iEDA-Tutorial 第五期

时钟树综合(iCTS)、布线(iRT)、设计规则检查(iDRC)

李伟国、曾智圣、刘继康、郭帆

2023/12/29

iEDA Tutorial 第五期议程

- **Part 1 iEDA-iCTS 问题、研究内容与计划（李伟国）**
- Part 2 iEDA-iRT RT整体与GR问题，研究内容与计划（曾智圣）
- Part 3 iEDA-iRT 详细布线问题、研究内容与计划 （刘继康）
- Part 4 iEDA-iDRC 问题、研究内容与计划 （郭帆）

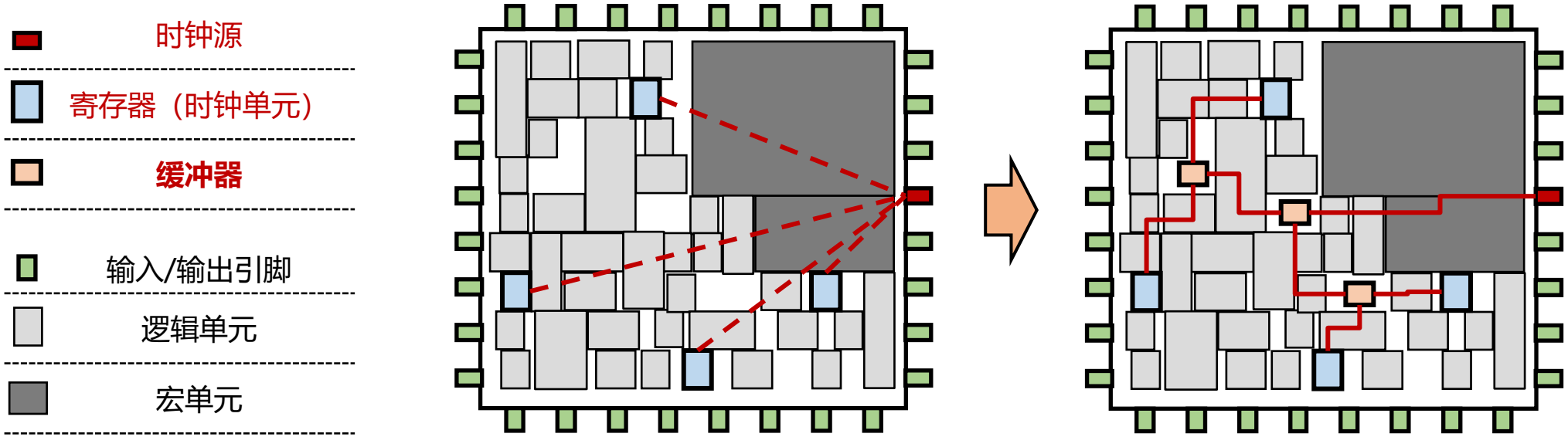
01 时钟树综合问题介绍

02 iCTS解决方案

03 未来计划

时钟树综合问题介绍

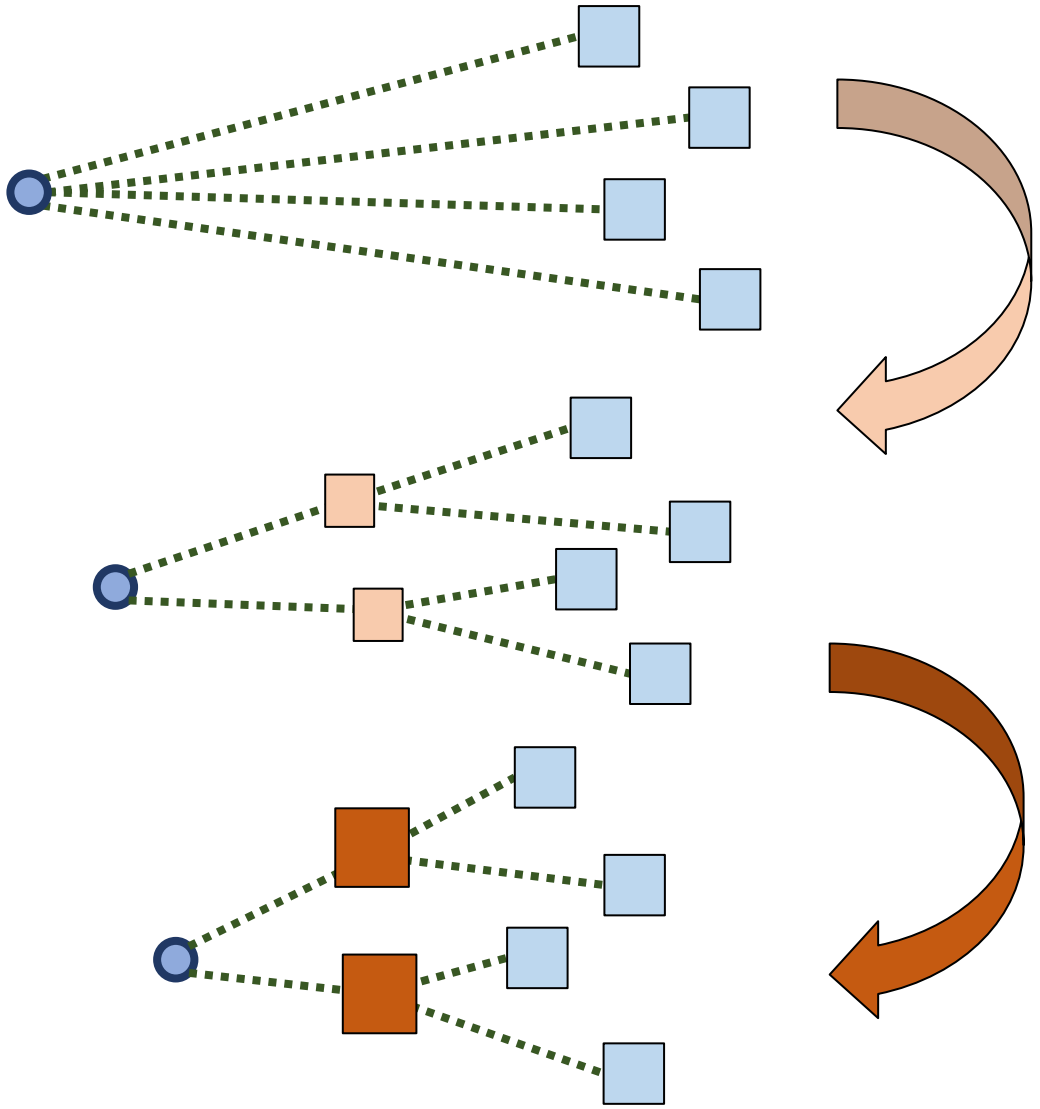
- 问题介绍
  - 给定时钟单元分布，构建时钟网络（树）以实现**时钟源**到达各个**寄存器**的时序需要，期间需满足给定的各类**约束**，并尽可能**节约设计资源**



  - **平衡**时钟源到寄存器的**路径时延**是时钟树综合基本目标
  - **缓冲方案**和**线网拓扑**是时钟树综合的关建

时钟树综合问题介绍

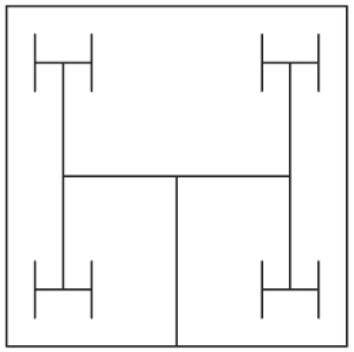
- 缓冲方案
  - 缓冲器的作用
    - 分解线网
    - 增强驱动能力
    - 修复时序、降低长线时延
  - 缓冲器的属性
    - 尺寸、面积
    - 功耗、驱动能力
    - 单元库



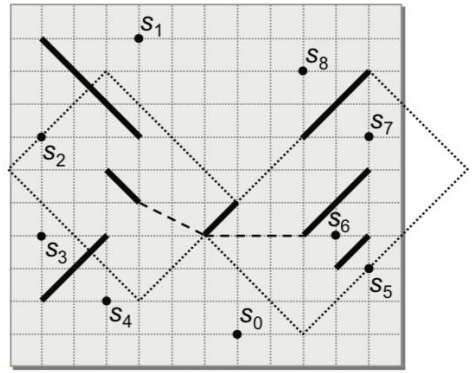
- 缓冲器插入
  - 减小扇出
  - 降低长线时延
  - 增加单元时延
- 缓冲器增强
  - 增强驱动能力
  - 增加单元面积
  - 增加功耗

时钟树综合问题介绍

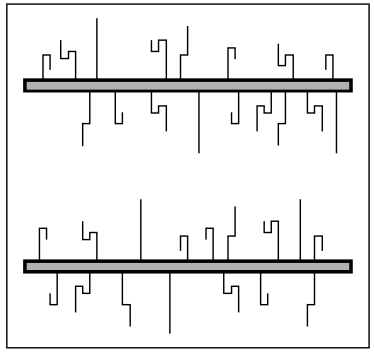
- 线网拓扑
  - 树形拓扑
    - H树[1]
    - 延迟合并嵌入[2]
  - 非树形拓扑
    - 鱼骨脊柱[3]
    - 网格型[4]
  - 混合拓扑
    - 网格-树[5]



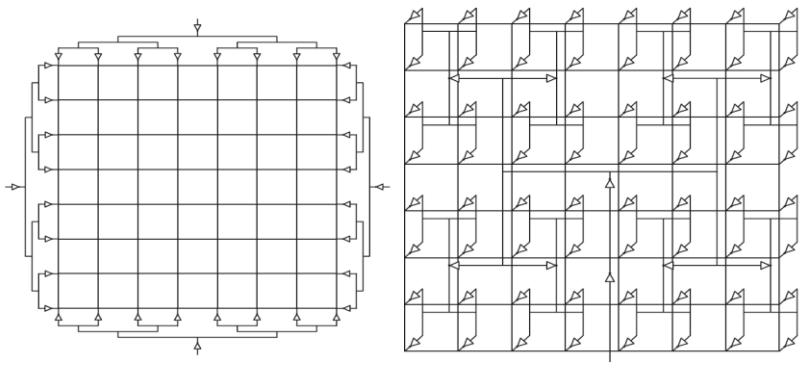
H树



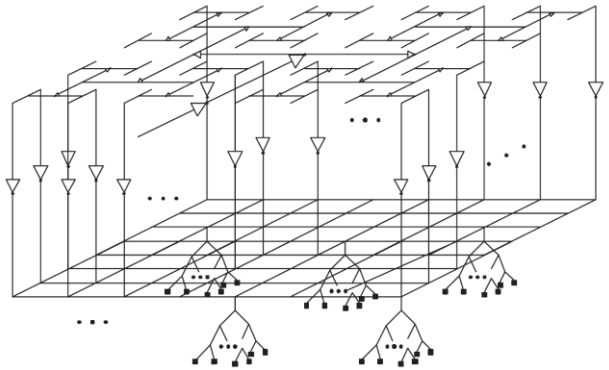
延迟合并嵌入



鱼骨脊柱



网格型



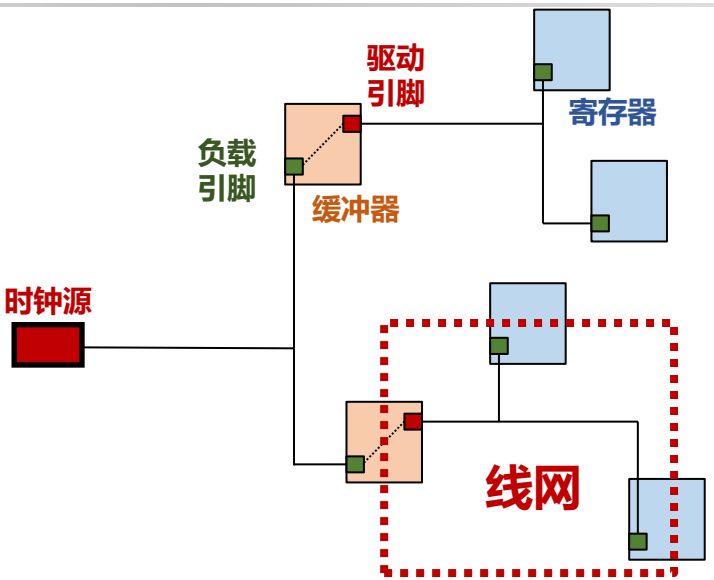
混合树

[1] Bakoglu H B. Circuits, interconnections, and packaging for VLSI[J]. 1990.
[2] Boese K D, Kahng A B. Zero-skew clock routing trees with minimum wirelength[C]. [1992] Proceedings. Fifth Annual IEEE International ASIC Conference and Exhibit, 1992: 17-21.
[3] Andreev A, Nikishin A, Gribok S, et al. Clock network fishbone architecture for a structured ASIC manufactured on a 28 NM CMOS process lithographic node: Google Patents, 2014.
[4] Chakrabarti P, Bhatt V, Hill D, et al. Clock mesh framework[C]. Thirteenth International Symposium on Quality Electronic Design (ISQED), 2012: 424-431.
[5] Abdelhadi A, Ginosar R, Kolodny A, et al. Timing–driven variation–aware synthesis of hybrid mesh/tree clock distribution networks[J]. Integration, 2013, 46(4): 382-391.

时钟树综合问题介绍

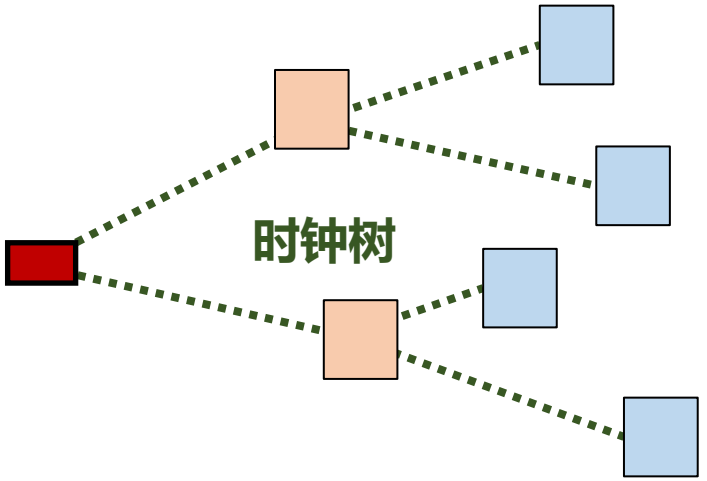
- **时钟树综合的结构**

  - **线网结构：**
    - 驱动引脚（Driver Pin）
    - 负载引脚（Load Pin）
    - **时序计算**的基本结构



线网结构信息	
负载电容	Cap
转换时间	Slew
线长	Wirelength
时延	Delay
扇出	Fanout

  - **树结构：**
    - 时钟源 → 缓冲器 → 寄存器
    - 抽象**时钟单元为节点**的树结构

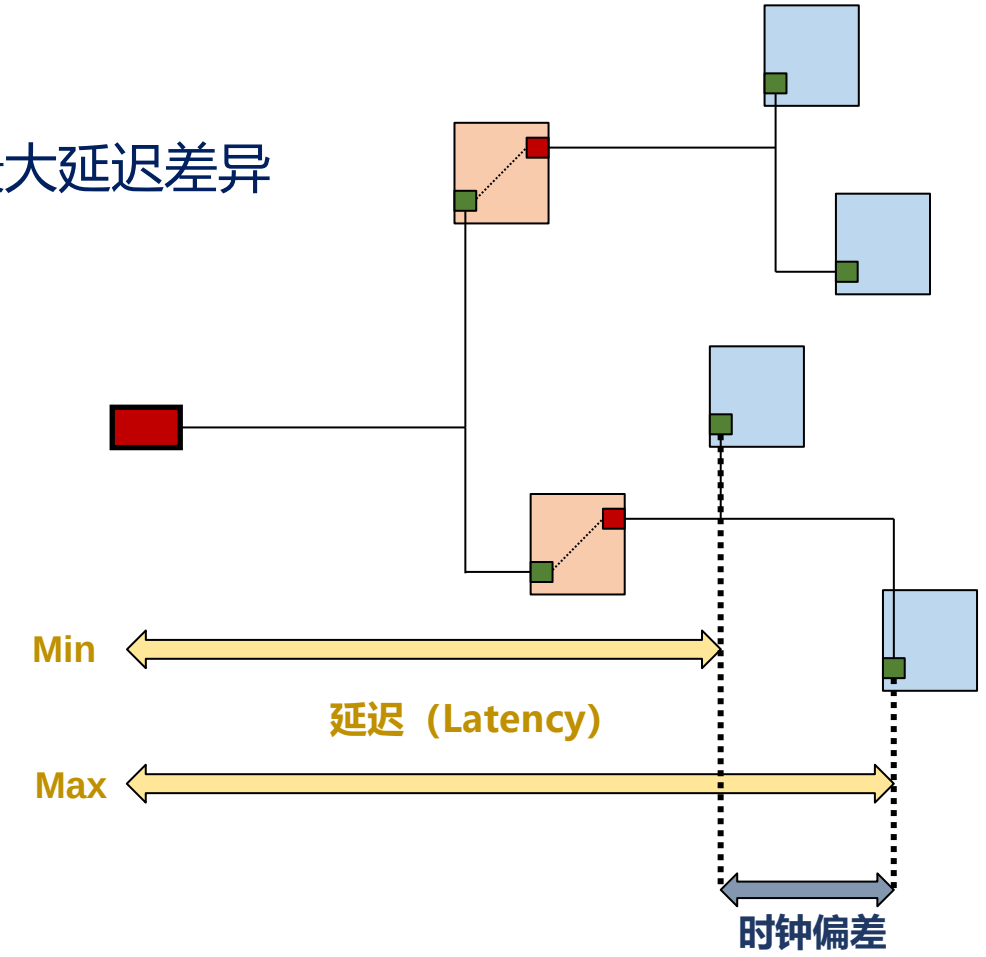


树结构信息	
时钟偏差	Skew
层次（深度）	Level
总线长	Total WL
延迟	Latency
总电容	Total Cap

时钟树综合问题介绍

- 问题概要

  - 约束
    - **时钟偏差（Skew）约束：**时钟源到达寄存器的最大延迟差异
    - 稳定性约束：
      - Cap：线网的最大**负载电容**
      - Slew：信号传播过程的最大**转换时间**
      - Fanout：线网的最大**扇出**
      - Wirelength：线网的最大**线长**
  - 操作
    - 构建缓冲方案
    - 构建各线网的拓扑
  - 目标
    - 总线长
    - 缓冲器数量、面积
    - 功耗、最大延迟



时钟树综合问题介绍

- **数学模型**

$\mathcal{S}$	时钟单元分布，$\forall s \in \mathcal{S}$表示寄存器单元
$lat(s)$	时钟源到单元s的延迟
$skew_{bound}$	时钟偏差约束
$\mathcal{L}$	缓冲器单元库，$\forall lib \in \mathcal{L}$表示某缓冲器的$library$信息，用于计算$cell\ delay$
$\mathcal{N}$	线网集合，$\forall net \in \mathcal{N}$，其时序计算依赖于驱动单元的$library$信息
$\mathcal{C}$	设计约束评估集合，$\forall C(net) \in \mathcal{C}$，$C(net): \mathcal{N} \to R$，代表线网的某类违例度量
$P(net)$	功耗评估（设计质量）函数，$P(net): \mathcal{N} \to R$，表示对线网的功耗评估

- **模型表征**

给定时钟单元分布$\mathcal{S}$、设计约束集$\mathcal{E}$和缓冲器单元库$\mathcal{L}$，构建**缓冲方案产生**若干线网$\mathcal{N}$，并**生成线网拓扑**使得时钟偏差约束成立：

$$\max_{\forall s_i, s_j \in \mathcal{S}} \left\{ \left| lat(s_i) - lat(s_j) \right| \right\} \leq skew_{bound}$$

并最小化设计成本：

$$\min \sum_{\forall net \in \mathcal{N}} P(net) + \lambda \cdot \sum_{\substack{\forall net \in \mathcal{N} \\ \forall C(x) \in \mathcal{E}}} C(net)$$

01 时钟树综合问题介绍

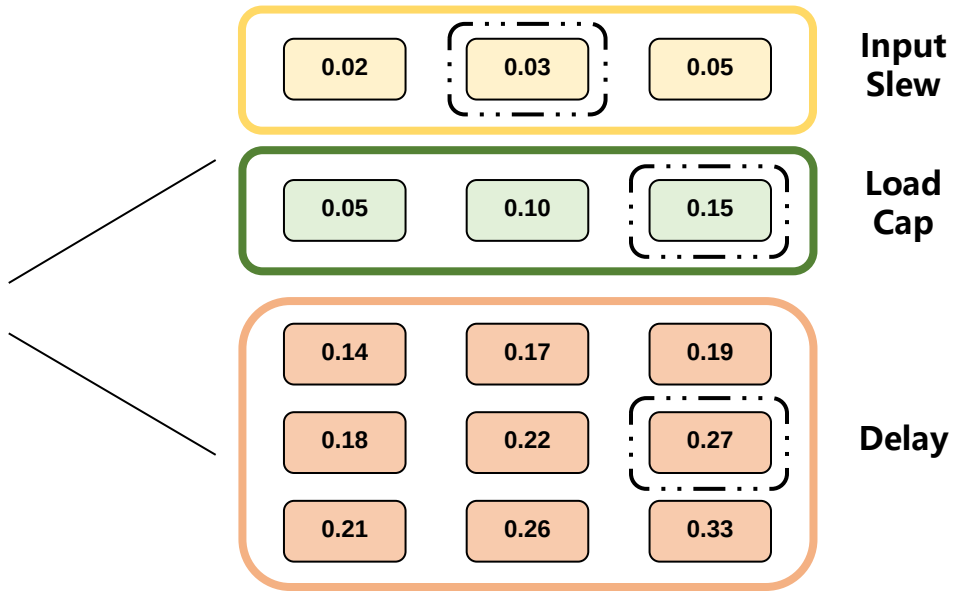
02 iCTS解决方案

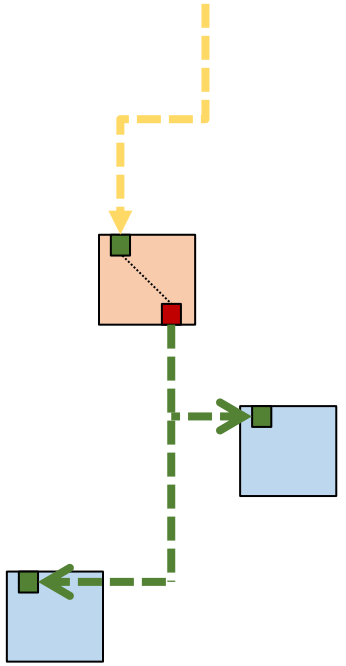
03 未来计划

时钟树综合问题的关键

- 缓冲器时延计算
  - 上游的转换时间（Slew）
  - 下游的负载电容（Cap）

```
timing() {
    ...
     "Slew" : [...],
     "Cap" : [...],
     "Delay" : [..., ..., ...],
    ...
}
```

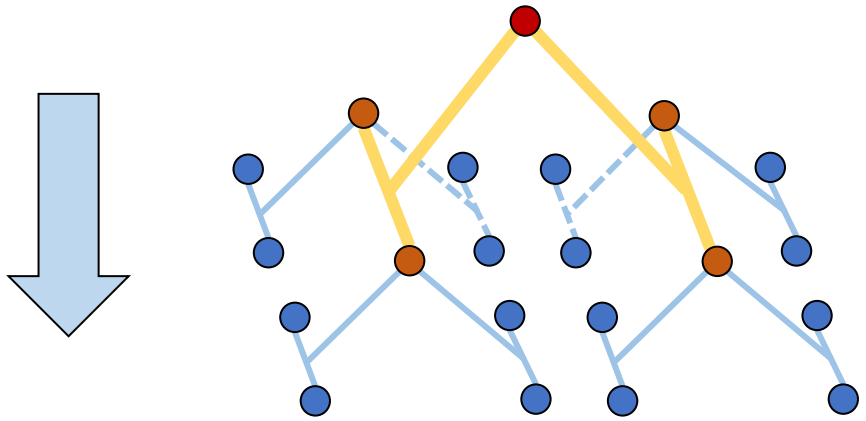


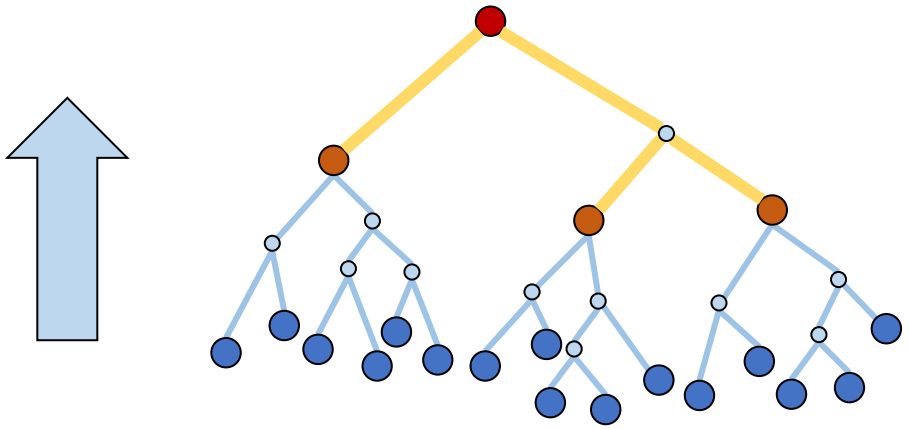


当上游（下游）时序信息未知时难以计算时延

时钟树综合问题的难点

- **构建方式**
  - 自上而下
    - 重要模块：H树、分支
    - 能够获取较为准确的上游转换时间
    - 时钟**单元分布不平衡**时效果较差
    - **设计成本较大**（线长、缓冲器插入数量）
  - 自下而上
    - 重要模块：聚类、划分、合并时钟树
    - 时序计算存在误差，**设计难度大**
    - 时钟偏差的**控制难度随着规模增大而增大**
    - 设计成本往往较小





向下构建无法兼顾分布和设计成本，向上构建存在时序误差且控制偏差难度大

iCTS解决的问题

- **层次化设计约束的时钟树综合问题：**

  在层次k中，给定对应层次的时钟单元分布$\mathcal{S}^k$、设计约束集$\mathcal{E}^k$和缓冲器单元库$\mathcal{L}$，构建缓冲方案产生若干线网$\mathcal{N}^k$，并生成线网拓扑使得**层次化时钟偏差约束**成立：

$$\max_{\forall s_i^k, s_j^k \in \mathcal{S}^k} \left\{ \left| lat\left(s_i^k\right) - lat\left(s_j^k\right) \right| \right\} \leq skew_{bound}^k$$

  每一层次k中最小化设计成本和违例代价：

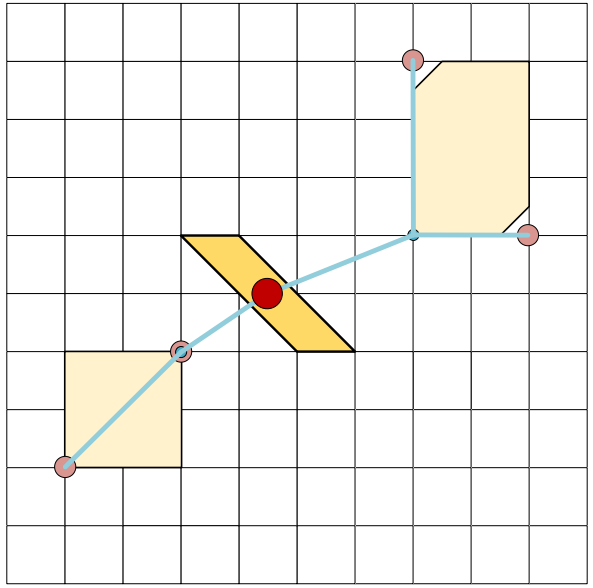
$$\min \sum_{\forall net \in \mathcal{N}^k} P(net) + \lambda \cdot \sum_{\substack{\forall net \in \mathcal{N}^k \\ \forall C(x) \in \mathcal{E}^k}} C(net)$$

  其中$\mathcal{N}_k$中的驱动单元将作为层次$k+1$的时钟单元输入

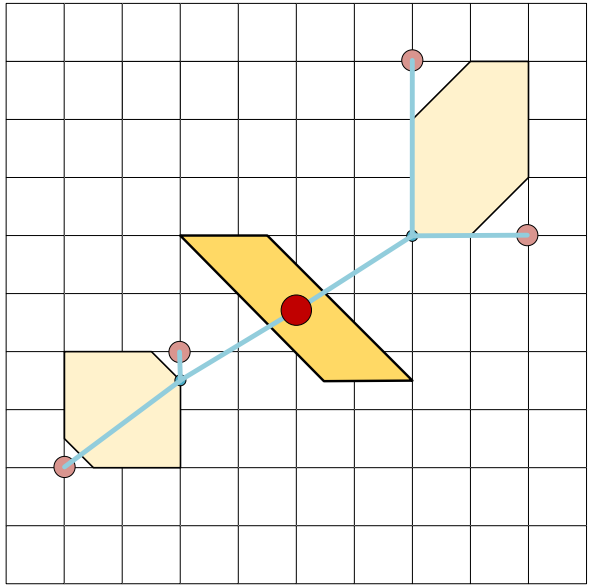
iCTS解决的问题

- **层次化设计的动机：**

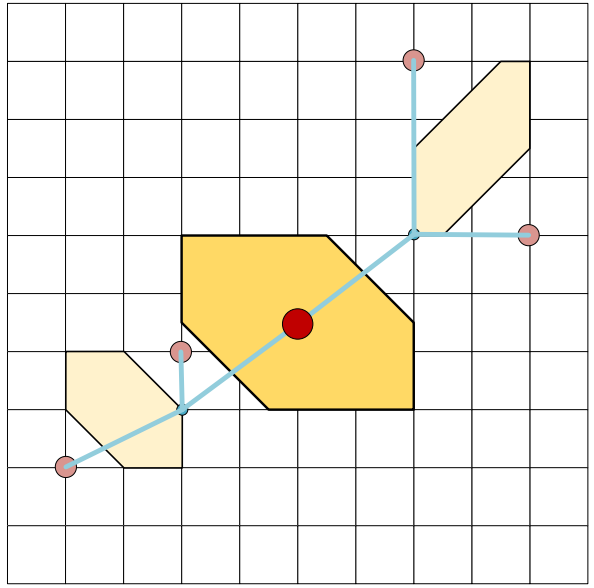
解空间余量，表示下游合并过程留给上层时钟可行合并区域的面积大小



Bottom Skew: 4 WL: 15
Root Skew: 4 Root Area: 2



Bottom Skew: 3 WL: 15.5
Root Skew: 4 Root Area: 3.75



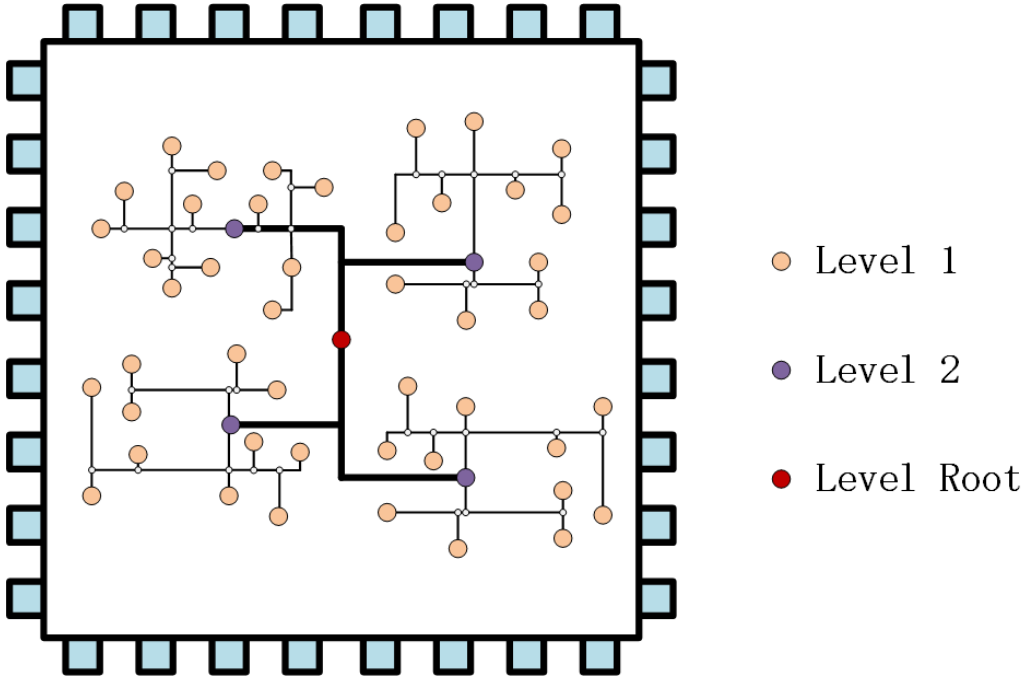
Bottom Skew: 2 WL: 16
Root Skew: 4 Root Area: 11.25

在这个例子中，若下游线网具有**较小**的$skew$，则下一阶段时钟树合并的解空间**更大**，从而更好实现**避障、解空间探索**等需求，代价是**较小**的额外线长

iCTS解决的问题

- **层次化设计的动机：**

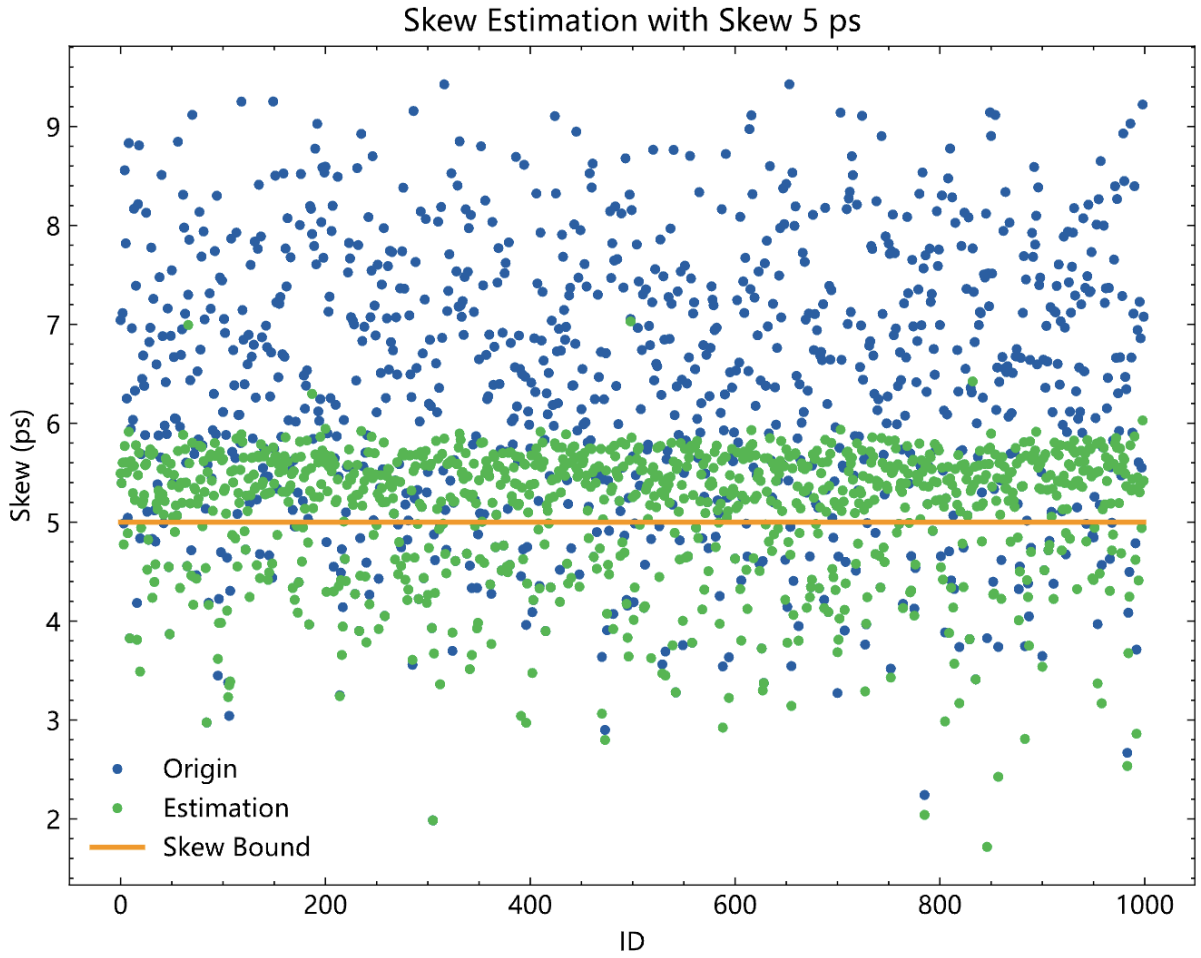
Constraint	Level 1	Level 2	Level Root
$Skew$	0.02	0.04	0.08
$Fanout$	16	4	--
$Wirelength$	200	300	--
Cap	0.05	0.15	--



除了对$skew$的层次控制外，我们也对线网约束进行层次化设计

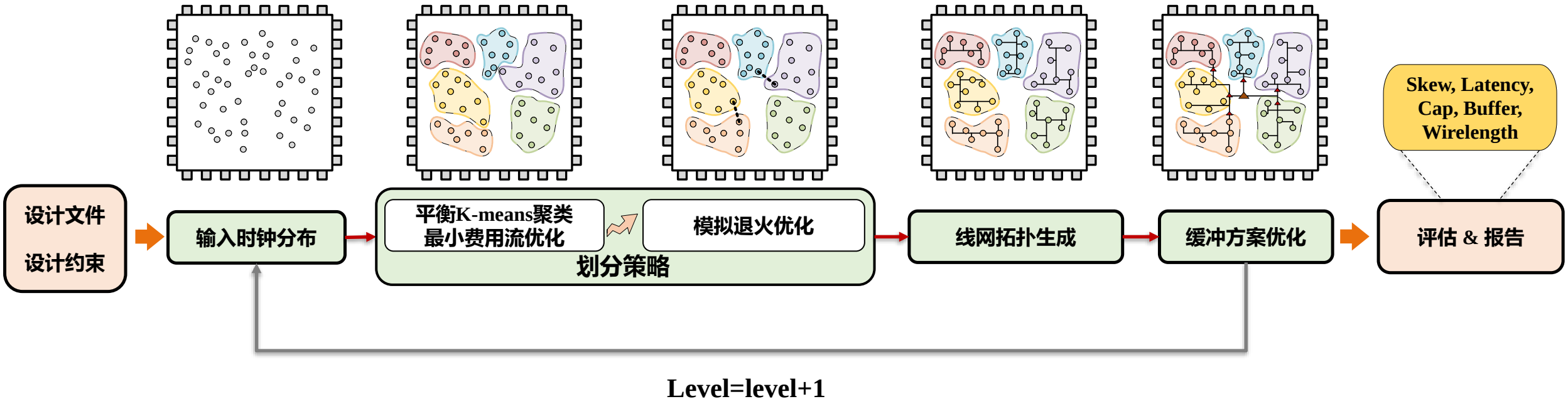
iCTS解决的问题

- 如何避免误差
  - 层次化设计创造更大解空间
  - 下界预估模型：
    - 利用负载电容和工艺库信息
    - 计算插入时延下界
    - 为负载引脚增加预估时延
    - 时钟偏差差异从26.84%降低至2.856%



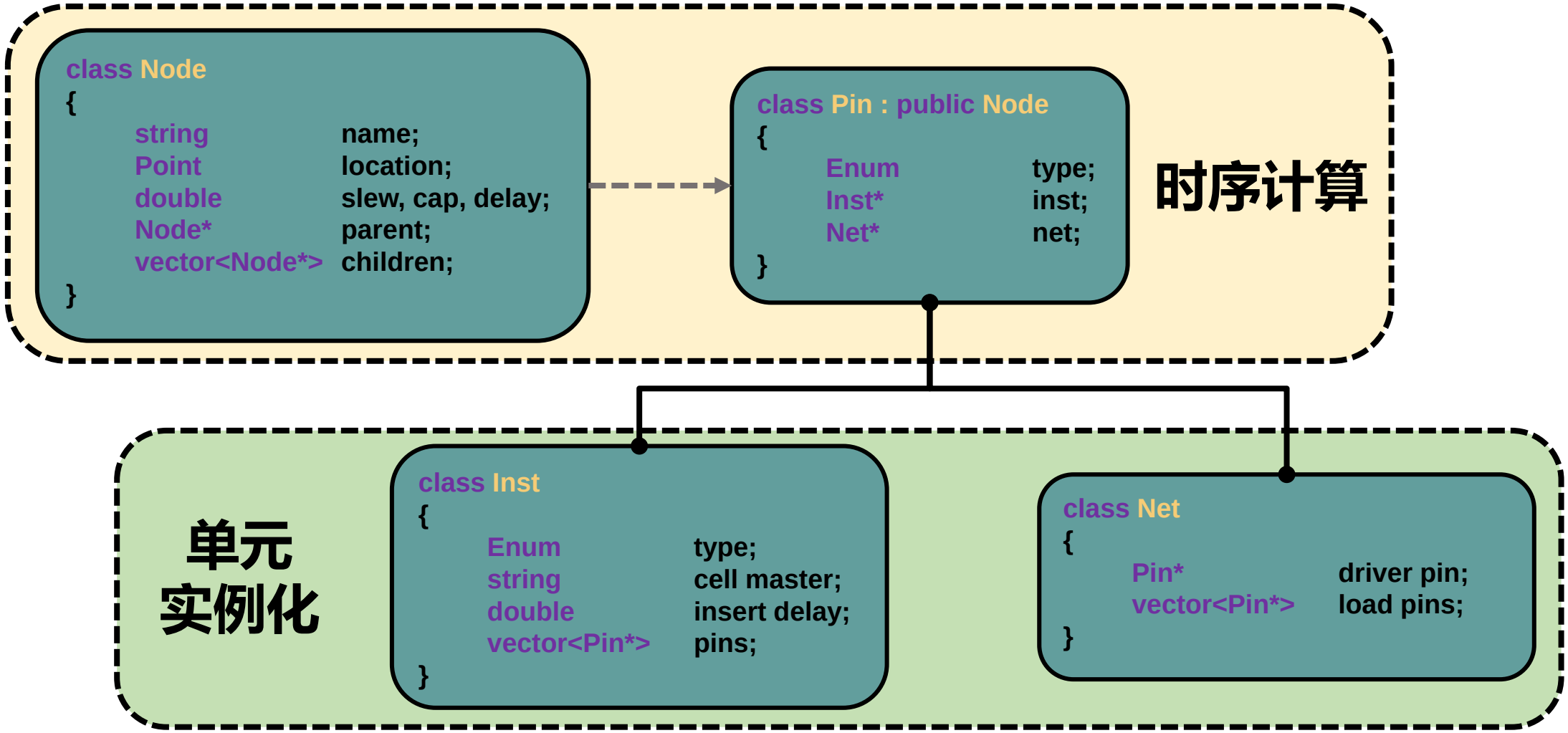
iCTS的解决方案

- 解决方案流程



iCTS的解决方案

- 数据结构



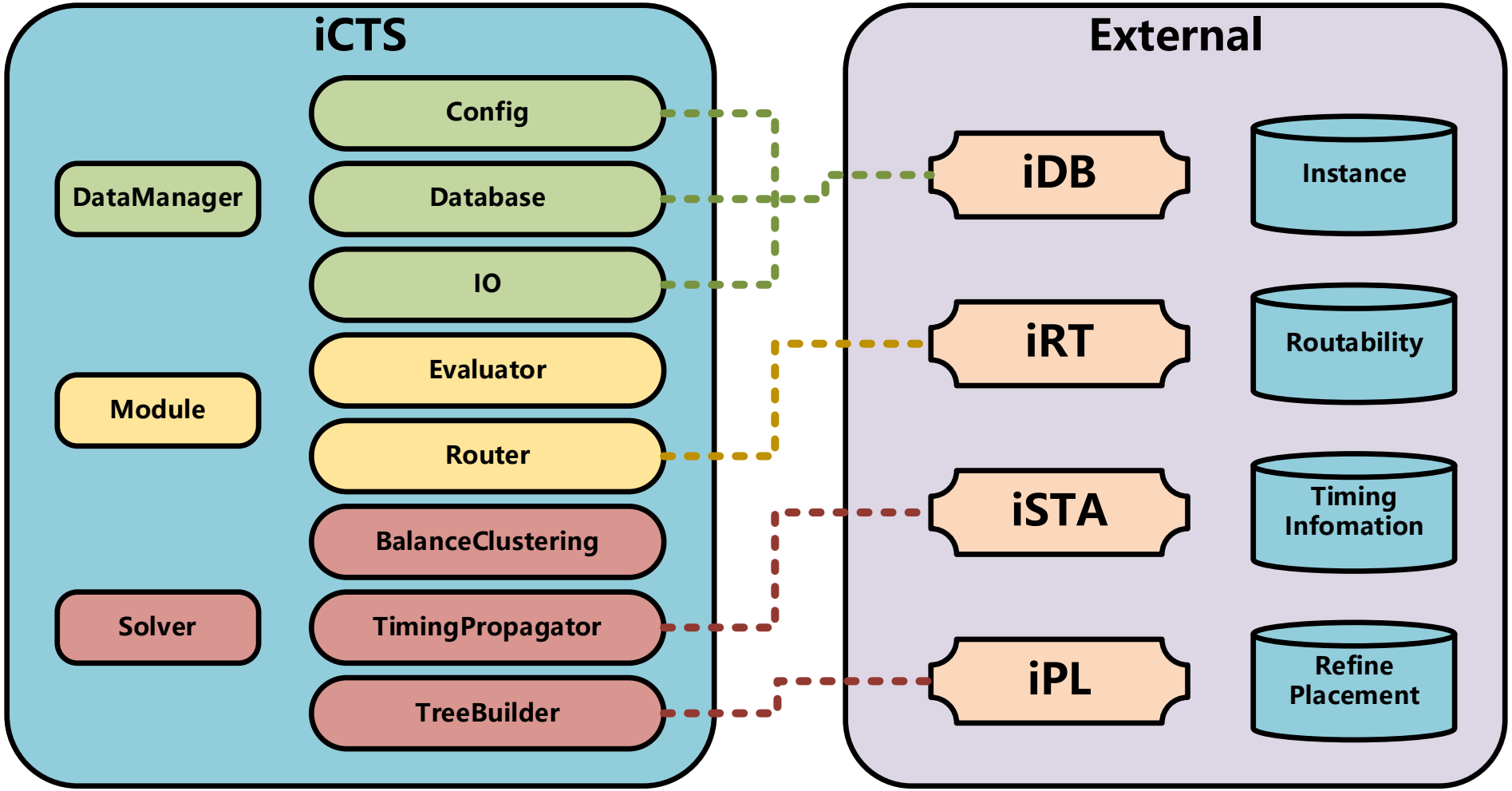
iCTS的解决方案

- **算法结构**

模块	功能	说明
聚类方法 BalanceClustering	K-means平衡聚类	电容、时延的自适应聚类
	最小费用流优化(MCF)	基于扇出约束的解空间剪枝
	模拟退火优化(SA)	综合设计成本和约束代价的解空间探索
时序计算 TimingPropagator	Wire Delay	Elmore模型
	Cell Delay	查表 & 双线性插值
	Capacitance	单位电容换算
	Slew	PERI模型
时钟树构建 TreeBuilder	FLUTE	快速线网布线方案，该方法的目标是实现最小的总线长
	BST-DME	有界偏差时钟树标准构建方案，基于给定的时钟偏差构建时钟树（内置多种拓扑生成方法）
	SALT	更近似Routing阶段的构建方案，能够实现更小的线网时延
	CBS	联合BST-DME和SALT的构建方案，在保证时钟偏差的控制下以更小的设计资源实现时钟树构建

iCTS的解决方案

- 软件架构



iCTS的解决方案

- 报告 & 评估
  - 缓冲方案
    - 数量、面积、分布
  - 线网拓扑
    - 总线长、半周线长情况、分布
  - 设计约束
    - 多类设计约束的统计信息
    - 违例数量、分布情况统计

Cell type	Count	Area	Capacitance
Buffer	574	413.91	0.45362

Name	Type	Inst Count	Inst Area (um^2)
BUFFD16BWP30P140LVT	Buffer	1	2.772
BUFFD2BWP30P140LVT	Buffer	4	2.016
BUFFD3BWP30P140LVT	Buffer	371	233.73
BUFFD4BWP30P140LVT	Buffer	195	171.99
BUFFD6BWP30P140LVT	Buffer	3	3.402

Type	Wire Length
Top	197.645
Trunk	589.764
Leaf	3069.801
Total	3857.211
Max net length	197.645

Type	HP Wire Length
Top	161.321
Trunk	384.282
Leaf	1356.420
Total	1902.022
Max net length	161.321

Level	Inst Num	Min Skew	Max Skew	Avg Skew	Violation
1	529	0.000235225	0.00230041	0.000684056	0
2	55	0.00216993	0.0186049	0.0122996	0
3	11	0.0171523	0.0438948	0.0284401	0
4	2	0.0306581	0.0544427	0.0425504	0

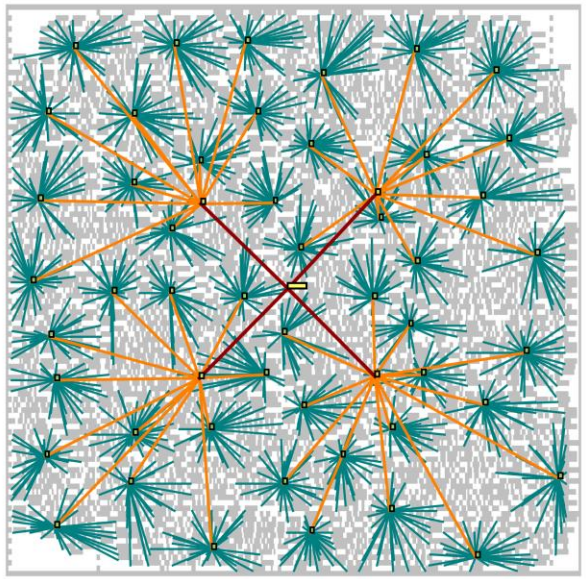
iCTS的解决方案

- 实验结果

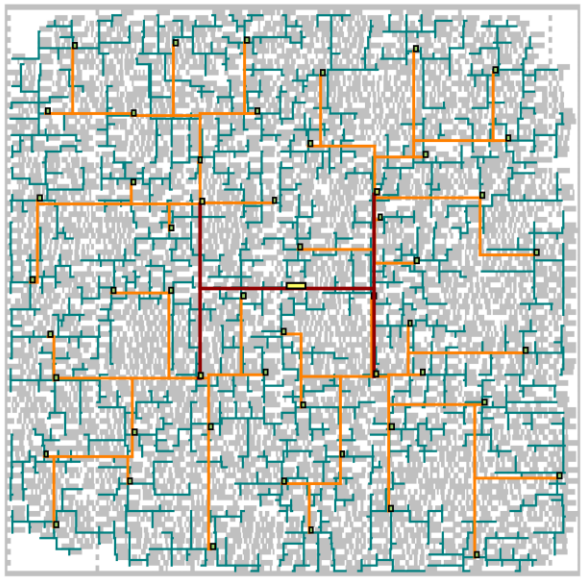
Case	Insts Num	FFs Num	Util	Flow	Max Latency	Skew	Buffers	Buf Area	Clk Cap	Clk WL	Runtime (s)
s38584	7510	1248	0.88	Ours	**71**	13	**43**	**26.586**	**904**	3382.049	**13**
				Commercial	76	**8**	**43**	26.838	1019	**3366.545**	326
				OpenROAD	93	17	45	39.69	1087	3478.911	52
s38417	6428	1564	0.881	Ours	**75**	**10**	**53**	**32.382**	**1083**	**3755.581**	**14**
				Commercial	84	19	54	33.642	1235	3867.445	357
				OpenROAD	94	16	55	48.51	1284	3870.452	47
s35932	6113	1728	0.887	Ours	**80**	13	**58**	**35.532**	**1217**	**4380.049**	**15**
				Commercial	81	**10**	59	36.54	1380	4420.92	334
				OpenROAD	100	17	64	56.448	1433	4449.412	52
salsa20	13706	2375	0.947	Ours	**82**	**19**	**81**	**49.644**	**1715**	**6446.891**	**17**
				Commercial	87	21	83	51.786	2050	6580.935	390
				OpenROAD	112	29	109	96.138	2160	6863.521	54
ethernet	39945	10015	0.835	Ours	**97**	34	**337**	**315.504**	**7314**	26113.92	**30**
				Commercial	104	**31**	352	320.418	8823	**26105.5**	416
				OpenROAD	159	51	455	408.87	9210	27248.841	50
vga_lcd	60541	16902	0.739	Ours	**134**	**41**	**575**	**416.934**	**12380**	46763.071	**55**
				Commercial	146	49	597	451.836	14920	**45969.845**	1271
				OpenROAD	206	92	775	812.07	15815	47484.091	59
Total	--	--	--	Ours	**539**	**130**	**1147**	**876.582**	**24613**	90841.561	**144**
				Commercial	578	138	1188	921.06	29427	**90311.19**	3094
				OpenROAD	764	222	1503	1461.726	30989	93395.228	314
				Incr. by Commercial	**6.75%**	**5.80%**	**3.45%**	**4.83%**	**16.36%**	**-0.59%**	**95.35%**
				Incr. by OpenROAD	**29.45%**	**41.44%**	**23.69%**	**40.03%**	**20.58%**	**2.73%**	**54.14%**

iCTS的解决方案

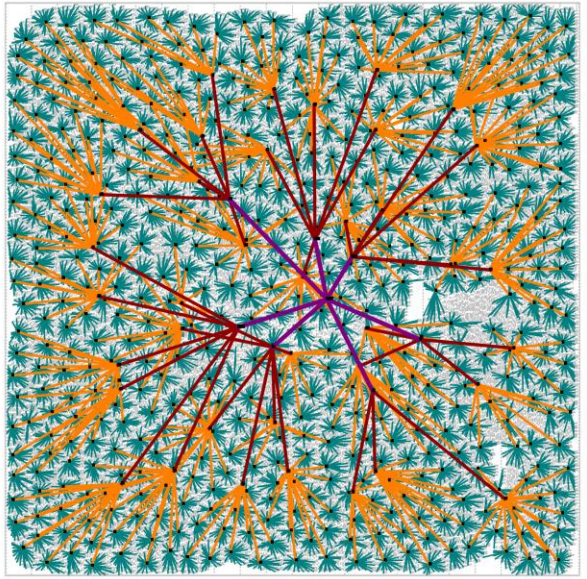
- Topology & Routing



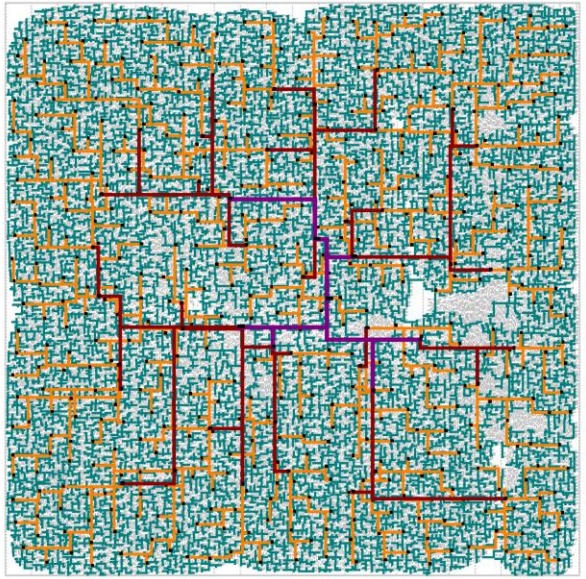
Topology of s38417



Routing of s38417



Topology of vga_lcd



Routing of vga_lcd

01 时钟树综合问题介绍

02 iCTS解决方案

03 未来计划

iCTS未来计划

- 时序计算改进
- Useful Skew的利用和优化
- 与PL的联合优化（CCOPT）
- 更多时钟树构建方法以及混合树框架
- 适用于时钟树综合的解空间高效探索方法

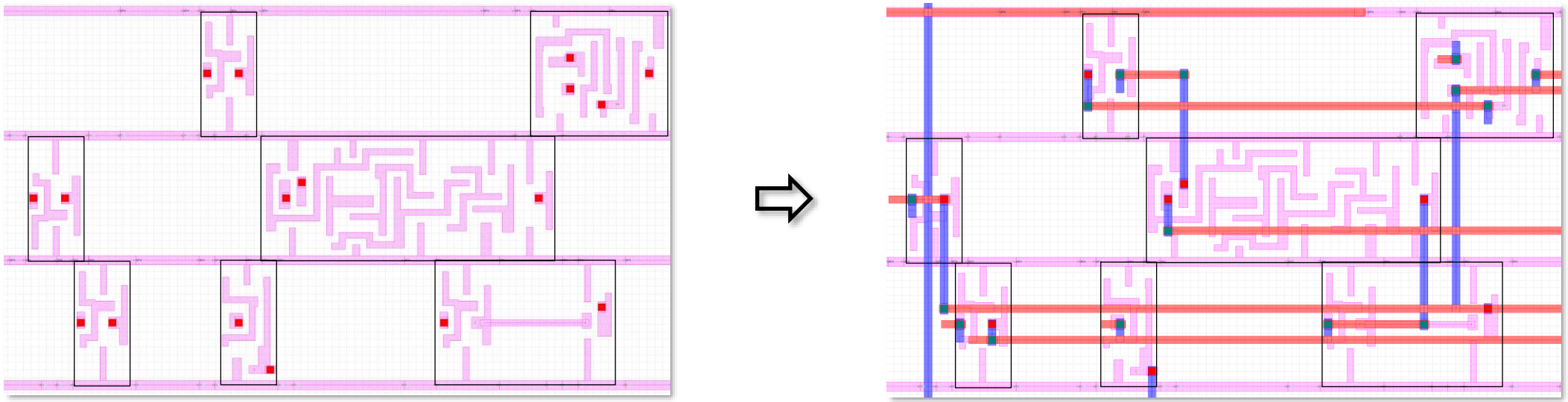
iEDA Tutorial 第五期议程

- Part 1 iEDA-iCTS 问题、研究内容与计划（李伟国）
- **Part 2 iEDA-iRT RT整体，GR问题，研究内容与计划（曾智圣）**
- Part 3 iEDA-iRT 详细布线问题、研究内容与计划 （刘继康）
- Part 4 iEDA-iDRC 问题、研究内容与计划 （郭帆）

布线问题

- **布线问题介绍**

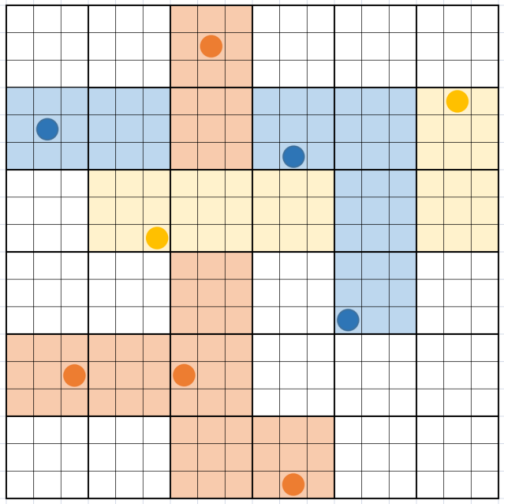
给定连接关系（线网）和一定的约束条件（如连通性，设计规则等），构建所有连接关系的最优（如线长，通孔，时序，信号完整性等）直角斯坦纳森林。



布线问题

- **布线划分两个子问题**

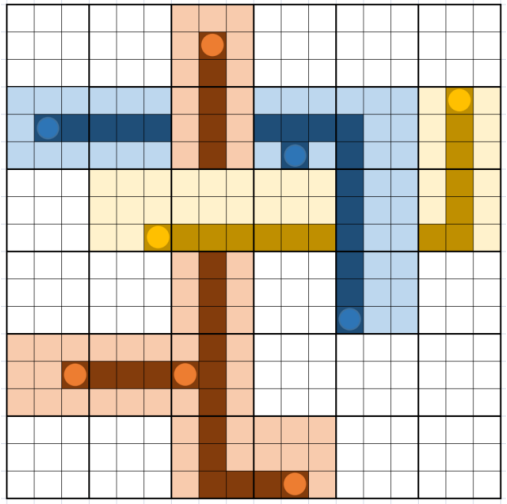
由于电路规模庞大（百万，千万条），布线问题复杂，直接求解时间复杂度高，所以将布线分为全局布线和详细布线。



● Pin

GR Grid

将设计以网格(GCell)划分，并以网格为单元布线

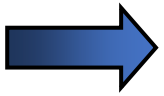


● Pin

DR Grid

在GR网格内进行更细粒度的绕线，并保证没有设计规则违例

全局布线
(Global Routing)



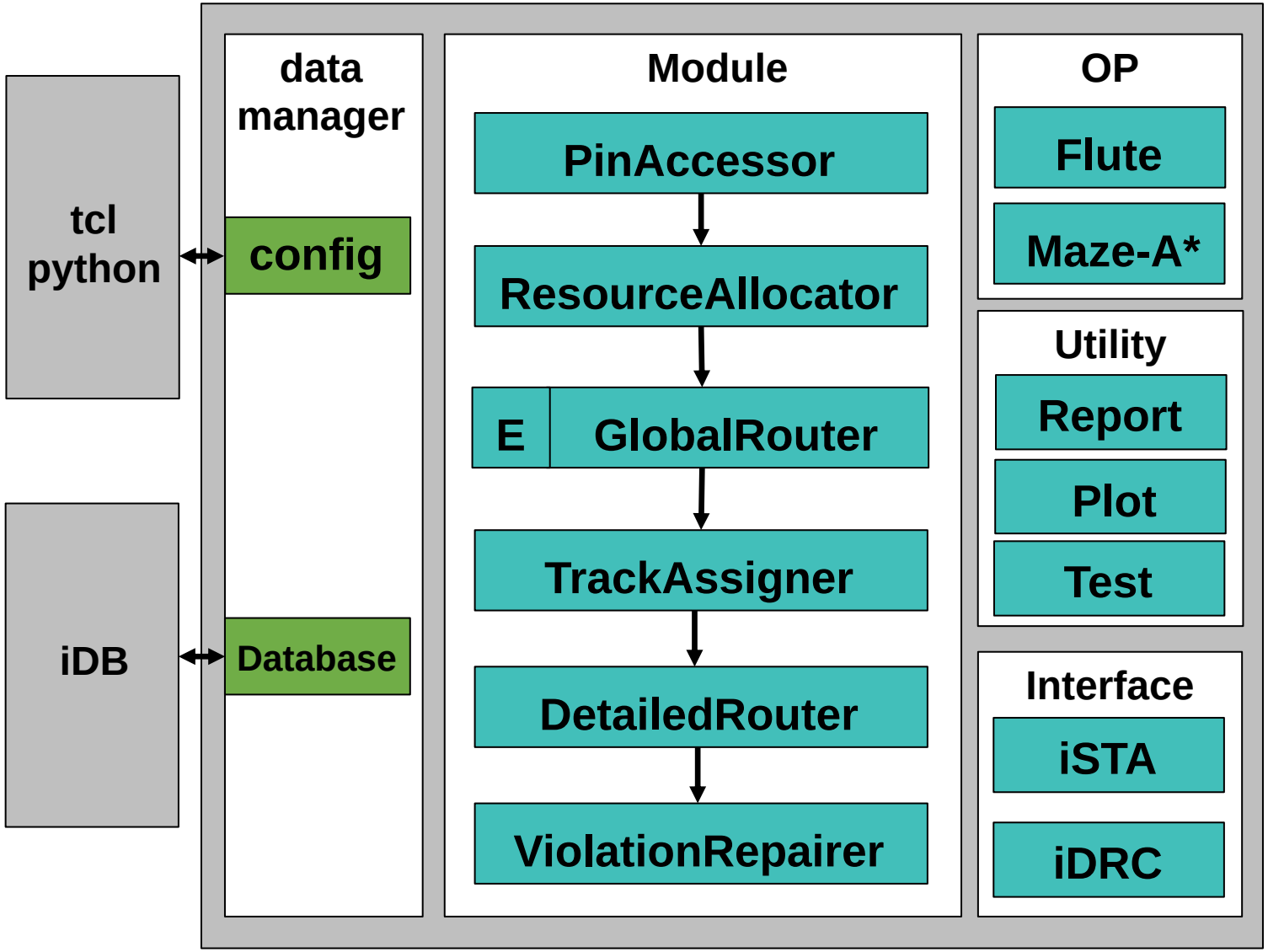
详细布线
(Detail Routing)

01 研究问题

02 研究内容

03 未来展望

iRT 概述



DataManager：iRT顶层数据管理器

- Config：通过tcl，python等脚本语言配置
- Database：数据由顶层数据库iDB获取

Module：iRT主要流程模块

- PinAccessor：pin接入器，在pin shape上找到合理的接入点
- ResourceAllocator：资源分配器，用于在布线前给线网分配资源
- EGlobalRouter：早期全局布线器，快速获取当前布局的拥塞
- GlobalRouter：全局布线器，在三维网格上通过寻路算法布线
- TrackAssigner：轨道分配器，将线网结果分配在布线轨道上
- DetailedRouter：详细布线器，在更细粒度的网格上进行三维布线

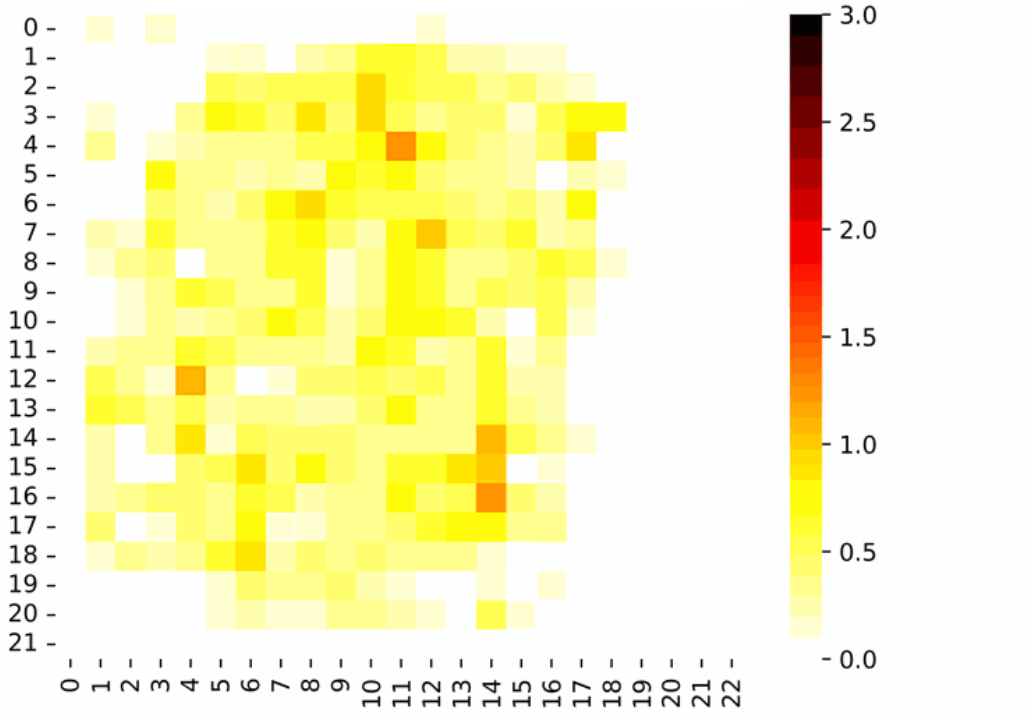
OP：iRT算子模块

- Flute：通过查表法快速生成候选斯坦纳树
- Maze-A*：A*寻路算法

iRT 概述

Utility：iRT公共库
- Report：报告
- Plot：布线可视化
- Test：测试

Interface：iRT接口库
- iSTA：时序评估
- iDRC：DRC评估



拥塞消除

线长
通孔数

Routing Layer	Prefer Wire Length	Nonprefer Wire Length	Wire Length / um	Cut Layer	Via number
M1	7512.39	314.885	7827.27	CO	0(0%)
M2	84962.9	941.465	85904.4	VIA1	191704(20.8551%)
M3	192310	383.63	192694	VIA2	338269(36.7996%)
M4	179986	405.97	180392	VIA3	229576(24.9751%)
M5	79713.2	167.662	79880.8	VIA4	88601(9.63874%)
M6	58483.3	141.4	58624.7	VIA5	46284(5.03515%)
M7	23884.5	257.135	24141.7	VIA6	24784(2.6962%)
M8	0	0	0	VIA7	0(0%)
M9	0	0	0	VIA8	0(0%)
AP	0	0	0	RV	0(0%)
Total	626853	2612.15	629465	Total	919218

DRC违例

blockage				net_shape			
Layer\DRC	Metal Short	Metal Spacing	Total	Layer\DRC	Cut Short	Total	
M1	280	231	511	CO	0	0	
M2	81	125	206	VIA1	54	54	
M3	0	0	0	VIA2	534	534	
M4	0	1	1	VIA3	477	477	
M5	0	0	0	VIA4	495	495	
M6	0	0	0	VIA5	211	211	
M7	0	0	0	VIA6	25	25	
M8	0	0	0	VIA7	0	0	
M9	0	0	0	VIA8	0	0	
AP	0	0	0	RV	0	0	
Total	361	357	718	Total	1796	1796	

时序评估

Endpoint	Clock Group	Delay Type	Path Delay	Path Required	CPPR	Slack	Freq(MHz)
chiplink_rx_tx_a_reg_12_:D	CLK_chiplink_rx_clk	max	5.903f	39.862	0.000	33.958	165.512
chiplink_rx_tx_a_reg_1_:D	CLK_chiplink_rx_clk	max	5.814f	39.863	0.000	34.049	168.040
chiplink_rx_tx_a_reg_3_:D	CLK_chiplink_rx_clk	max	5.813f	39.865	0.000	34.053	168.141
core_regs_2_reg_21_:D	CLK_osc_in	max	28.294f	9.643	0.000	-18.651	34.903
core_regs_2_reg_26_:D	CLK_osc_in	max	28.225f	9.652	0.000	-18.572	34.999
core_regs_2_reg_48_:D	CLK_osc_in	max	28.217f	9.657	0.000	-18.560	35.014
chiplink_rx_b2c_data_reg_31_:D	CLK_chiplink_rx_clk	min	0.161r	0.010	0.000	0.151	NA
chiplink_rx_b2c_data_reg_16_:D	CLK_chiplink_rx_clk	min	0.161r	0.010	0.000	0.151	NA
chiplink_rx_b2c_data_reg_2_:D	CLK_chiplink_rx_clk	min	0.161r	0.010	0.000	0.151	NA
chiplink_tx__T_401_3_reg_6_:D	CLK_osc_in	min	0.174r	0.003	0.000	0.171	NA
chiplink_tx__T_401_5_reg_16_:D	CLK_osc_in	min	0.174r	0.003	0.000	0.171	NA
chiplink_tx__T_401_3_reg_26_:D	CLK_osc_in	min	0.174r	0.003	0.000	0.171	NA

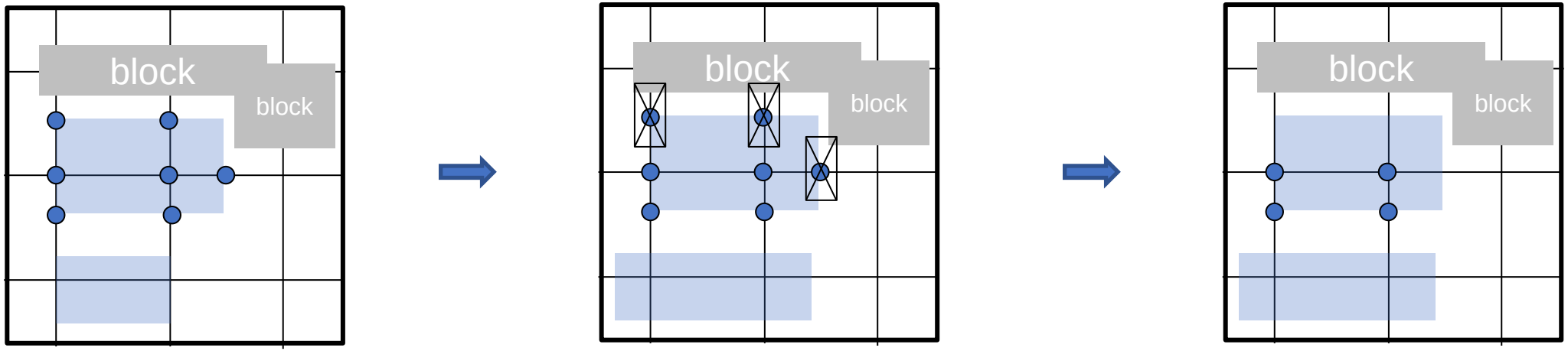
iRT 概述

Method	Argument	Description
init_rt	temp_directory_path	临时输出文件夹
	log_level	日志等级
	thread_number	允许的最大线程数
	bottom_routing_layer	最低布线层
	top_routing_layer	最高布线层
	gcell_pitch_size	gcell的size，以pitch为单位
	supply_utilization_rate	gcell内资源的利用率
run_rt	-flow	指定运行模块，"pa ra gr ta dr vr"
run_egr	temp_directory_path	临时输出文件夹
	congestion_cell_x_pitch	指定gcell的x方向上的size
	congestion_cell_y_pitch	指定gcell的y方向上的size
	bottom_routing_layer	最低布线层
	top_routing_layer	最高布线层
	accuracy	布线的精确值，控制不同的布线策略
report_rt_timing	-stage	输出布线结果的时序，"gr" or "dr"

PinAccessor

- 初始化单元连接点(access point)

首先，生成所有类型的单元连接点，之后对这些连接点进行合法性检查，包括与环境中障碍物、其它连接单元之间的检查。



生成连接点

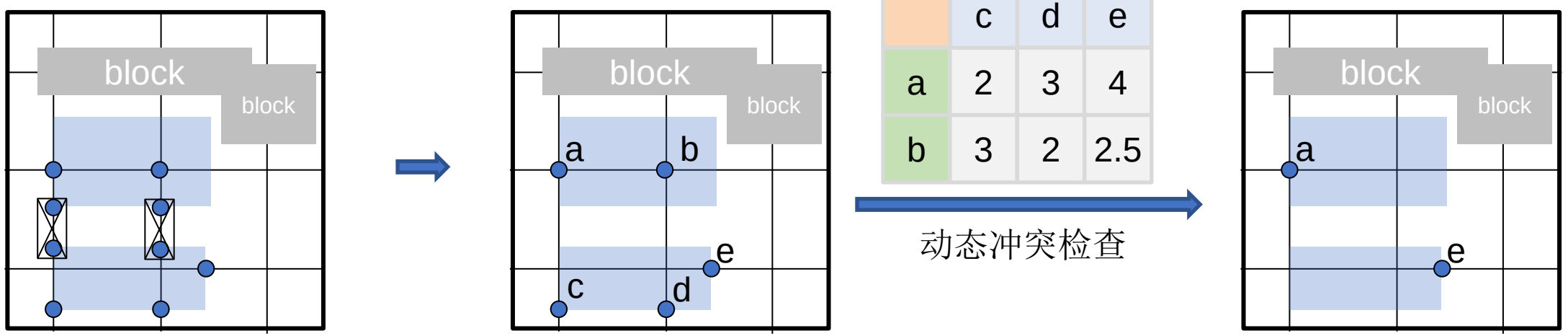
检查ap与block或者其他pin_shape之间的重叠

删除所有违例ap

PinAccessor

- 消除通孔冲突

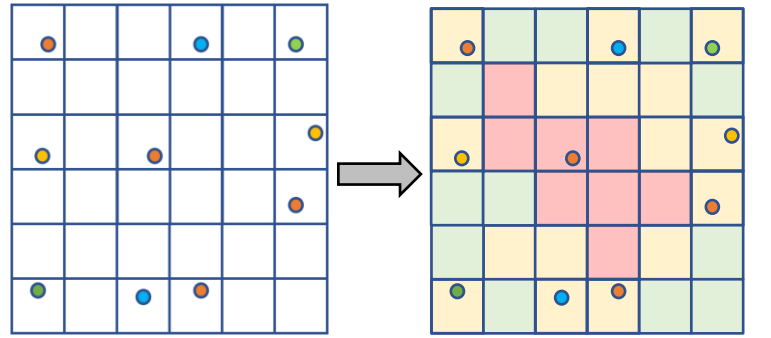
在生成每个pin的ap点时，并不能看到其他的pin的ap点，在连接过程中，可能会出现pin与pin之间的ap有冲突，通过静态冲突检查过滤,再通过冲突图，选出所有pin中最优的ap组合。



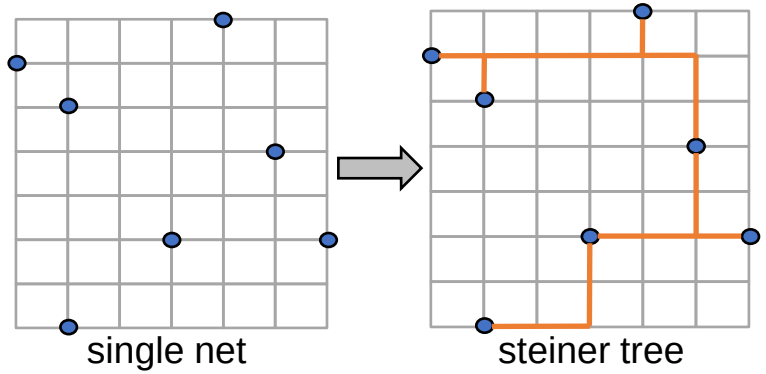
全局布线的问题

全局布线主要挑战

①降低版图绕线拥塞



②生成最优直角斯坦纳树

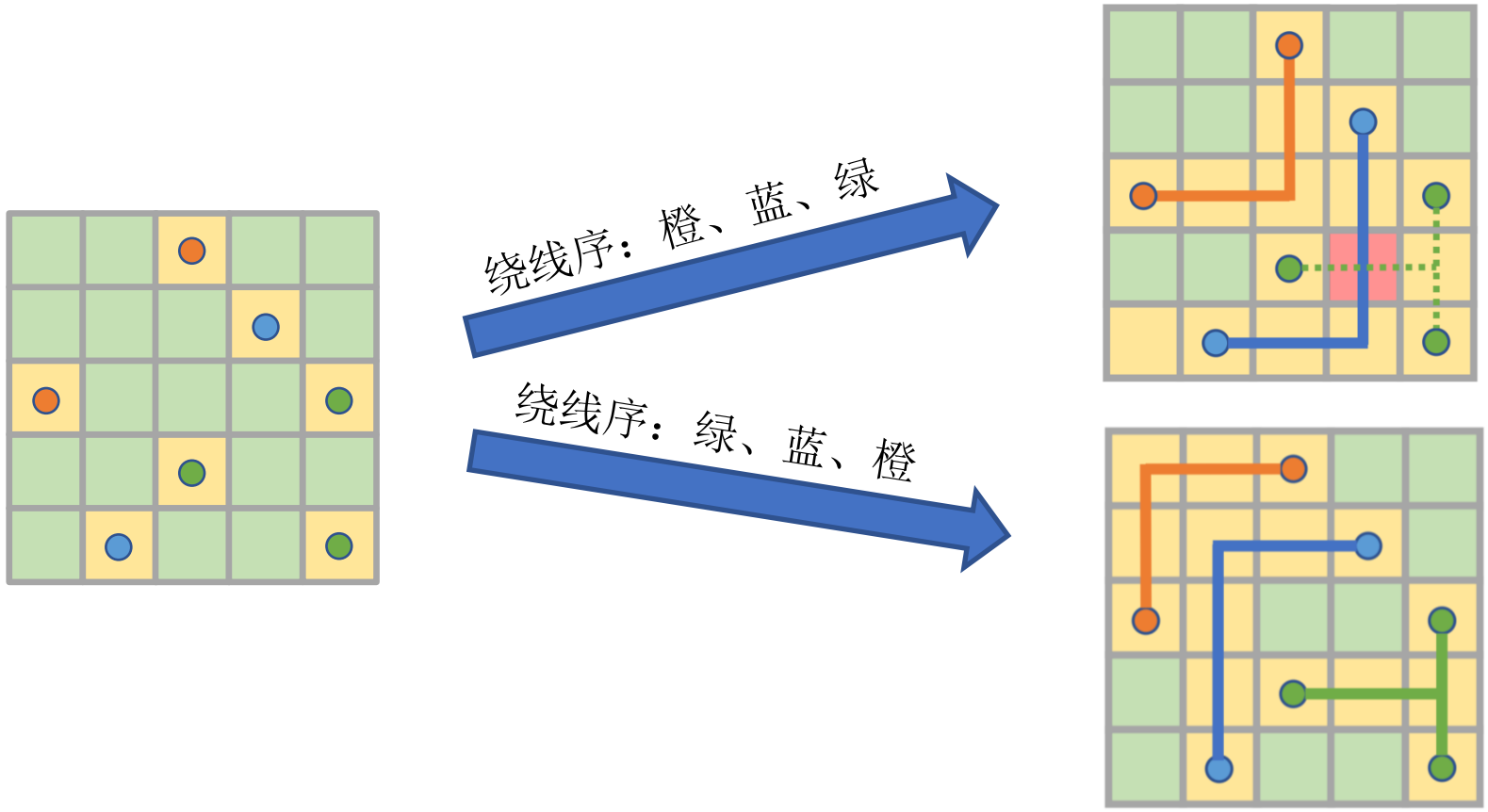


现有解决方法

- 通过**线网间的拥塞驱动布线**来解决大线网布线问题。
  [*Qien et al.,High performance global routing with fast overflow reduction.,ASP-DAC'2009*]
- 运用**三维迷宫路径搜索算法**进行避障并解决拥塞问题。
  [*Liu et al.,CUGR: Detailed-routability-driven 3D global routing with probabilistic resource model.,DAC'2020*]
- 在最拥塞的区域通过**整数线性规划**来求解绕线解。
  [*Cho et al.,BoxRouter2.0.,DAES'2009*]
- 对版图识别并分析拥塞区域进行**局部拆线重布**。
  [*Chang et al.,NTHU-route 2.0:A fast and stable global router.,ICCAD'2008*]
- 对线网**生成拥塞驱动的直角斯坦纳树拓扑**结构，再进行两点布线解决拥塞。
  [*Xu et al.,FastRoute 4.0:global router with efficient via minimization.,ASP-DAC'2009*]

全局布线的问题

- 现有处理方法的不足
  - 只是对已经产生的拥塞做**后处理优化**。
  - 没有解决**线网序**带来的拥塞影响，对布线顺序敏感



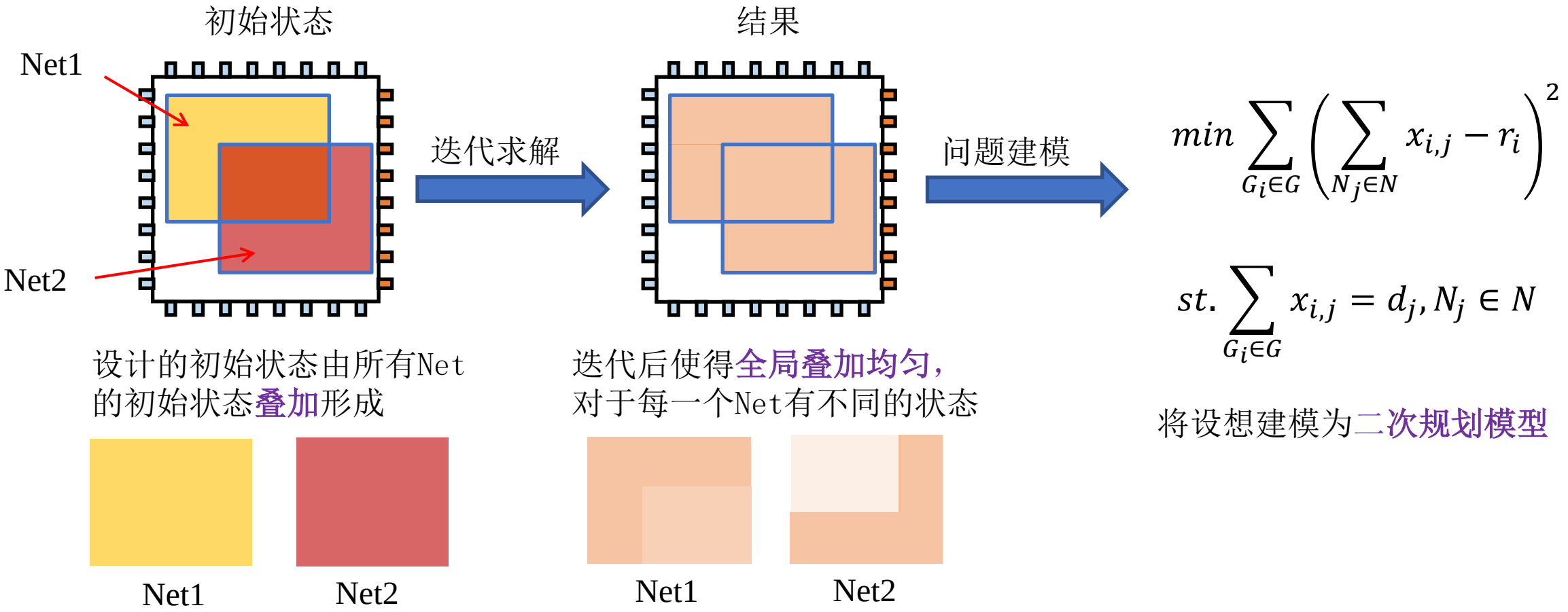
全局布线的问题

- 引入资源预分配

为解决以上问题，在传统框架上加入资源预分配。

资源预分配-建模

- 资源预分配原则：好的绕线结果是利用率高且接近均匀分布的



$$min \sum_{G_i \in G} \left(\sum_{N_j \in N} x_{i,j} - r_i \right)^2$$

$$st. \sum_{G_i \in G} x_{i,j} = d_j, N_j \in N$$

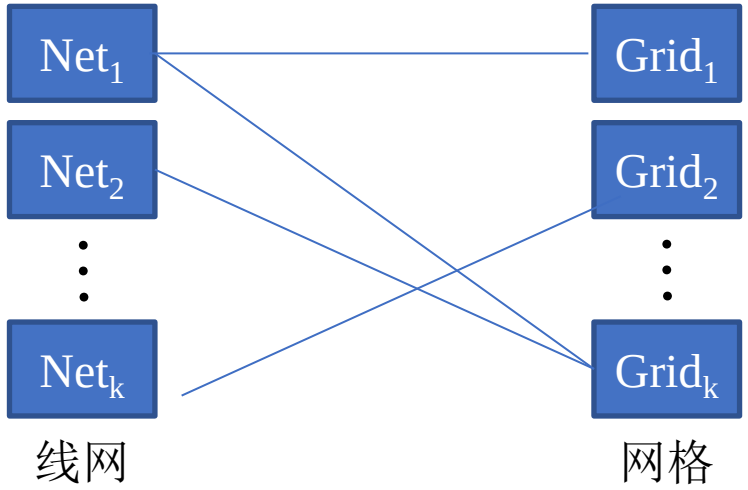
将设想建模为二次规划模型

资源预分配-迭代

资源预分配迭代模型

1 迭代模型

为了加速迭代过程中的数据存取过程，我们将线网与网格之间的多对多关系转换为**稀疏矩阵并压缩存储**。



2 目标函数

有约束求解方法难以求解，将原问题通过**罚方法**转换为无约束问题，再通过梯度下降方法，实现**可控迭代次数**。

有约束：$$min \sum_{G_i \in G} \left(\sum_{N_j \in N} x_{i,j} - r_i \right)^2 \quad st. \sum_{G_i \in G} x_{i,j} = d_j, N_j \in N$$

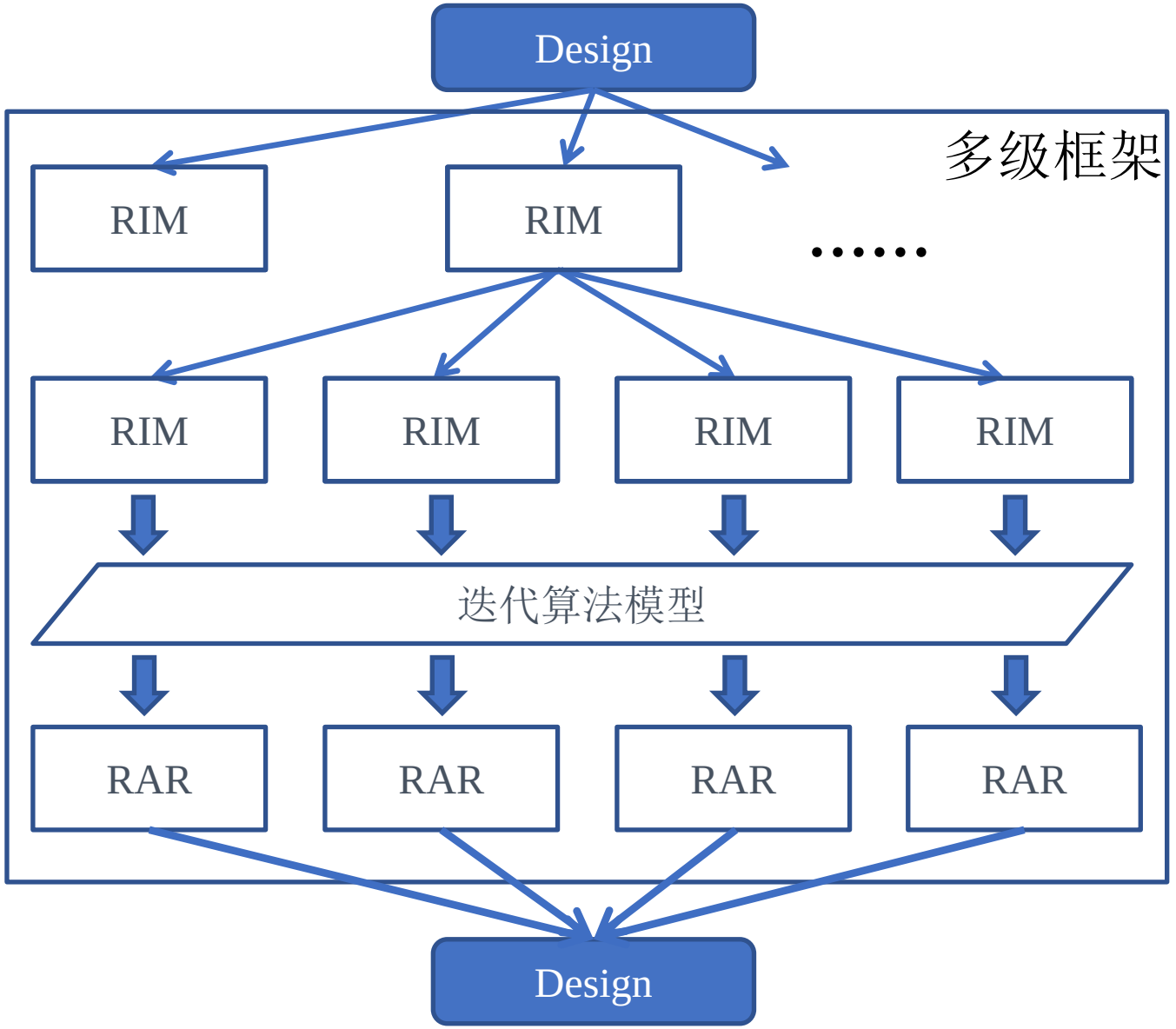
无约束：$$min \sum_{G_i \in G} \left(\sum_{N_j \in N} x_{i,j} - r_i \right)^2 + \frac{1}{2u} \cdot \sum_{N_j \in N} \left(\sum_{G_i \in G} x_{i,j} - d_j \right)^2$$

其中，N和G是在设计中的线网与方格。$x_{i,j}$是第i个方格在第j个线网上分配的资源数。r_i是第i个方格的总资源数(Resource)，d_j是第j个线网所需的资源数(Demand)。

资源预分配-加速

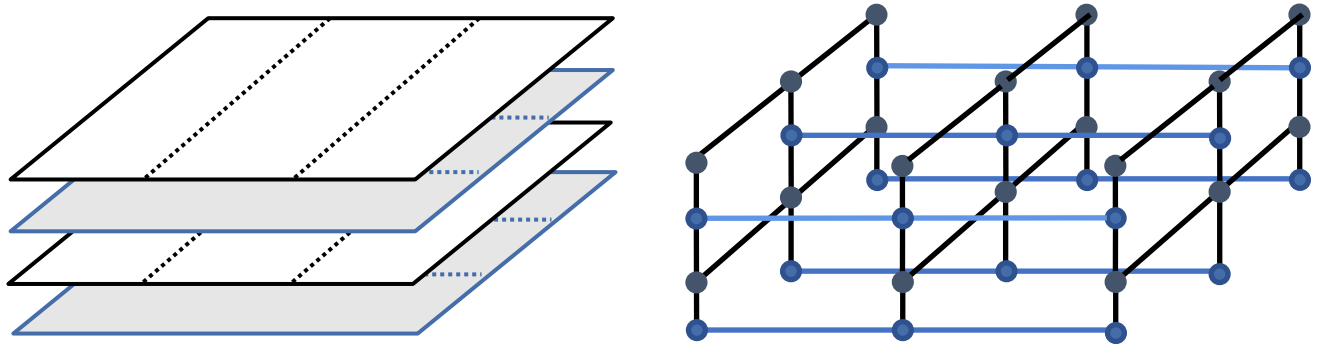
- 资源预分配多级框架

由于数据量庞大，直接对整个设计运算时间长。为了减少运算时间，设计了**资源预分配的多级框架**，将版图拆分为若干个资源迭代模型(RIM)，再对每个资源迭代模型求解得到资源分配结果(RAR)。

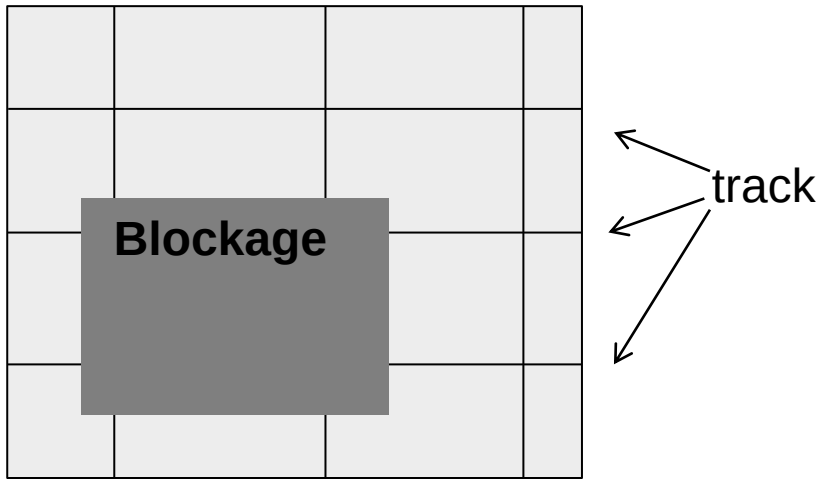


全局布线-建模

1 将设计建模为三维网格，对网格内的资源进行衡量



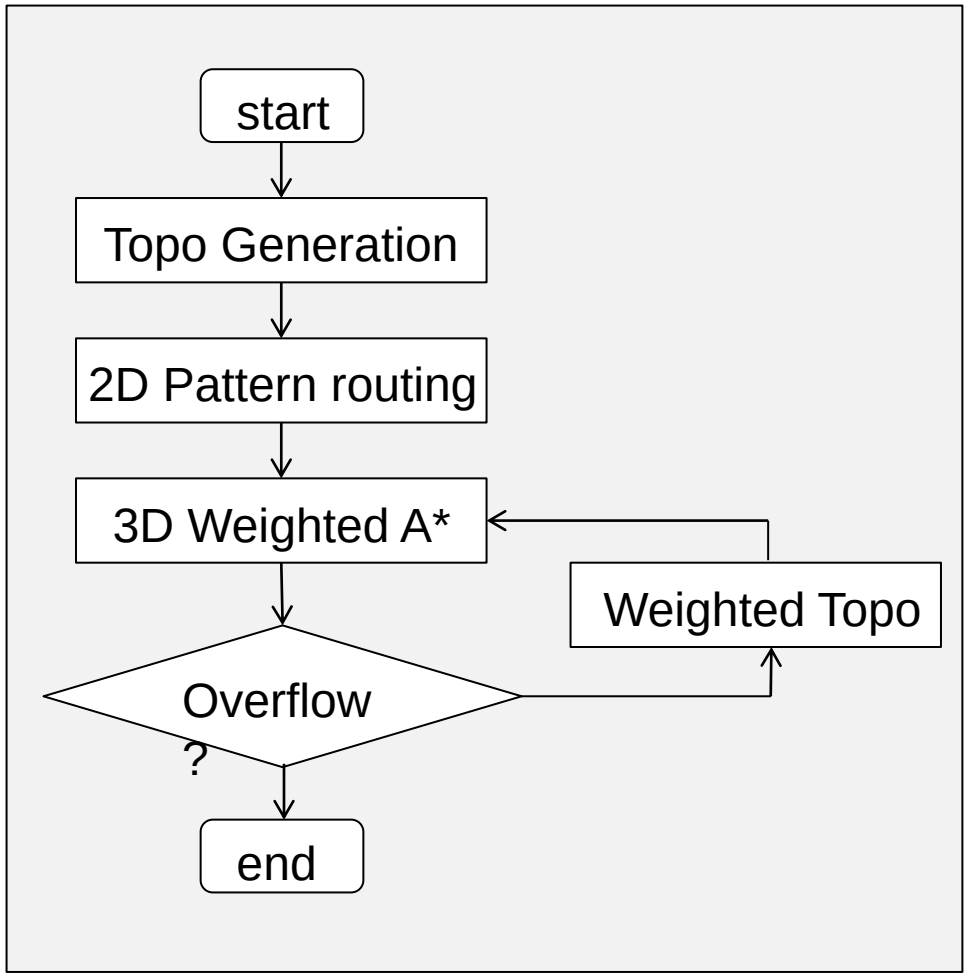
2 GCell资源将由track条数来确定



resource：能够贯穿GCell的wire作为资源

access：能够进入GCell边界的资源

全局布线-flow



1 使用查找表对每个线网生成topo

2 将topo拆解为多个两点线网，进行模式布线，结果作为下一步的输入

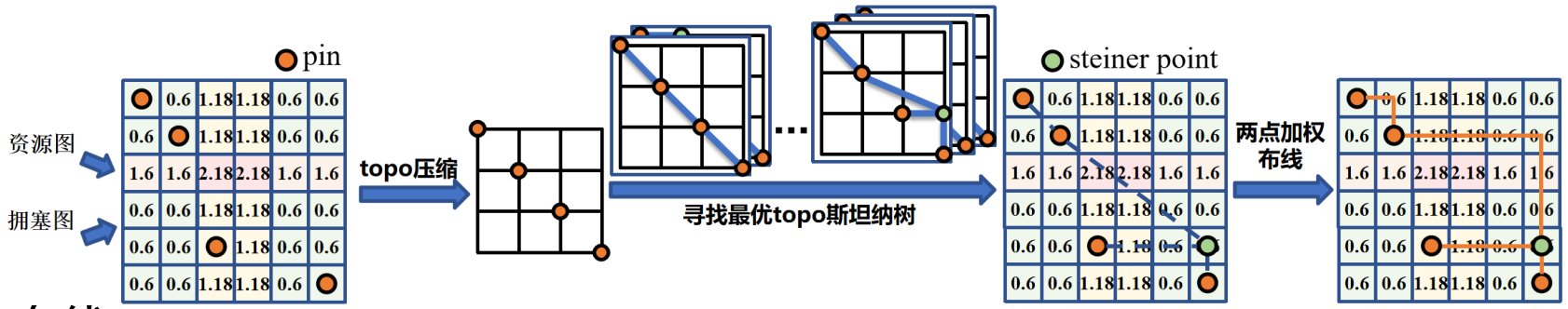
3 通过三维权值A*算法，生成两点的三维布线结果

4 若有overflow，将拆线重布，并用三维A*多源多汇算法计算每个线网的布线结果

全局布线-单线网

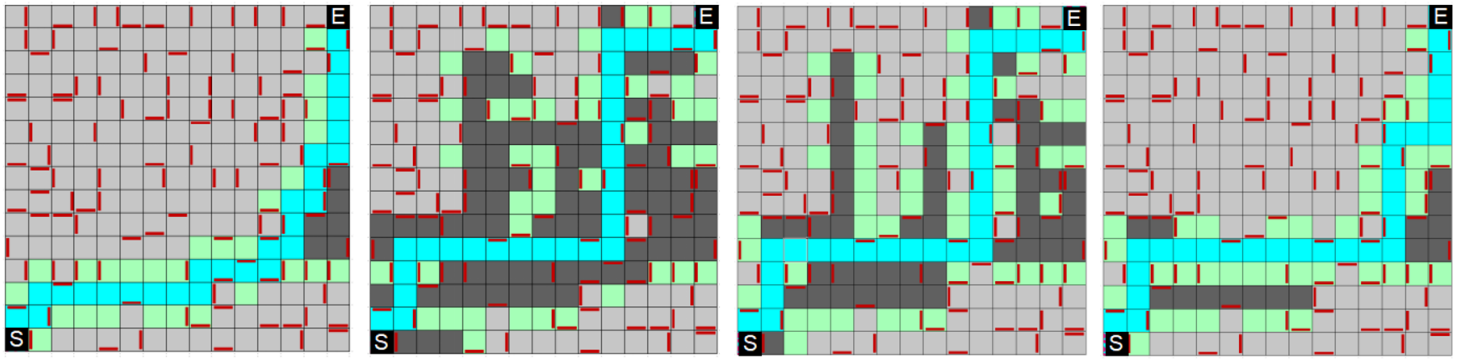
单线网布线

通过**最优直角斯坦纳树查找表**的方式，快速构建线网拓扑连接关系，再将拓扑解构为两点线网，由于资源预分配的影响，进行权值下两点布线。



两点布线

通过**L型拐点预测**，降低三维A*布线时的平面拐点数量。



无拐点优化
(拐点12，范围6)

简单拐点优化
(拐点6，范围78)

直线预测拐点优化
(拐点6，范围48)

L型预测拐点优化
(拐点6，范围15)

01 研究问题

02 研究内容

03 未来展望

未来展望

1 GR-TA-DR 全局详细多模块联合优化

2 GR需要更强大的解拥塞能力

3 加速GR过程，提出可延展性强的并行算法

4 考虑时序的GR

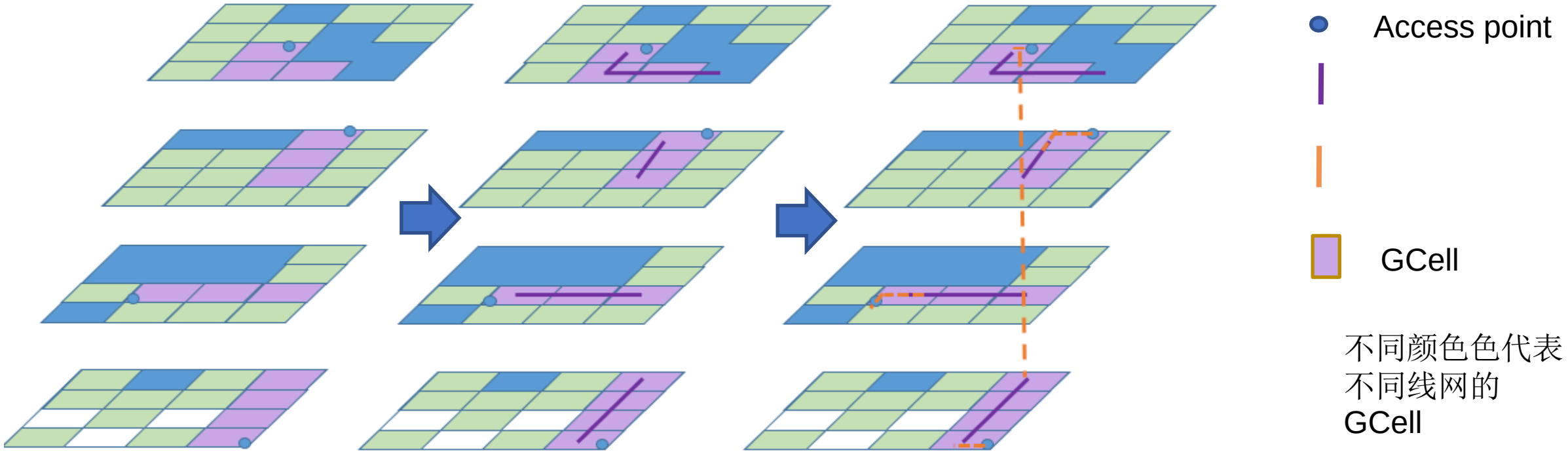
iEDA Tutorial 第五期议程

- Part 1 iEDA-iCTS 问题、研究内容与计划（李伟国）
- Part 2 iEDA-iRT 全局布线问题、研究内容与计划（曾智圣）
- **Part 3 iEDA-iRT 详细布线问题、研究内容与计划 （刘继康）**
- Part 4 iEDA-iDRC 问题、研究内容与计划 （郭帆）

详细布线问题

- **详细布线问题介绍**

对于给定布线任务（线网），要求根据全局布线（global routing）的结果，确定详细布线的布线区域，进而在满足设计规则的前提下，利用布线算法，得到设计指标最优（如线长，通孔，时序）的布线方案。

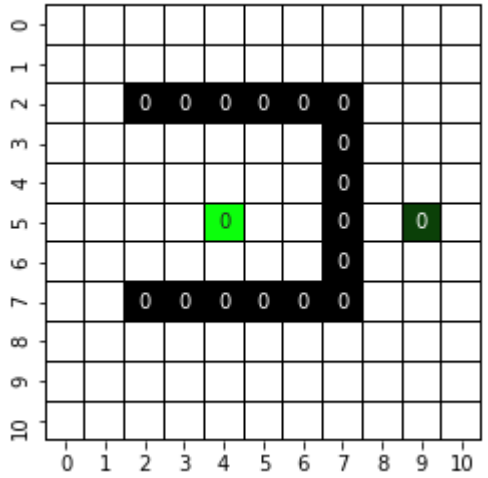


详细布线问题

目前详细布线的挑战

①布线单元可连接性问题：布线过程中，布线环境对器件单元可连接性的造成影响，难以确定合法的单元连接点。

②布线时间过长且存在大量冗余绕线。随着集成电路制造工艺进入7nm以下，数字芯片中标准单元数量已经达到亿数量级，EDA布线已经成为典型的数据密集型计算的典型代表，其详细绕线的计算时间非常冗长，以小时计。



现有解决方法

- 考虑了基本设计规则，加入了层内并行绕线，减少了Via与Wirelength。
[*Andrew B. Kahng, TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies., ICCAD'2018*]

- 提供了一个可能DRC clear的解决方案的开源详细绕线工具。
[*Andrew B. Kahng, TritonRoute: The Open-Source Detailed Router,. TCAD'2021*]

- 融合了全局和详细的开源绕线工具，提供了比较好绕线解决方案。
[*Andrew B. Kahng, TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine,. TCAD'2022*]

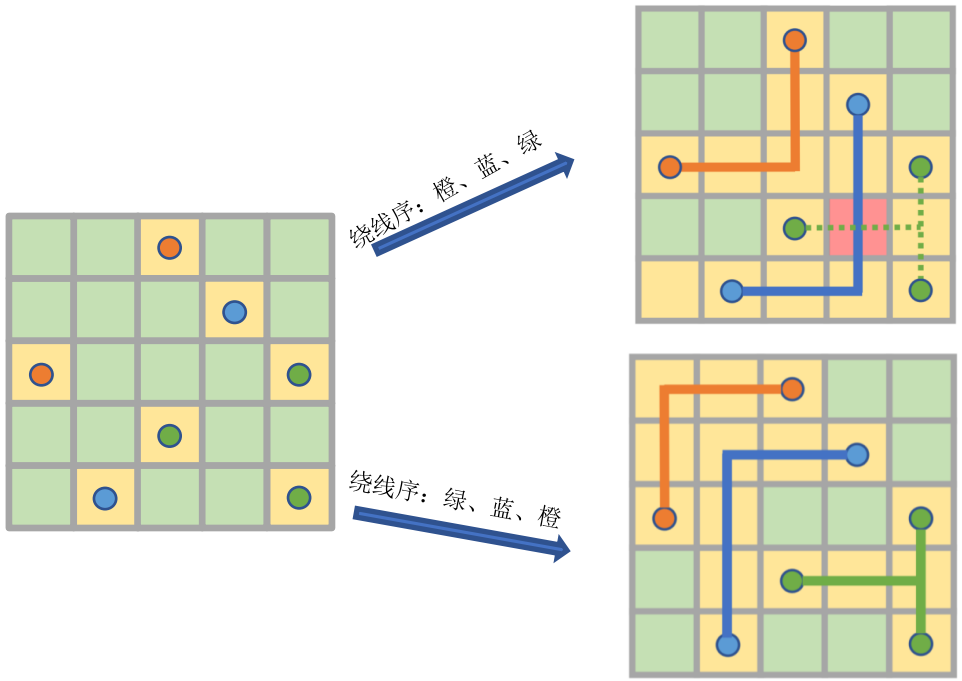
- 在绕线上使用了多源单目标的并行搜索方案，初始解考GlobalRouting
[*Fan-Keng Sun, A Multithreaded Initial Detailed Routing Algorithm Considering Global Routing Guides, ICCAD'2018*]

- 在3D绕线上使用了稀疏结构，缩短了搜索空间，并考虑了最小面积约束
[*Gengjie Chen, Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search, TCAD'2020*]

详细布线问题

- 目前详细布线的挑战

③布线资源竞争问题：布线任务规模庞大，但布线资源有限，并存在大量的障碍。前期的布线任务大量占用后期布线任务的资源，最终，造成布线拥塞与线网间违例。



- 考虑了基本设计规则，加入了层内并行绕线，减少了Via与Wirelength。
[*Andrew B. Kahng, TritonRoute: An Initial Detailed Router for Advanced VLSI Technologies., ICCAD'2018*]

- 提供了一个可能DRC clear的解决方案的开源详细绕线工具。
[*Andrew B. Kahng, TritonRoute: The Open-Source Detailed Router,. TCAD'2021*]

- 融合了全局和详细的开源绕线工具，提供了比较好绕线解决方案。
[*Andrew B. Kahng, TritonRoute-WXL: The Open-Source Router With Integrated DRC Engine,. TCAD'2022*]

- 在绕线上使用了多源单目标的并行搜索方案，初始解考GlobalRouting
[*Fan-Keng Sun, A Multithreaded Initial Detailed Routing Algorithm Considering Global Routing Guides, ICCAD'2018*]

- 在3D绕线上使用了稀疏结构，缩短了搜索空间，并考虑了最小面积约束
[*Gengjie Chen, Dr. CU: Detailed Routing by Sparse Grid Graph and Minimum-Area-Captured Path Search, TCAD'2020*]

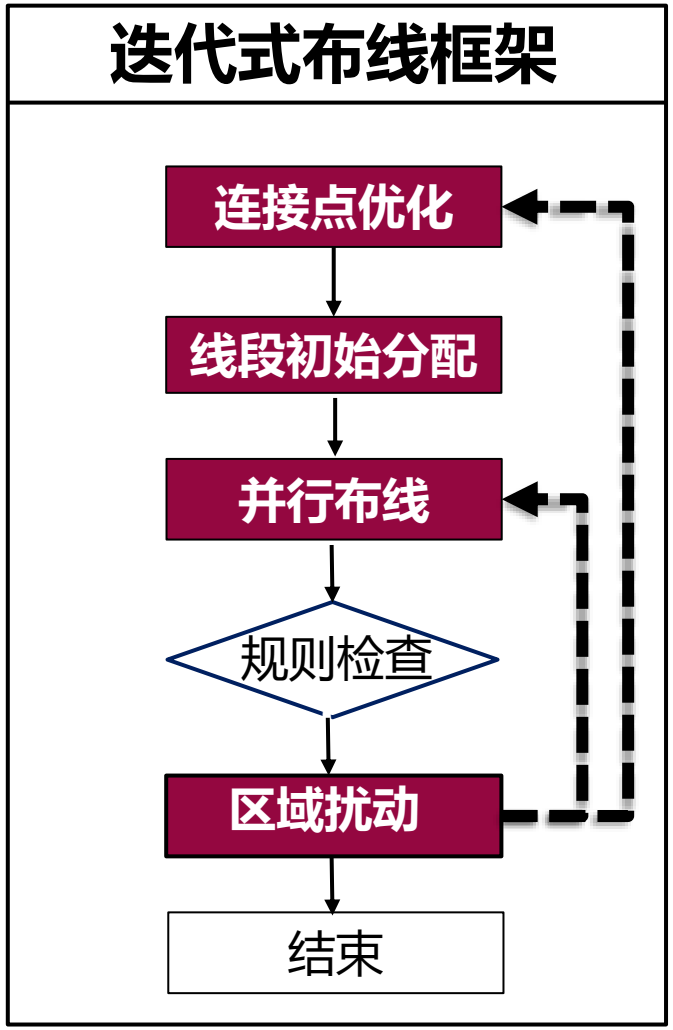
01 研究问题

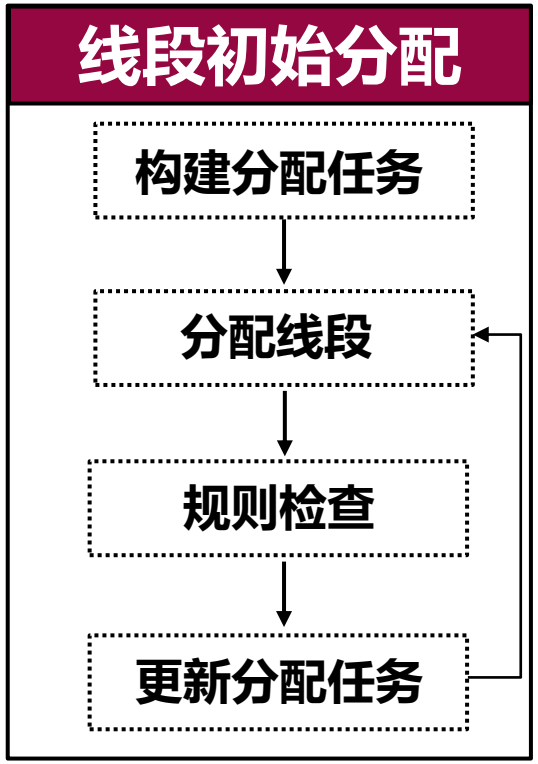
02 研究内容

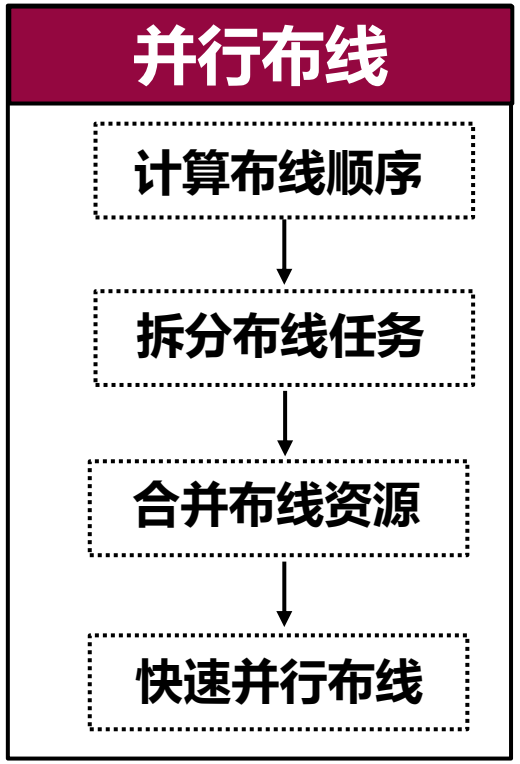
03 未来展望

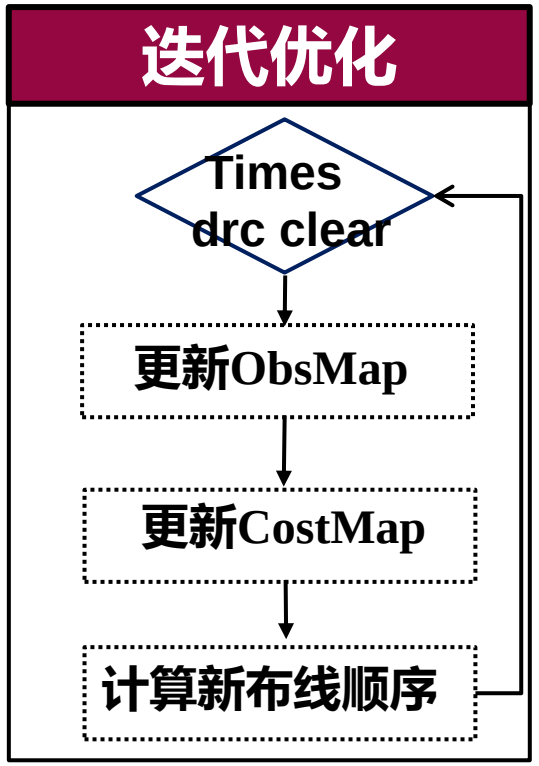
详细布线

- 迭代式布线框架：





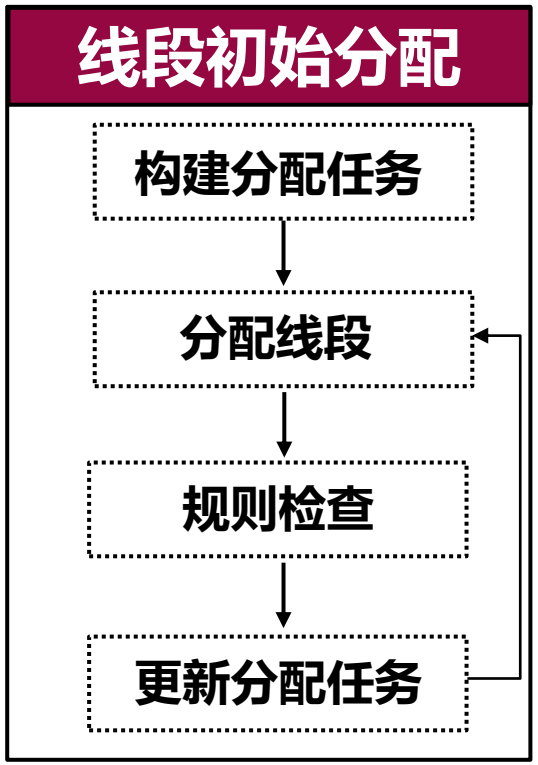


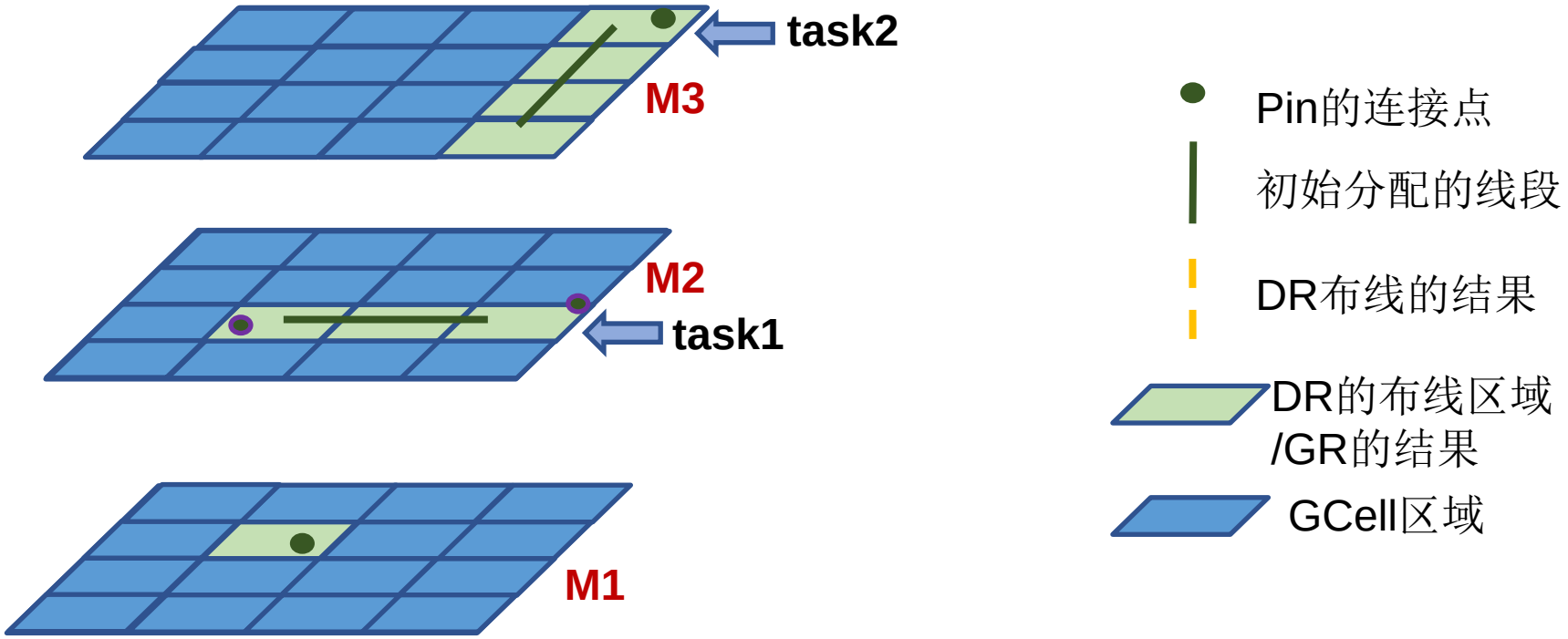


详细布线-线段初始分配(Track Assignment)

- 线段初始分配：构建分配任务

在Global Routing给出的布线区域，分配一个初始分配的线段，连接两端的区域，视作一个分配任务。

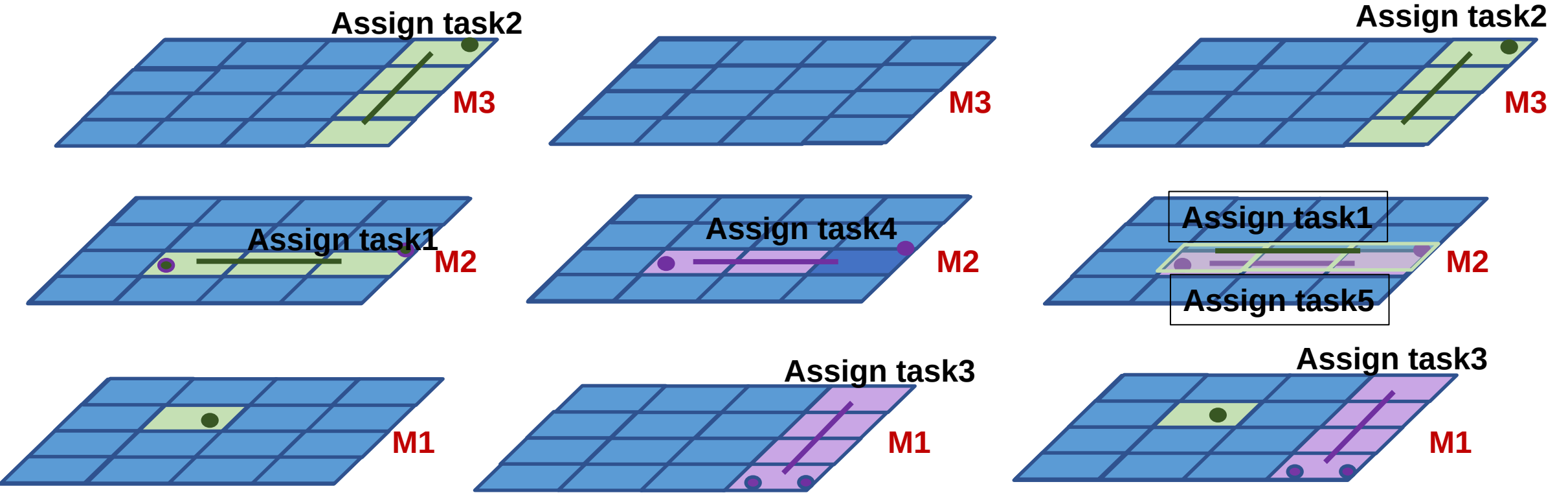




详细布线-线段初始分配(Track Assignment)

■线段初始分配：构建分配任务

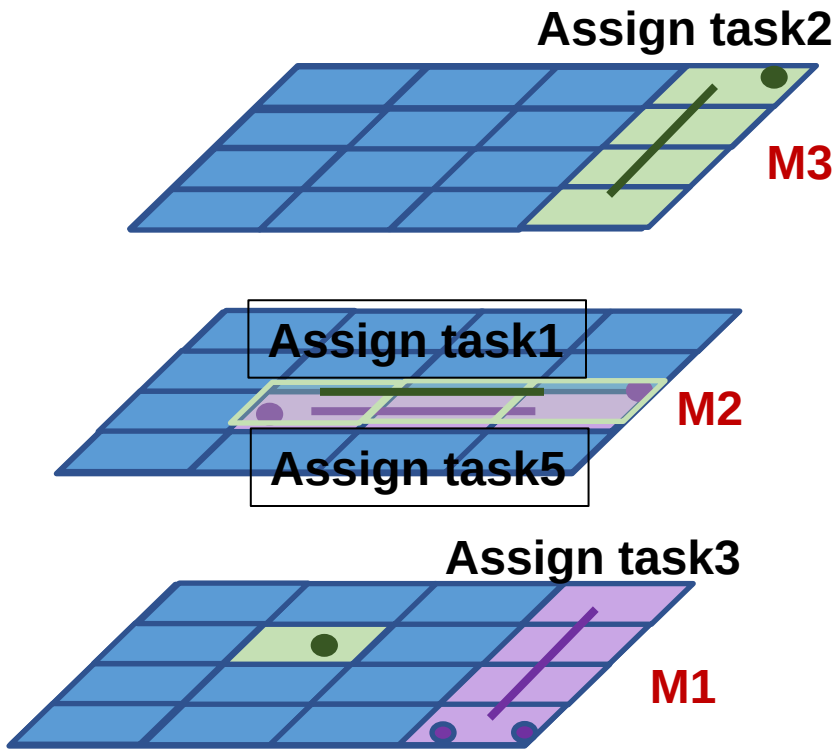
若布线任务所在区域在平面上的投影相同，则合并这些布线任务所拥有的资源，在更大空间范围内布线，在遵循Global Routing结果的同时增加了解空间。

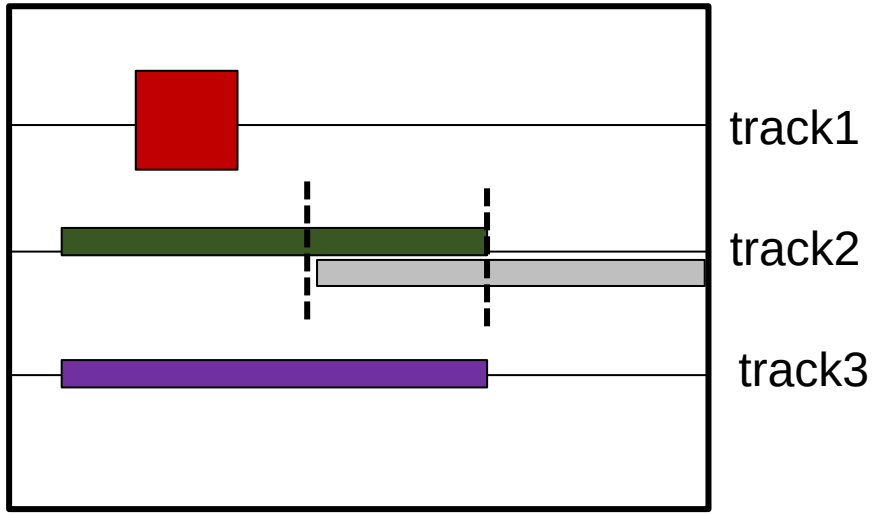


Net1的布线任务　　Net2的布线任务　…　NetN的布线任务　　合并Net1和NetN的布线任务

详细布线-线段初始分配(Track Assignment)

- 线段初始分配：分配线段

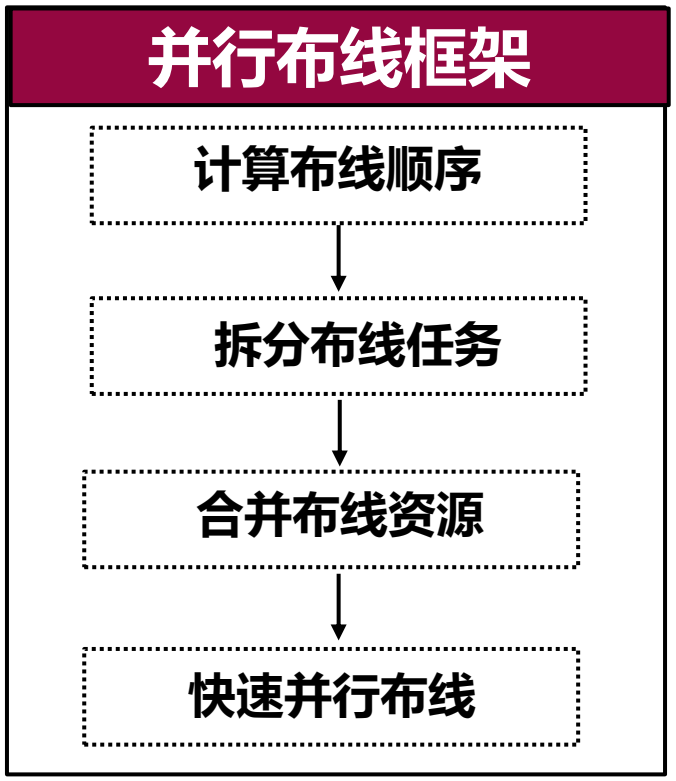


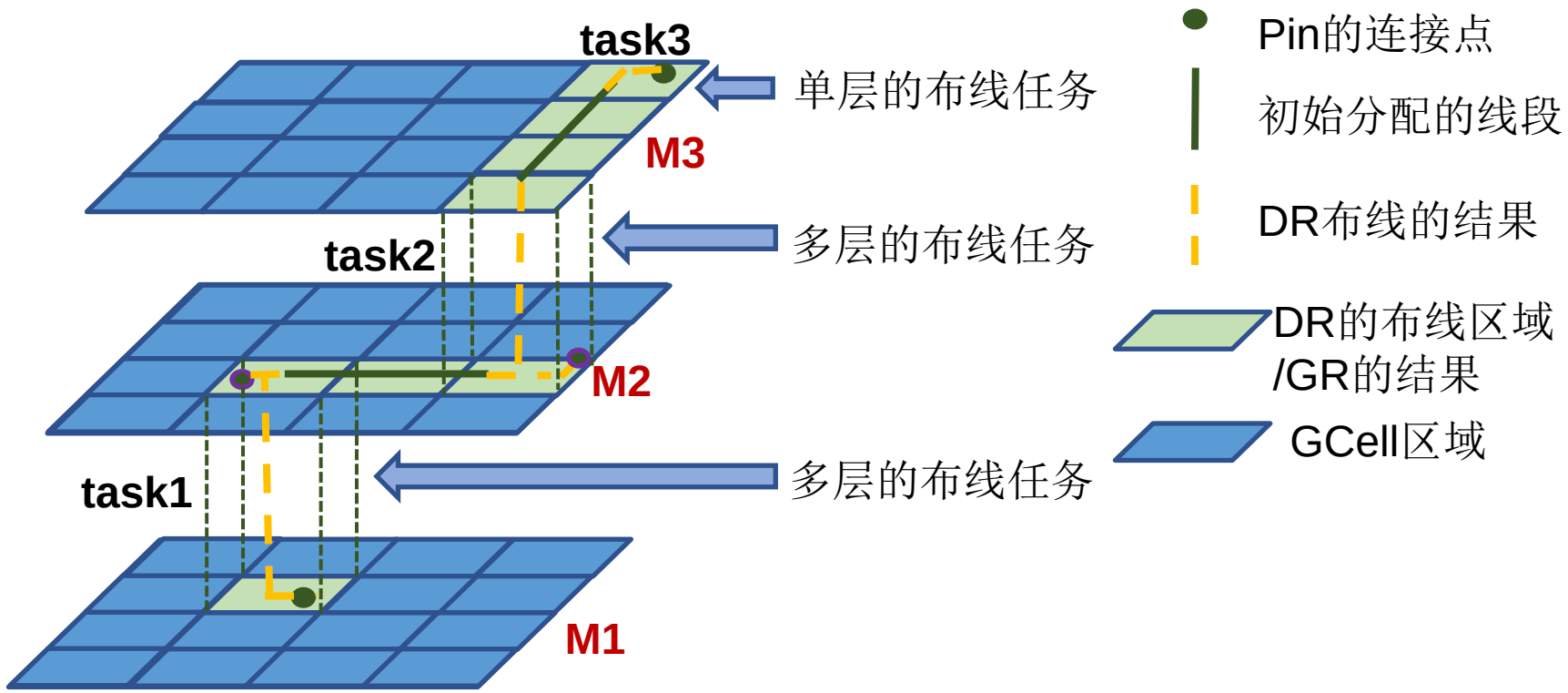


详细布线-并行框架

- 并行布线：构建布线任务

在Global Routing结果给出的布线区域内，将初始分配的线段、pin联通，视作一个布线任务。

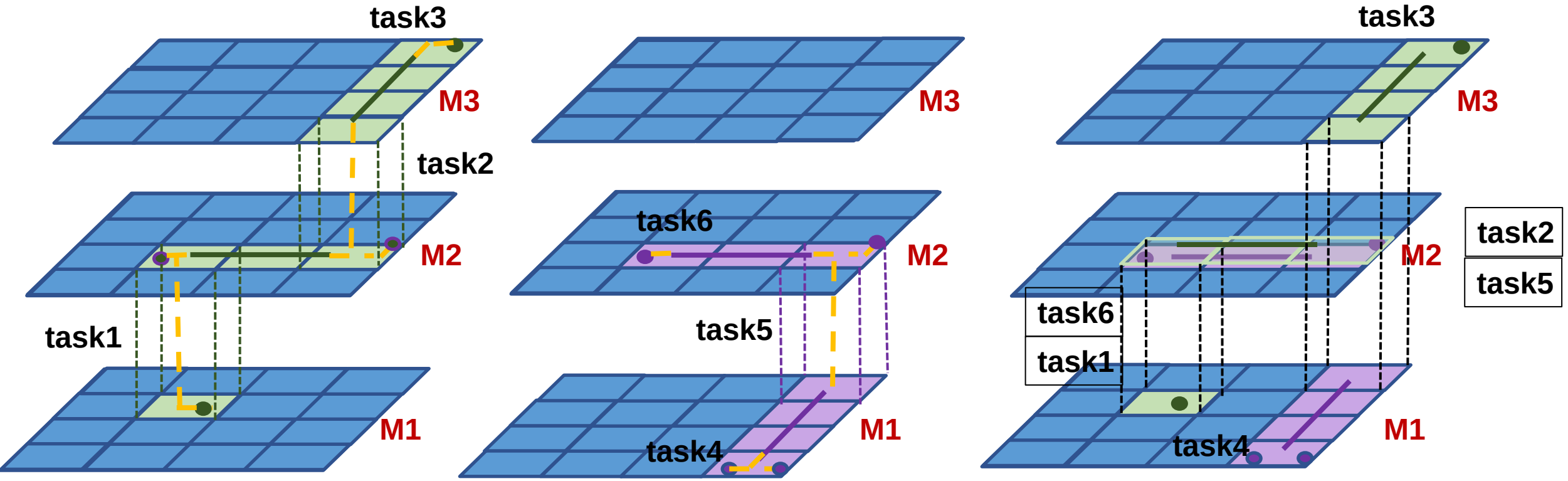




详细布线-并行框架

■并行布线：合并布线资源

若布线任务所在区域在平面上的投影相同，则合并这些布线任务所拥有的资源，在更大空间范围内布线，在遵循Global Routing结果的同时增加了解空间。

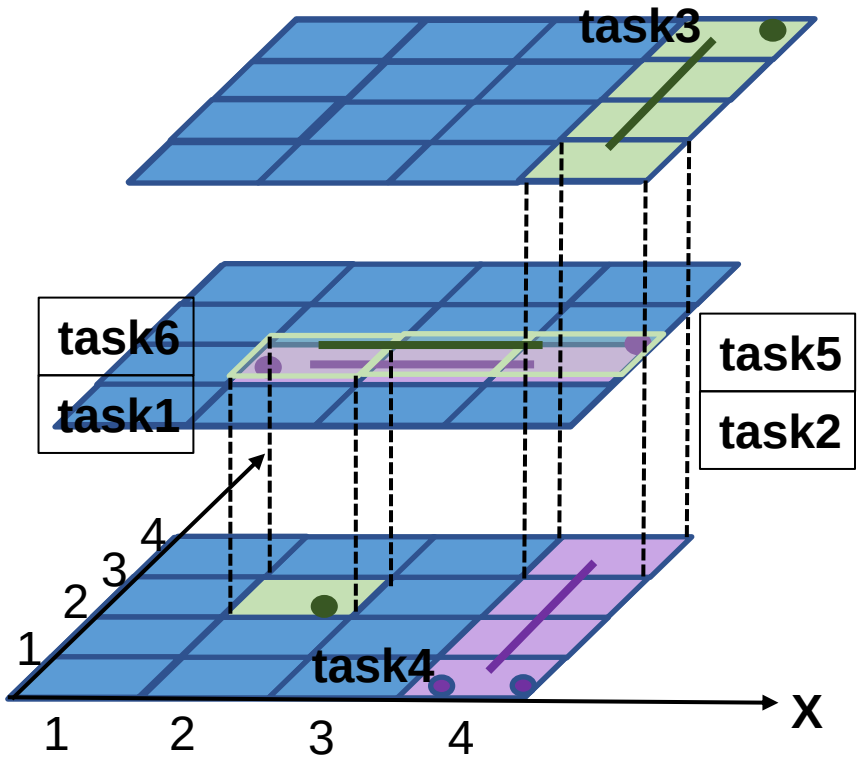


Net1的布线任务　　Net2的布线任务　…　NetN的布线任务　　合并Net1和NetN的布线任务

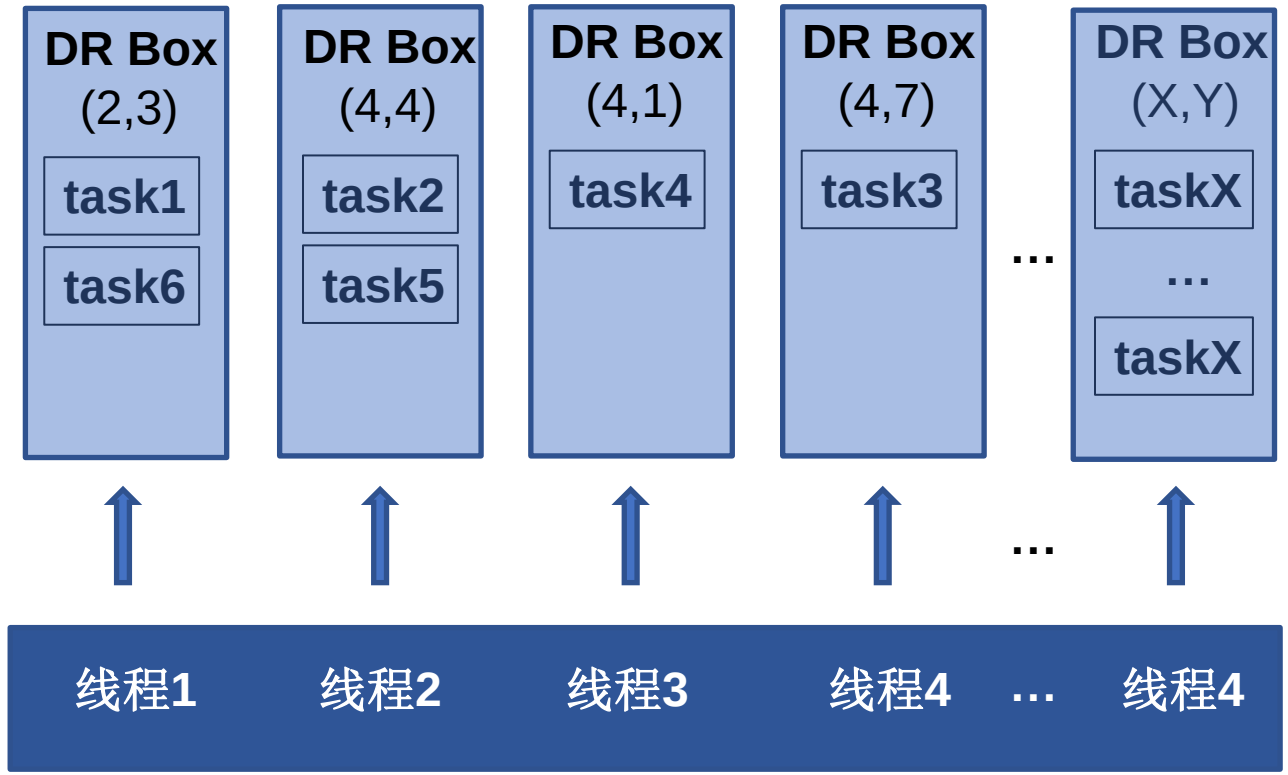
详细布线-并行框架

■并行布线：确定布线任务的执行顺序，并行多线网布线

对共享资源的布线任务(task)之间，按照连接点外接矩形大小关系，确定初始的布线顺序，实现Gcell粒度的并行。



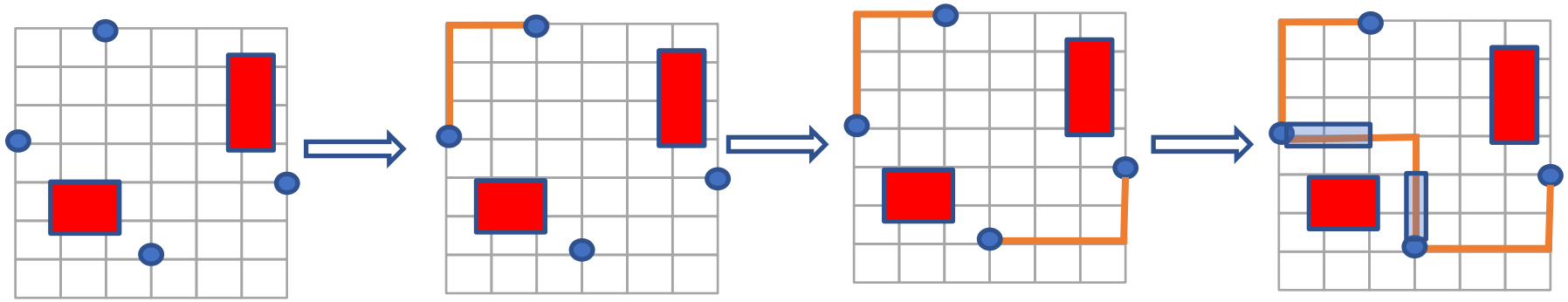
合并Net1和Net2的布线资源



详细布线-布线算法

- 布线算法： 传统A*路径搜索算法

现有工作主要使用**迷宫算法**和**A*算法**进行布线，缺点是没有复用前面的布线结果，导致冗余绕线，导致线长增加，消耗额外布线资源，降低了布线质量。



Access point

A* 搜索的起点

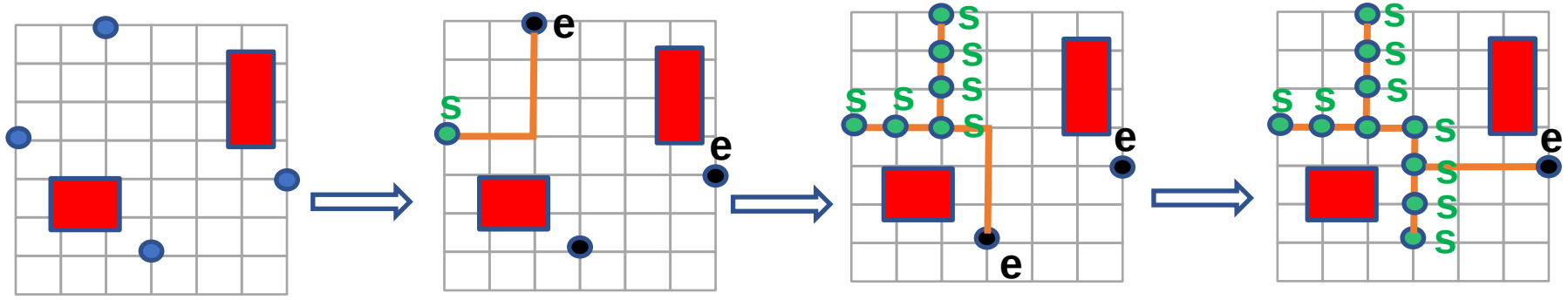
A* 搜索的终点

Blockage

详细布线-布线算法

- 多源多汇的A*路径搜索算法

针对**迷宫算法**和**A*算法**进行布线，没有复用前面的布线结果，导致冗余绕线，导致线长增加，我们改进了A*布线算法为多源多汇，避免冗余绕线。



e

Access point

A* 搜索的起点

A* 搜索的终点

Blockage

A* 路径搜索cost map

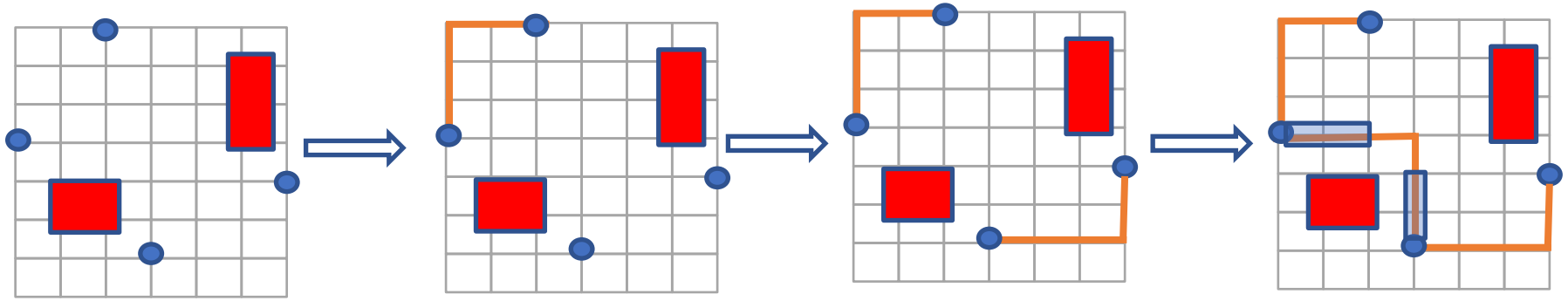
0.2	0.1	0.3	0.5
0.1	0.2	0.8	0.5
0.5	0.4	0.8	0.7
0.1	0.1	0.2	0.1

整数化

1	1	1	2
1	1	3	2
2	2	3	3
1	1	1	1

详细布线-布线算法

- 布线算法： 传统A*路径搜索算法



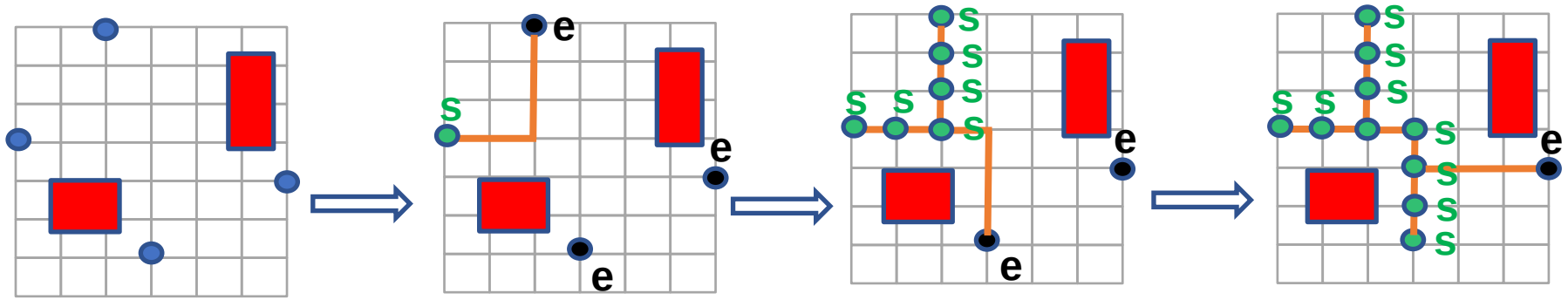
Access point

A* 搜索的起点

A* 搜索的终点

Blockage

- 布线算法： 多源多汇的A*路径搜索算法



A* 路径搜索cost map

0.2	0.1	0.3	0.5
0.1	0.2	0.8	0.5
0.5	0.4	0.8	0.7
0.1	0.1	0.2	0.1

整数化

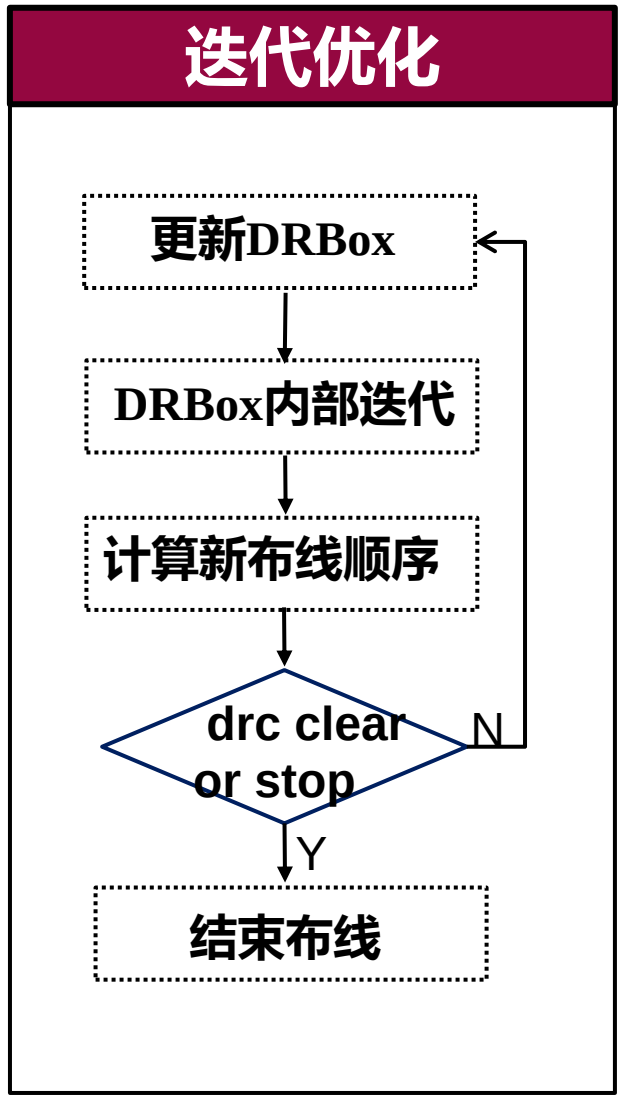
1	1	1	2
1	1	3	2
2	2	3	3
1	1	1	1

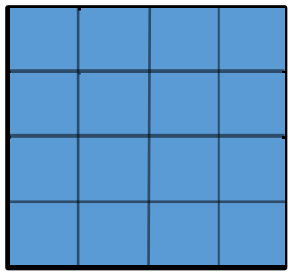
01 研究问题

02 研究内容

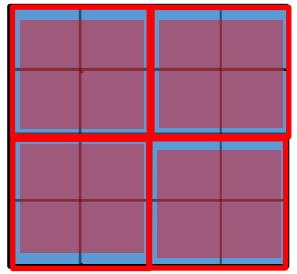
03 未来展望

详细布线-可变布线区域

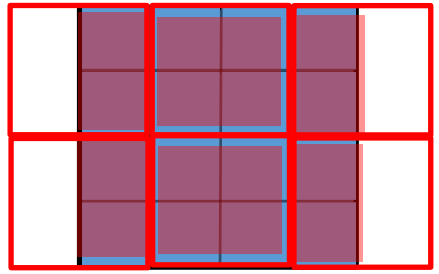




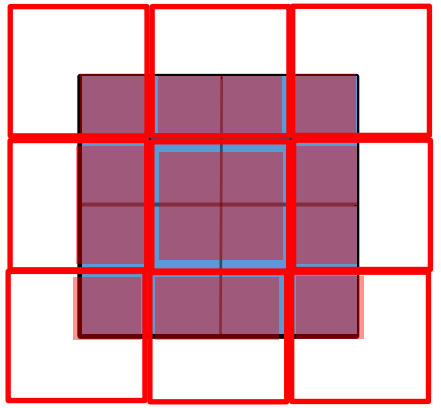
DRBox
size：2x2gcell
offset：x:0,y:0



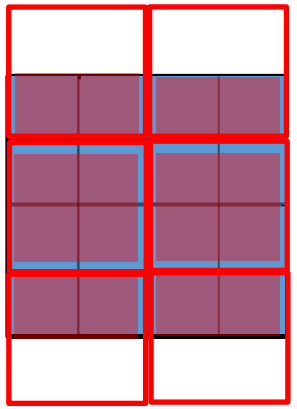
DRBox
size：2x2gcell
offset：x:0,y:0



DRBox
size：2x2gcell
offset：x:-1,y:0



DRBox
size：2x2gcell
offset：x:-1,y:-1



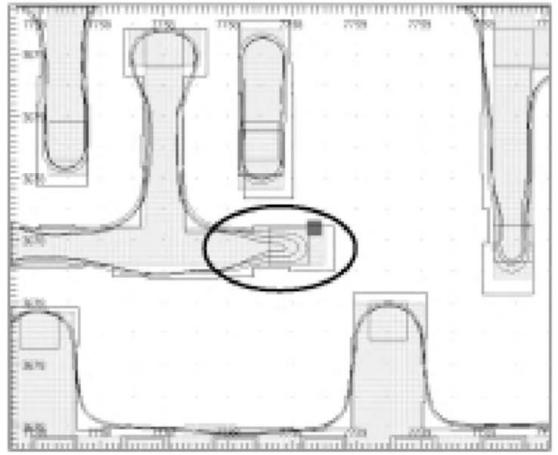
DRBox
size：2x2gcell
offset：x:0,y:-1

iEDA Tutorial 第五期议程

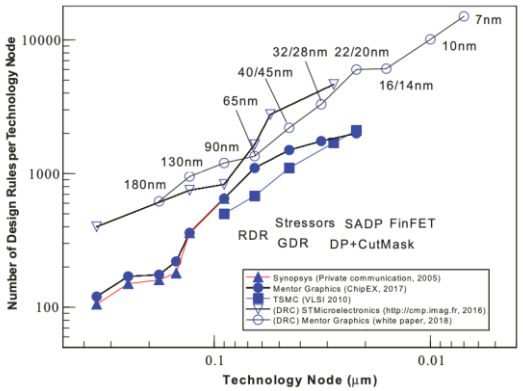
- Part 1 iEDA-iCTS 问题、研究内容与计划（李伟国）
- Part 2 iEDA-iRT 全局布线问题、研究内容与计划（曾智圣）
- Part 3 iEDA-iRT 详细布线问题、研究内容与计划 （刘继康）
- **Part 4 iEDA-iDRC 问题、研究内容与计划 （郭帆）**

研究问题一 设计规则

- **目的**
  - 为了向设计者传递工艺和制造的限制，确保芯片设计能够按预期稳定运行，并具有高良品率。
    - 实际光刻出来的图形会与设计的规则图形有偏差。
- **内容**
  - 大部分规则基于光刻要求设定。
  - 也约有 25% 的规则是为了支持平坦化工艺、湿法和干法刻蚀工艺、可靠性和其他工艺要求而定义的。
  - 20nm 工艺约 2000 条规则，数量还在增加。

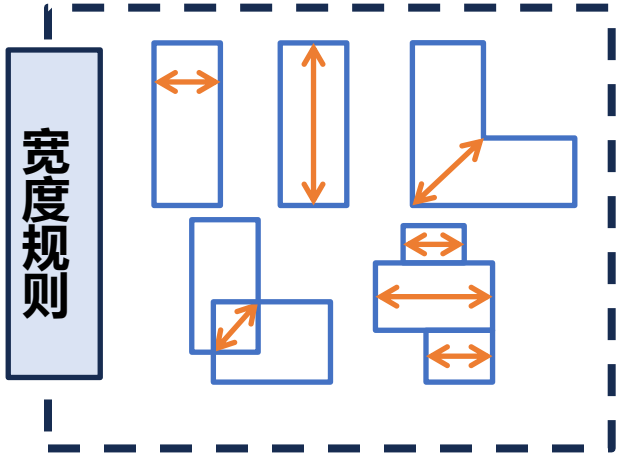


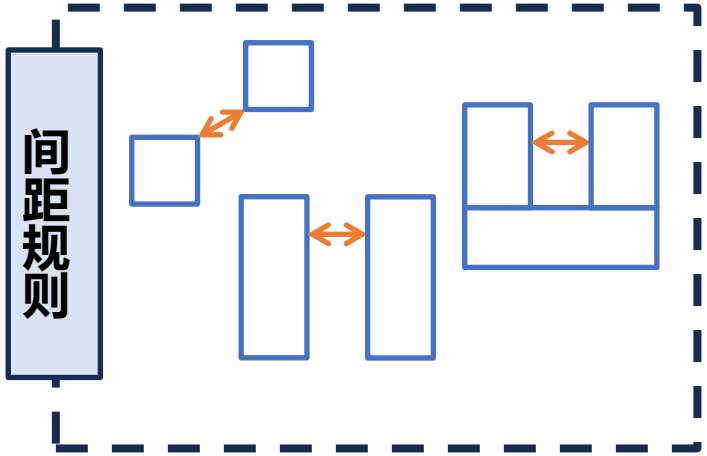
Printed shapes

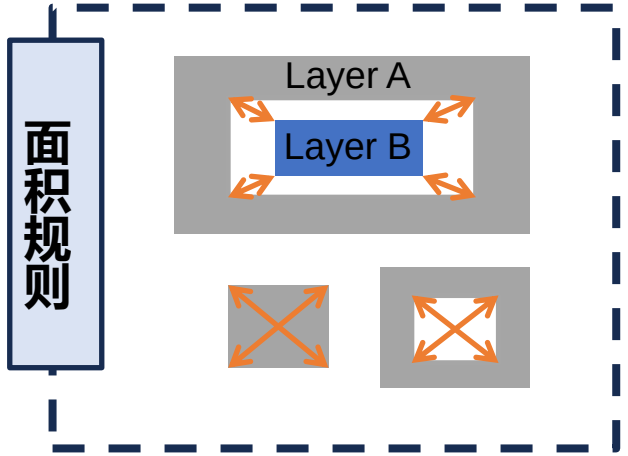


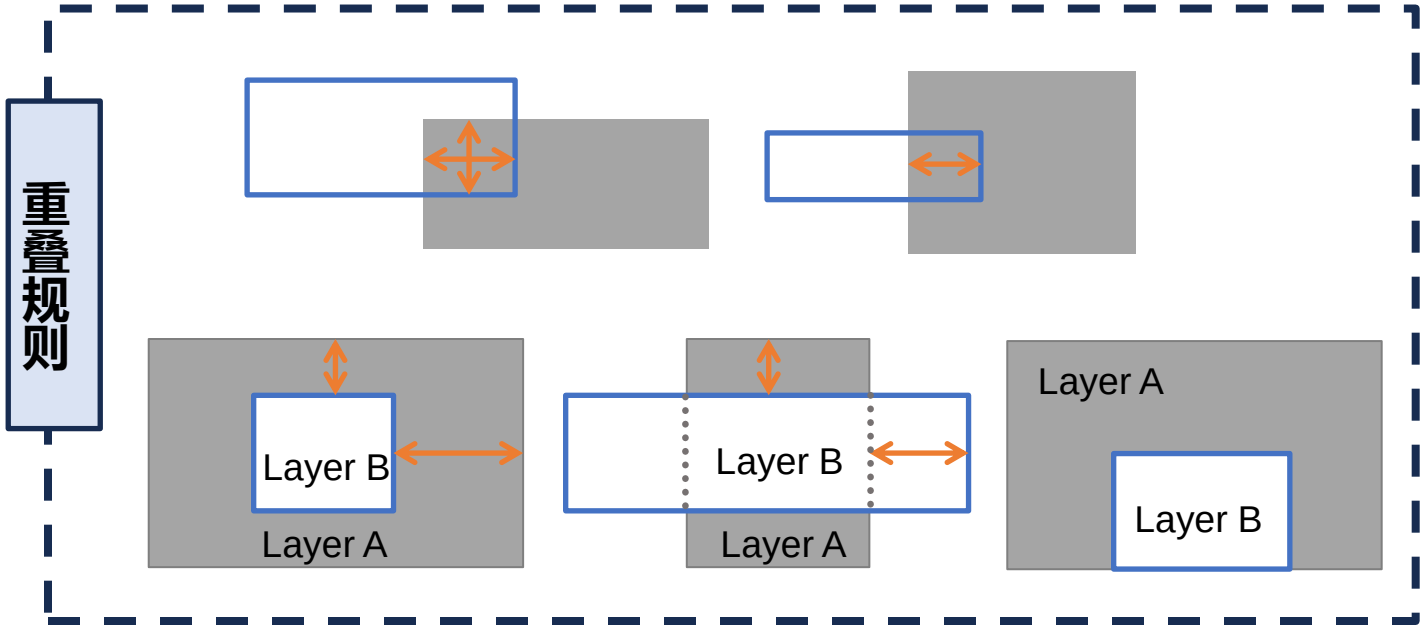
设计规则数量

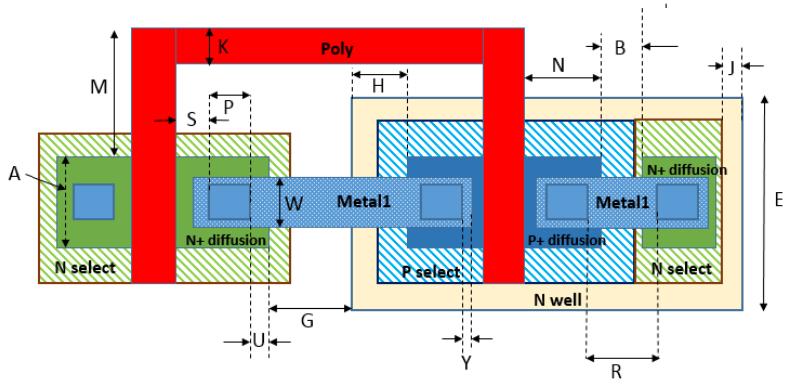
研究问题二 几何描述


宽度规则


间距规则


面积规则
Layer A
Layer B


重叠规则
Layer B
Layer A
Layer B
Layer A
Layer A
Layer B

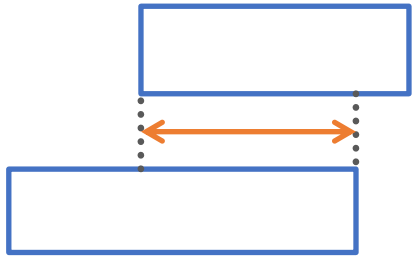

K
Poly
M
H
N
B
J
S
P
A
W
Metal1
N+ diffusion
P+ diffusion
N select
P select
N well
E
G
U
Y
R

研究问题三 形状定义

Relations

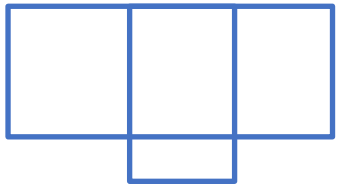
End of Line

Joint

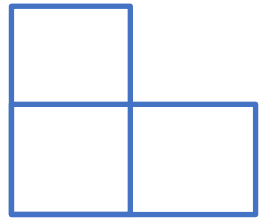


PRL

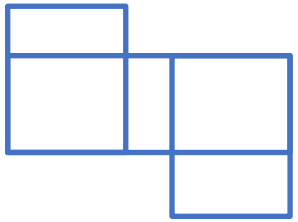
Jog



‘T’ Jog

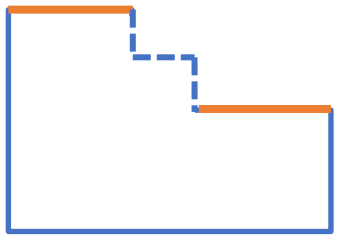


‘L’ Jog

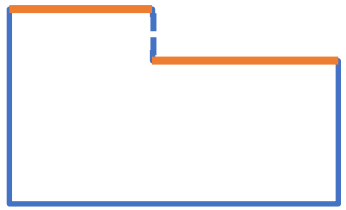


‘Z’ Jog

Step

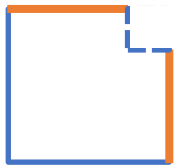


Step

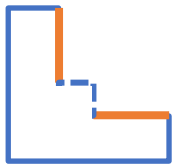


Step

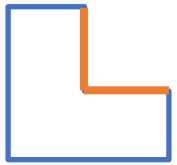
Corners



Outside Corner



Inside Corner

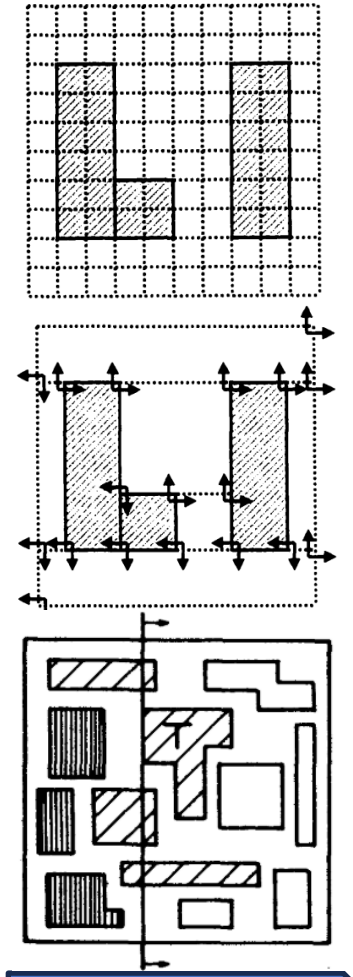


Concave Corner

Convex Corner

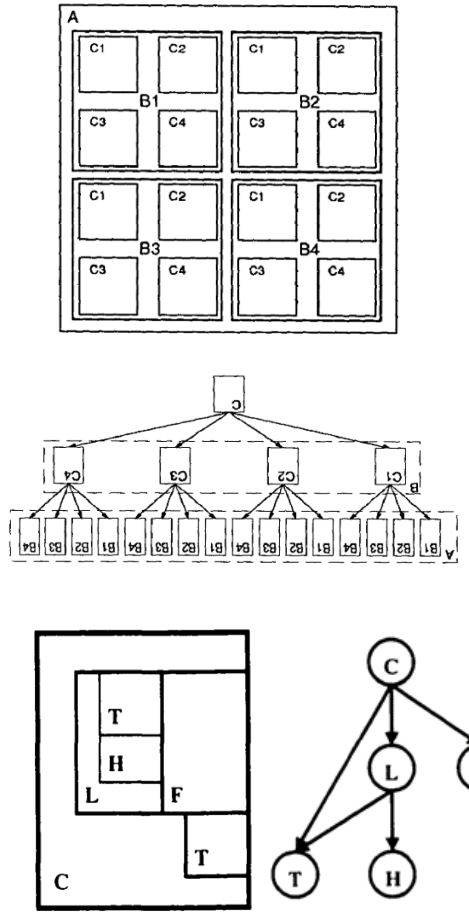
研究问题四 数据结构和算法

数据结构和算法



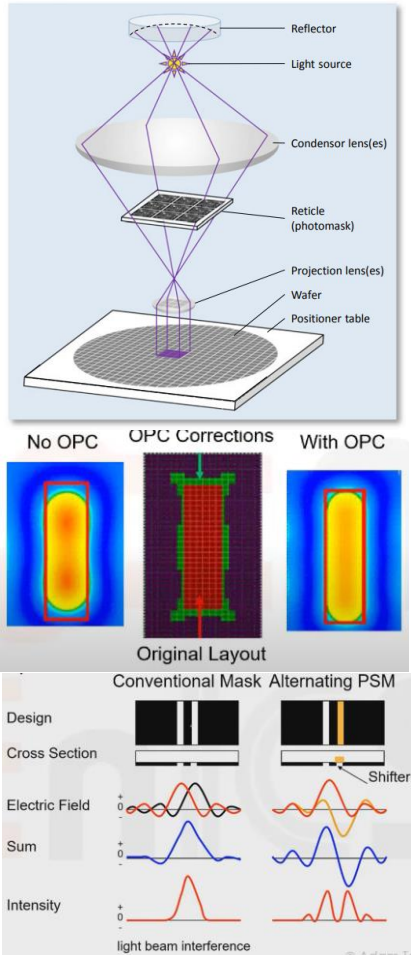
80年代

版图分区和分层



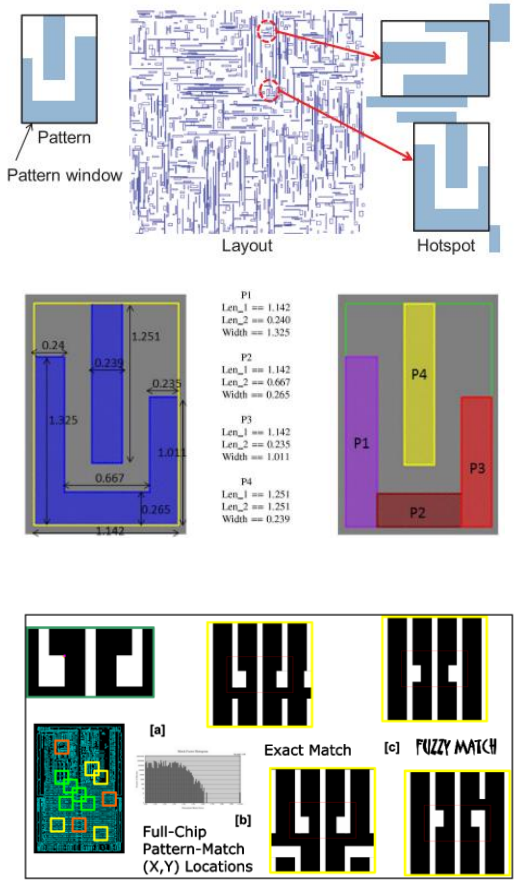
90年代

亚波长光刻



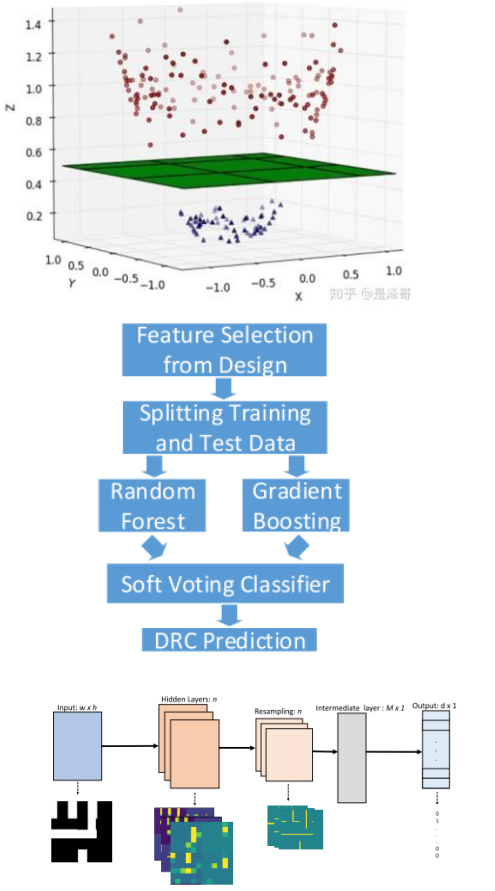
00年代

模式匹配



10年代

人工智能



20年代

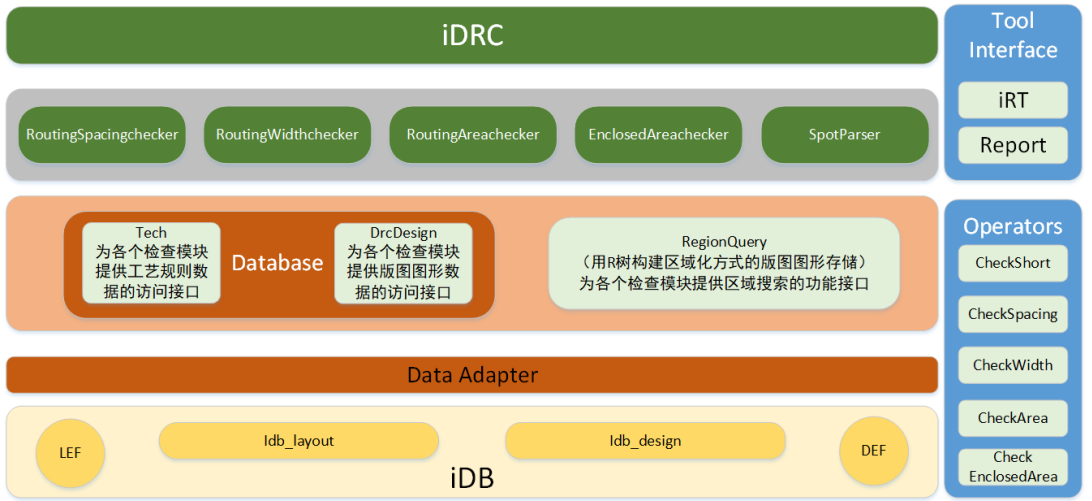
01 研究问题

02 研究内容

03 未来展望

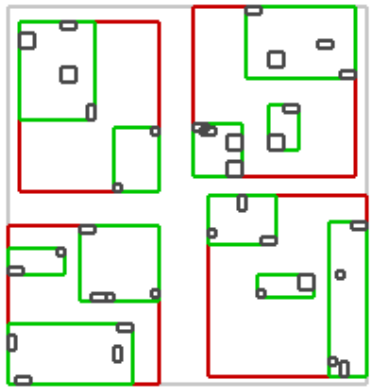
研究内容一 纯规则的检测

- **几何检测**
  - Region Query 将不同层的 Routing 矩形、Fixed 矩形、Cut、Block Edge、Routing Edge、Scope 等组织成不同的 R-Tree，由 R-Tree 强大的检索能力找到有可能与当前矩形产生违例的其他矩形，并进行详细检查。
  - 将详细检查结果输出到二进制文件中，供 GUI App 读取。

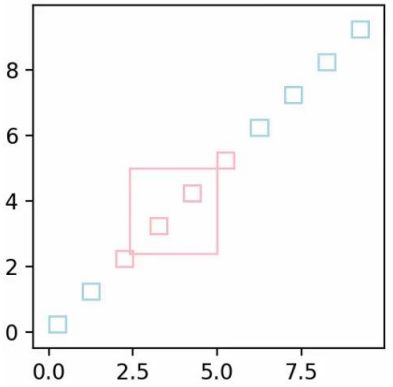


研究内容二 几何检查

R-Tree

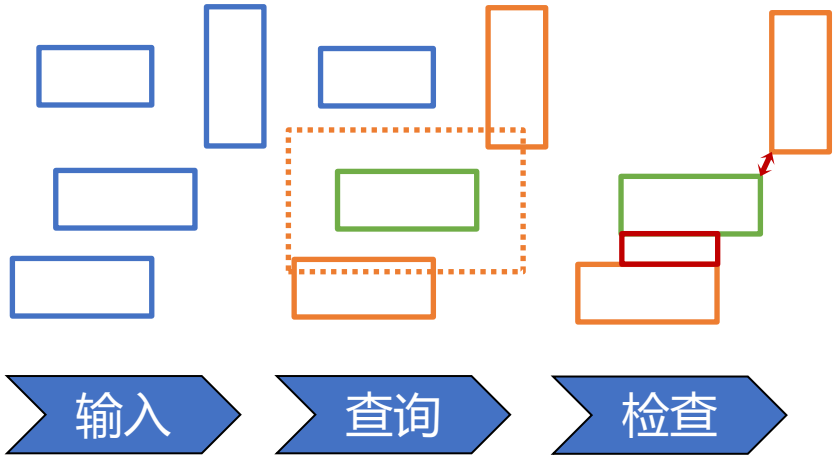


R*-Tree

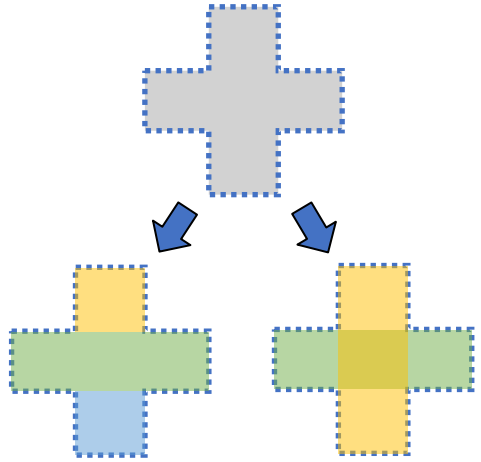


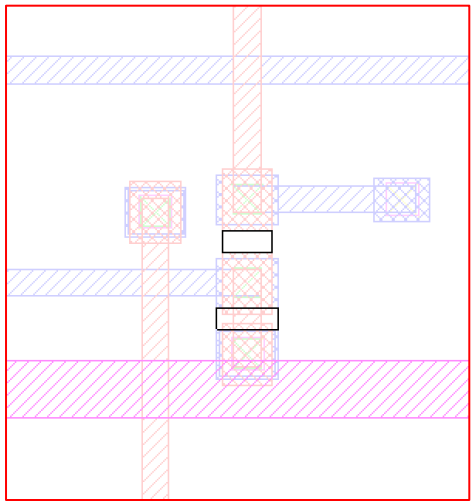
intersects

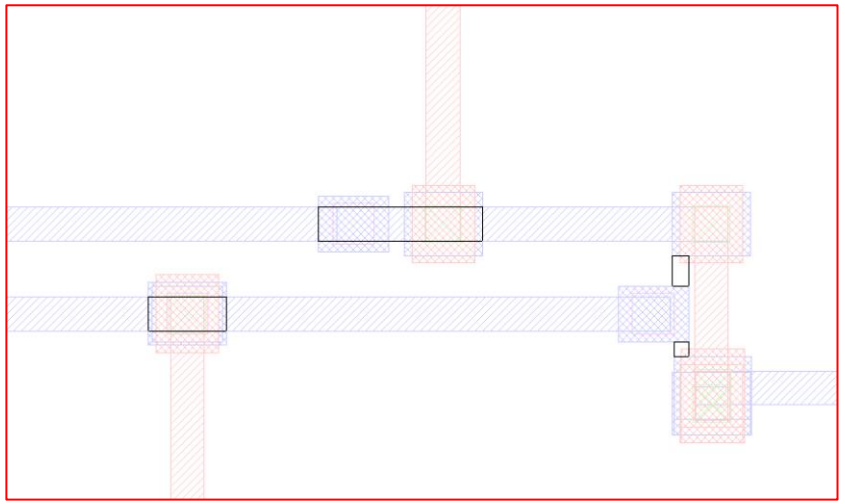
检查思路



矩形分割

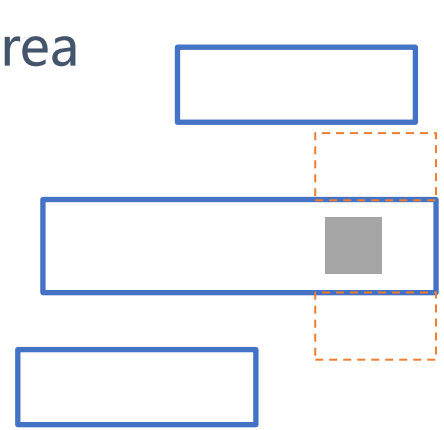




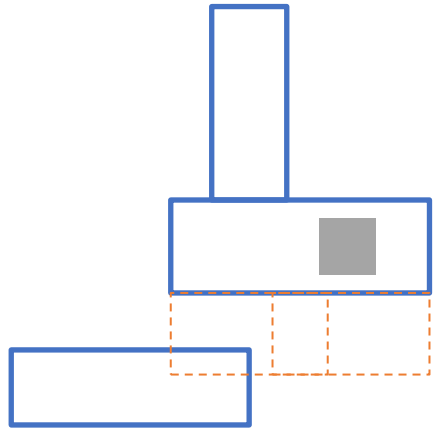


研究内容三 已支持的设计规则

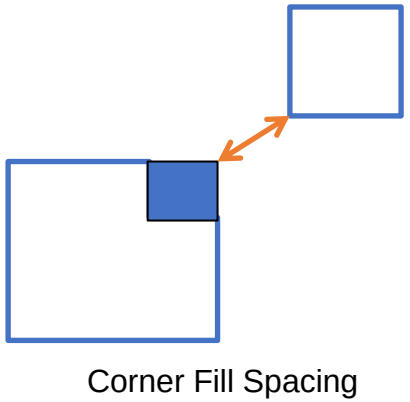
- **当前已基本支持 28nm 工艺下的设计规则检查**
  - ✓ Spacing
    - Routing Spacing, PRL Spacing, Cut Spacing, EOL Spacing, Notch Spacing, Corner Fill Spacing, Jog Spacing, Cut EOL Spacing
  - ✓ Routing Width
  - ✓ Area
    - Routing Area, Enclosed Area
  - ✓ Minimum Step
  - ✓ Short



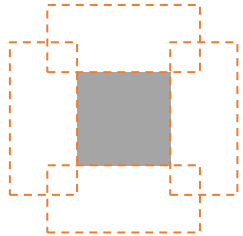
Cut EOL Spacing(1)



Cut EOL Spacing(2)



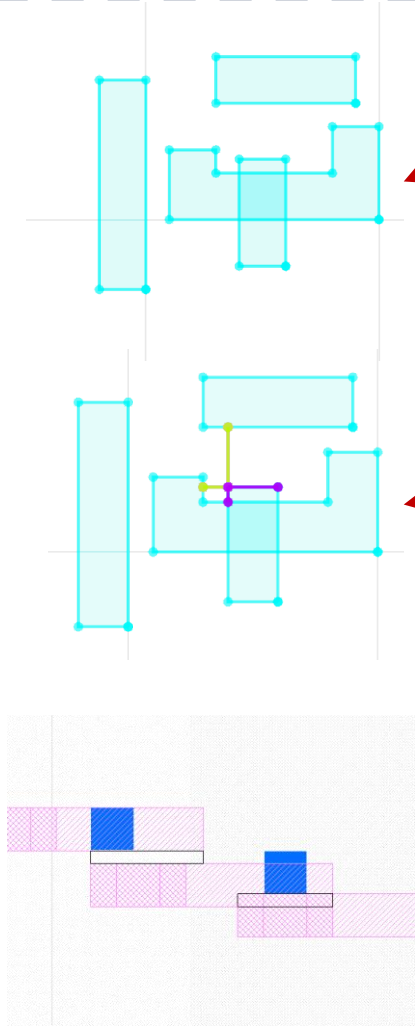
Corner Fill Spacing



Cut Spacing

研究内容四 新的软件框架

检测流程



输入多边形端点

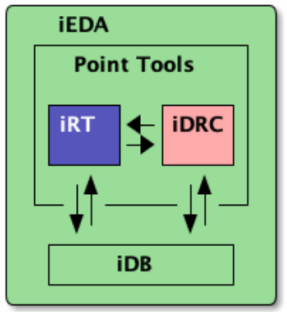
数据转换

构建数据关系

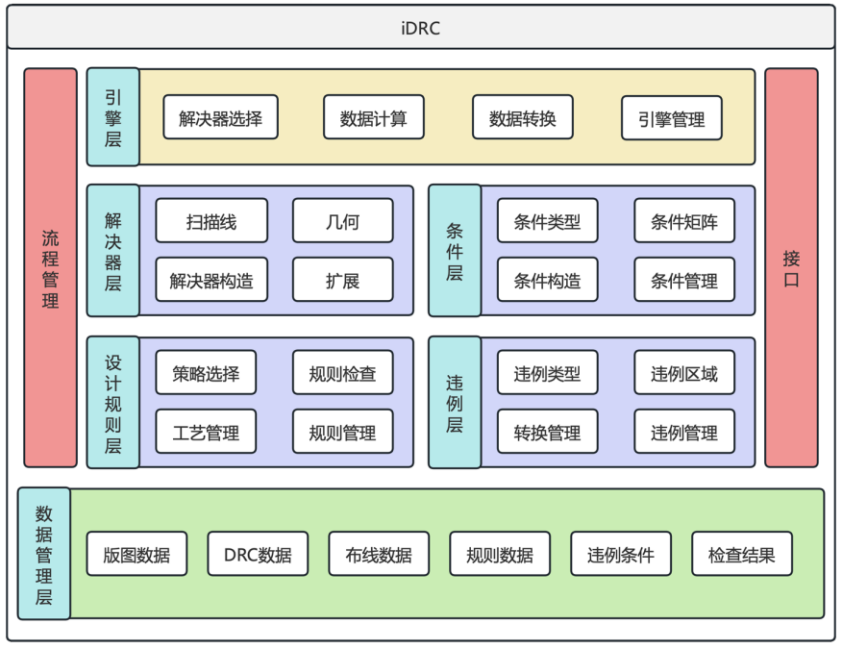
数据筛选
（数值、关系）

违例确认

软件架构



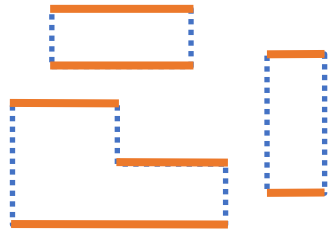
作为独立的点工具，
一方面可以与 iRT协作，
另一方面也可以对整个版图
进行设计规则检查



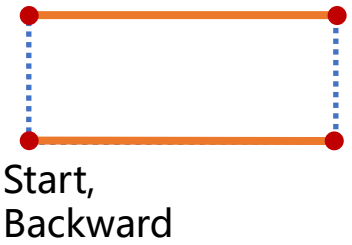
研究内容五 扫描线算法

数据表示

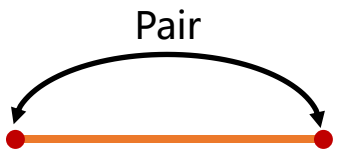
用边表示多边形



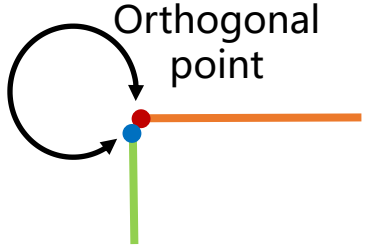
用两个点表示边



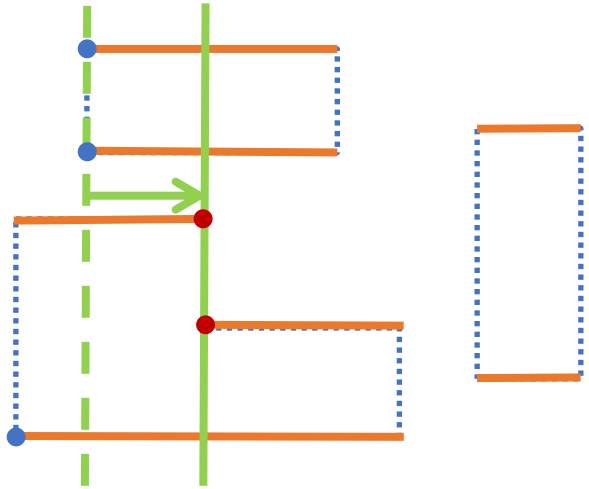
链接同一条边



链接同一位置

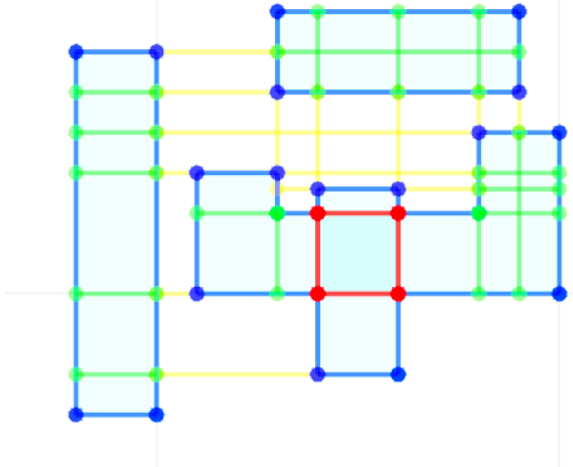


扫描线



- 添加bucket
- 处理扫描线
- 删除stop点

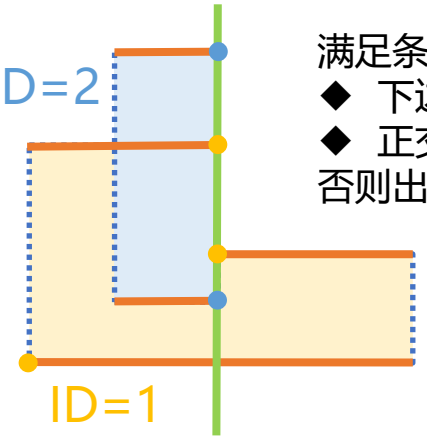
点网



- Overlap
- Edge
- Interspacing
- Width
- Spacing

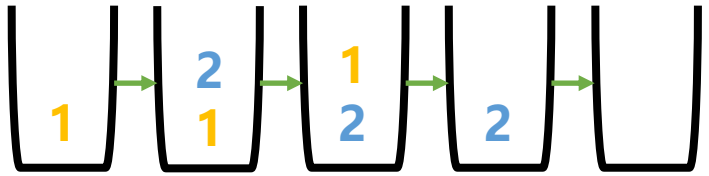
研究内容五 扫描线算法

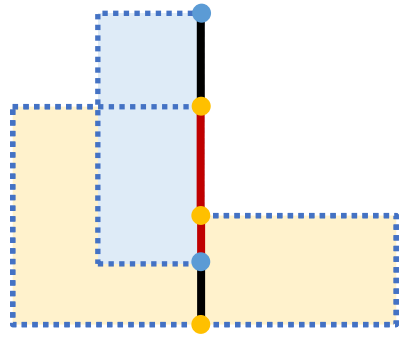
Polygon激活



满足条件之一则入栈：
- 下边沿
- 正交方向起始点

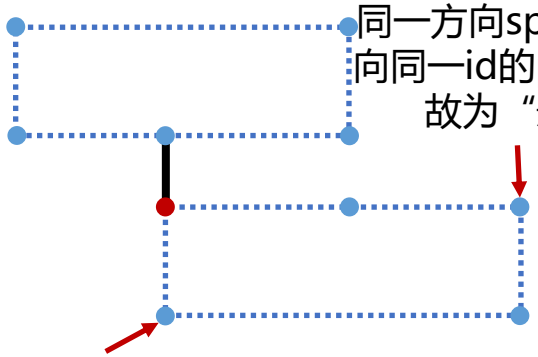
否则出栈





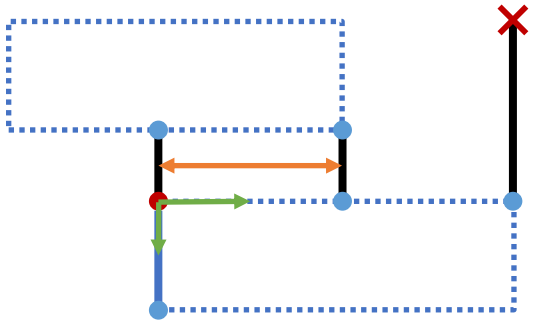
- Overlap
- Edge
- Interspacing
- Width
- Spacing

PRL



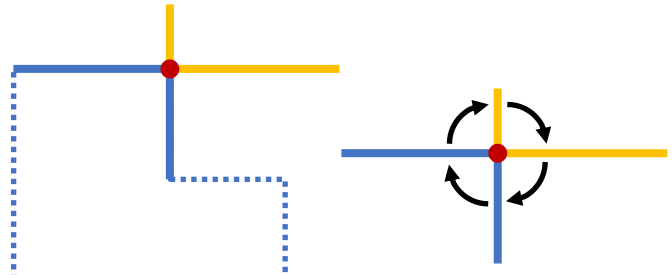
同一方向spacing不指向同一id的polygon，故为“悬崖”

因为没有同一方向的spacing，故为“悬崖”

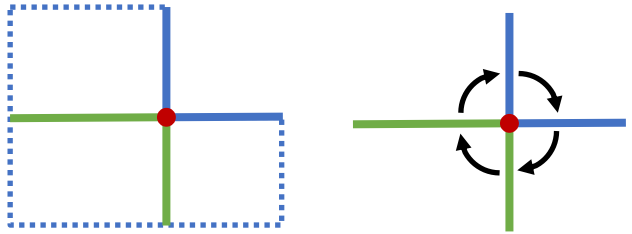


沿着polygon走到“悬崖处”求PRL

凹凸角



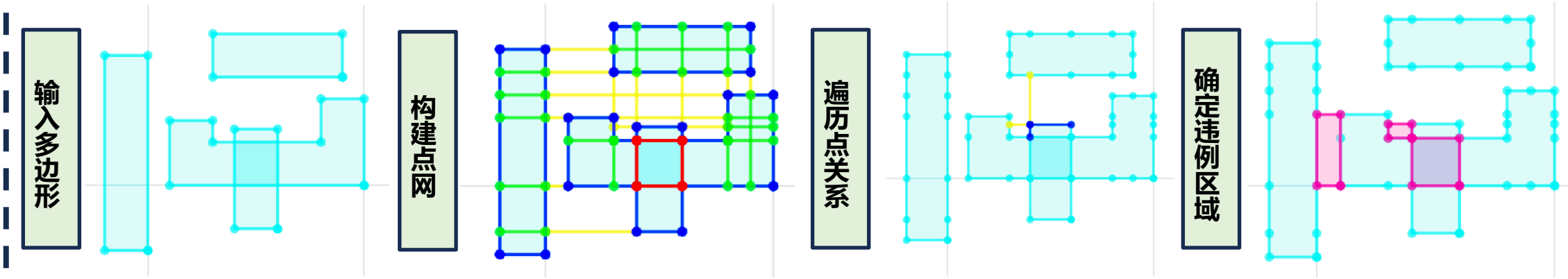
连续两个Spacing+连续两个Edge凸角



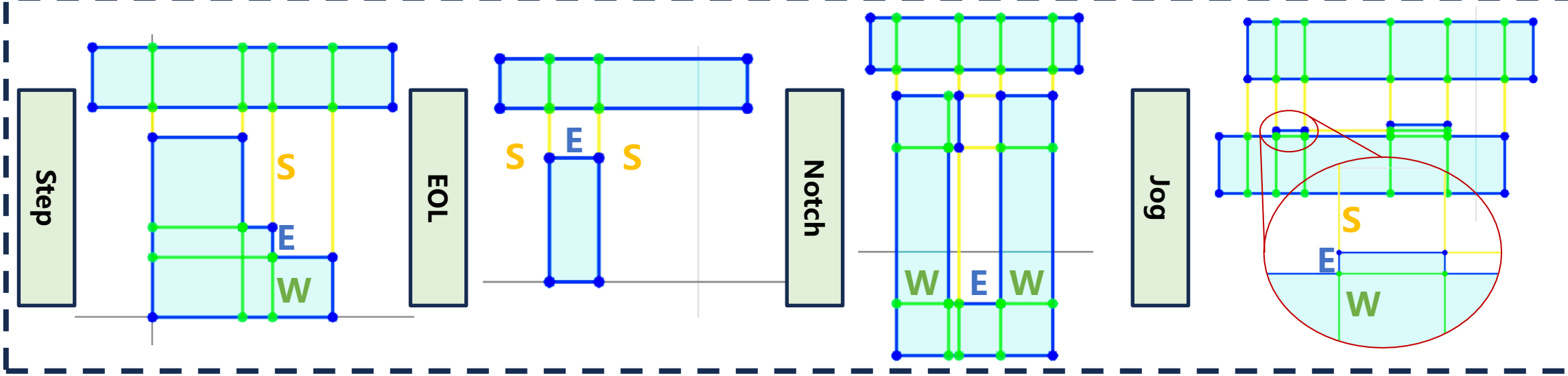
连续两个Width+连续两个Edge凹角

研究内容六 数据关系

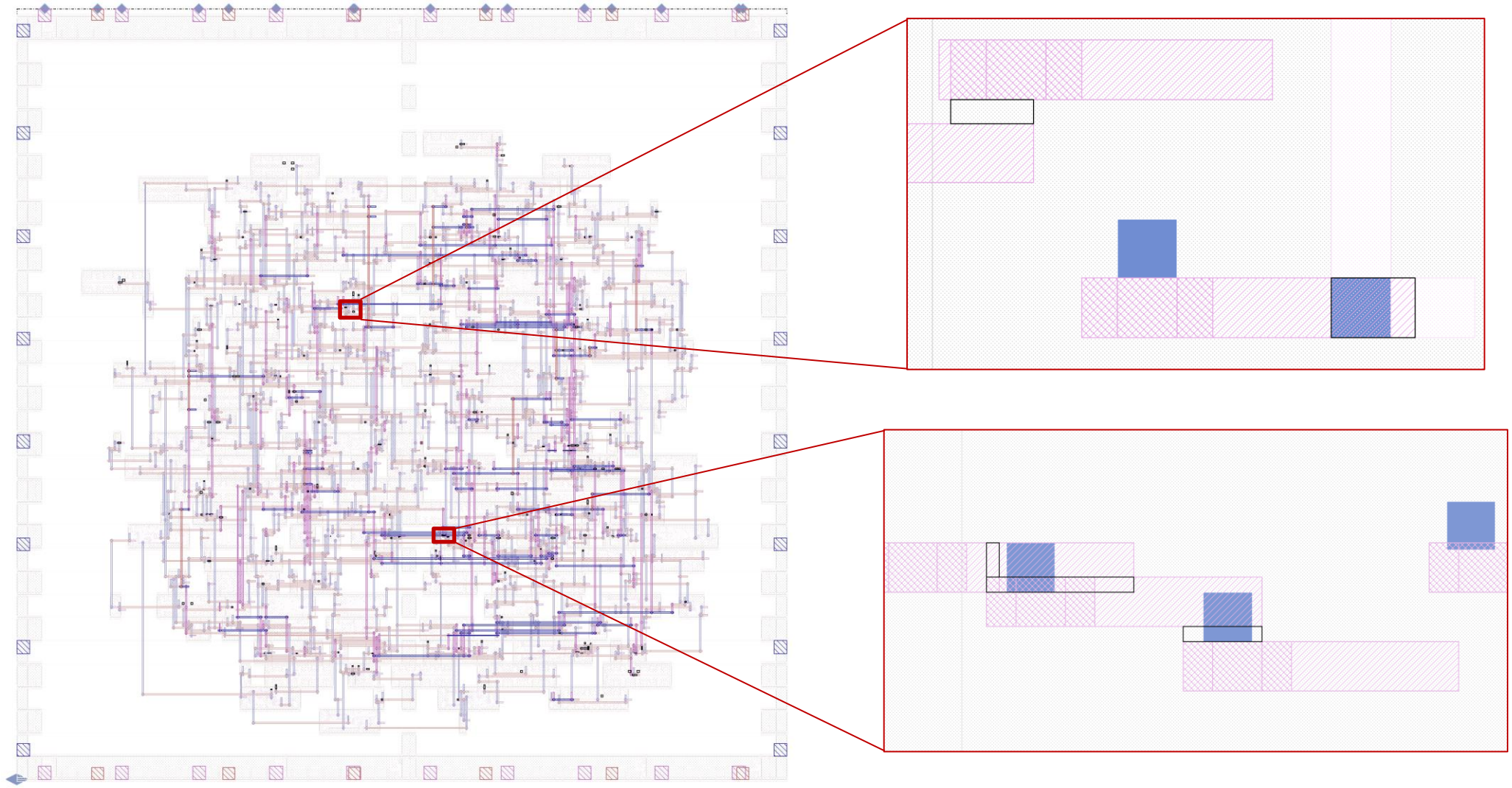
处理流程



数据过滤



研究内容三 检查结果展示



01 研究问题

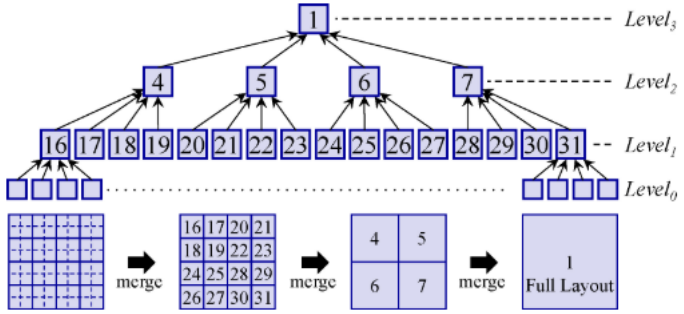
02 研究内容

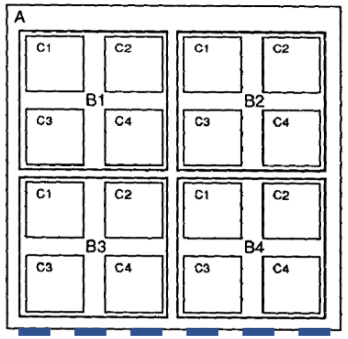
03 未来展望

未来展望一 并行技术

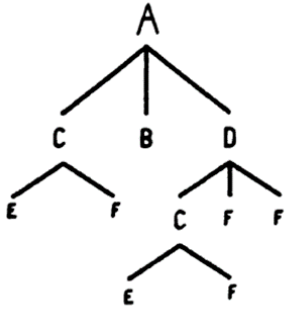
分区和分层方案

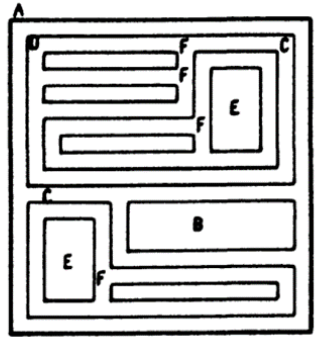
等面积划分





层次化重复





重叠交互问题的处理

扩展重叠区域

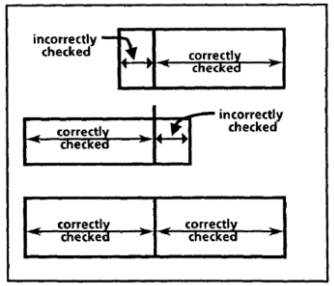
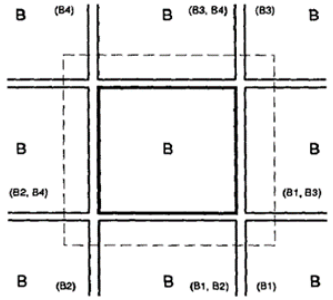
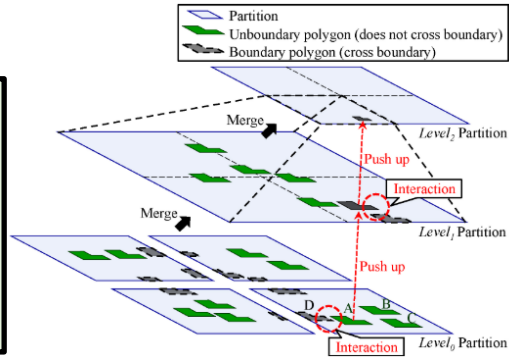
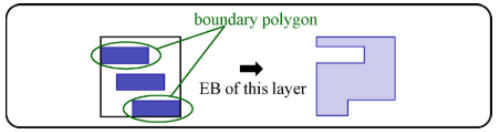


FIGURE 6: Remove the overlap which is incorrectly checked, leaving a correctly checked design

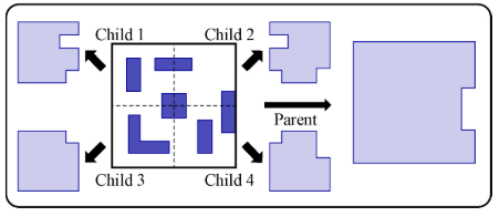


引入估计边界



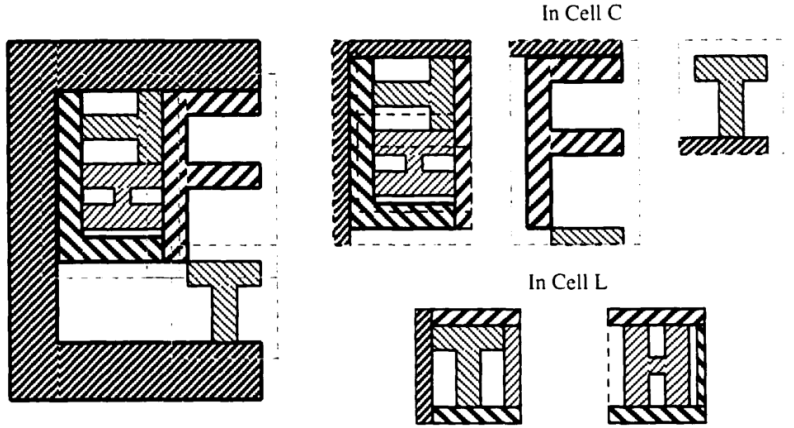


(a)

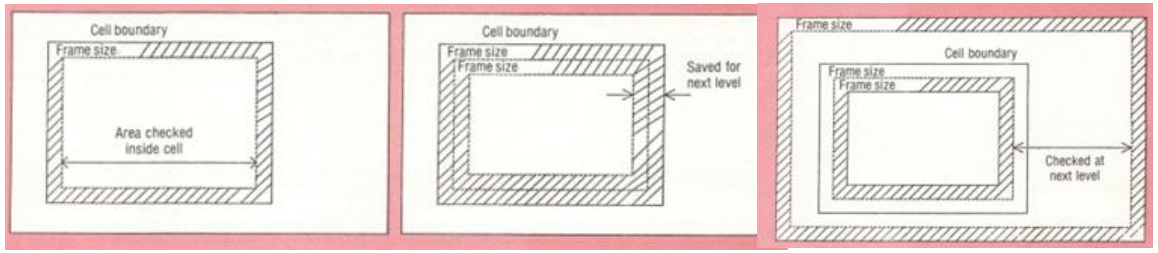


(b)

Halo 算法



划分交互区域



未来展望二 模式驱动的验证

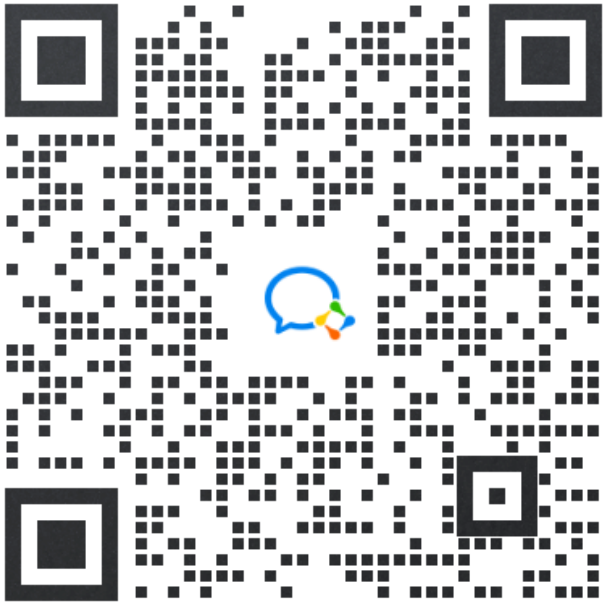
- **Pattern Matching**
  - 工艺的演进和诸如分辨率增强（RET）和光学临近修正（OPC）等技术的应用，确保可制造性的几何规则越来越复杂。
  - 一些 DRC Clean 的设计，对于制造来说也会非常困难。某些特定的图形会破坏可制造性，如果只是单纯地在设计规则上打补丁，由于情形不具有通用性，可能导致为了避免违例而出现另一种难以制造的图形。
  - 将这些发现的难以制造的图形做成 Pattern，使用 Pattern Match 技术在传统的 DRC 之上增强设计规则的检查，指导设计以提高流片成功率。

未来展望三 AI预测

- **预测 DRC 热点**
  - 芯片设计越发复杂，工艺越发先进，单位面积的芯片上集成度越来越高，给 DRC 工具提出了新的挑战。在实施流程的早期解决 DRC 挑战有助于加快 DRC 和时许收敛，同时改善 PPA[1]。
  - 机器学习算法利用 DRC 热图进行训练，找到产生违例的根本原因，预测 DRC 热点。
- **预测未来迭代**
  - 机器学习算法针对不同的 DRC 命令运行集作出针对性的调度策略调优，减少重复计算，更有效利用 CPU 资源。

[1] Synopsys. Machine-Learning....Everywhere![EB/OL]. (2023-10-26)[2023-10-26] https://www.synopsys.com/implementation-and-signoff/resources/blogs/looking-past-horizon/machine-learning-everywhere.html

iEDA开源交流群



感谢聆听

Thanks for your attention