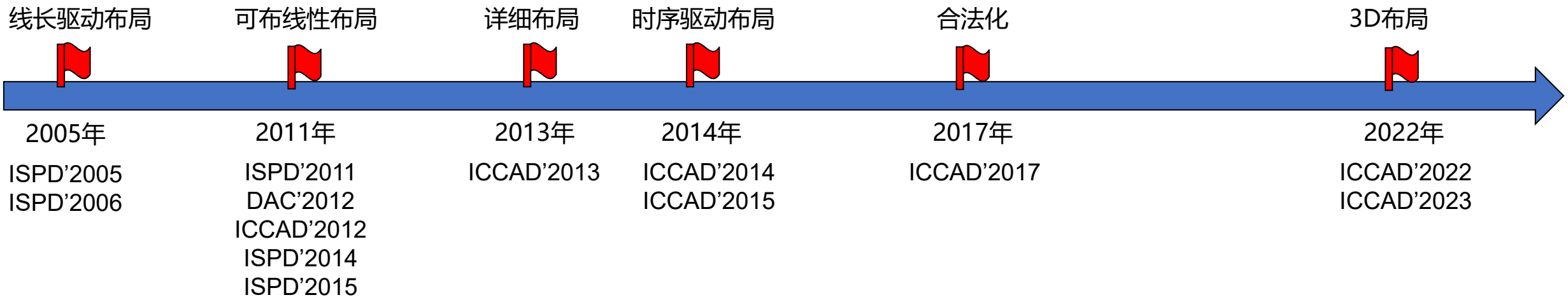
- **布局支持物理设计目标**

物理设计目标PPA（Performance, Power, Area）；布局需考虑一些重要特性

- 可布线性布局
- 时序驱动布局
- Fence Region约束布局
- 混合宏单元布局
- 2.5D/3D布局
- …

布局问题-竞赛问题时间轴

竞赛	题目
ISPD'2005	Placement
ISPD'2006	Placement
ISPD'2011	Routability-Driven Placement
DAC'2012	Routability-Driven Placement
ICCAD'2012	Design Hierarchy Aware Routability-Driven Placement
ICCAD'2013	Detailed Placement
ISPD'2014	Detailed Routing-Driven Placement
ICCAD'2014	Incremental Timing-Driven Placement
ISPD'2015	Blockage-Aware Detailed Routing-Driven Placement
ICCAD'2015	Incremental Timing-Driven Placement
ICCAD'2017	Multi-Deck Standard Cell Legalization
ICCAD'2022	3D Placement with D2D Vertical Connections
ICCAD'2023	3D Placement with Macros

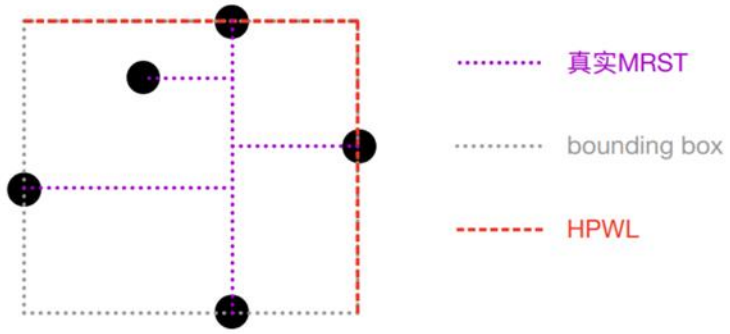


研究内容一：全局布局

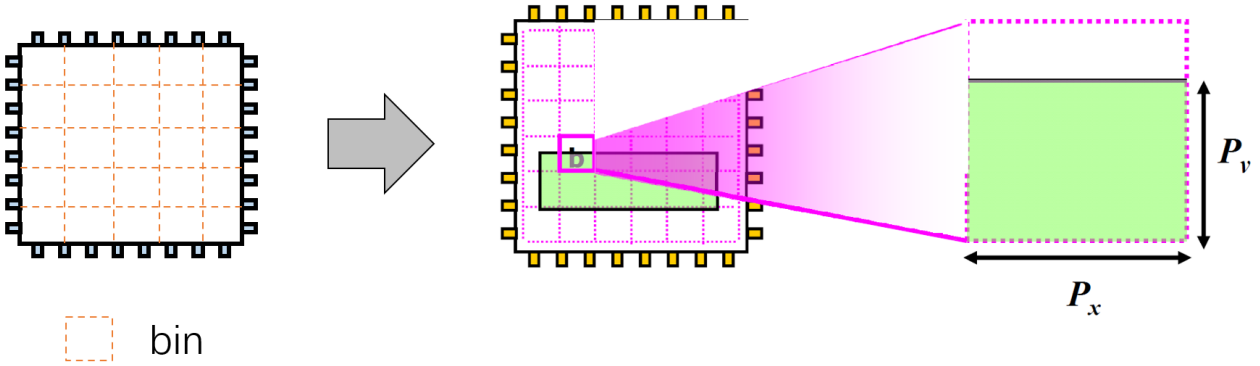
- **全局布局的优化问题形式**

布局目标采用半周长线长（Half-Perimeter Wirelength, HPWL），记为$HPWL_e(v)$，约束是划分网格（Bin）后各网格的密度，记为$\rho_b(v) \le \rho_t, \forall b \in Bin$。

$$\min_{v} \sum_{e \in E} HPWL_e(v) \quad s.t.\ \rho_b(v) \le \rho_t, \forall b \in Bin$$



（半周长线长模型）



（网格划分下的密度模型）

其中，全局布局目标与约束可以使用罚方法或者（增广）拉格朗日方法连接从而转化为无约束的优化问题。

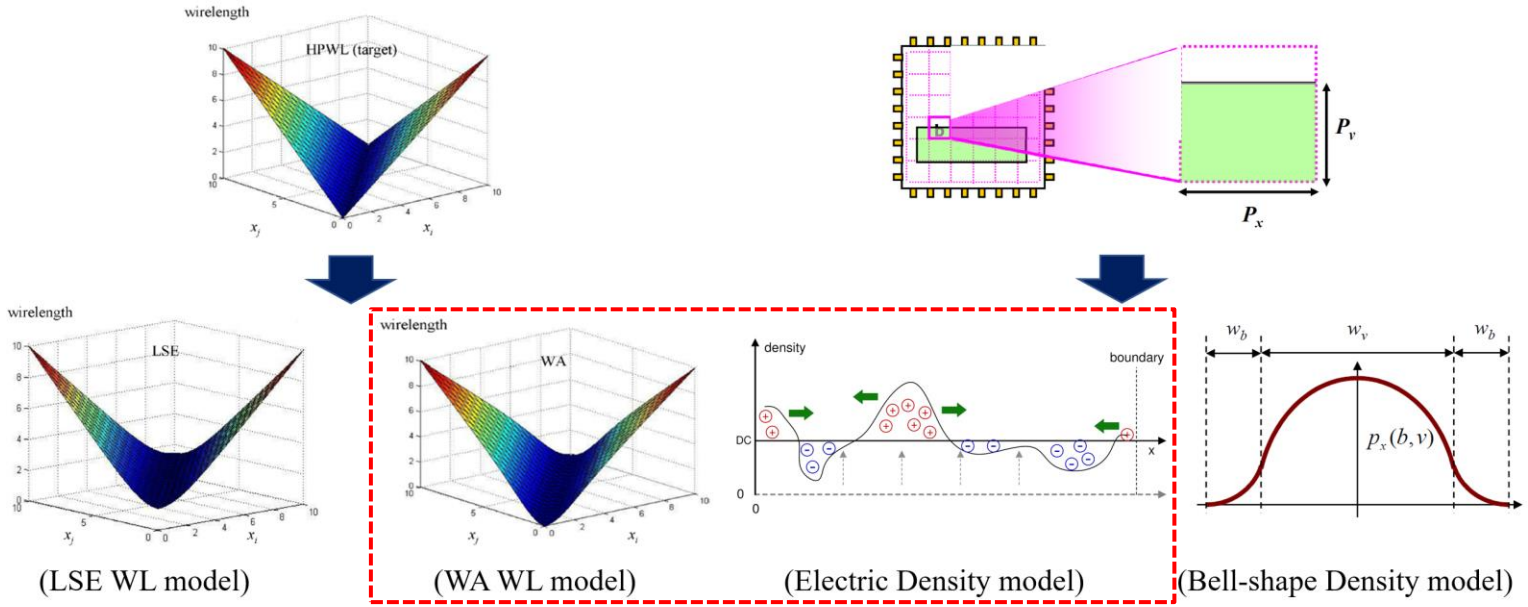
研究内容一：全局布局

- **主流方法使用梯度法进行迭代求解**

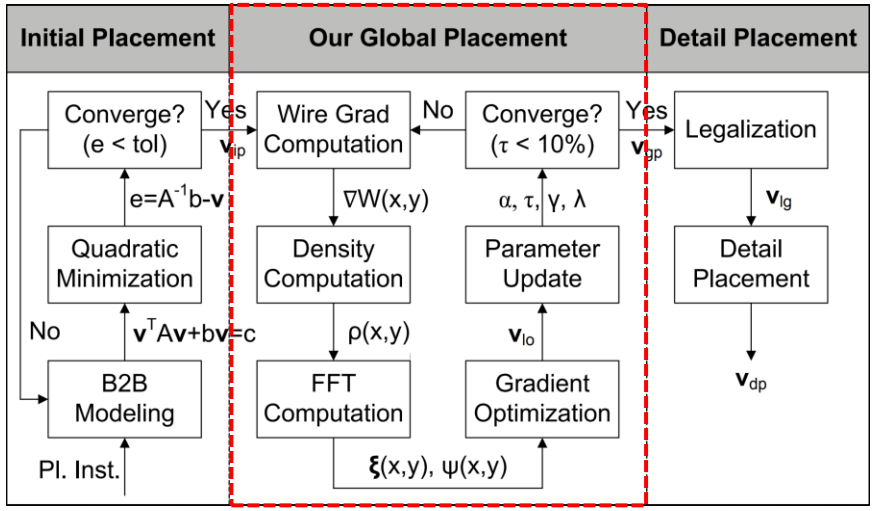
上文已提及的线长、密度模型均是非光滑的，若采用梯度法，需要二者的**平滑化近似**。求得梯度向量后，使用**求解器**（Nesterov、CG、ADMM、SGD等）进行全局布局问题迭代求解。

线长评估平滑化选择

密度评估平滑化选择



(LSE WL model) (WA WL model) (Electric Density model) (Bell-shape Density model)



WA线长模型和电场能模型平滑化近似程度较好

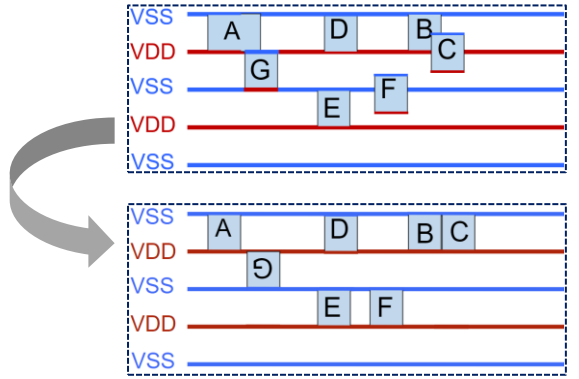
Lu J, Chen P, Chang C C, et al. ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method[J]. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2015, 20(2): 1-34

研究内容二：合法化与详细布局

- **合法化**

全局布局后所有可移动单元（通常等高Row，可能存在多倍高）需要放置在合法的Row/Site位置上，方向对齐Power Rail，且单元互相之间不重叠。合法化思路是尽可能保留全局布局结果，到达合法结果单元的总移动量最小。可行方法：

- 划分子单元（等高等宽），求解min-cost-max-flow问题
- 两阶段法，单元先分配到行上，在所建图上寻找最短路径
- Tetris、Abacus，QP，LP



（单元合法化）

- **详细布局**

详细布局在保持合法化结果下

进行指标（线长、时序、可布线性）的局部优化。可行方法：

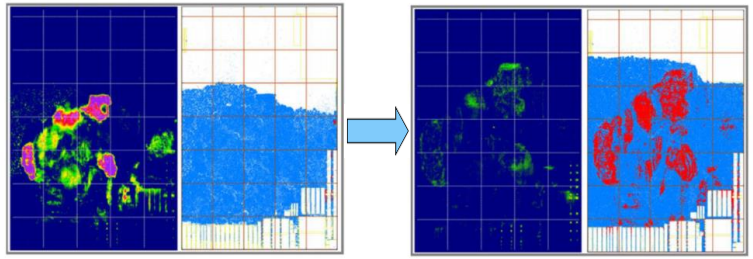
- 行内Shift
- 单元交换，选取独立集进行匹配
- 求解局部少量单元的最优分支定界结果

研究内容三：可布线性布局

研究动机

传统布局忽略单元位置带来的布线需求和布线资源的相对关系不匹配，导致后续布线困难。在布局阶段考虑可布线性十分重要，然而存在以下挑战：

- 布局前期迭代单元位置变动大，可布线性评估不准确
- 可布线性评估方法难以兼顾性能和精度，容易误导布局优化
- 可布线性优化可能导致单元散开，带来线长增加和时序收敛问题



(a) 线长驱动的全局布局结果　(b) 可布线性驱动的全局布局结果

(可布线性布局)

问题描述及方法

可布线性布局的常见评价指标是总溢出值（Total Overflow, TOF）和最大溢出值（Maximum Overflow, MOF）。根据当前单元的位置，采用特定的可布线性评估方法，可获得预估的布线拥塞图，布局根据拥塞图进行单元位置调整以满足可布线性要求。可布线性布局的内容包括：

- 可布线性评估。根据评估方式的不同，可分为静态法、概率法、构建法和网络法共四类方法。每类方法存在评估性能和精度的权衡。静态法采用简单指标如引脚密度表征拥塞，概率法基于布线模式如L-shaped布线表征拥塞，构建法调用布线器表征拥塞，网络法则训练神经网络模型预测拥塞。目前常用的方法是概率法和构建法。
- 可布线性优化。根据优化方式的不同，可分为调整密度间接优化拥塞、直接整合拥塞项参与优化这两种方法。第一种典型方法如单元膨胀，通过减小拥塞网格的单元密度进行优化。第二种则通过建立可微模型进行梯度优化。

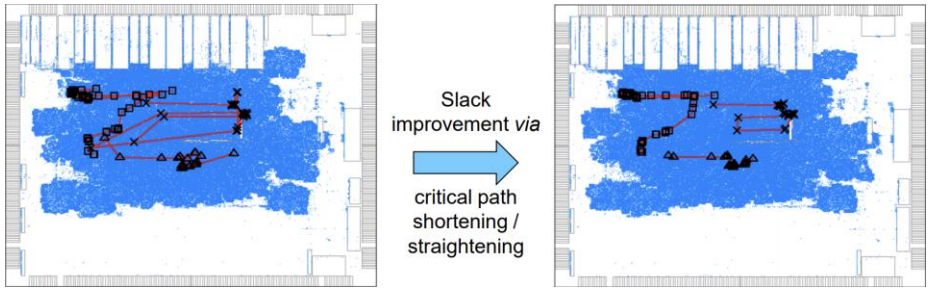
N. Viswanathan, et al. The ISPD-2011 routability-driven placement contest and benchmark suite[C]. ISPD, 2011.

研究内容四：时序驱动布局

研究动机

传统布局忽略一些对时序有关键影响的线网的重要性，导致后续时序收敛困难。在布局阶段考虑时序十分重要，然而存在以下挑战：

- 布局前期迭代单元位置变动大，时序评估不准确
- 修复一些关键线网可能会导致新的时序违例线网出现
- 时序优化可能导致单元聚集，影响可布线性



（时序驱动布局）

问题描述及方法

时序驱动布局的重要评价指标是总和负裕量（Total Negative Slack, TNS）和最差负裕量（Worst Negative Slack, WNS）。根据当前单元的位置构建绕线，结合必要的电容电阻信息给静态时序分析工具，布局能够获得当前状态的时序信息。时序驱动布局方法的分类有：

- 基于线网的方法。标注时序关键路径信息到线网层级从而赋权关键线网，该方法能够方便适配所有线长驱动布局。由于线网与路径是一对多的关系，线网赋权能够有效缩短关键路径，同时可能影响一些非关键的路径。
- 基于路径的方法。根据时序路径信息在布局新增时序约束项，通过求解数学规划的方法更新单元的位置。该方法能够有效减少目标违例，然而数学规划的求解引入了额外的计算时间。
- 可微的方法。通过时序分析传播过程的平滑化处理计算TNS、WNS的梯度，时序信息加入全局布局的目标。

Kim M C, Hu J, Li J, et al. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite[C] IEEE, 2015: 921-926.

研究内容五：Region约束布局

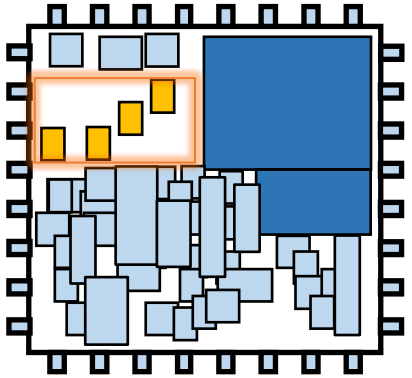
- **研究动机**

现代布局需求隔离电压区域放置特定单元以达到提升性能的目的。

- **问题描述**

Region区域（通常由一个或多个矩形子区域组成）排他地放置指定单元，其余单元不能放置在该区域内。Region约束需在全局布局阶段考虑，否则后续的合法化和详细布局会导致质量严重损失。

- **可行方法**
  - NTUplace4dr: 单元聚类、新的线长模型，新增虚拟线网
  - Eh?Placer、RippleDR: Look-ahead合法化、上下界优化
  - DREAMPlace3.0：多电场系统布局



（Region约束布局）

研究内容六：2.5D/3D布局

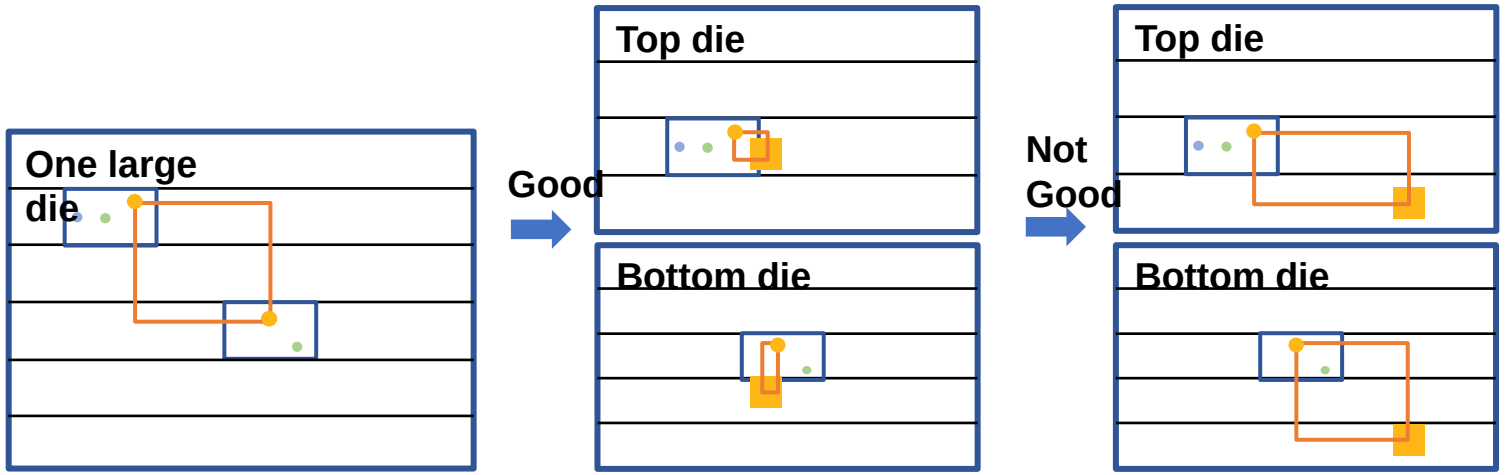
- **研究动机**

3D集成方法被认为有望进一步解决互联瓶颈，而3D中的布局对于3D集成质量的影响至关重要，因为其不仅需要决定水平位置，还需要决定所处的层。

- **问题描述**

在3D集成场景下，不同的die可能使用不同的工艺，同一个单元位于不同层时的尺寸有所区别，这对于划分来说引入了新的约束。同时，一个net中的单元可能属于不同的die，中间层通过Terminal进行连接，而这种线网模型也对布局提出了新的挑战

- **可行方法**
  - 双层规划
  - 网表划分
  - MIV合法化



其他研究内容

- 新的密度表达和计算（函数，方程或网络）
- 联合优化cell location/size和buffer
- 精准控制物理变量
- 可微的优化指标（WL, Timing, Congestion）
- 二阶优化方法
- 优化方向学习
- 拥塞估计
- 时序估计
- DRC估计
- 功耗估计
- 硬件加速技术

01 研究内容

02 研究进展

03 未来计划

工具进展：iPL工具原型

- **iPL 1.0完成了布局的基础功能**

  当前已完成功能：

  - 数据读入支持从iDB提取和处理布局相关数据；配置文件支持json读入
  - 满足布局流程需求
    - ✓ 全局布局：支持HPWL、STWL、WAWL线长+电场密度（梯度）评估，使用Nesterov法布局
    - ✓ 合法化：使用Abacus方法，支持完整和增量式合法化模式
    - ✓ 详细布局：支持单元单行Shift（行内最优化布局）、单元全局/垂直交换、单元局部重排序、行内移动消除峰值密度区域
    - ✓ Filler填充：支持指定filler类型填充布局区域的空白
    - ✓ 布局检查：单元放置在布局区域内、对齐Row/Site、对齐Power rail、单元是否有重叠

工具进展：iPL工具原型

- **iPL 1.0拓展布局与iEDA工具链的交互**

  当前已完成功能：

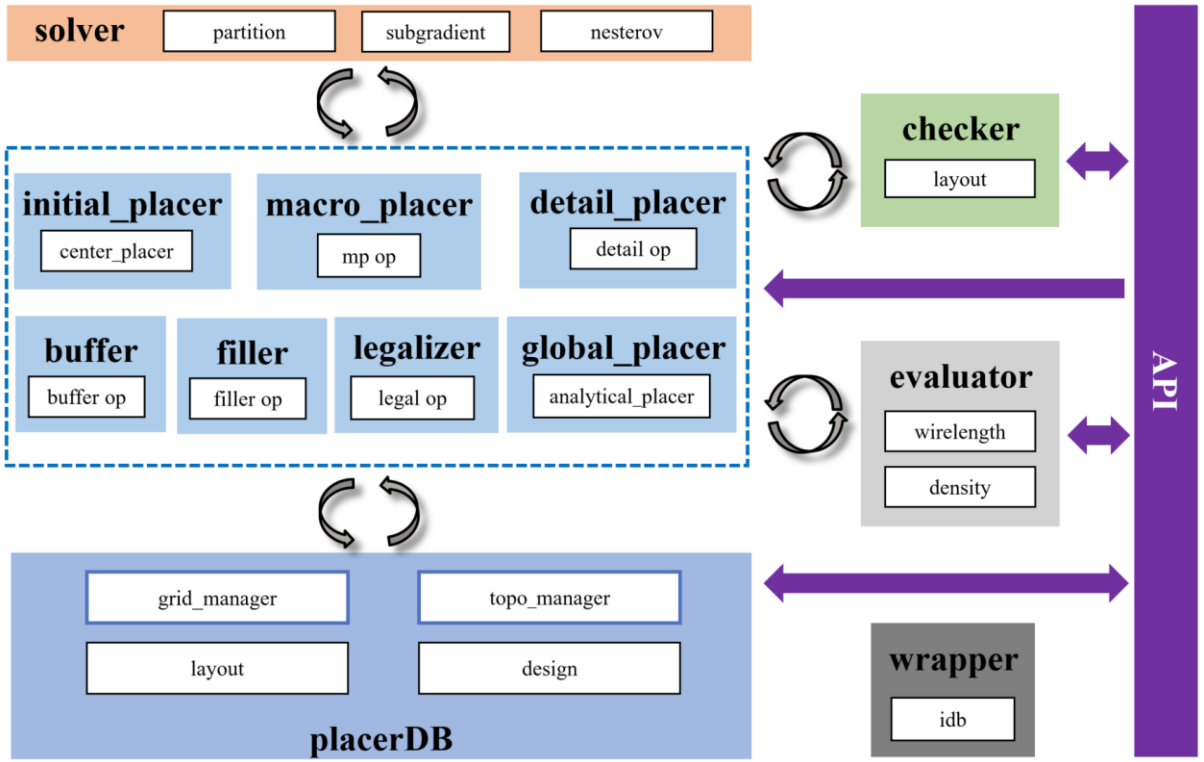
  - 调用eGR（Early Global Routing）对布局状态的初步绕线评估
  - 根据预绕线信息构建RC树，使用iSTA评估当前布局状态的时序信息
  - 支持指定区域返回未被占用的合法空间集合
  - 支持新增单元读入并进行增量式合法化
  - 支持在设计中插入Buffer
  - 待接入：可布线性布局
  - 待接入：时序驱动布局

iPL工具结构

- **结构设计**

瞄准代码功能可读性、可扩展性、易用性目标，工具成熟后再聚焦在性能上。

- ✓ **PlacerDB模块**：封装并维护布局所需的版图数据（layout）和设计数据（Design）
- ✓ **Operator模块**：提取布局数据进行操作，期间调用solver模块进行求解，evaluator模块评估指标，checker模块检查布局结果
- ✓ **Solver模块**：集合一些成熟的求解工具辅助布局
- ✓ **Checker模块**：对当前布局进行违例检查、功能检测、报告输出
- ✓ **Evaluator模块**：评估当前布局指标
- ✓ **Wrapper模块**：从数据源读取布局数据，布局结果写回数据源
- ✓ **API**：iPL与外部的交互接口



iPL工具和DREAMPlace ,OpenROAD-pl对比

- 与DREAMPlace对比
  - 全局布局结果指标线长在ISPD2019测试例子上相近，运行速度平均慢4-6倍（数据结构导致的转换开销）
  - iPL暂未支持Region约束布局、详细布局部分未充分并行
  - iPL支持与iEDA工具链交互、包括评估器（时序/可布线性）、Buffer插入、增量式合法化、布线后插入Filler等
- 与OpenROAD-pl对比
  - 全局布局结果指标相近，iPL合法化和详细布局的结果较好
  - iPL的文件、功能组织有层次，拓展性好
  - iPL与工具链交互的接口丰富，能够方便定制化功能需求，如调用STA时可以限制传播的级数快速评估时序

iPL工具的API

Method	SubMethod	Type	Argument	Return	Description
initAPI	*	action	pl_json_path, idb_builder	self	初始化iPL
runFlow	runGP	action	void	self	运行全局布局
	runBufferInsertion	action	void	self	运行Buffer插入
	runLG	action	void	self	运行合法化
	runDP	action	void	self	运行详细布局
	writeBackSourceDataBase	action	void	self	布局数据写回数据源
runIncrLG	*	action	inst_list	self	运行增量式合法化
updatePlacerDB	*	action	void/inst_list	self	从数据源更新（指定）数据
obtainAvailable WhiteSpaceList	*	action	row_range, site_range	rectangle list	获取指定区域下的可用布局空间（供插入单元使用）
checkLegality	*	accessor	void	bool	检查当前布局结果的合法性
isSTAStarted	*	accessor	void	bool	检查是否已启动STA
isPlacerDBStarted	*	accessor	void	bool	检查是否已初始化PlacerDB
isAbucasLGStarted	*	accessor	void	bool	检查是否已启动Abucas合法器

iPL工具的API

Method	SubMethod	Type	Argument	Return	Description
reportPLInfo	reportHPWLInfo	accessor	feed	self	报告HPWL信息
	reportSTWLInfo	accessor	feed	self	报告STWL信息
	reportLongNetInfo	accessor	feed	self	报告长线网信息
	reportLayoutInfo	accessor	feed	self	报告版图违例信息
	reportPeakBinDensity	accessor	feed	self	报告Bin区域的峰值密度
	reportTimingInfo	accessor	feed	self	报告布局时序信息
	reportCongestionInfo	accessor	feed	self	报告布局拥塞信息
obtainTimingInfo	obtainPinEarly(Late)Slack	action	pin_name	value	获取Pin上的Slack信息
	obtainPinEarly(Late)ArrivalTime	action	pin_name	value	获取Pin上的ArrivalTime信息
	obtainPinEarly(Late)RequiredTime	action	pin_name	value	获取Pin上的RequiredTime信息
	obtainWNS/TNS	action	clk_name	value	获取WNS/TNS信息
	updateTiming	action	void	self	更新时序评估
obtainCongesionInfo	obtainPinDens	action	void	value	获取Pin Density信息
	obtainNetCong	action	rudy_type	value	获取线网拥塞信息
	evalGRCong	action	void	value	更新拥塞评估

iPL工具的配置参数

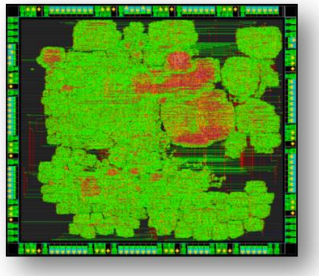
- JSON格式文件

JSON参数	功能说明	参数范围	默认值
is_max_length_opt	是否开启最大线长优化	[0,1]	0
max_length_constraint	指定最大线长	[0-1000000]	1000000
is_timing_aware_mode	是否开启时序模式	[0,1]	0
ignore_net_degree	忽略超过指定pin个数的线网	[10-10000]	100
num_threads	指定的CPU线程数	[1-64]	8
[GP-Wirelength] init_wirelength_coef	设置初始线长系数	[0.0-1.0]	0.25
[GP-Wirelength] reference_hpwl	调整密度惩罚的参考线长	[100-1000000]	446000000
[GP-Wirelength] min_wirelength_force_bar	控制线长边界	[-1000-0]	-300
[GP-Density] target_density	指定的目标密度	[0.0-1.0]	0.8
[GP-Density] bin_cnt_x	指定水平方向上Bin的个数	[16,32,64,128,256,512,1024]	512
[GP-Density] bin_cnt_y	指定垂直方向上Bin的个数	[16,32,64,128,256,512,1024]	512
[GP-Nesterov] max_iter	指定最大的迭代次数	[50-2000]	2000
[GP-Nesterov] max_backtrack	指定最大的回溯次数	[0-100]	10
[GP-Nesterov] init_density_penalty	指定初始状态的密度惩罚	[0.0-1.0]	0.00008
[GP-Nesterov] target_overflow	指定目标的溢出值	[0.0-1.0]	0.1
[GP-Nesterov] initial_prev_coordi_update_coef	初始扰动坐标时的系数	[10-10000]	100
[GP-Nesterov] min_precondition	设置precondition的最小值	[1-100]	1
[GP-Nesterov] min_phi_coef	设置最小的phi参数	[0.0-1.0]	0.95
[GP-Nesterov] max_phi_coef	设置最大的phi参数	[0.0-1.0]	1.05
[BUFFER] max_buffer_num	指定限制最大buffer插入个数	[0-1000000]	35000
[BUFFER] buffer_type	指定可插入的buffer类型名字	工艺相关	列表[...,...]
[LG] max_displacement	指定单元的最大移动量	[10000-1000000]	50000
[LG] global_right_padding	指定单元间的间距（以Site为单位）	[0,1,2,3,4...]	1
[DP] max_displacement	指定单元的最大移动量	[10000-1000000]	50000
[DP] global_right_padding	指定单元间的间距（以Site为单位）	[0,1,2,3,4...]	1
[Filler] first_iter	指定第一轮迭代使用的Filler	工艺相关	列表[...,...]
[Filler] second_iter	指定第二轮迭代使用的Filler	工艺相关	列表[...,...]
[Filler] min_filler_width	指定Filler的最小宽度（以Site为单位）	工艺相关	1
[MP] fixed_macro	指定固定的宏单元 (string macro_name)	设计相关	列表[...,...]
[MP] fixed_macro_coordinate	指定固定宏单元的位置坐标（int location_x, int location_y）	设计相关	列表[...,...]
[MP] blockage	指定宏单元阻塞矩形区域，宏单元应该避免摆放在该区域（i	设计相关	列表[...,...]
[MP] guidance_macro	指定指导摆放宏单元，每个宏单元可以设置期望摆放的区域	设计相关	列表[...,...]
[MP] guidance	指定对应宏单元的指导摆放区域（int left_bottom_x, int left_	设计相关	列表[...,...]
[MP] solution_type	指定解的表示方式	["BStarTree","SequencePair"]	"BStarTree"
[MP] perturb_per_step	指定模拟退火每步扰动次数	[10-1000]	100
[MP] cool_rate	指定模拟退火温度冷却率	[0.0-1.0]	0.92
[MP] parts	指定标准单元划分数（int）	[10-100]	66
[MP] ufactor	指定标准单元划分不平衡值 (int)	[10-1000]	100
[MP] new_macro_density	指定虚拟宏单元密度	[0.0-1.0]	0.6
[MP] halo_x	指定宏单元的halo（横向）	[0-1000000]	0
[MP] halo_y	指定宏单元的halo（纵向）	[0-1000000]	0
[MP] output_path	指定输出文件路径		"./result/pl"

iPL工具报告输出

- iPL支持一生一芯流片

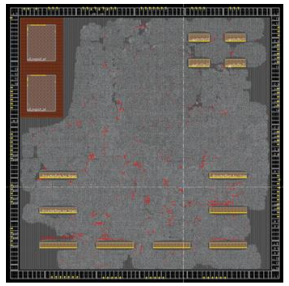
2022-02-02，1st Tapeout



110nm node，0.7M gates, 25MHz
(5-level pipeline，IP:Chiplink、UART、SPI)

Macro, Multi-clock,
Scale increasing,
Auto-design

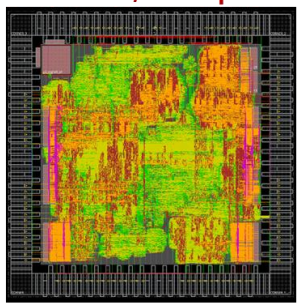
2022-08-12，2nd Tapeout



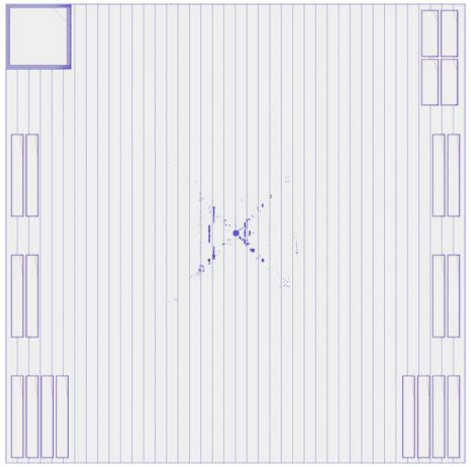
110nm node，1.5M gates
(11-level pipeline with cache、IP:
UART、VGA、PS/2、SPI、SDRAM,
Two PLL on SoC，Support Linux)

110nm -> 28nm

2023-01-04，3rd Tapeout



28nm node，1.5M gates
(11-level pipeline with cache、IP:
UART、VGA、PS/2、SPI、SDRAM,
Two PLL on SoC，Support Linux)



- iPL的终端输出

```
I0730 17:36:15.176146 1203780 PlacerDB.cc:303] Design name : gcd
I0730 17:36:15.176285 1203780 PlacerDB.cc:304] Database unit : 1000
I0730 17:36:15.176290 1203780 PlacerDB.cc:305] Die rectangle : 0,0 279960,280128
I0730 17:36:15.176292 1203780 PlacerDB.cc:307] Core rectangle : 9600,9990 269760,269730
I0730 17:36:15.176295 1203780 PlacerDB.cc:309] Row height : 3330
I0730 17:36:15.176296 1203780 PlacerDB.cc:310] Site width : 480
I0730 17:36:15.176301 1203780 PlacerDB.cc:348] Core area : 67573958400
I0730 17:36:15.176304 1203780 PlacerDB.cc:349] Non place instance area : 249350400
I0730 17:36:15.176307 1203780 PlacerDB.cc:350] Place instance area : 6638155200
I0730 17:36:15.176309 1203780 PlacerDB.cc:353] Utilization(%) : 9.85992
I0730 17:36:15.176327 1203780 PlacerDB.cc:468] Instances Num : 795
I0730 17:36:15.176330 1203780 PlacerDB.cc:469] 1. Macro Num : 0
I0730 17:36:15.176332 1203780 PlacerDB.cc:470] 2. Stdcell Num : 795
I0730 17:36:15.176337 1203780 PlacerDB.cc:471] 2.1 Logic Instances : 274
I0730 17:36:15.176340 1203780 PlacerDB.cc:472] 2.2 Flipflops : 0
I0730 17:36:15.176344 1203780 PlacerDB.cc:473] 2.3 Clock Buffers : 0
I0730 17:36:15.176347 1203780 PlacerDB.cc:474] 2.4 Logic Buffers : 0
I0730 17:36:15.176350 1203780 PlacerDB.cc:475] 2.5 IO Cells : 365
I0730 17:36:15.176353 1203780 PlacerDB.cc:476] 2.6 Physical Instances : 156
I0730 17:36:15.176357 1203780 PlacerDB.cc:477] Core Outside Instances : 0
I0730 17:36:15.176359 1203780 PlacerDB.cc:478] Fake Instances : 0
I0730 17:36:15.176363 1203780 PlacerDB.cc:479] Unplaced Instances Num : 639
I0730 17:36:15.176366 1203780 PlacerDB.cc:480] Placed Instances Num : 0
I0730 17:36:15.176369 1203780 PlacerDB.cc:481] Fixed Instances Num : 156
I0730 17:36:15.176373 1203780 PlacerDB.cc:482] Optional CellMaster Num : 418
I0730 17:36:15.176383 1203780 PlacerDB.cc:523] Nets Num : 675
I0730 17:36:15.176385 1203780 PlacerDB.cc:524] 1. ClockNets Num : 1
I0730 17:36:15.176388 1203780 PlacerDB.cc:525] 2. ResetNets Num : 0
I0730 17:36:15.176393 1203780 PlacerDB.cc:526] 3. SignalNets Num : 674
I0730 17:36:15.176394 1203780 PlacerDB.cc:527] 4. FakeNets Num : 0
I0730 17:36:15.176398 1203780 PlacerDB.cc:528] Don't Care Net Num : 0
```

（布局设计信息）

```
I0730 17:36:15.264536 1203780 NesterovPlace.cc:447] ----------------Start Global Placement----------------
I0730 17:36:15.276017 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 1 overflow: 0.926664 HPWL: 12609320
I0730 17:36:15.305334 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 10 overflow: 0.668487 HPWL: 14180134
I0730 17:36:15.337035 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 20 overflow: 0.657688 HPWL: 13699254
I0730 17:36:15.368213 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 30 overflow: 0.654918 HPWL: 13563537
I0730 17:36:15.397320 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 40 overflow: 0.655028 HPWL: 13493288
I0730 17:36:15.426545 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 50 overflow: 0.649733 HPWL: 13460770
I0730 17:36:16.613148 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 450 overflow: 0.150503 HPWL: 14113866
I0730 17:36:16.641381 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 460 overflow: 0.130303 HPWL: 14172192
I0730 17:36:16.669582 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 470 overflow: 0.11647 HPWL: 14252435
I0730 17:36:16.698113 1203780 NesterovPlace.cc:1217] [NesterovSolve] Iter: 480 overflow: 0.10064 HPWL: 14335239
I0730 17:36:16.701102 1203780 NesterovPlace.cc:1305] [NesterovSolve] Finished with Overflow:0.0993077 HPWL : 14343090
I0730 17:36:16.703059 1203780 NesterovPlace.cc:459] Global Placement Total Time Elapsed: 1.43847s
I0730 17:36:16.703070 1203780 NesterovPlace.cc:460] ----------------Finish Global Placement----------------
I0730 17:36:16.722851 1203780 Report.cc:686] Current Stage Total HPWL: 14343090

I0730 17:36:16.754271 1203780 Report.cc:35] ----------------Start iPL Report Generation----------------
I0730 17:36:16.754352 1203780 Report.cc:114] Detect Core outside Instances...
I0730 17:36:16.754364 1203780 Report.cc:126] Detect Instances' Alignment...
I0730 17:36:16.754375 1203780 Report.cc:138] Detect Power Alignment...
I0730 17:36:16.754379 1203780 Report.cc:150] Detect Overlap Between Instances...
I0730 17:36:16.754525 1203780 Report.cc:49] Violation Info Write to './result/pl/report/violation_record.txt'
I0730 17:36:22.382812 1203780 Report.cc:63] Wirelength Info Write to './result/pl/report/wirelength_record.txt'
I0730 17:36:22.383855 1203780 Report.cc:75] Density Info Write to './result/pl/report/density_record.txt'
```

（全局布局输出、布局检查）

```
I0730 17:36:16.723914 1203780 AbacusLegalizer.cc:415] ----------------Start Legalization----------------
I0730 17:36:16.733163 1203780 AbacusLegalizer.cc:422] Total Movement: 779548
I0730 17:36:16.733968 1203780 AbacusLegalizer.cc:431] Legalization Total Time Elapsed: 0.01s
I0730 17:36:16.733983 1203780 AbacusLegalizer.cc:432] ----------------Finish Legalization----------------
I0730 17:36:16.733985 1203780 Report.cc:686] Current Stage Total HPWL: 14631748

I0730 17:36:16.737555 1203780 DetailPlacer.cc:471] ----------------Start Detail Placement----------------
I0730 17:36:16.737598 1203780 DetailPlacer.cc:474] Execution Origin Instance Shift:
I0730 17:36:16.738569 1203780 DetailPlacer.cc:478] After RowOpt HPWL: 14546764
I0730 17:36:16.738711 1203780 DetailPlacer.cc:489] Execution Swap Iteration: 0
I0730 17:36:16.741622 1203780 DetailPlacer.cc:494] ---After Global Swap HPWL: 14494797
I0730 17:36:16.743422 1203780 DetailPlacer.cc:500] ---After Vertical Swap HPWL: 14473173
I0730 17:36:16.743824 1203780 DetailPlacer.cc:507] ---After Local Reorder HPWL: 14462013
I0730 17:36:16.745075 1203780 DetailPlacer.cc:524] ---After Row Opt HPWL: 14429151
I0730 17:36:16.745206 1203780 DetailPlacer.cc:489] Execution Swap Iteration: 1
I0730 17:36:16.747771 1203780 DetailPlacer.cc:494] ---After Global Swap HPWL: 14423772
I0730 17:36:16.749518 1203780 DetailPlacer.cc:500] ---After Vertical Swap HPWL: 14414569
I0730 17:36:16.749894 1203780 DetailPlacer.cc:507] ---After Local Reorder HPWL: 14413603
I0730 17:36:16.751096 1203780 DetailPlacer.cc:524] ---After Row Opt HPWL: 14416928
I0730 17:36:16.751260 1203780 DetailPlacer.cc:535] Execution Final Instance Shift Iteration: 0
I0730 17:36:16.752424 1203780 DetailPlacer.cc:545] ---After RowOpt HPWL: 14417052
I0730 17:36:16.753118 1203780 DetailPlacer.cc:556] Detail Plaement Total Time Elapsed: 0.015518s
I0730 17:36:16.753130 1203780 DetailPlacer.cc:557] ----------------Finish Detail Placement----------------
I0730 17:36:16.754197 1203780 Report.cc:686] Current Stage Total HPWL: 14417052
```

（合法化、详细布局）

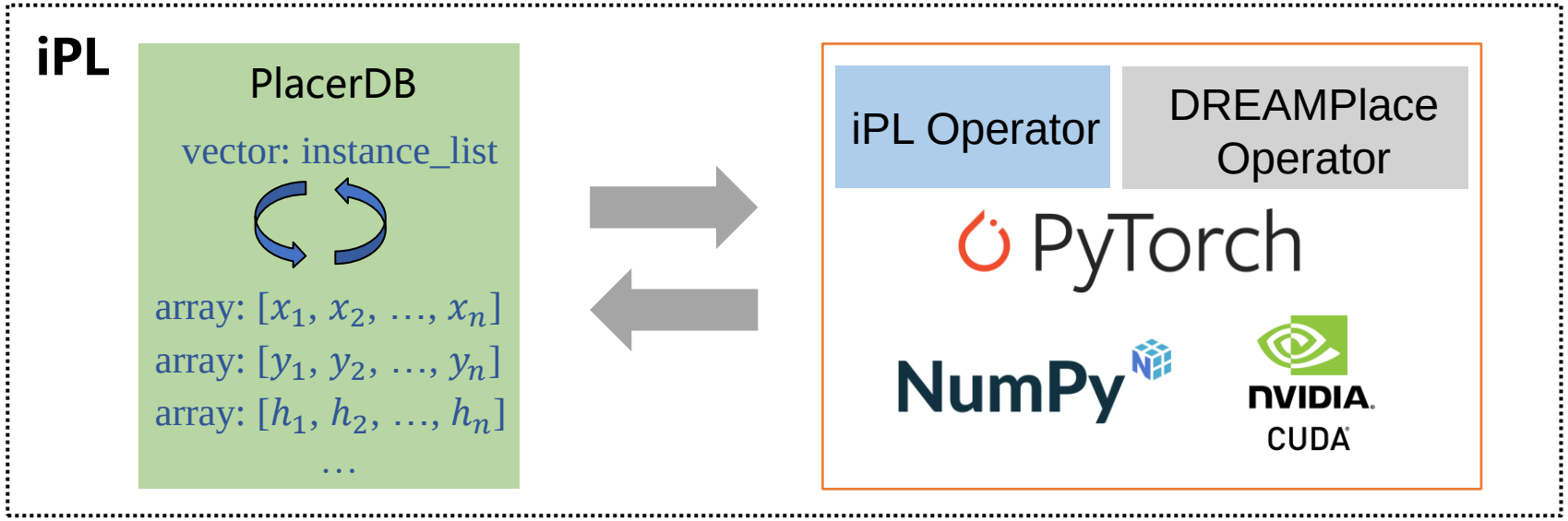
01 研究内容

02 研究进展

03 未来计划

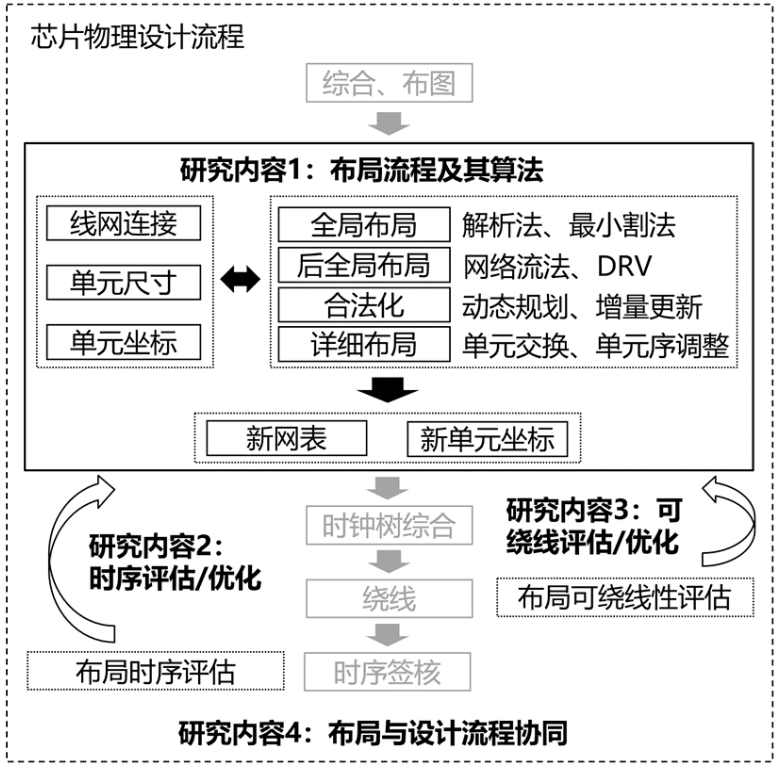
工具计划一：架构改进，性能提升

- **改进工具架构设计**
  - 组织布局数据排列方式（向量/矩阵），避免不必要的转换开销
  - 适配更多布局方法，支持已有开源工具/代码接入iPL布局流程
  - 工具接入python、torch、CUDA使用丰富的工具包



工具计划二：基于需求驱动迭代工具

- **一生一芯使用iEDA工具链，iPL协同芯片物理设计流程**
  - iPL提供物理综合信息，优化网表
  - iPL协同iMP（宏单元布局工具）评估标准单元位置
  - iPL提供可布线性和时序驱动布局
  - iPL改进布局的密度表达、优化方法
  - iPL协同iCTS（时钟树综合工具）放置时钟单元
  - iPL协同iTO（时序优化工具）进行单元调整/插入
  - iPL协同iRT（绕线工具）修正单元位置
  - …



总结

- **iPL是立足于iEDA物理设计流程的布局工具**
  - 布局问题规模大，性能与（局部）关键单元放置需要重点考虑
  - 布局预考虑指标（可布线性/时序/功耗）对物理设计最终PPA的影响需要评估
  - iPL参与一生一芯流片，可以发现布局的需求
  - iPL原型提供充足的API，支持协同物理设计流程的部分功能
  - iPL存在布局方法/特性不够丰富，存在某些corner case未考虑的情况
  - 欢迎各方面的指导/帮助/贡献

iEDA Tutorial 第二期议程

- Part 1 iEDA-iFP/iPDN工具架构、特性与使用（黄增荣）

- Part 2 iEDA-iMP 关键技术（黄富兴）

- Part 3 iEDA-iPL 问题介绍、架构、使用与规划 （陈仕健）

- **Part 4 iEDA-iPL 关键技术 （邱奕杭）**

- Part 5 iEDA-iTO工具架构、特性与关键技术 （吴鸿熙）

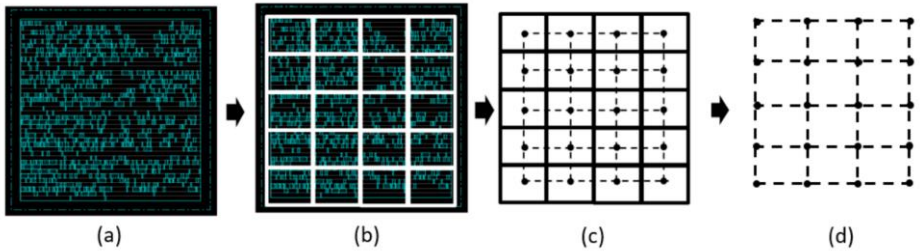
01 可布线性布局

02 时序驱动布局

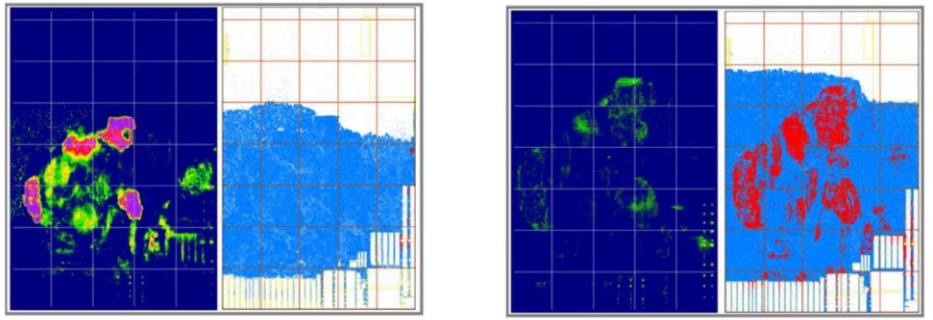
03 未来研究计划

可布线性布局：研究意义

- **可布线性与布局的关系**
  - 布局决定单元位置，影响线网走线导致布线图每条边的demand和capacity的值不相匹配
  - 可布线性布局通过预估demand和capacity的分布并主动调整单元位置来规避热点，有利于后续布线。若在布局阶段没有进行可布线评估和优化，可能导致后续布线出现迂回甚至单元之间无法连通，从而需要重新迭代而降低 VLSI 设计效率
  - 可布线性布局内容（关键是评估和优化）
    - 线长驱动布局，直到单元足够散开
    - 构建布线图
    - 可布线性评估：**拥塞**、DRC违例、引脚可抵达性
    - 可布线性优化：单元膨胀、拥塞梯度（全局布局阶段）



构建布线图的过程



(a) 线长驱动的全局布局结果　(b) 可布线性驱动的全局布局结果

线长驱动和可布线性布局的结果对比
左子图为拥塞图(Congestion Map)

可布线性布局：拥塞评估

- **评估方法**
  - 在布局过程中，需要对版图进行**快速、准确**的拥塞评估。已知版图内pin的位置分布，通过一些评估方法，就可以知道哪些区域容易出现布线拥塞，最终获得版图对应的congestion map
  - 每个grid的congestion：

$$capacity_{i,j} = \left\lfloor \frac{grid\_h}{d_{pitch}(layer)} \right\rfloor$$

$$overflow_{i,j} = \max[\,0, (\mathrm{demand_{i,j}} - \mathrm{capacity_{i,j}})]$$

$$util_{i,j} = \mathrm{demand_{i,j}} \,/\, \mathrm{capacity_{i,j}}$$

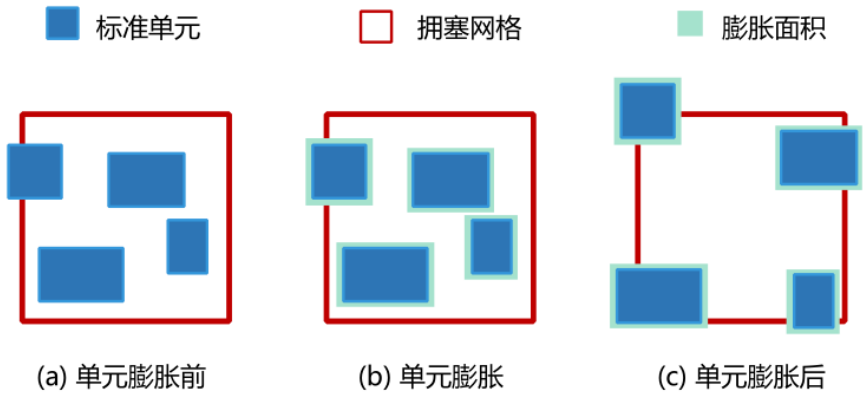
  - 输入
    - 线网/ pin坐标
    - Blockage
    - Track/ layer
  - 输出：
    - Congestion map
  - 指标：
    - 总溢出值 TOF
    - 最大溢出值 MOF
    - 平均峰值拥塞 PWC
      - $PWC = \frac{\sum(K_x \times ACE(x))}{\sum K_x}$
      - $ACE(x), x \in \{0.5,1,2,5\}$

拥塞评估的**四类方法**对比

方法	内容	精度	速度
静态法	基于简单的设计指标	4th	1st
概率法	基于布线模式和概率	3rd	3rd
构建法	调用全局布线工具	2nd	4th
网络法	训练机器学习模型	1st	2nd

可布线性布局：拥塞优化

- 优化方法
  - 通过调整密度间接考虑拥塞
    - **单元膨胀**：
      - 临时增大拥塞网格单元的尺寸
      - 关键点有三个：选择哪些单元膨胀、往哪个方向膨胀、膨胀多大
    - 阈值密度修正：
      - 减小拥塞网格的阈值密度，如根据pin密度修正
- 直接在目标函数中表征拥塞
  - 可微线网拥塞：
    - $\min\ \lambda_1 WL + \lambda_2 \sum (D - M_b)^2 + \lambda_3 \sum (C - S_b)^2$
  - 可微引脚密度：
    - $\min\ \lambda_1 WL + \lambda_2 \sum (D - M_b)^2 + \lambda_3 P$
  - 拥塞加权到线长：
    - $WL = \sum \left(\alpha_e \max |x_i - x_j| + \beta_e \max |y_i - y_j| \right)$
    - $\alpha_e = 1 + (f_y - f_x),\ \beta_e = 1 - (f_y - f_x)$



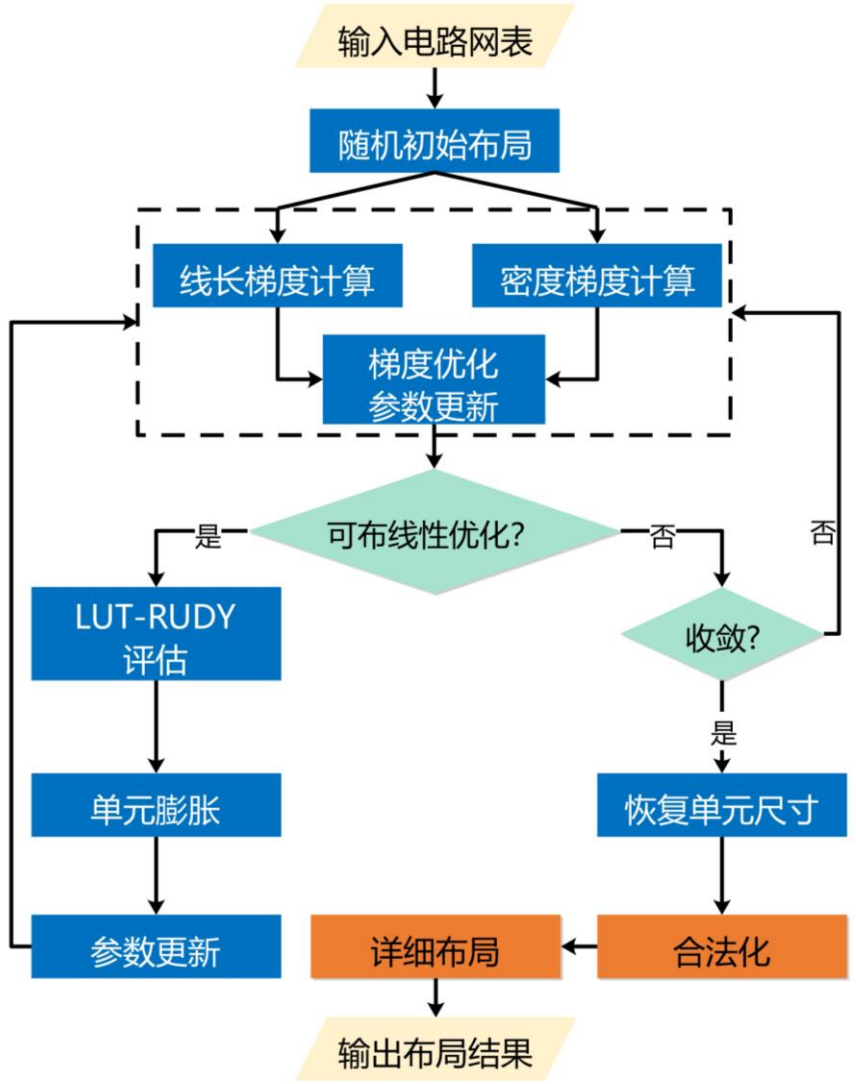
单元膨胀过程示意图

布局工具	时间	单元选择	膨胀方向	膨胀率
SimPLR	2011	前5%最拥塞网格	水平	简单加权函数
Ripple	2013	所有拥塞网格	水平/竖直	简单分段函数
POLAR	2014	前10%最拥塞网格	无区分	固定为10%
RePlAce	2019	所有拥塞网格	无区分	超线性函数

单元膨胀法的应用策略对比

iPL：可布线性方案

- **流程图**

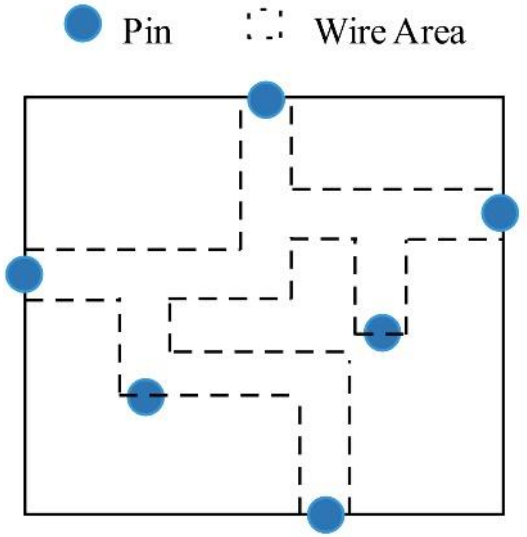


- 线长梯度：WA 线长光滑模型
- 密度梯度：e-Density 静电场模型
- 优化算法：Nesterov梯度下降算法
- 拥塞评估方法(由iEDA evaluator提供API)：
  - **LUT-RUDY**（Look Up Table-based RUDY）
  - Early-GR
- 细粒度单元膨胀：
  - 选择峰值拥塞网格的单元进行膨胀
  - H/V双方向独立膨胀
  - 动态膨胀率调整
  - 超线性膨胀指数修正

当全局布局所有单元足够散开(density overflow < 0.2)，开始进行可布线性评估和优化

iPL：评估法研究动机

- **现有方法的问题**
  - 静态法只能表征有限的版图信息如引脚密度分布，评估速度很快，但对于拥塞的表征精度很低
  - 构建法由于直接调用全局布线器来估计拥塞，评估精度很高但速度很慢，容易受布线器性能制约
  - 概率法通过表征线密度分布来估计拥塞，常用方法是RUDY (Rectangular Uniform wire DensitY)。RUDY在评估速度和精度上有较好的折衷，然而其存在假设上的不合理，限制了其评估精度
    - 假设1：无需对每个线网进行过于准确的布线需求估计
    - 假设2：布线器尽其所能在每个线网的外接矩形内部布线



RUDY示意图

$$D_e = \frac{wire\ area}{net\ area} = \frac{HPWL * Width}{BBox\ Area}$$

- 先前研究侧重关注可布线性优化，对于评估常采用默认方法如调用布线器或者概率法。实际上，评估和优化同等重要！
- 关键科学问题：
  - **如何设计一种兼顾性能和精度的可布线性评估方法?**

$$\frac{D(e)}{C(e)} = \sum_{i=1}^{m} \frac{HPWL(i) \times Width(i) \times Overlap(i)}{BBoxArea(i) \times C(e)}$$

m 表示所有net，$\frac{D(e)}{C(e)}$表示每个bin的拥塞值

$Overlap(i)$指net_i与当前bin的面积重叠率

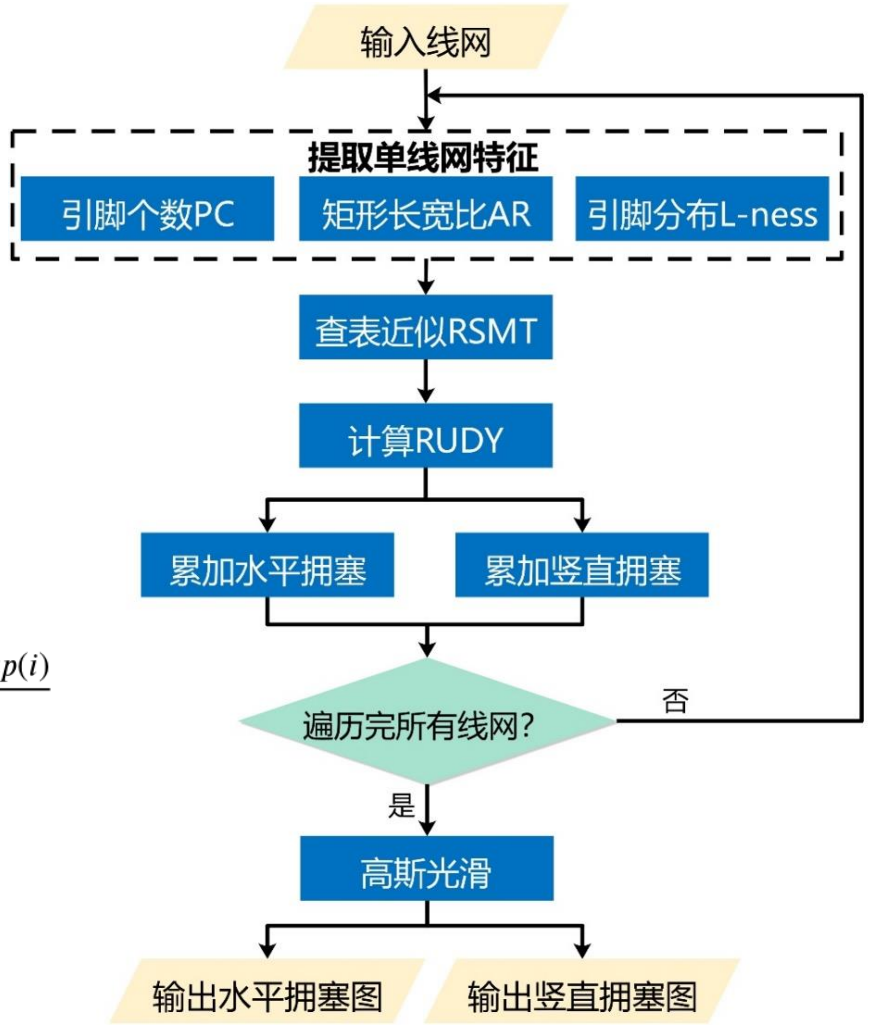
iPL：兼顾性能和精度的拥塞评估

- **提出方法：LUT-RUDY**
  - RUDY的两个假设
    - 假设1：无需对每个线网进行过于准确的布线需求估计
    - 假设2：布线器尽其所能在每个线网的外接矩形内部布线
  - 针对假设1：
    - 对于单个线网不准确的布线需求估计，会在现代设计数以万计的线网中累加后被放大，导致最终对整体布线需求的不精准估计——提出**基于查找表的真实线长逼近**策略，提高单线网的评估精度

$$\frac{D(e)}{C(e)} = \sum_{i=1}^{m} LUT(PC, AR, Lness) \times \frac{HPWL(i) \times Width(i) \times Overlap(i)}{BBoxArea(i) \times C(e)}$$

  - 针对假设2：
    - 真实布线器的行为难以预测，对于容易发生拥塞的设计往往需要迂回布线——提出**基于高斯光滑的布线行为模拟**策略，考虑线网矩形外部的布线情况

$$w_{ij} = \frac{1}{2\pi\sigma^2} \times e^{-\frac{(i-i_0)^2+(j-j_0)^2}{2\sigma^2}}$$



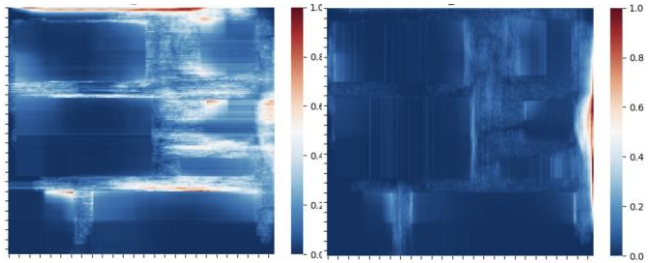
基于LUT-RUDY的拥塞评估流程

Yihang Qiu. LUTPLACE: AN IMPROVED LOOKUP TABLE-BASED PLACEMENT FOR ROUTABILITY[C]. CSTIC,2023

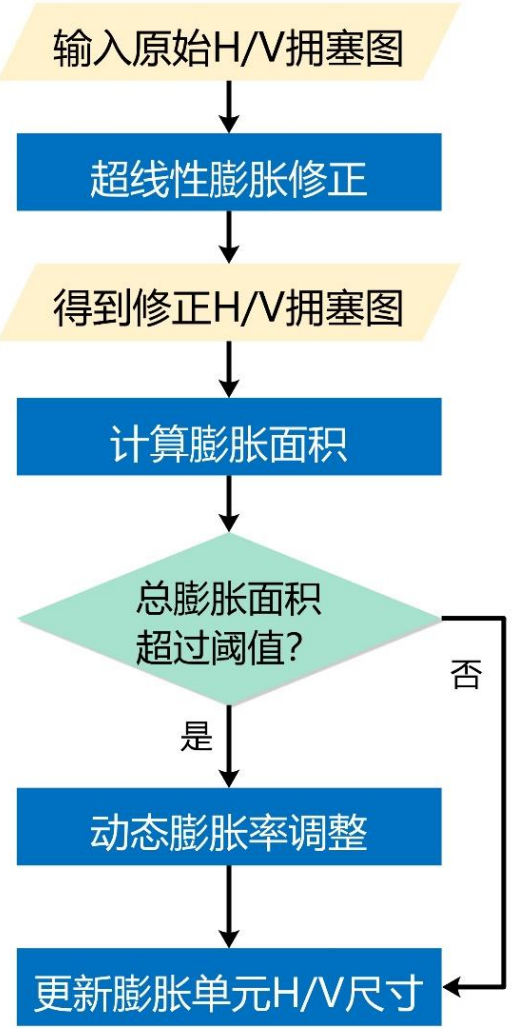
iPL：可布线性优化

- **细粒度单元膨胀策略**
  - 拥塞图预处理
    - 应用超线性修正函数，放大像素值差异以增强后续单元膨胀效果
      - 限制单方向膨胀率$h_{max}(v_{max})$不超过1.6，以保持算法稳定性
  - 选择哪些单元膨胀
    - 重点关注高度局部拥塞的区域
      - 对高于平均峰值拥塞$\sqrt{PWC}$的网格内的单元进行膨胀
  - 往哪个方向膨胀
    - 在H/V拥塞图指导下独立膨胀单元高宽
  - 膨胀多大
    - 膨胀率随平均峰值拥塞降低而降低
      - α取值为1.2　　$\gamma_{\text{super}}^{hor} = \alpha \cdot \text{PWC}_{hor}$
    - 动态膨胀率调整，其中总膨胀面积阈值为版图空白面积的10%

$$\text{Ratio}_h = \max\left(\left(\frac{D(e)}{C(e)}\right)^{\gamma_{\text{super}}^{hor}}, h_{max}\right)$$



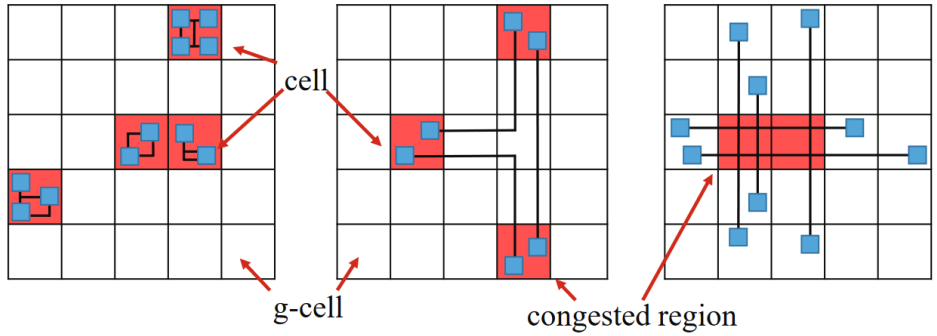
水平/竖直拥塞图



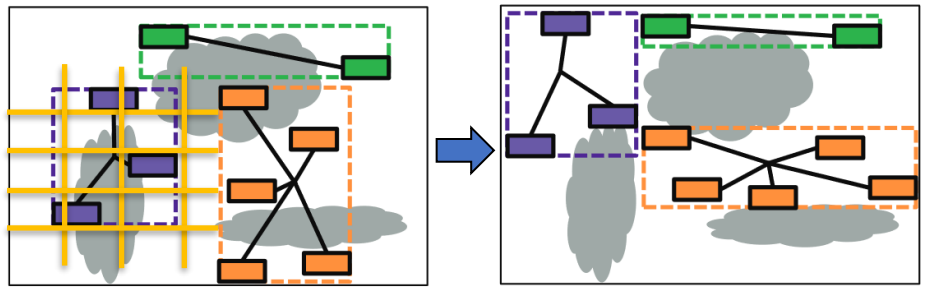
单元膨胀的算法流程

iPL：可布线性优化

- **可微拥塞梯度**
  - 研究动机
    - 单元膨胀法应对不了拥塞网格内没有单元的情况
    - $\min\ \lambda_1 WL + \lambda_2 \sum (D - M_b)^2 + \lambda_3 \sum (C - S_b)^2$ 最终结果是线密度均匀分布，而拥塞更应该关注局部性
  - 模型假设
    - 布线器只会在每个线网的外接矩形内部(BBox)走线
    - 线网内部所有位置的走线概率相同
  - 数学模型
    - $\min\ \lambda_1 WL + \lambda_2 \sum (D - M_b)^2 + \lambda_3 \sum C$
      - $C_i = \sum \frac{P_x(N_i, g_k) \times P_y(N_i, g_k) \times Overflow_k}{\sum P_x(N_i, g_k) \times P_y(N_i, g_k)}$
      - 引入Bell-shaped函数对$P_x(N_i, g_k)$和$P_y(N_i, g_k)$光滑



局部/全局拥塞示意图



应用线网拥塞惩罚进行线网移动示意图

- N_i 的中心点表示为 $x_{N_i} = \frac{\max x_k + \min x_k}{2}$
- N_i 的宽度表示为 $w_{N_i} = \max x_k - \min x_k$
- 引入WA函数对 x_{N_i} 和 w_{N_i} 光滑化

J.-M. Lin, et al. Routability-driven Global Placer Target on Removing Global and Local Congestion for VLSI Designs[C]. ICCAD,2021.

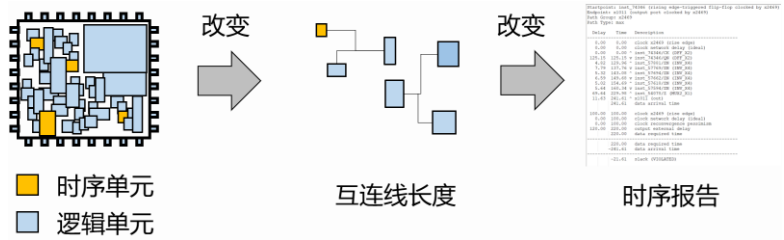
01 可布线性布局

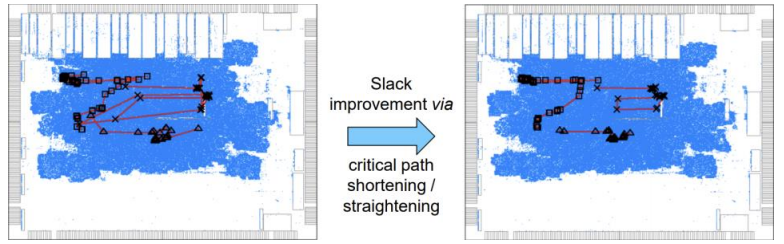
02 时序驱动布局

03 未来研究计划

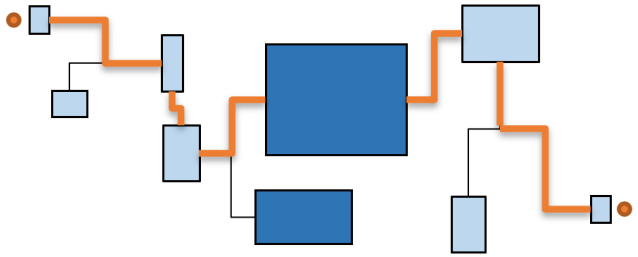
时序驱动布局：研究意义

- **布局与时序的关系**
  - 进入深亚微米制程后**互连线时延**成为影响时序的重要因素。布局通过改动单元位置，基本决定了布线空间从而影响时序收敛
  - 时序驱动布局标注“关键”线网/路径，引入优化手段改动单元位置，达到时序优化的目的。其中，**时序（趋势）变化的评估准确性**十分重要。
  - 时序驱动布局内容（关键是评估和优化）
    - 时序优化针对单元分布足够散开的布局状态
    - 时序评估：调用STA工具返回、直接法传播计算
    - 关键性标注：线网、时序路径上的pin点
    - 时序优化：线网赋权（全局布局阶段）、时序梯度（全局布局阶段）、单元选取+重放置（后全局布局/详细布局阶段）





（时序驱动布局）



时序存在违例的路径

iPL：问题处理

- **全局布局进行线网加权优化**

$$\underset{(x,y)}{\text{Minimize}} \sum_{e \in E} WL_e(x,y)$$

$$S.t. \quad \rho_b(x,y) \le \rho_0, \forall b \in B;$$
$$a_i + \hat{D}_{ij} \le a_j;$$
$$s_j \le r_j - a_j, \forall j \in V;$$

$$\underset{(x,y)}{\text{Minimize}} \sum_{e \in E} \boldsymbol{\omega_e} * WL_e(x,y) + \alpha^T N(x,y)$$

其中，

- $WL_e(x,y) = max_{i,j \in e}|x_i - x_j| + max_{i,j \in e}|y_i - y_j|$
- b 区域内的某个bin，B 区域内bin的集合
- $\rho_b(x,y)$ 单元在bin中的密度，ρ_0 密度阈值
- a_i/a_j节点i/j的到达时间变量
- $\hat{D}_{ij}$ 节点i与节点j关联边的时延
- s_j 裕量，r_j 需求到达时间，a_j 到达时间

- **时序违例（Slack）作为优化目标**

$$\text{Minimize} \quad -\sum_{k \in PO} \hat{s}_k \quad \Rightarrow \quad \text{Minimize} \sum_{k \in C} \left(\sum_{j \in \mathcal{F}_k} \boldsymbol{\lambda_{j,k} d_{j,k}} \right)$$

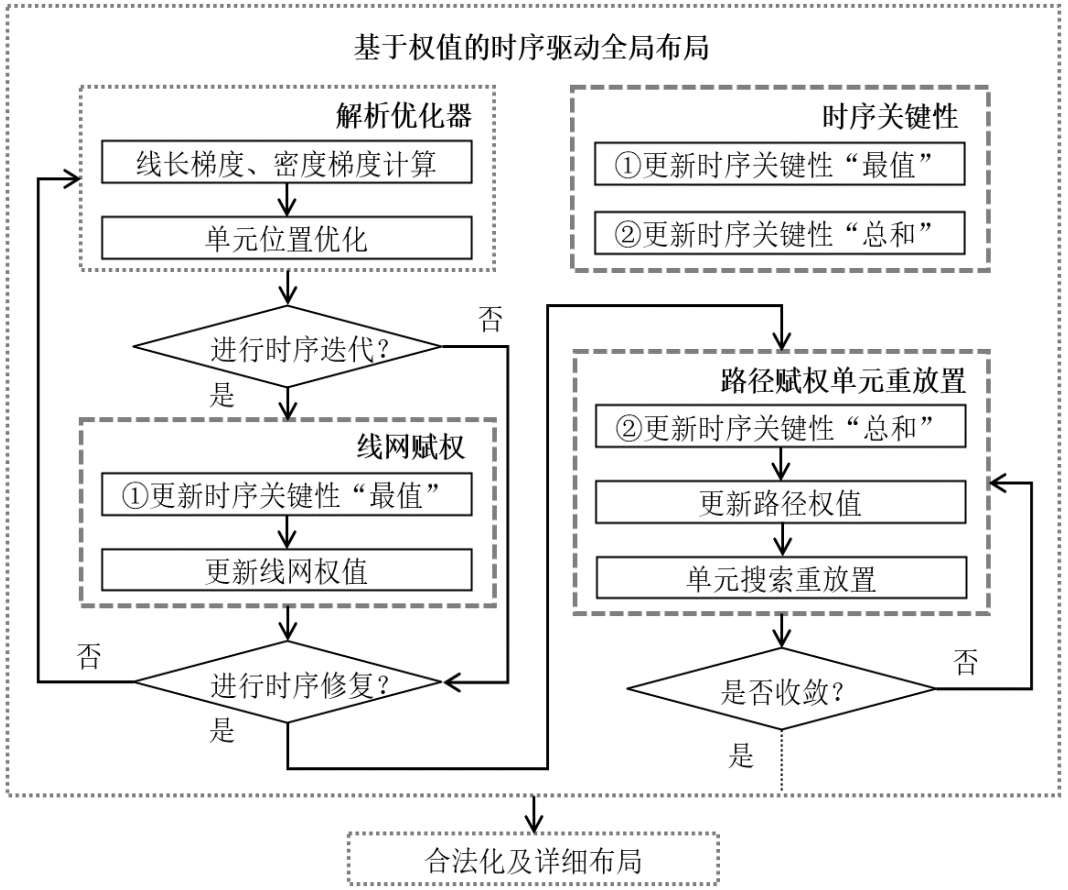
$$S.t. \quad \hat{s}_k \le 0, \ \forall k \in PO$$
$$r_k - a_k \ge \hat{s}_k, \forall k \in PO$$
$$a_j + d_{j,k} \le a_k, \forall k \in C$$

其中，

- $\hat{s}_k$ 指需要优化的负裕量
- a_k指到达时间，r_k指需求到达时间
- $d_{j,k}$指由引脚j到达引脚k所需的时延
- C指标准单元的集合，PO指时序路径终点集合

iPL：时序优化方案

- **流程图**



- **时序关键性**
  - 类WNS计算方式更新时序信息至各Pin点
  - 类TNS计算方式更新时序信息至各Pin点
- **解析优化器**：结合WA线长模型、e-Density静电场模型、使用Nesterov梯度下降算法进行单元位置更新
- **线网赋权**：线网权值根据时序关键性“最值”更新至线长梯度，融入解析法迭代过程
- **路径赋权单元重放置**
  - 使用时序关键性“总和”作为路径权值
  - 类比力的合成，横纵方向合成偏移搜索窗口
  - 设置搜索窗口，窗口内选点进行单元重放置
  - 计算单元重放置的Benefit，最终放置在最佳位置

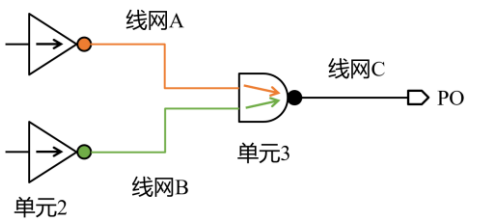
iPL：时序优化方案

- **时序关键性表示**
  - 时序路径终点（PO）评估时序违例
    - STA获取PO处的到达时间（ArrivalTime）和需求到达时间（RequiredTime）
    - 在PO处设初始关键性值λ=1，后续迭代乘以ArrivalTime与RequiredTime的比值**表示违例程度**
  - 时序关键性值的逆拓扑序传播
    - PO处值通过**逆拓扑序传播**至各个引脚Pin
    - 在经过**单元传播**时，单元输出引脚Pin值分配给各输入引脚Pin，分配方式是根据输入引脚Pin对造成路径违例的“贡献”决定
    - 在经过**线网传播**时，分别存下两种情况的值
      - “最值”模式：仅保留最大值
      - “总和”模式：保留累加值

$$SLACK^{Late} = RequiredTime^{Late} - ArrivalTime^{Late} \quad (1)$$

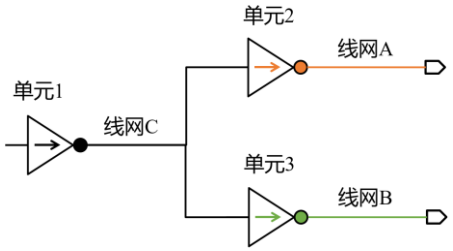
$$\lambda_{PO}^{0} = 1 \quad (2)$$

$$\lambda_{PO}^{k+1} = \lambda_{PO}^{k} * \left(\frac{ArrivalTime^{Late}}{RequiredTime^{Late}}\right) \quad (3)$$



$$\lambda_{A_1,3} = \lambda_{A_1,3} * \left(\frac{ArrivalTime_1^{Late} + Delay_{1,3}^{Late}}{ArrivalTime_3^{Late}}\right)$$

$$\lambda_{B_2,3} = \lambda_{B_2,3} * \left(\frac{ArrivalTime_2^{Late} + Delay_{2,3}^{Late}}{ArrivalTime_3^{Late}}\right)$$

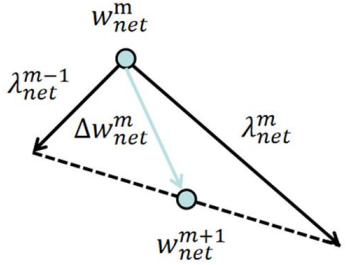


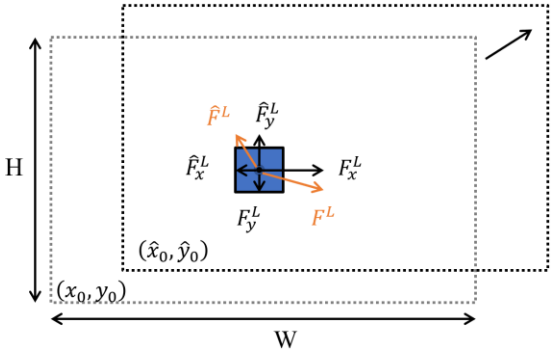
“最值”模式：$\lambda_{C_1} = max(\lambda_{A_2}, \lambda_{B_3})$

“总和”模式：$\lambda_{C_1} = \lambda_{A_2} + \lambda_{B_3}$

iPL：时序优化方案

- **线网赋权参与解析法迭代过程**
  1. 已知第m-1轮与第m轮迭代线网关键性λ_{net}^{m-1}、 λ_{net}^{m}
  2. 第m轮迭代线网权值变化量 $\Delta w_{net}^{m} = \boldsymbol{\beta} * \lambda_{net}^{m-1} + (\mathbf{1} - \boldsymbol{\beta}) * \lambda_{net}^{m}$
  3. 线网权值 $w_{net}^{m} = w_{net}^{(m-1)} + \Delta w_{net}^{m}$
  4. 更新$\boldsymbol{\omega_e}$ ⇒ $\underset{(x,y)}{\text{Minimize}} \ \sum_{e \in E} \boldsymbol{\omega_e} * WL_e(x,y)$





- **路径赋权单元重放置**
  1. 确定目标函数是最小化路径权值与时延的乘积 ⇔ $\text{Minimize} \ \sum_{k \in C} (\sum_{j \in \mathcal{F}_k} \lambda_{j,k} d_{j,k})$
  2. 目标函数表明：减小关键路径上的时延能够有效的减少时序违例，同时由于时延d是通过STA工具返回的全局值，难以使用梯度法进行求解，因此设立搜索窗口进行重放置搜索
  3. 搜索窗口的设置可以根据单元移动方向进行偏移，以加快收敛速度

01 可布线性布局

02 时序驱动布局

03 未来研究计划

研究计划

- **构建可微的可布线性/时序驱动全局布局优化模型**
  - 研究解决可布线性/时序问题更直接有效的建模和优化方案
- **在布局阶段提前预测布线拥塞和DRC违例**
  - 通过提取网表和版图特征，训练深度神经网络模型(如GNN, CNN等)，提前预测GR后的布线拥塞和DR后的DRC违例，并基于预测结果指导开展可布线性布局优化
- **新的密度表达和计算**
  - 构建密度可微函数，考虑多场的密度建模
- **联合优化cell location/size 和 buffer**
- **优化方向学习**
- **2.5D/3D布局**
  - 考虑宏单元的Die 2 Die 布局研究 (ICCAD@Contest 2023)

总结

- **iPL可布线性布局**
  - 可布线评估方法要求兼顾**性能**和**精度**
  - 可布线优化方法不仅要考虑如何消除**局部拥塞**，而且要关注消除**全局拥塞**
  - 应用深度学习**预测(评估)可布线性**成为研究热点
- **iPL时序驱动布局**
  - 时序评估方法重点是**性能**和（趋势变化的）**精度**
  - 时序优化方法需要注意**避免额外时序违例路径**的产生，尽量兼顾**不产生拥塞**区域
  - 应用深度学习**预测(评估)时序**成为研究热点

iEDA Tutorial 第二期议程

- Part 1 iEDA-iFP/iPDN工具架构、特性与使用（黄增荣）

- Part 2 iEDA-iMP 关键技术（黄富兴）

- Part 3 iEDA-iPL 问题介绍、架构、使用与规划 （陈仕健）

- Part 4 iEDA-iPL 关键技术 （邱奕杭）

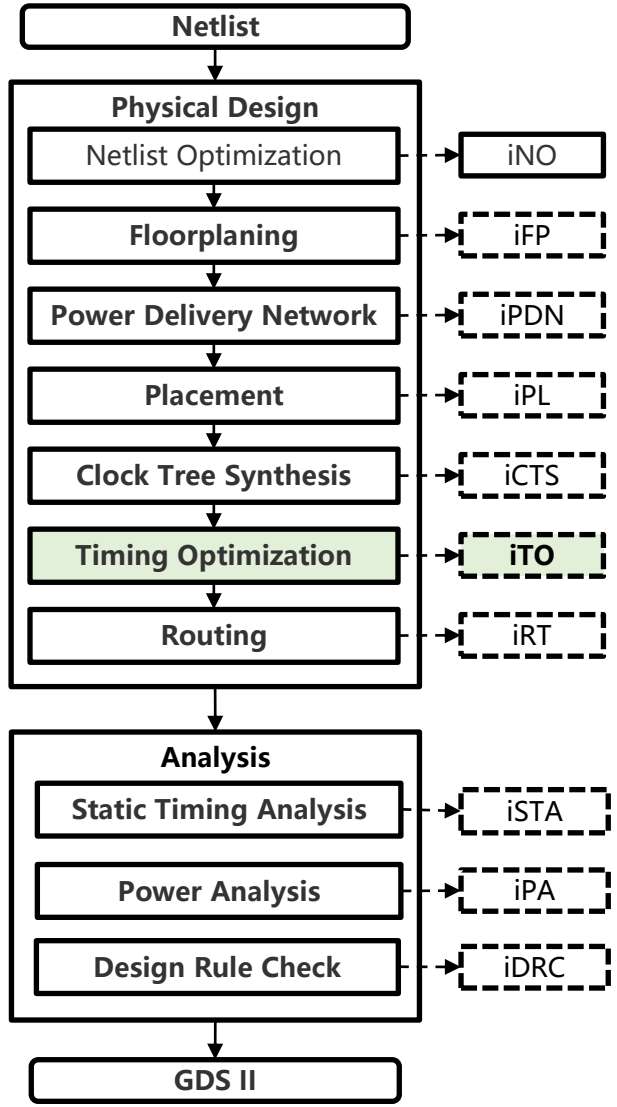
- **Part 5 iEDA-iTO工具架构、特性与关键技术 （吴鸿熙）**

iTO工具介绍

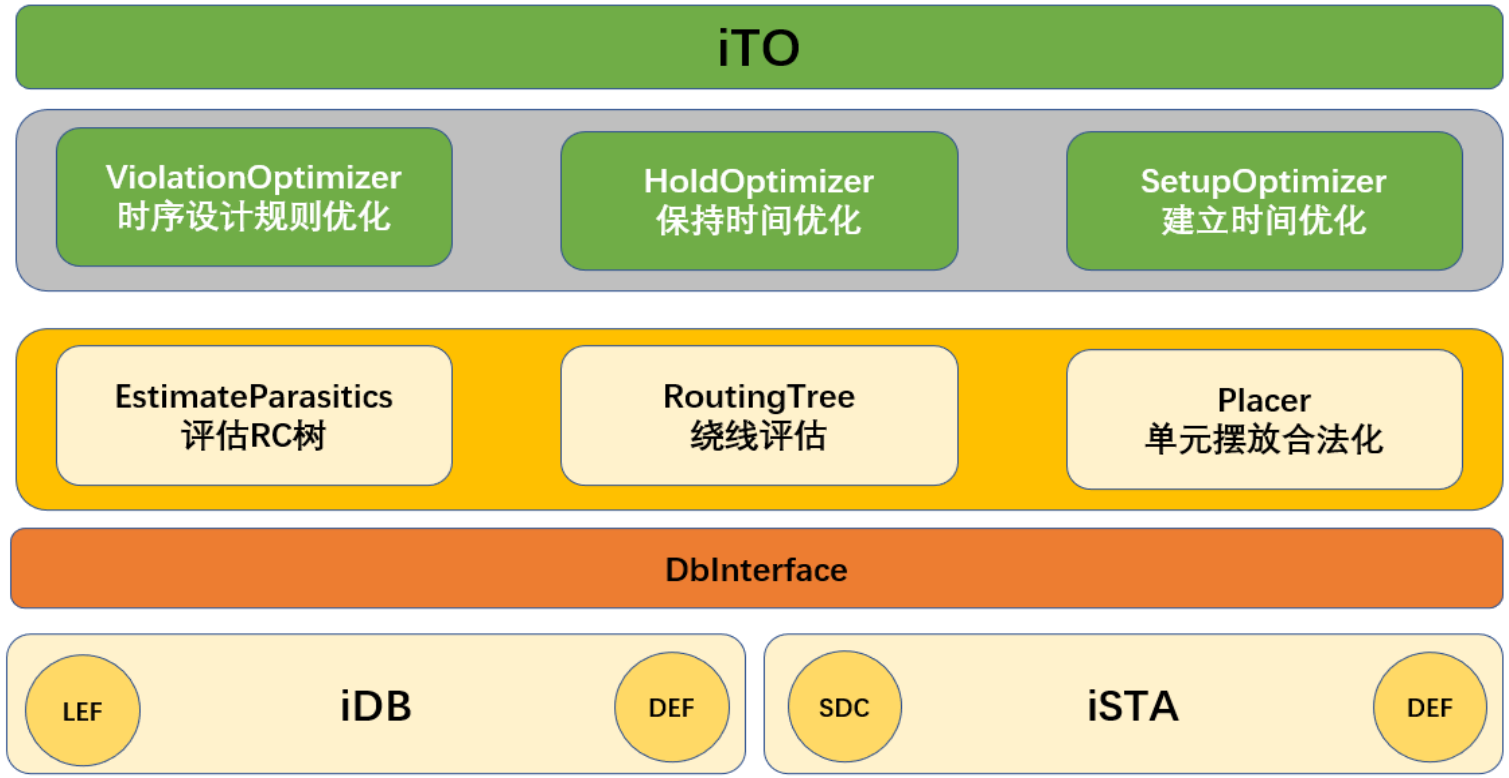
iTO是一款开源的芯片时序优化工具，是iEDA项目组开源EDA工具链的一部分。

设计目标：

TO的全称为Timing Optimization，时序优化。在该步骤，iTO工具会根据时序约束文件，对芯片进行时序分析，目的是通过单元尺寸调整和插入缓冲器等方法尽可能地修复芯片存在的时序违例。

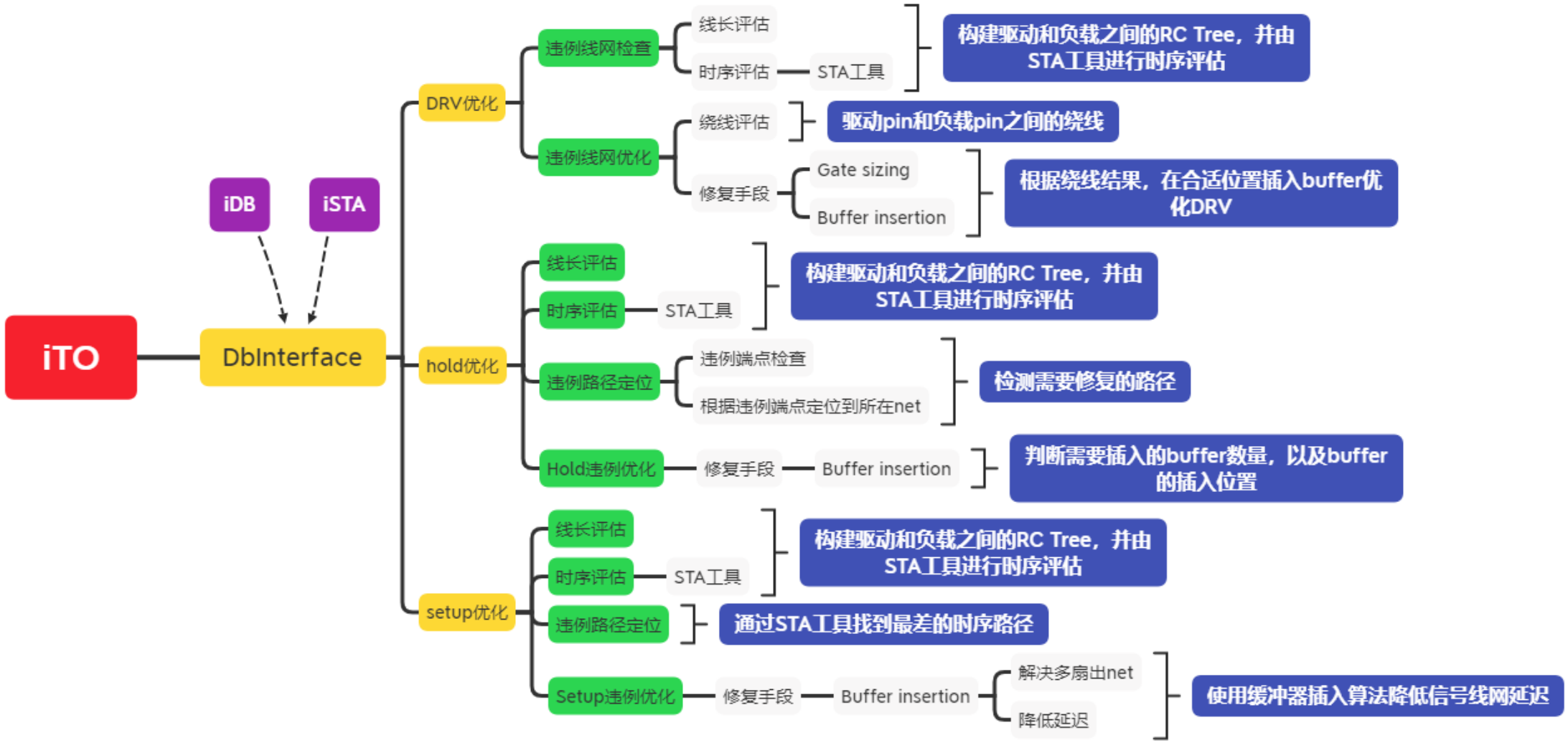


软件整体架构



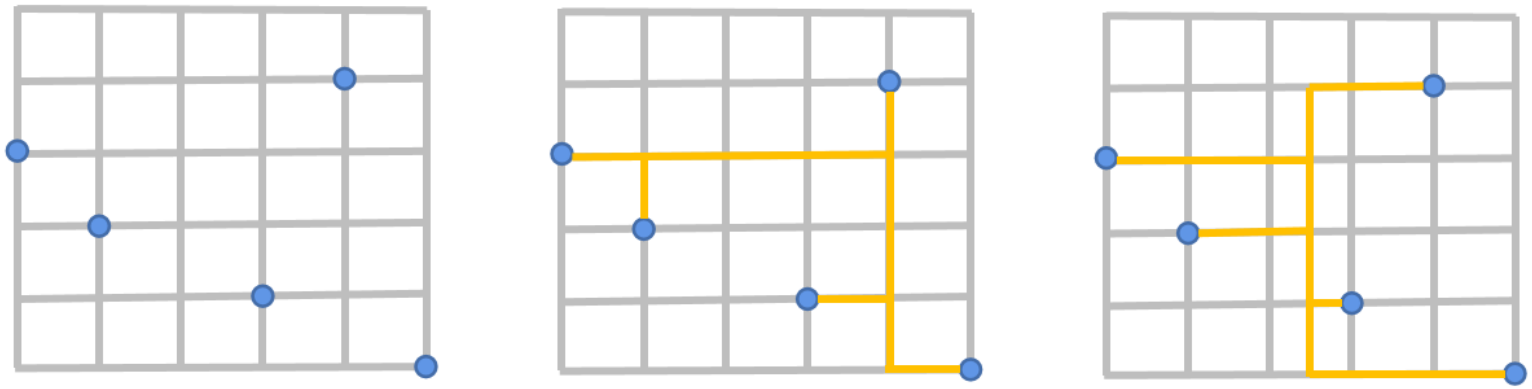
- **DbInterface（数据转换）：** 数据交互模块，启动 iDB 和 iSTA，并进行一些数据初始化
- **工具模块：** 工具模块包括RC树评估、绕线评估以及单元摆放合法化。
- **优化模块：** 优化模块包括时序设计规则违例优化、保持时间优化以及建立时间优化。

软件整体架构


iTO
DbInterface
iDB
iSTA
DRV优化
违例线网检查
线长评估
时序评估
STA工具
构建驱动和负载之间的RC Tree，并由STA工具进行时序评估
违例线网优化
绕线评估
驱动pin和负载pin之间的绕线
修复手段
Gate sizing
Buffer insertion
根据绕线结果，在合适位置插入buffer优化DRV
hold优化
线长评估
时序评估
STA工具
构建驱动和负载之间的RC Tree，并由STA工具进行时序评估
违例路径定位
违例端点检查
根据违例端点定位到所在net
检测需要修复的路径
Hold违例优化
修复手段
Buffer insertion
判断需要插入的buffer数量，以及buffer的插入位置
setup优化
线长评估
时序评估
STA工具
构建驱动和负载之间的RC Tree，并由STA工具进行时序评估
违例路径定位
通过STA工具找到最差的时序路径
Setup违例优化
修复手段
Buffer insertion
解决多扇出net
降低延迟
使用缓冲器插入算法降低信号线网延迟

工具模块

绕线评估



RC树评估：iTO 依赖于 iSTA 工具提供的时序信息，在更新时序信息之前需要更新每条线网的RC树信息。在更新 RC树之前需要调用绕线评估模块以产生绕线拓扑。

单元摆放合法化：单元在摆放时需要满足单元之间无重叠，行对齐和site对齐的规则，单元摆放合法化模块将单元合法的摆放在指定位置的附近。

01 软件架构

02 关键技术

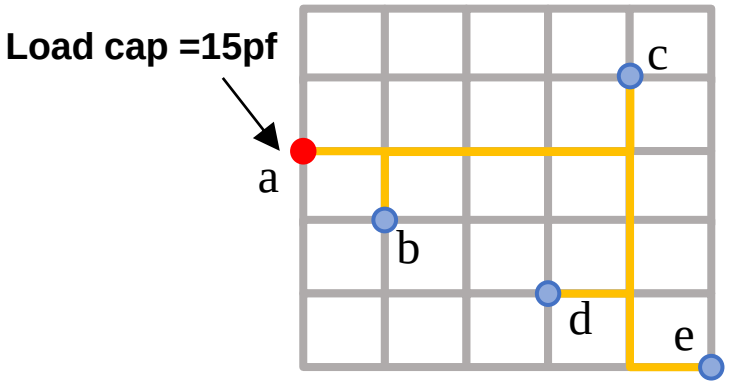
03 使用指南

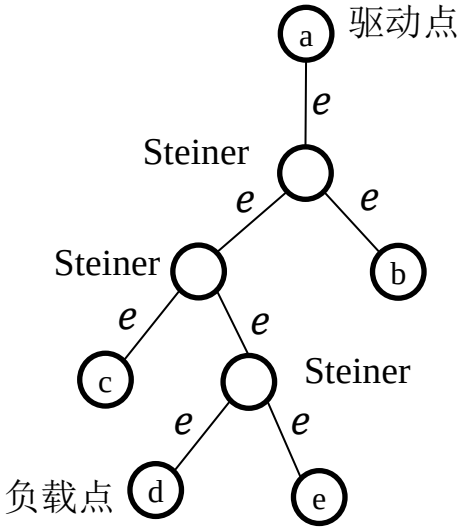
04 未来研究计划

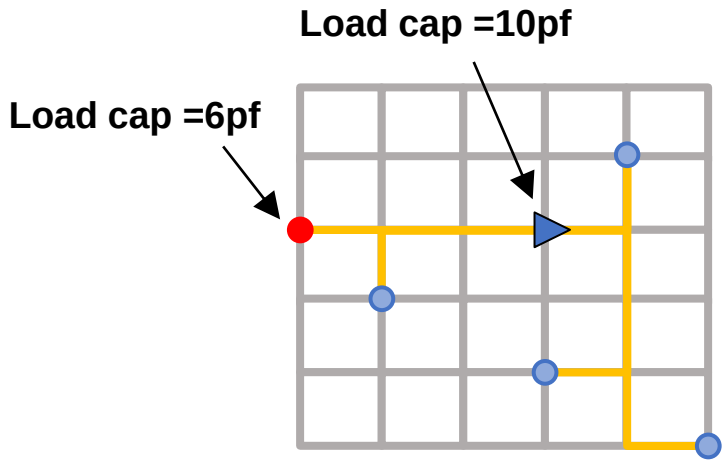
DRV优化

时序设计规则(Design Rule Violation, DRV)约束主要是指最大转换时间约束(max_transition)、最大电容(max_cap)约束和最大扇出(max_fanout)约束。前两个是硬性条件，在签核阶段必须满足要求

1. 使用FLUTE构建违例线网的拓扑，并表示为二叉树，该二叉树以驱动引脚为根
2. 自底向上遍历二叉树，使得驱动引脚以及插入的缓冲器的负载电容、slew、fanout满足约束

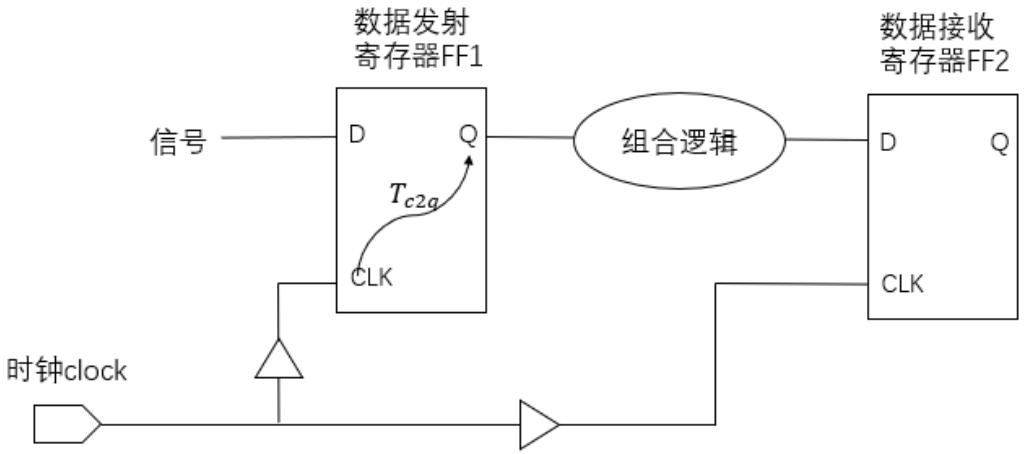


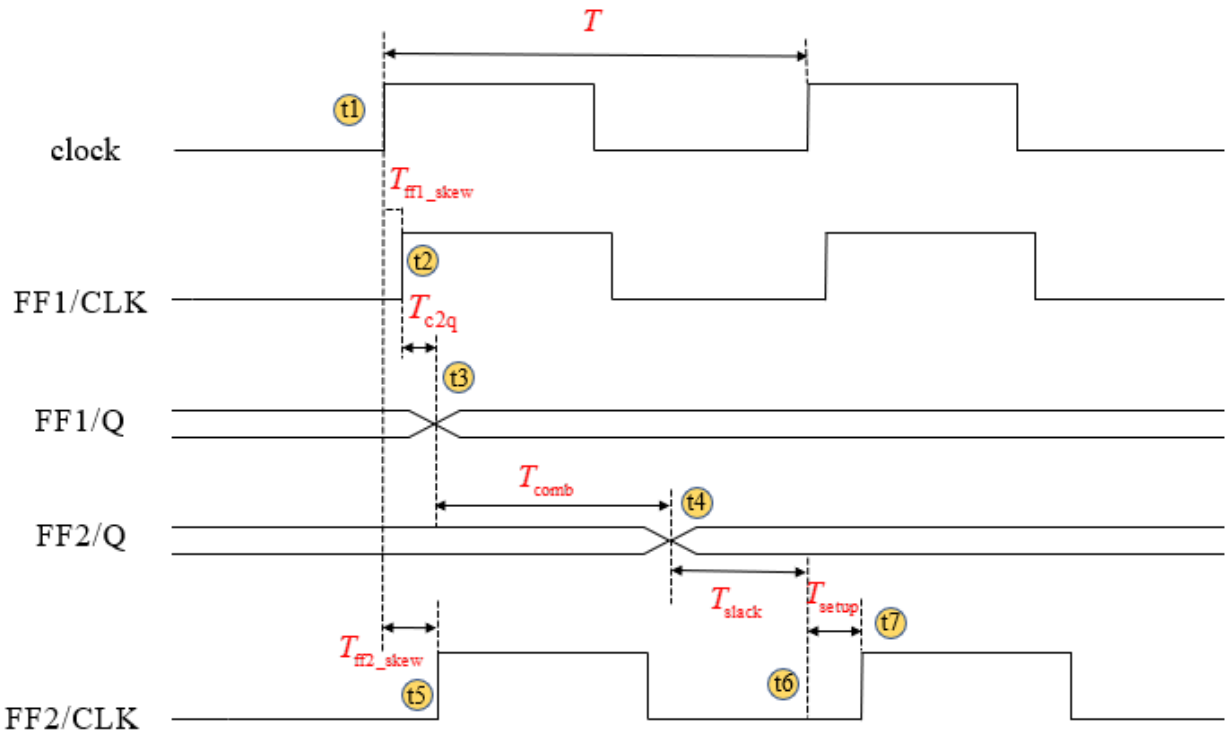




Setup优化

建立时间约束：对于存储元件，当时钟沿到达时，输入信号必须达到稳定(稳态)所需要的时间(例如：触发器和锁存器)。





T_{ff1_skew} 时钟源信号到达 FF1 clk 端的时间

T_{c2q} FF1 clk 端到触发 Q 端信号的时间

T_{comb} 信号经过组合逻辑到达FF2 D 端的时间

T_{ff2_skew} 时钟源信号到达 FF2 clk 端的时间

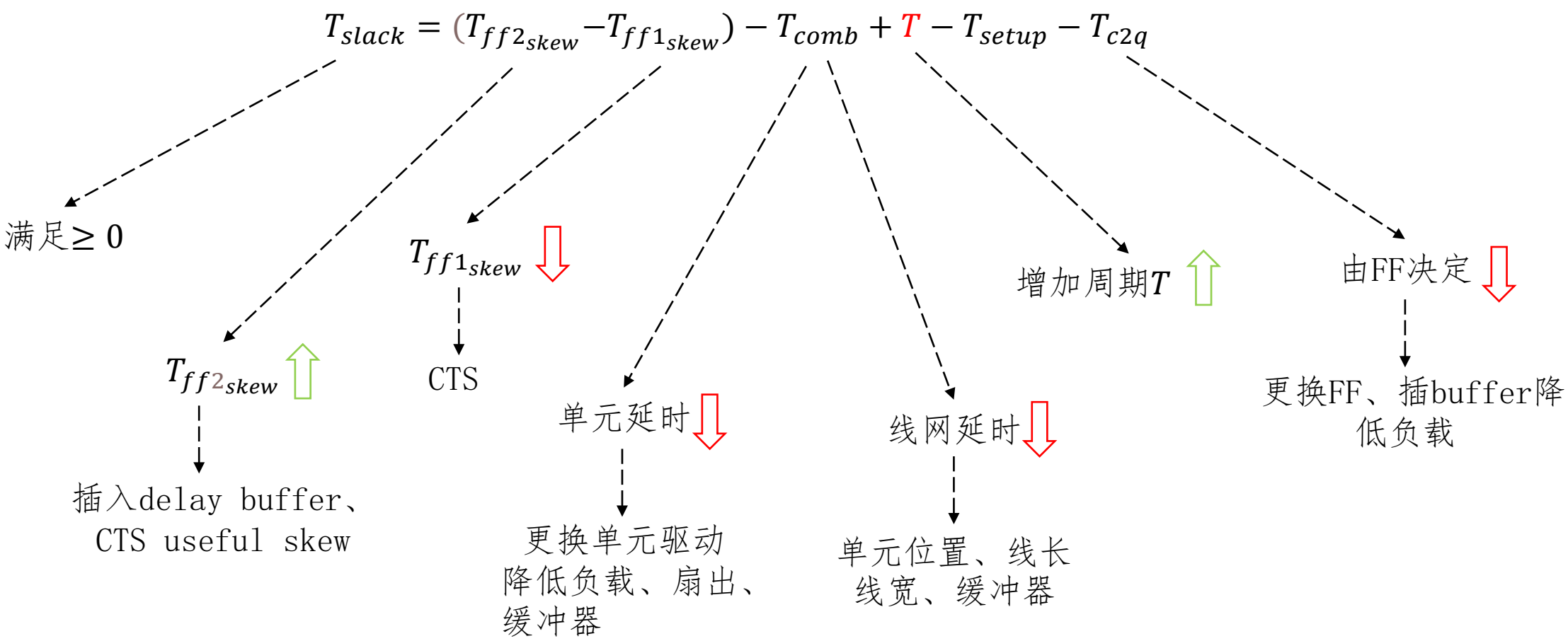
$$Actual\ \ Arrival\ \ Time(t4) = t1 + T_{ff1\_skew} + T_{c2q} + T_{comb}$$

$$Required\ Arrival\ \ Time(t6) = t1 + T_{ff2\_skew} + T - T_{setup}$$

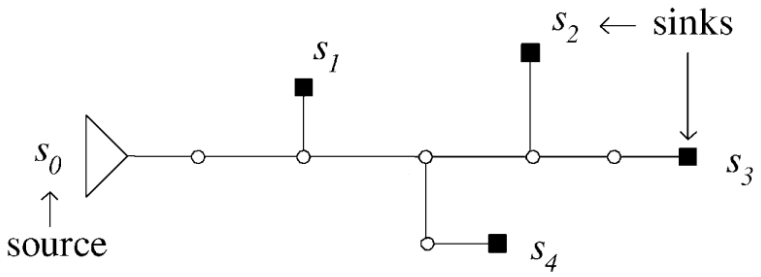
$$T_{slack} = t6 - t4 = \left(T_{ff2_{skew}} + T - T_{setup}\right) - \left(T_{ff1_{skew}} + T_{c2q} + T_{comb}\right)$$

保证建立时间的条件为$T_{slack} \geq 0$

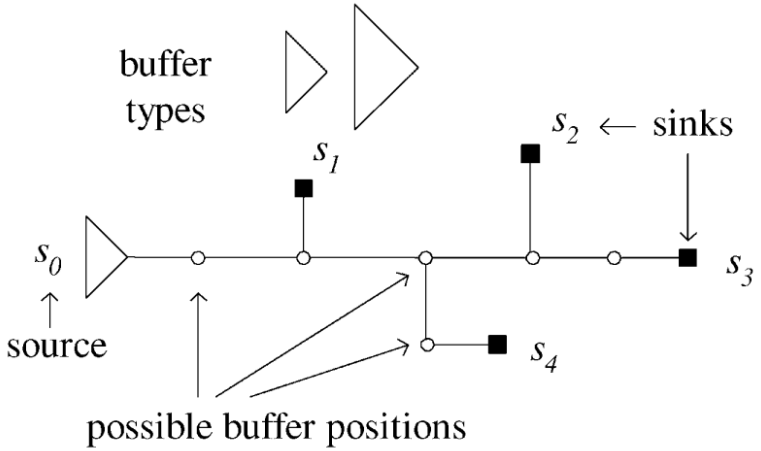
Setup优化


$T_{slack} = (T_{ff2_{skew}} - T_{ff1_{skew}}) - T_{comb} + T - T_{setup} - T_{c2q}$
满足≥ 0
$T_{ff2_{skew}}$
插入delay buffer、
CTS useful skew
$T_{ff1_{skew}}$
CTS
单元延时
更换单元驱动
降低负载、扇出、
缓冲器
线网延时
单元位置、线长
线宽、缓冲器
增加周期T
由FF决定
更换FF、插buffer降
低负载

Setup优化

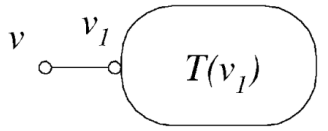


依赖于给定的拓扑



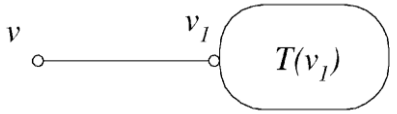
$C(\cdot)$ 负载电容
$Q(\cdot)$ 需要到达时间

1.插入缓冲器



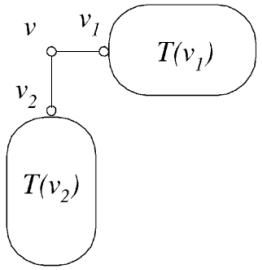
$$Q(v,\beta) = \max_{\alpha}\{Q(v_1,\alpha) - R(b)\cdot C(v_1,\alpha) - K(b)\}$$
$$C(v,\beta) = C(b)$$

2.添加互连线



$$Q(v,\alpha) = Q(v_1,\alpha) - R(e)C(e)/2 - R(e)C(v_1,\alpha)$$
$$C(v,\alpha) = C(v_1,\alpha) + C(e).$$

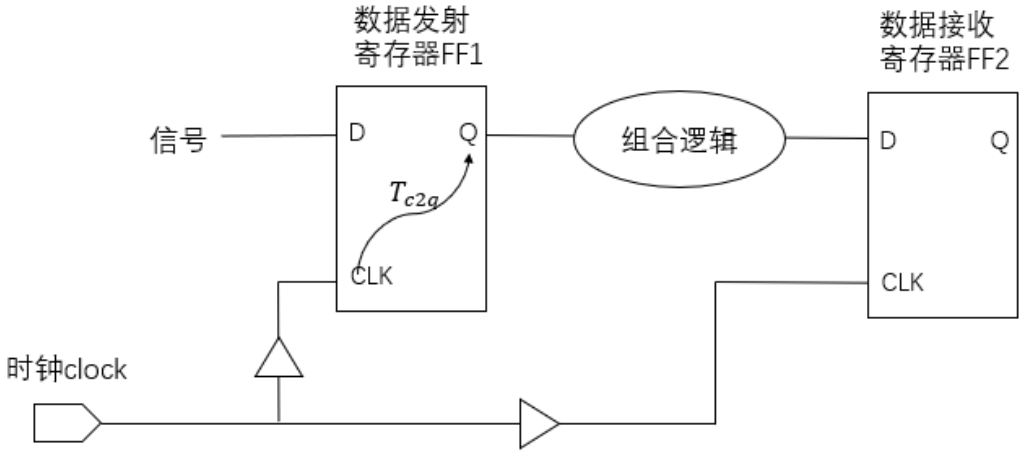
3.左右分支合并

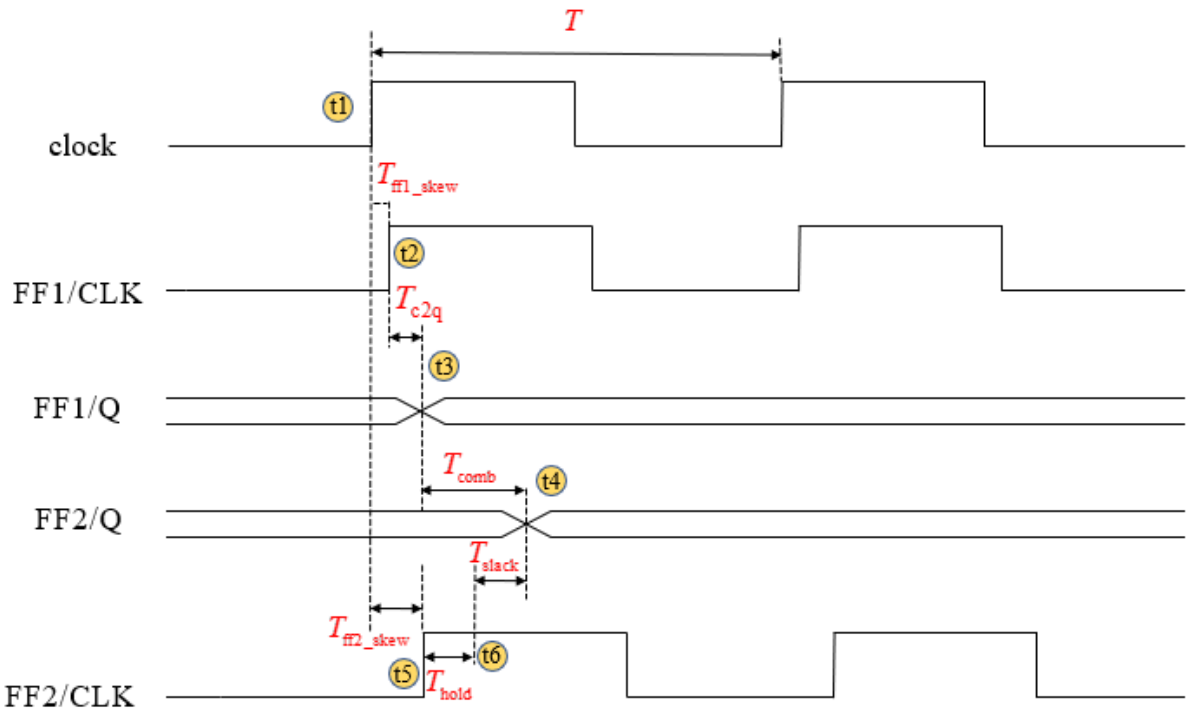


$$Q(v,\beta) = \min\{Q(v_1,\alpha_1), Q(v_2,\alpha_2)\}$$
$$C(v,\beta) = C(v_1,\alpha_1) + C(v_2,\alpha_2).$$

Hold优化

保持时间约束：对于存储元件，当时钟沿到达之后，输入信号仍需保持稳定的时间。





T_{ff1_skew} 时钟源信号到达 FF1 clk 端的时间

T_{c2q} FF1 clk 端到触发 Q 端信号的时间

T_{comb} 信号经过组合逻辑到达FF2 D 端的时间

T_{ff2_skew} 时钟源信号到达 FF2 clk 端的时间

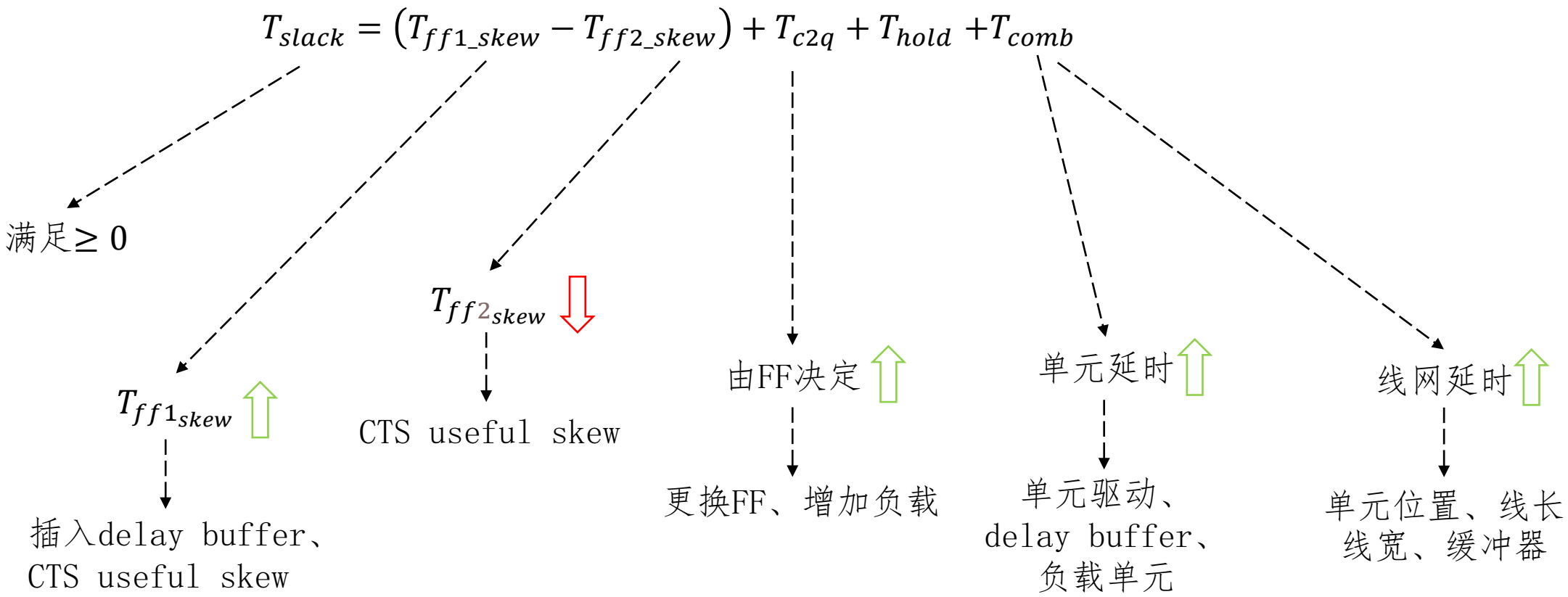
$$Actual\ \ Arrival\ Time(t4) = t1 + T_{ff1\_skew} + T_{c2q} + T_{comb}$$

$$Required\ Arrival\ \ Time(t6) = t1 + T_{ff2\_skew} + T_{hold}$$

$$T_{slack} = t4 - t6 = (T_{ff1\_skew} + T_{c2q} + T_{comb}) - (T_{ff2\_skew} + T_{hold})$$

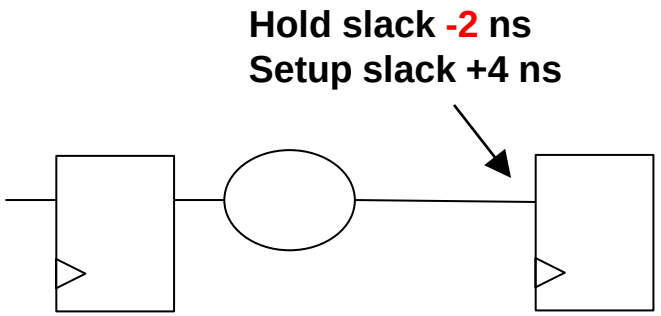
保证保持时间需要满足$T_{slack} \geq 0$

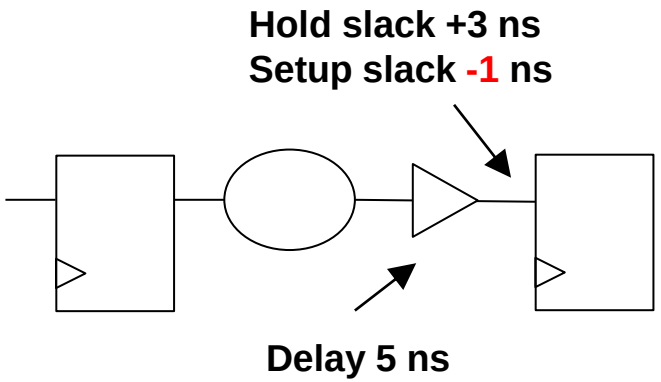
Hold优化


$T_{slack} = (T_{ff1\_skew} - T_{ff2\_skew}) + T_{c2q} + T_{hold} + T_{comb}$
满足≥ 0
$T_{ff1_{skew}}$
插入delay buffer、
CTS useful skew
$T_{ff2_{skew}}$
CTS useful skew
由FF决定
更换FF、增加负载
单元延时
单元驱动、
delay buffer、
负载单元
线网延时
单元位置、线长
线宽、缓冲器

Hold优化

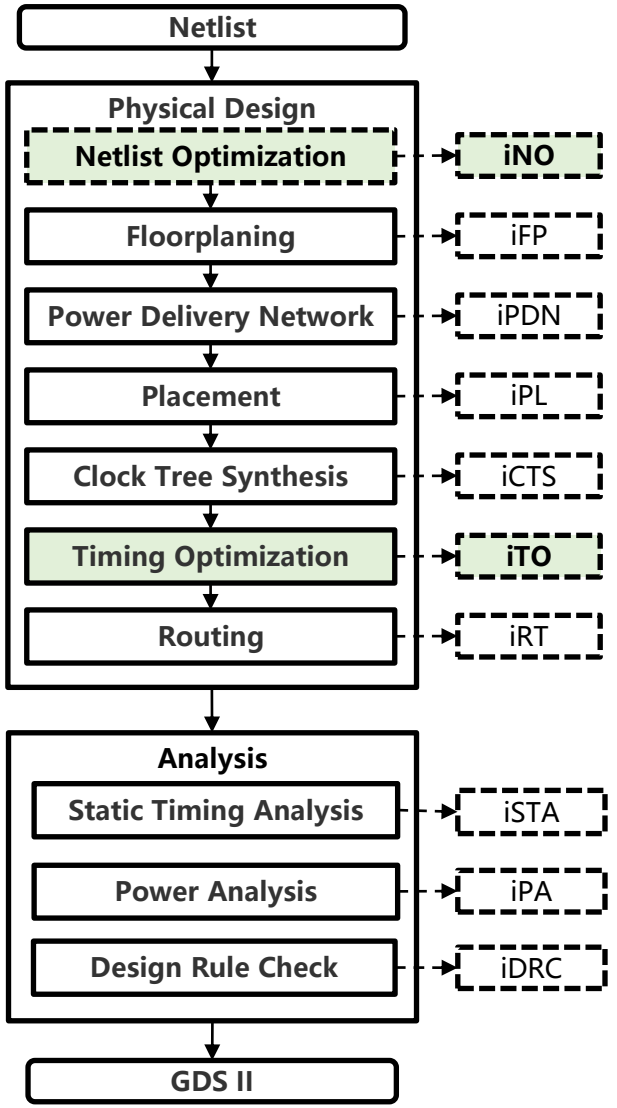
- 线长评估
- 时序评估
- 违例路径定位：
  - 根据iSTA工具报告的违例情况，检测需要优化的路径
- 违例路径优化：
  - 选择缓冲器，以及需要插入的缓冲器个数
  - 在违例pin和其驱动pin之间插入缓冲器





tcl命令

TCL命令	功能说明	参数	定义
run_to	自动执行时序优化，需在config文件中指定优化目标	-config	iTO配置文件
run_to_drv	进行时序设计规则违例优化	-config	iTO配置文件
run_to_setup	进行建立时间优化	-config	iTO配置文件
run_to_hold	进行保持时间优化	-config	iTO配置文件
run_no_fixfanout	进行扇出优化	-config	iNO配置文件



报告解读

1 DRV优化报告

```
Found 0 slew violations.
Found 0 capacitance violations.
Found 0 fanout violations.
Found 0 long wires.
Before ViolationFix | slew_vio: 0 cap_vio: 0 fanout_vio: 0 length_vio: 0    \\ 优化前违例情况
The 1th check
After ViolationFix | slew_vio: 0 cap_vio: 0 fanout_vio: 0 length_vio: 0 \\ 优化后违例情况
Inserted 0 buffers in 0 nets.
Resized 0 instances.
```

2 Setup优化报告

```
-0.304023 -0.204023    // setup优化过程中的WNS变化情况
Inserted 10 buffers.
Resized 10 instances.
Unable to repair all setup violations.
```

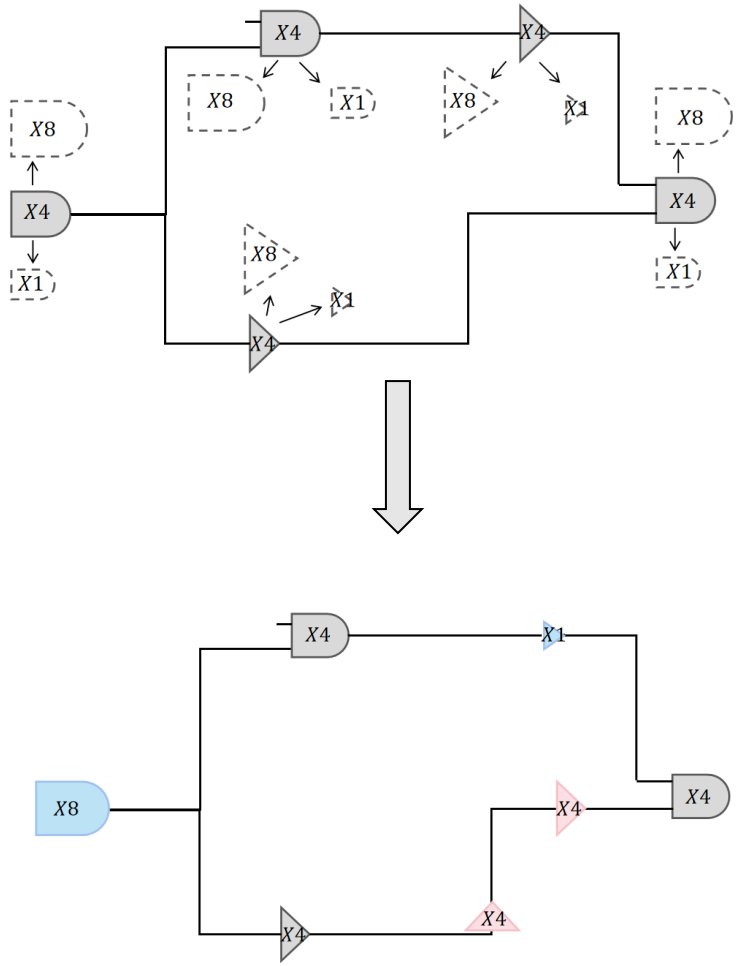
报告解读

3 Hold优化报告

```
// 优化前Hold违例情况。
--------------------------------------------------------------------------------
Clock Group                                          Hold TNS            Hold WNS
--------------------------------------------------------------------------------
core_clock                                                 0                  0
--------------------------------------------------------------------------------

Worst Hold Path Launch : dpath/a_reg/_145_:CLK
Worst Hold Path Capture: dpath/a_reg/_145_:CLK


Finish hold optimization!
Total inserted 0 hold buffers and 0 load buffers.

// 优化后Hold违例情况。
--------------------------------------------------------------------------------
Clock Group                                          Hold TNS            Hold WNS
--------------------------------------------------------------------------------
core_clock                                                 0                  0
--------------------------------------------------------------------------------
```

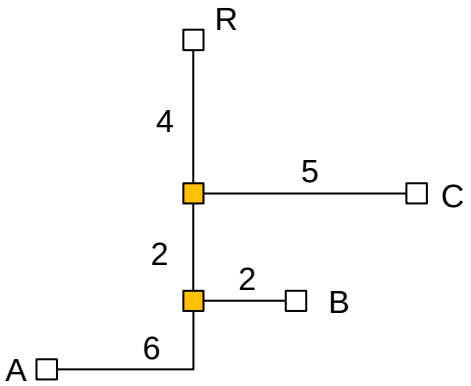
研究计划一：结合Gate sizing和Buffer insertion

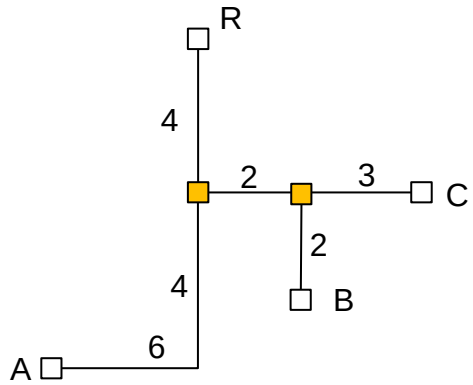
- 动机
  - gate sizing和buffer insertion有相同的核心目标
- 困难
  - 缓冲器插入会改变网表拓扑，需要注意更多问题
- 问题描述
  - 输入
    - 待优化的网表
  - 输出
    - 为每个instance实例进行size调整
    - 是否插入缓冲器及其size



研究计划二：缓冲树拓扑构建

- 动机
  - 不同的拓扑下时延不一样，直接影响到时序，以及优化的难度





A: 1*22+4*(2+18)+2*(1+10)+6*(3+1)=148
B: 1*22+4*(2+18)+2*(1+10)+2*(1+1)=128
C: 1*22+4*(2+18)+2*(2.5+1) =117.5

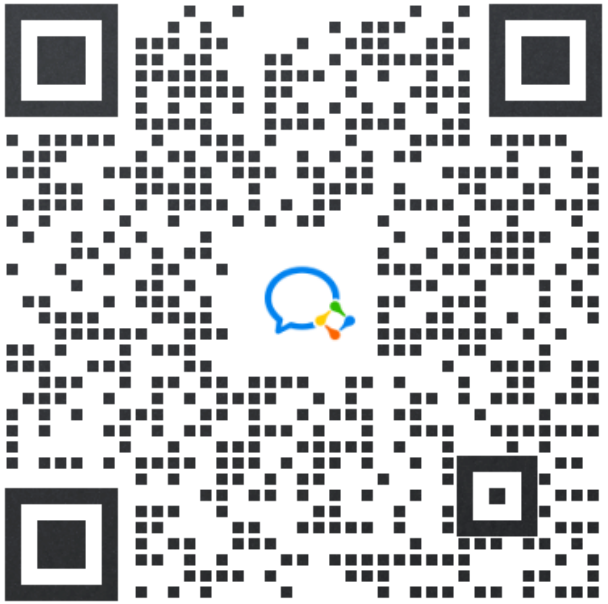
A: 1*22+4*(2+18)+8*(4+1) =142
B: 1*22+4*(2+18)+2*(1+7)+2*(1+1) =122
C: 1*22+4*(2+18)+2*(1+7)+3*(1.5+1)=125.5

- 问题描述
  - 如何构建拓扑，使得在Elmore模型进行缓冲器插入会更好修时序
- 该拓扑与真实绕线存在误差

总结

- iTO现有功能
  - DRV优化、Setup优化、Hold优化
- 未来计划
  - 多种优化方式结合
  - 缓冲树拓扑构建，更容易优化时序

iEDA开源交流群



感谢聆听

Thanks for your attention