Tutorial 7 - Part 6
iPA: Power Analysis Tool and Its Technologies

Simin Tao[1], Zheqing Shao[2], Xingquan Li[1]

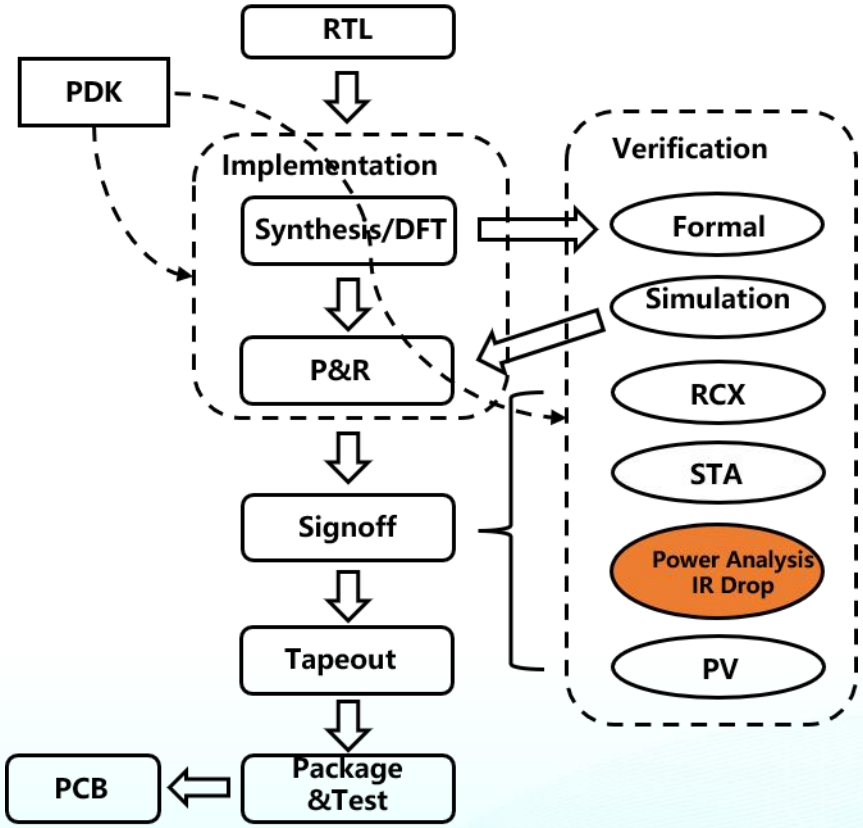
[1]Peng Cheng Laboratory;
[2]University Science and Technology of China;

iEDA Tutorial Agenda

- **Part 0:** iEDA Overview **(Xingquan Li)**
- **Part 1:** iEDA Infrastructure **(Zengrong Huang)**
- **Part 2:** iPL: Placement Tool and Its Technology **(Shijian Chen)**
- **Part 3:** iCTS: Clock Tree Synthesis Tool and Its Technologies **(Weiguo Li)**
- **Part 4:** iRT: Routing Tool and Its Technologies **(Zhisheng Zeng)**
- **Part 5:** iSTA: Static Timing Analysis Tool and Its Technologies **(Simin Tao/He Liu)**
- **Part 6: iPA: Power Analysis Tool and Its Technologies (Simin Tao)**

Background

- Power analysis importance and usage
  - ✓ Local overheating of chips can lead to chip failure
  - ✓ Chip packaging, heat dissipation
  - ✓ Power delivery network design



Background

- Existing popular open-source PA tools
  - OpenSTA
    - ✓ Point tool of openRoad Project
    - ✓ Support VCD anotate and Toggle/SP Propagation、analyze leakge、internal、switch power
  - OpenTimer
    - ✓ Analyze leakage power
    - ✓ Doesn’t support more functions yet

https://github.com/The-OpenROAD-Project/OpenSTA.git
https://github.com/OpenTimer/OpenTimer.git

Background

- iPA concerns
  - ✓ iPA is a power analysis tool in the open-source EDA toolchain **iEDA**. iEDA is dedicated to creating an open-source EDA foundation
  - ✓ The goal of iPA is to build a **modular, easily scalable, highly readable, and feature-rich** power analysis tool

TABLE 1: Features comparison among iPA, OpenTimer and OpenSTA.

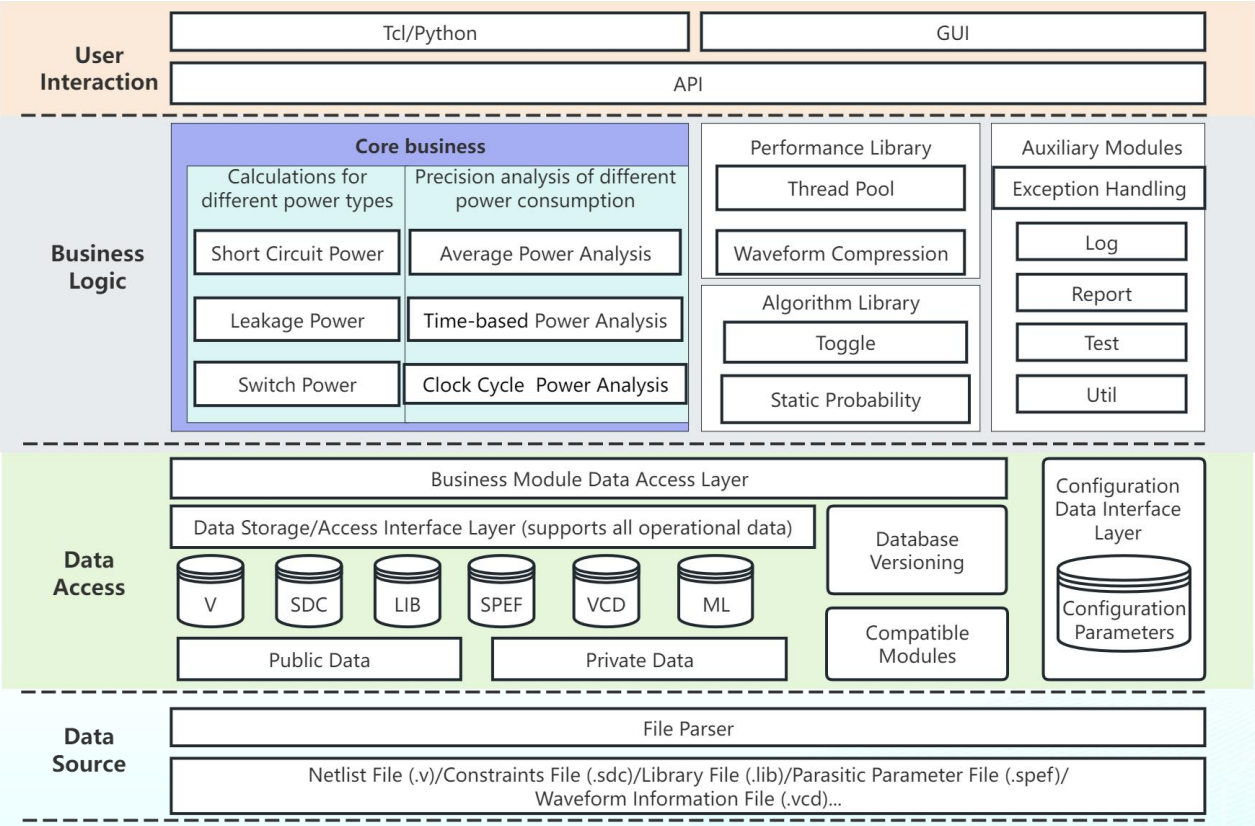
Feature	iSTA	OpenTimer	OpenSTA
leakage power analysis	✓	✓	✓
VCD load and annotate	✓	×	✓
Toggle/SP propagation	✓	×	✓
dynamic power analysis	✓	×	✓
IR drop analysis	✓	×	✓
EM analysis	×	×	✓
power analysis based clock cycle	×	×	×
low power analysis based CPF/UPF	×	×	×

Background

- iPA contribution
  - ✓ Common **power analysis and IR drop**, construct a set of functions and algorithms
  - ✓ **Power data and operators are separated**, can be integrated as a **power engine** into other tools, or run as a **standalone** sign-off tool
  - ✓ **Mixing C++20 and Rust** programming
  - ✓ By using **C++20’ s Latch** the parallelism is improved.

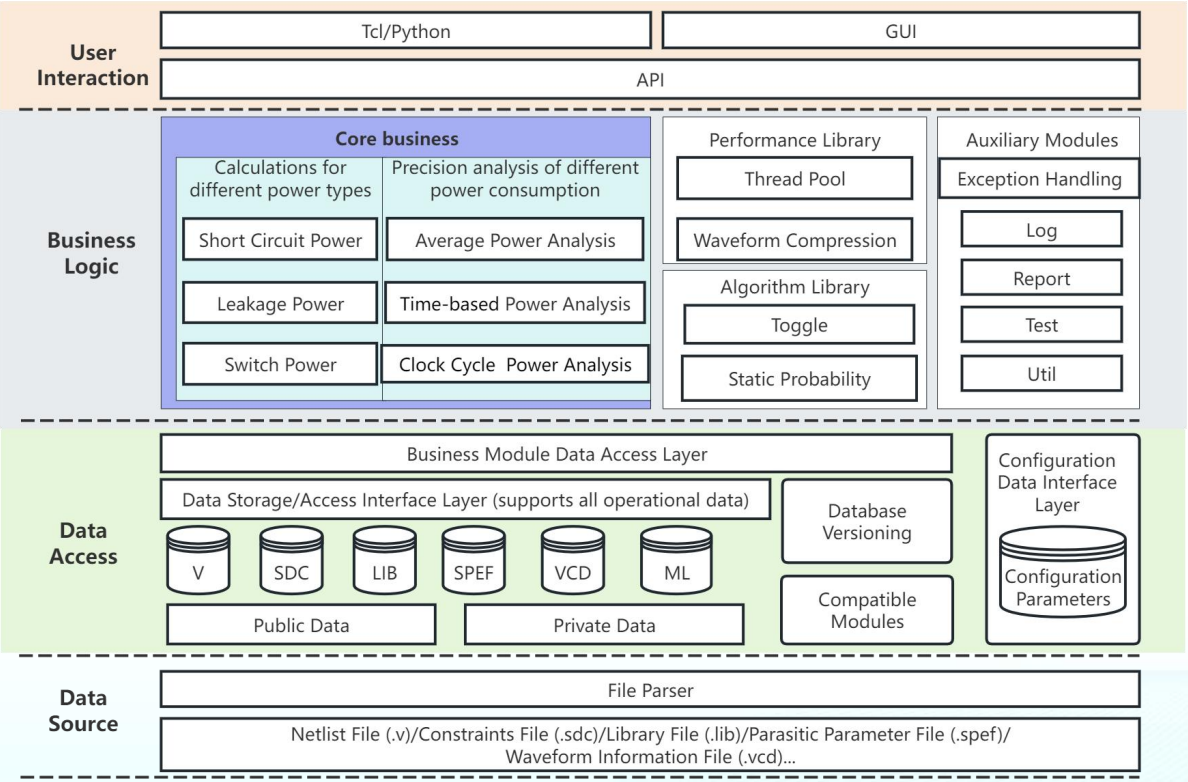
iPA structure

- iPA structure
  - ✓ **Three-layer** architecture
  - ✓ **Fully decoupled, pluggable, and easily scalable**



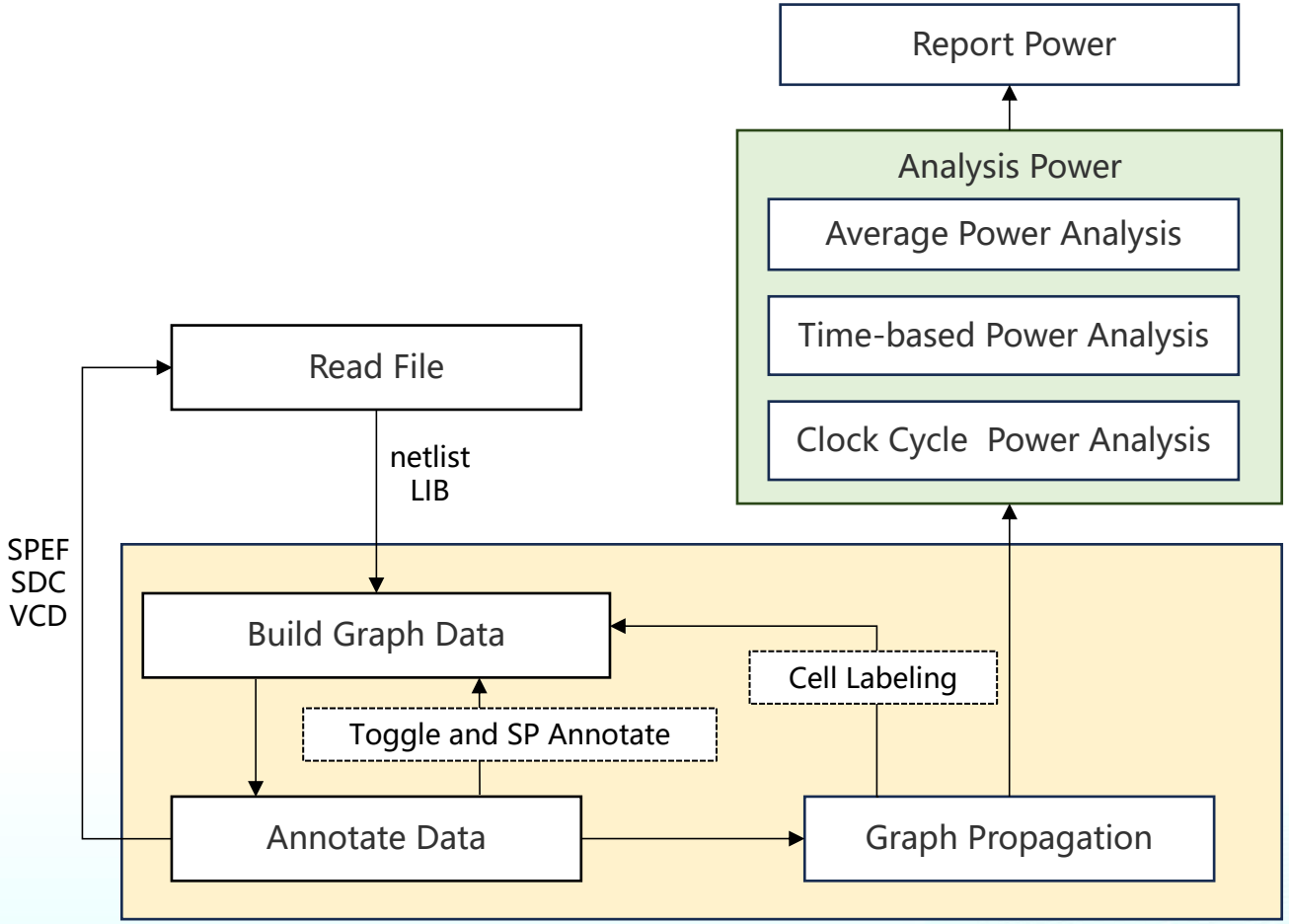
iPA structure

- Three Layer
  - ✓ **Data Access Layer**: interact with the data source
  - ✓ **Business Logic Layer**: implements the algorithms and calculations
  - ✓ **Interaction Layer**: providing a unified interface for both C++、TCL and Python applications



iPA flow

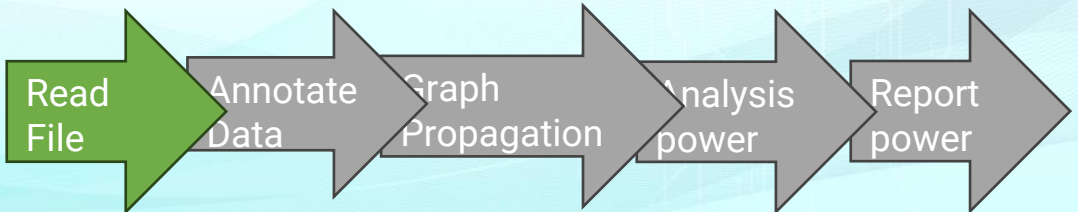
- iPA flow
  - ✓ Read File, build graph data
  - ✓ Annotate data
  - ✓ Graph propagation
  - ✓ Analysis power
  - ✓ Report power



Read File

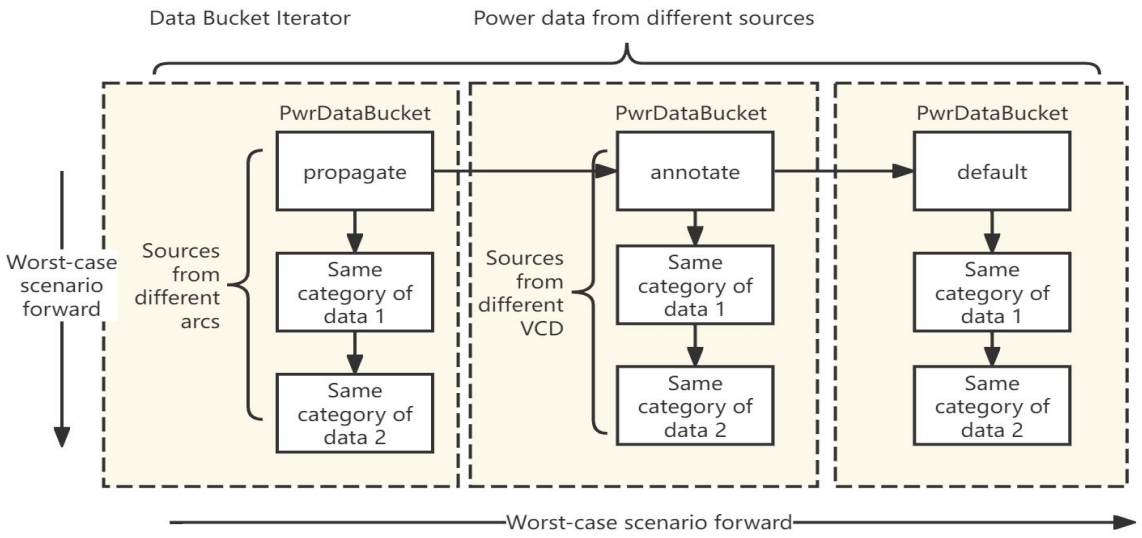
- VCD Load and Annotate
  - ✓ Read file(.v,.lib,.spef,.vcd) use Rust parser
  - ✓ Store database
  - ✓ Build power graph

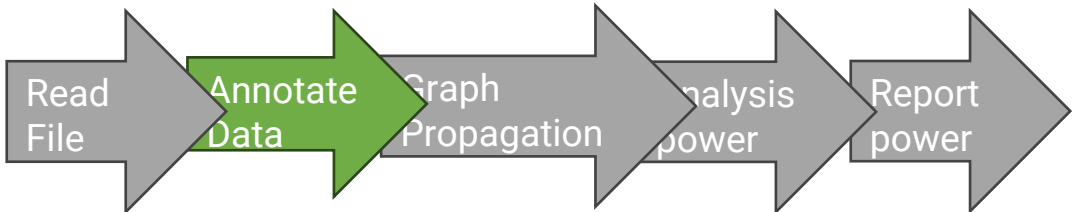
```
$date June 26, 1989 10:05:41
$end
$version VERILOG-SIMULATOR 1.0a
$end
$timescale 1 ns
$end
$scope module top $end
$scope module m1  $end
$var trireg 1 *@ net1 $end
$var trireg 1 *# net2 $end
$var trireg 1 *$ net3 $end
$upscope $end
$scope task t1 $end
$var reg 32 (k accumulator[31:0] $end
$var integer 32 {2 index  $end
$upscope $end
$upscope $end
$enddefinitions $end
$comment
  $dumpvars was executed at time '#500'.
  All initial values are dumped at this time.
$end
```



VCD Annotate

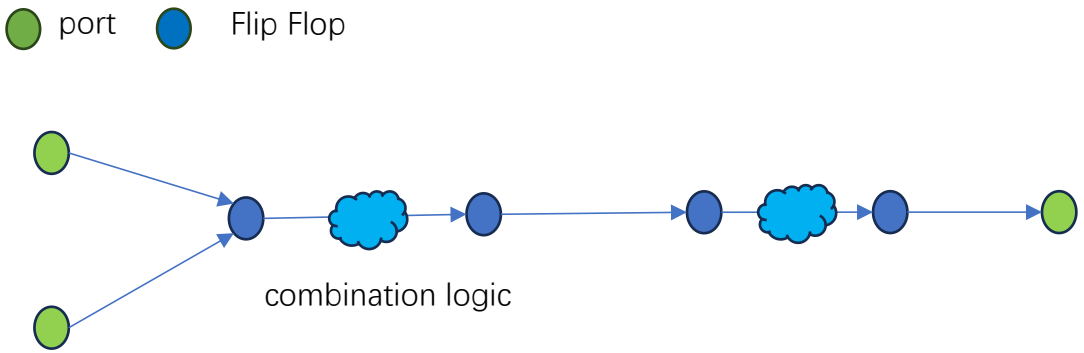
- Annotate Data
  - ✓ Annotate toggle/SP to power graph
  - ✓ different types of Toggle/SP data will be stored separately and stored in a linked list.

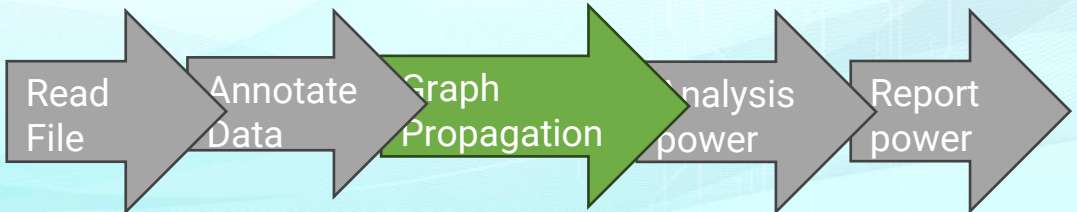




Toggle/SP Propagation

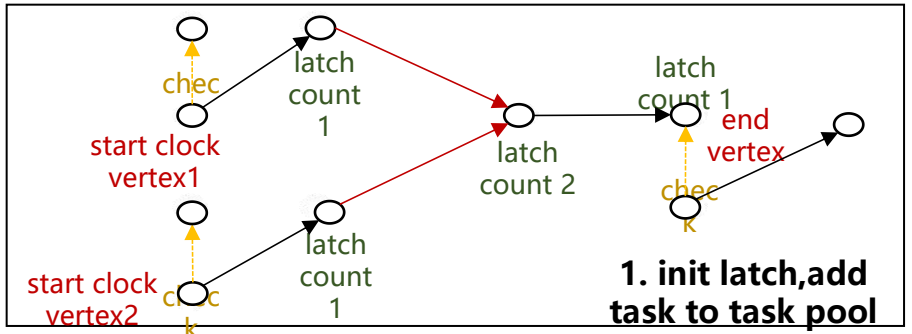
- Build Sequential Graph
  - ✓ Traverse the start vertex on the power graph
  - ✓ Building the power sequential vertex corresponding to the sequential cell
  - ✓ Constructs the port as sequential vertex
  - ✓ Find the successor power sequential vertex of the node, and build the sequential arc



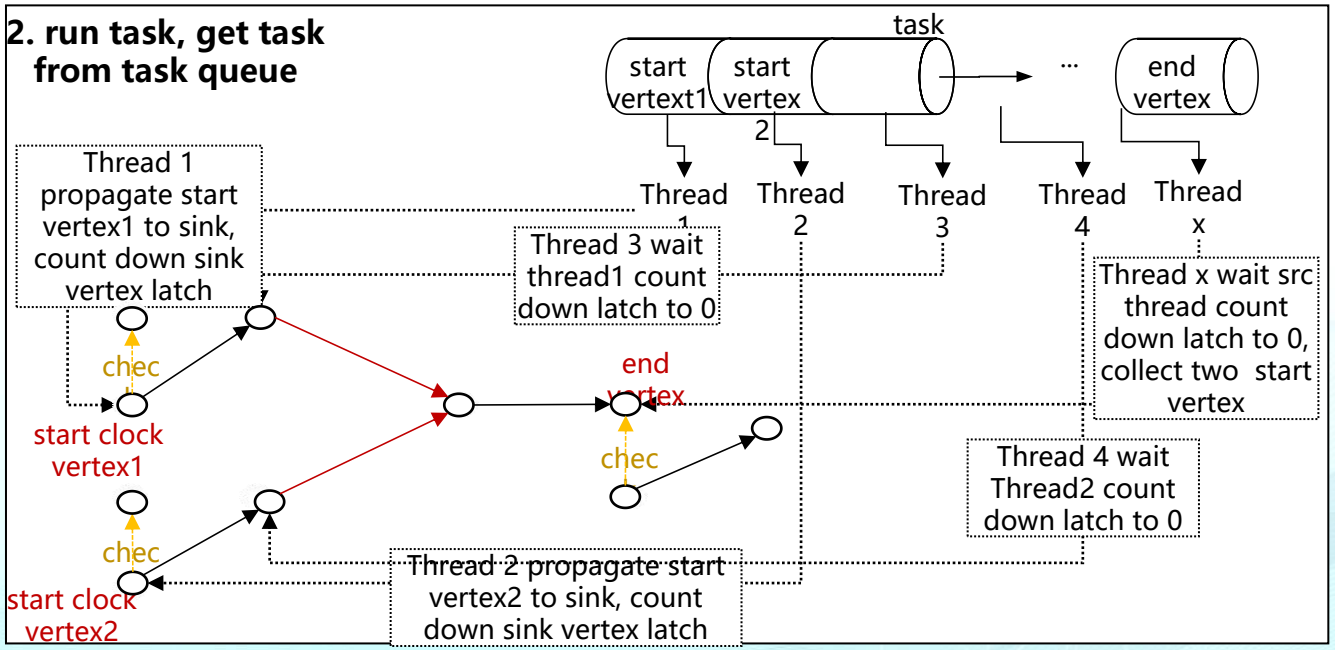


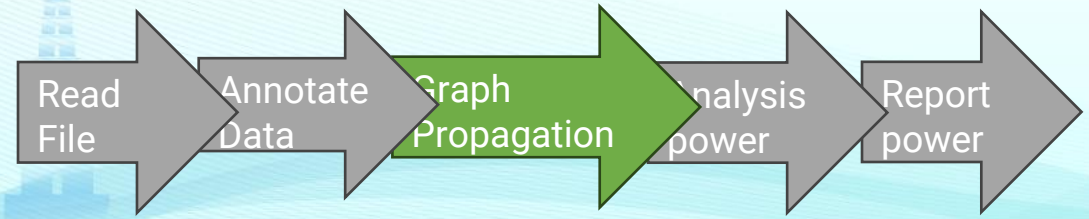
Toggle/SP Propagation

- LTD(Latch based on ordered Task thread pool DFS)
  - ✓ iPA has proposed a new method for traversing the sequential cell fanout, namely the LTD algorithm



1. init latch,add task to task pool latch count num is src num



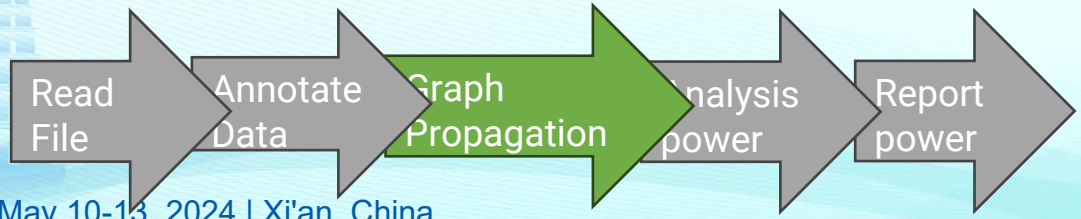


Toggle/SP Propagation

- Loop Detect in Sequential Graph
  - ✓ Uses the tricolor marking method to detect pipeline loops
  - ✓ The tricolor mark will be set on each power sequential vertex.White is not been visited.Gray means the current node and its predecessors are **being** visited.Black means the current node and its predecessors have all **been visited**

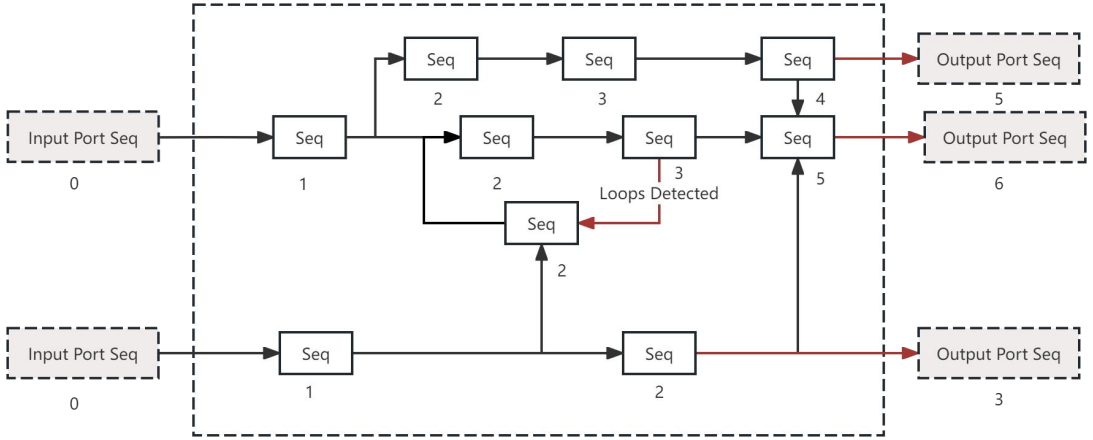
Algorithm 3 Loop Detection Algorithm

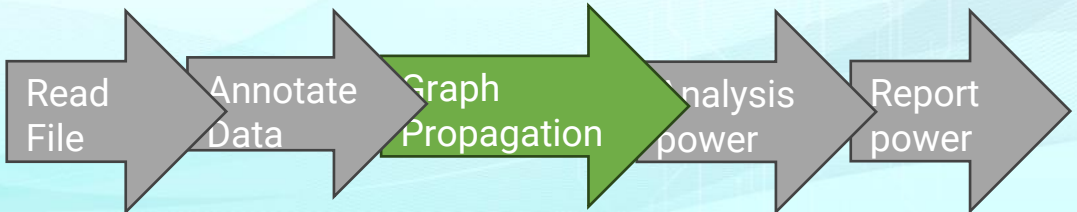
```
Input: sequential graph
Output: vertexes in loop path
function LoopDetection(G)
    Find all Output Power Sequence Vertices V_out in Power Sequence Graph G
    for v ∈ V_out do
        MarkAndVisit(G, v)
    end for
end function
function MarkAndVisit(G, v)
    Mark v as gray (being visited)
    for each predecessor u of v do
        if u is marked black (visited) then
            Continue
        else if u is marked white (unvisited) then
            if u is an Input Port then
                Continue                ▷ Propagation path ends
            else
                MarkAndVisit(G, u)
            end if
        else if u is marked gray (being visited) then
            Mark edge (u, v) as Pipeline Loop
        end if
    end for
    Mark v as black (visited)
end function
```



Toggle/SP Propagation

- Levelization in Sequential Graph
  - ✓ Start from the output port and annotate forward through backtracking
  - ✓ When propagating to an Input Port, annotated as Level 0
  - ✓ During backtracking, the **larger Level** need to be selected for re-annotation



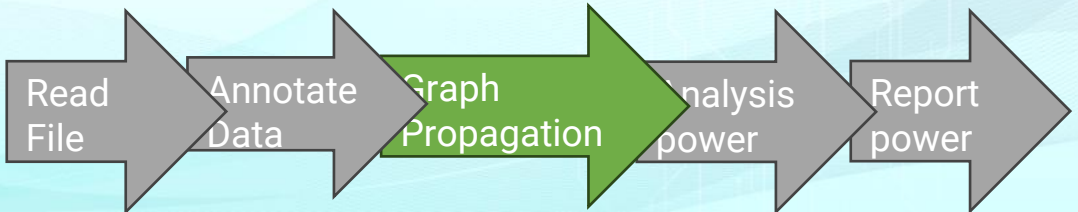


Toggle/SP Propagation

- Clock Propagation
  - ✓ The clock propagation starts from clock nodes, and the nodes traversed are annotated with the default clock toggle and SP
  - ✓ generally sets the **clock toggle to 2 and the clock SP to 0.5**

Algorithm 4 Clock Path Propagation

```
procedure CLOCKPATHPROPAGATION(G, Toggle_clk, SP_clk)
    V_clk ← Get all clock nodes from power graph G
    for each clock node v ∈ V_clk do
        MarkAndVisit(G, v, Toggle_clk, SP_clk)
    end for
end procedure
function MARKANDVISIT(G, v, Toggle_clk, SP_clk)
    Mark v as clock network node
    Annotate v with Toggle_clk and SP_clk
    for each unvisited neighbor u of v in G do
        if u is not a clock node of a sequential cell then
            MarkAndVisit(G, u, Toggle_clk, SP_clk)
        end if
    end for
end function
```

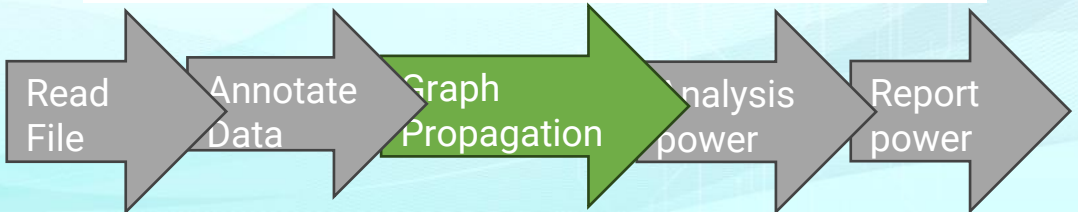


Toggle/SP Propagation

- Const Propagation
  - ✓ Data path propagation is divided into constant and non-constant data propagation
  - ✓ Constant needs to do before non-constant
  - ✓ Constant data propagation starts from the vertex initially annotated as constant nodes
  - ✓ The constant nodes is marked that **SP is 0 or 1, and Toggle is 0**

Algorithm 4 Constant Data Propagation

```
Input: power graph G
Output: const vertexes
procedure CONSTANTDATAPROPAGATION(G)
    Q ← Initialize priority queue
    Mark all initially constant PwrVertices in G
    Mark corresponding PwrSeqVertices as constant
    Add all constant PwrVertices to Q sorted by level
    while Q is not empty do
        v ← Dequeue vertex from Q
        u ← PwrSeqVertex containing v
        Propagate constants from u until Output Pin
        if Output Pin Instance has constant marking then
            Calculate Toggle and SP from Datain to Dataout
            for each Dataout node w do
                Calculate Toggle and SP of w using PWR-CALCTOGGLESP(w)
                if w is a constant node then
                    Add w to Q sorted by level
                end if
            end for
        end if
    end while
end procedure
```

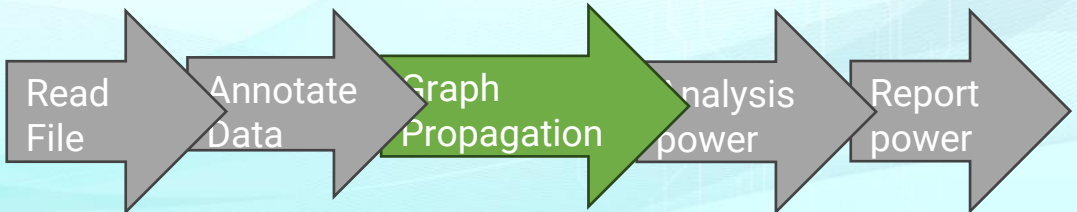


Toggle/SP Propagation

- Data Propagation
  - ✓ The process of non-constant data propagation is similar to constant propagation,starts from nodes with smaller levels
  - ✓ For the current level of data propagation, starting from the **data input** of the sequential cell propagated backward using DFS until the **output** of the previous level sequential cell
  - ✓ When backtracking, calculate the Toggle/SP of each cell

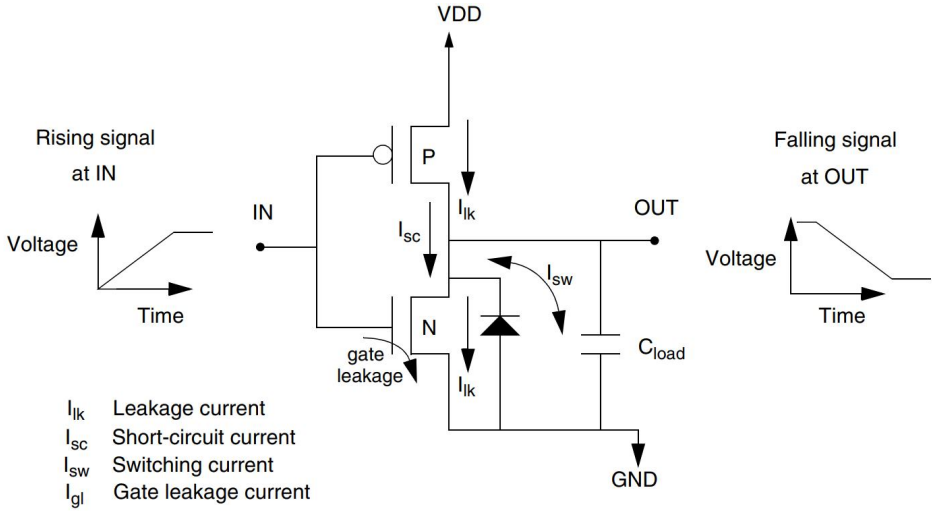
Algorithm 5 Non-Constant Data Propagation

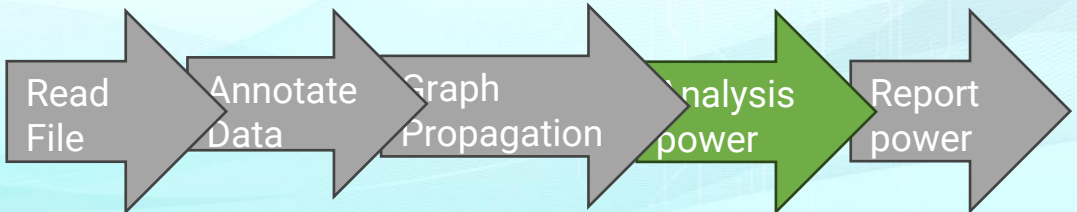
```
Input: power graph G
Output: non-const vertexes Toggle/SP
procedure NonConstantDataPropagation(G)
    Sort all levels in G in ascending order
    for each level L do
        for each timing path T in L do
            v ← data input of sequential cell at the endpoint of T
            MarkAndVisit(G, v, T)
        end for
    end for
end procedure
function MarkAndVisit(G, v, T)
    if v is constant-marked then
        return          ▷ Propagation along this path terminates
    end if
    Calculate Toggle/SP of v using sequential cell method
    for each predecessor u of v along T do
        MarkAndVisit(G, u, T)
    end for
end function
```



Power Calculation and Analysis

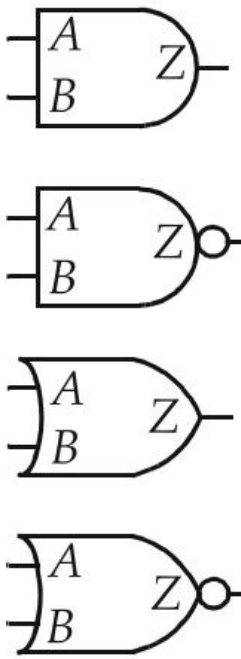
- Leakage Power
  - ✓ Leakage power refers to the power caused by the **leakage current of transistors** in CMOS circuits
  - ✓ When a CMOS circuit is in a static state, the leakage current of transistors will lead to energy loss, resulting in leakage power





Power Calculation and Analysis

- Cell Power - Leakage power
  - ✓ Static power under different states can be obtained from the technology library (.lib) file, and the **power weighted average can be calculated based on the probability of each input state**
  - ✓ If the states of different pins have complex correlations, **empirical formulas** can be obtained through simulation

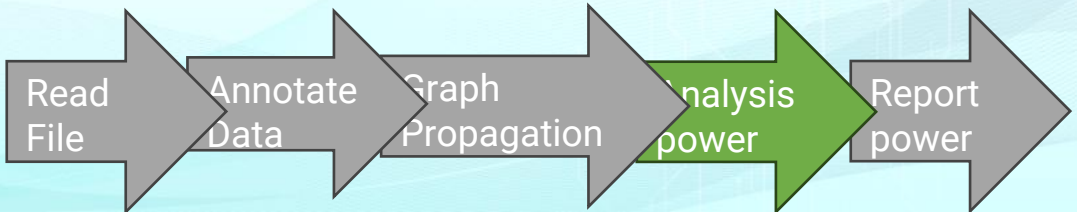


$Pz = Pa * Pb$

$Pz = 1 - (Pa * Pb)$

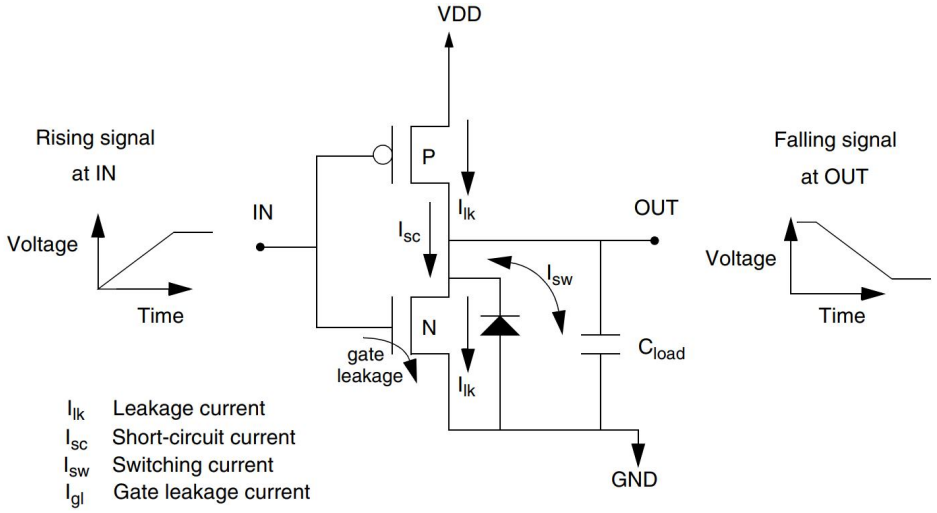
$Pz = 1 - (1 - Pa) * (1 - Pb)$

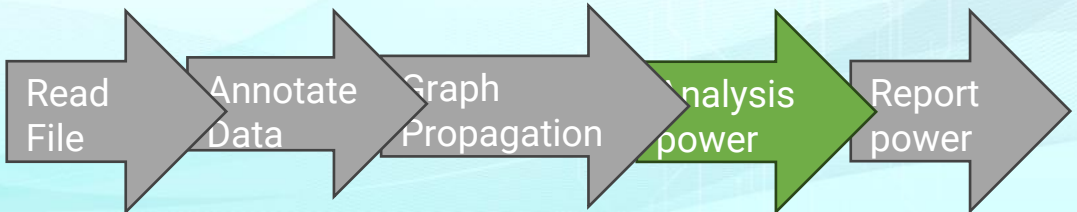
$Pz = (1 - Pa) * (1 - Pb)$



Power Calculation and Analysis

- Short-circuit power
  - ✓ Short-circuit power refers to the power caused by short-circuit current of transistors in CMOS circuits
  - ✓ When a CMOS circuit switches, **due to the delay in the opening and closing of transistors**, leading to short-circuit power



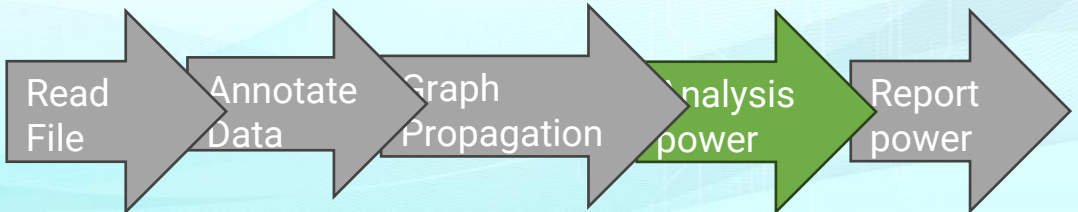


Power Calculation and Analysis

- Cell power - Internal power
  - ✓ Internal power is divided into rise power and fall power, with taking their average value (average power) combination with the toggle

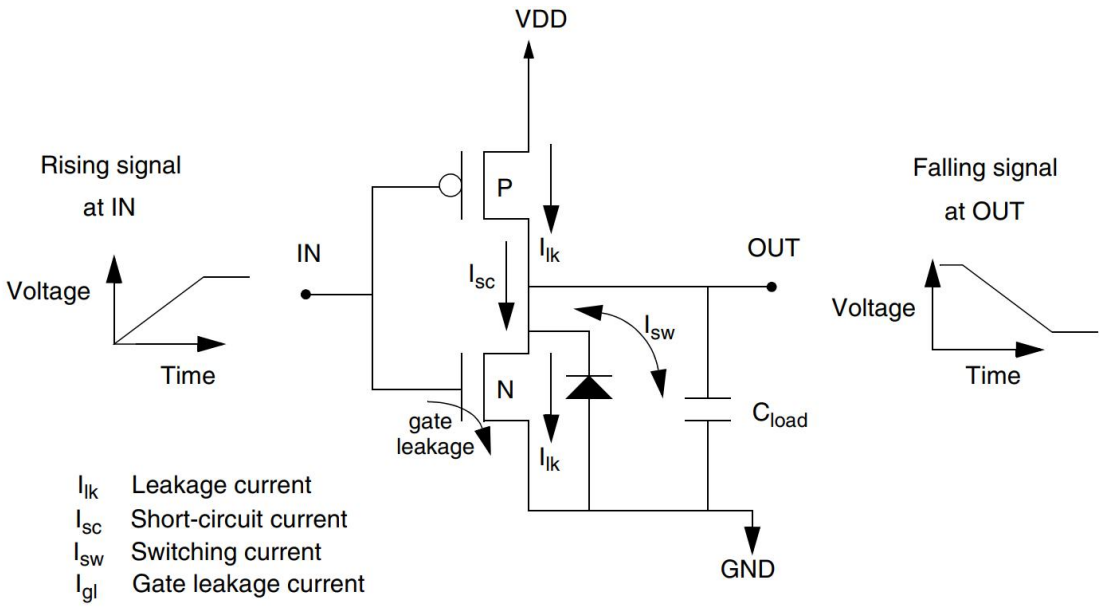
$$Power_{flip} = \frac{1}{2} \times Toggle \times (rise\_power + fall\_power)$$

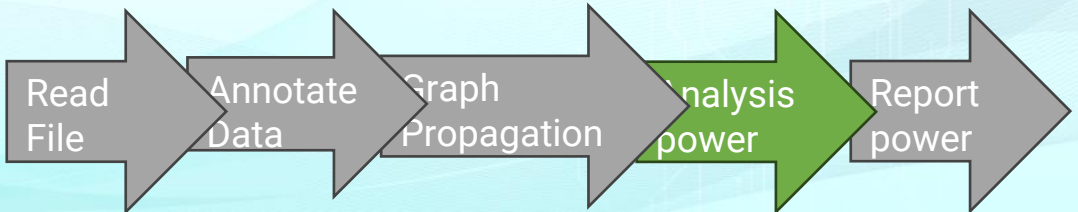
```
internal_power () {
  related_pin : "A1";
  related_pg_pin : VDD;
  rise_power (power_template_7x7) {
    index_1 ("0.0028, 0.0124, 0.0317, 0.0701, 0.1471, 0.3009, 0.6086");
    index_2 ("0.00022, 0.00116, 0.00304, 0.0068, 0.01432, 0.02937, 0.05947");
    values ( \
      "0.000355066, 0.000367437, 0.000374601, 0.000375905, 0.000378532, 0.000387316, 0.000380065", \
      "0.000345814, 0.00035723, 0.000366141, 0.00036999, 0.00037432, 0.000375085, 0.000382025", \
      "0.000335043, 0.000344772, 0.000353859, 0.000358469, 0.000365037, 0.000368394, 0.000371179", \
      "0.000331212, 0.000339181, 0.000346504, 0.000353696, 0.000363798, 0.00036253, 0.00036397", \
      "0.000345662, 0.000351309, 0.000354993, 0.000365775, 0.000367597, 0.000375976, 0.000376152", \
      "0.000398908, 0.000400613, 0.000403786, 0.000410047, 0.000411766, 0.000423029, 0.000418313", \
      "0.000534847, 0.000528473, 0.000528102, 0.000529924, 0.000532691, 0.000543678, 0.000544689" \
    );
  }
  fall_power (power_template_7x7) {
    index_1 ("0.0028, 0.0124, 0.0317, 0.0701, 0.1471, 0.3009, 0.6086");
    index_2 ("0.00022, 0.00116, 0.00304, 0.0068, 0.01432, 0.02937, 0.05947");
    values ( \
      "0.000689477, 0.000699258, 0.000702576, 0.000703651, 0.000703656, 0.00070372, 0.000703144", \
      "0.00067254, 0.000683008, 0.000688171, 0.000689639, 0.000689955, 0.000689765, 0.000689331", \
      "0.000658228, 0.000667582, 0.000674376, 0.000677719, 0.000678955, 0.000679228, 0.000678944", \
      "0.000651265, 0.000659093, 0.000666085, 0.000671119, 0.00067391, 0.00067501, 0.000675168", \
      "0.000670004, 0.000674685, 0.000679811, 0.00068542, 0.000689297, 0.000691454, 0.000692237", \
      "0.000736503, 0.00073615, 0.00073877, 0.000742947, 0.000747088, 0.000749985, 0.000751374", \
      "0.000903983, 0.000894878, 0.0008919, 0.000891403, 0.000893954, 0.000896121, 0.000898064" \
    );
  }
}
```



Power Calculation and Analysis

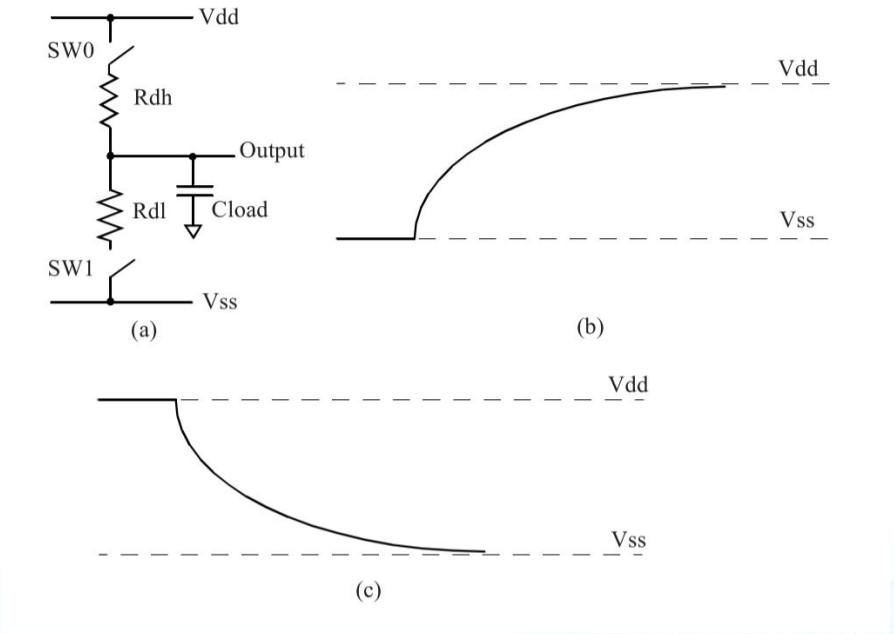
- Switching power
  - ✓ Switching power refers to the power caused by the **switching operation** of transistors in CMOS circuits
  - ✓ When a CMOS circuit performs a switching operation, the transistors will frequently switch from on to off or from off to on, generating switching power
  - ✓ The switching power in cell is called internal power(include short circuit power), and in net is called switch power



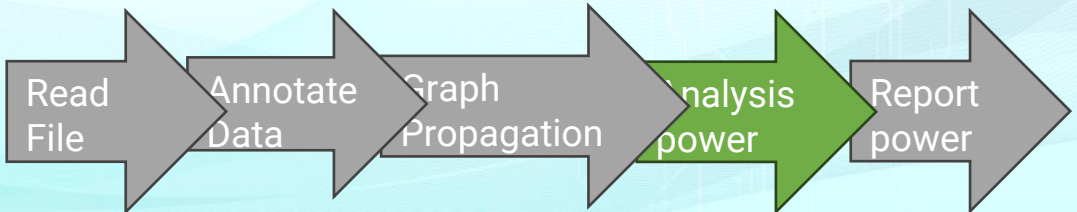


Power Calculation and Analysis

- Net power - Switching power
  - ✓ Switching power is the power of the interconnect net, related to load capacitance, toggle rate, and voltage
  - ✓ The load capacitance includes **interconnect load pin capacitance** (available in the technology library lib) and **interconnect equivalent parasitic capacitance** (obtained from parasitic parameter SPEF files)



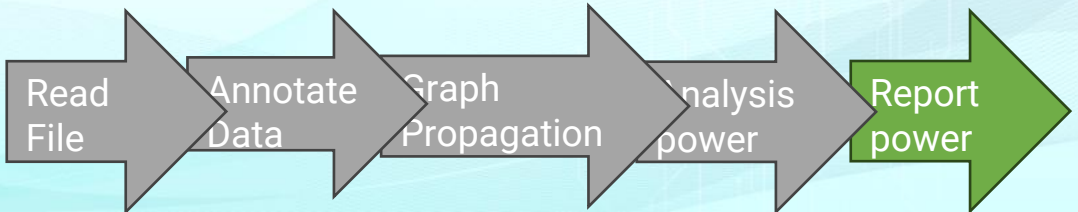
$$P_{cload} = KfC_LVDD^2$$



Report Power

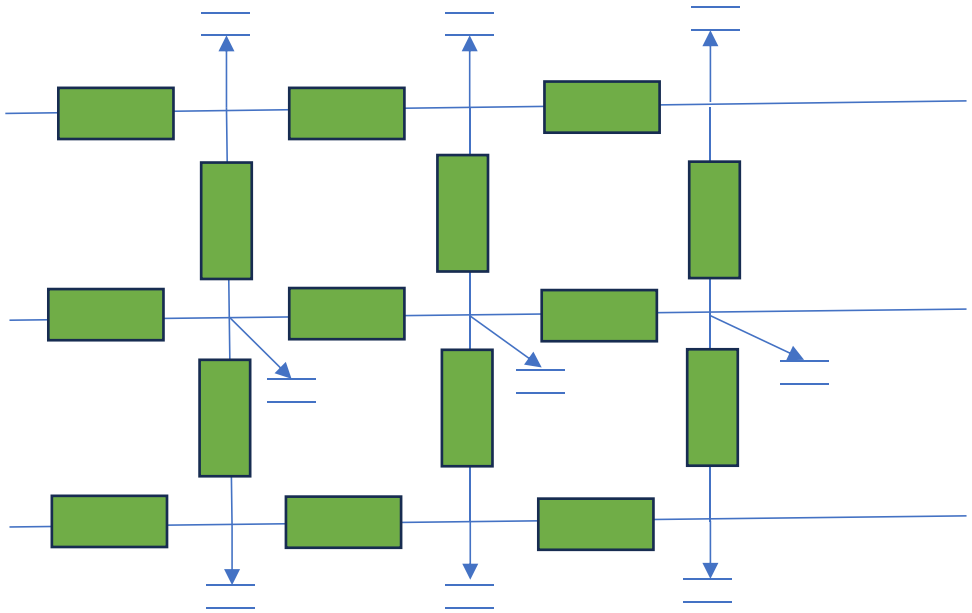
- Power report
  - Power report divides the power data into power group include **internal power, switch power, and leakage power**

```
Generate the report at 2023-08-31T20:32:04
Report : Averaged Power
 +----------------+----------------+-------------+---------------+------------+-----------+
| Power Group     | Internal Power | Switch Power | Leakage Power | Total Power | (%)       |
+----------------+----------------+-------------+---------------+------------+-----------+
| clock_net_work | 3.165e-05      | 5.552e-05    | 2.862e-10     | 8.717e-05   | (23.041%) |
| combinational  | 2.116e-04      | 7.953e-05    | 1.015e-09     | 2.911e-04   | (76.959%) |
+----------------+----------------+-------------+---------------+------------+-----------+
Net Switch Power   ==    1.351e-04 (35.698%)
Cell Internal Power   ==    2.433e-04 (64.301%)
Cell Leakage Power   ==    1.301e-09 (0.000%)
Total Power   ==  3.783e-04
```

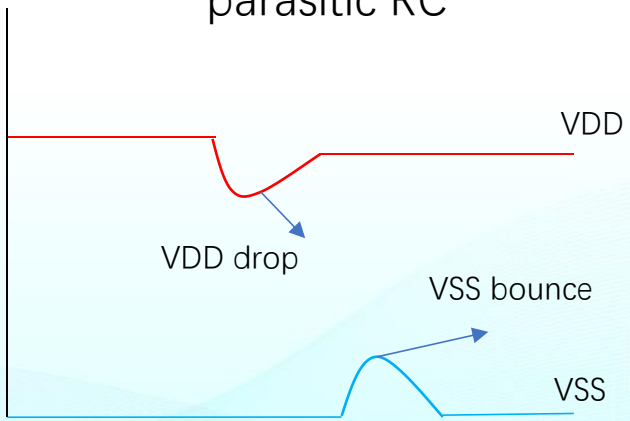


IR Drop Calculation

- Concept
  - ✓ IR drop refers to the **VDD drop/VSS bounce** caused by parasitic parameters on the power delivery network
  - ✓ Static IR drop uses the **average power** to calculate the current
  - ✓ Dynamic IR drop uses **dynamic power** to calculate the current that changes over time



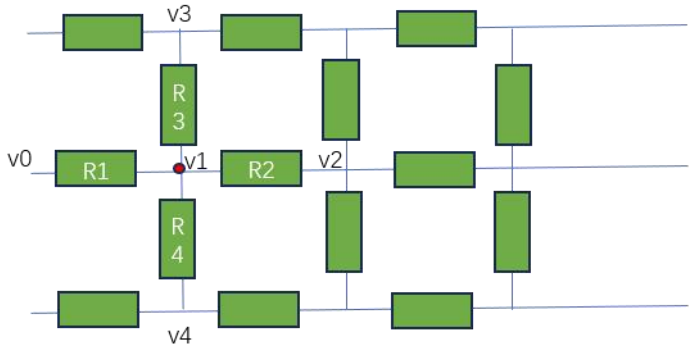
parasitic RC



dynamic IR drop

IR Drop Calculation

- Numerical Solver Method
  - ✓ Includes **direct methods and iterative methods**
  - ✓ Direct methods, such as **Gaussian elimination, LU decomposition, and QR decomposition**
  - ✓ Iterative methods, such as **Jacobi, Gauss-Seidel, and conjugate gradient**

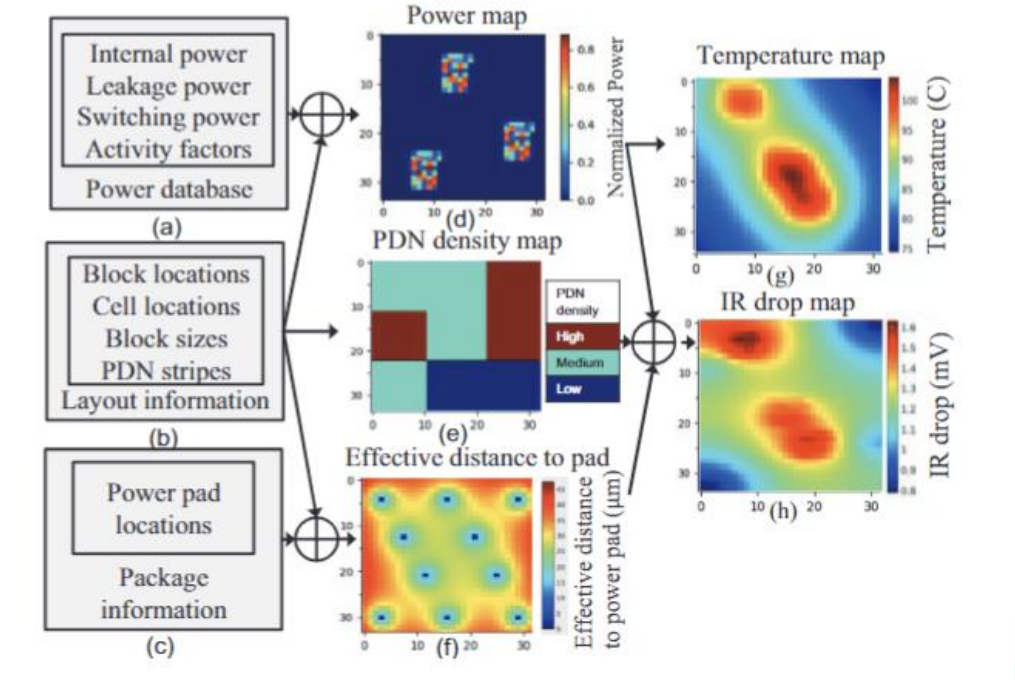


$$GX = J$$

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ G1 & -(G1+G2+G3+G4) & G2 & G3 & G4 & 0\cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} I_0 \\ 0 \\ \vdots \\ I_n \end{pmatrix} \quad (2)$$

IR Drop Calculation

- ML Method
  - ✓ Converting features into layout image features
  - ✓ Using a CNN-based model to predict IR Drop
  - ✓ Available features include: **current , resistance, and congestion**



V. A. Chhabria, V. Ahuja, A. Prabhu, N. Patil, P. Jain, and S. S.Sapatnekar, "Thermal and ir drop analysis using convolutional encoder_x0002_decoder networks," in Proceedings of the 26th Asia and South Pacific Design Automation Conference, 2021, pp. 690–696.

Results

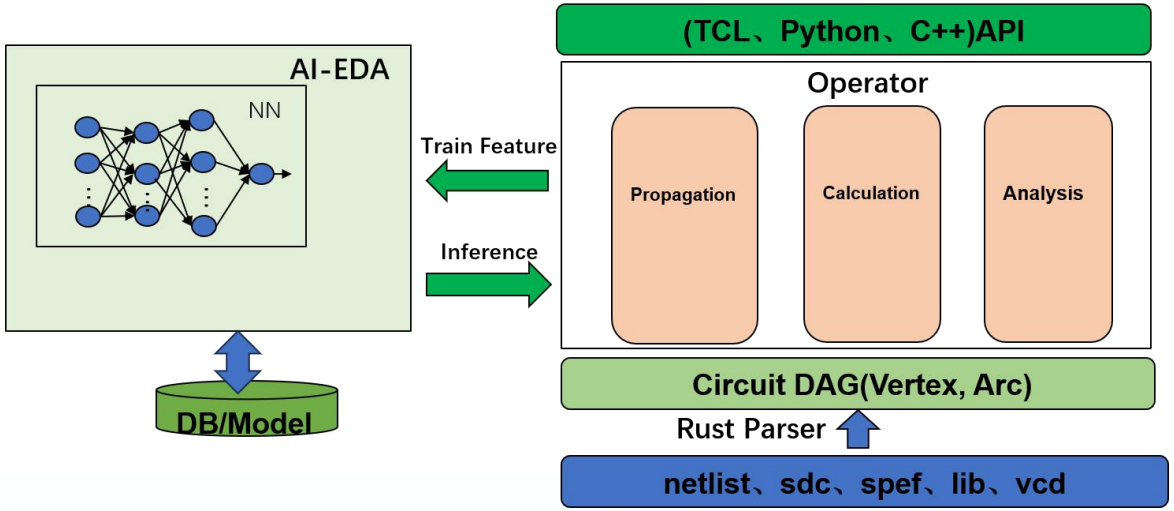
- The test examples in Table are derived from cases in the **Openlane**, based on the open-source **skywater130** process

Test Case	iPA Total Power	Innovus Total Power	Inaccuracy
aes_cipher_top	22.22mW	23.74mW	6.4%
gcd	0.38mW	0.37mW	3.6%
uart	0.51mW	0.49mW	3.9%

Future Work

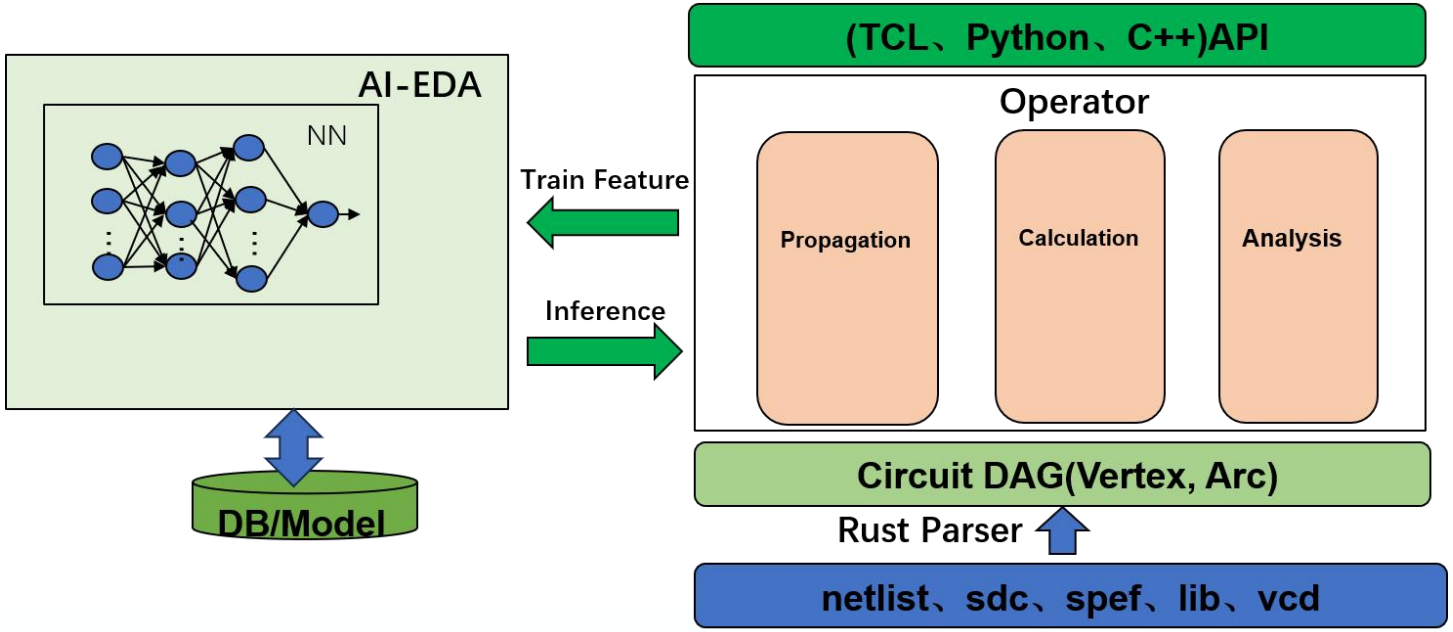
- Software

  - ✓ Design goal of iPA is to be **fully decoupled, pluggable, and easily scalable**
  - ✓ Hybrid programming model of **Rust and C++**
  - ✓ **Integrate AI models**
  - ✓ **Export the data needed for AI** model training



Future Work

- Flow
  - ✓ AI for VCD anotate
  - ✓ AI for Power Graph Reduce
  - ✓ AI for Toggle/SP Predict
  - ✓ AI for Early Power Estimate



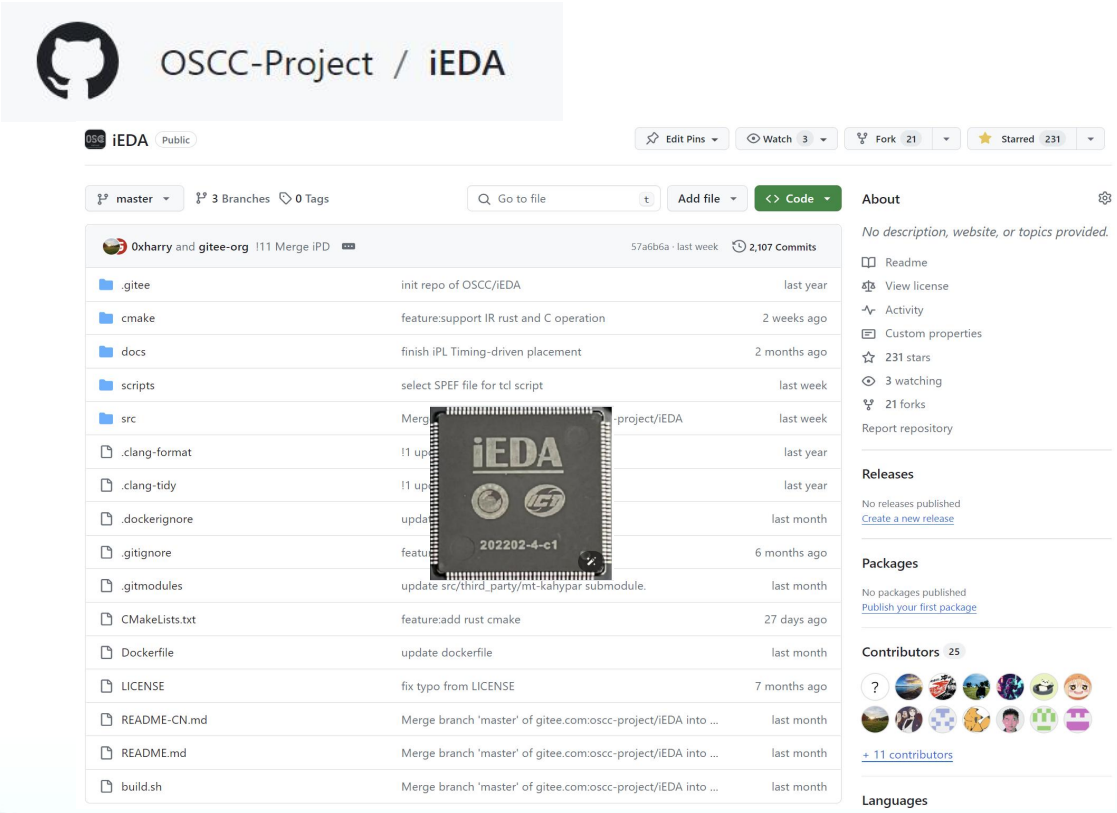
Conclusions

- The iPA tool supports
  - ✓ basic functions of power analysis and IR drop
  - ✓ VCD anotated Toggle/SP and Toggle/SP static propagation analysis methods

- In the future
  - ✓ use VCD cycle-level dynamic power analysis, combine AI/ML technology to improve analysis **accuracy**
  - ✓ use a Rust/C++ hybrid parallel framework to improve analysis **speed**
  - ✓ allow iPA to be used as an AI **feature extraction** tool and store analysis data for model use

iEDA Tutorial Agenda

- **Part 0:** iEDA Overview **(Xingquan Li)**
- **Part 1:** iEDA Infrastructure **(Zengrong Huang)**
- **Part 2:** iPL: Placement Tool and Its Technology **(Shijian Chen)**
- **Part 3:** iCTS: Clock Tree Synthesis Tool and Its Technologies **(Weiguo Li)**
- **Part 4:** iRT: Routing Tool and Its Technologies **(Zhisheng Zeng)**
- **Part 5:** iSTA: Static Timing Analysis Tool and Its Technologies **(Simin Tao/He Liu)**
- **Part 6:** iPA: Power Analysis Tool and Its Technologies **(Simin Tao)**



Thanks

Simin Tao
taosm@pcl.ac.cn