Design for Productivity -- Grid RO (GRO) Compiler

穩健 Adaption　崛智 Brilliance　創新 Creation　樂群 Diligence

The best solution supplier of energy efficiency

Introduction

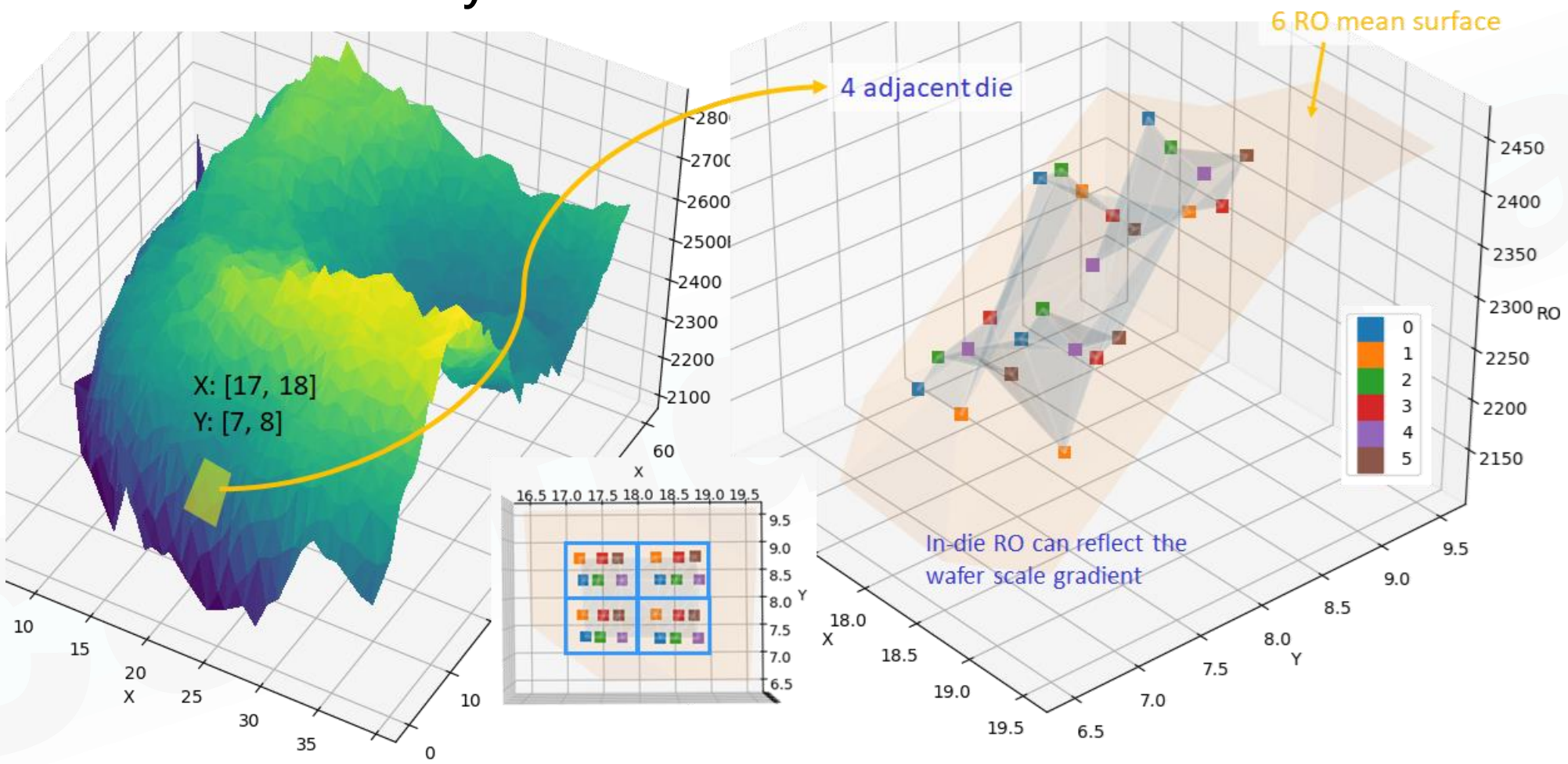
- **Pains of the industry in physical design**
  - Unscientific design margin
  - Insufficient match between signoff method and model

- **Our contribution to the industry**
  - On-chip sensor IP and analysis platform
  - Machine-learning framework for design & production strategy
  - Metric/NN model for EDA industry

Efficiency & Productivity

Uniformity Impact (Design Margin)

- **Feature Gradient Analysis**

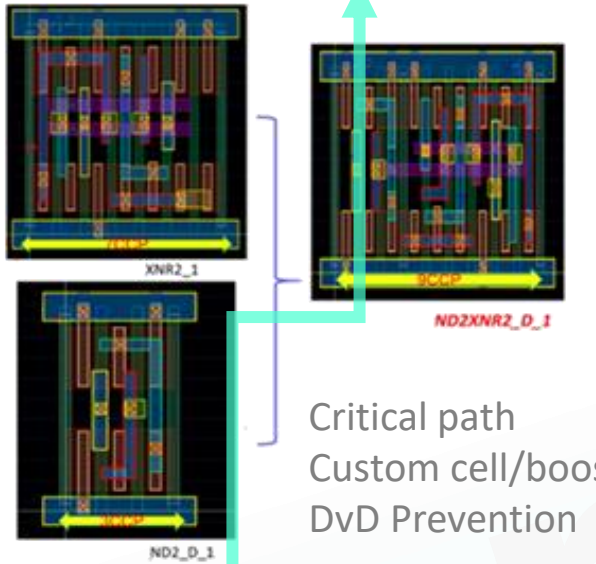


Design for Efficiency

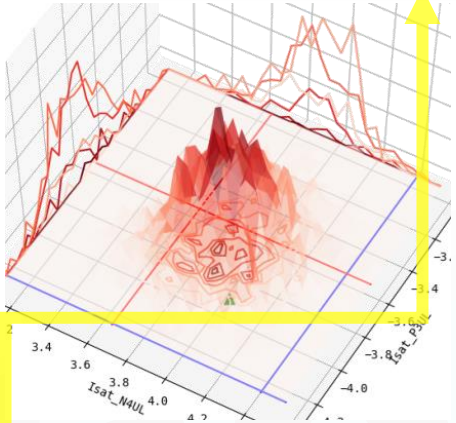
Library/Custom Cell

Architecture/Physical Design

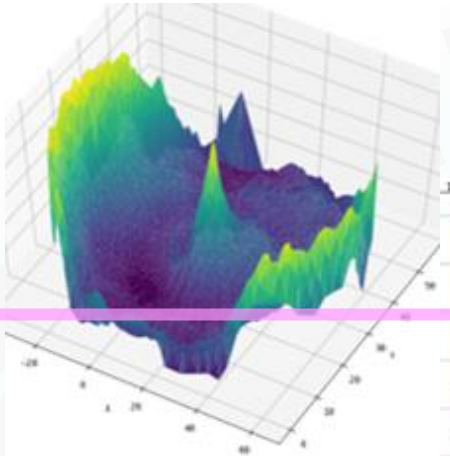
Process Management/Binning



Critical path
Custom cell/boost-RAM
DvD Prevention

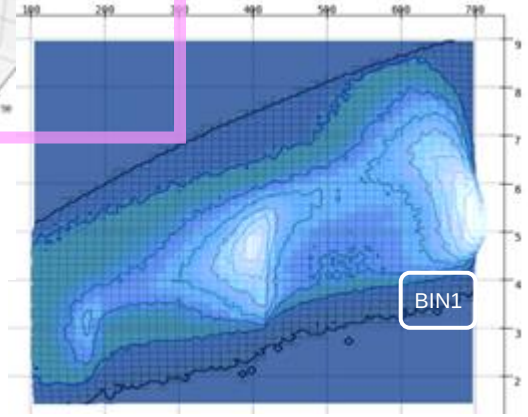


WAT/CP correlation
WAT-aware re-K
Sign-off recipe



Uniformity
OCV regression
Design recipe

Process recipe opt.
Binning strategy



Patent, I769829

Voltage stacking

Ldi/dt | Custom Cell | WAT-aware Design Recipe | Process Management

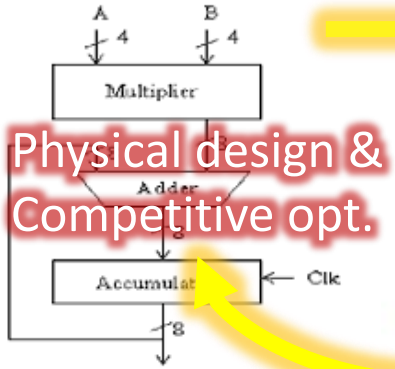
Positive Feedback

Ldi/dt | Library | Physical Design | Process Variation

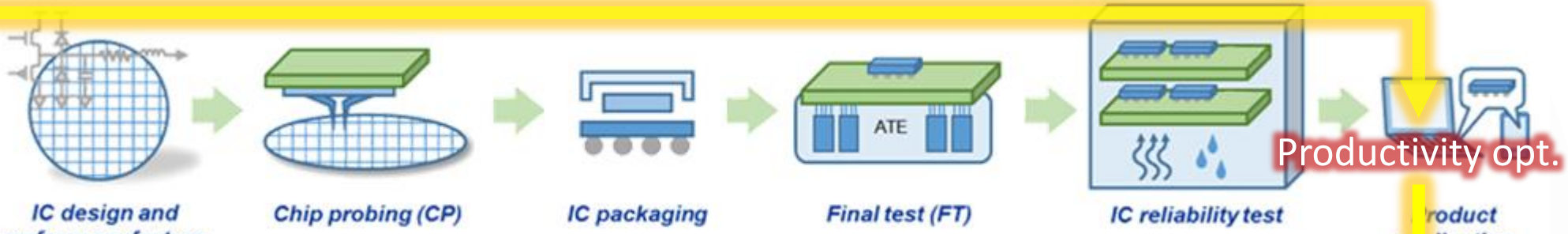
Vmin reduction &
Supporting measures

Energy Efficiency (J/TOPs)

Design for Productivity

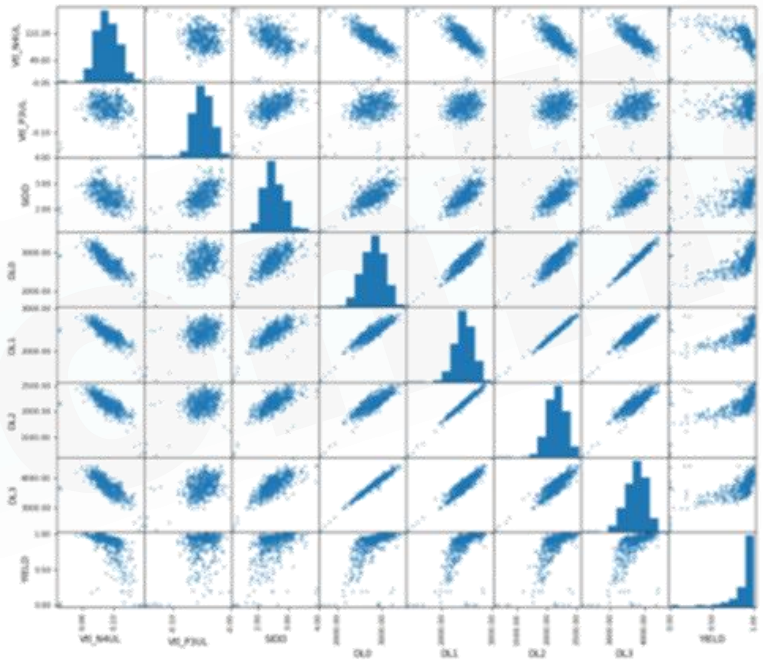


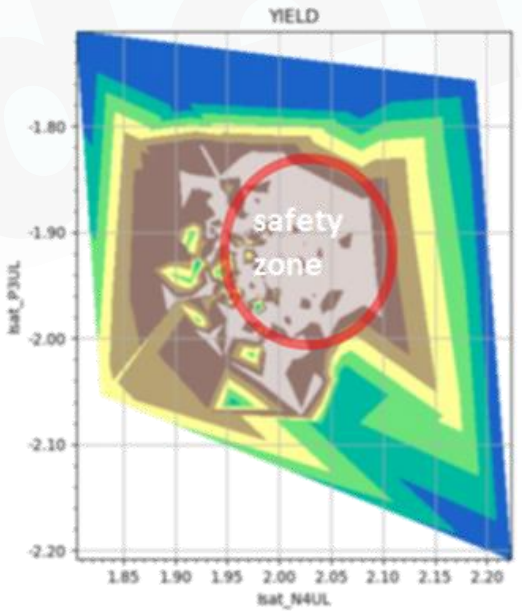
Physical design & Competitive opt.

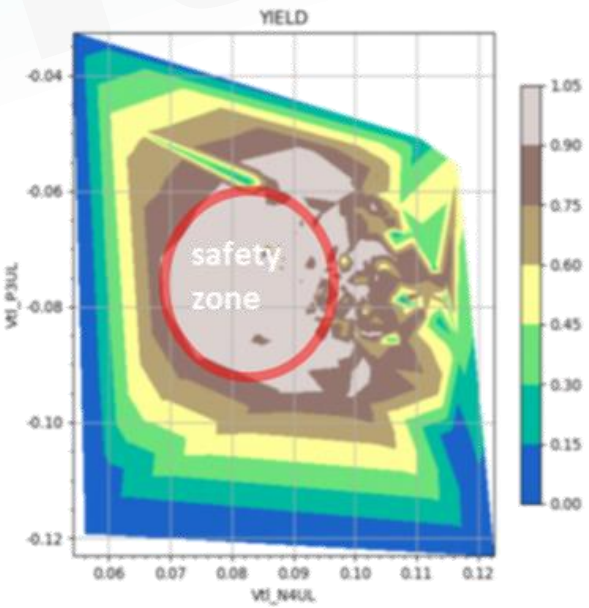


Productivity opt.

WAT ← CP ← Correlation & cost saving ← FT ← Correlation & productivity opt.

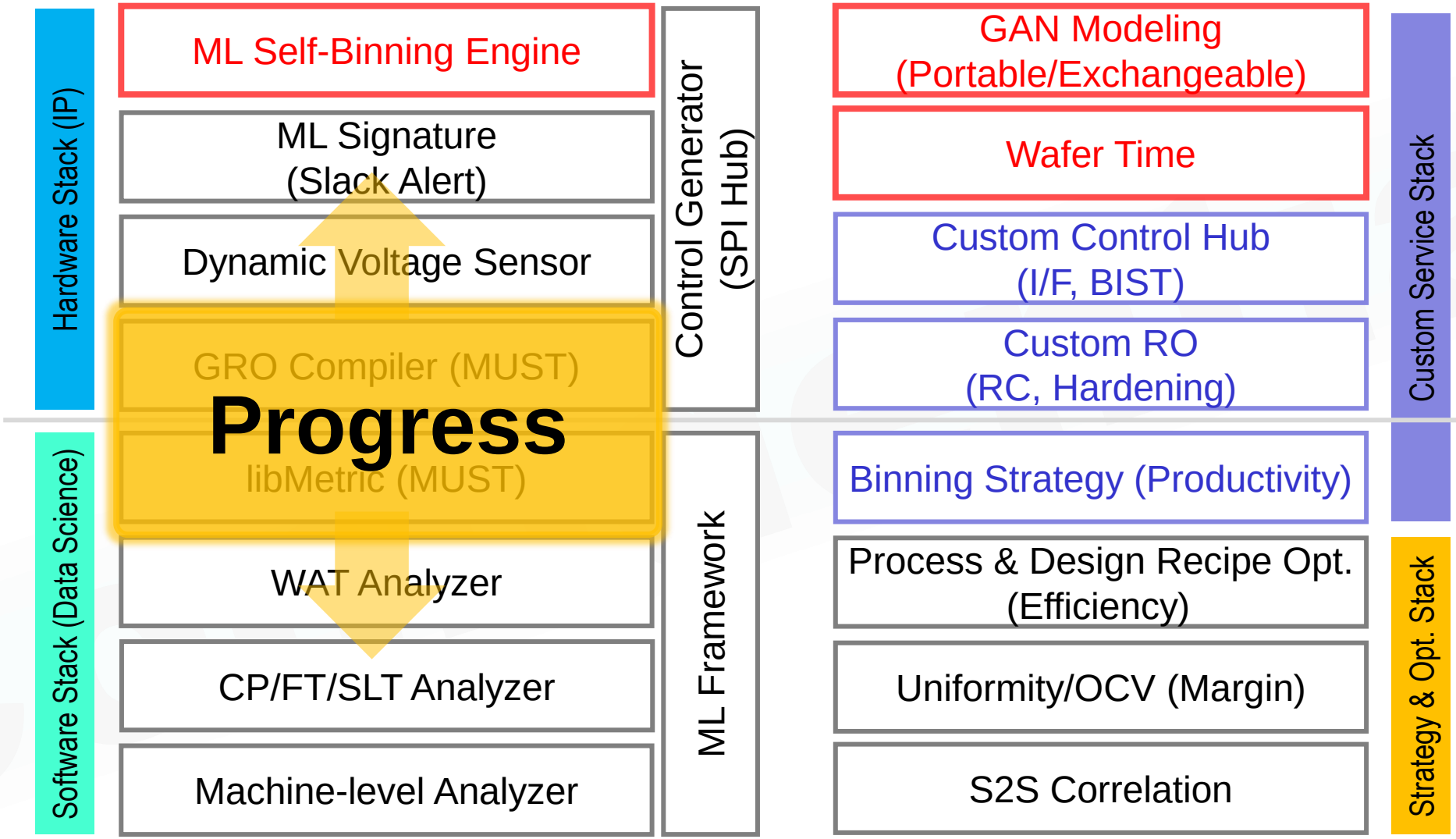






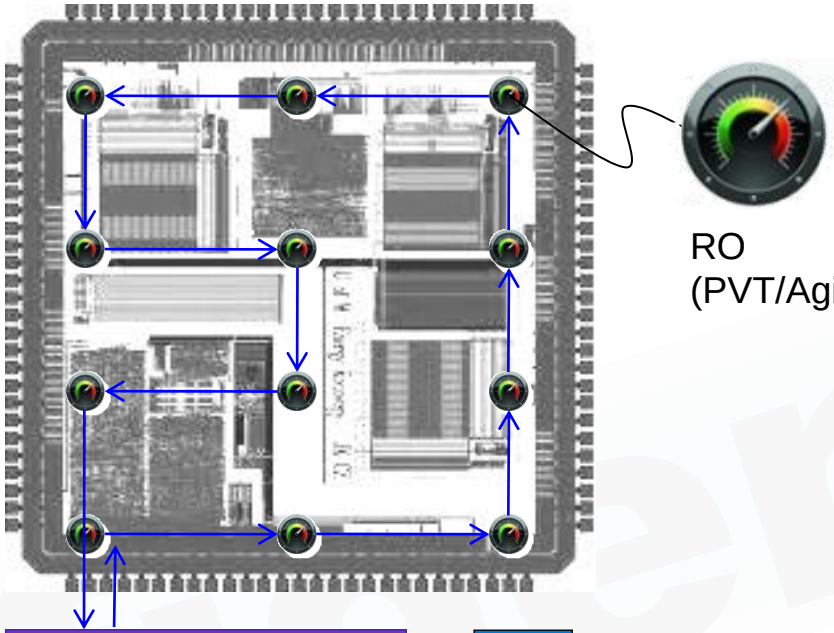
Patent, I700598

HW/SW IP Development Stack



SoC Integration

Grid RO (GRO) Architecture



RO
(PVT/Aging monitor)

ChainOut

ROSel

ROEn

GRO Control

OTP

OTP, eFuse

Self Binning

Machine-learning, Speed prediction

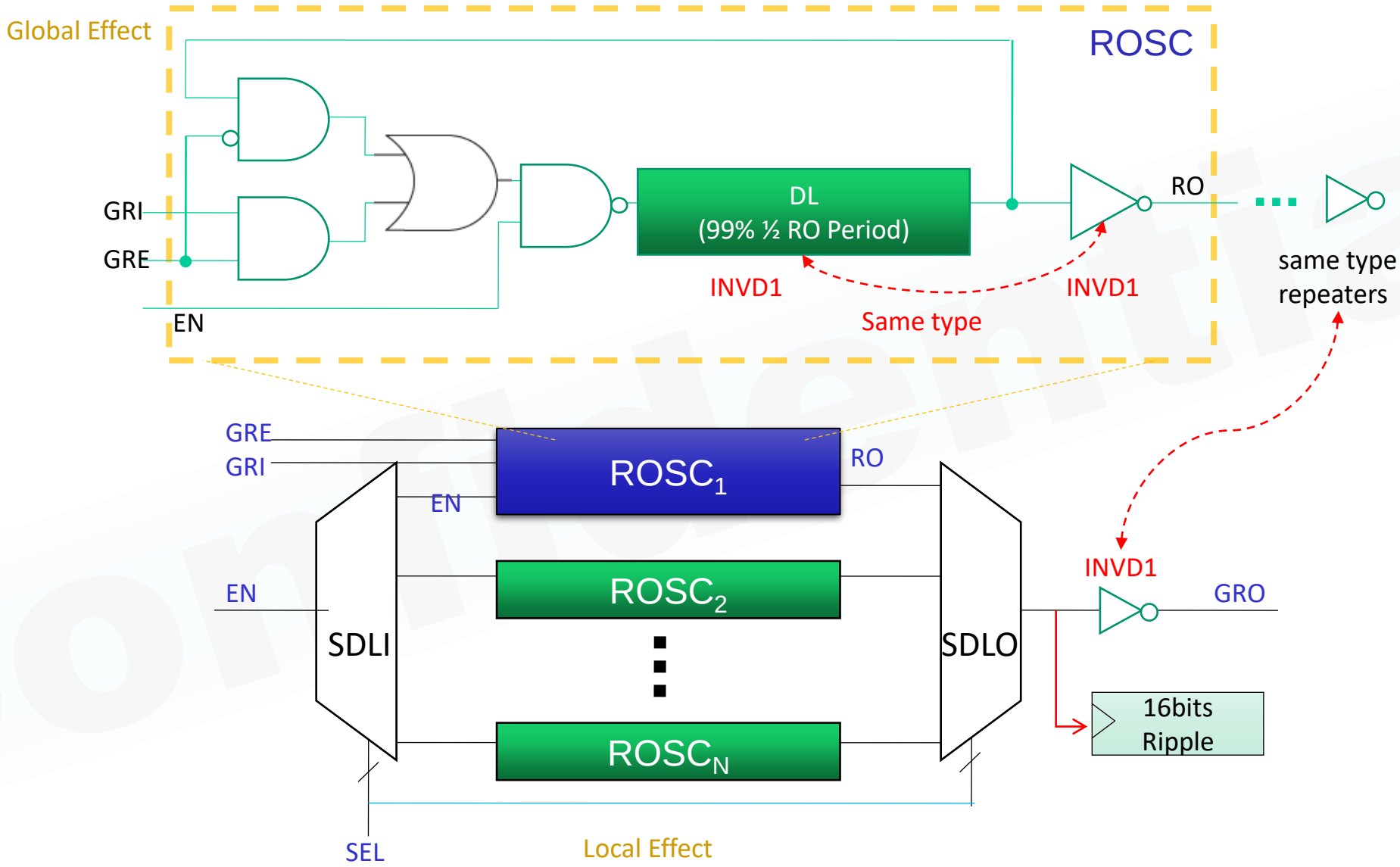
Speed/Voltage Calibration

PVT/Vt/Aging compensation

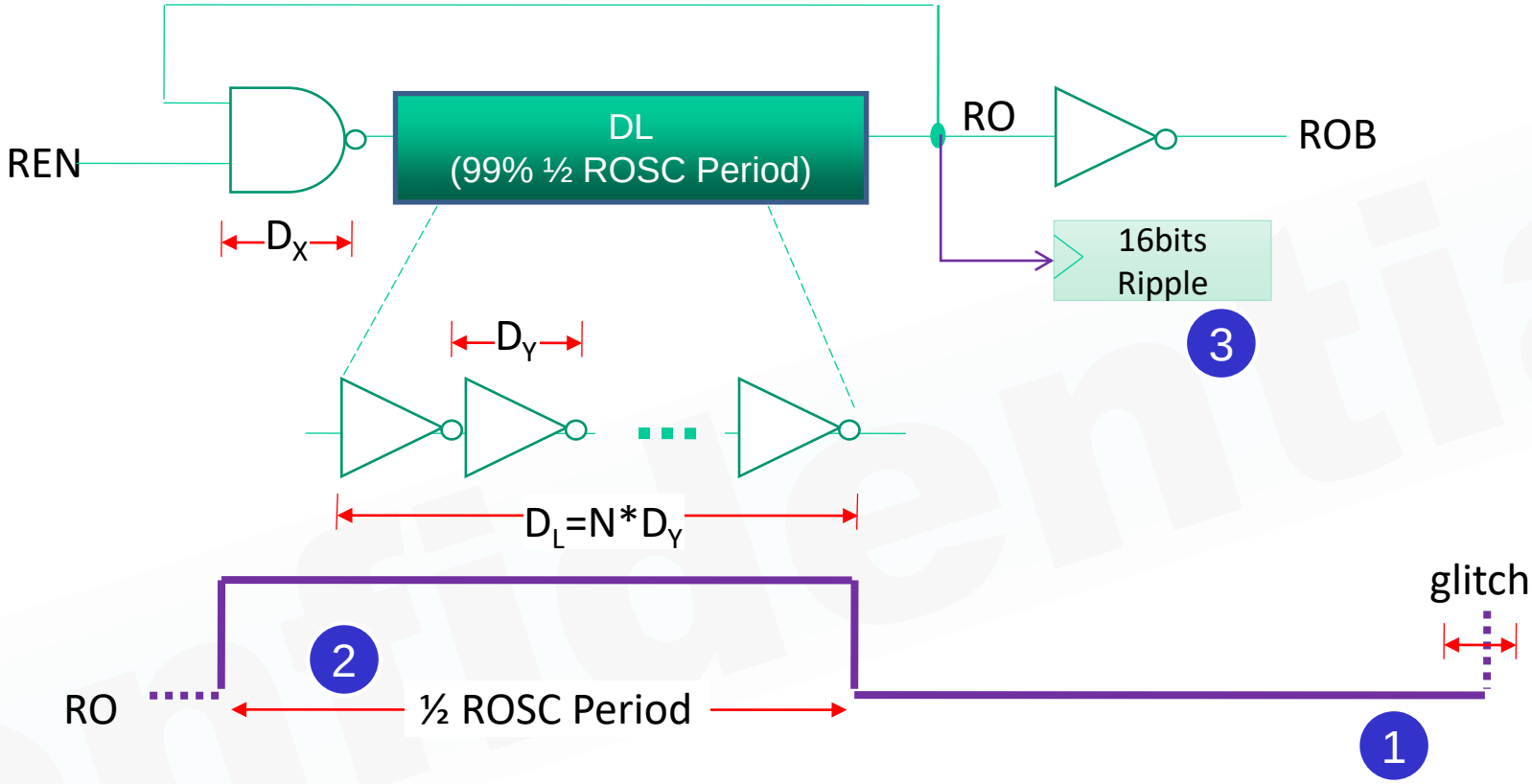
SVC protocol

PMIC/PLL

RO Considering Gross-effect

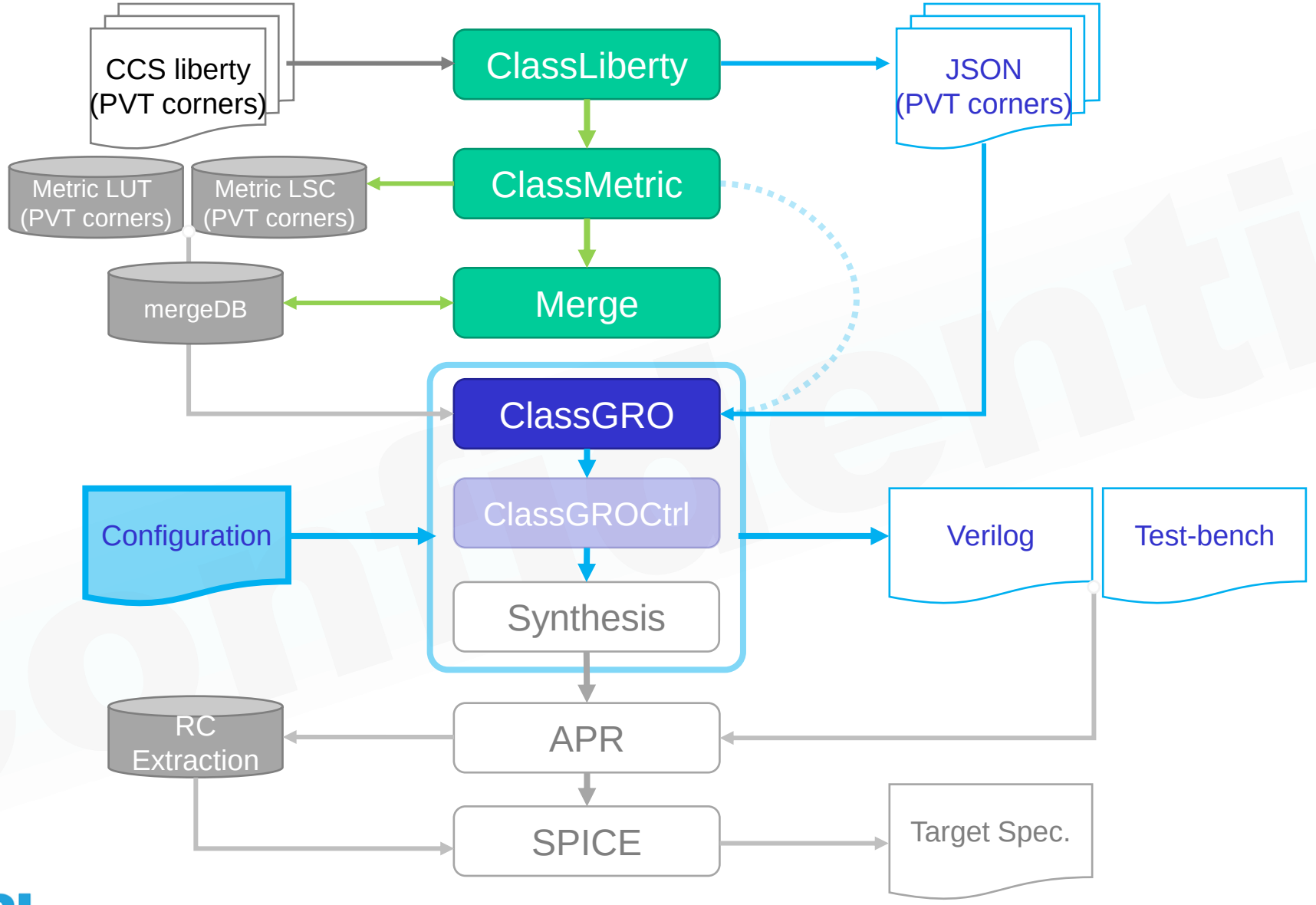


RO Design Guideline

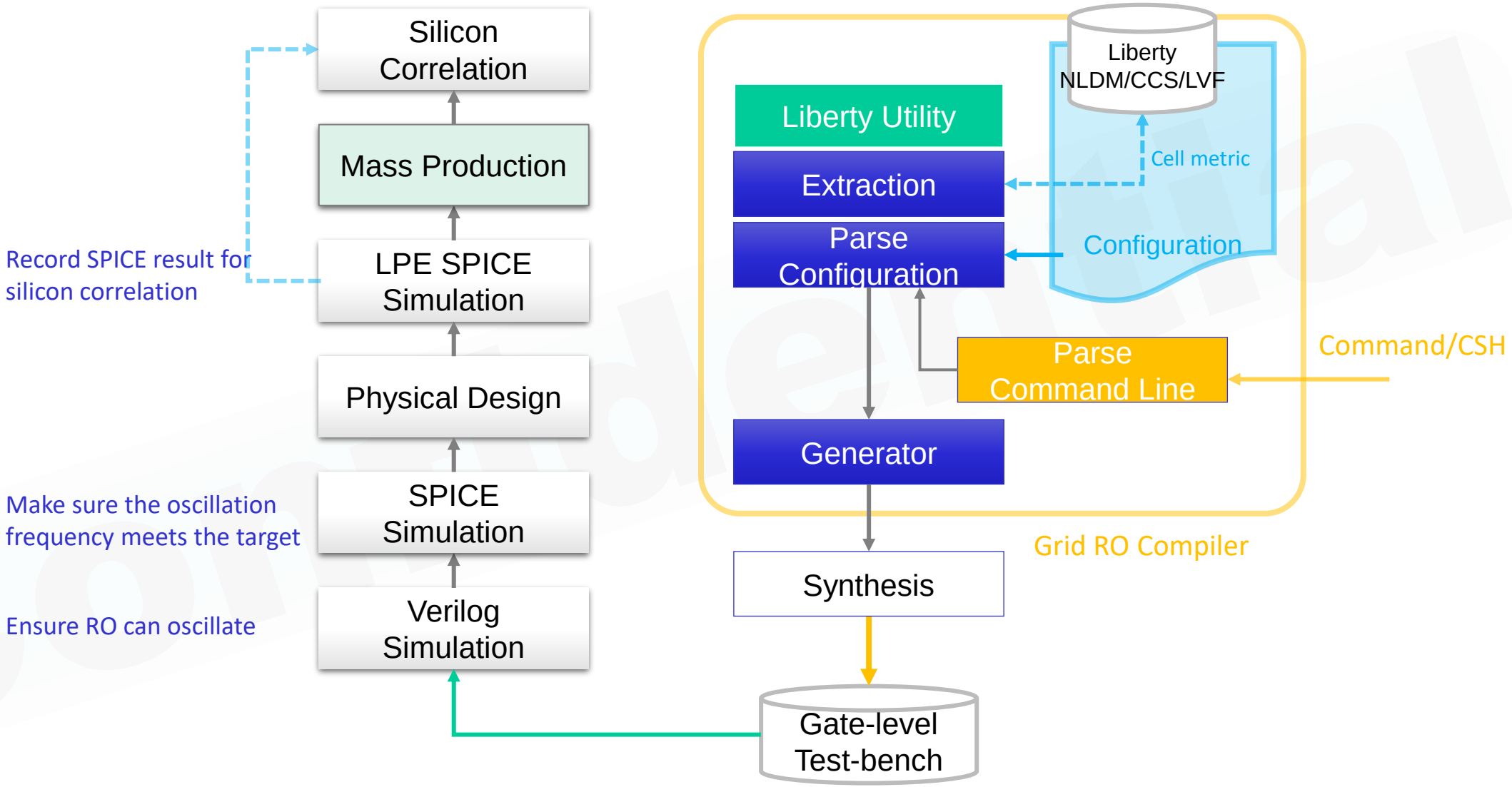


1. Minimum REN retain time > 200 Count to compensate for 1% counting error
2. Minimum DL stage N > $\lceil 100*D_X/D_Y \rceil$
3. Maximum REN retain time < 16,000 RO count for 14bits ripple counter

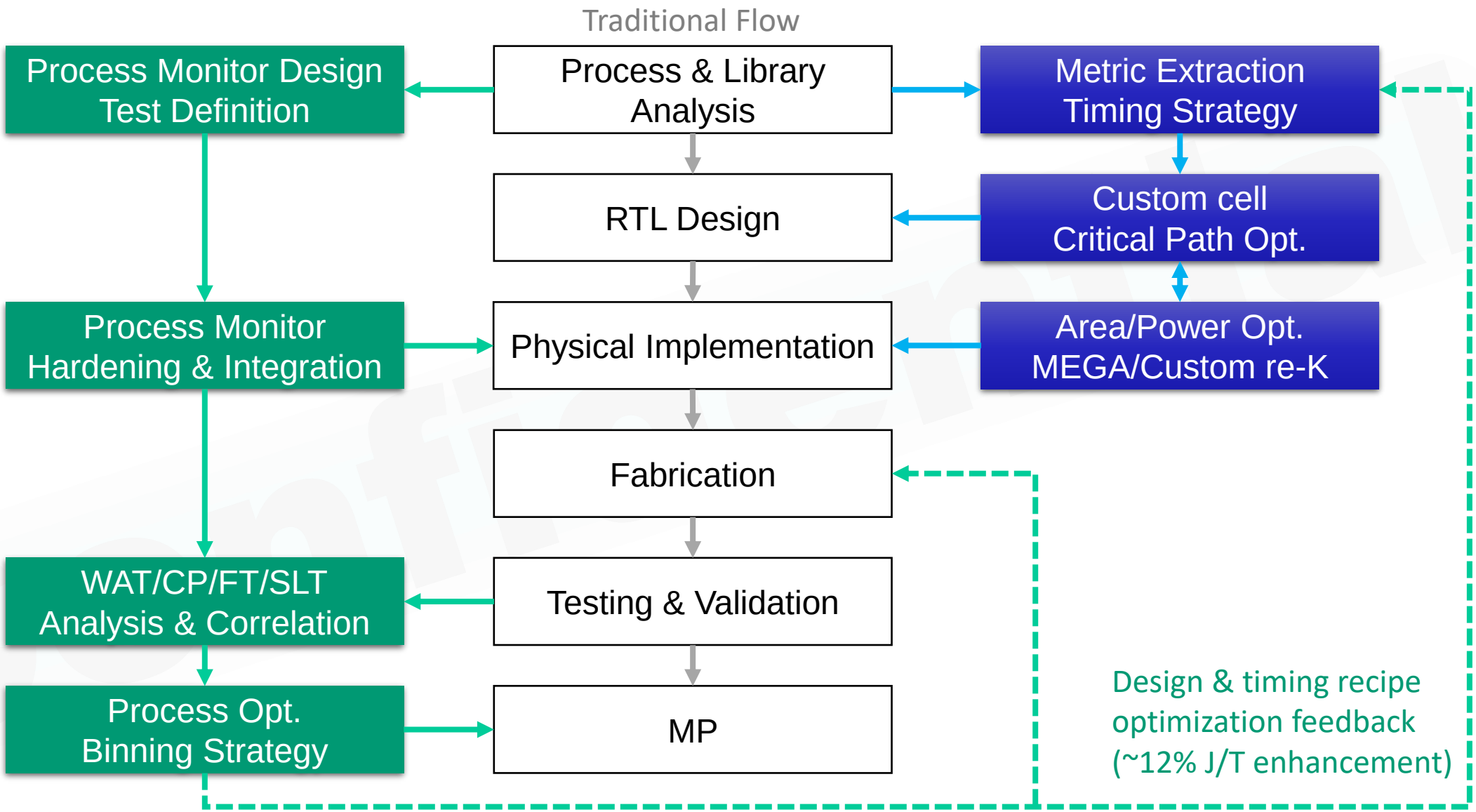
GRO Program Execution Flow



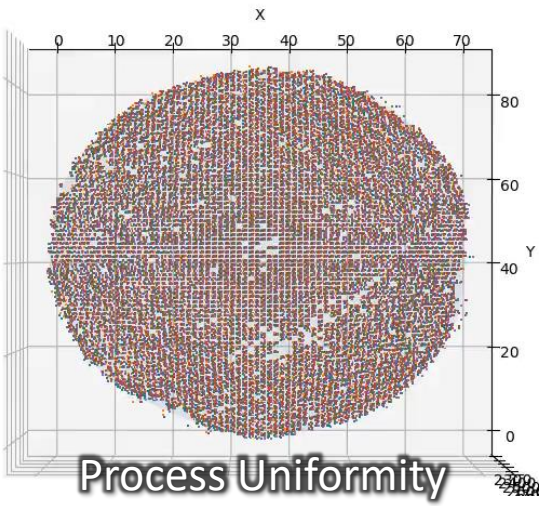
Physical Design & Correlation Flow

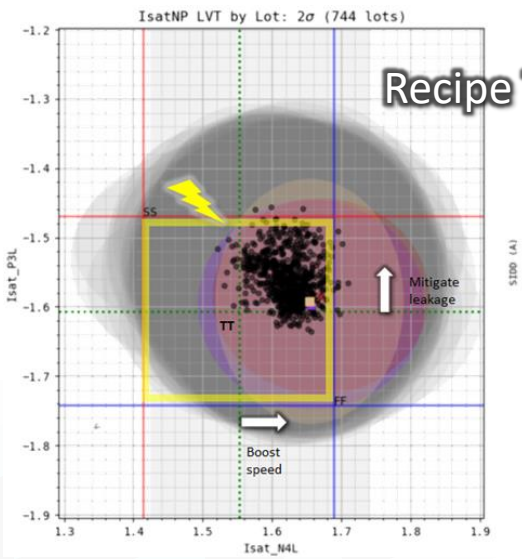


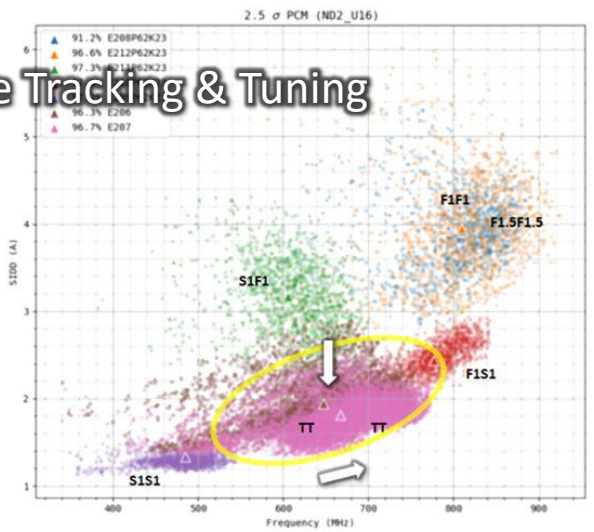
Design & Technology Co-optimization Flow

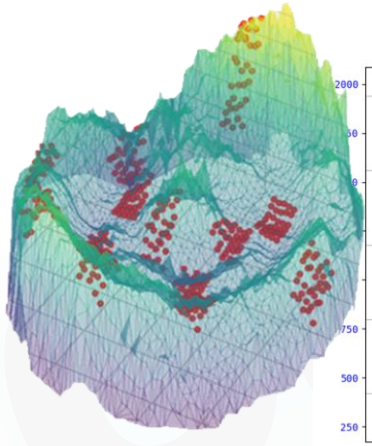


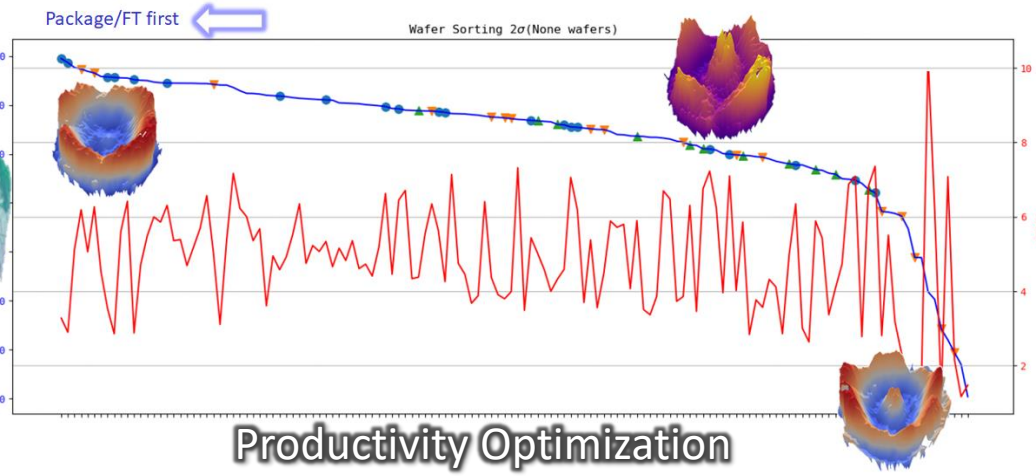
Machine-learning Framework



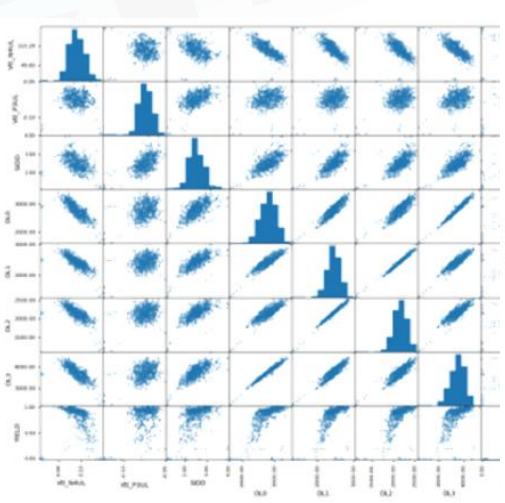


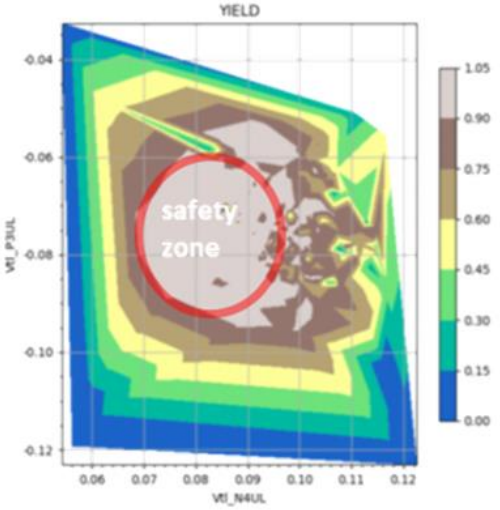






Productivity Optimization
Wafer & Die Sorting





Design & Process Recipe Optimization

DipSci

Blue Ocean

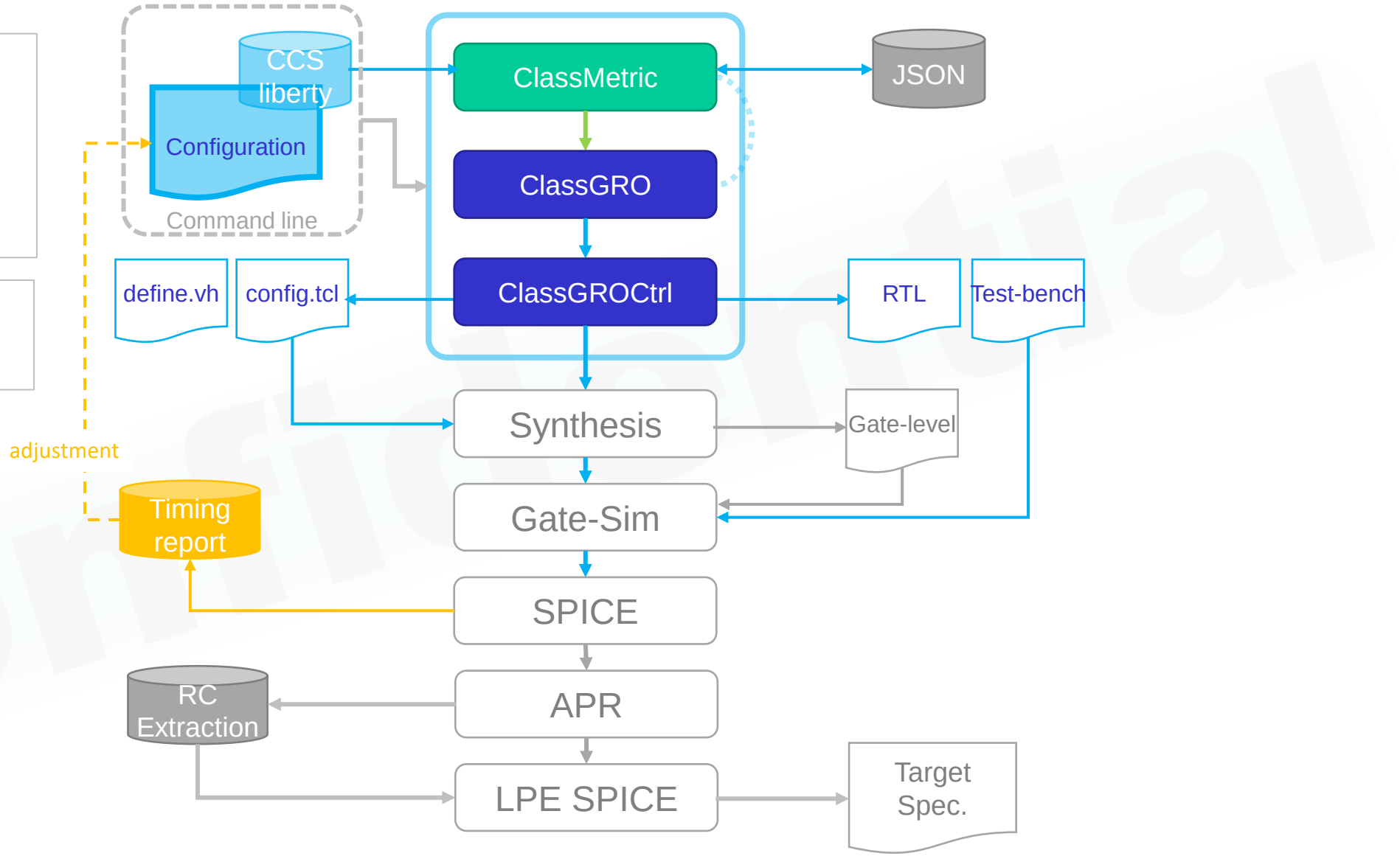
- **Efficiency & Productivity Optimization**
- **Sensor IP & Methodology Development**
- **Integration & Production Flow**
- **Data Science**
  - Sensing Data → IP/Platform Development
  - Feature Correlation → EDA Development
  - Recipe Optimization → Strategy Service

GRO Compiler
Detail Specification

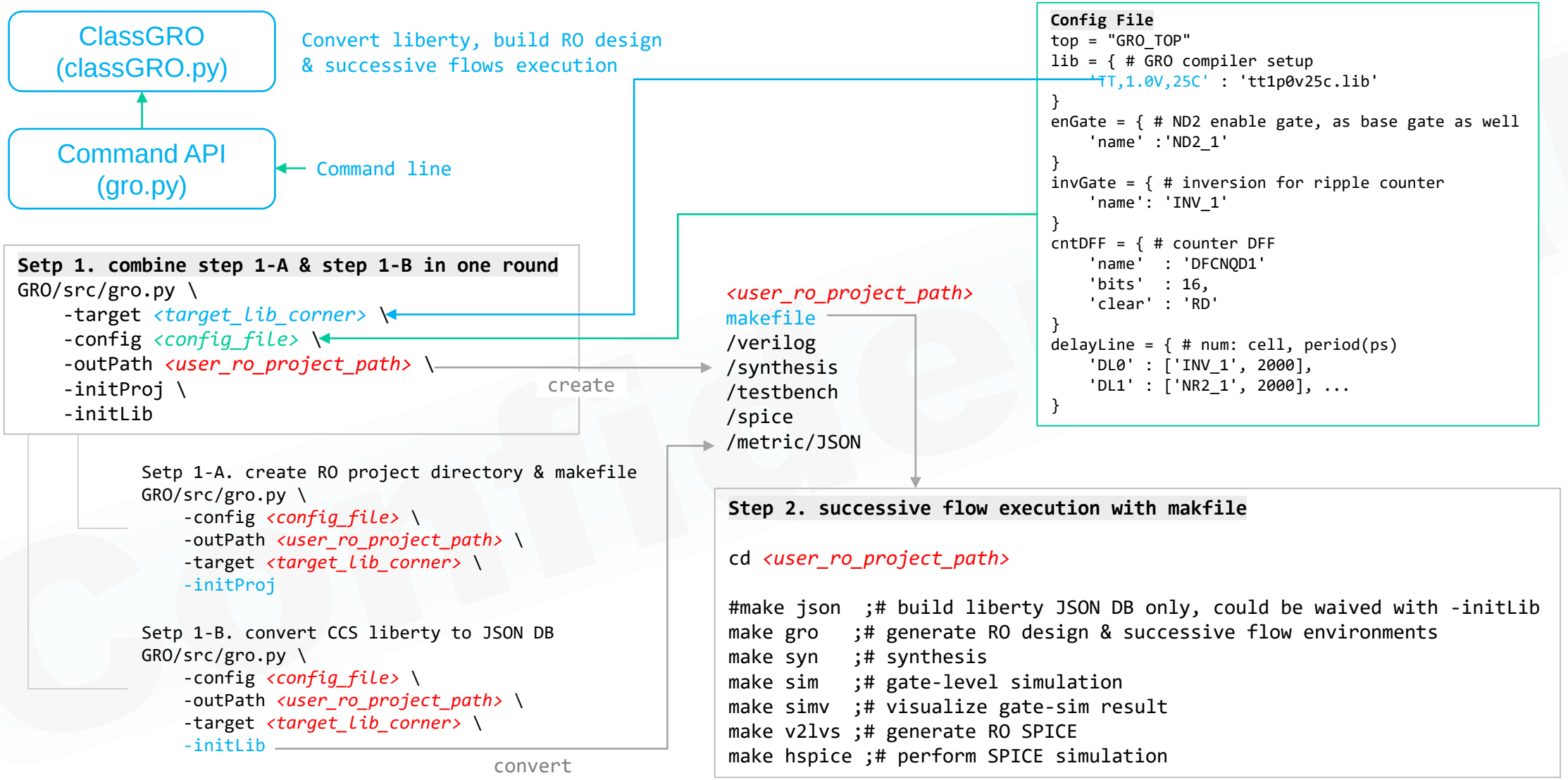
GRO Verification Flow

```
# config.tcl
namespace eval groCfg {
    array set DC {
        libName      “library_ccs"
        topModule    "GRO_TOP"
        opConditions "tt0p8v25c"
        clockPeriod  1.000
        vlog         [library.v']
    }
}
```

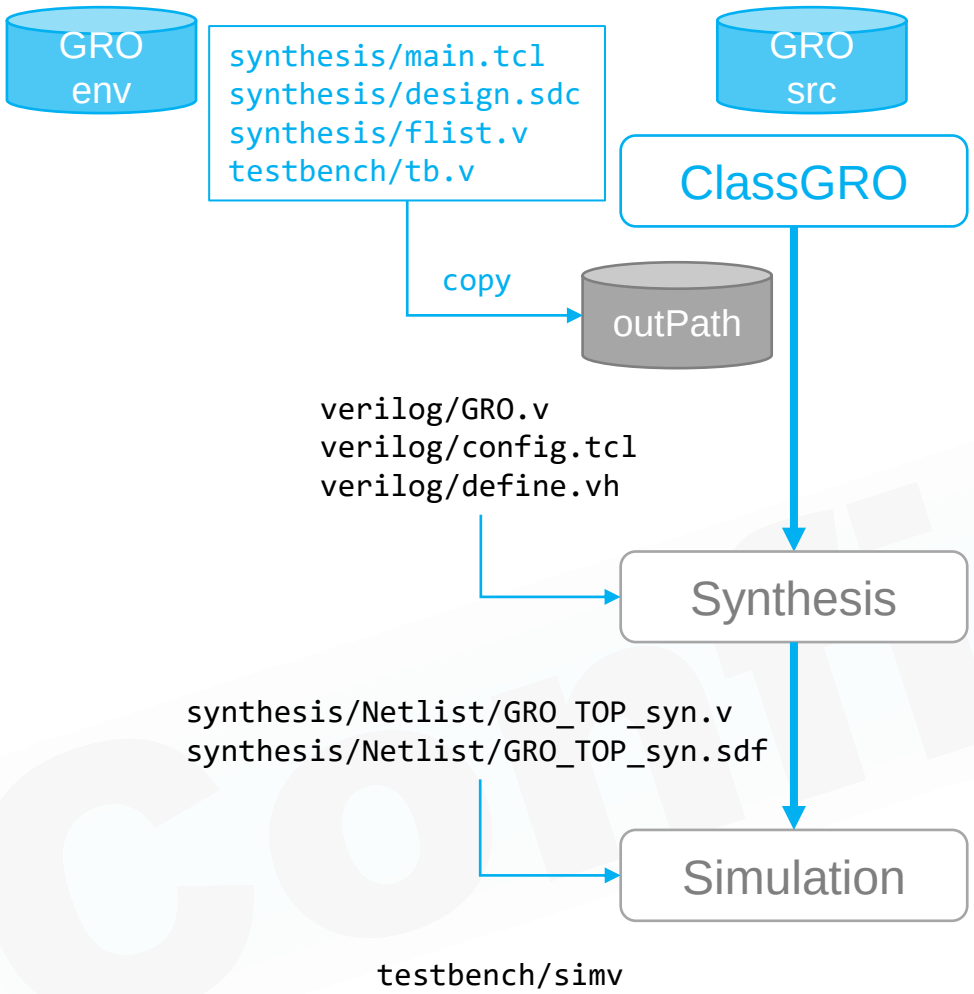
```
// define.vh
`define DL_NUM 6
`define SEL_BITS 3
`define RPC_BITS 16
```



GRO Wrapper

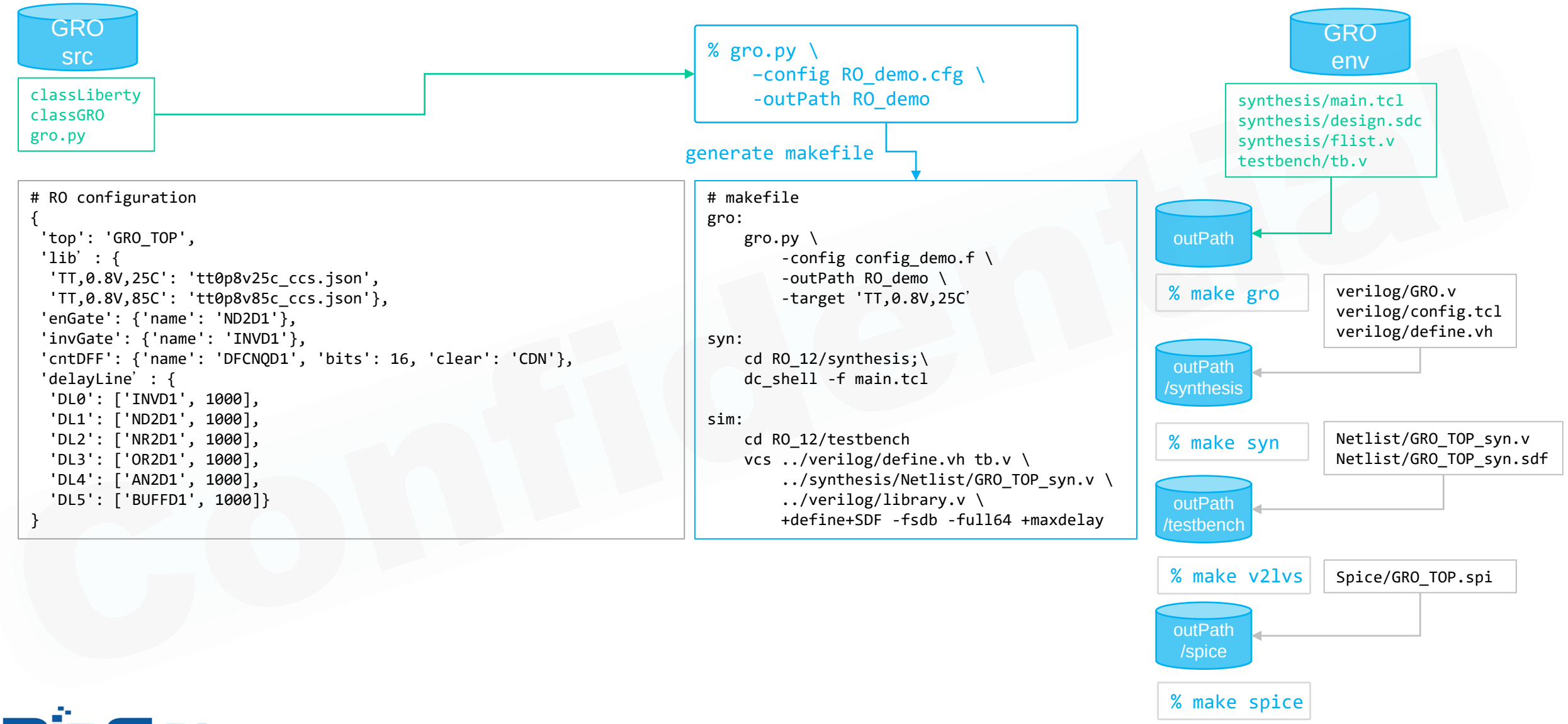


Command & I/O Specification




```
#!/bin/csh

# generate RO design
<GRO_root>/src/gro.py \
    -config config_demo.f \
    -outPath RO_demo \
    -target 'TT'

# synthesis
cd RO_demo/synthesis
dc_shell -f main.tcl # source verilog/config.tcl

# simulation
cd RO_demo/testbench
vcs ../verilog/define.vh tb.v ../synthesis/Netlist/GRO_TOP_syn.v \
    ../verilog/tcbn12ffcllbwp16p90cpd.v \
    +define+SDF -fsdb -full64 +maxdelay

./simv
```

Command Execution Flow




```
# RO configuration
{
 'top': 'GRO_TOP',
 'lib' : {
  'TT,0.8V,25C': 'tt0p8v25c_ccs.json',
  'TT,0.8V,85C': 'tt0p8v85c_ccs.json'},
 'enGate': {'name': 'ND2D1'},
 'invGate': {'name': 'INVD1'},
 'cntDFF': {'name': 'DFCNQD1', 'bits': 16, 'clear': 'CDN'},
 'delayLine' : {
  'DL0': ['INVD1', 1000],
  'DL1': ['ND2D1', 1000],
  'DL2': ['NR2D1', 1000],
  'DL3': ['OR2D1', 1000],
  'DL4': ['AN2D1', 1000],
  'DL5': ['BUFFD1', 1000]}
}
```

```
# makefile
gro:
    gro.py \
        -config config_demo.f \
        -outPath RO_demo \
        -target 'TT,0.8V,25C'

syn:
    cd RO_12/synthesis;\
    dc_shell -f main.tcl

sim:
    cd RO_12/testbench
    vcs ../verilog/define.vh tb.v \
        ../synthesis/Netlist/GRO_TOP_syn.v \
        ../verilog/library.v \
        +define+SDF -fsdb -full64 +maxdelay
```

Configuration Automation

cfg1

```
{'top': 'GRO_TOP',
 'lib' : {
  'TT,0.8V,25C': 'tt0p8v25c_ccs.json',
  'TT,0.8V,85C': 'tt0p8v85c_ccs.json'},
 'enGate': {'name': 'ND2D1'},
 'invGate': {'name': 'INVD1'},
 'cntDFF': {'name': 'DFCNQD1', 'bits': 16, 'clear': 'CDN'},
 'delayLine' : {
  'DL0': ['INVD1', 1000],
  'DL1': ['ND2D1BWP16P90CPD', 1000],
  'DL2': ['NR2D1BWP16P90CPD', 1000],
  'DL3': ['OR2D1BWP16P90CPD', 1000],
  'DL4': ['AN2D1BWP16P90CPD', 1000],
  'DL5': ['BUFFD1BWP16P90CPD', 1000]}}
```

Detail timing and in/out specification will be generated automatically

cfg2

```
{'top': 'GRO_TOP',
 'lib': {
  'TT,0.8V,25C': 'tt0p8v25c_ccs.json',
  'TT,0.8V,85C': 'tt0p8v85c_ccs.json'},
 'enGate': {'name': 'ND2D1',
  'tran': 0.009822304703498236,
  'load': 0.0012335879999999999,
  'delay': 0.011114778622743677,
  'en': 'A2',
  'in': 'A1',
  'out': 'ZN'},
 'invGate': {'name': 'INVD1', 'in': 'I', 'out': 'ZN'},
 'cntDFF': {'name': 'DFCNQD1',
  'bits': 16,
  'clear': 'CDN',
  'clock': 'CP',
  'in': 'D',
  'out': 'Q'},
 'delayLine': {
  'DL0': ['INVD1', 1000, ['I'], ['ZN']],
  'DL1': ['ND2D1', 1000, ['A1', 'A2'], ['ZN']],
  'DL2': ['NR2D1', 1000, ['A1', 'A2'], ['ZN']],
  'DL3': ['OR2D1', 1000, ['A1', 'A2'], ['Z']],
  'DL4': ['AN2D1', 1000, ['A1', 'A2'], ['Z']],
  'DL5': ['BUFFD1', 1000, ['I'], ['Z']]}}
```

```
gro = ROCompiler()
lnode = gro.loadLib('JSON/tt0p8v25c_ccs.json')
gro.baseGateInfo(lnode,cell='ND2D1')

cfg1 = gro.loadConfig('config.f')
cfg2 = gro.integrityCheck(lnode) # GRO component integrity based on the configuration
```

DipSci

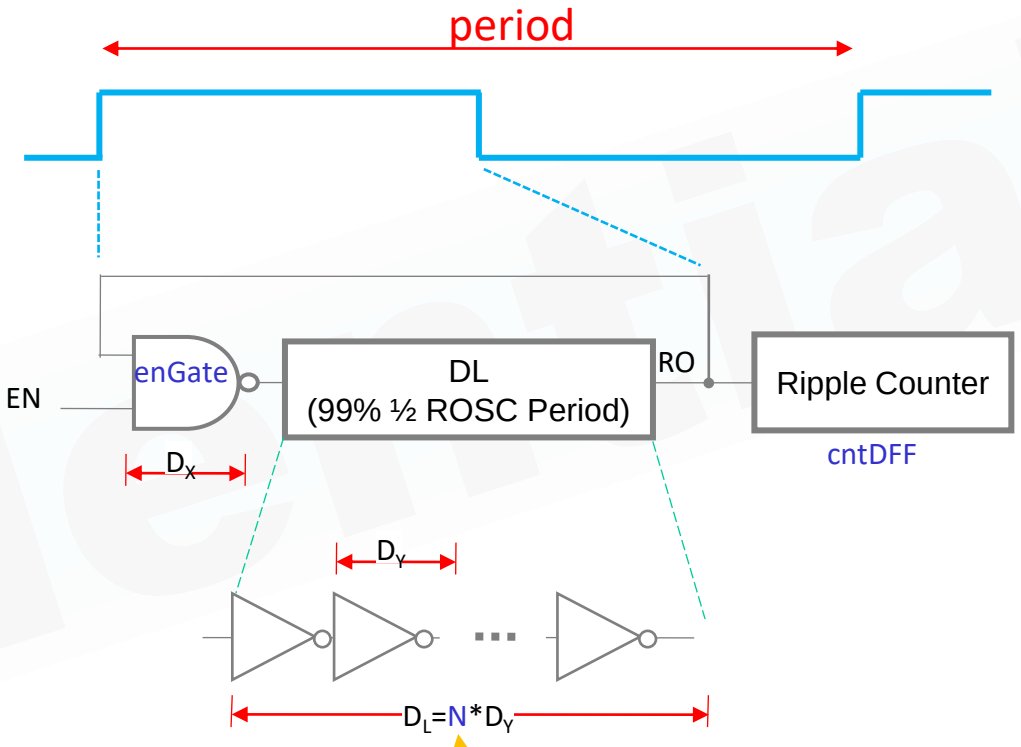
Configuration

```
#%% RO components
enGate = { # ND2 enable gate
    'name':'NAND2V1',
    'en'  : 'A2',
    'in'  : 'A1',
    'out' : 'ZN'
}
invGate = { # inversion for ripple counter
    'name': 'INV1',
    'in'  : 'I',
    'out' : 'ZN'
}
cntDFF = { # counter DFF
    'name'  :'DQV1E',
    'clock' : 'CK',
    'in'    : 'D',
    'out'   : 'Q',
    'bits'  : 4
}
delayLine = {
        # num: cell, period(ps), inPins, outPins
        'DL0': ['INVD1', 1000, [],[]],
        'DL1': ['ND2D1', 1000, [],[]],
        'DL2': ['NR2D1', 1000, [],[]],
    }
```

Ripple Counter Bits

Period (ps)

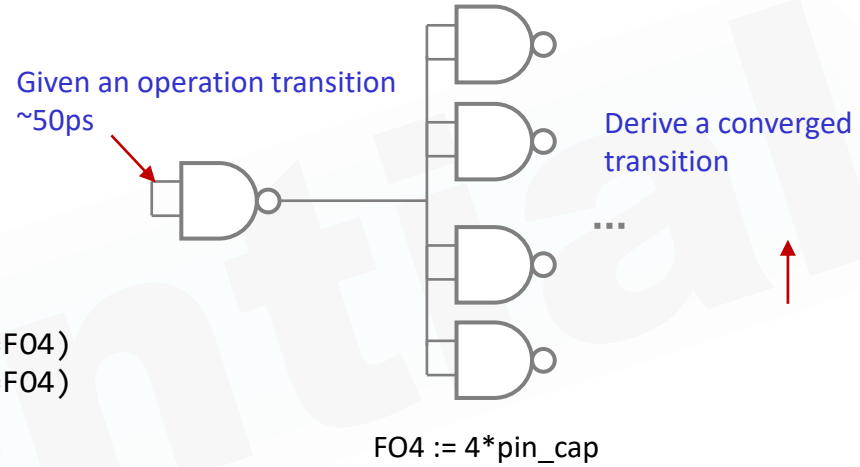
in/out pin list



Derive a proper stage N based on the cell metric

Metric Extraction

```
# grab transition & load of base gate
def baseGateInfo(self,lnode,cell=None,trans=0.05):
    cell = self.enGate['name'] if cell==None else cell
    cnode = self.get_cell(lnode,cell)
    cap = self.lookup_cell_pincap(cnode)
    FO4 = cap*4 # FO4
    ta = trans
    for ii in range(6): # grab converged transition
        tr = self.lookup_cell_timing(cnode,'rise_transition',trans=ta,load=FO4)
        tf = self.lookup_cell_timing(cnode,'fall_transition',trans=ta,load=FO4)
        ta = (tr+tf)/2 # averaged rise & fall transition
    return ta,FO4

# read liberty from CCS
lnode = gro.read_lib('tt0p8v25c_ccs.lib')

# read liberty from JSON
lnode = gro.load_json('tt0p8v25c_ccs.json')

# grab referenced transition & load condition, FO4(ND2)
gro.baseGateInfo(lnode,cell='ND2D1')

# integrity check of the necessary GRO components, specified in the configuration
gro.integrityCheck(lnode)
```

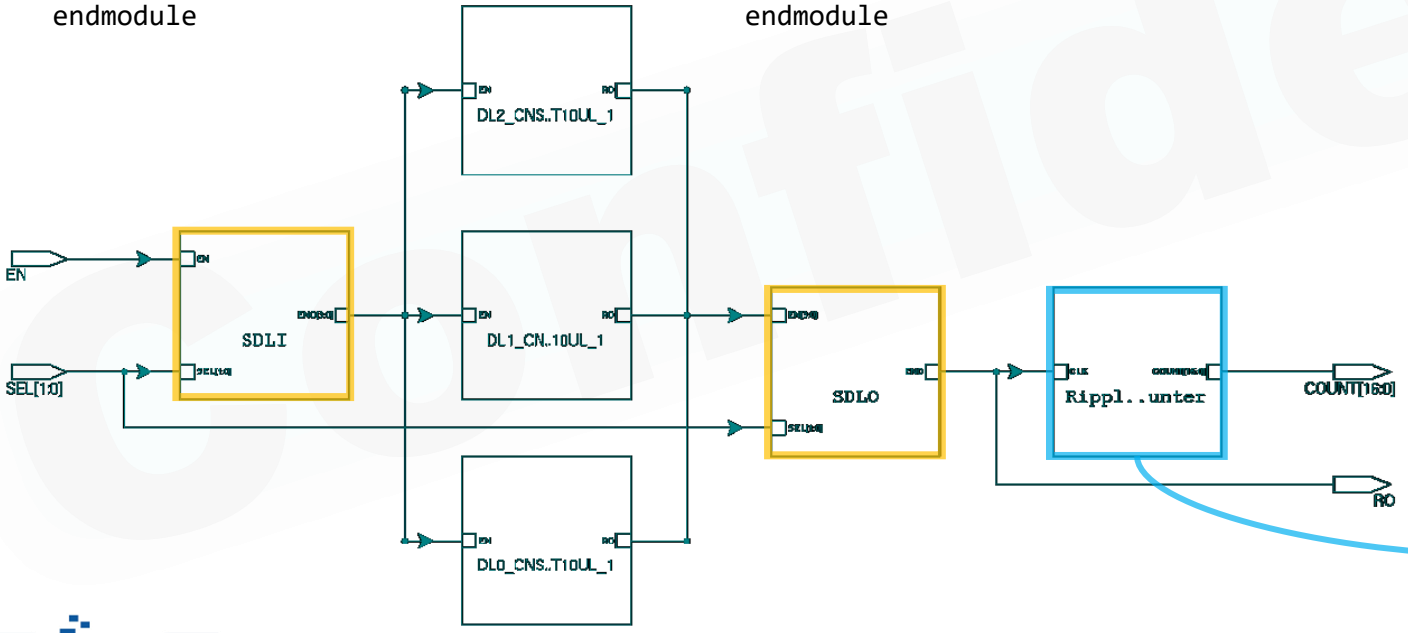


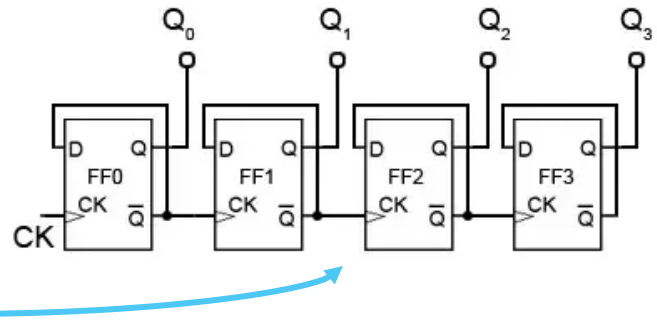
Verilog Module

```
// DL selection encode
module SDLI(EN, SEL, ENO);
output reg [3:0] ENO;
input [1:0] SEL;
input EN;
always @(SEL or EN) begin
case(SEL)
      0: ENO = 4'b0000;
      1: ENO = 4'b0001;
      2: ENO = 4'b0010;
      3: ENO = 4'b0011;
      default: ENO = 4'b0000;
    endcase
  end
endmodule
```

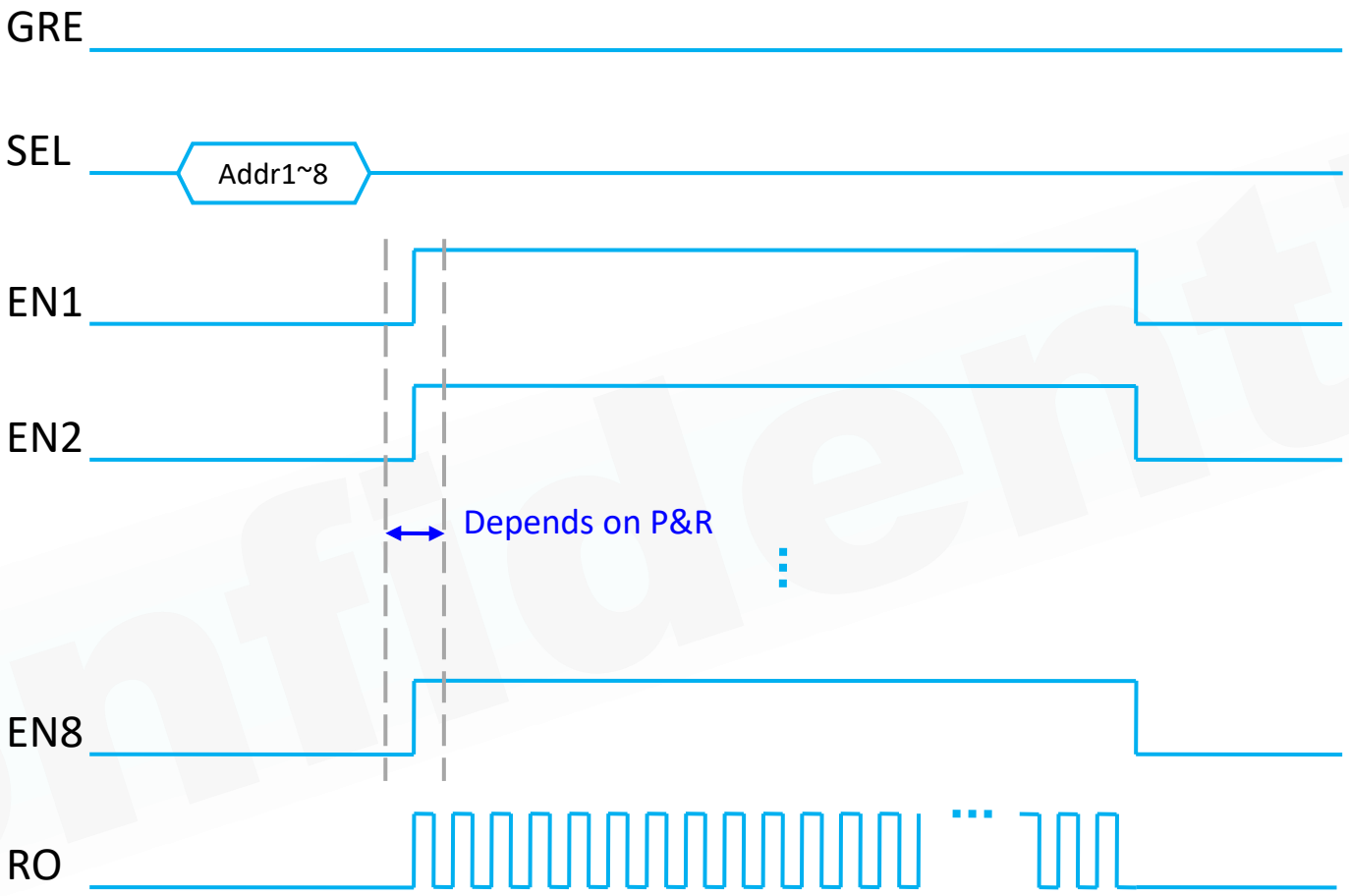
```
// RO output selection
module SDLO (EN, SEL, ENO);
output reg ENO;
input [3:0] EN;
input [1:0] SEL;
  always @ (EN or SEL) begin
    case (SEL)
      2'b00 : ENO = EN[0];
      2'b01 : ENO = EN[1];
      2'b10 : ENO = EN[2];
      2'b11 : ENO = EN[3];
      default : ENO = EN[0];
    endcase
  end
endmodule
```

```
module RippleCounter(CLK, COUNT);
input CLK;
output [3:0] COUNT;
DFQD1 u_dff0 (
    .Q(COUNT[0]),
    .CP(CLK), .D(n_inv0_o));
INVD1 u_inv0 (.ZN(n_inv0_o), .I(COUNT[0]));
DFQD1 u_dff1 (
    .Q(COUNT[1]),
    .CP(n_inv0_o), .D(n_inv1_o));
INVD1 u_inv1 (.ZN(n_inv1_o), .I(COUNT[1]));
DFQD1 u_dff2 (
    .Q(COUNT[2]),
    .CP(n_inv1_o), .D(n_inv2_o));
INVD1 u_inv2 (.ZN(n_inv2_o), .I(COUNT[2]));
DFQD1 u_dff3 (
    .Q(COUNT[3]),
    .CP(n_inv2_o), .D(n_inv3_o));
INVD1 u_inv3 (.ZN(n_inv3_o), .I(COUNT[3]));
endmodule
```

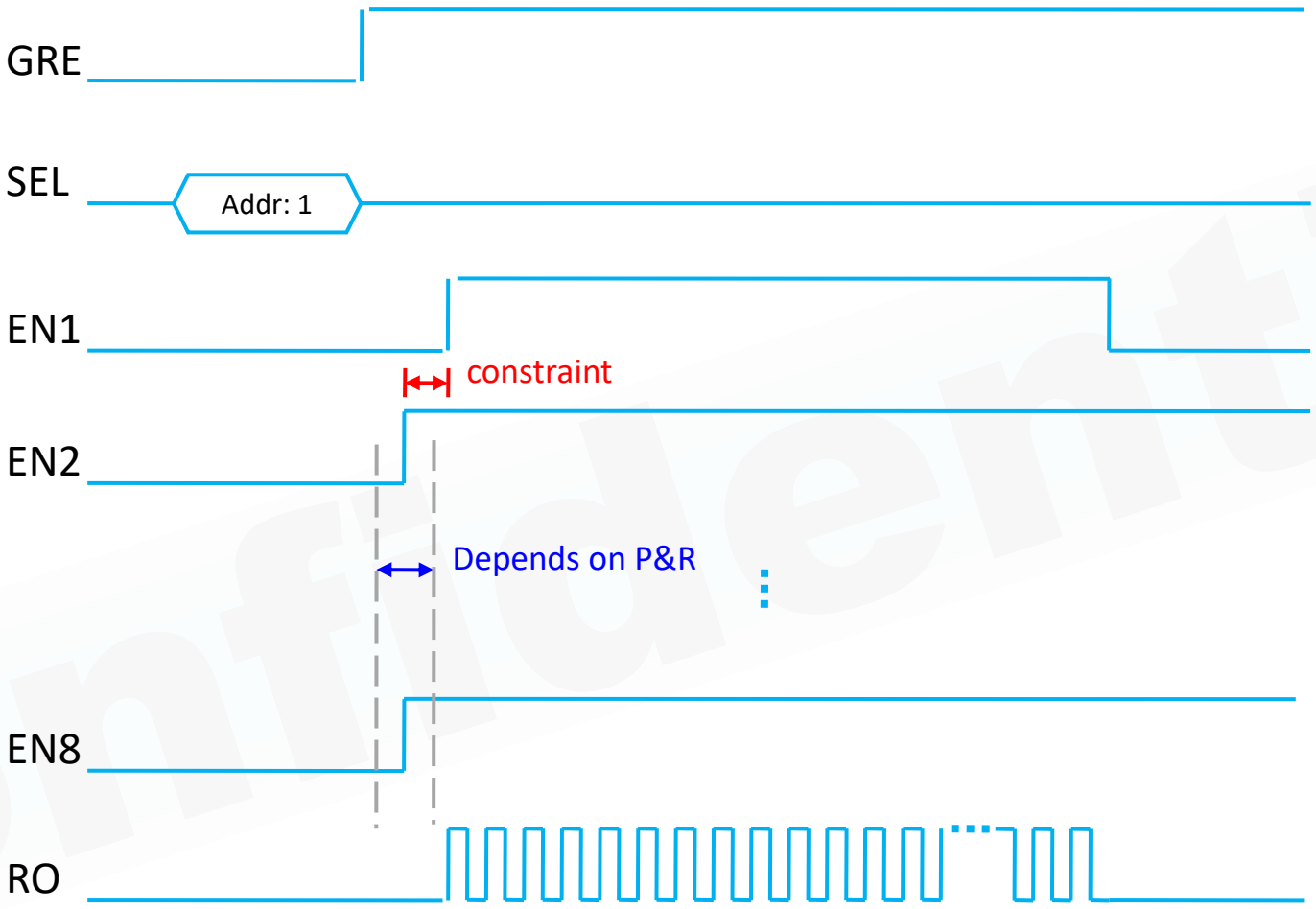




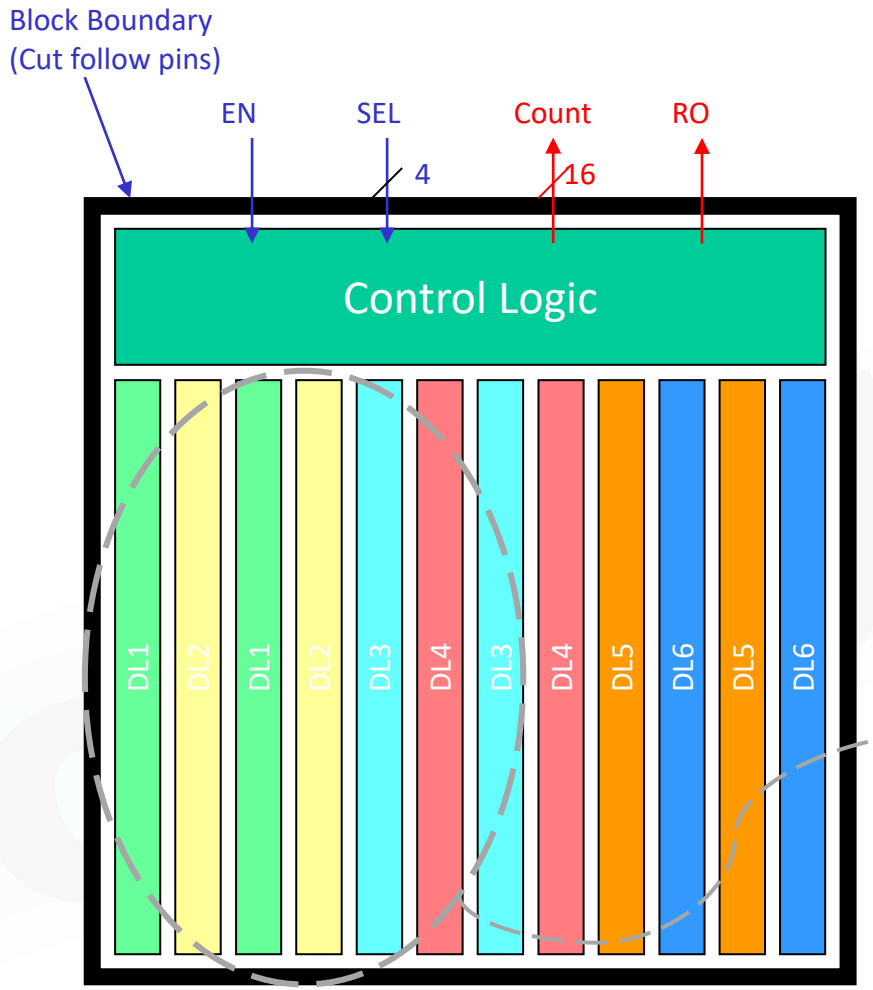
Control Signal (Local Ring)



Control Signal (Gross Ring)

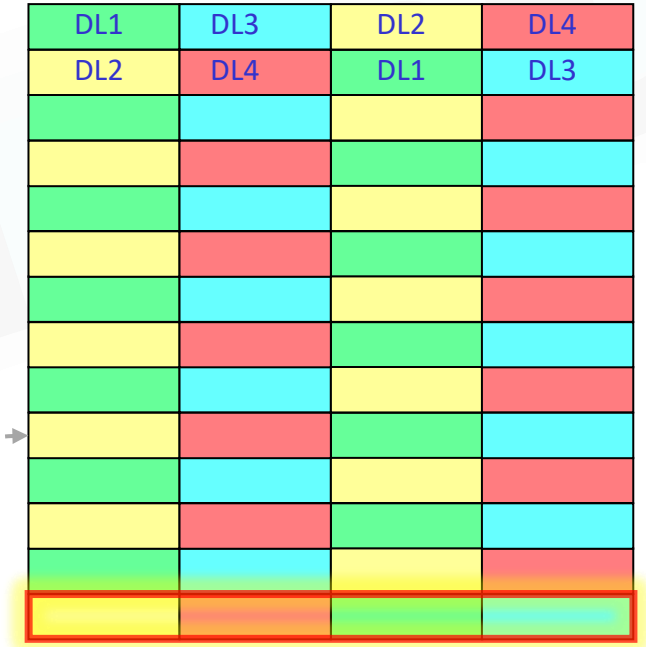


APR Hardening Guideline



1. Arrange delay-line (DL) in vertical
2. Fold DLs into even columns
3. Spread DL cells evenly
4. Stagger DLs evenly

Example



Avoid placing cells in the same row, causing high current
(stagger RO, stagger delay-line)

Test Plan – Voltage/Temperature Collection

```
// Voltage & RO characterization
Temperature: {-40, 25, 85, 125} begin
    Voltage Sweep: 0.7Vdd~1.3Vdd, Step 20mV begin
        Chain-Selection: 0~3 begin
            Dump RO1~N counter: loop x3
        end
    end
end
```