



# Introduction to Liberty Metric

Hockchen 2022/08

```

graph TD
    subgraph Traditional_Flow [Traditional Flow]
        direction TB
        A[Process & Library Analysis] --> B[RTL Design]
        B --> C[Physical Implementation]
        C --> D[Fabrication]
        D --> E[Testing & Validation]
        E --> F[MP]
    end

    subgraph Enhanced_Flow [Enhanced Flow]
        direction TB
        G[Process Monitor Design Test Definition] --> H[Process Monitor Hardening & Integration]
        H --> I[WAT/CP/FT/SLT Analysis & Correlation]
        I --> J[Process Opt. Binning Strategy]
    end

    A --> G
    A --> K[Metric Extraction Timing Strategy]
    K --> L[Custom cell/SRAM Critical Path Opt.]
    L <--> M[Area/Power Opt. MEGA/Custom re-K]
    M --> C
    J --> F

    F -.-> N[Design & timing recipe optimization feedback 10%+ productivity]
    N -.-> G
    N -.-> K
    N -.-> L
    N -.-> M
    N -.-> J
  
```

The diagram illustrates a semiconductor design flow, comparing a traditional sequential process with an enhanced flow that integrates process monitoring and optimization.

**Traditional Flow (Central Column):**

- Process & Library Analysis
- RTL Design
- Physical Implementation
- Fabrication
- Testing & Validation
- MP (Manufacturing Process)

**Enhanced Flow (Left Column - Green Boxes):**

- Process Monitor Design Test Definition
- Process Monitor Hardening & Integration
- WAT/CP/FT/SLT Analysis & Correlation
- Process Opt. Binning Strategy

**Enhanced Flow (Right Column - Blue and Gold Boxes):**

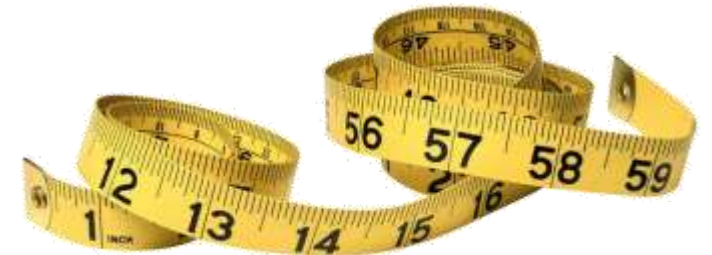
- Metric Extraction Timing Strategy
- Custom cell/SRAM Critical Path Opt.
- Area/Power Opt. MEGA/Custom re-K

**Annotations and Feedback:**

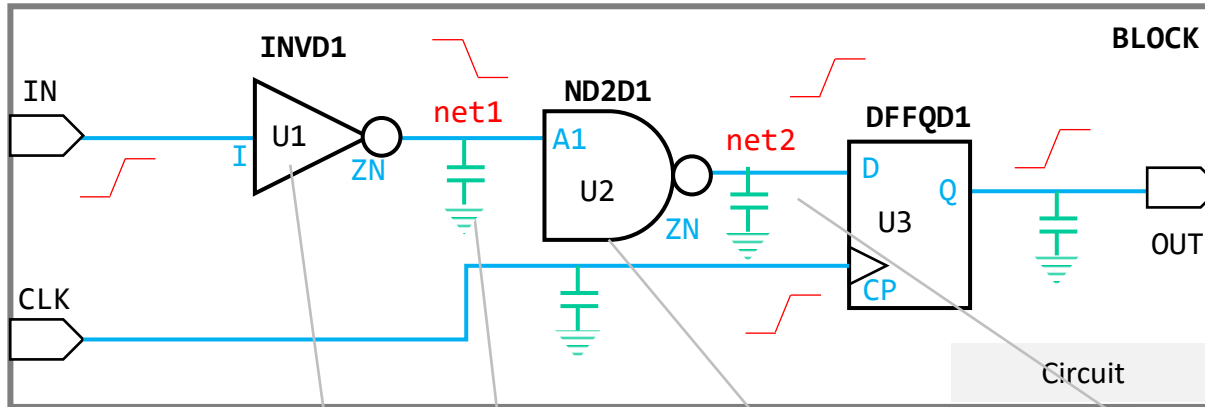
- Verilog (design spec.)** and **Liberty (timing/power)** are inputs to the RTL Design and Physical Implementation stages, respectively.
- A dashed green line at the bottom represents **Design & timing recipe optimization feedback (10%+ productivity)**, which loops back from the MP stage to the start of the enhanced flow.

# Metric

who is taller?



# Design Metric



```

module BLOCK(IN, OUT, CLK);
input IN, CLK;
output OUT;
INVD1 U1 ( .I(IN), .ZN(net1) );
ND2D1 U2 ( .A1(net1), .ZN(net2) );
DFFQD1 U3 ( .D(net2), .CP(CLK), .Q(OUT) );
endmodule
    
```

Verilog Model

```

Delay ('I,ZN,', 'combinational', 'cell_fall')
Trans: [0.0032, 0.0079, 0.0173, 0.036 , 0.0735, 0.1485, 0.2984, 0.5983]
Load:  [0.00018, 0.00052, 0.0012 , 0.00256, 0.00527, 0.01069, 0.02153, 0.04321]
Value:[[0.011, 0.016, 0.024, 0.041, 0.075, 0.143, 0.278, 0.548],
[0.014, 0.019, 0.028, 0.045, 0.079, 0.146, 0.281, 0.552],
[0.019, 0.025, 0.034, 0.052, 0.085, 0.153, 0.288, 0.559],
[0.028, 0.034, 0.045, 0.064, 0.099, 0.167, 0.302, 0.572],
[0.043, 0.051, 0.064, 0.085, 0.123, 0.194, 0.329, 0.6 ],
[0.068, 0.08 , 0.097, 0.122, 0.166, 0.242, 0.383, 0.655],
[0.103, 0.125, 0.153, 0.189, 0.239, 0.326, 0.48 , 0.762],
[0.153, 0.191, 0.239, 0.298, 0.371, 0.472, 0.648, 0.956]]
    
```

```

Trans ('A1,ZN,', 'combinational', 'rise_transition')
Trans: [0.0032, 0.0079, 0.0173, 0.036 , 0.0735, 0.1485, 0.2984, 0.5983]
Load:  [0.00018, 0.00041, 0.00087, 0.00179, 0.00364, 0.00733, 0.01472, 0.0295]
Value:[[0.015, 0.022, 0.035, 0.061, 0.113, 0.218, 0.427, 0.844],
[0.016, 0.022, 0.035, 0.061, 0.113, 0.218, 0.427, 0.844],
[0.018, 0.024, 0.037, 0.062, 0.113, 0.218, 0.426, 0.844],
[0.021, 0.028, 0.041, 0.066, 0.116, 0.218, 0.427, 0.844],
[0.025, 0.032, 0.047, 0.074, 0.124, 0.224, 0.429, 0.844],
[0.04 , 0.045, 0.056, 0.086, 0.141, 0.241, 0.44 , 0.847],
[0.07 , 0.077, 0.088, 0.107, 0.163, 0.274, 0.475, 0.872],
[0.126, 0.138, 0.153, 0.176, 0.211, 0.318, 0.541, 0.944]]
    
```

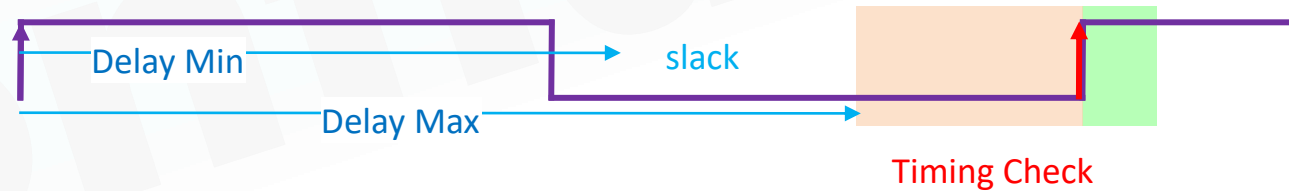
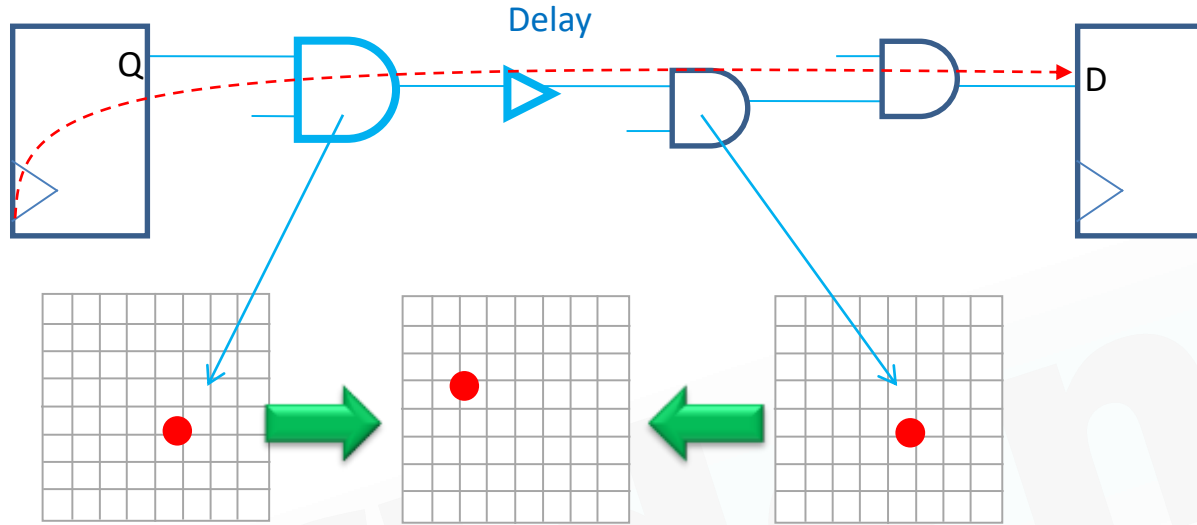
```

Trans ('I,ZN,', 'combinational', 'fall_transition')
Trans: [0.0032, 0.0079, 0.0173, 0.036 , 0.0735, 0.1485, 0.2984, 0.5983]
Load:  [0.00018, 0.00052, 0.0012 , 0.00256, 0.00527, 0.01069, 0.02153, 0.04321]
Value:[[0.01 , 0.017, 0.032, 0.062, 0.121, 0.24 , 0.477, 0.951],
[0.011, 0.018, 0.033, 0.062, 0.121, 0.24 , 0.477, 0.951],
[0.012, 0.02 , 0.034, 0.063, 0.121, 0.24 , 0.477, 0.951],
[0.014, 0.023, 0.038, 0.067, 0.124, 0.24 , 0.477, 0.951],
[0.02 , 0.027, 0.044, 0.074, 0.131, 0.245, 0.477, 0.951],
[0.035, 0.043, 0.054, 0.086, 0.147, 0.261, 0.487, 0.953],
[0.063, 0.073, 0.087, 0.109, 0.17 , 0.293, 0.52 , 0.973],
[0.112, 0.129, 0.15 , 0.178, 0.219, 0.337, 0.584, 1.038]]
    
```

```

Delay ('A1,ZN,', 'combinational', 'cell_rise')
Trans: [0.0032, 0.0079, 0.0173, 0.036 , 0.0735, 0.1485, 0.2984, 0.5983]
Load:  [0.00018, 0.00041, 0.00087, 0.00179, 0.00364, 0.00733, 0.01472, 0.0295]
Value:[[0.016, 0.02 , 0.027, 0.042, 0.071, 0.13 , 0.249, 0.485],
[0.019, 0.023, 0.03 , 0.045, 0.075, 0.134, 0.252, 0.489],
[0.025, 0.029, 0.037, 0.052, 0.082, 0.141, 0.259, 0.496],
[0.034, 0.039, 0.048, 0.065, 0.096, 0.155, 0.274, 0.51 ],
[0.051, 0.057, 0.068, 0.087, 0.121, 0.183, 0.303, 0.539],
[0.081, 0.09 , 0.103, 0.125, 0.164, 0.233, 0.357, 0.596],
[0.127, 0.142, 0.164, 0.193, 0.239, 0.318, 0.456, 0.706],
[0.196, 0.222, 0.259, 0.309, 0.375, 0.466, 0.626, 0.903]]
    
```

# PPA Problem Formulation



Given a logic path, minimize the timing path delay, area, energy and leakage while satisfying the maximum transition constraint for all operating corners, wherein cell sizing and load splitting (**buffering**) may be utilized.

# What is Liberty Metric

```
release/<version>/base_ulvt/lib/CCS
base_ulvttt_0p29v_105c_typical_ccs.lib
base_ulvttt_0p29v_50c_typical_ccs.lib
base_ulvttt_0p29v_85c_typical_ccs.lib
base_ulvttt_0p31v_25c_typical_ccs.lib
base_ulvttt_0p31v_50c_typical_ccs.lib
base_ulvttt_0p31v_85c_typical_ccs.lib
base_ulvttt_0p33v_25c_typical_ccs.lib
base_ulvttt_0p33v_50c_typical_ccs.lib
base_ulvttt_0p35v_50c_typical_ccs.lib
base_ulvttt_0p35v_85c_typical_ccs.lib
base_ulvttt_0p75v_25c_typical_ccs.lib
base_ulvttt_0p75v_85c_typical_ccs.lib
...
```

1000+ CCS, ~290MB/lib

metric/JSON/<version>/\*.json

~20MB/json

metric/metric/<version>/LSC/\*.csv  
metric/metric/<version>/merge\_LSC.csv

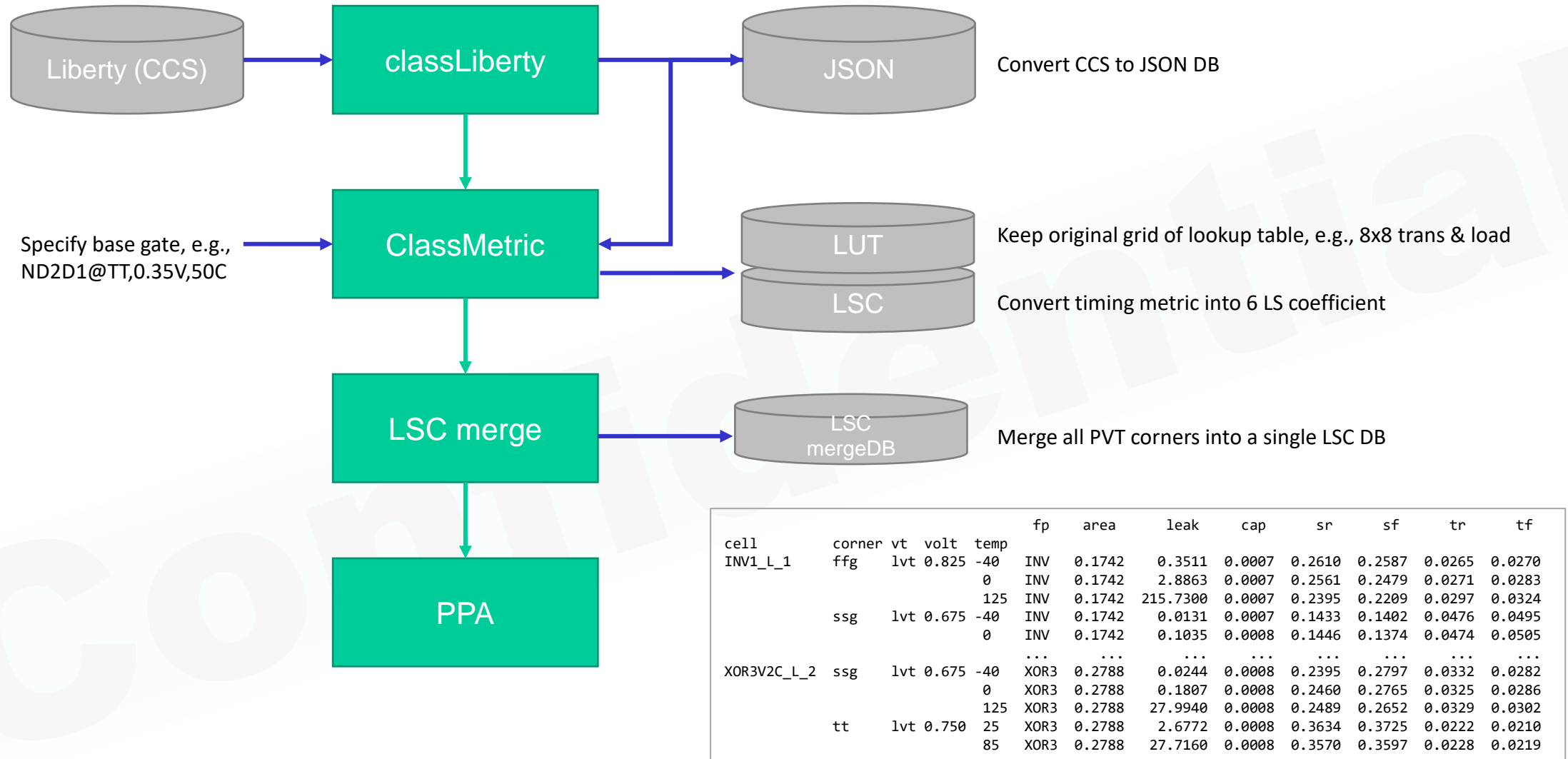
~3MB/csv

merge to a single DB

				area	leak	tr	tf	dr	df	metrics
cell	corner	volt	temp							
NAND2V1_UL_1	tt	0.29	50	0.05227	1.29540	0.08012	0.13440	0.07324	0.11323	LUT/LSC by timing-arc
			85	0.05227	4.28505	0.06820	0.11640	0.05909	0.09651	
			105	0.05227	7.76380	0.06399	0.11049	0.05224	0.08912	
	25	0.31	25	0.05227	0.52076	0.07656	0.11954	0.07460	0.10631	
			50	0.05227	1.41850	0.06894	0.10969	0.06389	0.09463	
			85	0.05227	4.68190	0.06081	0.09902	0.05239	0.08259	
	50	0.33	25	0.05227	0.56828	0.06579	0.09864	0.06514	0.08944	
			50	0.05227	1.54540	0.05998	0.09184	0.05707	0.08125	
			85	0.05227	5.09010	0.05404	0.08472	0.04780	0.07234	
	85	0.35	50	0.05227	1.67620	0.05320	0.07859	0.05172	0.07121	
			85	0.05227	5.50930	0.04886	0.07400	0.04409	0.06445	
			0.75	0.05227	1.88702	0.02228	0.02689	0.01787	0.02131	
	PVT as DB index	85	25	0.05227	1.88702	0.02228	0.02689	0.01787	0.02131	
			85	0.05227	15.81210	0.02312	0.02855	0.01639	0.02199	

extracted metric

# Metric Extraction Flow





# Metric Extraction: LUT

lib	cell	vt	P	V	T	fp	area	leak	cap	sr	sf	tr	tf	cr	cf	pr	pf	lutT	lutP
lvtfngp_0p36v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.36	12	nd2d1	0.0547	5.2074	0.0003	0.0540	0.0267	0.0460	0.0813	0.0414	0.0648	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p36v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.36	-4	nd2d1	0.0547	0.0073	0.0003	0.0205	0.0160	0.1060	0.1297	0.0946	0.1107	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p44v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.44	12	nd2d1	0.0547	6.8532	0.0003	0.0891	0.0470	0.0310	0.0492	0.0285	0.0407	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p44v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.44	-4	nd2d1	0.0547	0.0102	0.0003	0.0587	0.0421	0.0414	0.0525	0.0458	0.0519	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p48v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.48	12	nd2d1	0.0547	7.7496	0.0003	0.1067	0.0580	0.0270	0.0411	0.0247	0.0342	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p48v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.48	-4	nd2d1	0.0547	0.0119	0.0003	0.0816	0.0578	0.0319	0.0400	0.0366	0.0406	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p4v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.4	12	nd2d1	0.0547	6.0066	0.0003	0.0713	0.0364	0.0368	0.0615	0.0337	0.0502	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvtfngp_0p4v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lv	ffgnp	0.4	-4	nd2d1	0.0547	0.0087	0.0003	0.0377	0.0370	0.0400	0.0766	0.0616	0.0713	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p36v_0c	ND2D1LVT	lv	t	0.36	6	nd2d1	0.0547	0.0231	0.0003	0.0168	0.0179	0.1095	0.1792	0.1095	0.1442	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p36v_1t	ND2D1LVT	lv	t	0.36	6	nd2d1	0.0547	1.4619	0.0003	0.0359	0.0115	0.0581	0.1152	0.0581	0.0914	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p36v_25c_typical_ccs	ND2D1LVT	lv	tt	0.36	2	nd2d1	0.0547	0.0766	0.0003	0.0210	0.0133	0.1052	0.1572	0.0914	0.1270	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p36v_50c_typical_ccs	ND2D1LVT	lv	tt	0.36	50	nd2d1	0.0547	0.2188	0.0003	0.0255	0.0150	0.0881	0.1403	0.0780	0.1133	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p36v_85c_typical_ccs	ND2D1LVT	lv	tt	0.36	8	nd2d1	0.0547	0.7753	0.0003	0.0320	0.0173	0.0720	0.1227	0.0642	0.0983	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p44v_0c_typical_ccs	ND2D1LVT	lv	tt	0.44	0	nd2d1	0.0547	0.0312	0.0003	0.0487	0.0320	0.0494	0.0686	0.0509	0.0633	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p44v_105c_typical_ccs	ND2D1LVT	lv	tt	0.44	10	nd2d1	0.0547	1.9233	0.0003	0.0686	0.0372	0.0380	0.0611	0.0361	0.0515	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p44v_25c_typical_ccs	ND2D1LVT	lv	tt	0.44	2	nd2d1	0.0547	0.1024	0.0003	0.0536	0.0334	0.0457	0.0662	0.0466	0.0600	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p44v_50c_typical_ccs	ND2D1LVT	lv	tt	0.44	50	nd2d1	0.0547	0.2906	0.0003	0.0585	0.0347	0.0428	0.0643	0.0429	0.0571	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p44v_85c_typical_ccs	ND2D1LVT	lv	tt	0.44	8	nd2d1	0.0547	1.0230	0.0003	0.0651	0.0364	0.0395	0.0621	0.0384	0.0534	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p48v_0c_typical_ccs	ND2D1LVT	lv	tt	0.48	0	nd2d1	0.0547	0.0358	0.0003	0.0686	0.0450	0.0371	0.0506	0.0401	0.0484	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p48v_105c_typical_ccs	ND2D1LVT	lv	tt	0.48	10	nd2d1	0.0547	2.1747	0.0003	0.0862	0.0400	0.0317	0.0488	0.0303	0.0417	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p48v_25c_typical_ccs	ND2D1LVT	lv	tt	0.48	2	nd2d1	0.0547	0.1167	0.0003	0.0732	0.0459	0.0355	0.0500	0.0374	0.0466	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p48v_50c_typical_ccs	ND2D1LVT	lv	tt	0.48	50	nd2d1	0.0547	0.3301	0.0003	0.0776	0.0467	0.0341	0.0495	0.0350	0.0450	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p48v_85c_typical_ccs	ND2D1LVT	lv	tt	0.48	8	nd2d1	0.0547	1.1585	0.0003	0.0832	0.0476	0.0324	0.0490	0.0319	0.0428	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p4v_0c_typical_ccs	ND2D1LVT	lv	tt	0.4	0	nd2d1	0.0547	0.0270	0.0003	0.0309	0.0207	0.0736	0.1031	0.0699	0.0896	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p4v_105c_typical_ccs	ND2D1LVT	lv	tt	0.4	10	nd2d1	0.0547	1.6859	0.0003	0.0515	0.0273	0.0481	0.0807	0.0446	0.0665	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p4v_25c_typical_ccs	ND2D1LVT	lv	tt	0.4	2	nd2d1	0.0547	0.0890	0.0003	0.0358	0.0224	0.0647	0.0959	0.0620	0.0828	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p4v_50c_typical_ccs	ND2D1LVT	lv	tt	0.4	50	nd2d1	0.0547	0.2535	0.0003	0.0408	0.0240	0.0580	0.0900	0.0555	0.0769	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po
lvttt_0p4v_85c_typical_ccs	ND2D1LVT	lv	tt	0.4	8	nd2d1	0.0547	0.8954	0.0003	0.0477	0.0261	0.0511	0.0836	0.0481	0.0699	0.0000	0.0000	{{'A1,ZN','combina	{{'A1,ZN','fall_po

Library

PVT

Metric

Timing LUT

Power LUT

Metrics characterized @ trans, load = 40ps, 1.7fF

Encapsulated LUT with key values as (arc, ttype, ctype)

Timing/Power LUT

```

A1,ZN, combinational fall_transition
cell_rise
cell_fall
rise_transition
A2,ZN, combinational fall_transition
cell_rise
cell_fall
rise_transition

```

index_1	index_2	values
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.022591, 0.032241, 0.051562, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.014066, 0.017375, 0.023895, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.021565, 0.027368, 0.038805, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.013660, 0.019359, 0.030829, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.022592, 0.032264, 0.051622, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.015145, 0.018477, 0.025039, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.023371, 0.029119, 0.040510, ...
[0.0032, 0.0079, 0.0173,	[0.00018, 0.00041, 0.00087,	[0.015120, 0.020889, 0.032385, ...



# Metric Extraction: LSC

lib	cell	vt	P	V	T	fp	area	leak	cap	sr	sf	tr	tf	dr	df	pr	pf	metrics
lvtffgnp_0p36v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.36	125	nd2d1	0.0547	5.2074	0.0003	0.0540	0.0267	0.0460	0.0813	0.0414	0.0648	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p36v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.36	-40	nd2d1	0.0547	0.0073	0.0003	0.0205	0.0160	0.1060	0.1297	0.0946	0.1107	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p44v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.44	125	nd2d1	0.0547	6.8532	0.0003	0.0891	0.0470	0.0310	0.0492	0.0285	0.0407	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p44v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.44	-40	nd2d1	0.0547	0.0102	0.0003	0.0587	0.0421	0.0414	0.0525	0.0458	0.0519	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p48v_125c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.48	125	nd2d1	0.0547	7.7496	0.0003	0.1067	0.0580	0.0270	0.0411	0.0247	0.0342	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p48v_m40c_cbest_CCbest_T_ccs	ND2D1LVT	lvt	ffgnp	0.48	-40	nd2d1	0.0547	0.0119	0.0003	0.0816	0.0578	0.0319	0.0400	0.0366	0.0406	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p4v_125c_cbest_	ND2D1LVT	lvt	ffgnp	PVT		nd2d1	0.0547	6.0066	0.0003	0.0713	0.0364	Metric	3615	0.0337	0.0502	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvtffgnp_0p4v_m40c_cbest_	ND2D1LVT	lvt	ffgnp			nd2d1	0.0547	0.0087	0.0003	0.0377	0.0279		3766	0.0616	0.0713	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p36v_0c_typical_ccs	ND2D1LVT	lvt	tt	0.36	0	nd2d1	0.0547	0.0231	0.0003	0.0168	0.0117	0.1297	0.1792	0.1095	0.1442	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p36v_105c_typical_ccs	ND2D1LVT	lvt	tt	0.36	105	nd2d1	0.0547	1.4619	0.0003	0.0359	0.0186	0.0655	0.1152	0.0581	0.0914	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p36v_25c_typical_ccs	ND2D1LVT	lvt	tt	0.36	25	nd2d1	0.0547	0.0766	0.0003	0.0210	0.0133	0.1052	0.1572	0.0914	0.1270	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p36v_50c_typical_ccs	ND2D1LVT	lvt	tt	0.36	50	nd2d1	0.0547	0.2188	0.0003	0.0255	0.0150	0.0881	0.1403	0.0780	0.1133	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p36v_85c_typical_ccs	ND2D1LVT	lvt	tt	0.36	85	nd2d1	0.0547	0.7753	0.0003	0.0320	0.0173	0.0720	0.1227	0.0642	0.0983	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p44v_0c_typical_ccs	ND2D1LVT	lvt	tt	0.44	0	nd2d1	0.0547	0.0312	0.0003	0.0487	0.0320	0.0494	0.0686	0.0509	0.0633	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p44v_105c_typical_ccs	ND2D1LVT	lvt	tt	0.44	105	nd2d1	0.0547	1.9233	0.0003	0.0686	0.0372	0.0380	0.0611	0.0361	0.0515	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p44v_25c_typical_ccs	ND2D1LVT	lvt	tt	0.44	25	nd2d1	0.0547	0.1024	0.0003	0.0536	0.0334	0.0457	0.0662	0.0466	0.0600	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p44v_50c_typical_ccs	ND2D1LVT	lvt	tt	0.44	50	nd2d1	0.0547	0.2906	0.0003	0.0585	0.0347	0.0428	0.0643	0.0429	0.0571	0.0000	0.0000	{{'A1,ZN,', 'combinat
lvttt_0p44v_85c_typical_ccs	ND2D1LVT	lvt	tt	0.44	85	nd2d1	0.0547	1.0230	0.0003	0.0651	0.0364	0.0395	0.0621	0.0384	0.0534	0.0000	0.0000	{{'A1,ZN,', 'combinat

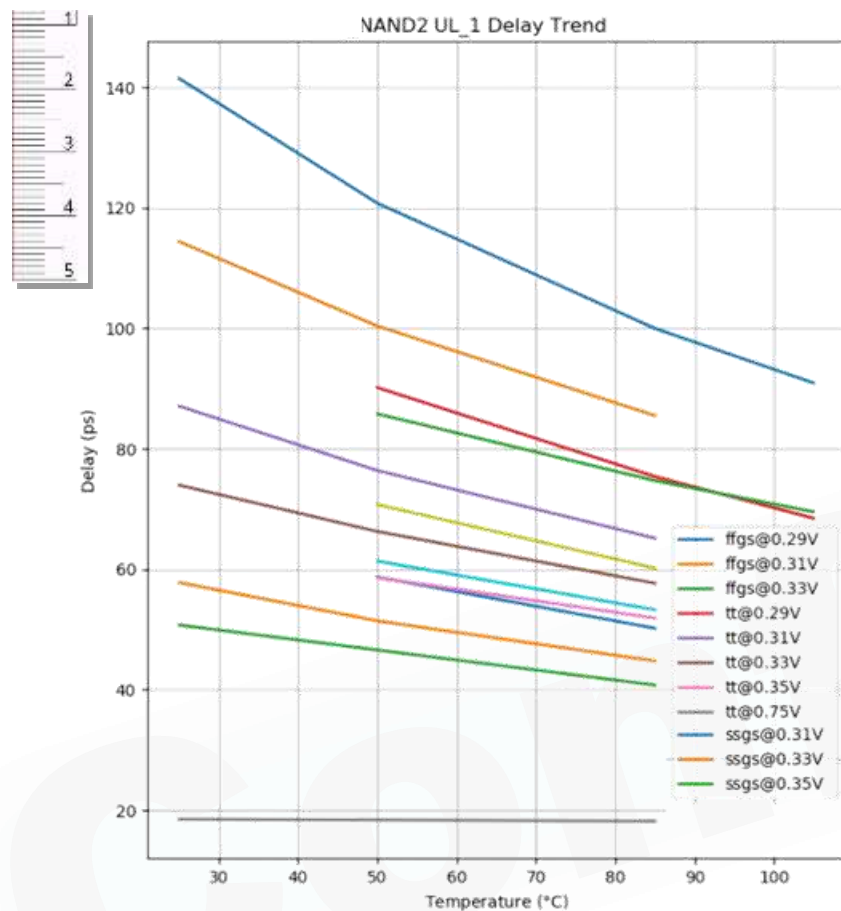
Encapsulated LSC with key values as (arc, ttype, ctype)

LSC metrics

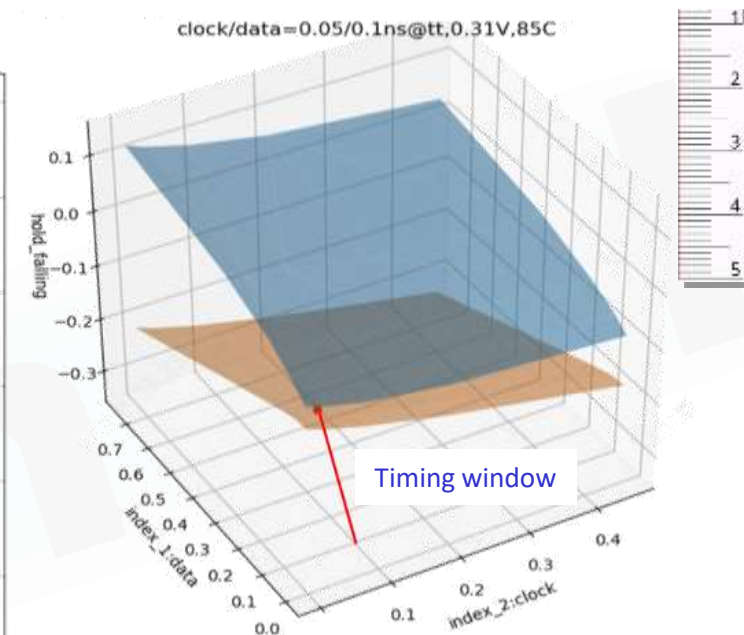
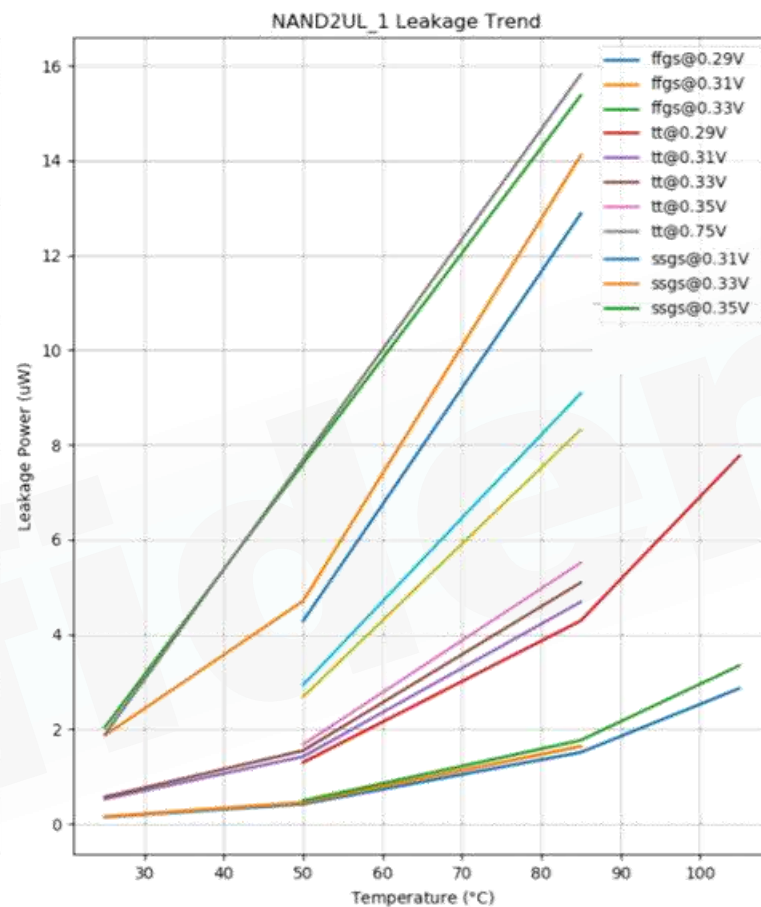
			metric	coeff
A1,ZN, combinational	fall_transition		0.0908024	[0.015407755176671863, 41.29951297003873, 22.3...
	cell_rise		0.0547047	[0.006486304221577642, 16.77994615785185, -104...
	cell_fall		0.0758079	[0.012672860017655243, 26.979833277102507, -91...
	rise_transition		0.0573029	[0.00922035581253383, 24.4780841525638, 7.2453...
A2,ZN, combinational	fall_transition		0.0892851	[0.015248332470100986, 41.46343045194753, 18.7...
	cell_rise		0.0563164	[0.007993053419577371, 16.586521826143425, -97...
	cell_fall		0.0780313	[0.01565863319027824, 26.50575687412519, -74.1...
	rise_transition		0.0586536	[0.010574893715241061, 24.638121118893128, 2.3...
A1,ZN, fall_power		NaN	-5.16309e-06	[-4.840359560913905e-06, 0.0003662896964675783...
	rise_power	NaN	7.86713e-05	[7.924763904973348e-05, 0.00022636803988297093...
A2,ZN, fall_power		NaN	-4.36705e-06	[-4.0917817024074655e-06, 0.000287274152767297...
	rise_power	NaN	9.37976e-05	[9.467791446892202e-05, -2.9697067520466334e-0...
,A1,!A2	fall_power	NaN	3.23029e-05	[3.228303397738344e-05, 5.271234649423687e-07,...
	rise_power	NaN	-2.86602e-05	[-2.86192925503315e-05, -1.0786302709023273e-0...
,A2,!A1	fall_power	NaN	2.75222e-05	[2.791441165194605e-05, -1.0310684745203868e-0...
	rise_power	NaN	-2.50182e-05	[-2.5004590914418427e-05, -3.5513090962259144e...

Metrics characterized @ trans, load = 40ps, 1.7fF

# Prediction & Insight



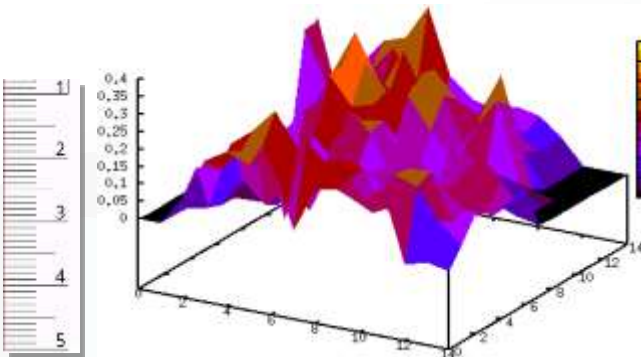
Temperature inversion



# Metric Applications

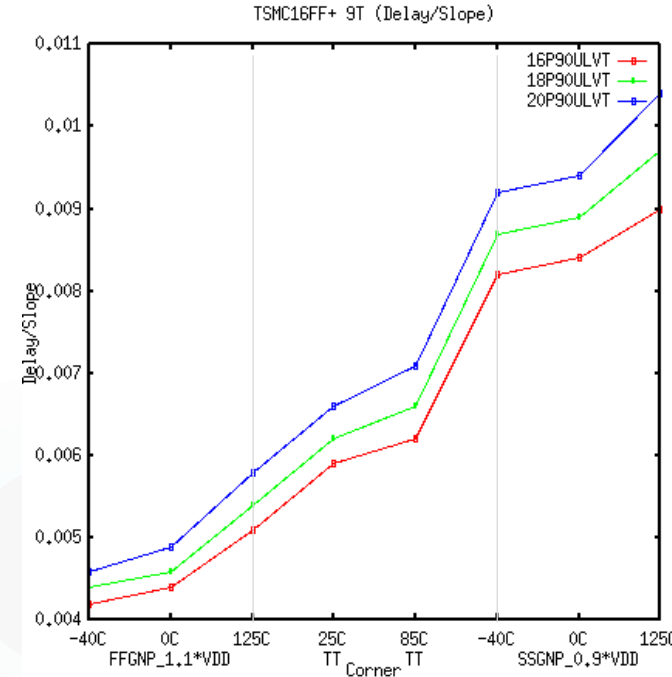
## Liberty Metric

Cell	FP	Slope	xD	Area	Current	Dmin	Dmax	DCmin	DCmax
AN2DQBWP20P90	an2d1	0.131	1.0	0.259	0.0160	0.021	0.043	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP16P90CPDLVT	an2d1	0.172	2.0	0.207	0.0165	0.019	0.035	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP16P90LVT	an2d1	0.177	2.0	0.259	0.0172	0.019	0.033	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP20P90CPDLVT	an2d1	0.157	2.0	0.207	0.0162	0.021	0.037	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP20P90CPDULVT	an2d1	0.204	2.0	0.207	0.0193	0.018	0.030	FF, 0.935, -40	SS, 0.765, 125
AN2DQBWP20P90LVT	an2d1	0.165	2.0	0.259	0.0175	0.019	0.034	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP7D5T16P96CPDLVT	an2d1	0.175	2.0	0.184	0.0135	0.021	0.037	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP7D5T20P96CPDULVT	an2d1	0.207	2.0	0.184	0.0160	0.021	0.033	FF, 0.935, -40	SS, 0.765, 125
AN2DQP75BWP20P90	an2d1	0.198	2.0	0.259	0.0218	0.020	0.040	FF, 0.935, -40	SS, 0.765, -40
AN2DQBWP16P90CPDULVT	an2d1	0.218	3.0	0.207	0.0199	0.018	0.029	FF, 0.935, -40	SS, 0.765, 125
AN2DQBWP16P90LVT	an2d1	0.230	3.0	0.259	0.0224	0.016	0.025	FF, 0.935, -40	SS, 0.765, 125
AN2DQBWP18P90LVT	an2d1	0.222	3.0	0.259	0.0218	0.016	0.026	FF, 0.935, -40	SS, 0.765, 125
AN2DQBWP20P90LVT	an2d1	0.214	3.0	0.259	0.0213	0.017	0.027	FF, 0.935, -40	SS, 0.765, 125
AN2DQBWP7D5T16P96CPDLVT	an2d1	0.222	3.0	0.184	0.0171	0.020	0.031	FF, 0.935, -40	SS, 0.765, 125
AN2DQP75BWP16P90CPDLVT	an2d1	0.260	3.0	0.207	0.0218	0.018	0.032	FF, 0.935, -40	SS, 0.765, -40



CPU Current Density

## Technology Trend Analysis



Corner Selection  
Library/Cell Selection

Instance	Cell	Delay	Dmin	Dmax
u_ca53_noram/u_ca53ifu_pf/avalid_if3_reg	SDFCNQND4BWP16P90LVT	0.047	0.017	0.029
u_ca53_noram/u_ca53ifu_pf/FE_OCPC219072	INVD6BWP16P90LVT	0.009	0.002	0.002
u_ca53_noram/u_ca53ifu_pf/FE_OCPC224282	BUFFD10BWP16P90LVT	0.022	0.005	0.008
u_ca53_noram/u_ca53ifu_pf/g479784	MUX2D8BWP16P90LVT	0.021	0.008	0.012
u_ca53_noram/u_ca53ifu_pf/FE_OCPC219057	INVD16BWP16P90LVT	0.006	0.002	0.002
u_ca53_noram/u_ca53ifu_pf/FE_OCPC219058	TNVD12BWP16P90LVT	0.029	0.002	0.002
u_ca53_noram/u_ca53ifu_pf/FE_RC_14708_0	ND2D6BWP16P90LVT	0.023	0.002	0.003
u_ca53_noram/u_ca53ifu_pf/FE_RC_14707_0	OAI21D8BWP16P90LVT	0.012	0.003	0.004
u_ca53_noram/u_ca53ifu_pf/n0001D35	NR2D8BWP16P90LVT	0.013	0.002	0.004
u_ca53_noram/u_ca53ifu_pf/n0026D478725	AN2D8BWP16P90LVT	0.020	0.007	0.011

## Identify room for improvement (Critical Path)

# Liberty Format

## Comment Syntax

```
/* ... */
```

## Single Attribute

```
name : value ;
```

## Complex Attribute

```
name (value1 [, value2, ...]) ;
```

## Group

```
type (name) { ... }
```

## Library Group Example

```
library (...) {  
    name : value ;  
    name (value1 [, value2, ...]) ;  
    type (...) {  
        name : value;  
        name (value1 [, value2, ...]) ;  
        type (...) {  
            ...  
        }  
    }  
}
```

```
library(ulvttt_0p35v_85c_typical_ccs) {  
    delay_model : table_lookup ;  
    time_unit : 1ns ;  
    voltage_unit : 1V ;  
    current_unit : 1mA ;  
    capacitive_load_unit(1, pf);  
    leakage_power_unit : 1nW ;  
    nom_process : 1 ;  
    nom_temperature : 85 ;  
    nom_voltage : 0.35 ;  
    operating_conditions(tt_0p35v_85c_typical) {  
        process : 1 ;  
        process_label : tt ;  
        temperature : 85 ;  
        voltage : 0.35 ;  
    }  
    lu_table_template(tmg_ntin_oload_9x8) {  
        variable_1 : input_net_transition ;  
        variable_2 : total_output_net_capacitance ;  
        index_1("1, 2, 3, 4, 5, 6, 7, 8, 9");  
        index_2("1, 2, 3, 4, 5, 6, 7, 8");  
    }  
    cell(INV1_UL) {  
        area : 0.226512 ;  
        cell_footprint : INV ;  
        cell_leakage_power : 69.744 ;  
        leakage_power() {  
            related_pg_pin : "VDD" ;  
            when : "!I&ZN" ;  
            value : "60.746" ;  
        }  
        ...  
        pin(I) {  
            capacitance : 0.0041901 ;  
            direction : input ;  
            ...  
        }  
    }  
}
```

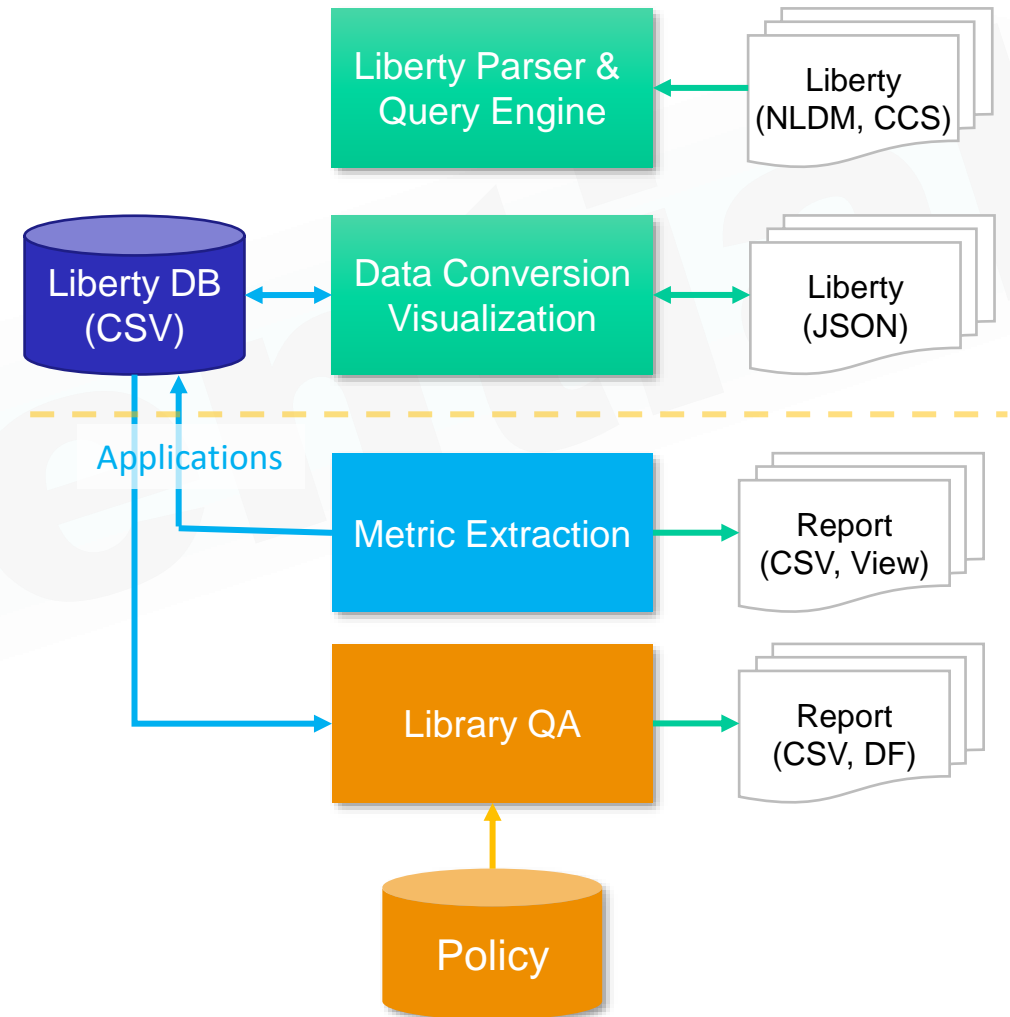
```
pin(ZN) {  
    direction : output ;  
    function : "(!I)" ;  
    max_capacitance : 0.33677 ;  
    max_transition : 0.52597 ;  
    min_capacitance : 0.0002 ;  
    related_power_pin : VDD ;  
  
    internal_power() {  
        related_pg_pin : "VDD" ;  
        related_pin : "I" ;  
        fall_power(pwr_tin_oload_9x8) {  
            index_1("0.0015, 0.0037, 0.008, 0.0164, ...");  
            index_2("0.0002, 0.0044133, 0.019203, ...");  
            values("-0.00013627, -0.00010, -9.61e-05, ...",\  
                "-0.00014769, -0.00011, -9.83e-05, ...",\  
                ...  
                "0.000973, 0.000858, 0.000639, ...");  
        }  
    }  
    timing() {  
        related_pin : "I" ;  
        timing_sense : negative_unate ;  
        timing_type : combinational ;  
  
        cell_fall(tmg_ntin_oload_9x8) {  
            index_1("0.0015, 0.0037, 0.008, 0.0164, ...");  
            index_2("0.0002, 0.0044133, 0.019203, ...");  
            values("0.0042712, 0.0076647, 0.018101, ...",\  
                "0.0050221, 0.0086186, 0.019159, ...",\  
                ...  
                "0.0352, 0.06013, 0.1127, 0.1758, ...");  
        }  
    }  
} /* end pin */  
} /* end cell */  
} /* end library */
```



# Liberty Utility Specification

## LibertyClass Functions:

- Convert liberty into JSON format
- Convert lookup table into numpy.array
- Convert lookup table into pandas.DataFrame
- Construct library DB and metric CSV from JSON
- Perform timing/power interpolation
- Perform timing/power regression (LS coefficient)
- Extract liberty metric (LUT)
- Extract liberty metric (LSC)
- Provide `get_*` APIs to access liberty data structure
- Provide `lookup_*` APIs to calculate/interpolate timing/power
- Provide `plot_*` APIs for visualization
- Provide `dump_*` APIs for data extraction/conversion





# Liberty Parser & Data Structure

```
import sys
sys.path.append({path_to_liberty_parser})
from LibertyClass import liberty as lutil alias

# load liberty
lib = 'tcbn16ffc1lbwp16p90cpdlvtssgnp0p36vm40c.lib'
lnode = lutil.read_lib(lib,gzFlag=False)
lnode.keys()
lnode['cell'].keys()

P,V,T = lnode['nom_process'], \
        lnode['nom_temperature'], \
        lnode['nom_voltage'] # library PVT

# grab information by dictionary structure
# cell node
[v for v in lnode['cell']] # cell list in library
cnode = lnode['cell']['AN2D1LVT']
cnode = lutil.get_cell(lnode,'AN2D1LVT')
cnode.keys()
cnode['name']
cnode['leakage_power']
cnode['cell_footprint']
cnode['area']
cnode['pin'].keys()

# pin node
inode = lnode['cell']['AN2D1LVT']['pin']['Z']
inode = cnode['pin']['Z']
inode.keys()
inode['name']
inode['direction']
inode['function']
inode['max_transition']
```

```
# timing table
tnodeL = lnode['timing '] # list of timing table
tnode = tnodeL[0] # 1st timing table
tnode.keys()
tnode['related_pin']
tnode['timing_sense']
tnode['timing_type']

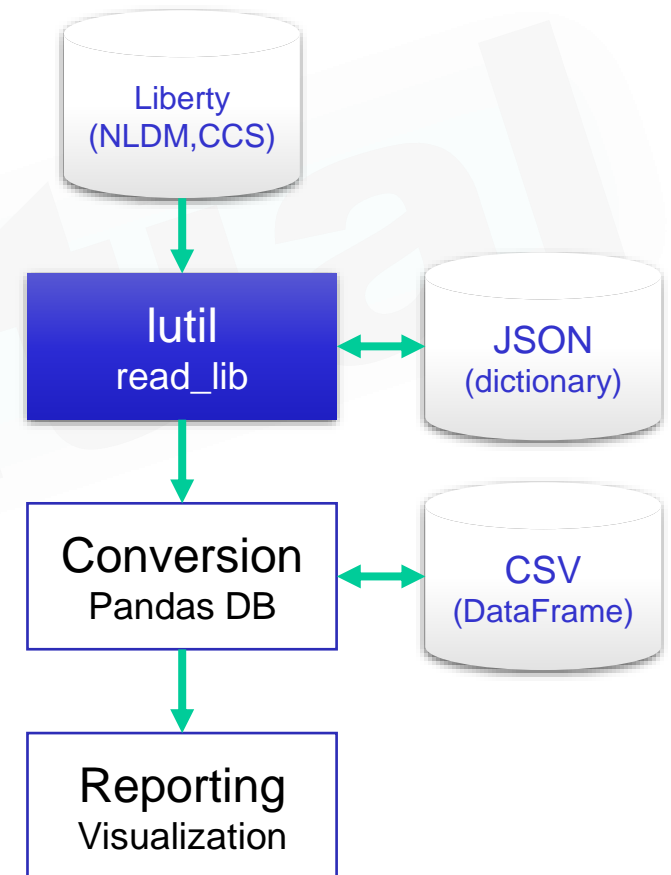
# delay lookup table
lut = tnode['cell_rise'] # a dictionary
lut.keys()
shape = len(lut['index_1']),len(lut['index_2'])
tran,load = lut['index_1'],lut['index_2']
values = np.array(lut['values']).reshape(shape)

# convert to pandas DataFrame
import pandas as pd
df = pd.DataFrame(values,columns=load,index=tran)
print(df)

# map into numpy array
import numpy as np
y,x,cfall = tnode['cell_fall'].values() # delay fall
y,x,cfall = map(np.array,tnode['cell_fall'].values())

# encapsulate tables indexed by arc and timing type
lutL = lutil.get_cell_timing(cnode) # query all tables
timing tables
[v for v in lutL]

# encapsulate timing table into dataframe
dt = lutil.get_cell_timing(cnode,todf=True) # as dataframe
print(dt)
```

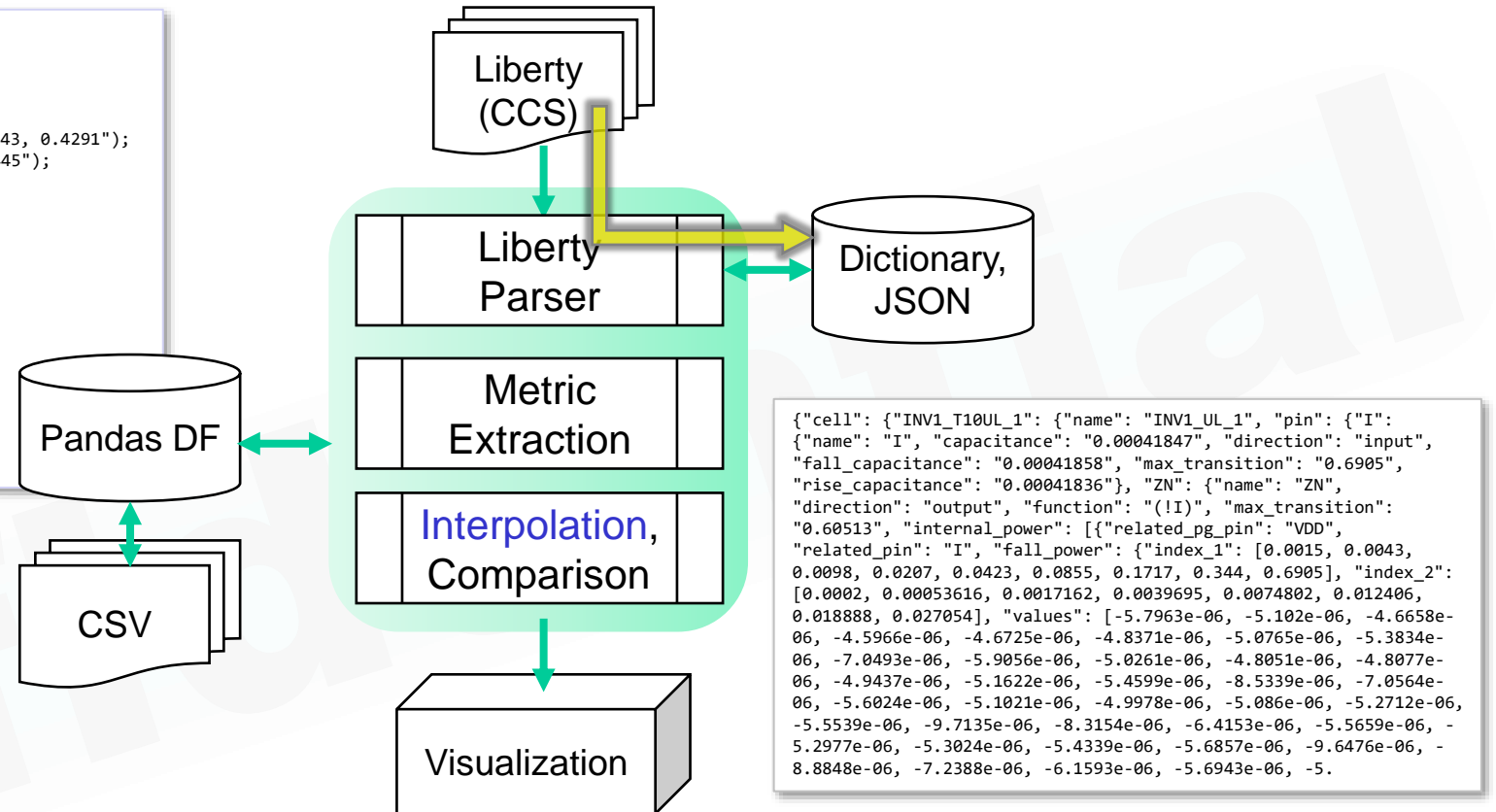


# Liberty to JSON

```
timing () {
  related_pin : "CP";
  timing_sense : non_unate;
  timing_type : rising_edge;
  cell_rise (delay_template_8x8) {
    index_1 ("0.0016, 0.00255, 0.00595, 0.0127, 0.0262, 0.0531, 0.10685, 0.2143, 0.4291");
    index_2 ("0.0002, 0.00043, 0.00094, 0.0023, 0.0044, 0.0095, 0.02056, 0.0445");
    values ( \
      "0.1037, 0.1075, 0.1141, 0.1257, 0.1476, 0.1924, 0.2837, 0.4947", \
      "0.1042, 0.1082, 0.1146, 0.1263, 0.1481, 0.1929, 0.2887, 0.4953", \
      "0.1065, 0.1099, 0.1164, 0.1281, 0.1499, 0.1948, 0.2905, 0.4971", \
      "0.1096, 0.1134, 0.1201, 0.1317, 0.1536, 0.1984, 0.2940, 0.5007", \
      "0.1163, 0.1217, 0.1268, 0.1384, 0.1603, 0.2051, 0.3008, 0.5073", \
      "0.1279, 0.1171, 0.1383, 0.1500, 0.1718, 0.2166, 0.3124, 0.5190", \
      "0.1478, 0.1513, 0.1582, 0.1699, 0.1918, 0.2366, 0.3324, 0.5393", \
      "0.1806, 0.1844, 0.1911, 0.2027, 0.2246, 0.2694, 0.3652, 0.5720", \
      "0.2257, 0.2295, 0.2362, 0.2479, 0.2698, 0.3146, 0.4105, 0.6174" \
    );
  };
}
```

```
# load CCS
import glob
libL = glob.glob(f'{path}/liberty/*.lib')
lnode = lutil.read_lib(libL[0],gzFlag=False)
lnode.keys()
[v for v in lnode['cell']]

# convert to JSON
for lib in libL:
  lnode = lutil.read_lib(lib,gzFlag=False)
  lname = lnode['library']
  lutil.dump_json(lnode,
    f'{path}/exercise/JSON/{lname}.json',
    cname_re='.*OR2.*|.*NAND2.*|.*INV1.*')
```



Characteristic, Trend, PPA

# Cell & Timing Table

```

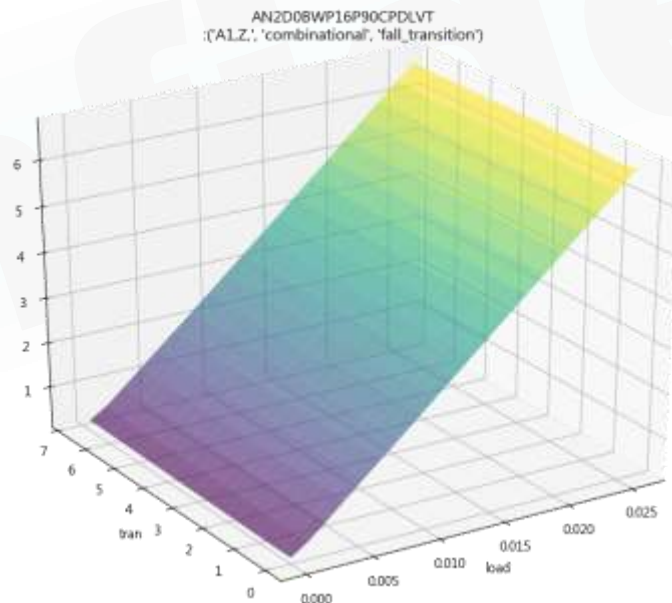
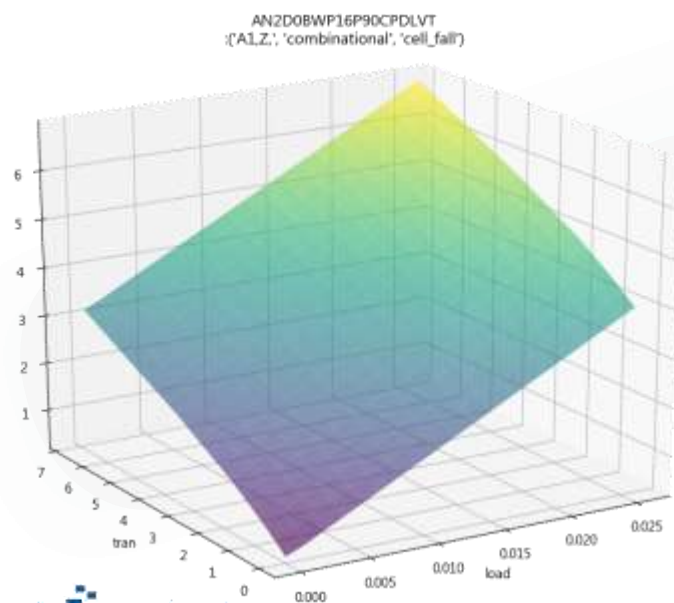
%% grab cell & timing table with API
cnodeL = lutil.get_cells(lnode,'AN2D1 ') # list of cell node
cnode = cnodeL[0]

tnodeL = cnode['pin']['Z']['timing'] # list of timing nodes

lutT = lutil.get_cell_timing(cnode) # grab timing table as dictionary
[v for v in lutT] # key value of (arc,ttype,ctype)

lutT = lutil.get_cell_timing(cnode,ctype='fall') # delay & transition fall table
[v for v in lutT]

lutil.plot_cell_timing(cnode) # grab all timing tables
lutil.plot_cell_timing(cnode,ctype='cell_fall') # grab delay table
lutil.plot_cell_timing(cnode,ctype='fall_transition') # grab transition table
lutil.plot_lut(lutT,keys=('A1,Z,', 'combinational', 'cell_fall')) # visualization
    
```



Library Node: **lnode**  
(dictionary)

`lnode.keys()`

Cell Node: **cnode** ← `lnode['cell']`  
(dictionary)

`cnode.keys()`  
`cnode['pin'].keys()`

Pin Node: **inode** ← `cnode['pin']['Z']`  
(dictionary)

`inode.keys()`

Timing Tables: **tnodeL** ← `inode['timing']`  
(list)

Timing Node: **tnode** ← `tnodeL[0]`  
(dictionary)

`tnode.keys()`

Lookup Table Node: **lut** ← `tnode['cell_rise']`  
(dictionary)

`lut.keys()`

index\_2:  
load/clock

index\_1:  
trans/data

lut

# Pandas DataFrame Conversion

```
import pandas as pd
lnode = lutil.load_json('ulvttt_0p33v_85c_typical_ccs.json')
[v for v in lnode['cell']] # cell list
cnode = lutil.get_cell(lnode,'INV1_UL_1') # lnode['cell']['INV1_UL_1']

lutL = lutil.get_cell_timing(cnode)
[v for v in lutL] # all available hash keys

lut = lutL[('I,ZN,', 'combinational', 'cell_rise')] # delay rise lookup table
tran,load,tr = map(np.array,lut.values())
tran,load,tf = map(np.array,lutL[('I,ZN,', 'combinational', 'cell_fall')].values())
tr = tr.reshape(len(tran),len(load))
tf = tf.reshape(len(tran),len(load))

tran = (tran*1000).round(3) # ps
load = (load*1000).round(3) # fF
dr = pd.DataFrame(tr,columns=load,index=tran)
df = pd.DataFrame(tf,columns=load,index=tran)
(dr/df).round(2) # delay imbalance

# INV1_UL_1
lutL = lutil.get_cell_timing(liberty.get_cell(lnode,'INV1_UL_1'))
tran,load,tr = map(np.array,lutL[('I,ZN,', 'combinational', 'cell_rise')].values())
tran,load,tf = map(np.array,lutL[('I,ZN,', 'combinational', 'cell_fall')].values())
tr = tr.reshape(len(tran),len(load))
tf = tf.reshape(len(tran),len(load))

tran = (tran*1000).round(3) # ps
load = (load*1000).round(3) # fF
dr = pd.DataFrame(tr,columns=load,index=tran)
df = pd.DataFrame(tf,columns=load,index=tran)

(dr/df).round(2) # delay imbalance
```

INV1_UL_1 delay imbalance								
	0.200	0.536	1.716	3.970	7.480	12.406	18.888	27.054
1.5	0.95	0.94	0.93	0.92	0.92	0.92	0.92	0.92
4.3	0.94	0.94	0.93	0.92	0.92	0.92	0.92	0.92
9.8	0.93	0.93	0.93	0.92	0.92	0.92	0.92	0.92
20.7	0.92	0.92	0.92	0.92	0.92	0.92	0.92	0.92
42.3	0.87	0.90	0.92	0.92	0.92	0.92	0.92	0.92
85.5	0.80	0.85	0.90	0.92	0.92	0.92	0.92	0.92
171.7	0.70	0.77	0.86	0.90	0.91	0.92	0.92	0.92
344.0	0.56	0.66	0.79	0.86	0.90	0.91	0.91	0.92
690.5	0.38	0.52	0.69	0.79	0.85	0.88	0.90	0.91

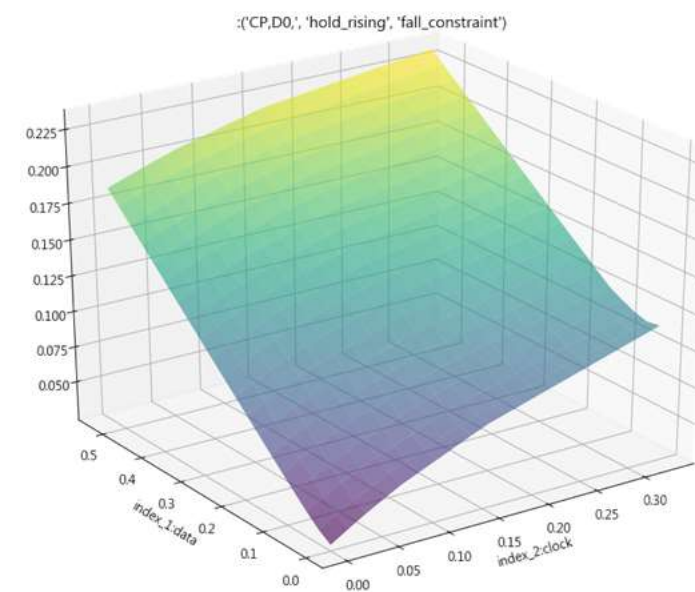
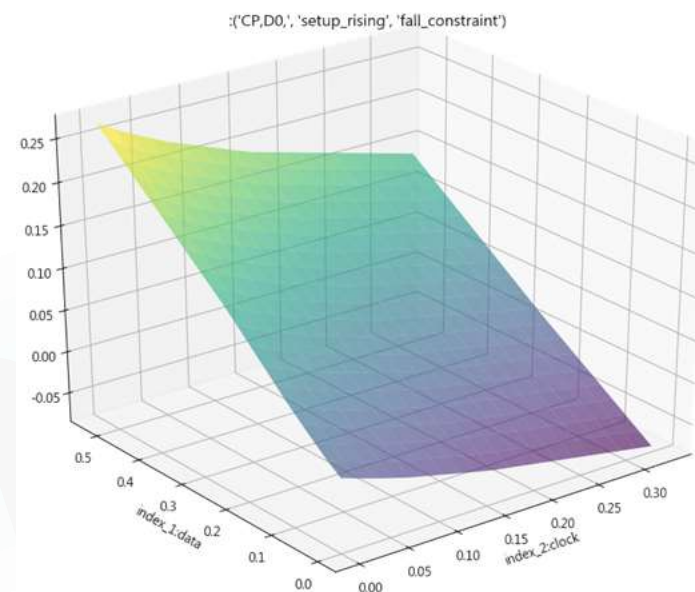
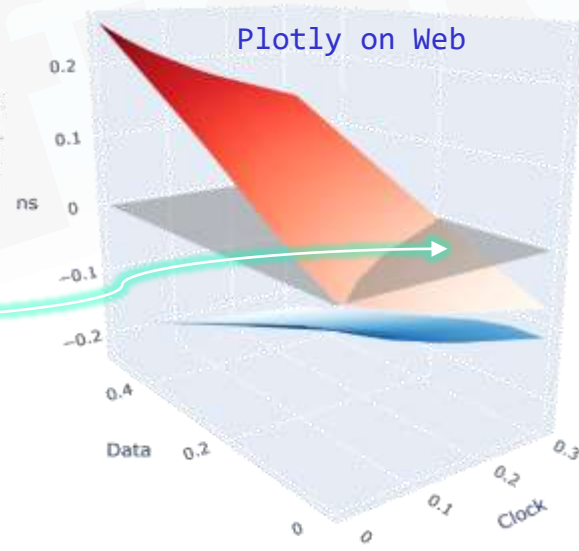
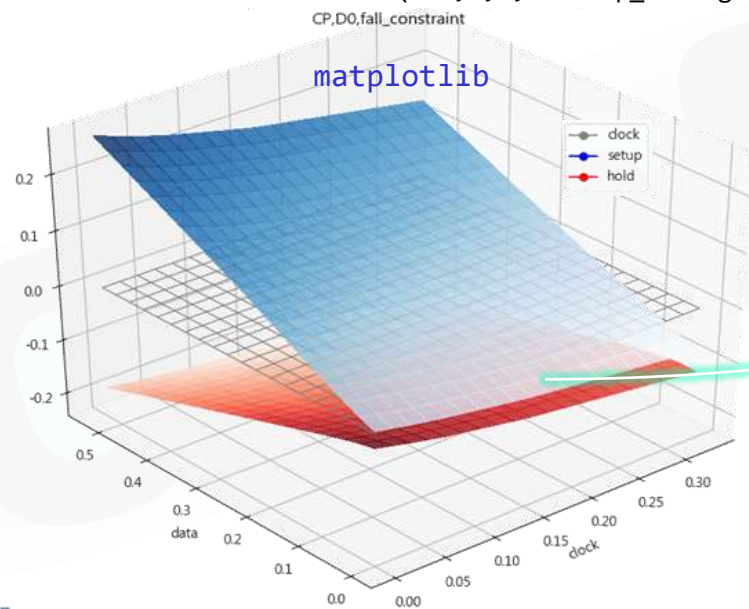
INV1_UL_1 delay imbalance								
	0.200	0.483	1.475	3.370	6.321	10.463	15.913	22.779
1.5	1.05	1.06	1.06	1.06	1.07	1.06	1.07	1.07
4.3	1.04	1.04	1.05	1.06	1.06	1.06	1.06	1.07
9.8	1.02	1.03	1.04	1.05	1.06	1.06	1.06	1.07
20.7	1.00	1.01	1.03	1.04	1.05	1.06	1.06	1.06
42.3	0.97	0.99	1.01	1.03	1.04	1.05	1.05	1.06
85.5	0.92	0.95	0.99	1.01	1.02	1.03	1.04	1.05
171.7	0.85	0.89	0.95	0.99	1.00	1.02	1.03	1.04
344.0	0.76	0.82	0.90	0.95	0.98	1.00	1.01	1.02
690.5	0.63	0.72	0.83	0.90	0.94	0.97	0.99	1.00

# Timing & Power Visualization

```
# load liberty from JSON
lib = 'ulvttt_0p33v_85c_typical_ccs.json'
lnode = lutil.load_json(lib)
cnode = lutil.get_cell(lnode, 'DFFQD1')
lutT = lutil.get_cell_timing(cnode) # all timing tables
lutP = lutil.get_cell_power(cnode) # all power tables

# plot cell delay & transition
lutil.plot_lut(lutT, keys=('CK,Q,D', 'rising_edge', 'cell_rise'))
lutil.plot_lut(lutT, keys=('CK,Q,D', 'rising_edge', 'rise_transition'))

# plot timing constraint
lutil.plot_lut(lutT, keys=[('CK,D,', 'hold_rising', 'rise_constraint'),
                           ('CK,D,', 'setup_rising', 'rise_constraint')])
lutil.plot_lut(lutT, keys=[('CK,D,', 'hold_rising', 'fall_constraint'),
                           ('CK,D,', 'setup_rising', 'fall_constraint')])
```





# Timing/Power Table Lookup

```
lutT = lutil.get_cell_timing(cnode)
[v for v in lutT]

# decapsulate lookup table
lut = lutT[('A1,ZN,', 'combinational', 'cell_rise')]
y,x,v = map(np.array,lut.values())

# convert lut into dataframe
d = lutil.lut2df(lut)

# timing interpolation based on the specified transition & load
lutil.table_lookup(lut,trans=0.0207,load=0.0010072,dflag=True)

# lookup cell arc timing based on the specified transition & load
lutil.lookup_cell_timing(cnode,ctype='cell_rise',
    trans=0.0207,load=0.0010072,dflag=True)

lutil.lookup_cell_timing(cnode,ctype='cell_fall',
    trans=0.0207,load=0.0010072,dflag=True)
```

transition

	load							
	index_2	0.000200	0.000379	0.001007	...	0.006699	0.010150	0.014497
	index_1				...			
	0.0015	0.008991	0.010693	0.016545	...	0.068989	0.10068	0.14068
	0.0043	0.010278	0.012048	0.017969	...	0.070451	0.10222	0.14225
	0.0098	0.012435	0.014374	0.020632	...	0.073292	0.10503	0.14510
	0.0207	0.015996	0.018175	0.025128	...	0.078947	0.11063	0.15071
	0.0423	0.021470	0.024484	0.032539	...	0.089738	0.12184	0.16192
	0.0855	0.027694	0.032269	0.044478	...	0.108550	0.14241	0.18354
	0.1717	0.034010	0.040676	0.058925	...	0.140210	0.17730	0.22168
	0.3440	0.038747	0.048142	0.074508	...	0.193120	0.23657	0.28577
	0.6905	0.037839	0.050992	0.087983	...	0.262930	0.32811	0.39416

[9 rows x 8 columns]

interpolation(0.0207,0.0010072)= 0.025128

A1->ZN@()= 0.025128

	load							
	index_2	0.000200	0.000379	0.001007	...	0.006699	0.010150	0.014497
	index_1				...			
	0.0015	0.009524	0.011232	0.017124	...	0.069588	0.10132	0.14144
	0.0043	0.010827	0.012572	0.018499	...	0.070966	0.10275	0.14272
	0.0098	0.013105	0.014983	0.021179	...	0.073836	0.10558	0.14560
	0.0207	0.016830	0.018942	0.025760	...	0.079452	0.11118	0.15114
	0.0423	0.022852	0.025595	0.033373	...	0.090275	0.12237	0.16238
	0.0855	0.030123	0.034314	0.045807	...	0.109250	0.14290	0.18403
	0.1717	0.037926	0.044104	0.061178	...	0.141050	0.17805	0.22247
	0.3440	0.044806	0.053520	0.078357	...	0.194530	0.23748	0.28641
	0.6905	0.047286	0.059499	0.094357	...	0.265280	0.32978	0.39562

[9 rows x 8 columns]

interpolation(0.0207,0.0010072)= 0.02576

A2->ZN@()= 0.02576

# Interpolation

```
from scipy import interpolate
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

lutT = lutil.get_cell_timing(liberty.get_cell(lnode,'INV1_UL'))
tran,load,tr = map(np.array,lutT[('I,ZN','combinational','rise_transition')].values())

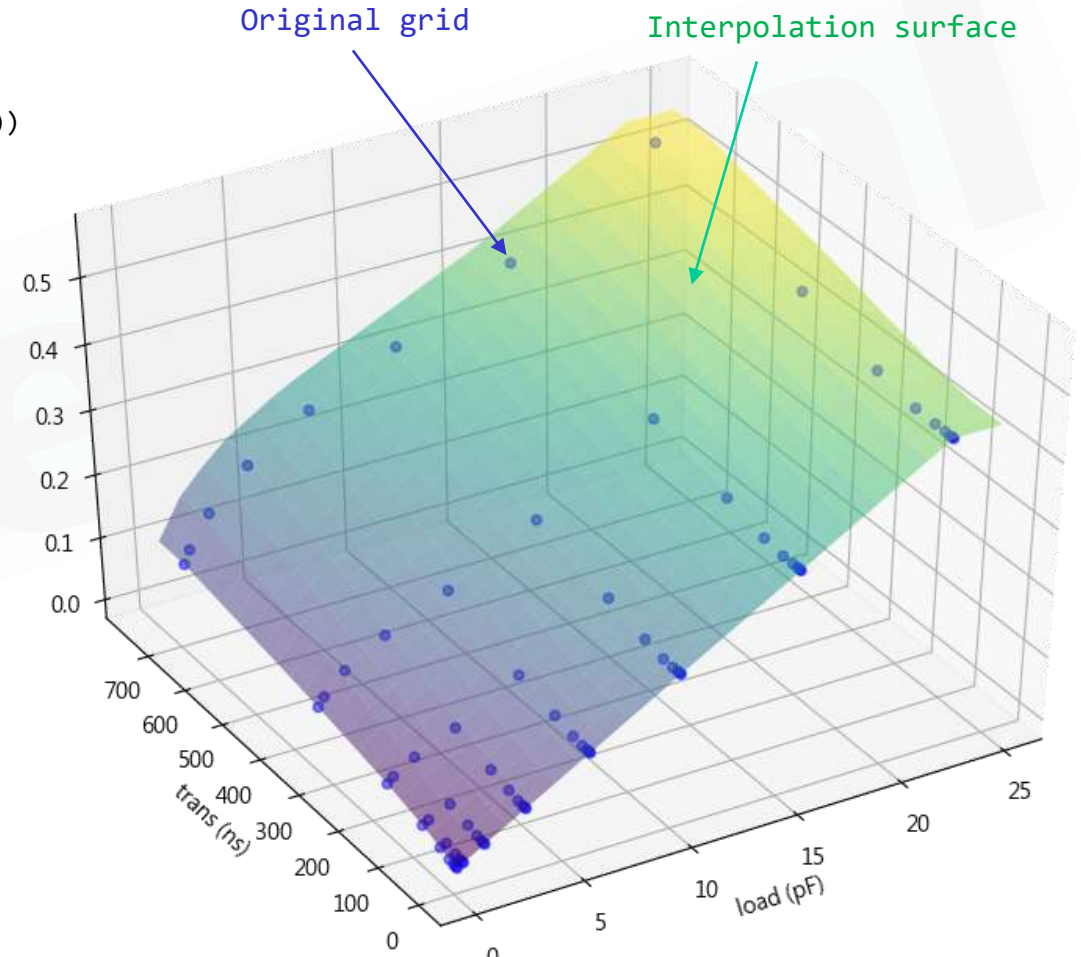
tran = tran*1000 # ps
load = load*1000 # fF
tr = tr.reshape(len(tran),len(load))

# fine-grained grid
ptran = np.arange(tran[0],tran[-1]*1.1,(tran[-1]-tran[0])/20)
pload = np.arange(load[0],load[-1]*1.1,(load[-1]-load[0])/20)

# interpolation
ip = interpolate.interp2d(load,tran,tr,kind='linear')
pz = ip(pload,ptran) # predict

x,y = np.meshgrid(load,tran) # original grid
px,py = np.meshgrid(pload,ptran) # prediction grid

f = plt.figure(figsize=(8,6))
ax = Axes3D(f)
ax.scatter(x,y,tr,color='b',alpha=0.5,label='LUT ') # original grid
ax.plot_surface(px,py,pz,cmap=plt.cm.viridis,alpha=0.5,label='Interpolation')
ax.set_xlabel('load (pF)')
ax.set_ylabel('trans (ns)')
ax.view_init(60,250)
plt.show()
```



# Batch Comparison

```
import numpy as np
ts = lutL(['CP,D0,', 'setup_rising', 'rise_constraint'])
th = lutL(['CP,D0,', 'hold_rising', 'rise_constraint'])

# sweep data=10ps~400ps per 10ps @clock=100ps
data,cock = np.arange(10,410,10),[0.1]
y,x,z = ts['index_1'], ts['index_2'], np.array(ts['values'])
lutil.interpolate_luty,x,z,data,clock)

### convert to DataFrame & comparison
import glob, re
libL = glob.glob('lib/*.lib')
tagL, dfL = [],[]
for lib in libL:
    tag = re.match('.*tcn(.*)lib',lib).groups()[0]
    lnode = lutil.read_lib(lib,gzFlag=False)
    d = lutil.lib2df(lnode)
    tagL += [tag]
    dfL += [d]
df = pd.concat(dfL,keys=tagL,names=['lib'])
df.iloc[:, :1]
```

```
###
df = df.reset_index().set_index(['lib','cell','ctype','ttype'])
d1 = df.loc['lvtssgnp0p36vm40c','AN2D1LVT','cell_rise','combinational']
D2 = df.loc['ulvtffgnp0p44vm40c','AN2D1LVT','cell_rise','combinational']

# comparison based on the same transition range 10ps~400ps @the proper clock=10ps
data,clock = [0.01,0.1,0.2,0.3,0.4],[0.1]
y,x,z = np.array(d1['index_1']),np.array(d1['index_2']),np.array(d1['values'])
ts1 = lutil.interpolate_lut(y,x,z,[0.01,0.1,0.2,0.3,0.4],[0.1])

y,x,z = np.array(d2['index_1']),np.array(d2['index_2']),np.array(d2['values'])
ts2 = lutil.interpolate_lut(y,x,z,[0.01,0.1,0.2,0.3,0.4],[0.1])
```

					index_1
lib	cell	arc	ttype	ctype	
lvtssgnp0p36vm40c	AN2D1LVT	A1,Z,	combinational	rise_transition	[0.0039, 0.0317, 0.0872, ...
				cell_fall	[0.0039, 0.0317, 0.0872, ...
				fall_transition	[0.0039, 0.0317, 0.0872, ...
				cell_rise	[0.0039, 0.0317, 0.0872, ...
lvttt0p4v25c	AN2D1LVT	A1,Z,	combinational	rise_transition	[0.0039, 0.0317, 0.0872, ...
				cell_fall	[0.0039, 0.0317, 0.0872, ...
				fall_transition	[0.0039, 0.0317, 0.0872, ...
				cell_rise	[0.0039, 0.0317, 0.0872, ...
ulvtffgnp0p44vm40c	AN2D1ULVT	A1,Z,	combinational	rise_transition	[0.0039, 0.0317, 0.0872, ...
				cell_fall	[0.0039, 0.0317, 0.0872, ...
				fall_transition	[0.0039, 0.0317, 0.0872, ...
				cell_rise	[0.0039, 0.0317, 0.0872, ...

# Visualization & Comparison

```
cnode = lutil.get_cell(lnode,'DFQD1_UL')
lutT = lutil.get_cell_timing(cnode)
[v for v in lutT]
```

# combine constraint rise & fall

```
keyL = [('CK,CK,', 'min_pulse_width', 'rise_constraint'),
        ('CK,CK,D', 'min_pulse_width', 'fall_constraint')]
lutil.plot_lut(lutT,keys= keyL)
```

# combine delay rise & fall

```
keyL = [('CK,Q,D', 'rising_edge', 'cell_rise'),
        ('CK,Q,!D', 'rising_edge', 'cell_fall')]
lutil.plot_lut(lutT,keys= keyL)
```

# combine setup & hold

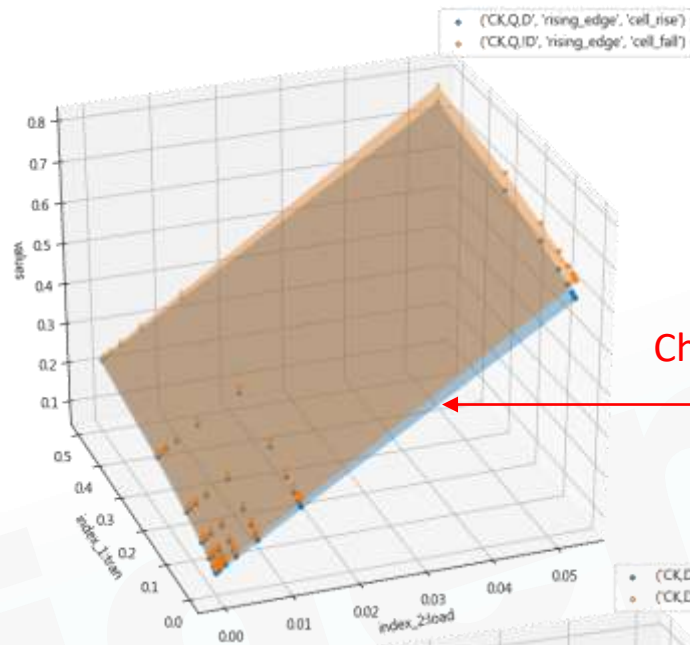
```
keyL = [('CK,D,', 'hold_rising', 'rise_constraint'),
        ('CK,D,', 'setup_rising', 'rise_constraint')]
lutil.plot_lut(lutT,keys= keyL,xlabel=('clock','data'))
```

# plot cell timing API

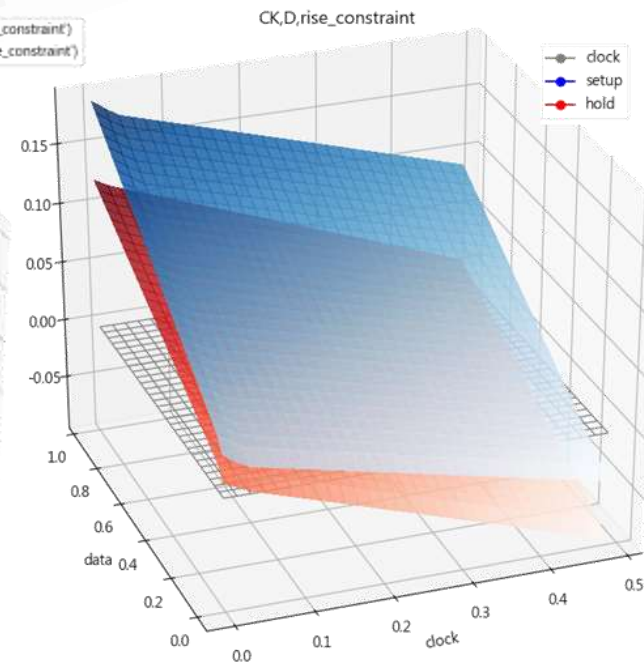
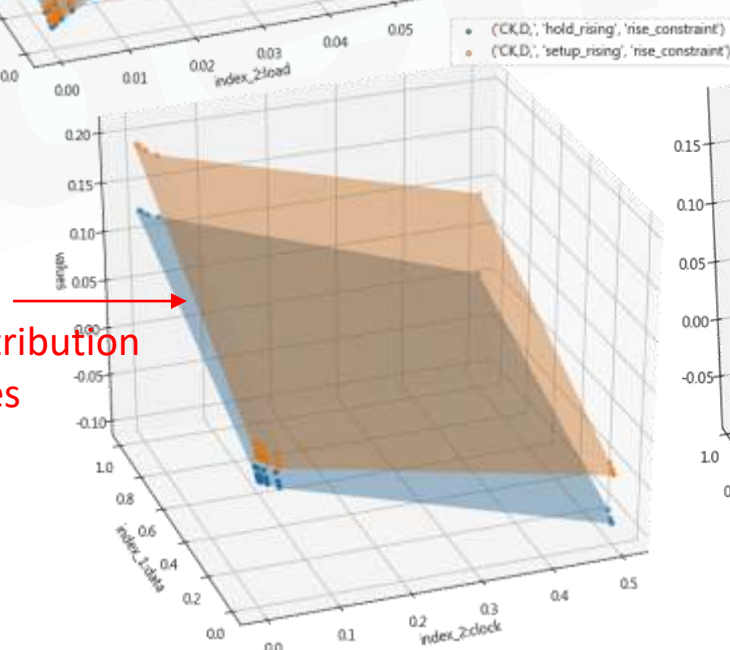
```
lutil.plot_cell_timing(cnode,arc='CK,Q,',ctype='cell')
```

```
lutil.plot_cell_timing(cnode,
    arc='CK,D,',ctype='rise_constraint',
    xlabel=('clock','data'))
```

```
lutil.plot_cell_constraint(cnode,
    arc='CK,D',ctype='rise_constraint')
```



Check the distribution  
of index ranges

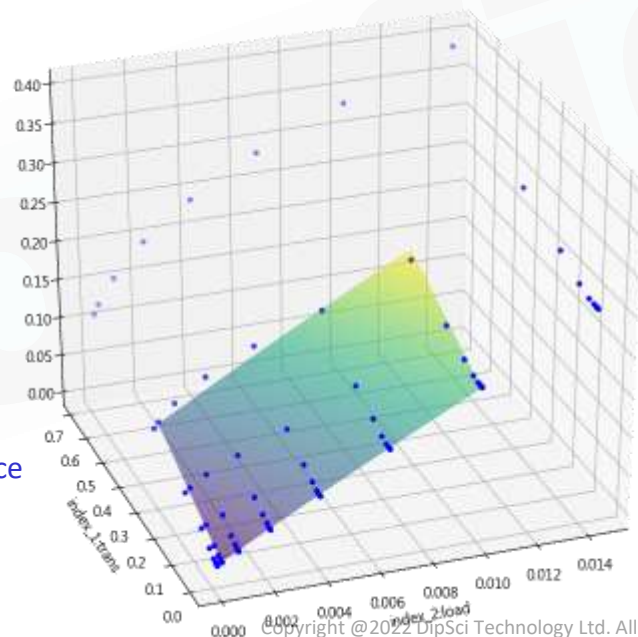


# Driving Slope

Slope := loading/transition

```
def lutSlope(lut):
    d = lutil.lut2df(lut)
    delta_load = d.columns[-2]-d.columns[1]
    delta_tran = d.iloc[1:-2,-2]-d.iloc[1:-2,1]
    s = delta_load/delta_tran
    return s.mean()
```

```
lnode = libDB['ulvttt_0p33v_85c_typical_ccs']
cnode = lnode['cell']['NAND2_UL']
lutT = lutil.get_cell_timing(cnode)
tr = lutil.lut2df(lutT[('A1,ZN','combinational','rise_transition')])
tf = lutil.lut2df(lutT[('A1,ZN','combinational','fall_transition')])
sr,sf = lutil.lookup_cell_slope(cnode,dflag=True)
```



Driving slope surface

lookup table: ('A1,ZN','combinational','fall\_transition'), slope=0.030267

index_2	0.000200	0.000379	0.001007	0.002207	0.004076	0.006699	0.010150	0.014497
index_1	0.0015	0.018769	0.024801	0.045968	0.086483	0.14956	0.23825	0.35457
0.0043	0.018816	0.024794	0.045979	0.086449	0.14961	0.23806	0.35416	0.50100
0.0098	0.019326	0.025249	0.046107	0.086573	0.14954	0.23819	0.35390	0.50119
0.0207	0.020805	0.026734	0.047298	0.087011	0.14954	0.23825	0.35458	0.50112
0.0423	0.023963	0.029383	0.050440	0.089751	0.15132	0.23851	0.35436	0.50108
0.0855	0.034962	0.039962	0.056701	0.095947	0.15737	0.24292	0.35700	0.50206
0.1717	0.052500	0.059251	0.078376	0.109870	0.16930	0.25581	0.36837	0.51063
0.3440	0.079855	0.089415	0.116250	0.153520	0.20156	0.28021	0.39359	0.53631
0.6905	0.123240	0.137060	0.175010	0.227640	0.28910	0.35813	0.45092	0.58412

lookup table: ('A2,ZN','combinational','fall\_transition'), slope=0.030113

index_2	0.000200	0.000379	0.001007	0.002207	0.004076	0.006699	0.010150	0.014497
index_1	0.0015	0.018751	0.024799	0.046008	0.086478	0.14939	0.23795	0.35467
0.0043	0.018770	0.024785	0.046008	0.086444	0.14949	0.23807	0.35411	0.50147
0.0098	0.019109	0.025096	0.046126	0.086475	0.14934	0.23801	0.35431	0.50034
0.0207	0.020098	0.026076	0.046907	0.086821	0.14958	0.23814	0.35438	0.50169
0.0423	0.021939	0.027725	0.048988	0.088639	0.15072	0.23848	0.35446	0.50121
0.0855	0.030324	0.035371	0.053351	0.092982	0.15474	0.24138	0.35669	0.50146
0.1717	0.046197	0.052202	0.070547	0.103380	0.16309	0.25057	0.36416	0.50768
0.3440	0.072674	0.080559	0.103590	0.139310	0.18867	0.26746	0.38153	0.52613
0.6905	0.113290	0.124930	0.157470	0.204240	0.26191	0.33204	0.42651	0.55984

Falling slope



# Table Lookup & Metric Extraction

```
%% timing lookup
lib = 'ulvttt_0p33v_85c_typical_ccs.json'
lnode = lutil.load_json(lib)
cnode = lnode['cell']['INV1_UL' ]

# grab timing table
lutL = lutil.get_cell_timing(cnode)
tran,load,cr = map(np.array,lutL[('I,ZN,', 'combinational', 'rise_transition')].values())
cr = cr.reshape(len(tran),len(load))
v = lutil.table_lookup(lut,0.04,0.004,dflag=True) # 40ps, 4fF

# grab cell information with API
c,s = lutil.lookup_cell_pincap(cnode,dflag=True)
l,s = lutil.lookup_cell_leakage(cnode,dflag=True)
tr,s = lutil.lookup_cell_timing(cnode,ttype='rise_transition',trans=0.04,load=0.004,dflag=True)
tf,s = lutil.lookup_cell_timing(cnode,ttype='fall_transition',trans=0.04,load=0.004,dflag=True)
dr,s = lutil.lookup_cell_timing(cnode,ttype='cell_rise',trans=0.04,load=0.004,dflag=True)
df,s = lutil.lookup_cell_timing(cnode,ttype='cell_fall',trans=0.04,load=0.004,dflag=True)
pr,s = lutil.lookup_cell_power(cnode,ttype='rise_power',trans=0.04,load=0.004,dflag=True)
pf,s = lutil.lookup_cell_power(cnode,ttype='fall_power',trans=0.04,load=0.004,dflag=True)

%% metric extraction, cell,fp,leak,[tr,tf,dr,df,pr,pf]
lib = 'ulvttt_0p33v_85c_typical_ccs.json'
lnode = lutil.load_json(lib)

#cell fp leak [tr,tf,dr,df,pr,pf]
d = lutil.dump_cells(lnode,'INV1_UL_1')

# metrix extraction: all cells
d = lutil.dump_cells(lnode,.*')
d.round(4)
```

cell	fp	area	leak	tr	tf	dr	df	pr	pf
INV1_UL_1	INV	0.0348	4.7055	0.0082	0.0086	0.0081	0.0085	0.0000	-0.0
INV1_UL_1	INV	0.0348	3.7393	0.0091	0.0083	0.0090	0.0085	0.0001	-0.0
NAND2V1_UL_1	ND2	0.0523	5.1413	0.0103	0.0188	0.0093	0.0172	0.0001	-0.0
NAND2V1_UL_1	ND2	0.0523	4.1366	0.0115	0.0182	0.0103	0.0170	0.0001	-0.0
NOR2V1_UL_1	NR2	0.0523	5.3066	0.0187	0.0108	0.0168	0.0096	0.0001	-0.0
NOR2V1_UL_1	NR2	0.0523	4.1848	0.0215	0.0106	0.0193	0.0097	0.0001	-0.0

# Design & Library Metric Extraction

```
timing () {
  related_pin : "CP";
  timing_sense : non_unate;
  timing_type : rising_edge;
  cell_rise (delay_template_8x8) {
    index_1 ("0.0016, 0.00255, 0.00595, 0.0127, 0.0262, 0.0531, 0.10685, 0.2143, 0.4291");
    index_2 ("0.0002, 0.00043, 0.00094, 0.0023, 0.0044, 0.0095, 0.02056, 0.0445");
    values ( \
      "0.1037, 0.1075, 0.1141, 0.1257, 0.1476, 0.1924, 0.2837, 0.4947", \
      "0.1042, 0.1082, 0.1146, 0.1263, 0.1481, 0.1929, 0.2887, 0.4953", \
      "0.1065, 0.1099, 0.1164, 0.1281, 0.1499, 0.1948, 0.2905, 0.4971", \
      "0.1096, 0.1134, 0.1201, 0.1317, 0.1536, 0.1984, 0.2940, 0.5007", \
      "0.1163, 0.1217, 0.1268, 0.1384, 0.1603, 0.2051, 0.3008, 0.5073", \
      "0.1279, 0.1171, 0.1383, 0.1500, 0.1718, 0.2166, 0.3124, 0.5190", \
      "0.1478, 0.1513, 0.1582, 0.1699, 0.1918, 0.2366, 0.3324, 0.5393", \
      "0.1806, 0.1844, 0.1911, 0.2027, 0.2246, 0.2694, 0.3652, 0.5720", \
      "0.2257, 0.2295, 0.2362, 0.2479, 0.2698, 0.3146, 0.4105, 0.6174" \
    );
  };
};
```

Liberty  
(CCS)

Liberty  
Parser

Dictionary,  
JSON

Metric  
Extraction

rule check, operations  
Comparison

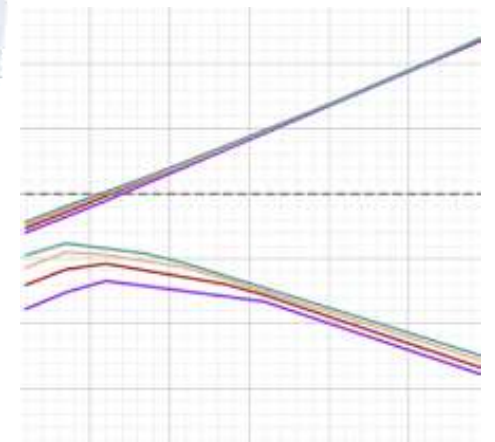
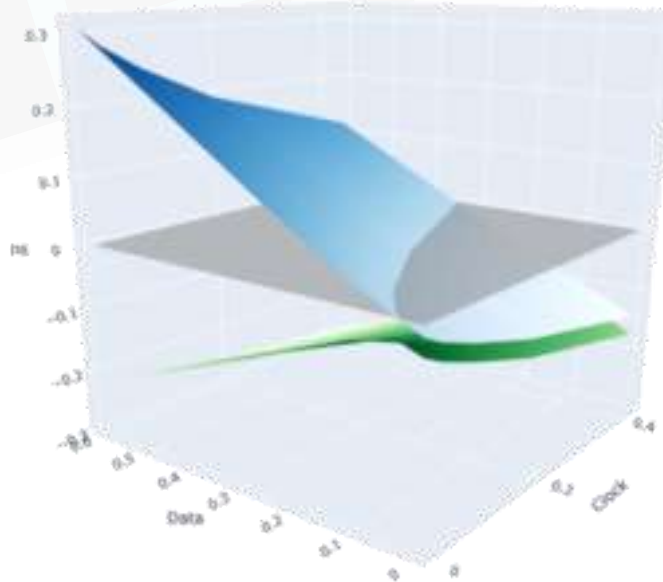
Visualization

Pandas DF

CSV

```
index,cell,src,pin,ctype,ttype,when,index_1,index_2,values
0,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00020,0.1037
1,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00043,0.1075
2,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00094,0.1141
3,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00203,0.1257
4,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00440,0.1476
5,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.00950,0.1924
6,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.02056,0.2837
7,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00160,0.04450,0.4947
8,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00255,0.00020,0.1042
9,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.00255,0.00043,0.1082
...
62,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.2143,0.02056,0.3652
63,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.2143,0.04450,0.5720
64,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00020,0.2257
65,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00043,0.2295
66,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00094,0.2362
67,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00230,0.2479
68,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00440,0.2690
69,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.00950,0.3146
70,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.02060,0.4105
71,BDFULVT,CP,Q0,cell_rise,rising_edge,,0.4291,0.04450,0.6174
```

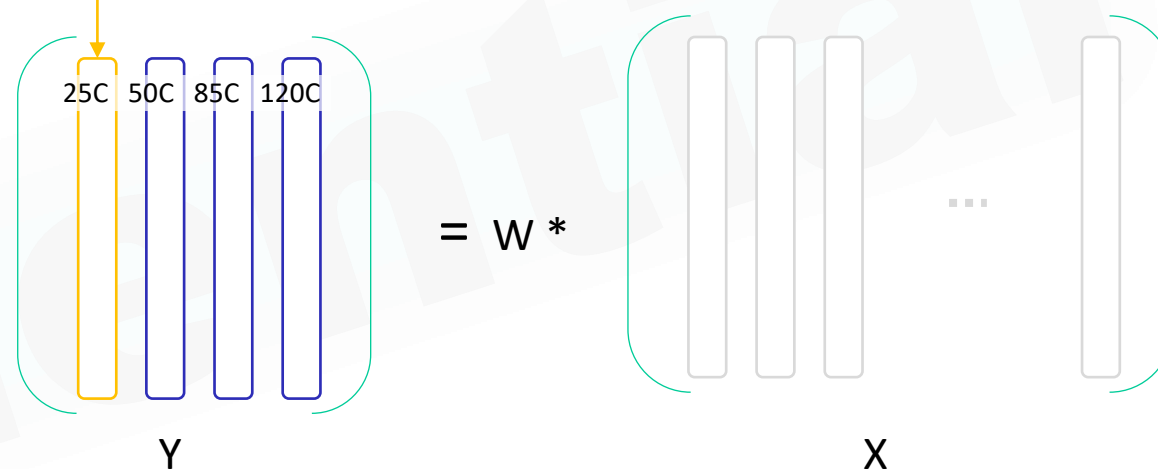
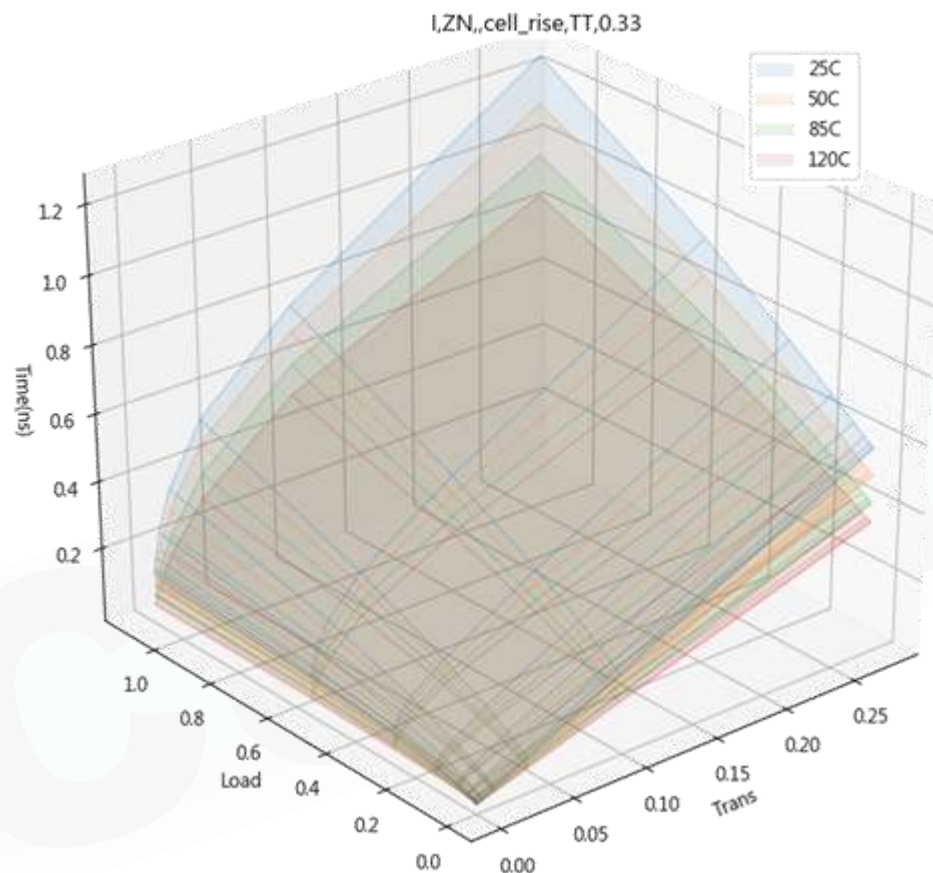
Sweep transition/PVT for comparison



Characteristic, Trend, PPA

# Machine-learning for Physical Design

T	index_1	index_2	values
25	[0.0016, 0.006, 0.0147, ...]	[0.0002, 0.000562037, ...]	[0.00766415, 0.00858535, 0.0108921, ...]
50	[0.0016, 0.006, 0.0147, ...]	[0.0002, 0.000562037, ...]	[0.00674201, 0.00755051, 0.0095632, ...]
85	[0.0016, 0.006, 0.0147, ...]	[0.0002, 0.000562037, ...]	[0.00577513, 0.00646485, 0.0081795, ...]
120	[0.0016, 0.006, 0.0147, ...]	[0.0002, 0.000562037, ...]	[0.00506849, 0.00567545, 0.0071815, ...]



Temperature regression

$$Y = w_1 * T + w_2 * \text{tran} + w_3 * \text{tran}^2 + w_4 * \text{load} + w_5 * \text{load}^2 + w_6 * \text{tran} * \text{load} + w_7 * \text{tran}^2 * \text{load} + w_8 * \text{tran} * \text{load}^2 + b$$

# LS Regression

```
# predict timing & power vectors from LS coefficient
T = np.array(
    [np.ones(gx.shape)]+
    [gx**i for i in range(1,order+1)]+
    [gy**i for i in range(1,order+1)]+
    [gx**i * gy**j for i,j in mxy] # covariate (order-1)
).T
C,_,_,_ = scipy.linalg.lstsq(T,z) # LS regression
p = np.dot(T,C) # predict with LS coefficient
```

$$\text{Predict} = \begin{pmatrix} 1 & X & X^2 & Y & Y^2 & XY \\ & T & & & & \end{pmatrix} * [C_0, C_1, C_2, C_3, C_4, C_5]$$

LS coefficient

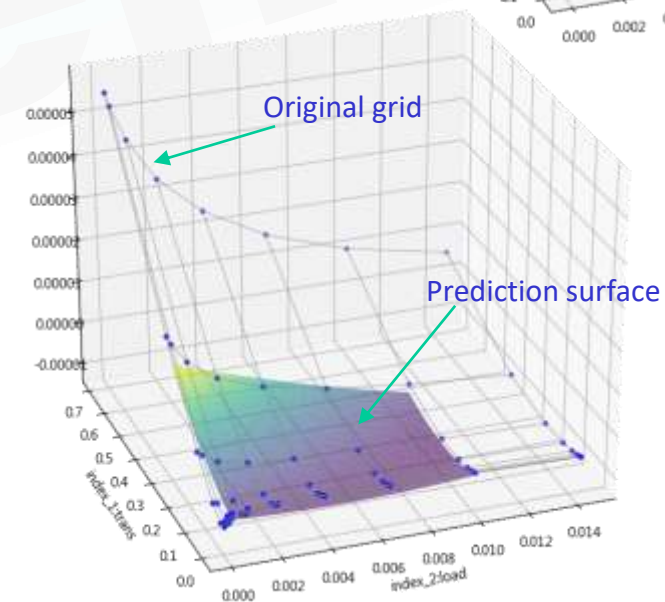
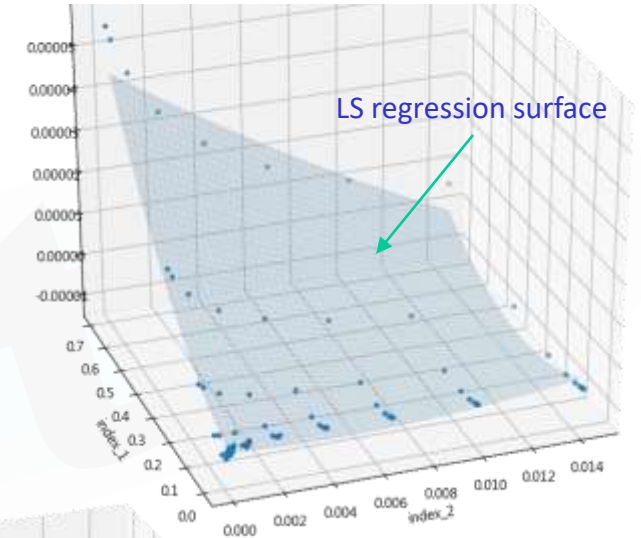
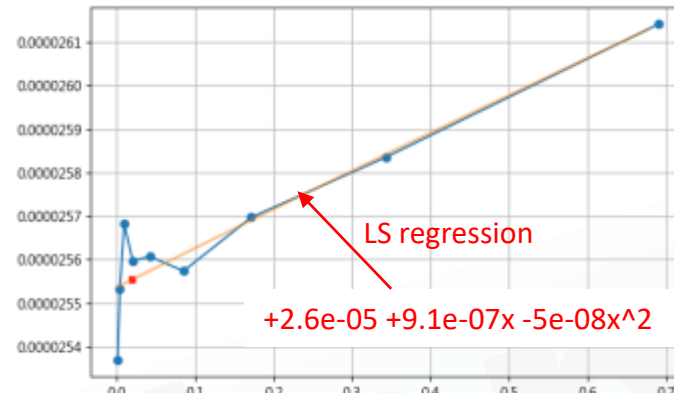
```
# 1-D
lut = lutP['A1,A2', 'fall_power']
C,p = lutl.lut2lsCoeff(lut,order=2,dflag=True)
d = lutl.lut2df(lut)
```

```
# 2-D
lut = lutP['A1,ZN,A2', 'fall_power']
C,p = lutl.lut2lsCoeff(lut,order=2,dflag=True)
d = lutl.lut2df(lut)
```

```
# predict one point
d = lutl.lut2df(lut)
tran,load = 0.0423,0.002207 # trans & load
```

```
# prediction surface
p = np.dot(np.array([1,load,load**2,tran,tran**2,load*tran]).T,C)
```

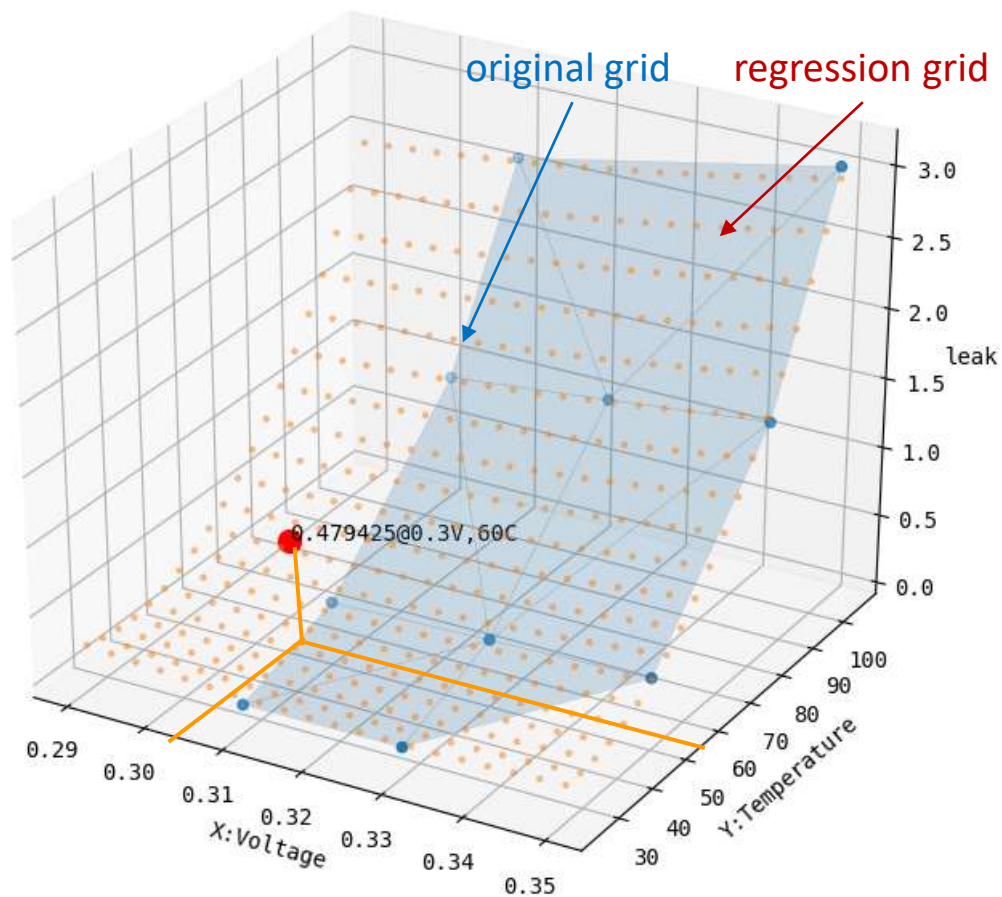
$$-7.1e-06 -0.00036x + 0.041x^2 + 2.8e-05y + 7.6e-05y^2 - 0.0051x*y$$



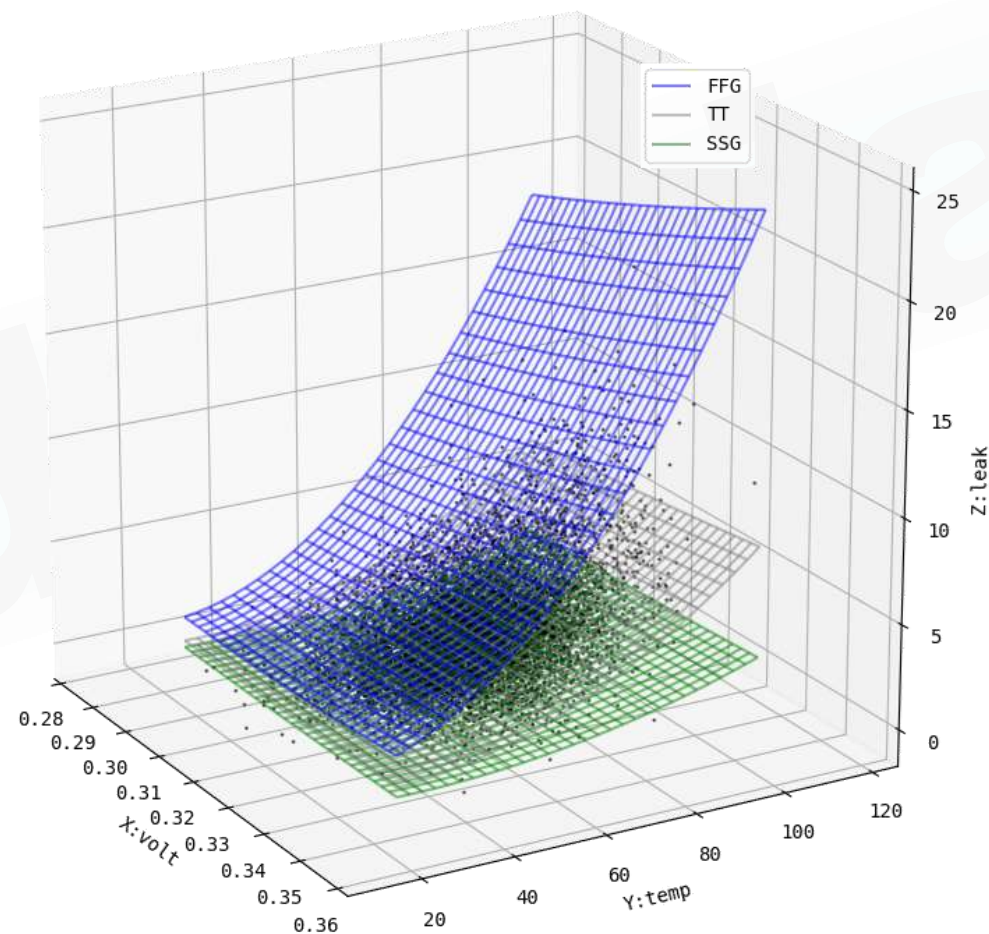


# Precise Leakage Modeling & Evaluation

**Step2:** fit 3 process corners respectively, and formulate process as one of the input parameters, which can be perfectly fitted with 10~16 LS coefficients, as  $f(V,T,P)$



**Step1:** each process corner, ~3 voltage x 3 temperature, ~9 grid points, which can be perfectly fitted with 6 LS coefficients, as  $f(V,T)$



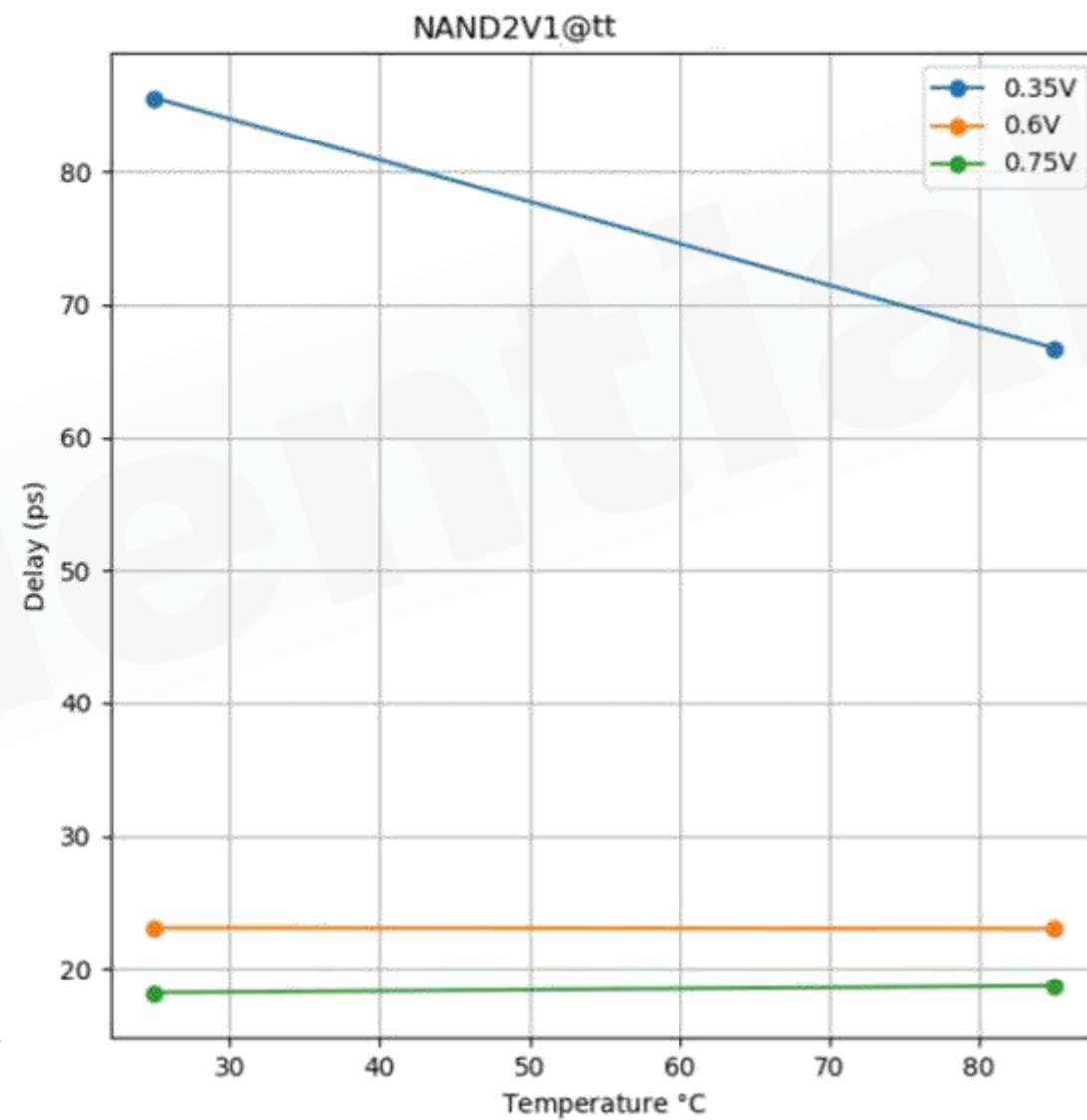
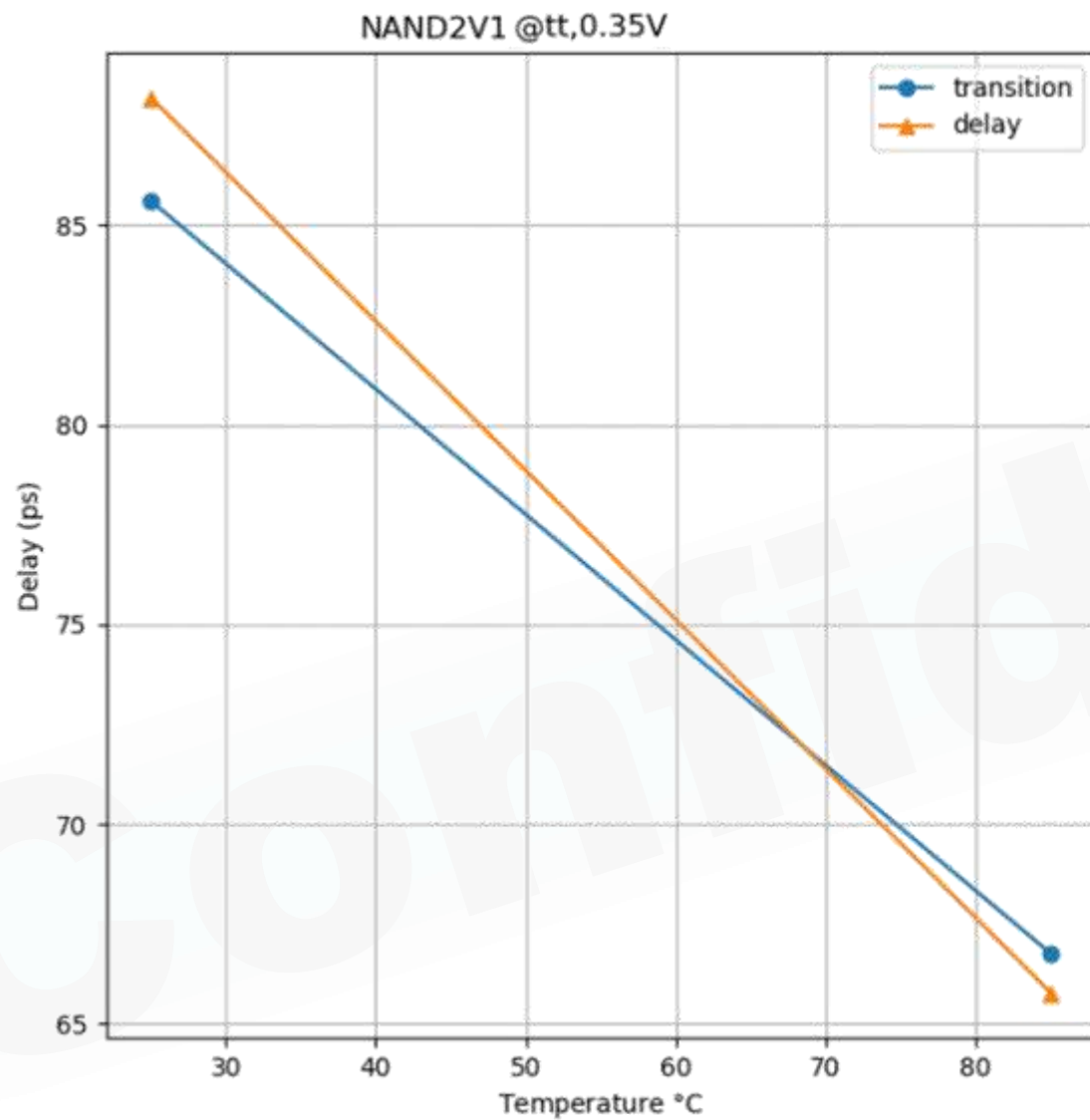
**Step3:** evaluate the leakage distribution following a probability density function



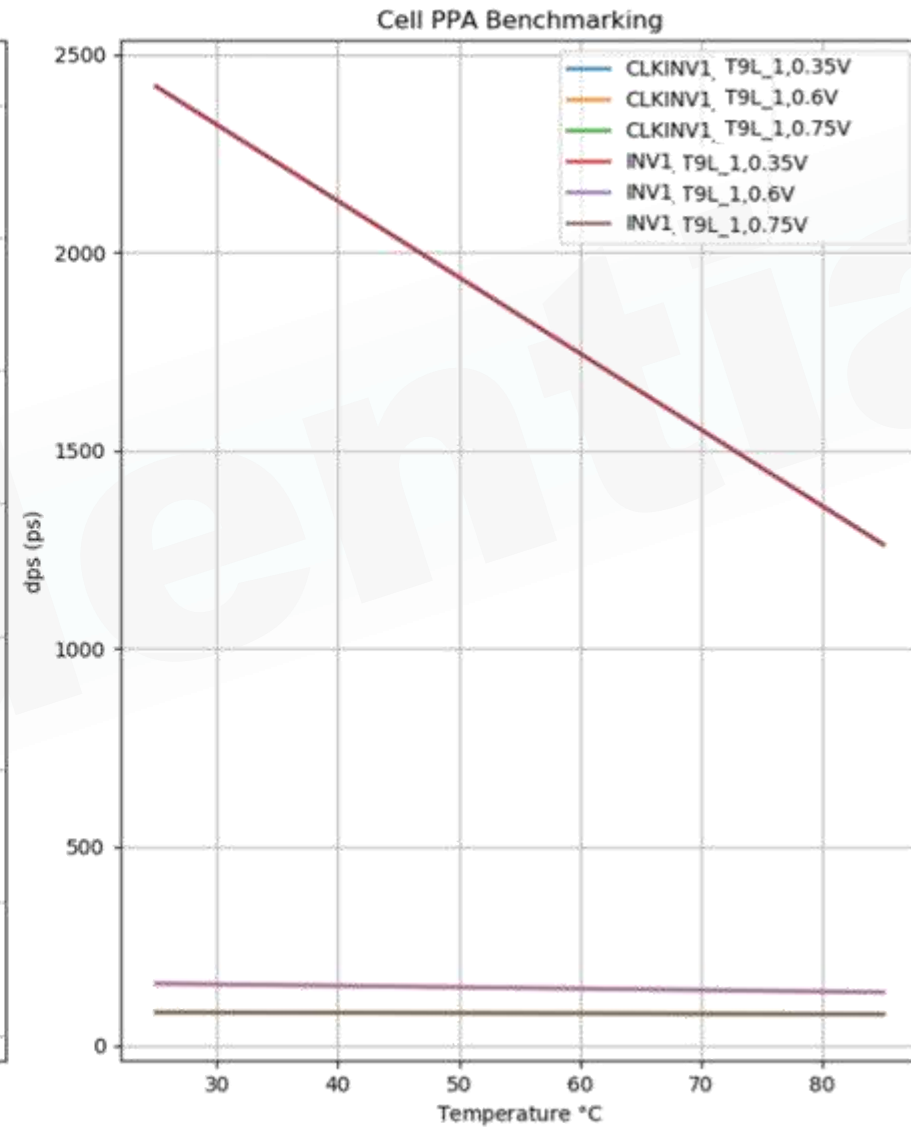
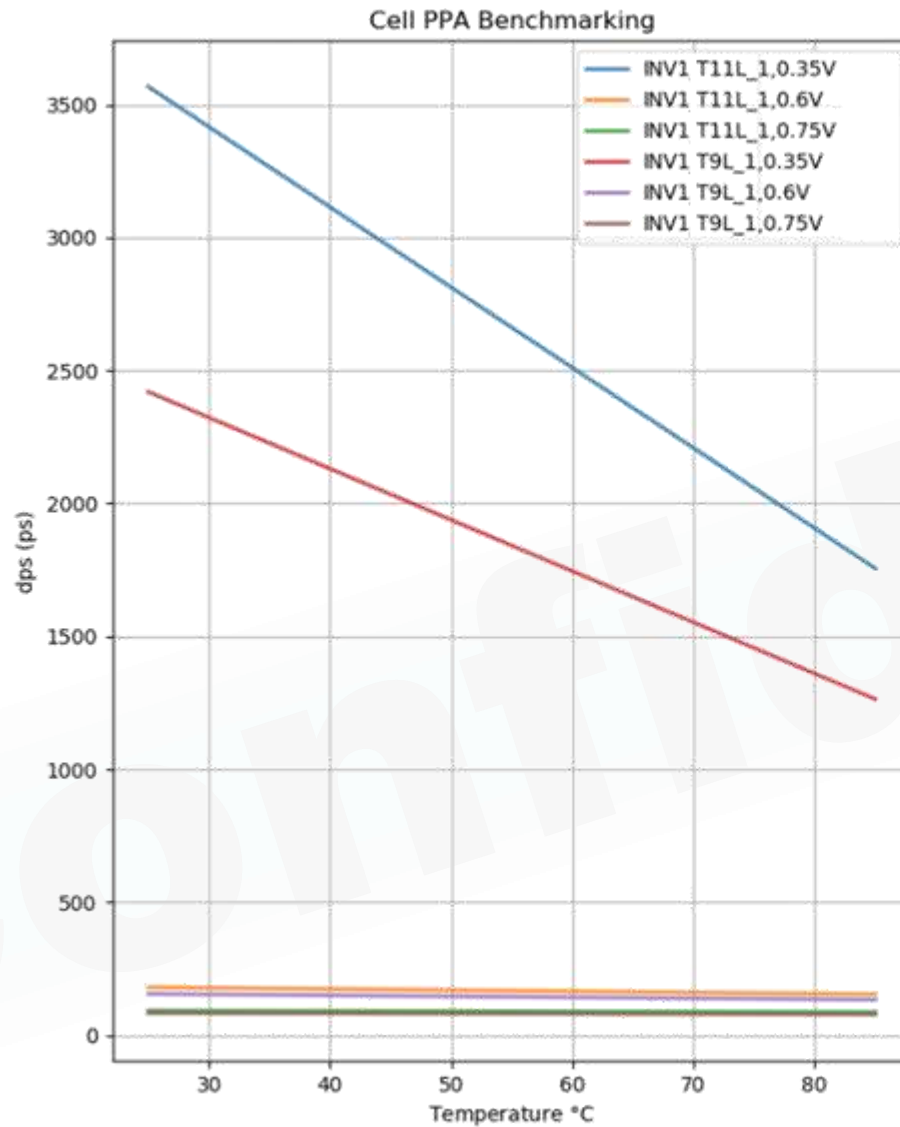


# Application

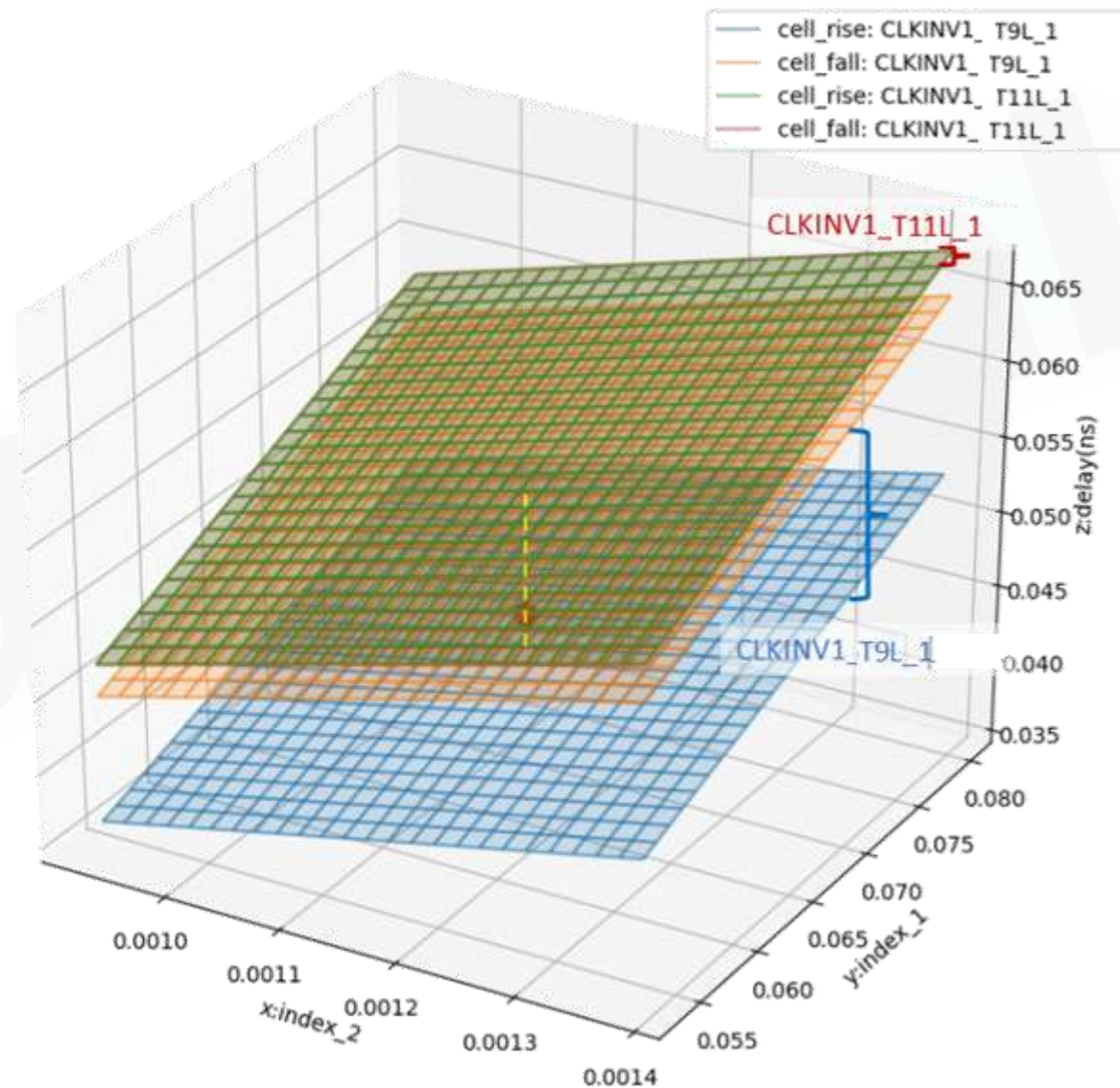
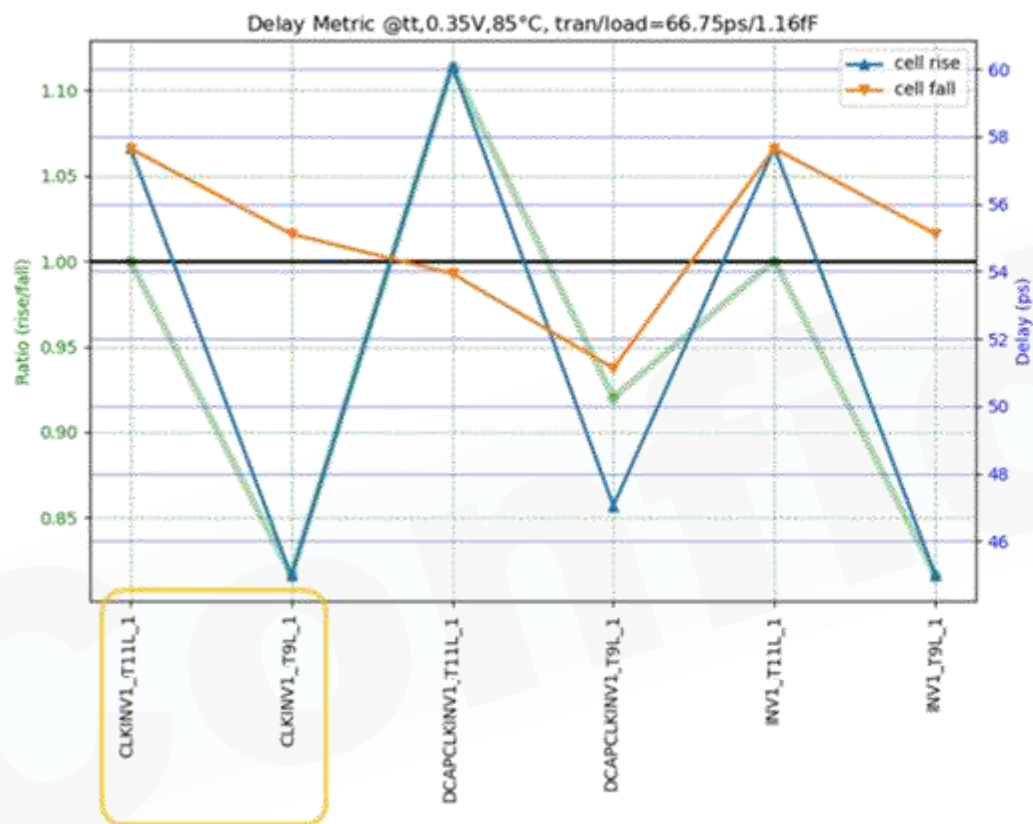
# Metric Visualization



# Temperature Inversion



# Slew Imbalance



# Timing Constraint Batch Comparison

