iEDA-Tutorial 第一期

时序（iSTA）和功耗（iPW）分析工具

陶思敏、龙帅英、刘贺、邵哲青

2023/06/30

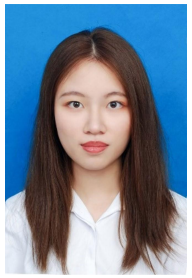
iEDA Tutorial 第一期议程

- Part1 iEDA-iSTA和iPW整体介绍 40min （陶思敏）

- Part2 iSTA工具架构、特性、API与使用 25min（龙帅英）

- Part3 iSTA关键技术研究 30min（刘贺）

- Part4 iPW工具架构、特性、关键技术与使用 25min（邵哲青）



研究总览-动机

- **动机（Why）**

为什么要研发自己的时序和功耗工具，现有开源工具OpenTimer、OpenSTA存在什么问题?

- OpenTimer对时序分析的功能特性支持不够，如CCS模型，Arnoldi降阶。另外对输入数据的parser可扩展性不好，如liberty（不支持CCS），verilog（不支持Hierarchy结构），进一步扩展成本较高；
- OpenSTA的GPL协议、使用限制，以及代码理解成本，使用难度等；

OpenSTA is open source, meaning the sources are published and can be compiled locally. Derivative works are supported as long as they adhere to the GPL license requirements. However, OpenSTA is not supported by a public community of developers as many other open source projects are. The copyright and develpment are exclusive to Parallax Software. OpenSTA does not accept external code contributions.

研究总览-目标

- **动机（Why）**

有需求，可以推动工具不断演进。

  - iSTA和iPW集成在iEDA物理设计工具中，提供时序和功耗的Engine，全物理设计工具链条集成，协同创新，对其有API接口需求；
  - 芯片设计（如：一生一芯、香山等开源芯片）的用户使用需求；
  - 云化（Cloud Compute）和人工智能（AI）的发展对EDA工具技术升级需求。

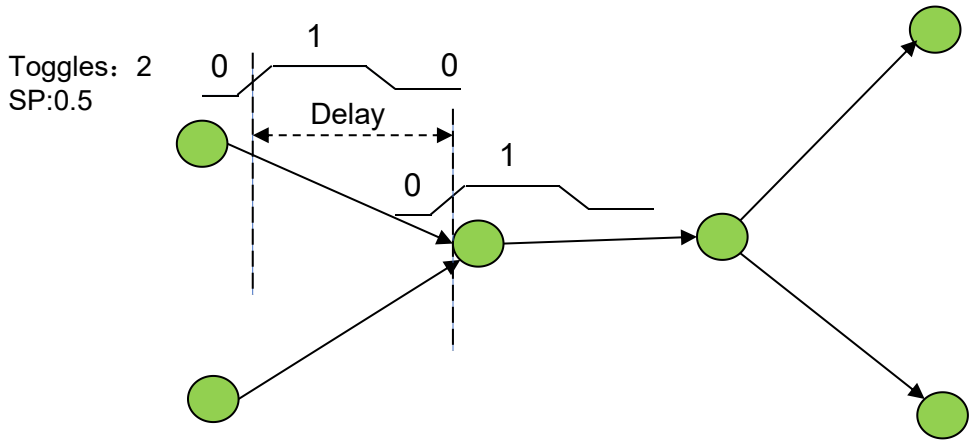
- **目标（Goal）**

科研、工程结合，做创新实用的静态时序分析和功耗分析工具，从实际使用中提炼出科研问题，科研成果落地到工具中。

研究总览-问题

- **问题（What)**

电路网表可以抽象成电路图DAG（V，E），时序分析是检查波形传播的延时（Delay）是否违反时序要求，功耗分析是根据波形翻转率（Toggle）计算动态功耗，以及静态概率（SP）计算静态功耗。



Delay：信号传播时延
Toggle：信号发生翻转的次数
SP：信号为高电平的概率

研究总览-挑战

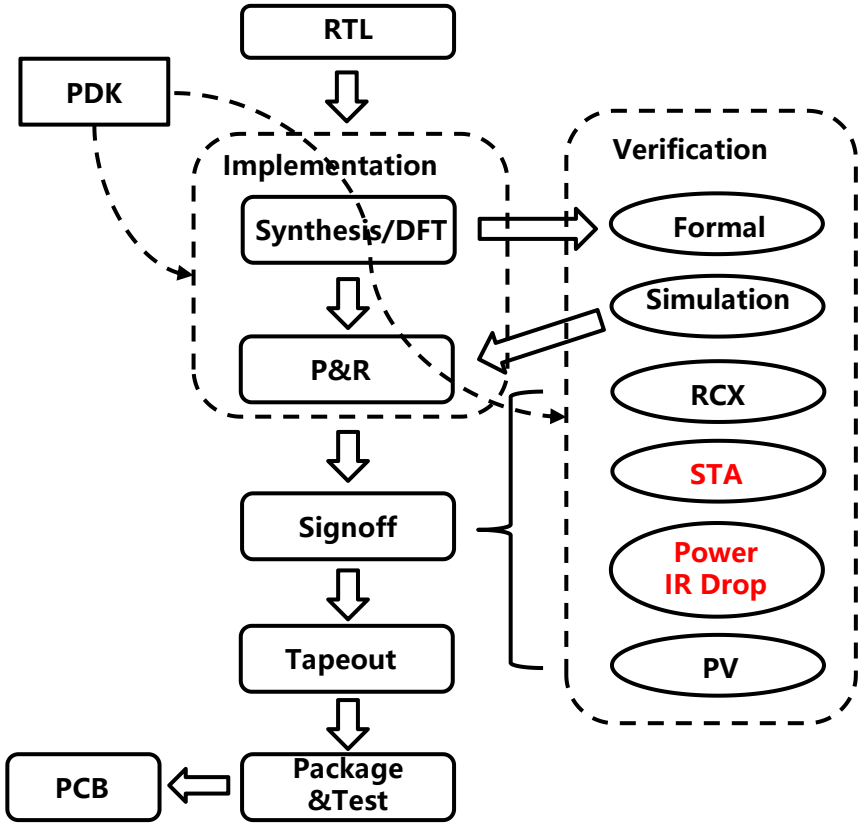
- **挑战（Challenge）**
  - 时序和功耗分析的难点在于准确的计算延时（Delay），数据翻转率（Toggle）/静态概率（SP）。如Delay受到波形传播的影响，在互连线上的传播受互连线电阻电容（RC）的影响，电容有耦合电容，米勒电容等复杂因素，另外就是互连线之间的CrossTalk、信号完整性等因素，而Toggle/SP传播难以用一个标准的公式计算，随单元类型不同而不同；
  - 问题的规模比较大，时序路径经过组合逻辑的Fanout组合，十万规模单元的芯片可能会有上千万条时序路径。
  - 为了平衡运行时间和精度，需要找到近似求解法如低阶方法，快速近似计算。

研究总览-方法

- **方法（How）**
  - 网表转换成图的方法，包括缩减图的规模，图计算、划分；
  - 电路图上并行、分布式、加速方法（GPU加速，云化等）；
  - 波形传播延时计算近似方法（传统解析法Pade近似，泰勒展开近似，机器学习ML方法）；
  - 翻转率和静态概率传播计算方法（传统公式法，Cycle统计法、机器学习ML方法）。

研究总览-静态时序分析和功耗分析

- **静态时序分析和功耗**
  - 静态时序分析和功耗分析在芯片设计中所处位置如图所示，在综合（Synthesis）和物理实现（PR）以及签核（Signoff）阶段都可能需要调用STA和Power等。
  - 在综合和物理实现中，此时可能物理信息还不完整，需要用一些模型比如WLM（wire load model）、斯坦纳树（Steiner Tree）模型来预估互连线RC和延时情况。
  - 在签核阶段，此时有了完整的RC信息，可以通过方程求解真实的延时和功耗情况。



研究总览-静态时序分析

- **“静态”时序分析的含义**

时序分析可以通过以下几种方式完成：

1）电路仿真（spice） 优势：准确度最高 缺点：运行时间最长

2）带SDF的门级Verilog仿真 优势：考虑电路功能 缺点：运行时间长

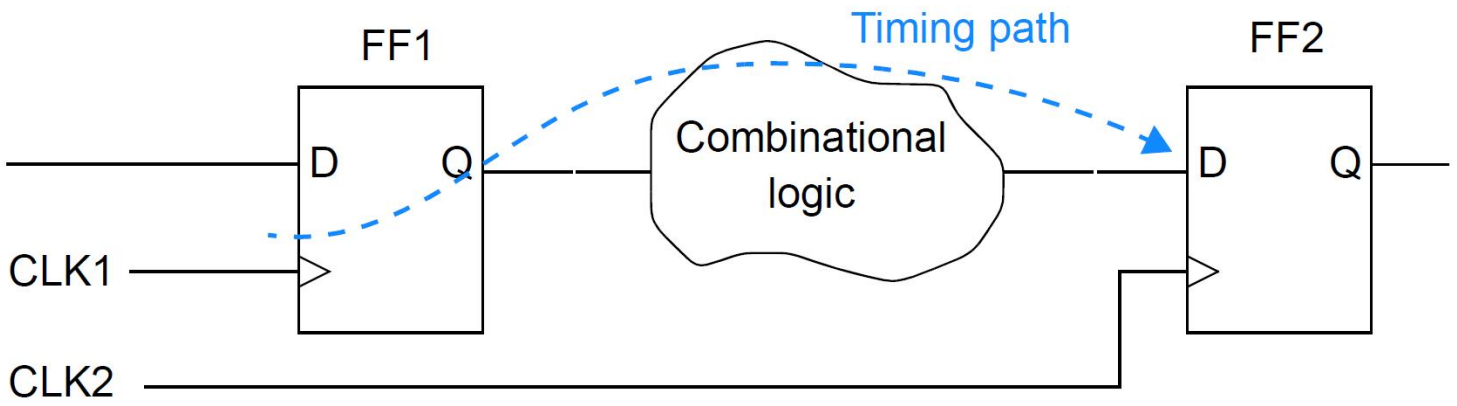
3）静态时序分析（STA）实现 优势：运行时间最快 缺点：不考虑电路功能，准确度比不过spice

相比于前两者，“静态”可以不依赖输入端口激励，只分析单个时钟周期内的时序worst情况，快速判定电路能否以指定频率运行。工程上往往多种方法结合，每种方法都有各自的应用场景。

研究总览-静态时序分析

- **iSTA（静态时序分析工具）**

  如图所示，是一条典型的Flip-Flop（FF）到Flip-Flop之间的时序路径。



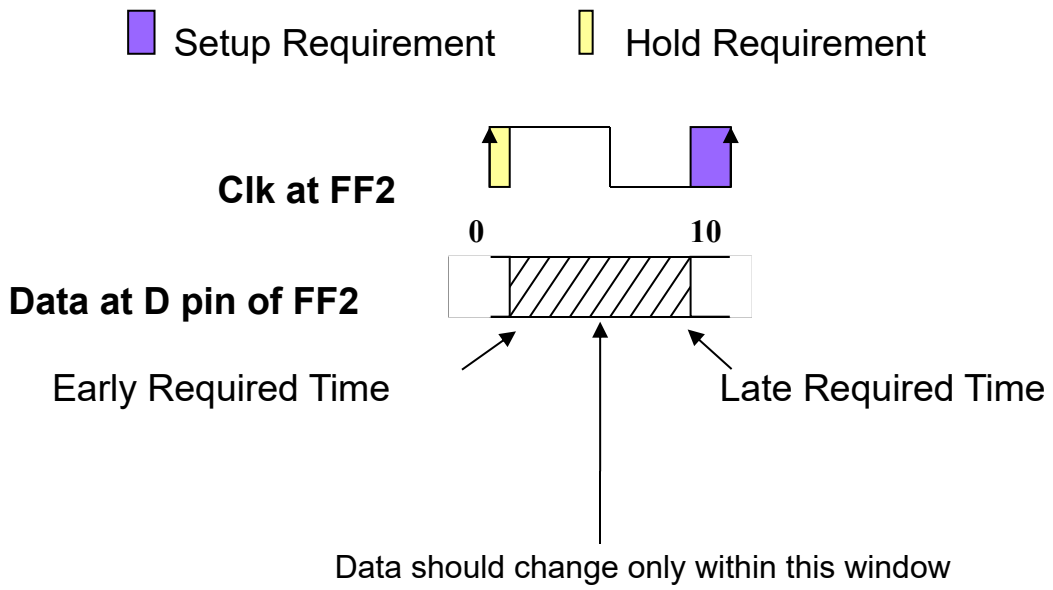
研究总览-静态时序分析

- **iSTA（静态时序分析工具）**
  - 静态时序分析检查如图所示，捕获Flip-Flop接收数据需要满足Setup和Hold要求。Setup检查是在发射Flip-Flop发送数据后的下一个周期（图中周期10ns），数据需要在周期减去Setup时间前到达，需要满足如下公式，周期对应芯片的频率受到此公式的限制：

    $$Tlaunch + Tck2q + Tdp < Tcapture + Tcycle - Tsetup$$

  - Hold检查则是在发射Flip-Flop发送数据后的当前周期，确保数据的最早到达时间不能早于Hold时间，需要满足如下公式：

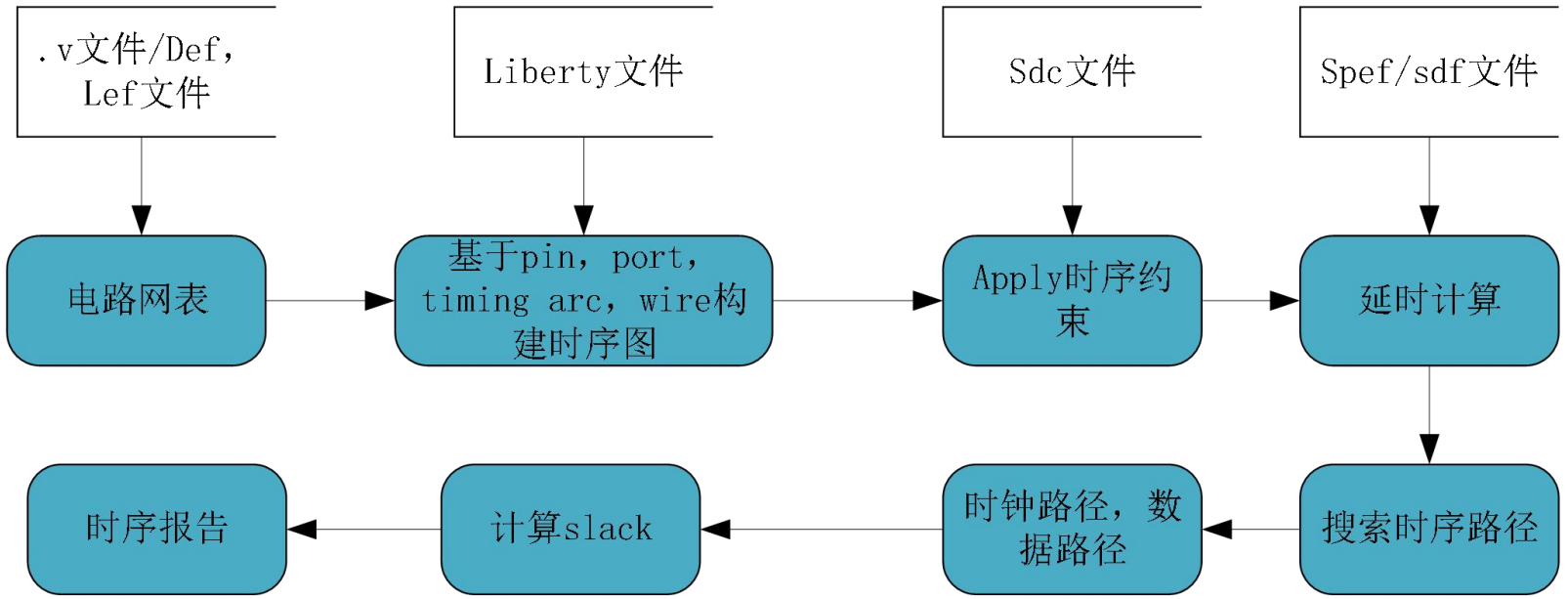
    $$Tlaunch + Tck2q + Tdp > Tcapture + Thold$$



研究总览-静态时序分析

- **iSTA（静态时序分析工具）**

  时序分析工具流程如下，在把网表转为电路图后，就抽象为图遍历问题，搜索出时序路径进行分析和检查。

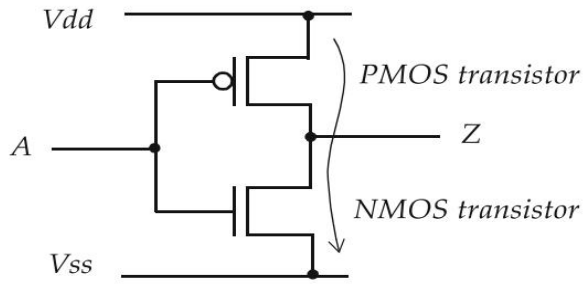


研究总览-低功耗设计和功耗分析

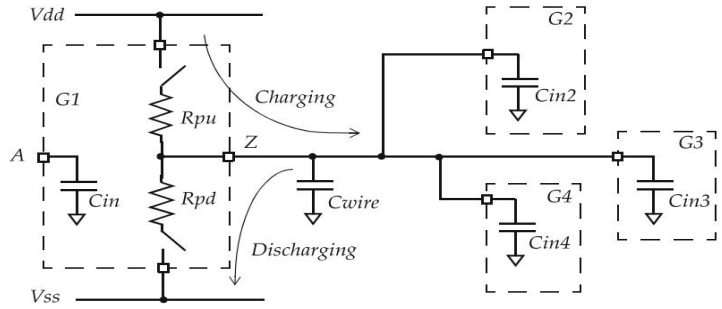
- **低功耗设计和功耗分析**
  - 低功耗设计主要通过开关关断电源（ power gate ）或者时钟（clock gate），或者多电源域设计（CPF/UPF）等；
  - 功耗分析是计算芯片的功耗，需要考虑低功耗的设计对功耗的影响。但我们现在的功耗分析（iPW）工具还没有支持对power gate，clock gate，多电源域设计等低功耗设计进行分析，当前只支持了一个无上述功能的基础版本。
  - 功耗分析可以读入VCD等波形文件反标翻转率到信号上，也可以通过设置输入翻转率借助传播的方式得到后续信号翻转率，因此支持静态和动态分析两种方式。

研究总览-功耗分析

- **iPW（功耗分析工具）**
  - 功耗分析主要分析芯片内的三种功耗漏电功耗（Leakage Power），内部功耗（Internal Power），开关功耗（Switch Power）；
  - 漏电功耗是工艺特性决定的，是单元在没有信号翻转时候的功耗；
  - 内部功耗：是单元内部在发生信号翻转时的功耗，包括短路功耗、内部的充放电功耗；
  - 开关功耗：是互连线在发生信号翻转时的功耗。



短路功耗

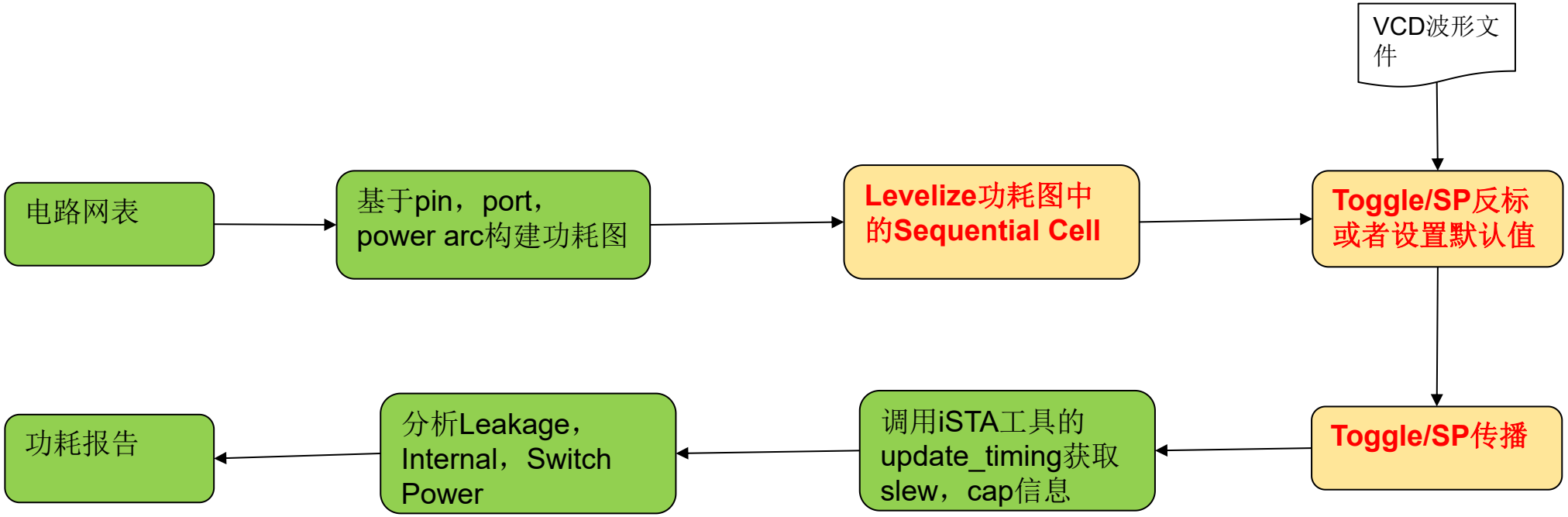


开关功耗

研究总览-功耗分析

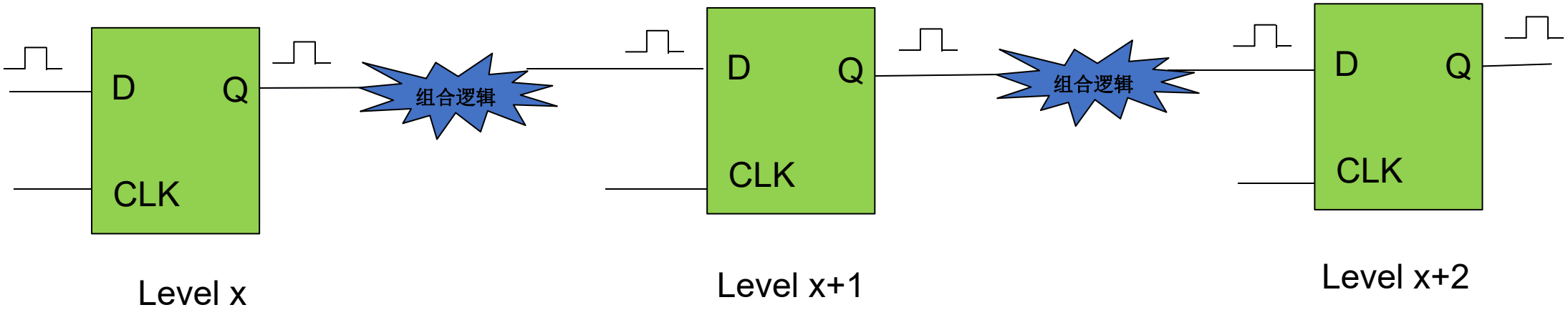
- **iPW（功耗分析工具）**

  功耗分析工具流程如下，在把网表转为电路图后，就抽象为图遍历问题，按照Level进行时序路径传播，从iSTA工具中获取时序方面的数据，最终计算出静态（Leakage Power），动态（Internal，Switch Power）。



研究总览-功耗分析

- **iPW（功耗分析工具）**
  - 由于时序路径之间存在数据翻转率的传递，因此需要对时序路径进行分级；
  - 分级后进行功耗Toggle/SP传播；



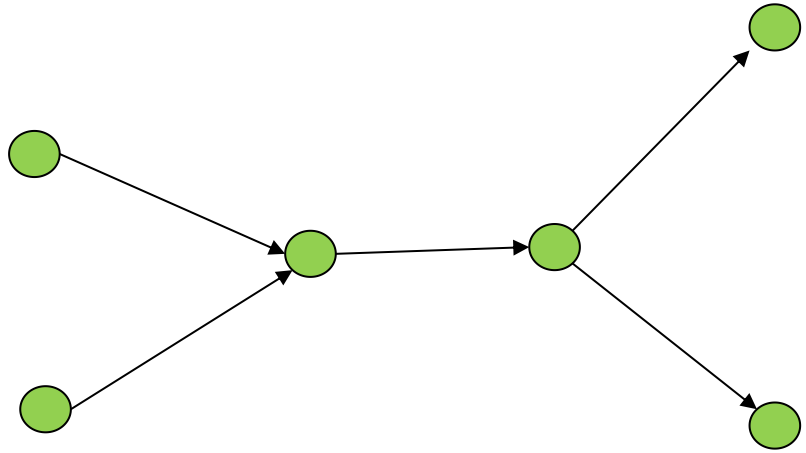
研究内容一：电路图上并行、分布式、云化加速

- **研究动机**

在电路图（DAG）上做波形传播（Delay/Toggle），由于路径的规模非常大，因此需要使用分布式并行技术。

- **问题描述**

如图所示，DAG上传播是有方向的，节点之间存在依赖关系，通过建立线程之间的依赖关系，让传播复杂度能控制在O(n+m)。



- **可行方法**

多线程并行技术、多机分布式、云化

研究内容二：单元延时计算插值方法

- **研究动机**

对于单元内部的延时，是从lib库中通过查表LUT得到。
由于波形本身的复杂性，使用线性插值虽然能比较快速获得结果，但可能会出现外插或者部分不符合线性部分造成的误差。

- **问题描述**

给定NLDM/CCS单元模型，使用不同插值方法获得符合波形规律的结果。

- **可行方法**

线性插值、AI插值、其他传统插值

```
pin (OUT) {
  . . .
  timing () {
    related_pin : "IN"-;
    . . .
    output_current_fall () {
      vector ("LOOKUP_TABLE_1x1x5") {
        reference_time : 5.06; /* Time of input crossing
          threshold */
        index_1("0.040"); /* Input transition */
        index_2("0.900"); /* Output capacitance */
        index_3("5.079e+00, 5.093e+00, 5.152e+00,
                 5.170e+00, 5.352e+00");/* Time values */
        /* Output charging current: */
        values("-5.784e-02, -5.980e-02, -5.417e-02,
                -4.257e-02, -2.184e-03");
      }
      . . .
    }
    . . .
  }
  . . .
}
```

```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP";
    timing_sense : negative_unate;
    rise_transition(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values ( /*   0.16      0.35     1.43 */ \
        /* 0.1 */  "0.0417,  0.1337,  0.4680", \
        /* 0.3 */  "0.0718,  0.1827,  0.5676", \
        /* 0.7 */  "0.1034,  0.2173,  0.6452");
    }
    fall_transition(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values ( /*    0.16      0.35     1.43 */ \
        /* 0.1 */   "0.0817,  0.1937,  0.7280", \
        /* 0.3 */   "0.1018,  0.2327,  0.7676", \
        /* 0.7 */   "0.1334,  0.2973,  0.8452");
    }
```

研究内容三：互连线延时计算近似快速方法

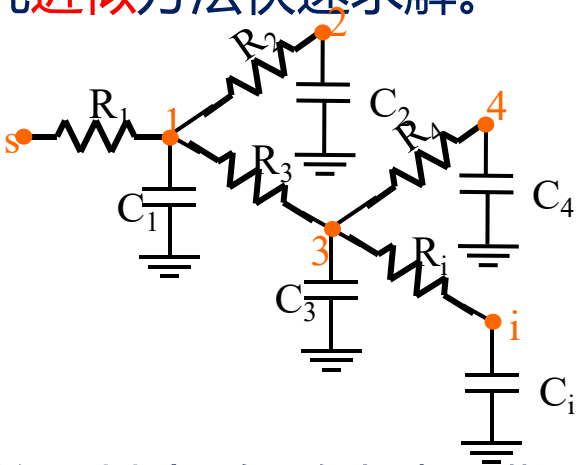
- **研究动机**

互连线方程的求解需要做到快速准确。在计算延时时，对于信号的波形并不要要求波形上所有的点都精确，只要保证波形的极值点是对的就可以。因此可以研究近似方法快速求解。

- **问题描述**

$$C\frac{dx(t)}{dt} = -Gx(t) + Bu(t)$$

$$y(t) = L^T x(t)$$



通过构造近似传递函数Hr(s)替代原传递函数H(s)，达到快速求解目的。其中C矩阵考虑米勒效应后会发生变化，C是对角矩阵，G是对称矩阵，对角为0，B是输入向量选择，u是随时间变化的电流，x是节点电压。

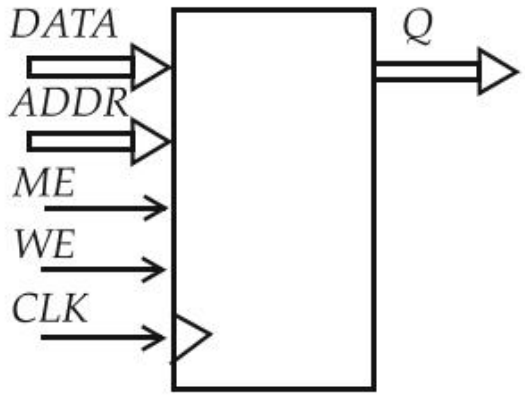
- **可行方法**

模型降阶方法、微分方程近似求解等

研究内容四：信号翻转率计算

- **研究动机**

根据信号波形文件（如VCD文件）直接统计输出信号翻转率是一种比较慢的方法，而根据公式可以快速得到结果。



- **问题描述**

给定输入信号翻转率，怎么根据公式得出输出信号翻转率，单元类型的不同，公式也会不同，有时可能很难给出一个公式。

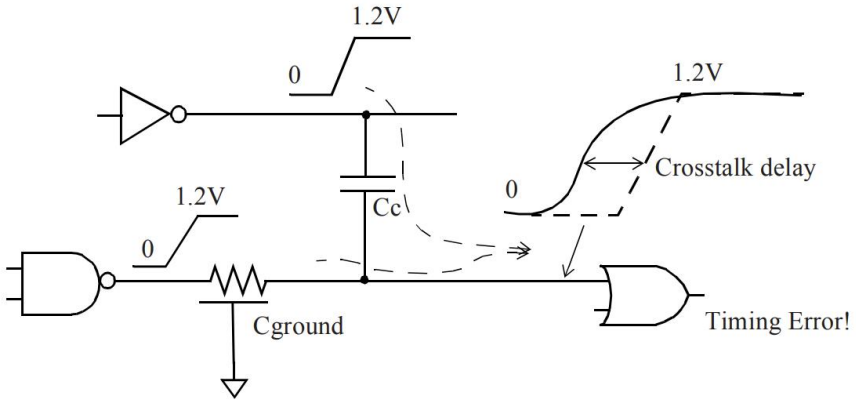
- **可行方法**

推导公式法、启发式方法、ML方法、加速波形读入Cycle级分析

研究内容五：CrossTalk对时序的影响

- **研究动机**

随着工艺的演进，邻近net翻转造成的CrossTalk delay对net delay占比在变大，因此需要快速准确计算这部分delay。

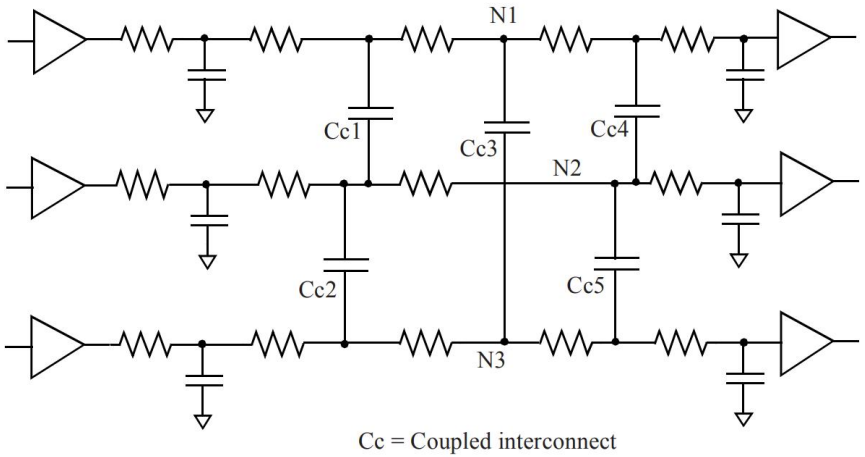


- **问题描述**

CrossTalk Delay快速计算以及多条aggressor对victim的影响造成的波形叠加的复杂性，并且由于耦合的原因，降阶使用的波形近似方法也受到影响。

- **可行方法**

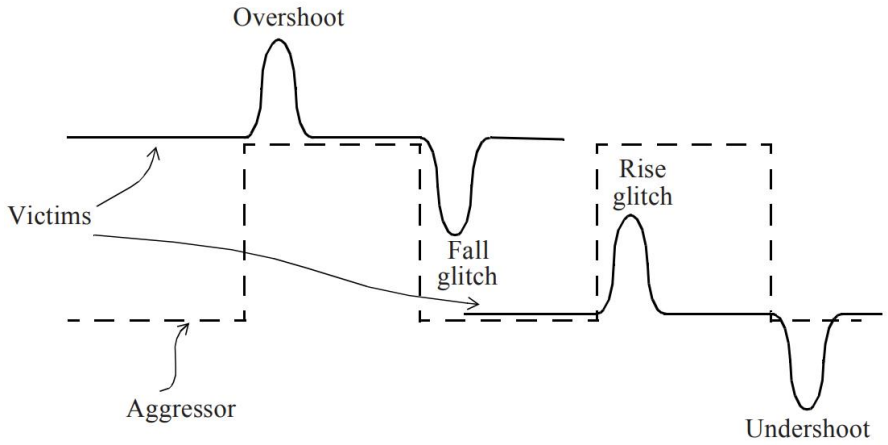
耦合公式2-pi模型法，ML预测CrossTalk Delay



研究内容六：Glitch对功耗的影响

- **研究动机**

Glitch从信号波形上看是一个脉冲信号，这种无意义的信号会造成无效的功耗，且随着工艺的演进，Glitch功耗在功耗中的占比也在变大。



- **问题描述**

Glitch可以由耦合电容产生，也可以由组合逻辑运算产生，需要完整全面的分析各种Glitch，并能判断Glitch的传播。

- **可行方法**

时序和功耗结合传统分析方法、ML预测Glitch

研究内容七：ML（GPU加速）辅助时序功耗

- **研究动机**

ML的技术推动带来了很多可用的基础设施，如GPU作为加速手段应用在时序功耗工具中，AI算法做单元插值，微分方程快速求解等。

- **问题描述**

GPU和ML辅助时序功耗分析

- **可行方法**
  - ✓ GPU加速CPPR(Clock Path Pessimism Removal)、 GPU加速图传播、GPU加速时延Delay计算求解等。
  - ✓ ML预测时序（PreRouting时序预测、 CrossTalk预测、 GBA预测PBA 、Multi-Corner预测、 GNN时序分析过程），ML预测功耗（预测Toggle，预测RTL功耗、预测IRDrop）

01 研究内容

02 研究进展

03 未来计划

工具进展一：iSTA工具原型

- **iSTA 1.0完成了静态时序分析的基础功能**

  当前已完成功能：

  - 数据读入支持了：def/verilog，sdc，spef，liberty，且语法维护可扩展性好
  - 支持了时序分析常用算子:
    - ✓ build timing graph(from netlist to DAG)
    - ✓ propagation(Slew，Delay，AT，RT)
    - ✓ delay calculation（NLDM/Elmore，D2M，ECM，MD2M，CCS/Arnoldi）
    - ✓ timing analysis（Setup/Hold，Recovery/Removal，Clock Gate）

工具进展二：iPW工具原型

- **iPW1.0完成了功耗分析的基础功能**

  当前已完成功能：

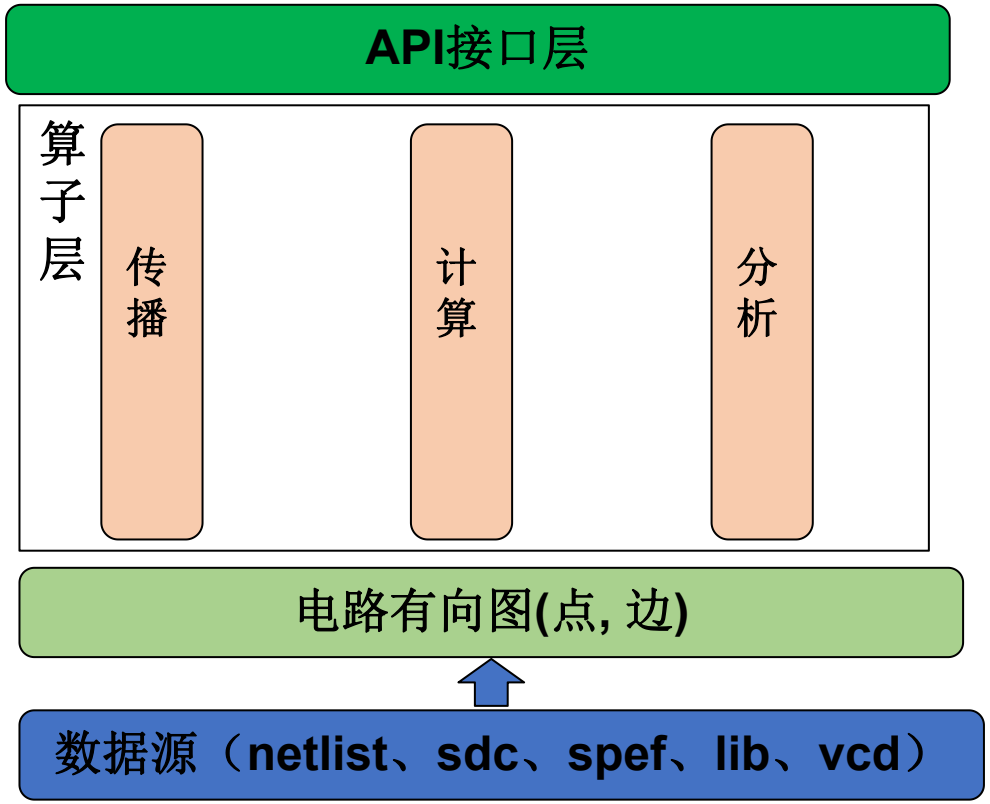
  - 数据读入支持了：VCD波形文件读取，其他复用iSTA工具，后续还将支持SAIF波形文件
  - 支持了功耗分析常用算子:
    - ✓ build power graph(from netlist to DAG)
    - ✓ propagation(Toggle，SP)
    - ✓ toggle/sp calculation（Propagated，Annotated，Default）
    - ✓ power analysis（Leakage，Internal，Switch Power）

iSTA和iPW工具设计理念

- **架构设计**

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。

- ✓ 使用现代C++，基于最新C++20版本语言；
- ✓ 统一数据存储，数据都保存在图DAG上；
- ✓ 数据和算子分离，数据提供统一接口给算子接入，松耦合、易扩展；
- ✓ 统一的API接口层。



- **代码行数**

- ✓ iSTA （包括Parser）代码行数接近3万行，核心代码（不含Parser）2万行，开发时长2.5年;
- ✓ iPW核心代码（不含Parser）代码行数在5000行左右，开发时长0.5年；

iSTA工具和OpenTimer，OpenSTA对比

Feature	iSTA	OpenTimer	OpenSTA
支持Hierarchy网表和def输入	✔	✖	✔
基础的setup/hold分析	✔	✔	✔
支持NLDM/Elmore模型	✔	✔	✔
支持CCS电流源模型	✔	✖	✔
延时计算使用模型降阶	✔	✖	✔
支持sdf反标	✖	✖	✔
OCV	✔	✔	✔
AOCV	✔	✖	✔
POCV	✖	✖	✔
多电压域考虑IRDrop分析	✖	✖	✖
Hierarchy分析	✖	✖	✖
Crosstalk分析	✔	✖	✖
clock gate分析	✔	✖	✔
Latch分析	✖	✖	✔

iPW工具和OpenTimer，OpenSTA对比

Feature	iPW	OpenTimer	OpenSTA/OpenRoad
Leakage Power分析	✔	✔	✔
VCD读取和反标	✔	✖	✔
Toggle/SP传播	✔	✖	✔
动态功耗分析	✔	✖	✔
支持IRDrop分析	✖	✖	✔
支持EM分析	✖	✖	✔
支持Cycle级的功耗分析	✖	✖	✖
支持低功耗设计 CPF/UPF	✖	✖	✖

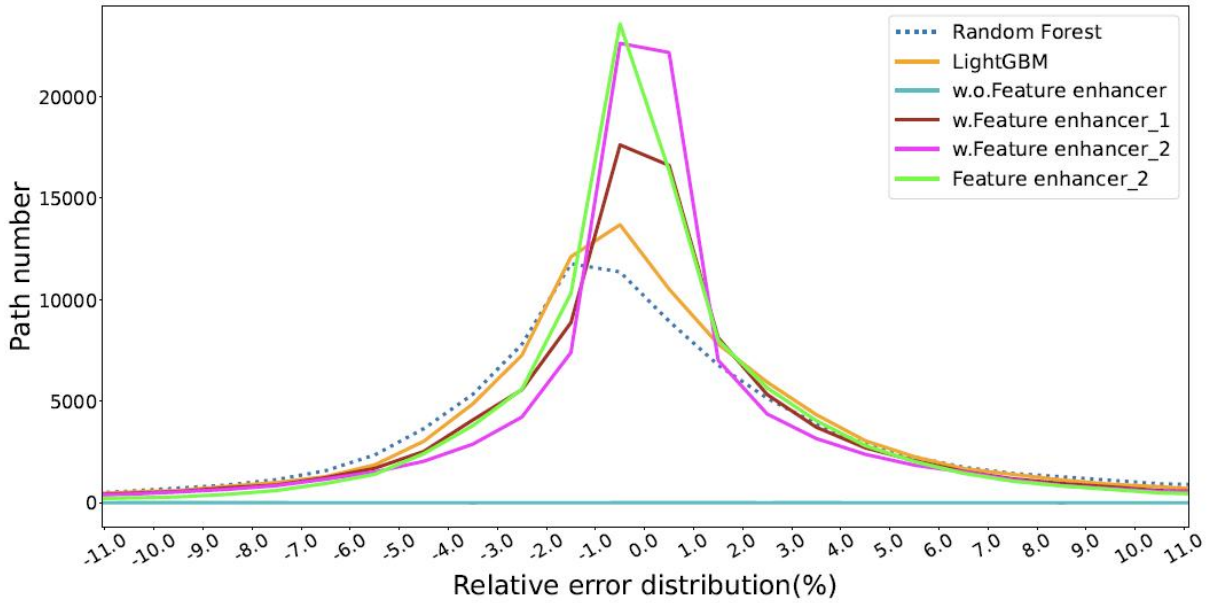
iSTA和PT对比

- 采用开源工艺skywater130，工艺对标商业110nm，在这个水平上iSTA计算出来的路径延时和PT的差距不大。
- 28nm工艺下，对各种先进算法功能要求比较高，如考虑CrossTalk的影响等，有一定差距。

pt/ista ratio分布	value
mean	1.11
variance	0.00095
median	1.107
maximum	1.5404
minimum	0.9035

研究进展一：基于ML方法校准Post-Routing延时

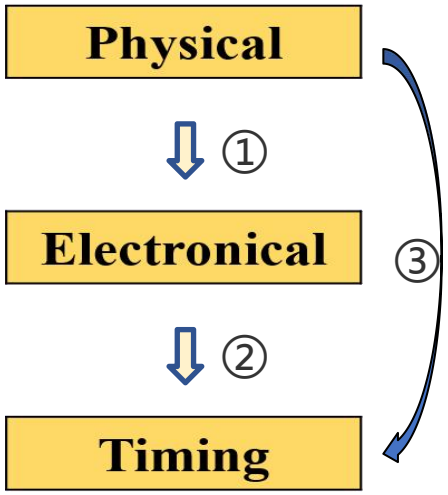
- 基于机器学习ML的方法，在Elmore，D2M时延计算的基础上预测Golden(如PT做标签)结果，校准低阶方法，可以应用在关键路径上；
- 98%的timing path 的delay和PT的误差在5%以内

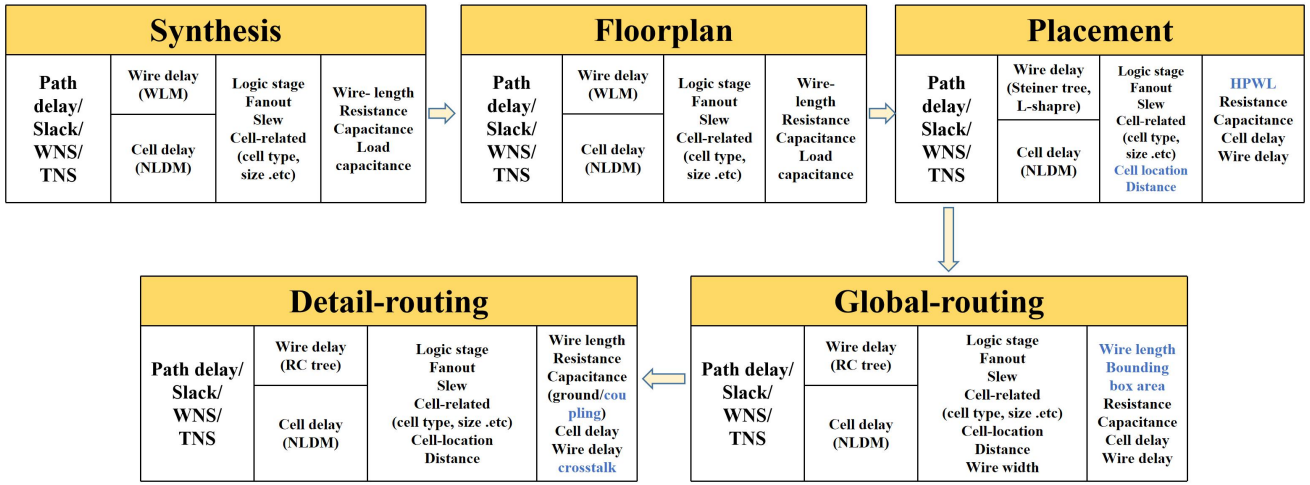


He Liu etc. ISEDA《*Accurate Timing Path Delay Learning using FeatureEnhancer with Effective Capacitance*》

研究进展二：基于ML方法从物理变量预测时序

- 研究在物理设计中基于物理特征（物理位置、单元类型、阈值电压、线网扇入扇出、绕线所在层、线宽，线距等）预测时序变化趋势。



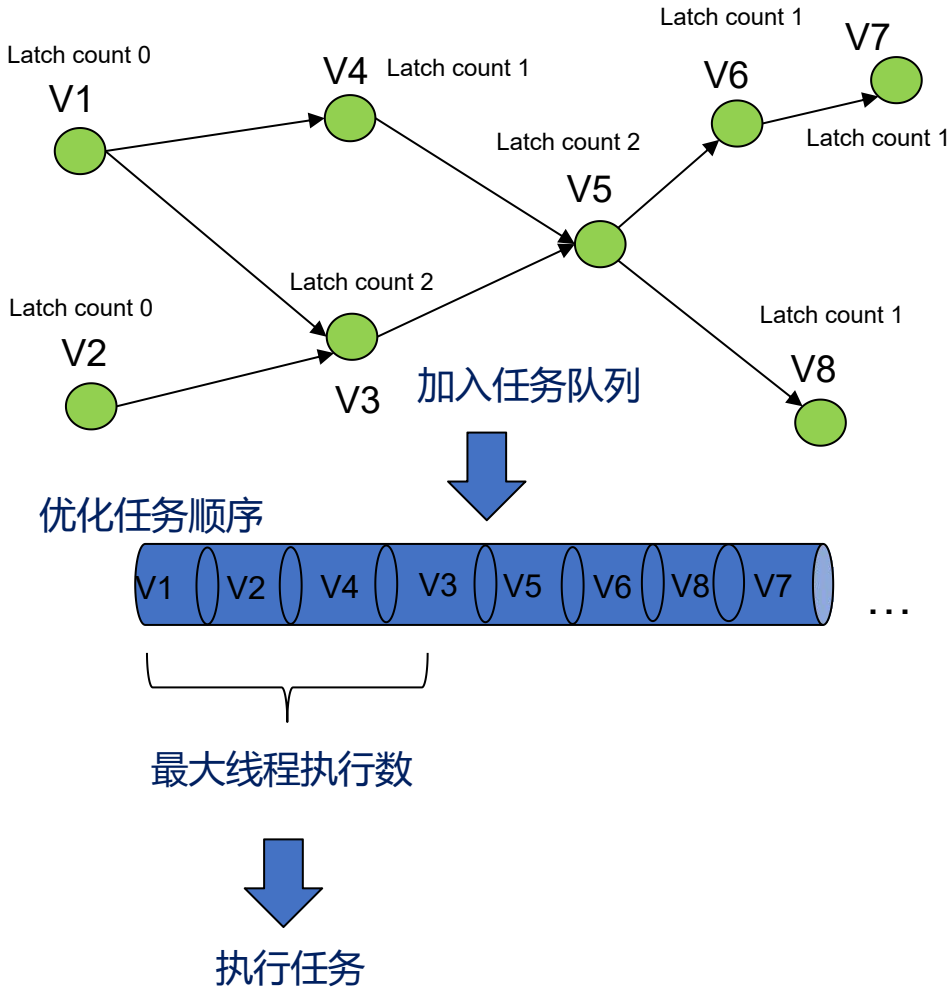


研究进展三：基于LTD算法加速图遍历

- **使用基于C++20特性的std::latch的高度并行执行遍历任务**

LTD算法（Latch based on ordered Task thread pool DFS algorithm）

- 分配任务。提前初始化了一个容量为图DAG中所有节点数量的latch pool，在分配完latch后，把节点任务按照一定顺序放入任务线程池中，保证本节点的前驱节点先放入队列。
- 优化顺序。通过重排序让存在依赖关系的任务具有相对合理的间隔时间，让后继任务执行时可以不用等待。
- 执行任务。latch为0的起始节点的线程第一个先执行。当一个节点往后传播时，将会对遍历到的后继点（snk vertex）的latch减1，latch减为0时，该节点对应的线程任务开始执行。当任务队列都执行完毕时结束任务。



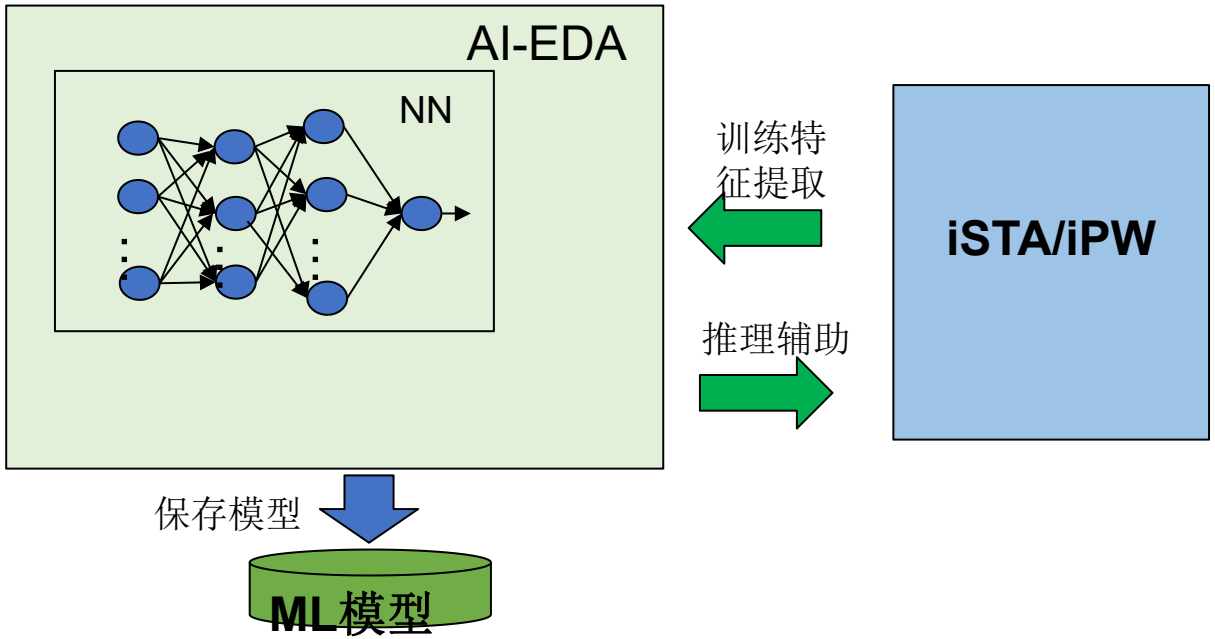
01 研究内容

02 研究进展

03 未来计划

工具计划一：架构改进，支持AI交互

- **改进工具架构设计**
  - 工具使用PYBIND/Cython接入Python->Pytorch->AI/ML模型；
  - 工具接入ML预测模型，通过模型校准关键路径时序和关键功耗分析结果。



工具计划二：基于需求驱动迭代工具

- **一生一芯将使用iEDA工具链，其中Yosys+iSTA辅助前端设计**
  - iSTA将支持WLM(wire load model)、时序ML预测校准结果；
  - iPW将支持基础IRDrop分析，ML预测IRDrop等。

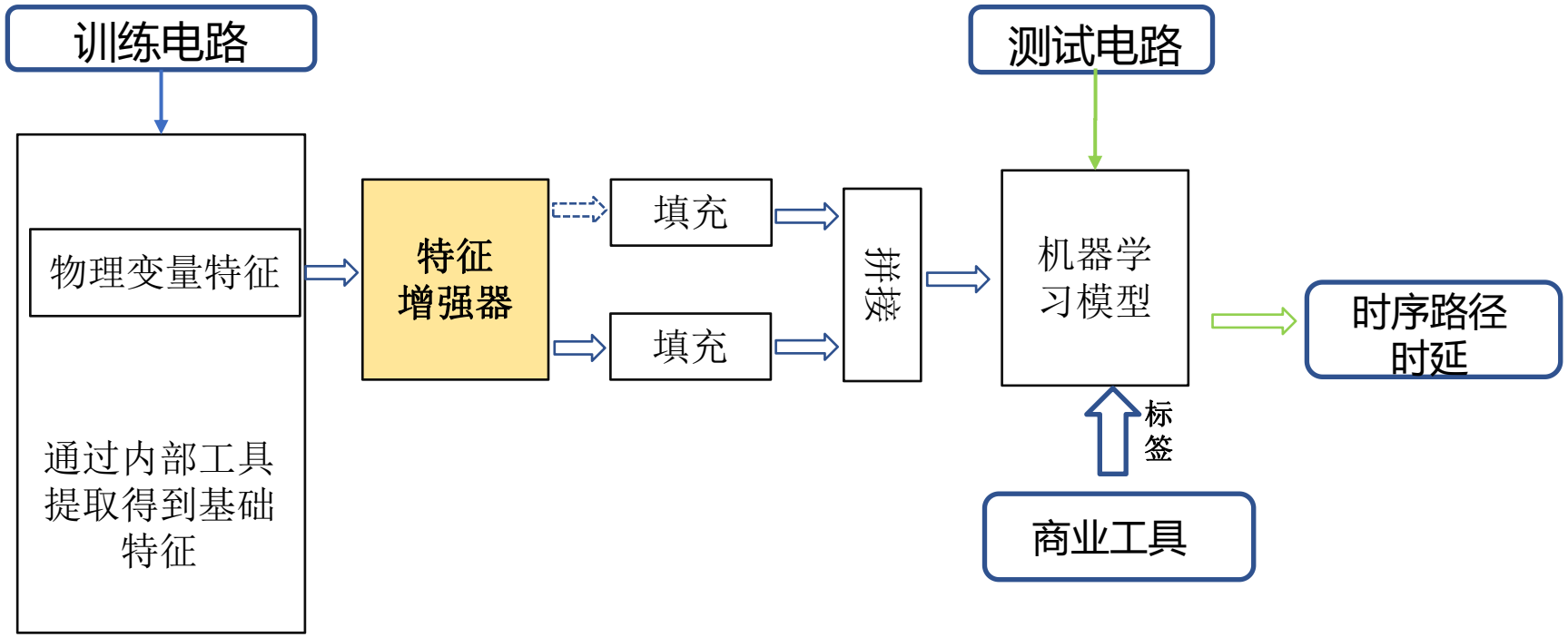
第三季度工具研发计划

工具特性	工具	支持时间
WLM	iSTA	8月31日
时序ML校准模型	iSTA	8月31日
IRDrop分析	iPW	8月31日
ML预测IRDrop	iPW	8月31日

研究计划一：ML在时序中的应用

- **ML预测物理变量的变化导致的时序变化趋势**

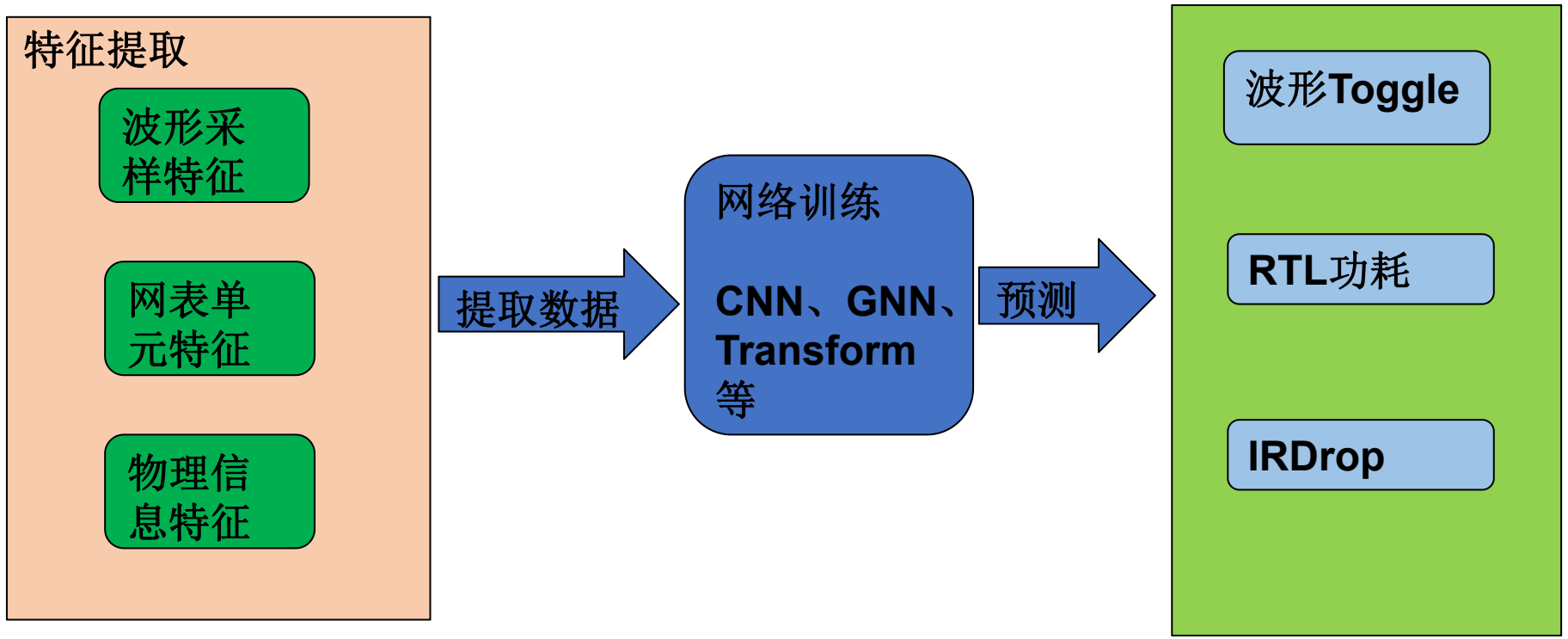
通过使用网络模型（如GNN，Transformer或者结合），研究物理变量中的变化因素导致时序的变化趋势。



研究计划二：ML在功耗中的应用

- **ML在RTL功耗、IRDrop预测中的研究**

通过使用CNN网络或者GNN网络，预测数据翻转率Toggle、RTL功耗、IRDrop。



总结

- **iSTA静态时序分析**
  - 规模大，性能是关键因素；
  - 准确度决定了时序分析的可用性，涉及到微分方程快速精确求解，插值方法精度，Crosstalk分析等。
- **iPW功耗分析**
  - Toggle/SP传播的方式快速但不精确，反标VCD会很精确，但波形文件会比较大，运行时间成为瓶颈，性能同样是关键因素；
  - 设计是决定功耗大小的关键因素，因此在RTL阶段能够准确预测功耗具有比较大的意义。

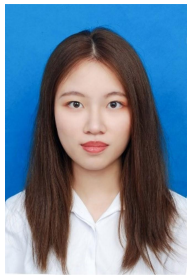
iEDA Tutorial 第一期议程

- Part1 iEDA-iSTA和iPW整体介绍 40min （陶思敏）

- Part2 iSTA工具架构、特性、API与使用 25min（龙帅英）

- Part3 iSTA关键技术研究 30min（刘贺）

- Part4 iPW工具架构、特性、关键技术与使用 25min（邵哲青）



iSTA总览

- **iSTA是一款开源静态时序分析工具，旨在实现易使用、高效并行、全流程覆盖的目标，用以支撑28nm流片需求**
  - 易使用：

兼容商业工具使用方式，

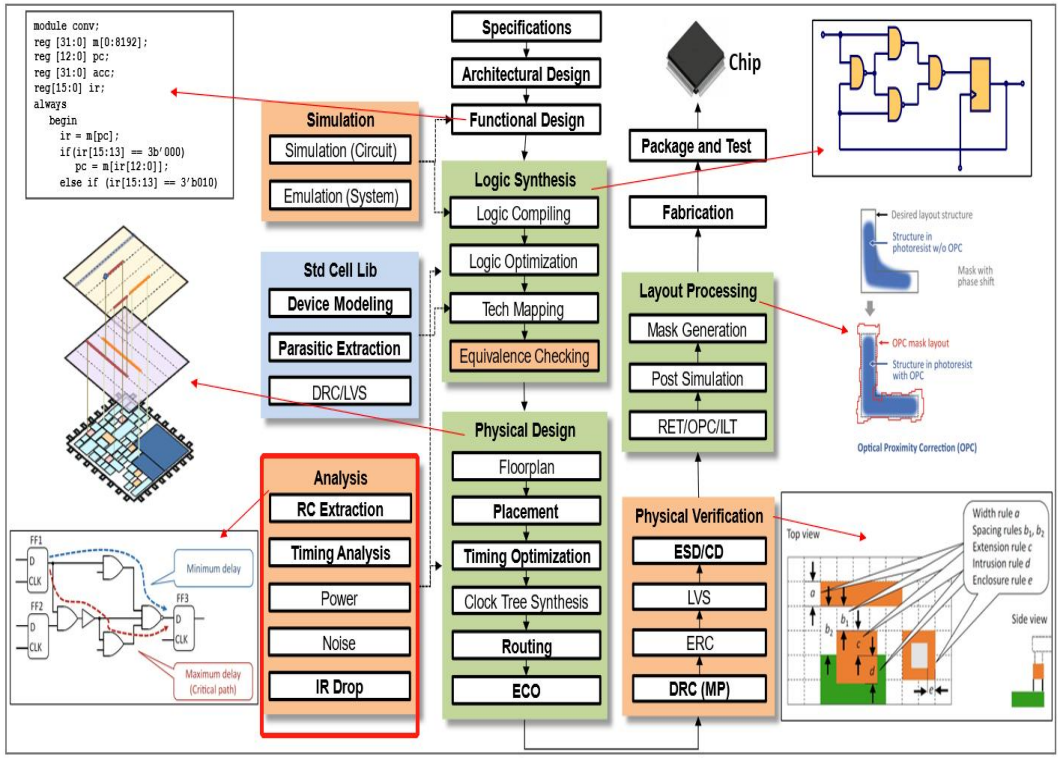
可以通过TCL脚本调用

  - 高效并行：

通过多线程能够充分并行地执行流程和算法

  - 全流程覆盖：

在物理设计各个流程和签核阶段都可以调用



iSTA支持功能

- 完善地支持标准输入文件（Def/Verilog，sdc，spef/sdf, liberty）读取
- 延时计算除了支持NLDM/Elmore计算模型，还支持CCS电流模型，Arnoldi降阶模型
- 时序分析支持Clock Gate分析，Removal/Recovery分析和Muliticycle分析
- 时序路径分析模式支持OCV模式和AOCV模式
- 噪声分析初步支持了Crosstalk的影响，未来将进一步完善
- 提供时序分析引擎timing engine供物理设计调用

iSTA总体架构

- STA-01 时序图构建
- STA-02 延时计算
- STA-03 路径搜索
- STA-04 时序分析和报告

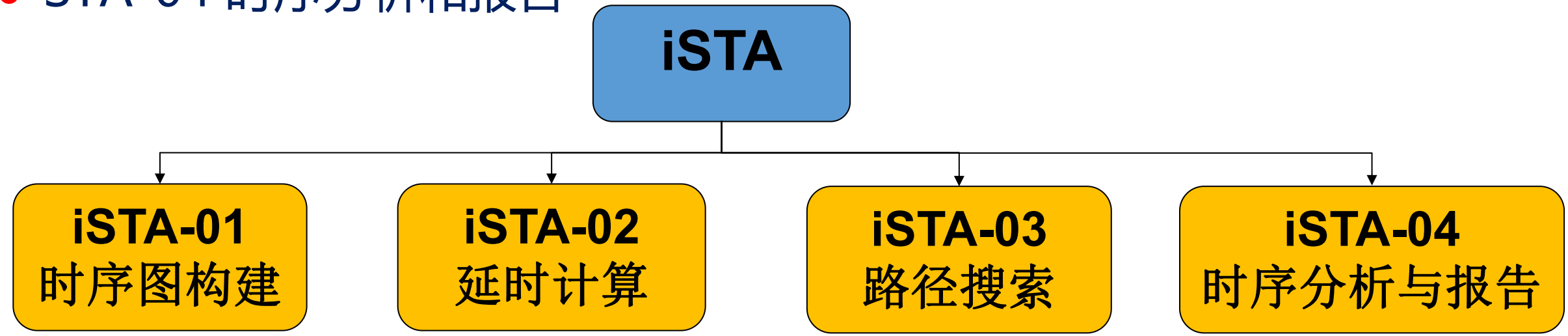


图1 总体架构图

iSTA总体架构—时序图构建

- 时序图构建：载入时序网表，并从网表中抽象出电路图，并应用时序约束到图当中。

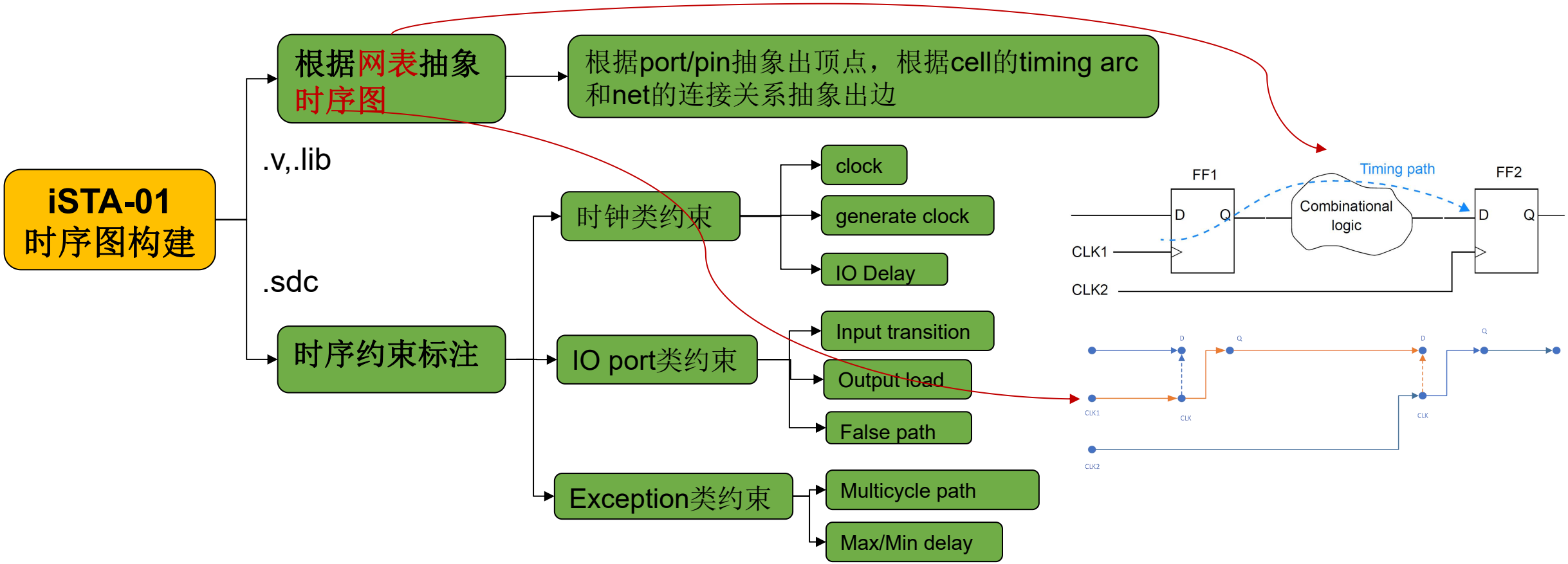


图2 时序图构建

iSTA总体架构—延时计算

- 延时计算： 从Liberty和SPEF中读取单元和互连线数据，计算单元和互连线Delay和Slew。

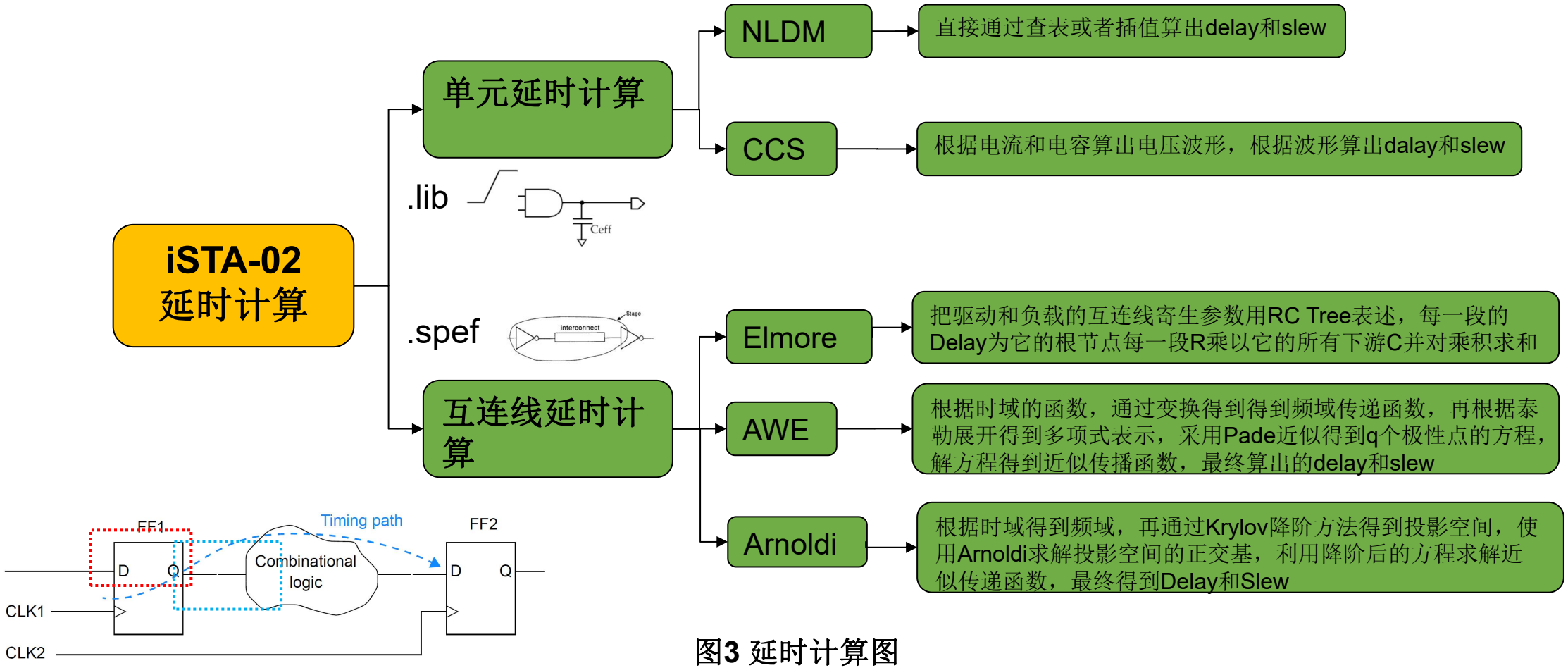


图3 延时计算图

iSTA总体架构—路径搜索

- 路径搜索：从电路图中搜索出需要分析的时序路径并记录时序路径。

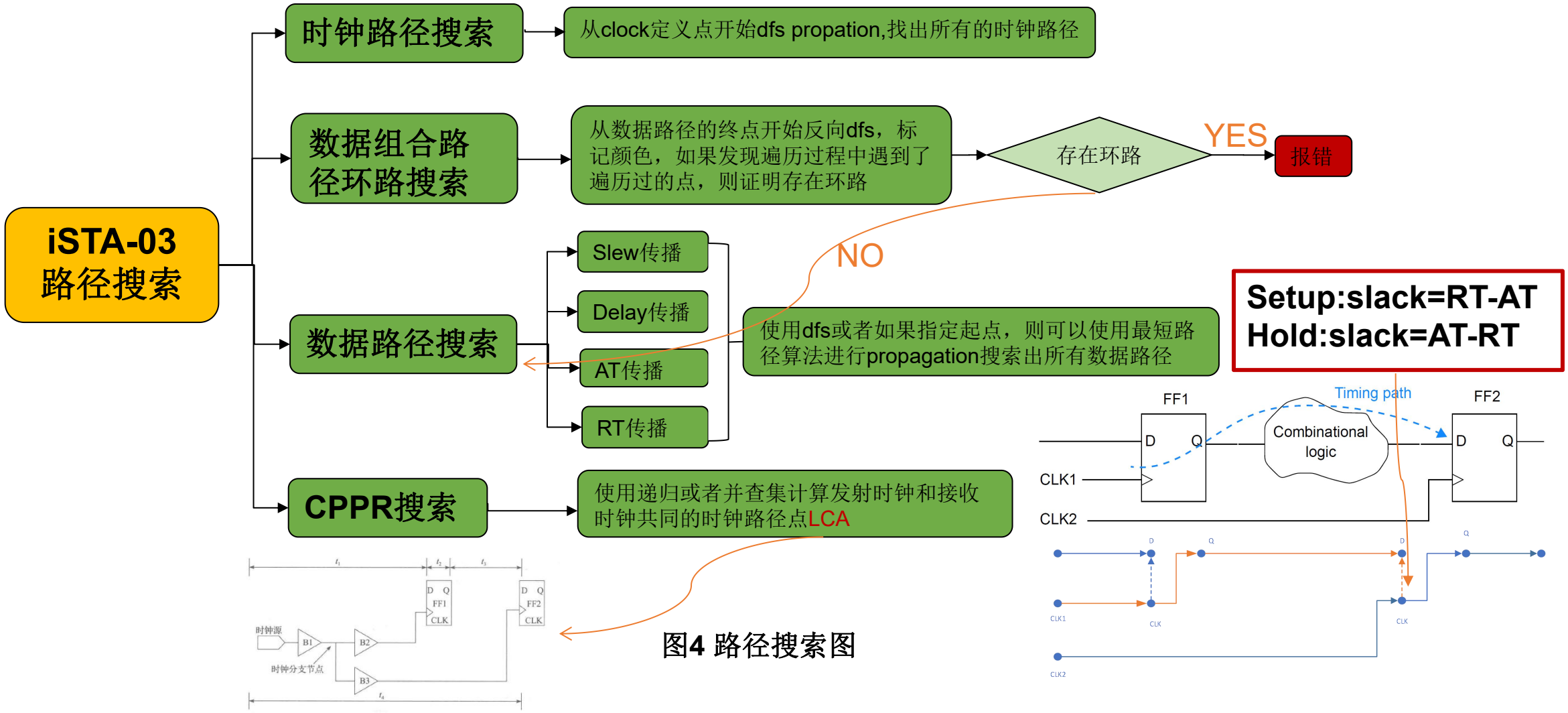


图4 路径搜索图

iSTA总体架构—时序分析和报告

- 时序分析和报告： 对时序路径进行slack计算并以文本形式输出报告

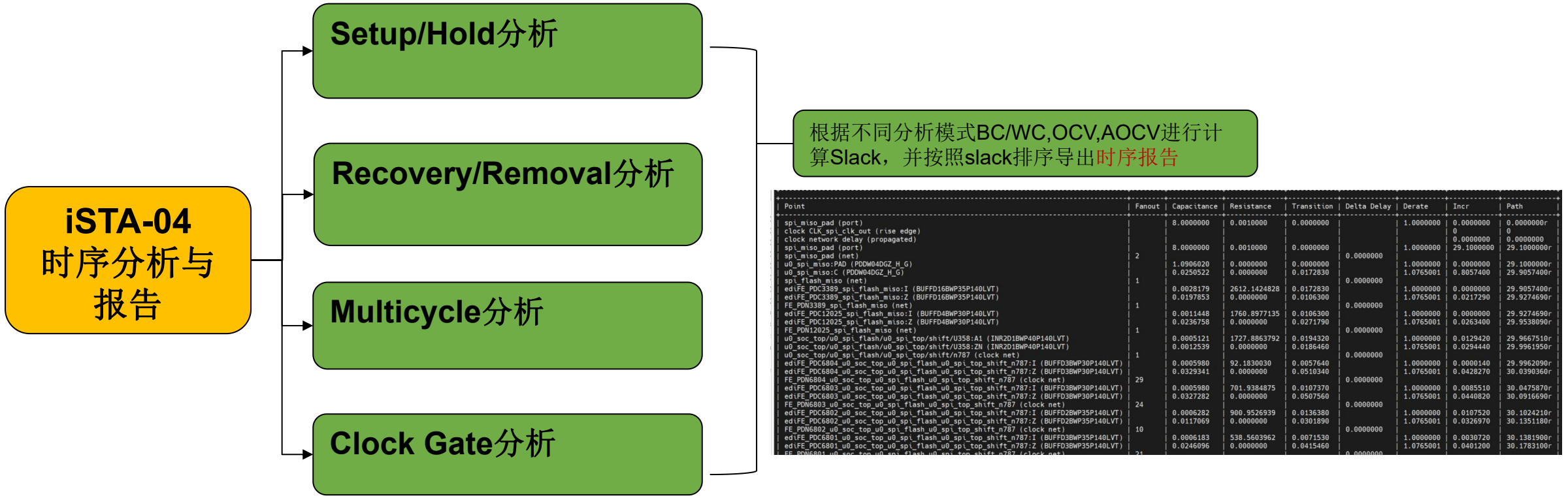
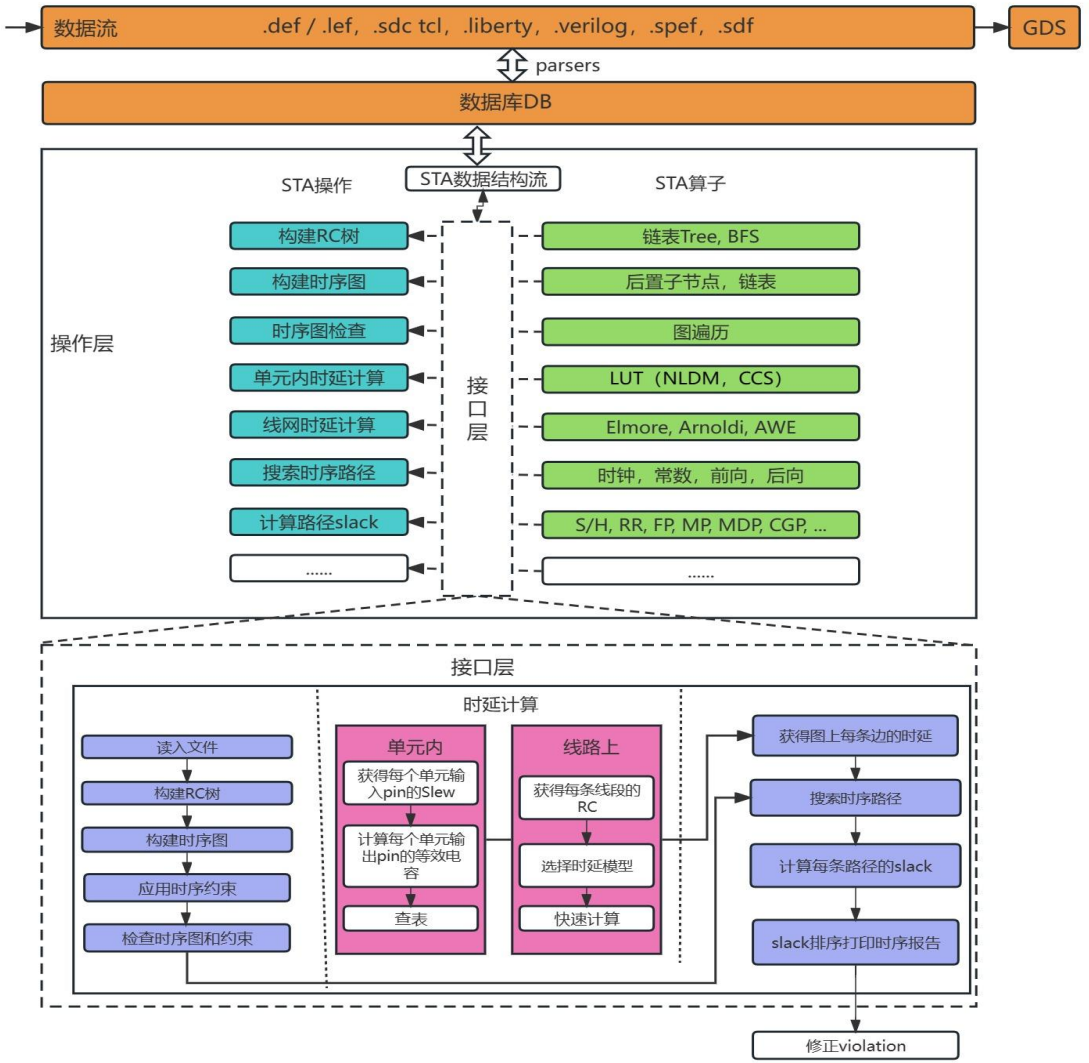


图5 时序分析和报告图

iSTA软件流程

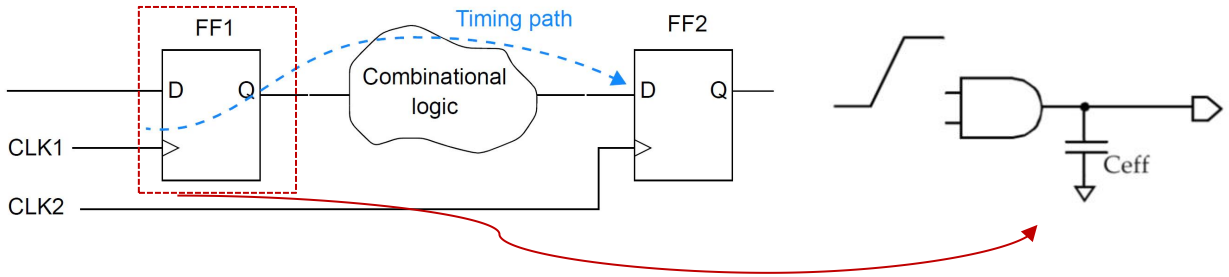
- **数据库DB**
  - def、.v、.sdc、.lib、spef文件parser到自定义数据结构
- **操作层**
  - 构建RC树、时序图、延时计算、搜索路径的操作
- **STA算子**
  - 操作层所使用的具体方法
- **接口层**
  - 通过接口层将STA操作和STA算子串通整个时序分析流程



支持特性一：NLDM&Elmore延时计算

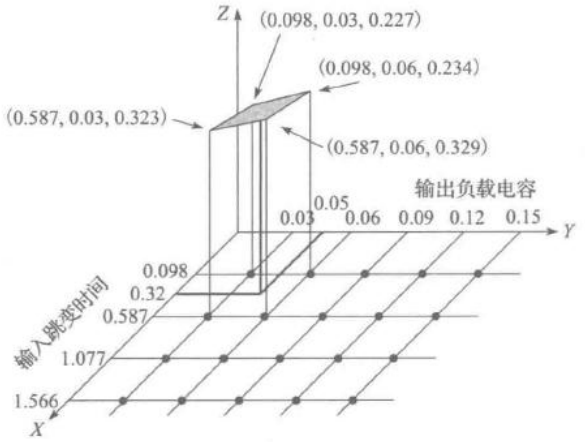
- NLDM(Non-Linear Delay Model) ——cell延时模型

>通过查表的方式获得cell的单元延时，是个二维表，表的维度一般是input transition和output load。




```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7");  /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0513,  0.1537,  0.5280", \
        /* 0.3 */   "0.1018,  0.2327,  0.6476", \
        /* 0.7 */   "0.1334,  0.2973,  0.7252");
      }
    cell_fall(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0617,  0.1537,   0.5280", \
        /* 0.3 */   "0.0918,  0.2027,   0.5676", \
        /* 0.7 */   "0.1034,  0.2273,   0.6452");
    }
```

>对于不能通过input transition和output load直接在NLDM表获取deley值的需要进行插值，获取其值，下面是线性插值的方法。

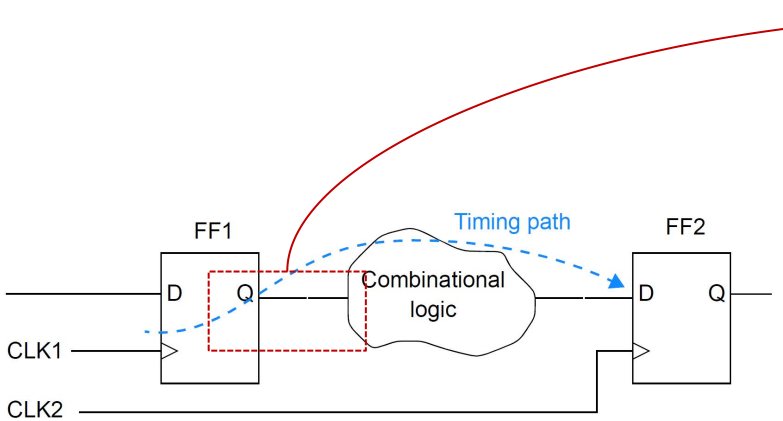


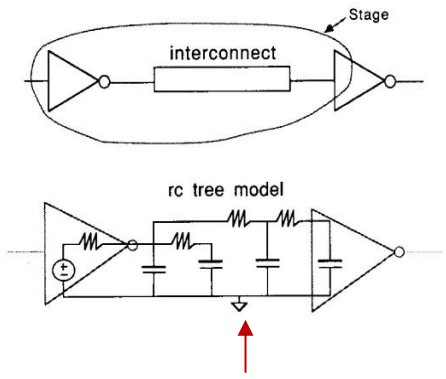
$$Z = A + B \cdot X + C \cdot Y + D \cdot X \cdot Y$$

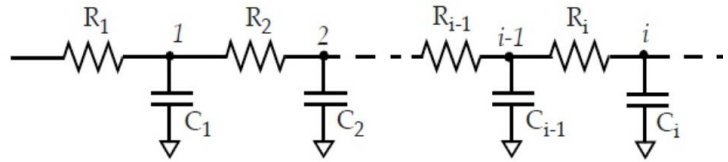
支持特性一：NLDM&Elmore延时计算

- Elmore——互连线延时模型

  >互连线微分方程的一阶矩，可以快速近似得到延时结果，且能证明其是延时的上界。







$$T_{d2} = C_1 * R_1$$

$$T_{d2} = C_1 * R_1 + C_2 * (R_1 + R_2)$$

...

$$T_{dn} = \sum_{i=1}^{n} C_i \left(\sum_{j=1}^{i} R_j \right)$$

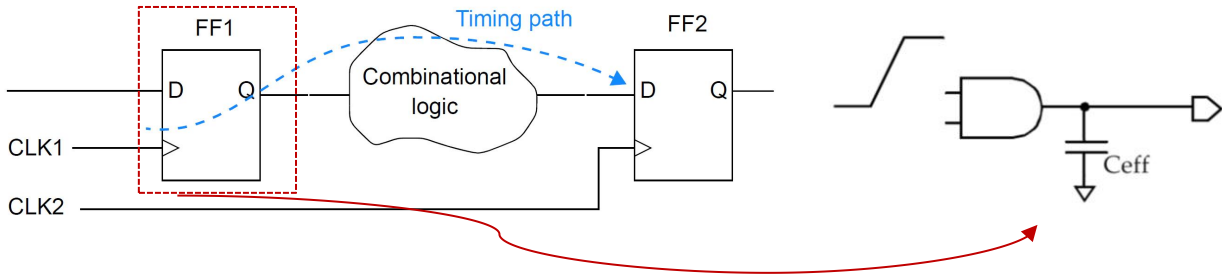
.spef

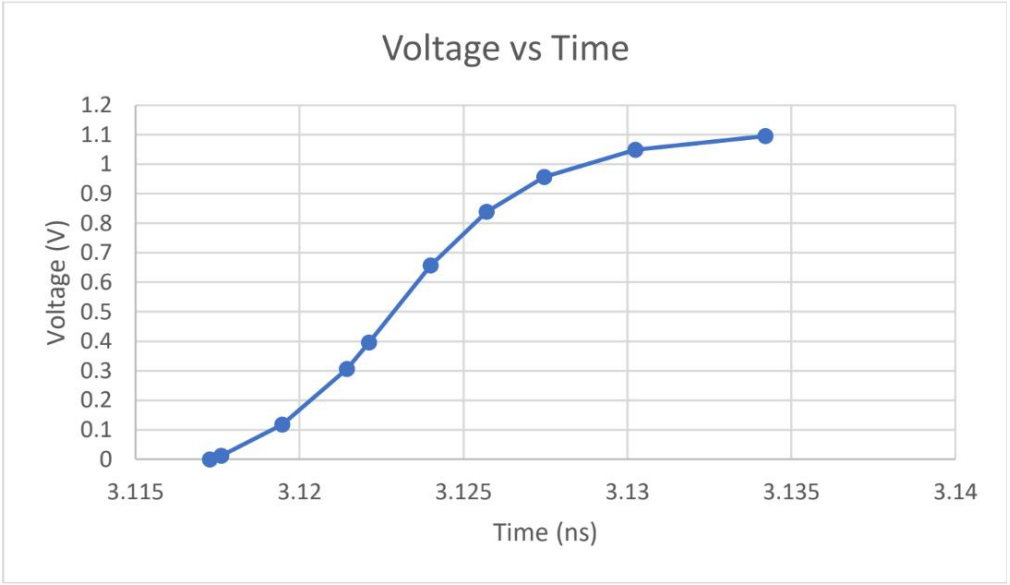
```
*D_NET *5423 2.69358

*CONN
*I *14207:D I *C 21.7450 94.3150
*I *14205:D I *C 21.7450 90.4900
*I *14211:Q O *C 21.4900 83.8800 *D DFFQX1

*CAP
1 *5423:10107 *547:12722 0.202686
2 *5423:10107 *5116:10594 0.104195
3 *5423:10107 *5233:9552 0.208867
4 *5423:10107 *5265:9483 0.0225810
5 *5423:10107 *267:9668 0.0443454
6 *5423:10107 *5314:7853 0.120589
7 *5423:10212 *2109:996 0.0293744
8 *5423:10212 *5187:7411 0.526945
9 *5423:14640 *6577:10075 0.126929
10 *5423:10213 1.30707

*RES
1 *5423:10107 *5423:10212 2.07195
2 *5423:10107 *5423:10106 0.340000
3 *5423:10212 *5423:10211 0.340000
4 *5423:10212 *5423:14640 1.17257
5 *5423:14640 *5423:10213 0.340000
6 *5423:10213 *14207:D 0.0806953
7 *5423:10211 *14205:D 0.210835
8 *5423:10106 *14211:Q 0.0932139
*END
```

支持特性二：CCS&Arnolidi延时计算

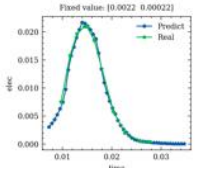
- CCS（Composite Current Source）——更精确的cell延时模型

>通过把电流转成对应的电压值，可以获取到待求解cell的slew和Delay。






```
pin (OUT) {
  . . .
  timing () {
    related_pin : "IN"-;
    . . .
    output_current_fall () {
      vector ("LOOKUP_TABLE_1x1x5") {
        reference_time : 5.06; /* Time of input crossing
          threshold */
        index_1("0.040"); /* Input transition */
        index_2("0.900"); /* Output capacitance */
        index_3("5.079e+00, 5.093e+00, 5.152e+00,
                 5.170e+00, 5.352e+00");/* Time values */
        /* Output charging current: */
        values("-5.784e-02, -5.980e-02, -5.417e-02,
                -4.257e-02, -2.184e-03");
      }
      . . .
    }
    . . .
  }
  . . .
}
```

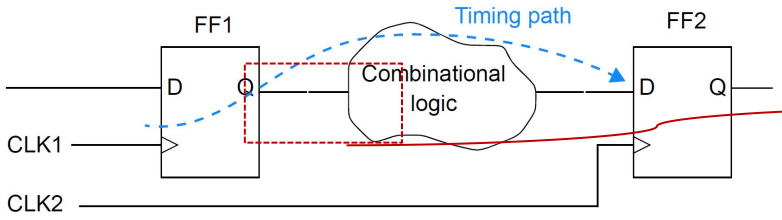


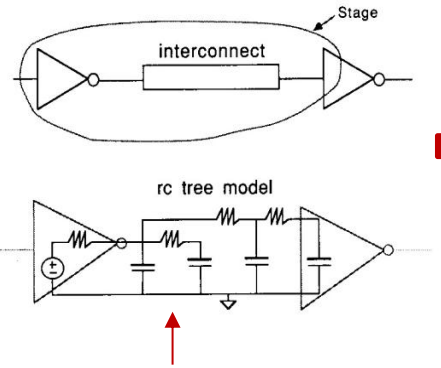
$$i(t) = C(t)\frac{dV(t)}{dt}$$

支持特性二：CCS&Arnoldi延时计算

- Arnoldi——更精确的互连线延时模型

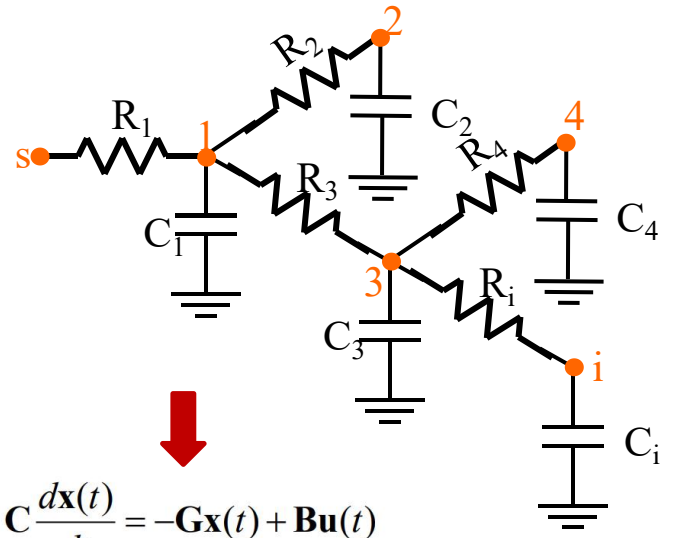
  >原电路方程，通过构建Arnoldi正交基，然后对原方程做降阶，可以缩小方程的矩阵规模，进而再继续求解缩小规模后的微分方程，可以快速得到延时结果。





.spef

```
*D_NET *5423 2.69358

*CONN
*I *14207:D I *C 21.7450 94.3150
*I *14205:D I *C 21.7450 90.4900
*I *14211:Q O *C 21.4900 83.8800 *D DFFQX1

*CAP
1 *5423:10107 *547:12722 0.202686
2 *5423:10107 *5116:10594 0.104195
3 *5423:10107 *5233:9552 0.208867
4 *5423:10107 *5265:9483 0.0225810
5 *5423:10107 *267:9668 0.0443454
6 *5423:10107 *5314:7853 0.120589
7 *5423:10212 *2109:996 0.0293744
8 *5423:10212 *5187:7411 0.526945
9 *5423:14640 *6577:10075 0.126929
10 *5423:10213 1.30707

*RES
1 *5423:10107 *5423:10212 2.07195
2 *5423:10107 *5423:10106 0.340000
3 *5423:10212 *5423:10211 0.340000
4 *5423:10212 *5423:14640 1.17257
5 *5423:14640 *5423:10213 0.340000
6 *5423:10213 *14207:D 0.0806953
7 *5423:10211 *14205:D 0.210835
8 *5423:10106 *14211:Q 0.0932139
*END
```



$$C\frac{dx(t)}{dt} = -Gx(t) + Bu(t)$$

$$y(t) = L^T x(t)$$

$$A = -G^{-1}C, b = G^{-1}B$$

$$v_1 = \frac{b}{\| b \|};$$

$$v_2 = Av_1 - (Av_1)^T v_1 v_1, v_2 = \frac{v_2}{\| v_2 \|};$$

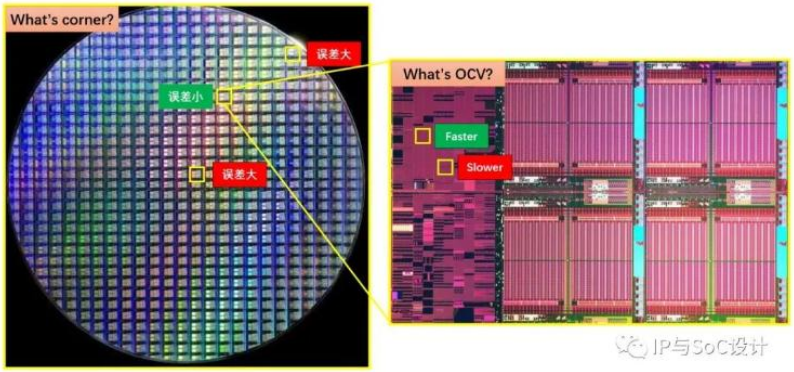
C:电容
G:电导
x:电压(待求解值)
u:电流
B:输入矩阵
[1,0,0,…] 向量化

支持特性三：OCV

- OCV(On Chip Variation，片上偏差)

On Chip Variations，反应芯片die上不同位置的延时波动。主要的影响因素（PVT）包括：

- 生产工艺的波动；（Process）
- 信号翻转导致的电压波动，影响电源的供电；（Voltage）
- 不同位置温度不一样。（Temperature）



- 具体实现

用户通过set_timing_derate命令设置derate值，iSTA在计算时钟路径，数据路径时将会apply对应的derate到路径延时值上面。

```
set_timing_derate -net_delay -early 0.915
set_timing_derate -net_delay -late 1.085
set_timing_derate -cell_delay -clock -early 0.8845
set_timing_derate -cell_delay -clock -late 1.0395
set_timing_derate -cell_delay -data -early 0.8195
set_timing_derate -cell_delay -data -late 1.0765
```

支持特性四：AOCV

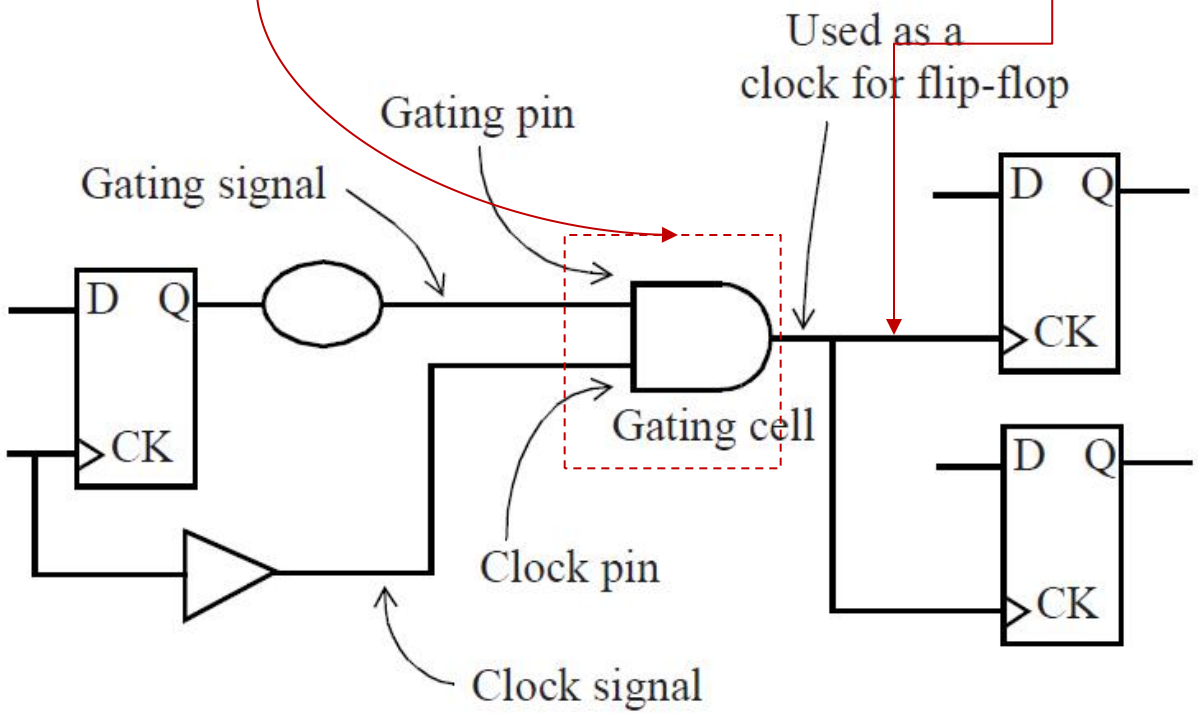
- **需求：**随着制造工艺越来越先进，在时序分析规模不断增大同时，对时序分析精度的要求也越来越高。AOCV（Advanced On Chip Variation)分析模式可以通过更详细的模拟仿真结果来最大程度地接近真实情况下片上变化影响的随机性，通过动态地调整时序减免值来达到更加接近真实状态，从而减少悲观结果带来的时序收敛难度。
- **AOCV分析模式查找表：**反映了片上偏差（时序减免值）与时序路径的逻辑深度和物理距离的关系。

```
object_type: lib_cell
rf_type: rise fall
delay_type: cell
derate_type: late
path_type: clock
object_spec: -quiet tcbn28hpcplusbwp40p140lvtssg0p81v125c_ccs/*
depth: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80
distance:
table: 1.225085 1.196231 1.167520 1.153743 1.137582 1.123300 1.109967 1.104879 1.101338 1.095725 1.092799 1.089171 1.086665 1.083740 1.081598 1.079181 1.077337 1.075298 1.073695 1.071947 1.070539 1.069021 1.067773 1.066438 1.065323 1.064137 1
.063134 1.062073 1.061164 1.060207 1.059379 1.058510 1.057752 1.056958 1.056261 1.055532 1.054888 1.054216 1.053619 1.052997 1.052441 1.051863 1.051344 1.050805 1.050319 1.049815 1.049358 1.048885 1.048456 1.048011 1.047606 1.047187 1.046804
1.046408 1.046045 1.045670 1.045326 1.044971 1.044643 1.044306 1.043994 1.043672 1.043375 1.043069 1.042785 1.042492 1.042221 1.041941 1.041681 1.041413 1.041164 1.040908 1.040668 1.040422 1.040192 1.039956 1.039735 1.039508 1.039295 1.039076
```

- **具体实现：**获取cell在时序路径的逻辑深度，在aocv查找表通过逻辑深度获取时序减免值，apply 时序减免值到arc_delay上。

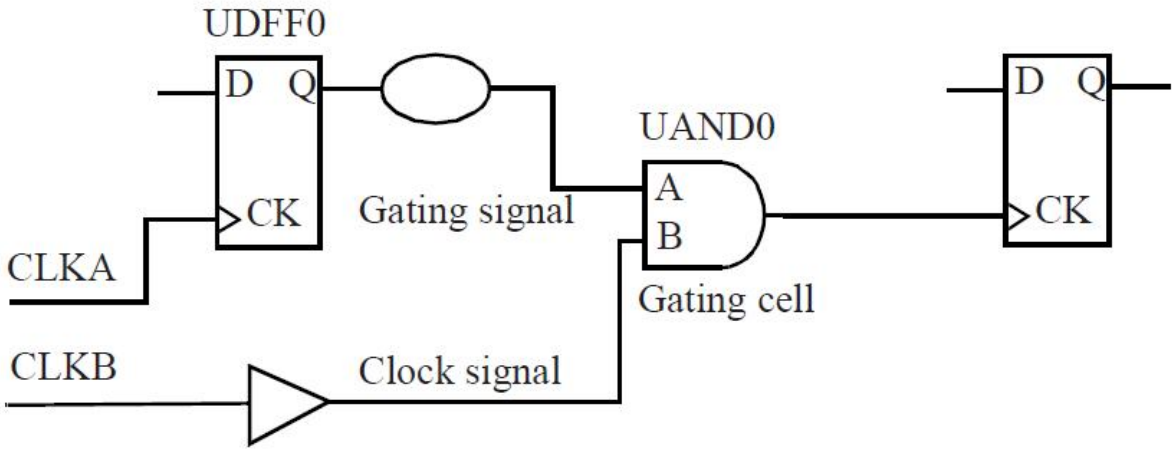
支持特性五：Clock Gate Checks

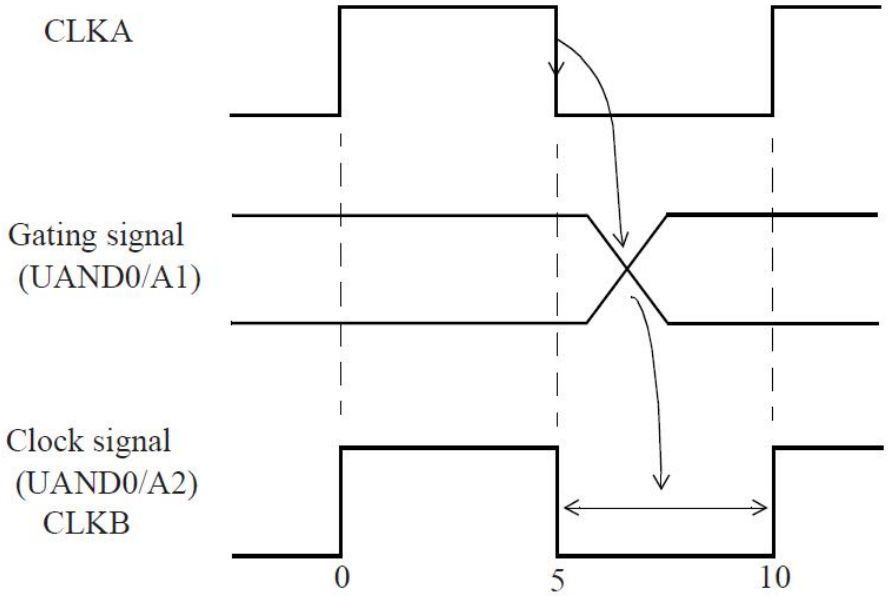
- 时钟门控单元（Clock Gate Cell）由于其减少了flip-flop的clock pin的翻转行为，能够有效地降低功耗，而被用在电路中，而时钟门控单元正常工作需要对其进行Clock Gate Checks。



支持特性五：Clock Gate Checks

- Clock Gate Checks要满足的**时序约束**

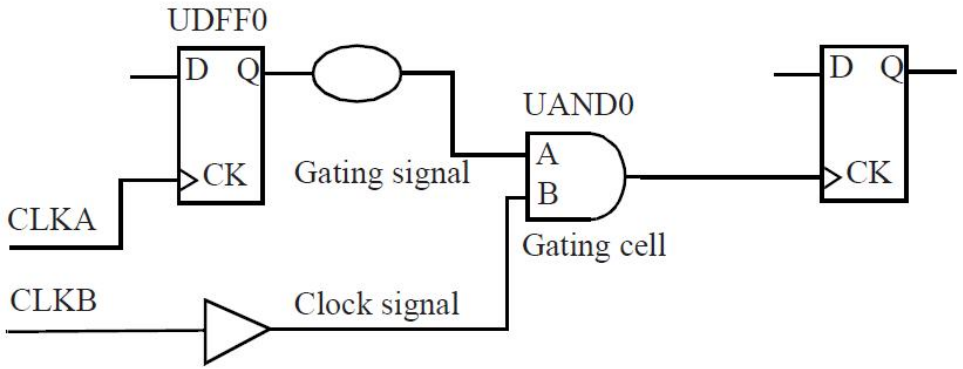


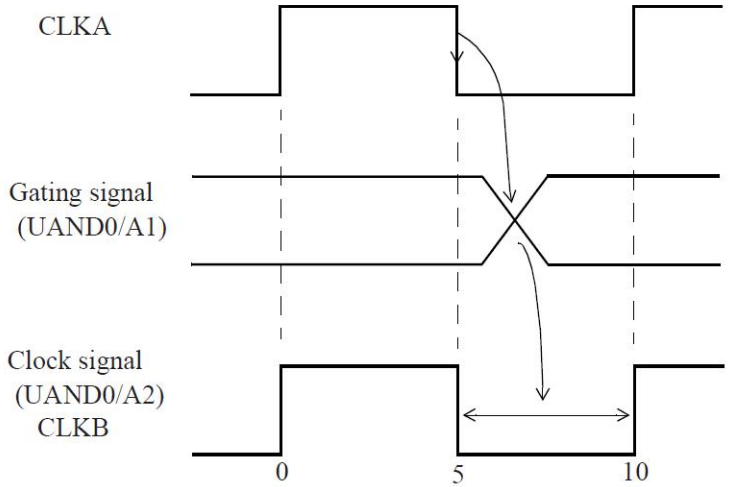


- Setup checks：requires that the gating signal changes before the clock goes high.
- Hold checks：requires that the gating signal changes only after the falling edge of the clock.

支持特性五：Clock Gate Checks

- **具体实现**





>Liberty Parser支持clock gating cell相关属性;

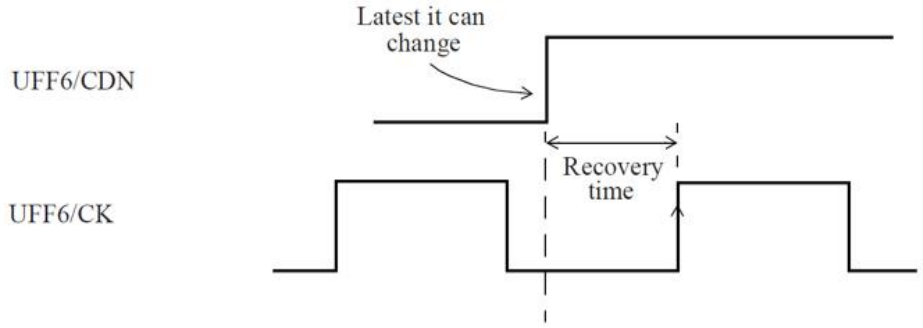
>Build Graph标记出clock gating cell的clock_gate_enable_pin到clock_gate_clock_pin的arc需要做Clock Gate Checks;

>获取到达gating pin的arrival time和clock pin的required time，检查是否满足Clock Gate Checks的时序约束。

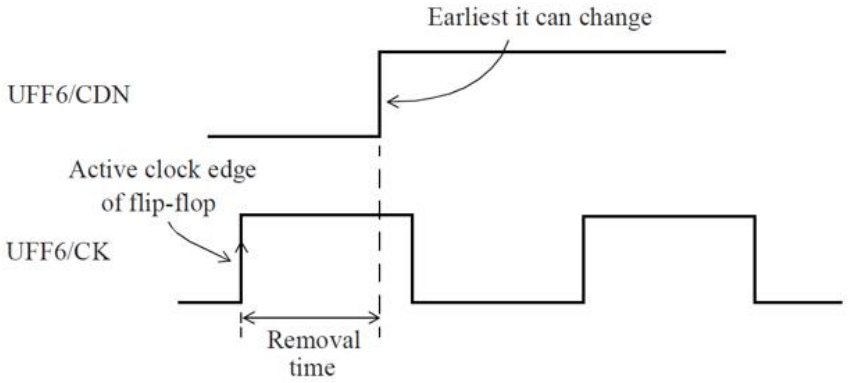
支持特性六：Recovery&Removal分析

- Recovery/Removal分析是指时序单元的复位/置位信号与时钟信号的时序约束。

>Recovery分析保证异步复位释放信号和下一工作时钟沿之间有足够的时间能让FF下一个周期进入正常工作状态，类似于Setup分析



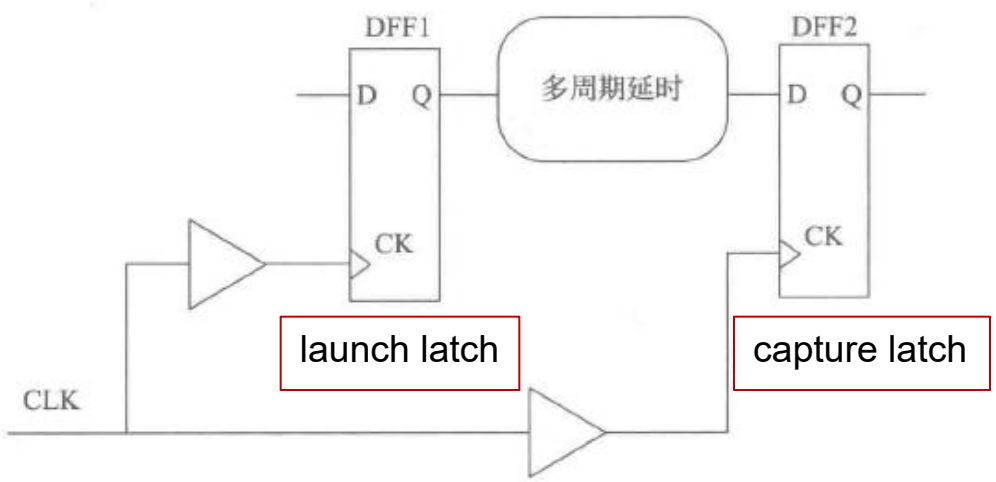
>Removal分析针对复位信号，当复位信号撤离时，时钟工作的边沿和复位释放的时间需要满足removal要求才能保证复位信号正常工作，类似于hold分析

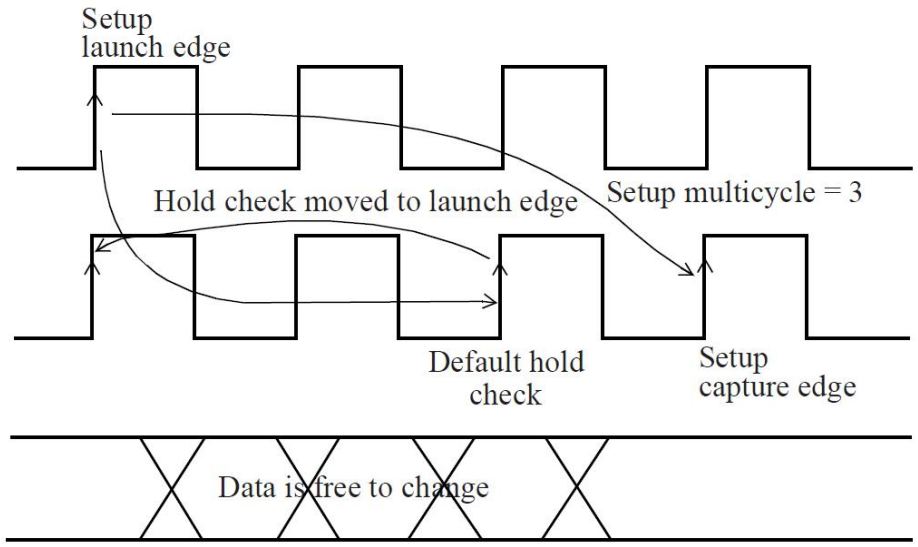


支持特性七： Multicycle分析

- 正常时序分析都是launch latch和capture latch相差一个时钟周期，而有的时序路径比较长，因此用户在设计时会设置launch latch和capture latch的setup/hold checks在多个时钟周期内完成。

- 通过命令set_multicycle_path可以设置多周期分析。





iSTA：面向开发者的API—读文件

- 基于iEDA物理设计工具的开发过程中的需求，iSTA提供了一系列API。

Method	Type	Argument	Return	Description
readLiberty	builder	file	self	read the liberty files.
readDesign	builder	file	self	read the design verilog file.
readSpef	builder	file	self	read the spef file.
readSdc	builder	file	self	read the sdc file.
readAocv	builder	file	self	read the aocv files.

iSTA：面向开发者的API——构建RCTree

Method	Type	Argument	Return	Description
makeOrFindRCTreeNode	builder	net, id	RctNode	make RC tree internal node.
makeOrFindRCTreeNode	builder	pin_or_port	RctNode	make RC tree pin node.
incrCap	builder	node, cap , is_incremental	void	set the node’s cap
makeResistor	builder	net, from_node, to_node,res	void	make resistor edge of rc tree
updateRCTreeInfo	builder	net	void	update rc info after make rc tree
buildRCTree	builder	spef_file, kmethod	self	build RC tree according to the spef file
initRcTree	builder	net	void	init one rc tree
resetRcTree	builder	net	void	reset rc tree to nullptr

iSTA：面向开发者的API—操作StaGraph

Method	Type	Argument	Return	Description
buildGraph	builder	void	unsigned	build the graph data
isBuildGraph	builder	void	unsigned	judge whether it has build graph
resetGraph	builder	void	self	reset graph
resetGraphData	builder	void	self	reset graph data

iSTA：面向开发者的API—修改网表

Method	Type	Argument	Return	Description
insertBuffer	builder	instance_name	void	insert buffer need to change the netlist.
removeBuffer	builder	instance_name	void	remove buffer need to change the netlist
repowerInstance	builder	instance_name, cell_name	void	change the size or level of an existing instance
moveInstance	builder	instance_name , update_level , prop_type	void	move the instance to a new location
writeVerilog	builder	void	file	write verilog file according to the netlist

iSTA：面向开发者的API—时序传播及获取

Method	Type	Argument	Return	Description
incrUpdateTiming	action	void	self	incremental propagation to update timing data
updateTiming	action	void	self	update timing data
setSignificantDigits	action	significant_digits	self	Set the significant digits of timing report
reportTiming	action	exclude_cell_names , is_derate , is_clock_cap	self	generate the timing report.

iSTA：面向开发者的API—报告时序结果

Method	Type	Argument	Return	Description
reportSlew	accessor	pin_name	double	report the slew of the pin.
reportAT	accessor	pin_name	std::optional <double>	report the arrival time at a pin
reportRT	accessor	pin_name	std::optional <double>	report the required arrival time at a pin
reportSlack	accessor	pin_name	std::optional <double>	report the slack at a pin
reportWNS	accessor	clock_name	double	report the total negative slack of the clock group path
reportTNS	accessor	clock_name	double	report the worst negative slack of the clock group path
reportClockSkew	accessor	src_clock_pin_name, snk_clock_pin_name	double	report the skew between two clocks

iSTA：面向开发者的API—报告时序结果

Method	Type	Argument	Return	Description
reportInstDelay	accessor	inst_name	double	report instance delay
reportInstWorstArcDelay	accessor	inst_name	double	report the worst arc delay for the specified instance
reportNetDelay	accessor	net_name	double	report net delay
…	…	…	…	…

iSTA：面向开发者的API—check cap/slew/fanout

Method	Type	Argument	Return	Description
checkCapacitance	accessor	pin_name	void	check the real pin cap and the limit pin cap in liberty, calculate the cap slack.
checkFanout	accessor	pin_name	void	check the real fanout nums and the limit fanout nums in liberty, calculate the fanout slack.
checkSlew	accessor	pin_name	void	check the real slew and the limit slew in liberty,calculate the slew slack.

iSTA：面向使用者的tcl命令—时序约束标注

- 支持的sdc命令

sdc-cmd	Description
AllClocks	Creates a collection of all clocks in the current design. You can assign these clocks to a variable or pass them into another command.
CreateClock	Creates a clock object.
GetClocks	Creates a collection of clocks from the current design. You can assign these clocks to a variable or pass them into another command.
GetPins	Creates a collection of pins from the netlist. You can assign these pins to a variable or pass them into another command.
GetPorts	Creates a collection of ports from the current design or instance. You can assign these ports to a variable or pass them into another command.
SetClockGroups	Specifies clock groups that are mutually exclusive or asynchronous with each other in a design so that the paths between these clocks are not considered during the timing analysis.
SetClockLatency	Specifies the latency of the clock network.
SetClockUncertainty	Specifies the uncertainty (skew) of specified clock networks.

iSTA：面向使用者的tcl命令

- 支持的sdc命令

sdc-cmd	Description
SetInputTransition	Sets a fixed transition time on input or inout ports.
SetInputDelay	Defines the arrival time relative to a clock.
SetLoad	Sets the capacitance on the specified ports and nets in the current design.
SetMaxCapacitance	Sets maximum fanout for input ports or designs.
SetMaxFanout	Sets maximum fanout for input ports or designs.
SetMaxTransition	Sets maximum transition for pins, ports, clocks, or designs with respect to the main library trip-points
SetMulticyclePath	Defines the multicycle path.
SetPropagatedClock	Specifies propagated clock latency.
CmdSetTimingDerate	Adjusts the calculated cell and net delays by a specified factor to model the effects of on-chip variation.

iSTA：面向使用者的tcl命令

- 编写run_ista.tcl文件，如右图所示
- 运行该文件，得到6个时序报告：

>asic_top.rpt（报告 WNS,TNS 和时序路径）

>asic_top.cap（报告违例电容）

>asic_top.fanout（报告违例扇出）

>asic_top.trans（报告违例转换时间）

>asic_top_hold .skew（报告hold模式下的时钟偏斜）

>asic_top_setup.skew（报告setup模式下的时钟偏斜）

```
set_design_workspace "/var/lib/jenkins/ysyx/"

read_verilog 1214/asic_top.v

set LIB_FILES { \
  1101/lib/S011HD1P1024X64M4B0_SS_1.08_125.lib  \
  1101/lib/S011HD1P128X21M2B0_SS_1.08_125.lib  \
  1101/lib/S011HD1P256X8M4B0_SS_1.08_125.lib \
  1101/lib/S011HD1P512X19M4B0_SS_1.08_125.lib \
  1101/lib/S011HD1P512X73M2B0_SS_1.08_125.lib \
  1101/lib/S011HDSP4096X64M8B0_SS_1.08_125.lib \
  1101/lib/S013PLLFN_v1.5.1_typ.lib \
  1101/lib/SP013D3WP_V1p7_typ.lib \
  1101/lib/SP013D3WP_V1p7_typ1.lib \
  1101/lib/scc011ums_hd_hvt_ss_v1p08_125c_ccs.lib \
  1101/lib/scc011ums_hd_lvt_ss_v1p08_125c_ccs.lib \
  1101/lib/scc011ums_hd_rvt_ss_v1p08_125c_ccs.lib \
}

foreach LIB_FILE $LIB_FILES { \
    read_liberty $LIB_FILE    \
}

link_design asic_top

read_sdc 1214/asic_top.sdc
read_spef 1214/asic_top.spef

report_timing
```

run_ista.tcl

iSTA：timing report解读

- asic_top.rpt（报告 WNS,TNS 和时序路径）

Generate the report at 2023-06-19T11:54:45

Endpoint	Clock Group	Delay Type	Path Delay	Path Required	CPPR	Slack	Freq(MHz)
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_34_:D	CLK_u0_clk_XC	max	30.7017020r	40.1711020	0.0000000	9.4694000	NA
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_7_:D	CLK_u0_clk_XC	max	30.7010500r	40.1708440	0.0000000	9.4697940	NA
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_114_:D	CLK_u0_clk_XC	max	30.6965330r	40.1707670	0.0000000	9.4742340	NA
u0_soc_top/u0_sdram_axi/u_core/addr_q_reg_10_:D	CLK_u0_pll_FOUTPOSTDIV	max	4.0010110r	3.5354060	0.0063250	-0.4656050	263.2545369
u0_soc_top/u0_sdram_axi/u_core/delay_state_q_reg_2_:D	CLK_u0_pll_FOUTPOSTDIV	max	4.0193970r	3.5543930	0.0063250	-0.4650040	263.2961945
u0_soc_top/u0_sdram_axi/u_core/addr_q_reg_10_:D	CLK_u0_pll_FOUTPOSTDIV	max	4.0010110r	3.5382220	0.0063250	-0.4627890	263.4498388
u0_soc_top/u0_sdram_axi/u_core/addr_q_reg_10_:D	CLK_u1_clk_XC	max	4.0219070r	10.2233020	0.0094510	6.2013950	263.2545369
u0_soc_top/u0_sdram_axi/u_core/delay_state_q_reg_2_:D	CLK_u1_clk_XC	max	4.0402930r	10.2422890	0.0094510	6.2019960	263.2961945
u0_soc_top/u0_sdram_axi/u_core/addr_q_reg_10_:D	CLK_u1_clk_XC	max	4.0219070r	10.2261180	0.0094510	6.2042110	263.4498388
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_2_:D	CLK_u0_clk_XC	min	0.7698750f	0.4061450	0.0000000	0.3637300	NA
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_120_:D	CLK_u0_clk_XC	min	0.7697530f	0.4050820	0.0000000	0.3646710	NA
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_102_:D	CLK_u0_clk_XC	min	0.7699230f	0.4051990	0.0000000	0.3647240	NA
u0_soc_top/u0_sdram_axi/u_axi/req_axburst_q_reg_0_:D	CLK_u0_pll_FOUTPOSTDIV	min	0.7970260f	0.4106080	0.0063250	0.3864180	NA
u0_soc_top/u0_sdram_axi/u_axi/req_hold_rd_q_reg:D	CLK_u0_pll_FOUTPOSTDIV	min	0.8211760f	0.4139360	0.0063250	0.4072400	NA
u0_soc_top/u0_sdram_axi/u_axi/req_prio_q_reg:D	CLK_u0_pll_FOUTPOSTDIV	min	0.8252400f	0.4136610	0.0063250	0.4115790	NA
u0_soc_top/u0_sdram_axi/u_axi/req_axburst_q_reg_0_:D	CLK_u1_clk_XC	min	0.8147960f	0.4283780	0.0094510	0.3864180	NA
u0_soc_top/u0_sdram_axi/u_axi/req_hold_rd_q_reg:D	CLK_u1_clk_XC	min	0.8389460f	0.4317060	0.0094510	0.4072400	NA
u0_soc_top/u0_sdram_axi/u_axi/req_prio_q_reg:D	CLK_u1_clk_XC	min	0.8430100f	0.4314310	0.0094510	0.4115790	NA

Clock	Delay Type	TNS
CLK_u0_clk_XC	max	0.0000000
CLK_u0_pll_FOUTPOSTDIV	max	-19.0360560
CLK_u1_clk_XC	max	0.0000000
CLK_u0_clk_XC	min	0.0000000
CLK_u0_pll_FOUTPOSTDIV	min	0.0000000
CLK_u1_clk_XC	min	0.0000000

Point	Fanout	Capacitance	Resistance	Transition	Delta Delay	Derate	Incr	Path
spi_miso_pad (port)		8.0000000	0.0010000	0.0000000		1.0000000	0.0000000	0.0000000r
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_34_:D (DFCNQD1BWP40P140LVT)		0.0006225	68.0671997	0.0204720		1.0000000	0.0000000	30.7017020r
u0_soc_top/u0_spi_flash/u0_spi_top/shift/data_reg_34_:CP (DFCNQD1BWP40P140LVT)								r
library setup time							-0.0243050	40.2521020
clock uncertainty							-0.0810000	40.1711020
data require time								40.1711020
data arrival time								30.7017020
slack (MET)								9.4694000

iSTA：timing report解读

- asic_top.cap（报告违例电容）

```
Generate the report at 2023-06-19T11:54:48
+-----------------------------------------------+--------------+--------------+--------------+-------------------------+--------+
| Net / InstPin                                 | MaxCap       | Cap          | CapSlack     | CellPort                | Remark |
+-----------------------------------------------+--------------+--------------+--------------+-------------------------+--------+
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/n682 |              |              |              |                         |        |
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/U671:ZN | 0.035r/0.035f | 0.002r/0.002f | 0.034r/0.034f | ND3D1BWP35P140LVT/ZN |        |
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/n444 |              |              |              |                         |        |
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/U357:ZN | 0.034r/0.034f | 0.000r/0.000f | 0.034r/0.034f | OAI211D1BWP40P140LVT/ZN |        |
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/n337 |              |              |              |                         |        |
| u0_soc_top/u0_spi_flash/u0_spi_top/shift/U457:ZN | 0.035r/0.035f | 0.001r/0.001f | 0.035r/0.035f | ND3D1BWP35P140LVT/ZN |        |
+-----------------------------------------------+--------------+--------------+--------------+-------------------------+--------+
```

- asic_top.fanout（报告违例扇出）

```
Generate the report at 2023-06-19T11:54:49
+-------------------+-----------+---------+--------------+--------------------+--------+
| Net / InstPin     | MaxFanout | FanLoad | FanLoadSlack | CellPort           | Remark |
+-------------------+-----------+---------+--------------+--------------------+--------+
| clk_peri_20       |           |         |              |                    |        |
| clk_peri_20_buf:Z | 32        | 32      | 0            | CKBD16BWP35P140/Z  |        |
| clk_peri_39       |           |         |              |                    |        |
| clk_peri_39_buf:Z | 32        | 32      | 0            | CKBD16BWP35P140/Z  |        |
| clk_peri_52       |           |         |              |                    |        |
| clk_peri_52_buf:Z | 32        | 32      | 0            | CKBD16BWP35P140/Z  |        |
+-------------------+-----------+---------+--------------+--------------------+--------+
```

iSTA：timing report解读

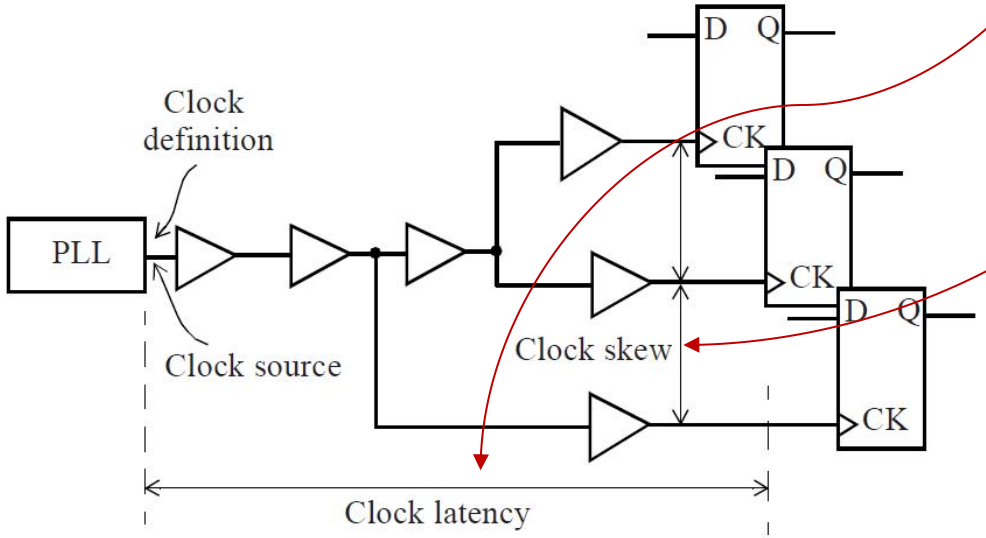
- asic_top.trans（报告违例转换时间）

```
Generate the report at 2023-06-19T11:54:47
+---------------------------+---------------+----------------+-----------------+-----------------------------+--------+
| Net / InstPin             | MaxTranTime   | TranTime       | TranSlack       | CellPort                    | Remark |
+---------------------------+---------------+----------------+-----------------+-----------------------------+--------+
| osc_100m_out_pad          |               |                |                 |                             |        |
| u1_clk:XOUT               | 5.000r/5.000f | 32.076r/1.552f | -27.076r/3.448f | PDXOEDG_V_G/XOUT            | R      |
| osc_25m_out_pad           |               |                |                 |                             |        |
| u0_clk:XOUT               | 5.000r/5.000f | 32.067r/1.552f | -27.067r/3.448f | PDXOEDG_V_G/XOUT            | R      |
| u0_rcg/u0_pll_clk         |               |                |                 |                             |        |
| u0_rcg/u0_pll:FOUTPOSTDIV | 0.267r/0.267f | 0.000r/0.000f  | 0.267r/0.267f   | PLLTS28HPMLAINT/FOUTPOSTDIV |        |
+---------------------------+---------------+----------------+-----------------+-----------------------------+--------+
```

iSTA：timing report解读

- asic_top_hold .skew（报告hold模式下的时钟偏斜）
- asic_top_setup.skew（报告setup模式下的时钟偏斜）

```
Generate the report at 2023-06-19T11:54:49
Clock: CLK_u0_clk_XC
+-----------------------------------------------------------------------------------------------------------------------------------------+---------+-------+------+
| Clock Pin                                                                                                                               | Latency | Skew  |      |
+-----------------------------------------------------------------------------------------------------------------------------------------+---------+-------+------+
| u0_soc_top/u0_nic400_bus/u_cd_hs_peri/u_ib_nic400_axi4_sdram_ib_m/u_maskcntl/reg_mask_r_reg:CP (DFCNQD1BWP40P140LVT)                     | 0.318   |       | rp-+ |
| u0_soc_top/u0_sdram_axi/u_core/active_row_q_reg_2__11_:CP (DFCNQD1BWP40P140LVT)                                                         | 0.409   | 0.082 | rp-+ |
| u0_soc_top/u0_nic400_bus/u_cd_hs_peri/u_ib_nic400_axi4_sdram_ib_m/u_maskcntl/reg_mask_r_reg:CP (DFCNQD1BWP40P140LVT)                     | 0.318   |       | rp-+ |
| u0_soc_top/u0_sdram_axi/u_core/active_row_q_reg_1__4_:CP (DFCNQD1BWP40P140LVT)                                                          | 0.409   | 0.082 | rp-+ |
| u0_soc_top/u0_nic400_bus/u_cd_hs_peri/u_ib_nic400_axi4_sdram_ib_m/u_maskcntl/reg_mask_r_reg:CP (DFCNQD1BWP40P140LVT)                     | 0.318   |       | rp-+ |
| u0_soc_top/u0_sdram_axi/u_core/active_row_q_reg_2__4_:CP (DFCNQD1BWP40P140LVT)                                                          | 0.409   | 0.082 | rp-+ |
+-----------------------------------------------------------------------------------------------------------------------------------------+---------+-------+------+
```



总结

- **iSTA在使用层面支持了常用Tcl命令和sdc命令，具体如下**
  - Tcl主要包括read_verilog、link_design、read_liberty、read_sdc、read_spef、report_timing
  - sdc约束包含create_clock、create_generate_clock、set_multicycle_path
- **iSTA在算法层面实现了多种延时计算算子，支持了多种路径分析模式，具体如下**
  - Slew，Delay计算，包括NLDM/Elmore、CCS/Arnoldi方法
  - 传播算子：slew、delay、clock、data arrive time、data require time
  - 时序路径分析：setup/hold、removal/recovery、multicycle path、clock gate、AOCV等
- **未来计划：提升速度、准确度**

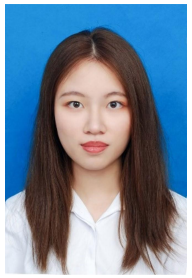
iEDA Tutorial 第一期议程

- Part1 iEDA-iSTA和iPW整体介绍 40min （陶思敏）

- Part2 iSTA工具架构、特性、API，使用 25min（龙帅英）

- Part3 iSTA关键技术研究 30min（刘贺）

- Part4 iPW工具架构、特性、关键技术、使用 25min（邵哲青）



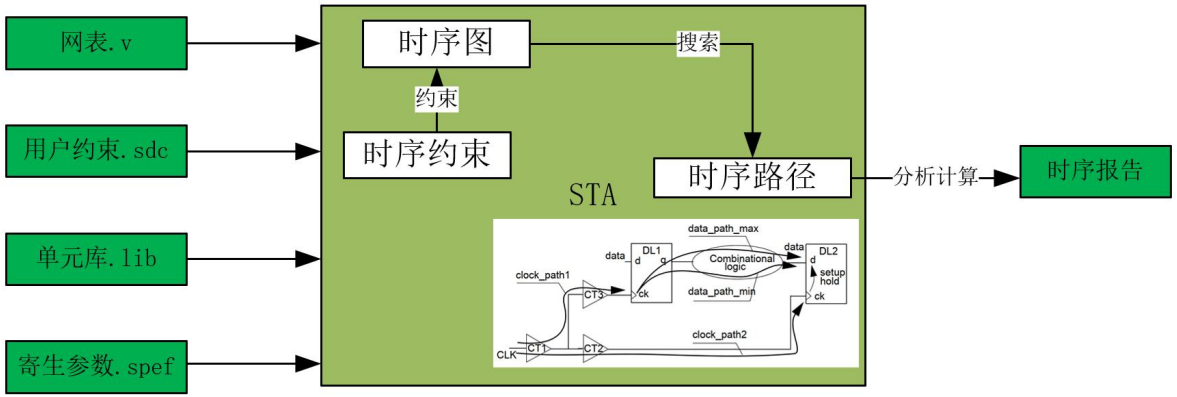
01 iSTA key technology

02 AI for iSTA

03 Future work

Key technologies in STA

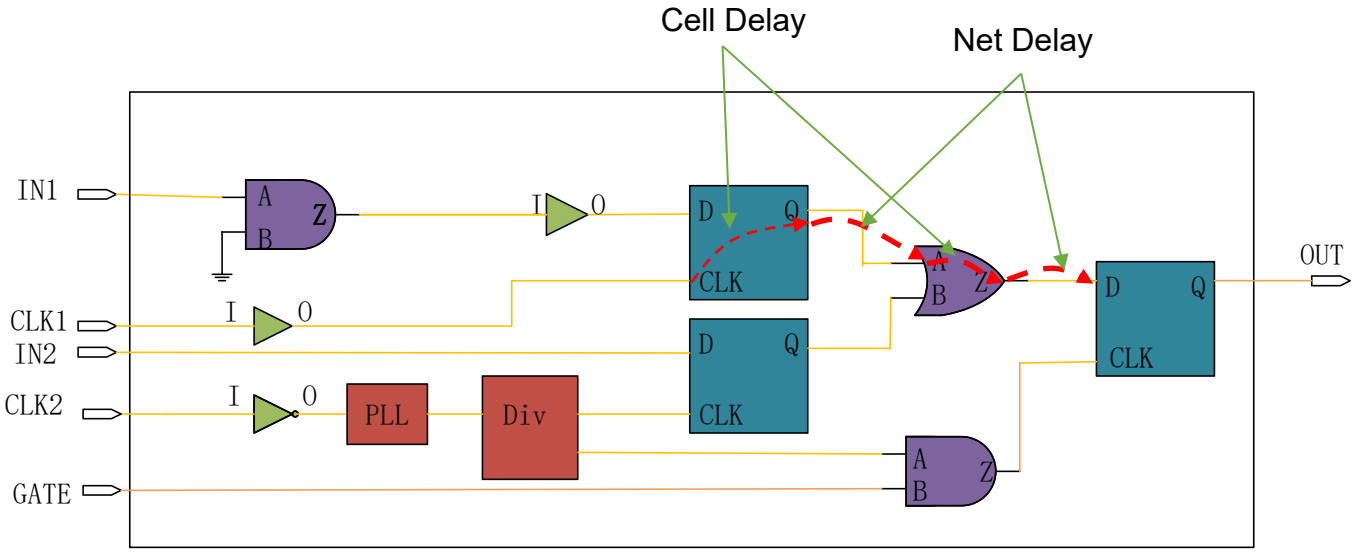
- Accuracy
  - Cell delay
  - Interconnect delay
  - Crosstalk
  - Miller capacitance
  - CPPR (Common Path Pessimism Removal)
- Performance
  - Graph propagation and traversal
  - Delay calculation model acceleration
    - GPU
    - AI



$$clockpath1 + data\_path\_max - clockpath2 + setup \leq clockperiod$$
$$clockpath1 + data\_path\_min - clockpath2 - hold \geq 0$$

Path delay calculation

- Cell delay
  - NLDM
- Interconnect delay
  - First order moment model
    - Elmore delay metric
  - Second order moment model
    - D2M delay metric
    - ECM delay metric
    - MD2M delay metric
  - High order model
    - Arnoldi delay metric



$$Path_{Delay} = Cell_{Delay} + Net_{Delay}$$

Cell delay calculation

- NLDM (Non-Linear Delay Model)
- LUT (Look-up table)
  - Liberty file (.lib)
  - Linear interpolation
  - Index_1 : Input transition
  - Index_2 : Output capacitance

```
pin (OUT) {
  max_transition : 1.0;
  timing() {
    related_pin : "INP1";
    timing_sense : negative_unate;
    cell_rise(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35     1.43 */ \
        /* 0.1 */   "0.0513,  0.1537,  0.5280", \
        /* 0.3 */   "0.1018,  0.2327,  0.6476", \
        /* 0.7 */   "0.1334,  0.2973,  0.7252");
      }
    cell_fall(delay_template_3x3) {
      index_1 ("0.1, 0.3, 0.7"); /* Input transition */
      index_2 ("0.16, 0.35, 1.43"); /* Output capacitance */
      values (  /*  0.16      0.35      1.43 */ \
        /* 0.1 */   "0.0617,  0.1537,   0.5280", \
        /* 0.3 */   "0.0918,  0.2027,   0.5676", \
        /* 0.7 */   "0.1034,  0.2273,   0.6452");
    }
```

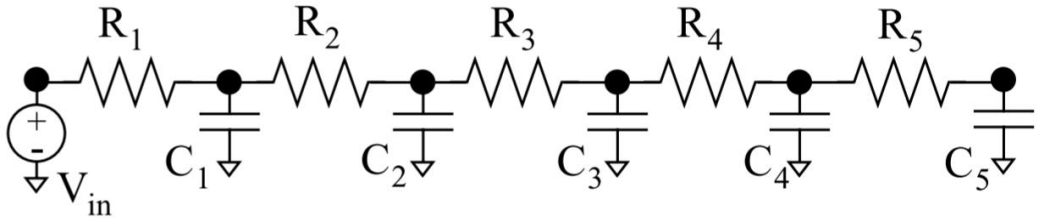
[1] J. Bhasker and R. Chadha, Static timing analysis for nanometer designs: A practical approach. Springer Science & Business Media, 2009.

Elmore delay metric

- Elmore delay metric
  - Update downstream capacitances of each node
  - Update delay from root to each node
- Characteristics
  - Simple closed form expression

$$\text{delay}_{\text{root}\to node\ j} = \sum_{\substack{nodes\ i \\ from \\ root\ to\ node\ j}} R_i\left(\sum downstream\ caps\right)$$

  - Fast computation speed
  - First moment of the impulse response
  - Errors are pronounced for near-end nodes
    - Resistance shielding is less dominent for far-end nodes
    - Upper bound on delay



A simple RC tree network

[2] Elmore W C. The transient response of damped linear networks with particular regard to wideband amplifiers[J]. Journal of applied physics, 1948, 19(1): 55-63.

[3] Alpert C J, Devgan A, Kashyap C. A two moment RC delay metric for performance optimization[C]//Proceedings of the 2000 international symposium on Physical design. 2000: 69-74.

D2M delay metric

- Delay with 2 moments
  - Step 1: Calculate Elmore delay

$$ED_i = \sum_{k=1}^{N} R_{ki} C_k$$

  - Step 2: Second moment of the impulse response, $m_0 = 1$

$$m_j^{(i)} = -\sum_{k=1}^{N} R_{ki} C_k m_{j-1}^{(k)}$$

  - Step 3:

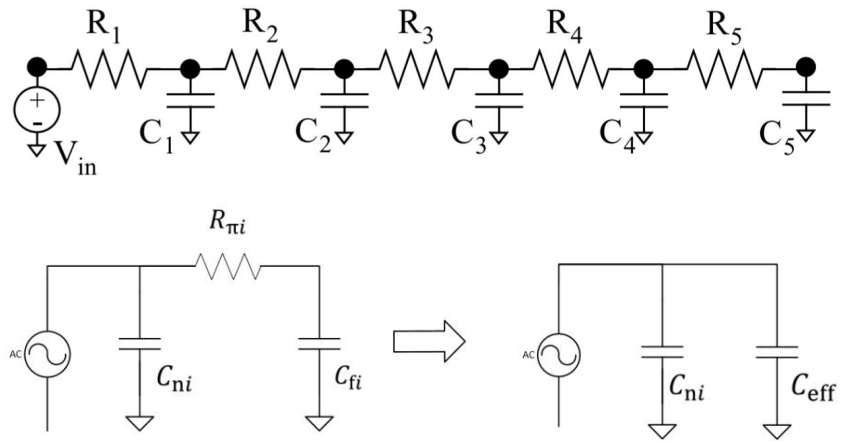
$$D2M = \frac{{m_1}^2}{\sqrt{m_2}} \ln 2$$

  - As simple and fast as the Elmore delay metric
  - Significantly more accurate than Elmore
  - Remarkably accurate at the far end of RC lines
  - Ignore resistance shielding

[4] Alpert C J, Devgan A, Kashyap C. A two moment RC delay metric for performance optimization[C]//Proceedings of the 2000 international symposium on Physical design. 2000: 69-74.

ECM delay metric

- Effective capacitance metric
  - Downstream capacitance is modeled by an effective capacitance insteads of the sum of capacitances
  - Consider resistance shielding



Effective capacitance computatiaon based on a reduced model.

Formulated in a taylor series

Step1: $Y_i(s) = y_{1,i}s + y_{2,i}s^2 + y_{3,i}s^3 + \cdots$

Calculated with the three moments

Step2: $R_{\pi i} = -\frac{y_{3,i}^2}{y_{2,i}^3}, \quad C_{\mathrm{f}i} = \frac{y_{2,i}^2}{y_{3,i}}, \quad C_{\mathrm{n}i} = y_{1,i} - C_{\mathrm{f}i}$

Effective capacitance

Step3: $C_{\mathrm{eff}} = C_{\mathrm{f}i}\left(1 - \mathrm{e}^{-ED_i/\tau_i}\right)$

Effective load capacitance

Step4: $C_{leff} = C_{\mathrm{n}i} + C_{\mathrm{eff}}$

[5] Kashyap, C. V, C. J. Alpert, and A. Devgan, “An” effective” capacitance based delay metric for rc interconnect,” in IEEE/ACM International Conference on Computer Aided Design. ICCAD-2000. IEEE/ACM Digest of Technical Papers (Cat. No. 00CH37140), pp. 229–234, IEEE, 2000.

ECM delay metric

- Expression

$$D_{ECM} = \sum_{i \in N} (R_i * \sum C_{leff})$$

- Same form as the Elmore formula
- Same complexity as the Elmore delay
- More accurate than Elmore delay
- Do not require the computation of multiple moments

MD2M delay metric

- Modified delay with 2 moments

$$D_{MD2M} = \frac{m_1'^2}{\sqrt{m_2'}} \ln 2$$

  - where m_1' and m_2' are the modified first two moments of the impulse response. Utilize the effective capacitance of the downstream capacitances from node i to substitute the sum of downstream capacitances.
- As simple and fast as the D2M delay metric
- Significantly more accurate than D2M
- Remarkably accurate at the far end of RC lines
- Consider resistance shielding

High order moment delay metric

Parematers extraction, Modeled as RLC/RC circuit, KVL/KCL

$$\begin{cases} G_x x(t) + C_x \dot{x}(t) = B_x u(t) \\ y(t) = L_x^T x(t) \end{cases}$$

Laplace transform

$$\begin{cases} (G_x + sC_x)x(s) = B_x u(s) \\ y(s) = L_x^T x(s) \end{cases}$$

Transfer function

$$H_f(s) = \frac{y(s)}{u(s)} = L_x^T (G_x + sC_x)^{-1} B_x$$

$$A_x = G_x^{-1} C_x, \quad R_x = G_x^{-1} B_x$$
$$H_f(s) = L_x^T (I + sA_x)^{-1} R_x$$

Original circuit system

Algorithm 1 Arnoldi Algorithm

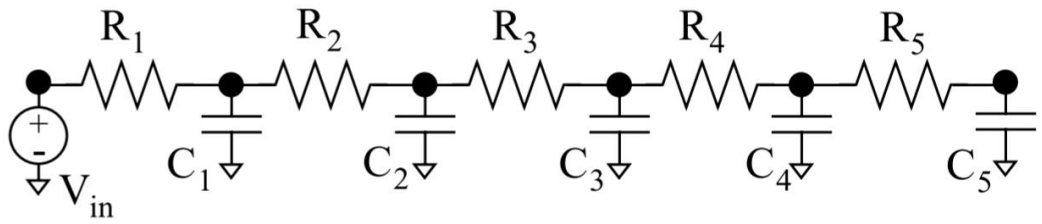
Input: A, b

Output: V, H

1: Compute $v_1 = \frac{b}{\|b\|_2}$
2: **for** $j = 1$ to q **do**
3: Compute $w_j = A \cdot v_j$
4: **for** $i = 1$ to j **do**
5: $h_{ij} = v_i^T \cdot w_j$
6: $w_j = w_j - h_{ij} \cdot v_i$
7: **end for**
8: **if** $w_j = 0$ **then**
9: Break and return
10: **else**
11: $h_{j+1,j} = \|w_j\|_2$
12: $v_{j+1} = \frac{w_j}{h_{j+1,j}}$
13: **end if**
14: **end for**

$$\tilde{G}_x = W^T G_x V, \quad \tilde{C}_x = W^T C_x V, \quad \tilde{B}_x = W^T B_x \quad \tilde{L}_x = V^T L_x$$

Moment: $\bar{M}_k = \bar{L}^T \bar{A}^k \bar{R}$



$$\begin{cases} \left(\tilde{G}_x + s\tilde{C}_x\right)\tilde{x}(s) = \tilde{B}_x u(s) \\ \tilde{y}(s) = \tilde{L}_x^T \tilde{x}(s) \end{cases}$$

$$\tilde{H}_f(s) = \frac{\tilde{y}(s)}{u(s)} = \tilde{L}_x^T \left(\tilde{G}_x + s\tilde{C}_x\right)^{-1} \tilde{B}_x$$

Reduced order system

1. Match q moments for qth-order approximation
2. Numerical stability
3. Passivity preservation

[8] Odabasioglu A, Celik M, Pileggi L T. PRIMA: Passive reduced-order interconnect macromodeling algorithm[J]. IEEE TCAD, 1998, 17(8): 645-654.

Elmore metric – First order moment

Transfer function

$$H_f(s) = \frac{y(s)}{u(s)} = L_x^T(G_x + sC_x)^{-1}B_x$$

$$A_x = G_x^{-1}C_x, \quad R_x = G_x^{-1}B_x$$
$$H_f(s) = L_x^T(I + sA_x)^{-1}R_x$$

Taylor expansion

$$\mathrm{h(s)} = \sum_{k=0}^{\infty} m_k s^k$$

$$m_k = \frac{1}{k!} \times \frac{d^k h(s)}{ds^k}|_{s=0}$$

where the k[th] coefficeint of *h(s)*, m_k is called the k^{th} moment.

For the impulse function δ(t), its Laplace transformation is 1. The transfer function is also the impulse response at the port. For impulse function h(t), after Laplece transform and Taylor expand

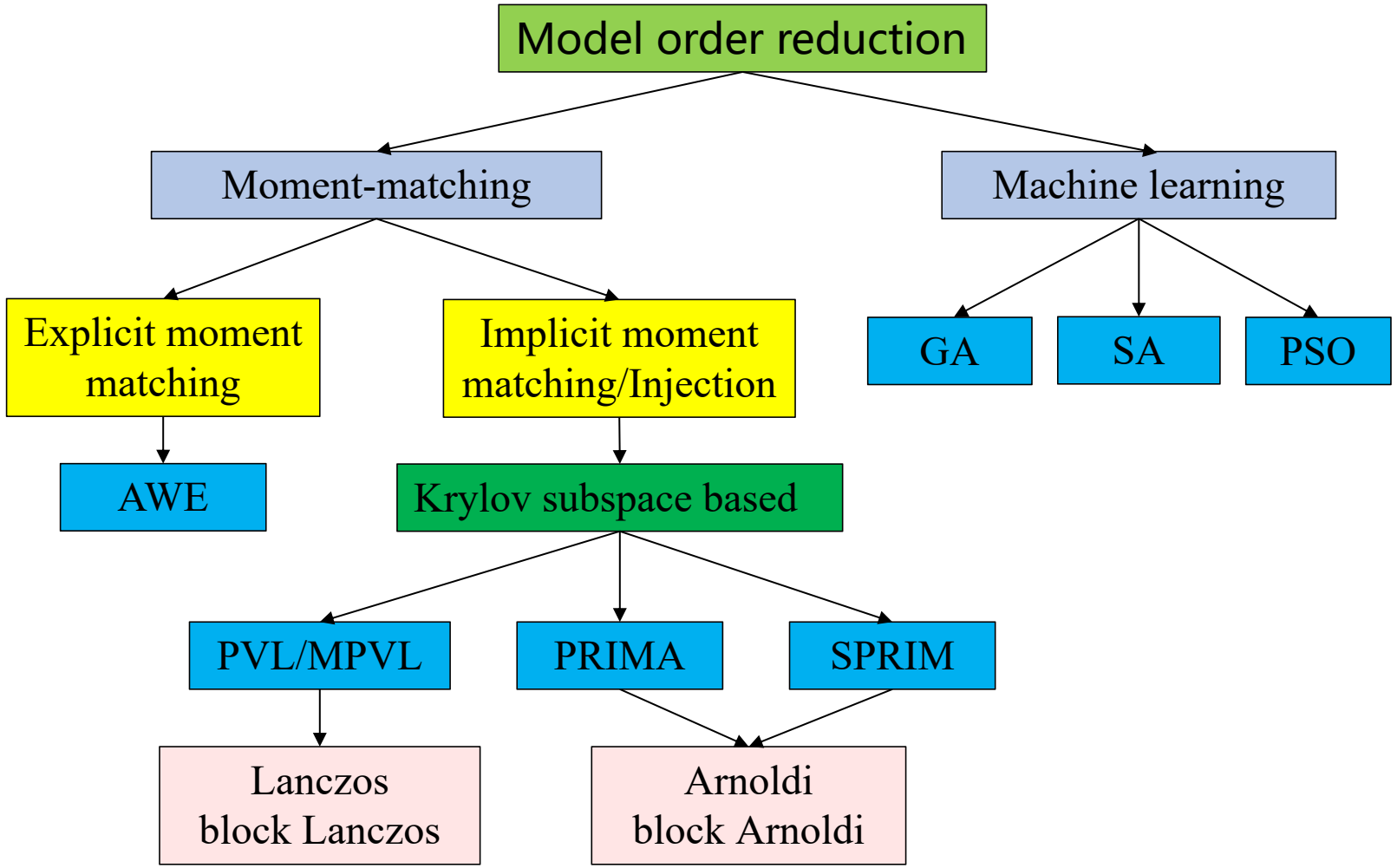
$$\mathrm{h}(s) = \int_0^{\infty} e^{-st}\,\mathrm{h(t)dt}$$

For ideal delay element's transformer function is

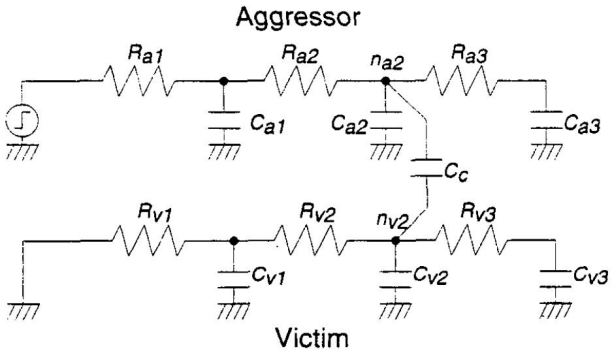
$$\mathrm{h}_d(\mathrm{s}) = \mathrm{e}^{-sT}$$

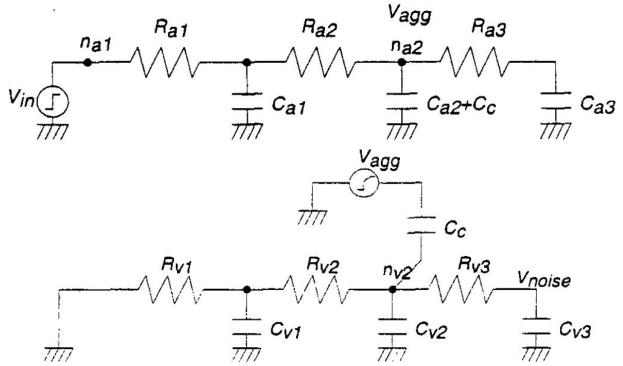
$$\mathrm{T}_d = -\frac{\mathrm{d}(\mathrm{h}_d(\mathrm{s}))}{ds}|_{s=0} = m_1$$

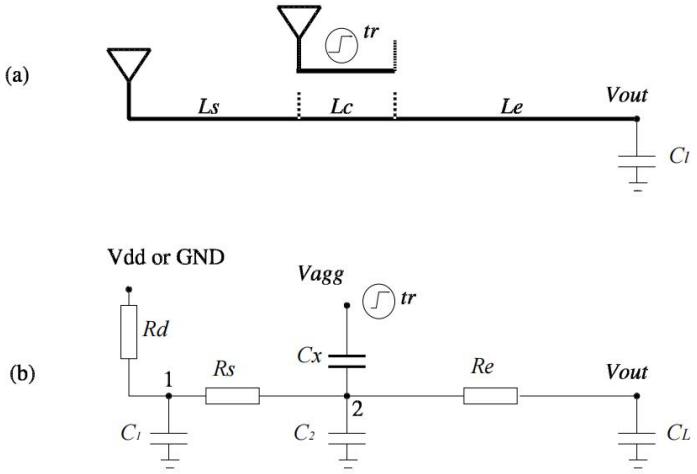
Model order reduction methods

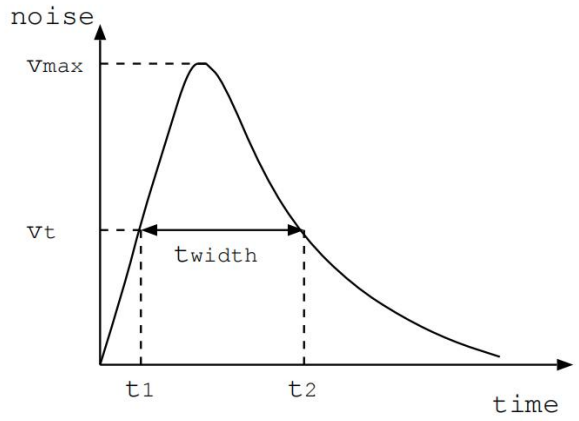


Crosstalk noise calculation









- The impedance at node 1, satisfying the following

$$\frac{1}{Z_1} = \frac{1}{R_d} + sC_1$$

- Then at node 2, we have

$$\frac{1}{Z_2} = \frac{1}{(Z_1 + R_s)} + sC_2 + \frac{1}{R_e + \frac{1}{sC_L}}$$

- Denote the **s**-domain voltage at node 2 by $V_2(S)$, then

$$V_2(s) = \frac{Z_2}{Z_2 + \frac{1}{sC_x}} \cdot V_{agg}(s).$$

- The output voltage $V_{out}(S)$ in the s-domain is

$$V_{out}(s) = V_2(s) \cdot \frac{\frac{1}{sC_L}}{R_e + \frac{1}{sC_L}}.$$

- Substituting Z_1, Z_2 and V_2 into $V_{out}(S)$, we have

$$\begin{aligned} V_{out}(s) &= \frac{Z_2}{Z_2 + (\frac{1}{sC_x})} \cdot \frac{1/sC_L}{R_e + 1/sC_L} \cdot V_{agg}(s) \\ &= \frac{a_2 s^2 + a_1 s}{s^3 + b_2 s^2 + b_1 s + b_0} \cdot V_{agg}(s) \end{aligned}$$

[9]Cong J, Pan D Z, Srinivas P V. Improved crosstalk modeling for noise constrained interconnect optimization[C]//Proceedings of the 2001 Asia and South Pacific Design Automation Conference. 2001: 373-378.

[10]Takahashi M, Hashimoto M, Onodera H. Crosstalk noise estimation for generic RC trees[C]//Proceedings 2001 IEEE International Conference on Computer Design: VLSI in Computers and Processors. ICCD 2001. IEEE, 2001: 110-116.

Crosstalk noise calculation

- For the aggressor with saturated ramp input with normalized $V_{dd=}$ 1 and transition time t_r

$$v_{agg}(t) = \begin{cases} t/t_r & 0 \le t \le t_r \\ 1 & t > t_r, \end{cases}$$

- Its Laplace transformation is

$$V_{agg}(s) = \frac{1 - e^{-st_r}}{s^2 t_r}$$

- Using dominant-pole approximation method

$$V_{out}(s) \approx \frac{a_1 s}{b_1 s + b_0} \cdot V_{agg}(s) = \frac{t_x(1 - e^{-st_r})}{st_r(st_v + 1)}$$

- Computing the inverse Laplace transform, time domain waveform

$$v_{out}(t) = \begin{cases} \frac{t_x}{t_r}(1 - e^{-t/t_v}) & 0 \le t \le t_r \\ \frac{t_x}{t_r}(e^{-(t-tr)/t_v} - e^{-t/t_v}) & t > t_r \end{cases}$$

$$v_{max} = \frac{t_x}{t_r}(1 - e^{-t_r/t_v})$$

- v'_{max} is indeed a first-order approximation of v_{max}

$$\begin{aligned} \frac{t_x}{t_r}(1 - e^{-t_r/t_v}) &= \frac{t_x}{t_v}[1 - \frac{1}{2}\frac{t_r}{t_v} + \ldots] \\ &\approx \frac{t_x}{t_v}\frac{1}{1 + \frac{1}{2}\frac{t_r}{t_v}} = \frac{t_x}{t_v + t_r/2} \end{aligned}$$

- Noise Width: Given certain threshold voltage level v_t , the noise width for a noise pulse is defined to be the length of time interval that noise spike voltage v is larger or equal to v_t.

$$t_2 - t_1 = t_v ln\left[\frac{(t_x - t_r v_t)(e^{t_r/t_v} - 1)}{t_r v_t}\right]$$

$$v_t = v_{max}/2$$

$$t_{width} = t_2 - t_1 = t_r + t_v ln\left[\frac{1 - e^{-2t_r/t_v}}{1 - e^{-t_r/t_v}}\right]$$

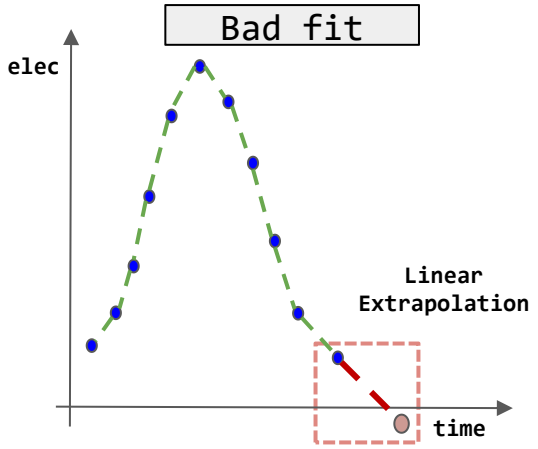
01 iSTA key technology

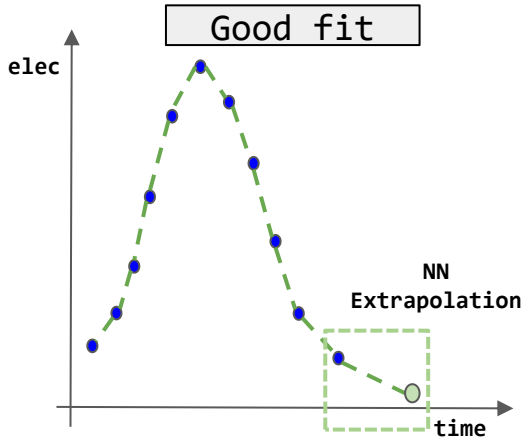
02 AI for iSTA

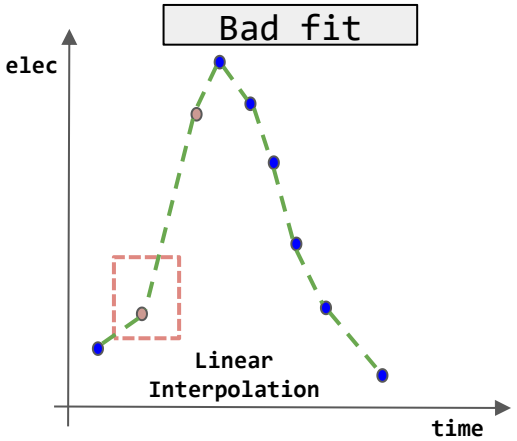
03 Future work

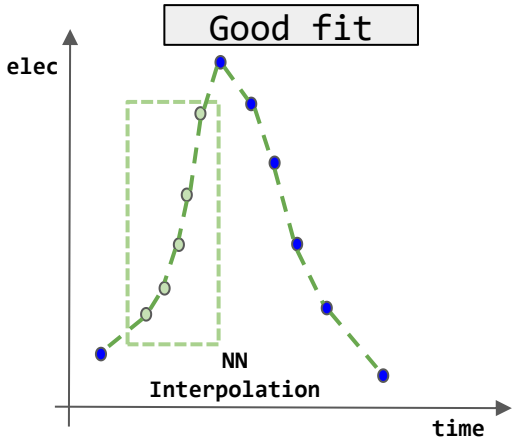
① AI for interpolation

- Instability of the extrapolation
- Uneven distribution of difference points



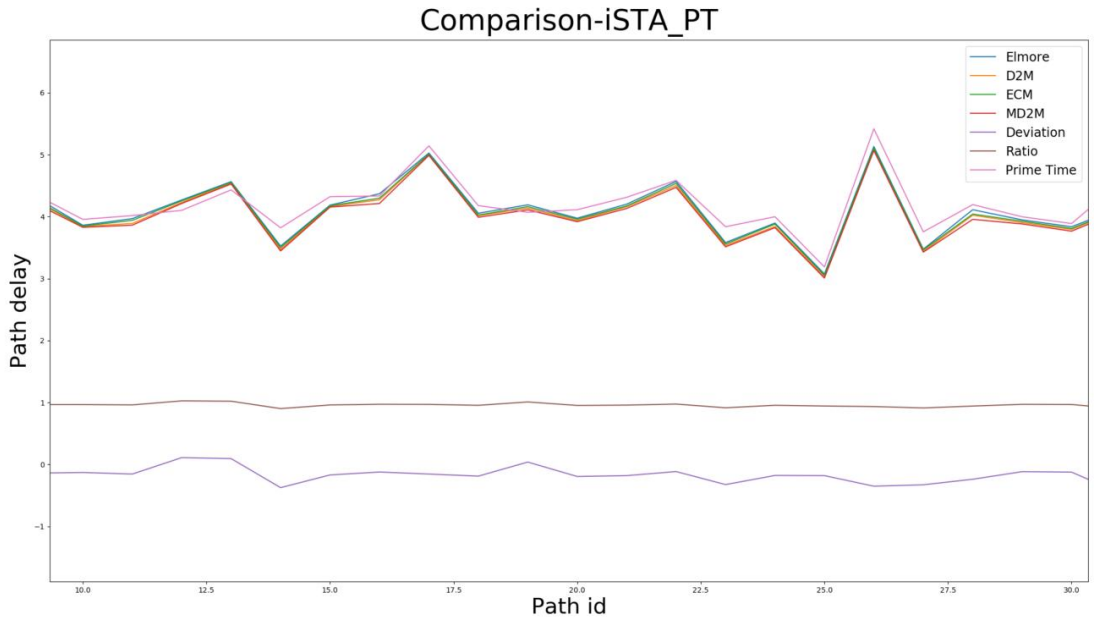


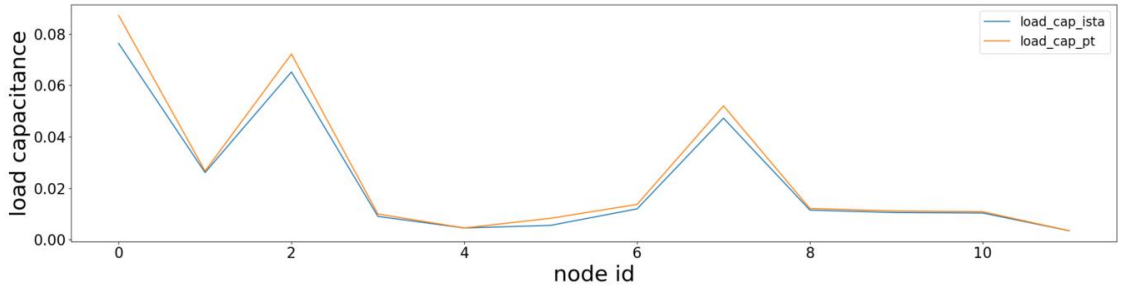


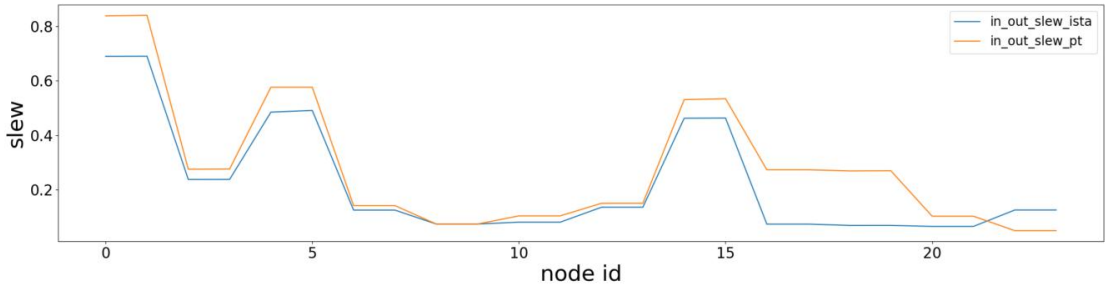


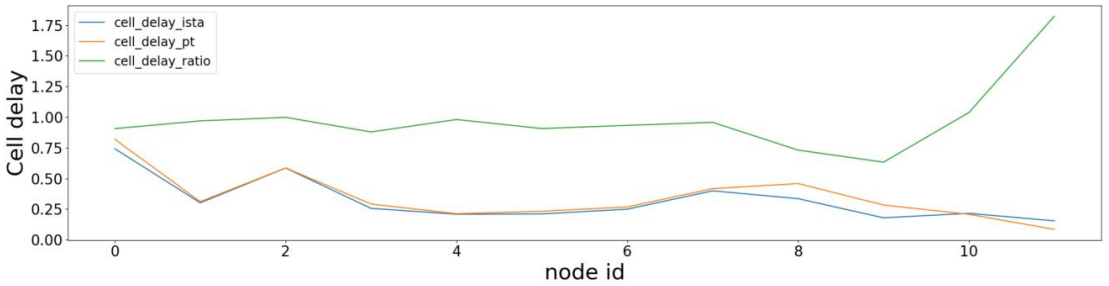
② Timing path delay calibration

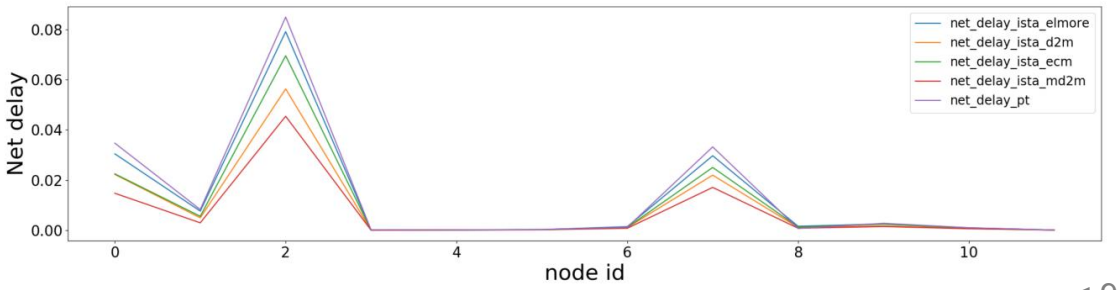
- Our goal:
  - Machine learning model
  - Lower-order moments delays which are efficient and easily computable
  - More accurate results in a acceptable short time



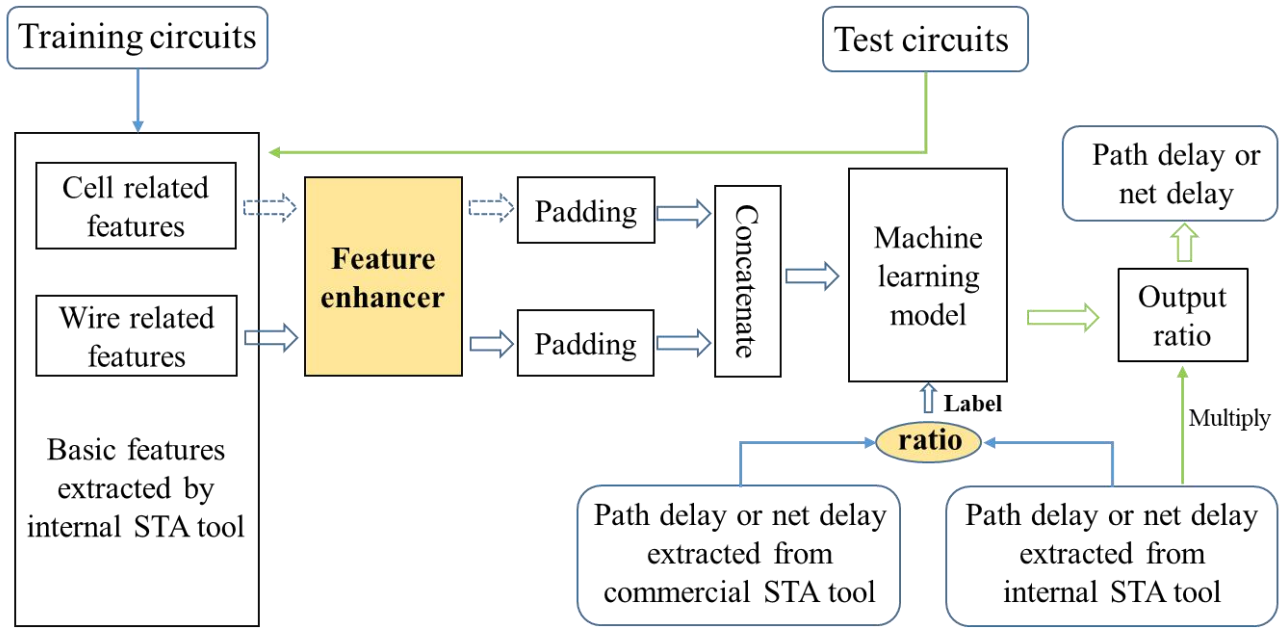








Overall Flow



Overall flow architecture.

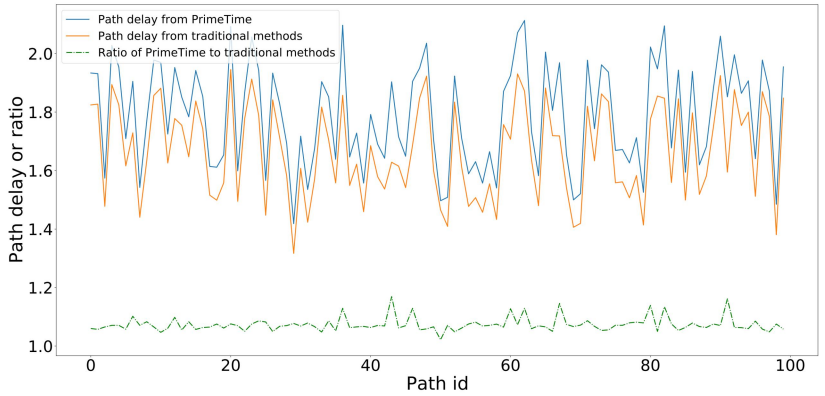
- Features extraction
  - iEDA-iSTA internal tool
- Label extraction
  - Prime Time
- Model selection
  - Transformer+ResNet
- Feature enhancer
  - Resistance shielding
  - Effective capacitance
  - ECM、MD2M
- Trick
  - Ratio of ground truth to traditional methods

Features and Label Selection

- **Timing path delay learning**
  - **Feature selection**
    - Input slew
    - Load capacitance
    - Output slew
    - Signal polarity
    - Cell type
    - Cell port name
    - Arrive time
      - Cell delay
      - Wire delay
        - Elmore delay

- **Label selection**
  - PD_{PT}
  - $\frac{PD_{PT}}{PD_{IE}}$

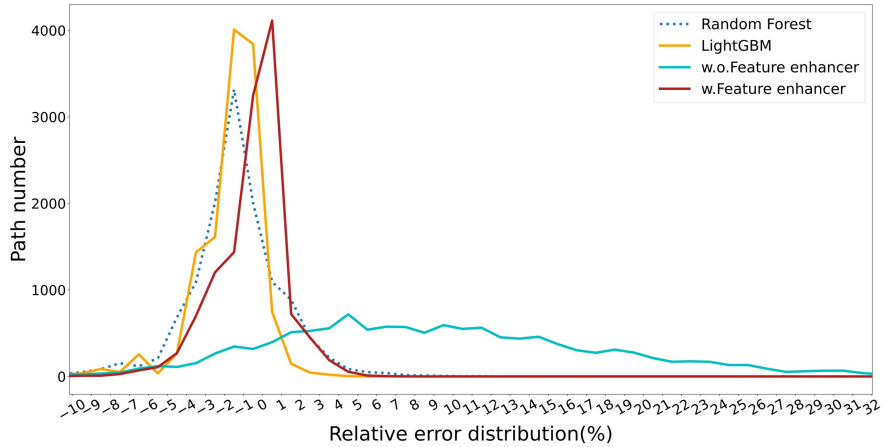
$$PD_{\text{pred}} = f\left(S_i, SP_{r/s}, CT, CP, S_o, C_l, PD_{FE}\right) \quad (9)$$



The distribution of 100 paths delay of design “aes cipher top”. The pathdelay curves are marked in blue (PD_{PT}) and orange (PD_{IE}), respectively. Andthe curve marked in green represents PD_{PT}/PD_{IE}.

Experiment result

- As for the slow corner, the performance of our model with feature enhancer in terms of average rRMSE values are 1.1% and 2.5% for trained designs and unseen designs,which are reduced by 1.55 × and 10.48 × compared with model trained without feature enhancer, respectively.



Error distribution of different models on unseen circuit “gcd”

	Design	Accuracy of slow corner (rRMSE / AAE (ps))							Accuracy of fast corner (rRMSE / AAE (ps))						
		RF	LGBM	w.o.FE	w.FE	$Ratio_1$	$Ratio_2$	$Ratio_3$	RF	LGBM	w.o.FE	w.FE	$Ratio_1$	$Ratio_2$	$Ratio_3$
Train	picorv32	0.008/ 8.10	0.007/5.06	0.016/23.02	0.007/5.61	1.14/1.44	1.0/0.90	2.29/4.10	0.007/2.60	0.008/4.88	0.021/18.83	0.012/11.14	0.58/0.23	0.67/0.44	1.75/1.69
	jpeg_encoder	0.075/170.66	0.069/147.85	0.015/29.83	0.014/22.86	5.36/7.47	4.93/6.47	1.07/1.30	0.039/25.38	0.039/24.28	0.033/23.76	0.032/22.10	1.22/1.15	1.22/1.10	1.03/1.08
	aes_cipher	0.046/83.10	0.046/84.09	0.022/30.48	0.018/20.37	2.56/4.08	2.56/4.13	1.22/1.50	0.023/25.46	0.023/23.46	0.018/22.70	0.017/21.68	1.35/1.17	1.35/1.08	1.06/1.05
	ibex_core	0.138/1053.81	0.128/977.36	0.007/37.81	0.007/37.88	19.71/27.82	18.29/25.80	1.0/0.998	0.022/25.0	0.021/19.19	0.023/32.36	0.021/25.83	1.05/0.97	1.0/0.74	1.10/1.25
Average		0.032/84.75	0.030/77.80	0.017/26.54	0.011/13.20	2.91/6.42	2.73/5.89	1.55/2.01	0.019/17.45	0.019/16.06	0.022/23.97	0.018/19.16	1.06/0.91	1.06/0.84	1.22/1.25
Test	riscv	0.029/125.04	0.019/79.94	0.133/559.52	0.027/119.70	1.07/1.04	0.70/0.67	4.93/4.67	0.021/59.01	0.020/54.10	0.058/164.08	0.015/39.57	1.40/1.49	1.33/1.37	3.87/4.15
	gcd	0.028/23.33	0.025/20.75	0.361/286.60	0.022/18.88	1.27/1.24	1.14/1.10	16.41/15.18	0.025/20.39	0.025/19.43	0.128/109.95	0.019/13.98	1.32/1.46	1.32/1.39	6.74/7.86
Average		0.0286/89.75	0.021/59.41	0.212/464.84	0.025/84.72	1.14/1.06	0.84/0.70	10.48/5.49	0.022/52.24	0.021/48.02	0.070/154.59	0.016/35.08	1.38/1.49	1.31/1.37	4.38/4.41

③ Wire delay learning

- **Wire delay learning**
  - **Feature selection**

Category	Features
Basic features	Port name
	Resistance
	Capacitance
Feature enhancer_1	Elmore delay
	D2M delay
Feature enhancer_2	Elmore delay
	D2M delay
	ECM delay
	MD2M delay

- **Label selection**
  - D_{Golden}
  - $\frac{D_{Golden}}{D_{Avg}}$

Choose the ratio D_{Golden}/D_{Avg} of golden wire delay D_{Golden} to average wire delay D_{Avg} of the four methods above as the label.

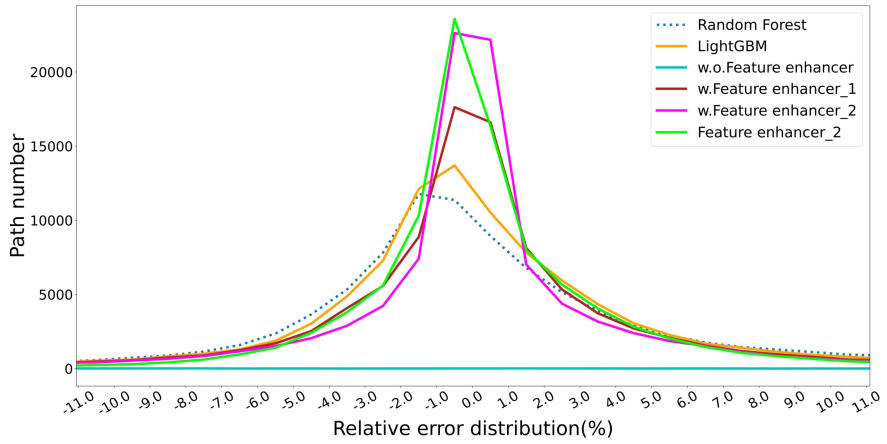
$$D_{\text{pred}} = f\left(D_{Elmore}, D_{D2M}, D_{ECM}, D_{MD2M}, D_{Avg}\right)$$

Where D_{pred} is the predicted wire delay.

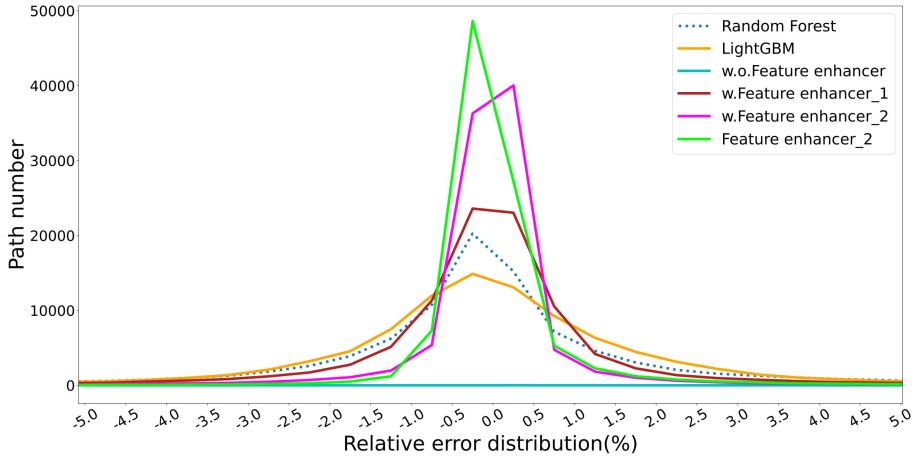
Experiment result

- **Experiment 2 – Wire delay learning**
  - Designs from 4th integrated circuit EDA elite challenge, based on 28nm technology.
  - our model achieves an average rRMSE values of 0.4% for trained designs and 3.3% for unseen designs.

	Design	Net	Wire path	Accuracy of multi-corners (rRMSE / AAE (ps))											FE runtime (s)
				RF	LGBM	w.o.FE	w.FE_1	w.FE_2	FE_2	$Ratio_1$	$Ratio_2$	$Ratio_3$	$Ratio_4$	$Ratio_5$	
Train	Group0	53599	99465	0.033/0.87	0.021/0.81	0.218/8.70	0.023/0.48	0.005/0.21	0.005/0.19	6.6/4.58	4.2/4.26	43.6/45.79	4.6/2.53	1.0/1.11	69.7
	Group1	53599	100227	0.035/1.04	0.021/0.89	0.195/9.28	0.026/0.56	0.005/0.23	0.005/0.23	7.0/4.52	4.2/3.87	39.0/40.35	5.2/2.43	1.0/1.0	57.9
	Group2	53599	98871	0.033/1.01	0.020/0.91	0.199/8.79	0.023/0.56	0.005/0.24	0.004/0.23	8.25/4.39	5.0/3.96	49.75/38.22	5.75/2.43	1.25/1.04	65.4
	Group5	53599	98460	0.037/1.39	0.022/1.11	0.191/10.20	0.023/0.70	0.004/0.30	0.004/0.28	9.25/4.96	5.5/3.96	47.75/36.43	5.75/2.50	1.0/1.07	47.9
	Group6	53599	99539	0.034/1.25	0.021/1.09	0.195/10.25	0.023/0.68	0.004/0.29	0.004/0.28	8.5/4.46	5.25/3.89	48.75/36.61	5.75/2.43	1.0/1.04	47.8
	Group7	53599	98429	0.033/1.01	0.021/0.91	0.202/8.64	0.022/0.54	0.005/0.24	0.004/0.22	8.25/4.60	5.25/4.14	50.5/39.27	5.5/2.46	1.25/1.09	65.0
	Group8	53599	99353	0.032/0.75	0.022/0.71	0.248/9.01	0.023/0.40	0.005/0.16	0.005/0.16	6.4/4.69	4.4/4.44	49.6/56.31	4.6/2.5	1.0/1.0	67.9
	Group9	53599	98413	0.036/1.09	0.022/0.94	0.207/8.78	0.023/0.54	0.005/0.23	0.004/0.21	9.0/5.19	5.5/4.48	51.75/41.81	5.75/2.57	1.25/1.10	67.1
Average				0.034/1.05	0.021/0.92	0.207/9.21	0.023/0.56	0.005/0.24	0.004/0.23	8.5/4.57	5.25/4.0	51.75/40.04	5.75/2.43	1.25/1.04	61.1
Test	Group3	53599	100413	0.094/3.76	0.076/2.92	0.205/9.84	0.083/2.20	0.042/1.91	0.032/1.61	2.94/2.34	2.38/1.81	6.41/6.11	2.59/1.37	1.31/1.19	59.6
	Group4	53599	97956	0.090/3.15	0.074/2.45	0.211/9.50	0.090/1.86	0.042/1.58	0.033/1.39	2.73/2.27	2.24/1.76	6.39/6.83	2.73/1.34	1.27/1.14	62.2
Average				0.092/3.46	0.075/2.69	0.208/9.67	0.087/2.03	0.042/1.75	0.033/1.5	2.79/2.31	2.27/1.79	6.30/6.45	2.64/1.35	1.27/1.17	60.9



Relative error distribution of different models and features on unseen benchmark “Group4”.



Relative error distribution of different models and features on trained benchmark “Group5”.

01 iSTA key technology

02 AI for iSTA

03 Future work

Future work for iSTA

- Crosstalk calculation and prediction
  - Timing windows-multi aggressors
  - Crosstalk delta delay
  - Crosstalk prediction
- Miller capacitance
  - NLDM provides a single capacitance
  - Ignores the Miller effect
  - CCS – provides two different pin capacitance
- Relationship between variables of different stages
  - Relationship between physical and electrical variables
  - Relationship between electrical and timing variables

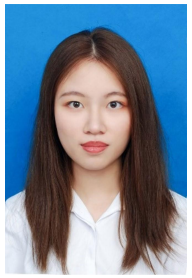
iEDA Tutorial 第一期议程

- Part1 iEDA-iSTA和iPW整体介绍 40min （陶思敏）

- Part2 iSTA工具架构、特性、API与使用 25min（龙帅英）

- Part3 iSTA关键技术研究 30min（刘贺）

- Part4 iPW工具架构、特性、关键技术与使用 25min（邵哲青）

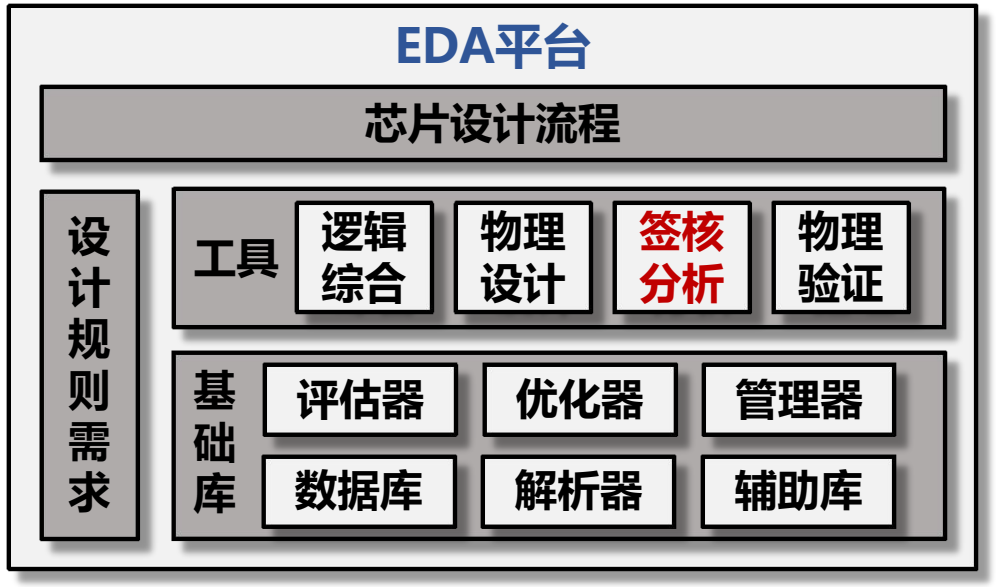


iPW工具介绍

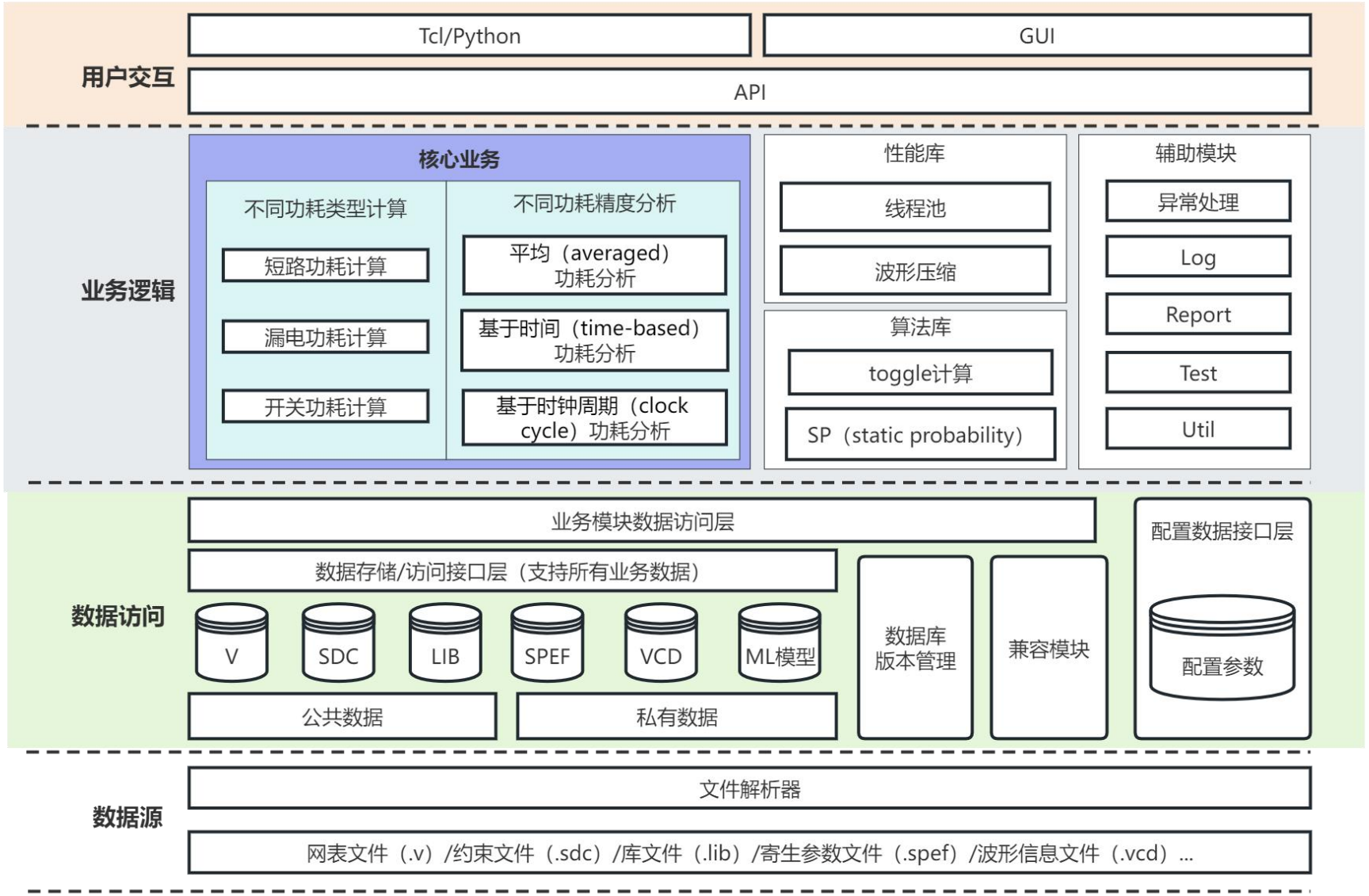
一款开源的芯片功耗分析工具，作为iEDA项目组开源EDA工具链的一部分。

设计目标：

1. 做到在芯片设计的签核阶段分析芯片功耗，提供分析报告；
2. 将功耗分析算法拆解成基础模块，可以作为功耗引擎提供丰富接口供物理设计调用，并支持机器学习的功耗预测技术



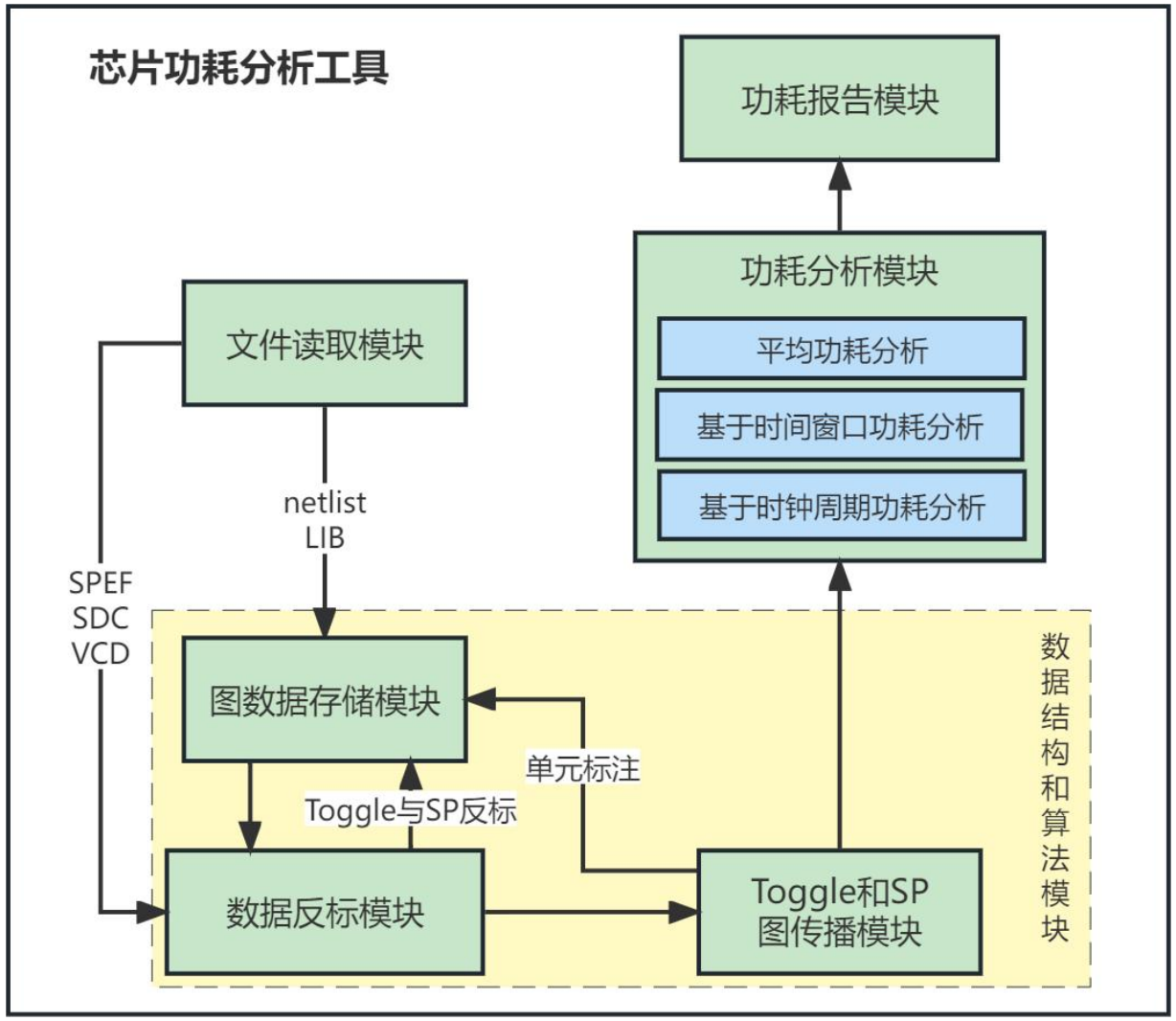
软件整体架构



软件分层为三层模型，包括数据访问层，业务逻辑层，用户交互层

- **数据访问层**：读取数据源层的数据文件。数据层的数据来源有一部分是从时序分析模块获取，可以使用已有的parser，功耗重点在VCD的读取。
- **业务逻辑层**：开发的重点，实现功耗分析的目的。业务逻辑层根据功耗计算算法对各类功耗进行计算。
- **用户交互层**：提供C++和Python的统一接口（PYBIND）。

模块设计



- 图数据存储模块是把所有数据源读取的数据抽象到一个电路图上面；
- 算法模块包括翻转率（Toggle）和静态概率（SP）的在抽象的电路图上传播；
- 功耗分析模块包含三种模式的功耗分析。平均功耗分析是基于整个仿真时间内的平均翻转率和SP来计算功耗，而基于时间窗口或时钟周期则粒度更细，可以计算峰值功耗。

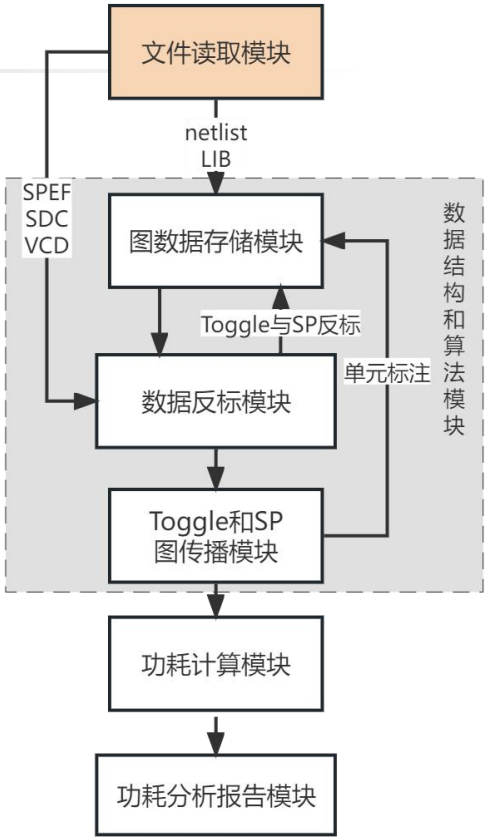
解析VCD文件

当前特性：

- Flex & Bison 解析与构造VCD文件内容
- 按照SAIF文件格式存储

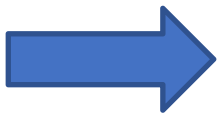
后续优化计划：

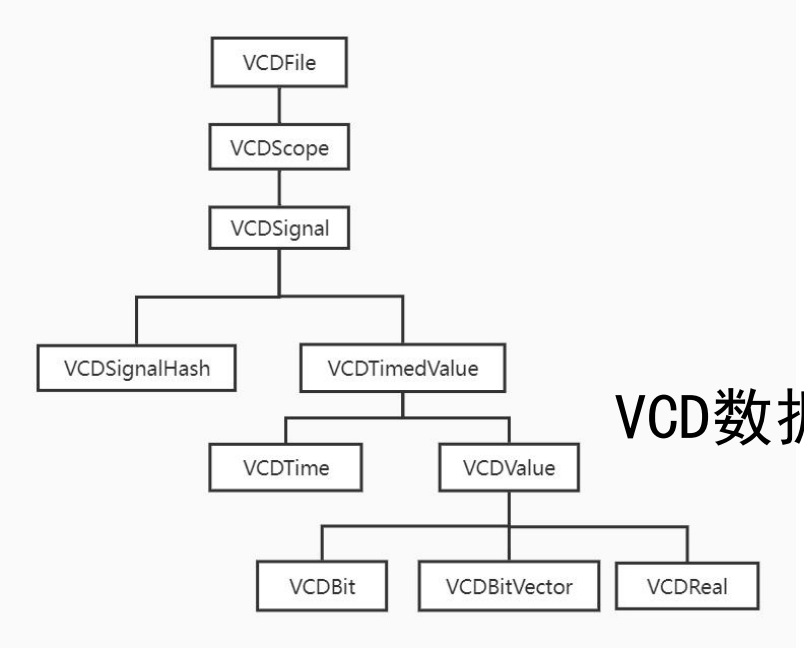
- Glich处理
- 重写VCD Parser
- 支持SAIF文件格式的波形信息文件




```
value_change_dump_definitions ::=
        { declaration_command } { simulation_command }
declaration_command ::=
        declaration_keyword
        [ command_text ]
        $end
simulation_command ::=
          simulation_keyword { value_change } $end
        | $comment [ comment_text ] $end
        | simulation_time
        | value_change
declaration_keyword ::=
          $comment | $date | $enddefinitions | $scope | $timescale | $upscope
        | $var | $version
simulation_keyword ::=
        $dumpall | $dumpoff | $dumpon | $dumpvars
simulation_time ::=
        # decimal_number
value_change ::=
          scalar_value_change
        | vector_value_change
scalar_value_change ::=
        value identifier_code
value ::=
        0 | 1 | x | X | z | Z
vector_value_change ::=
          b binary_number identifier_code
        | B binary_number identifier_code
        | r real_number identifier_code
        | R real_number identifier_code
identifier_code ::=
        { ASCII character }
```

VCD文件格式



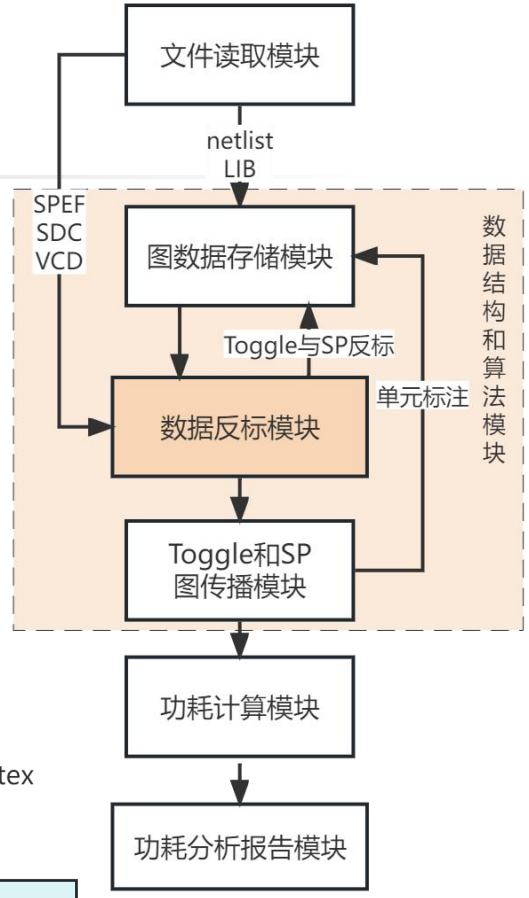


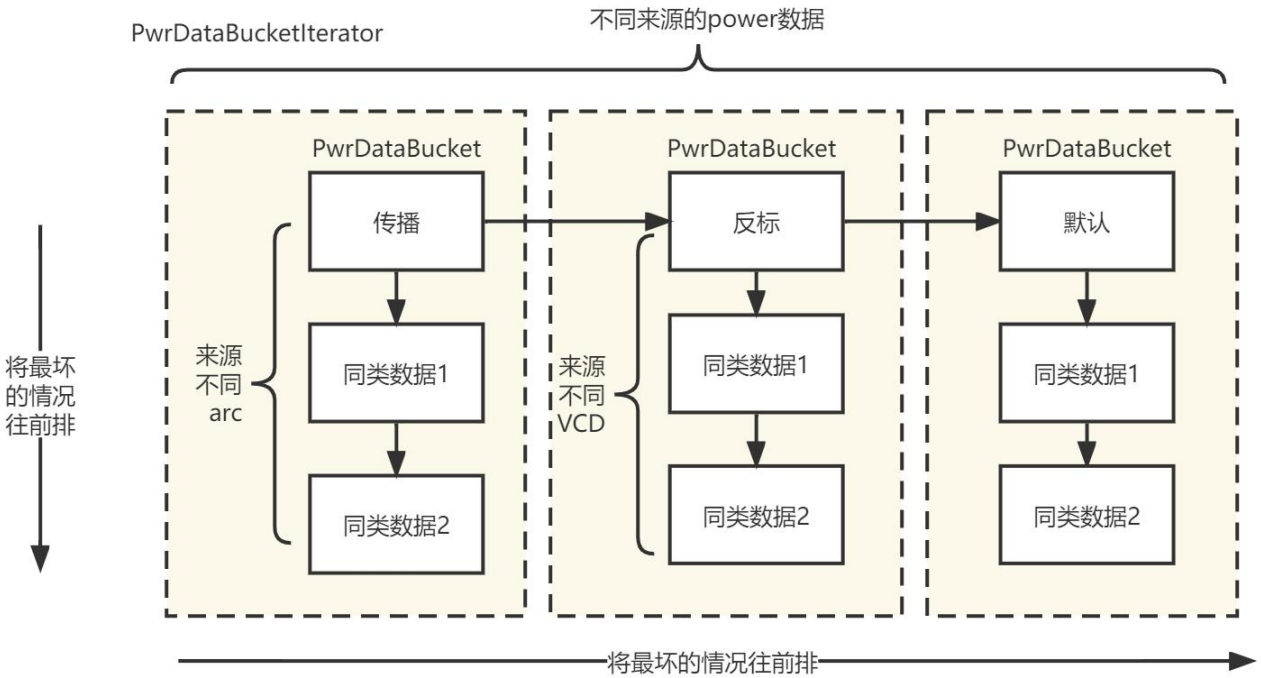
VCD数据结构

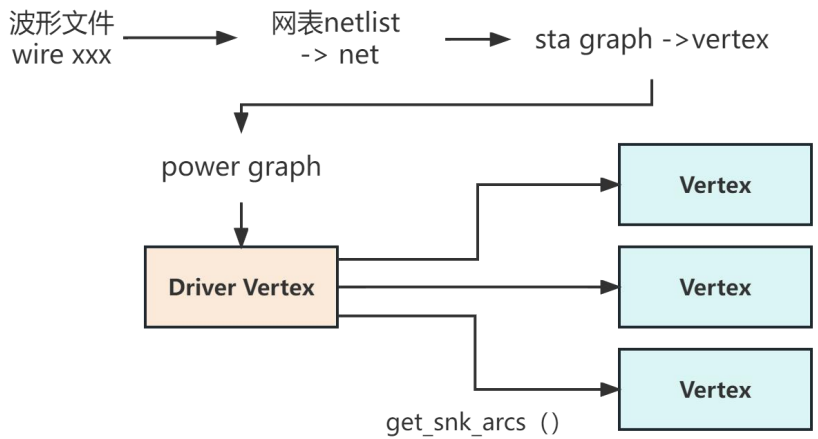
Toggle与SP数据反标

- **Toggle**：信号单位时间内翻转的次数
- **SP**：静态概率（Static Probability），
  信号处于特定逻辑状态0或1的概率

1. 根据duration计算平均SP与Toggle；
2. 将计算结果存入Data Bucket；
3. Data Bucket 与 Power Vertex进行绑定。

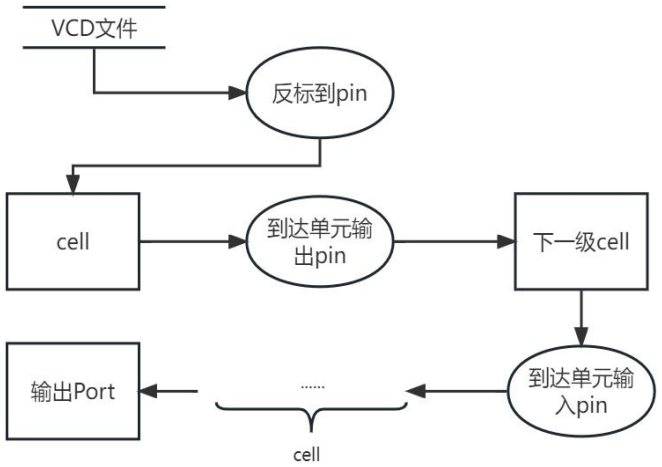




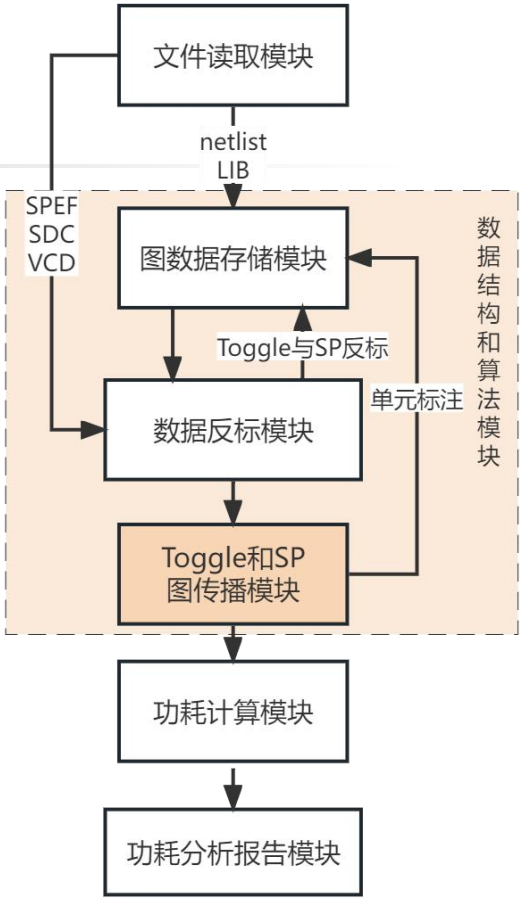


后续计划：引入机器学习模型预测Toggle和SP

Toggle与SP数据传播

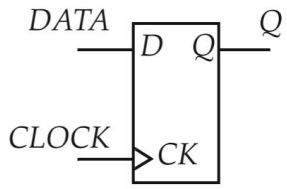


1. 网表顶层模块的主要输入Port上指定默认的全局Toggle和SP；
2. 在通过VCD解析反标过的单元Pin为**起点**，对应Power Graph上的Power Vertex进行**图传播。**

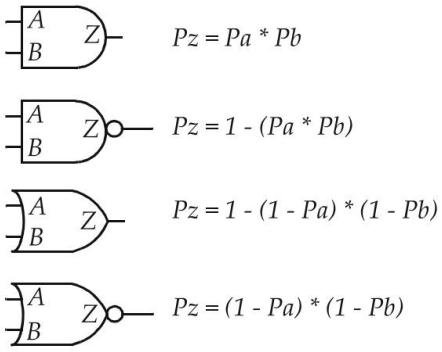


对于不同单元Cell在传播时需要根据单元的功能计算传播的结果。

例1：



对于D型Flip-Flop Q处的翻转率至多是CLOCK的一半



对于组合逻辑单元，其输出Pin的静态概率是根据输入Pin的静态概率值计算

难点：确定从单元输入到输出的**toggle**变换公式

不同功耗类型计算

1 漏电功耗计算

- 工艺库 （.lib） 文件中，获得不同状态下的漏电功耗，并且根据各个输入状态的概率（SP）进行功耗加权平均。

```
leakage_power () {
  value : 42.2;
}
leakage_power () {
  value : 26.1;
  when : "!A1 !A2";
}
leakage_power () {
  value : 33.0;
  when : "!A1 A2";
}
leakage_power () {
  value : 27.0;
  when : "A1 !A2";
}
leakage_power () {
  value : 82.7;
  when : "A1 A2";
}
```

2 开关功耗计算

电容

$$P_{cload} = KfC_L VDD^2$$

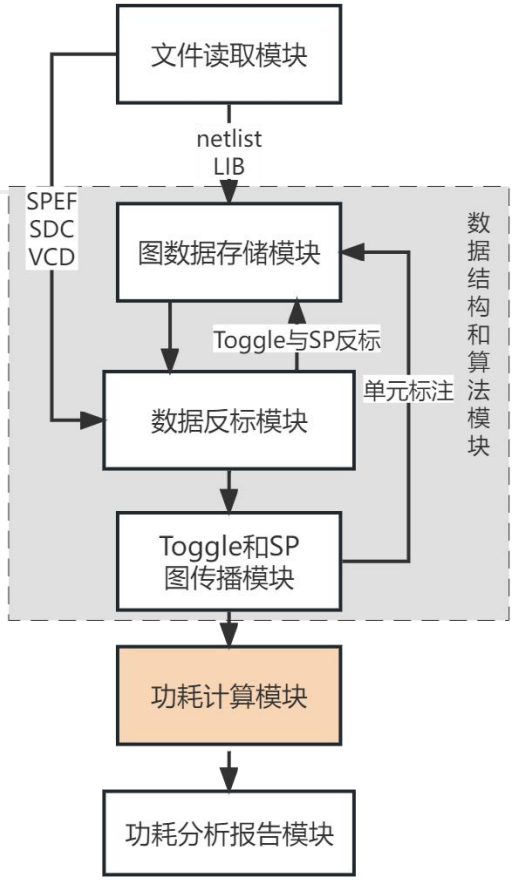
翻转率 供电电压

3 内部功耗计算

```
pin (Z1) {
  . . .
  power_down_function : "!VDD + VSS";
  related_power_pin : VDD;
  related_ground_pin : VSS;
  internal_power () {
    related_pin : "A";
    power (template_2x2) {
      index_1 ("0.05, 0.1"); /* Input transition */
      index_2 ("0.1, 0.25"); /* Output capacitance*/
      values ( /*              0.1          0.25 */ \
      /* 0.05 */              "0.045,      0.050", \
      /* 0.1 */               "0.055,      0.056");
    }
  }
}
```

信号下降（上升）

针对不同单元的不同**pin**，分别计算内部功耗



API与tcl命令

1 部分关键面向开发者的API

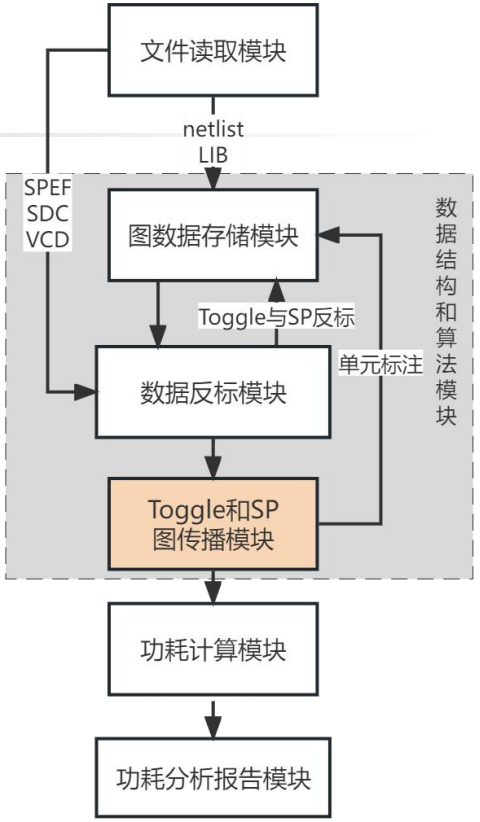
API	功能简述
buildGraph	构建iPW图数据结构
readVCD	解析VCD文件
buildSeqGraph	构建时序单元子图
checkPipelineLoop	检测PipeLine环路
levelizeSeqGraph	对时序单元子图进行分级
propagateToggleSP	在图上传播Toggle与SP数据
calcLeakagePower	计算漏电功耗
calcInternalPower	计算内部功耗
calcSwitchPower	计算开关功耗
analyzeGroupPower	分析功耗数据
reportPower	输出功耗报告

API与tcl命令

2 面向用户的tcl命令

TCL命令	功能说明	参数	定义
read_verilog	读入netlist文件	netlist文件名	read_verilog asic_top.v
link_design	对读入的netlist构建网表	顶层名	link_design asic_top
read_liberty	载入liberty	lib文件名	read_liberty $LIB_FILE
read_sdc	读入sdc	sdc文件名	read_sdc asic_top.sdc
read_spef	读入spef	spef文件名	read_spef asic_top.spef
report_timing	分析和报告timing	无	无
read_vcd	读入vcd	vcd文件名 -top_name vcd顶层instance，设置解析范围	read_vcd test.vcd -top_name top-i
report_power	分析和报告power	无	无

LTD图遍历算法



问题描述：寄存器之间可达路径数量的问题（时序路径搜索）

问题难点：图规模大，分叉路径多

寄存器fanout遍历新方法，即LTD算法（Latch based on ordered Task thread pool DFS algorithm）
LTD算法在时间与空间上具有综合优势

算法的时间与空间复杂度均为O(m+n)

1 DFS算法任务

1. 寄存器的时钟节点作为起点节点（Start Vertex）；
2. 每个节点使用B树实现的Set作为并查集来存储和查询；
3. 直到遇到下一个寄存器数据输入节点（End Vertex）则传播路径结束；

优势

- 充分利用多线程的优势
- 时间复杂度最优；
- 用于Set使用B树实现。

LTD图遍历算法

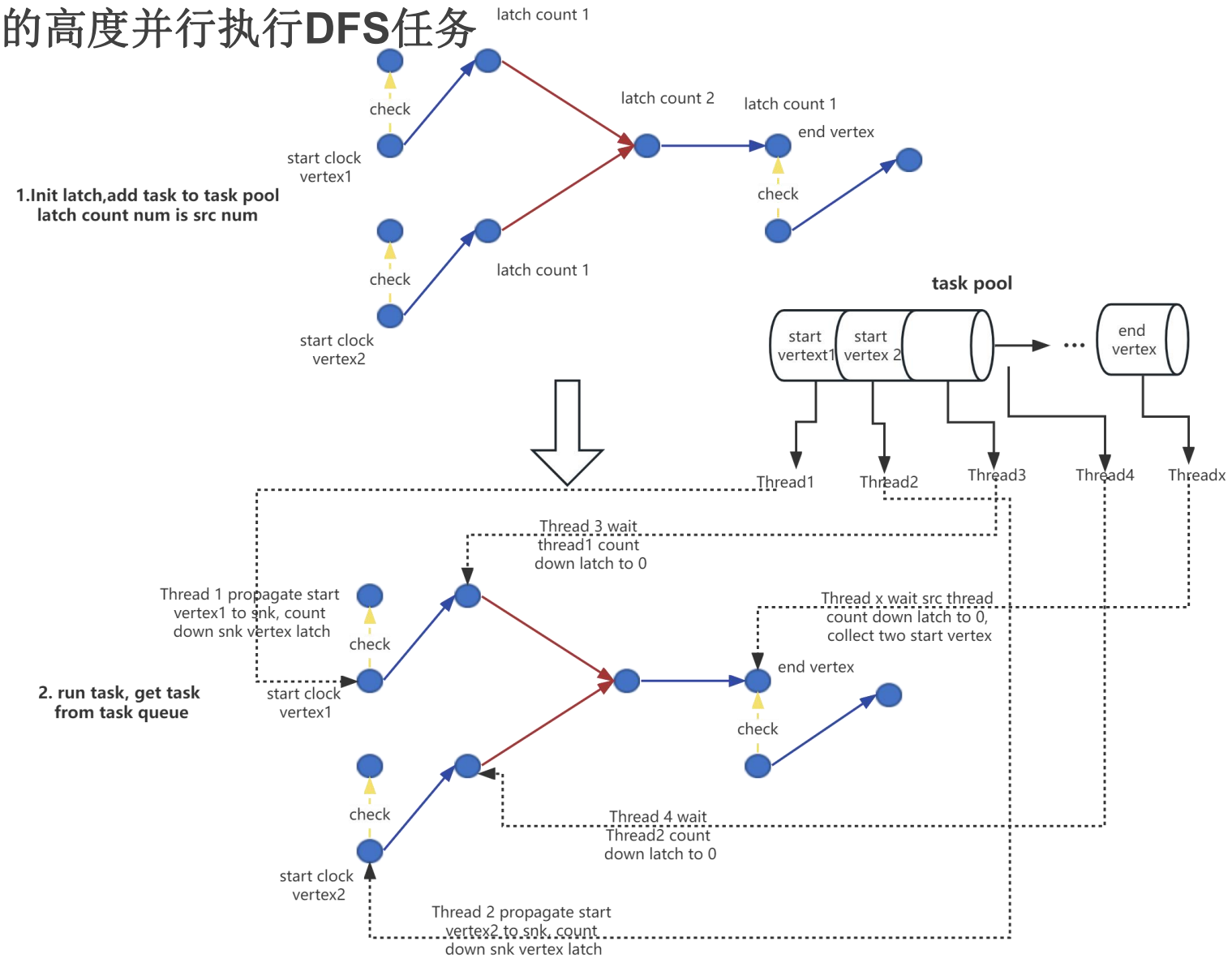
2 使用基于c++20特性的std::latch的高度并行执行DFS任务

1. 分配任务
   1. 初始化latch pool
   2. 为所有vertex分配latch

2. 排序任务

   按序放入线程池，保证顺序较优，减少后续任务等待

3. 执行任务
   1. latch为0的起始节点的线程先执行
   2. 往后传播时，把传播到本节点的起始节点放入后继节点并查集B树中，后继节点latch减1
   3. latch减为0时，该节点对应的线程任务开始执行
   4. 最终传播到end vertex



LTD图遍历算法

3 后续优化设想

1. 使用机器学习对线程池Task排序

特点：

- 每个节点的传播视为一个task
- 每个task的执行时间基本一致
- 节点之间存在依赖关系
- 后继节点线程阻塞时间尽可能减少

排出最优序列 → 无锁运行

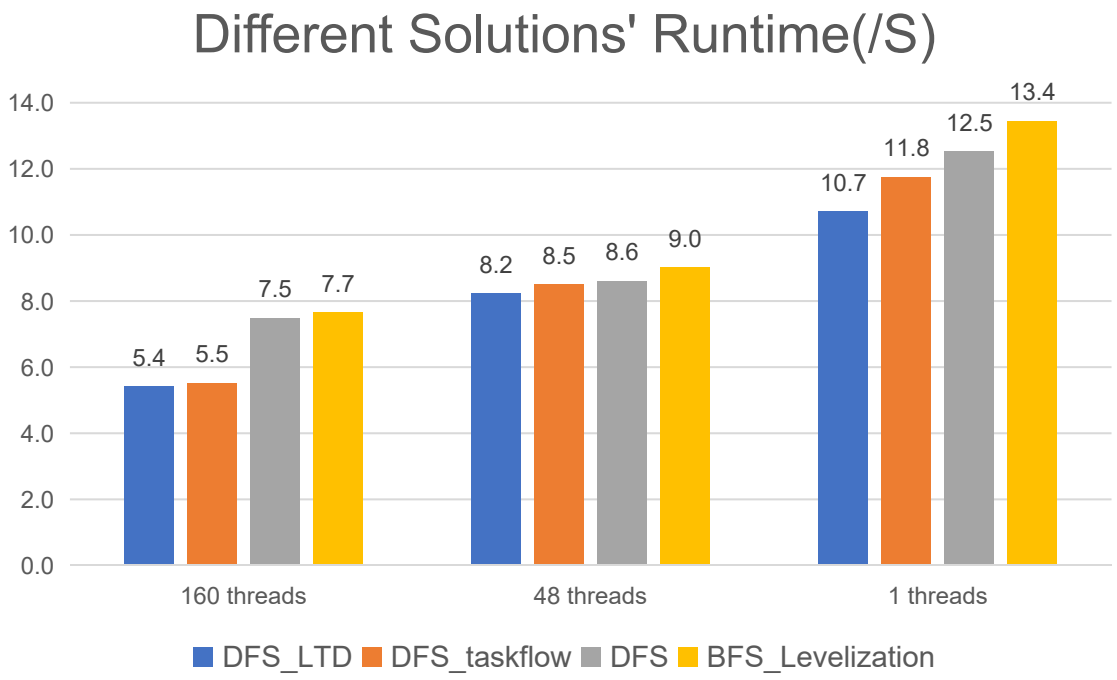
2. 使用机器学习提前预测功耗所需数据

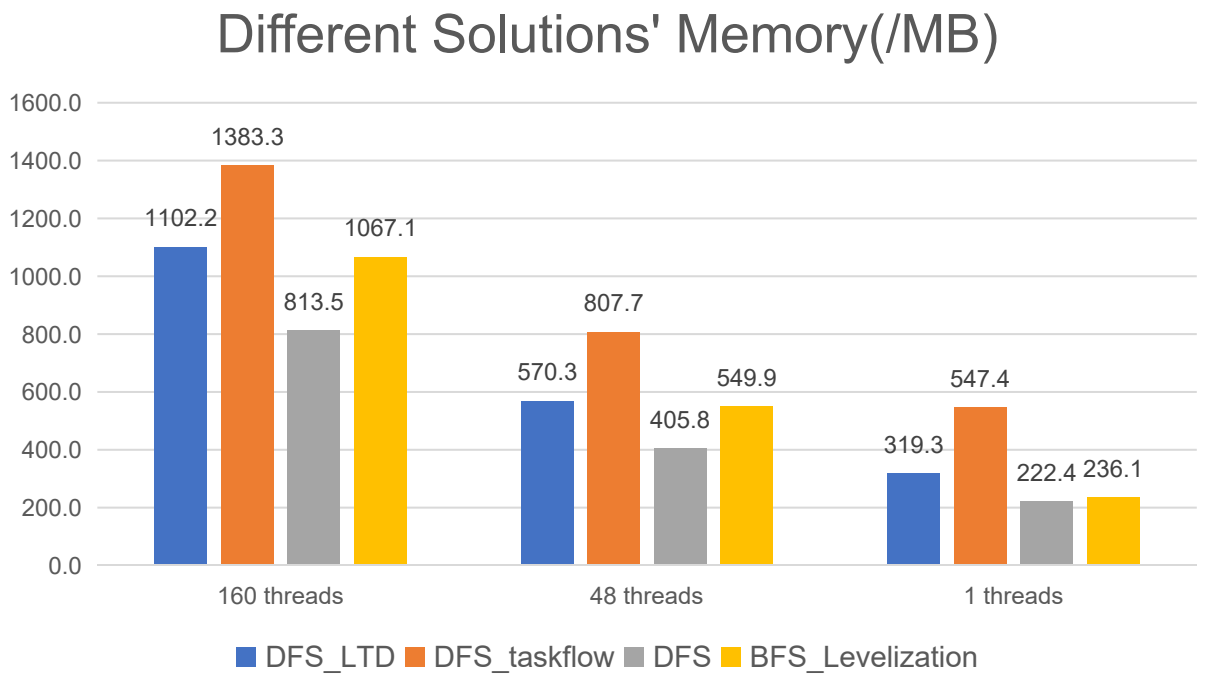
- 使用精确算法结果作为标签
- 特征：根据DFS遍历到已有的组合逻辑单元类型和其后继节点数量等

LTD图遍历算法

4 性能测试

基于台积电28nm工艺的一生一芯RISC-V芯片作为测试例子，芯片规模150万门。





Glitch 分析

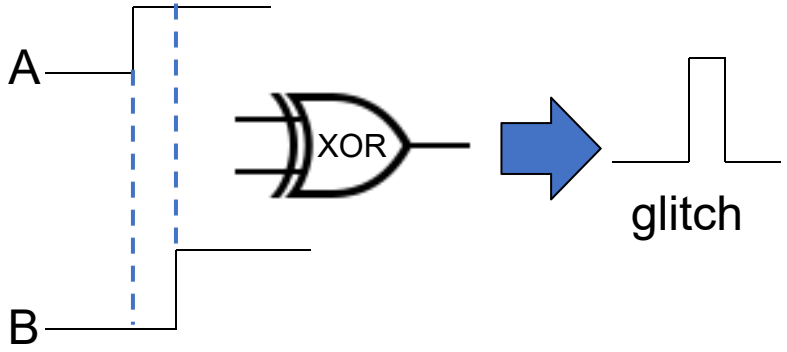
分析步骤：

1 识别glitch

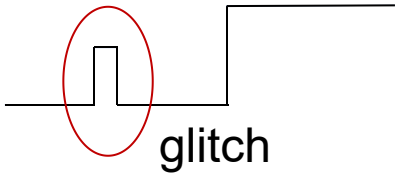
- *Transport Glitch*： cell输出在到达稳定状态前的无效翻转，比如门电路输入延时不一致造成输出的glitch。
- *Inertial Glitch*：cell 输入翻转由于宽度小于cell delay 未传到cell 输出，是个脉冲或者亚稳态信号，最后会被吸收掉。

Input		Output
A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Transport Glitch 举例



Inertial Glitch 举例



2 统计glitch的toggle计数

把glitch的toggle计数从正常的翻转计数中减去，单独统计glitch power

3 分析power

分析计算glitch部分的power，并report出来指导设计。

电压降 IRDrop 分析

大量信号同时发生翻转

电源电压VDD发生跌落

产生IRDrop

影响芯片时序以及功能等

1. 计算功耗
   - 平均功耗 -> 静态IRDrop
   - Cycle功耗 -> 动态IRDrop （更精确）
2. 根据功耗计算电流
3. 根据电流，电源网络电阻，计算节点电压，得到IRDrop。

$$GV = J$$

AI预测RTL功耗、IRDrop

1 预测Toggle/SP

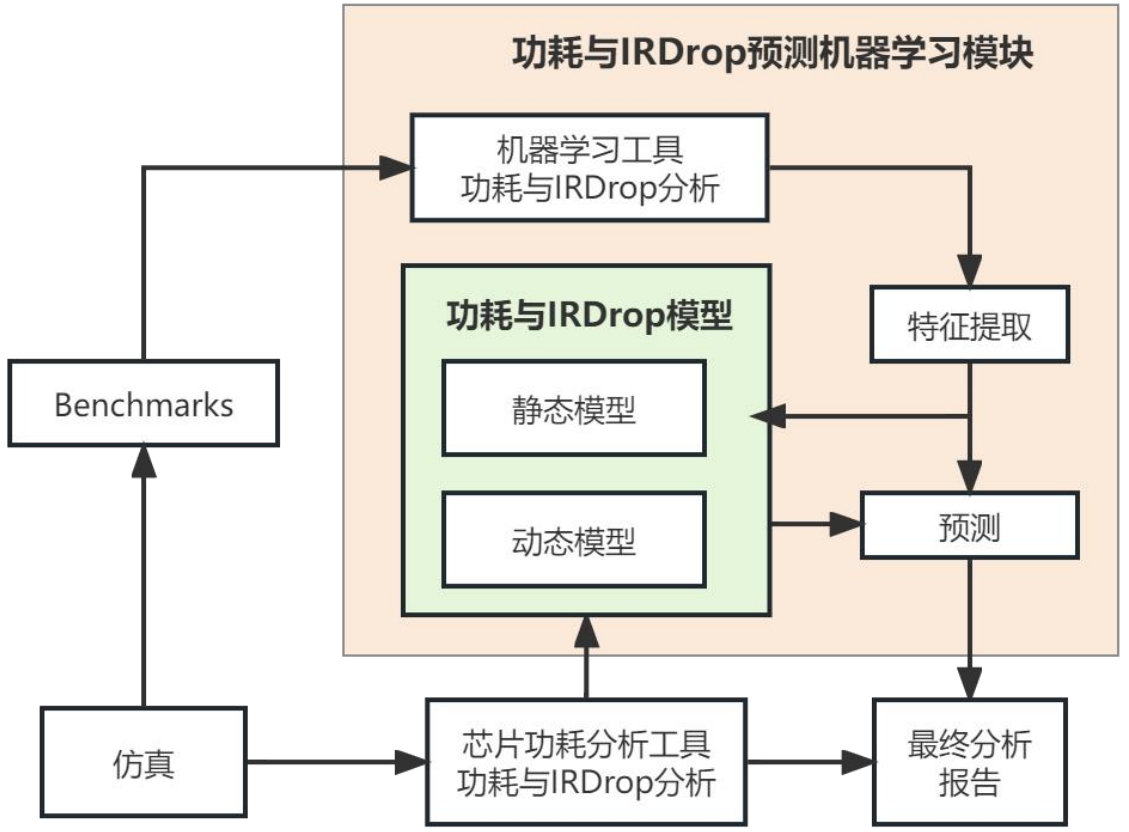
方法：波形采样——通过小样本波形预测toggle

2 预测RTL功耗

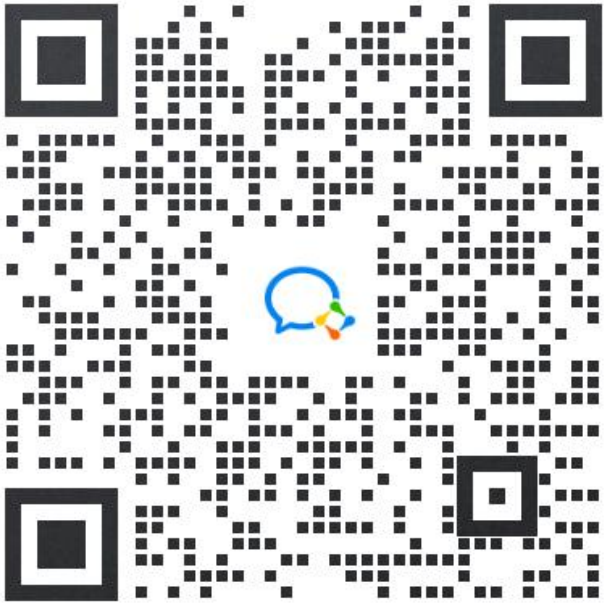
在RTL阶段，根据芯片设计的架构特征，对信号Toggle/SP建模，预测芯片RTL功耗。

3 预测IRDrop

在物理设计阶段，抽取电源网络特征、功耗密度特征、节点到电源的线长等特征使用CNN或者GNN预测IRDrop



iEDA开源交流群



感谢聆听

Thanks for your attention

陶思敏
taosm@pcl.ac.cn