RedHawk-SC Quick Start Training
Release: 2022_R2.x

Aug 31st, 2022

Ansys

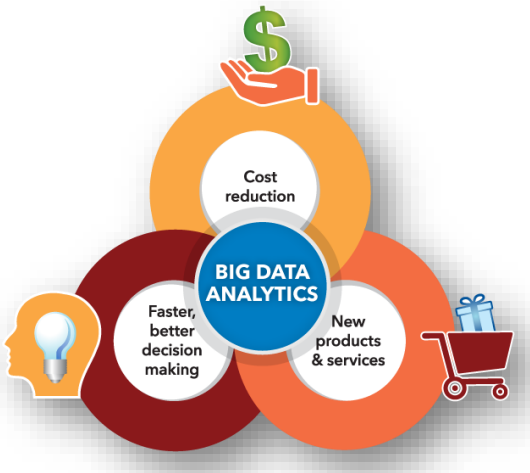
Training Agenda

- Introduction to Big Data and SeaScape Platform
- Challenges in Power Integrity Signoff and Methodology
- Setting up RedHawk-SC & Performing EM/IR Analysis
  - Reading in input data and performing data integrity checks
  - Grid robustness and construction integrity checks
  - Power Calculation
  - Static voltage drop analysis
  - Dynamic voltage drop analysis
  - Electromigration analysis
- Introduction to SC Help system & APIs

Introduction to Big Data and SeaScape Platform

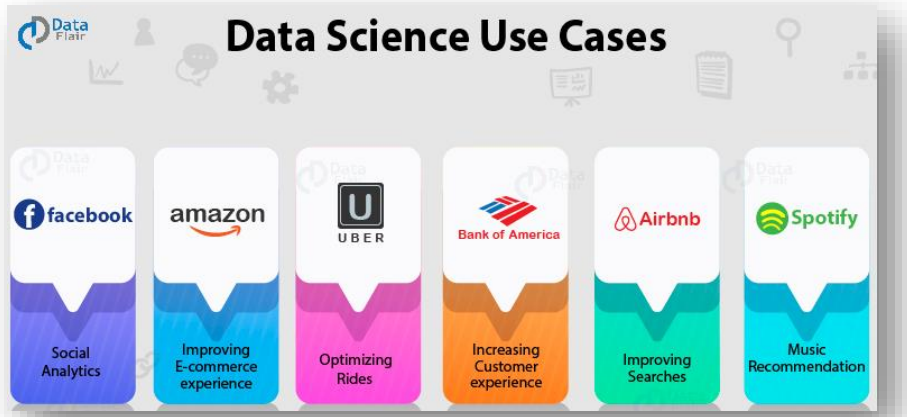
Ansys

Introduction to Big Data Analytics

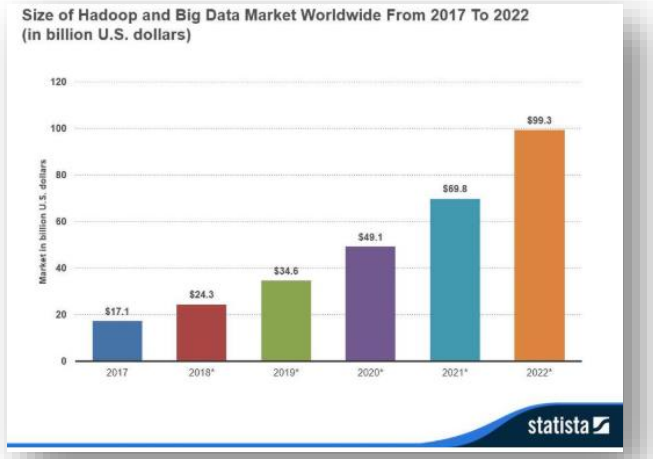


How big data analytics help ?
Image courtesy : https://sas.com

Challenges in Big Data
Image courtesy : https://edureka.com



Top Applications of Big Data
Image courtesy : https://data-flair.training



Growing market of big data science – 100 B dollar by 2022
Image courtesy : forbes.com

"Big Data Analytics" is the use of advanced analytic techniques for large, diverse data sets that include structured, semi-structured and unstructured data, from different sources, and in different sizes from terabytes to zettabytes

Big Data Techniques

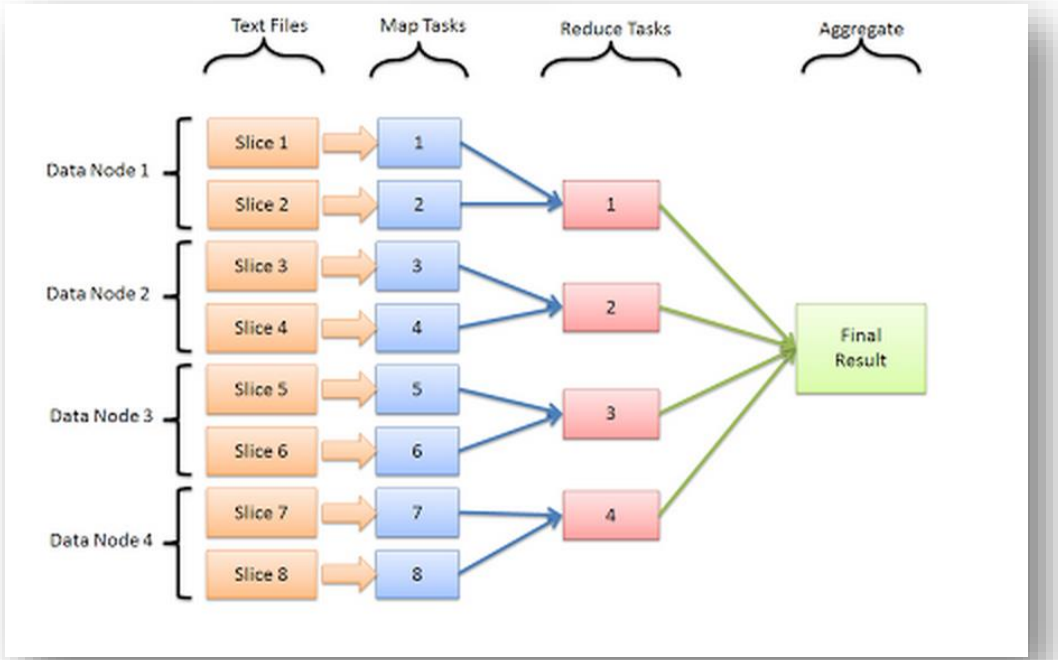
Google search says :

MapReduce is the heart of **Apache® Hadoop®.** It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a **Hadoop** cluster.

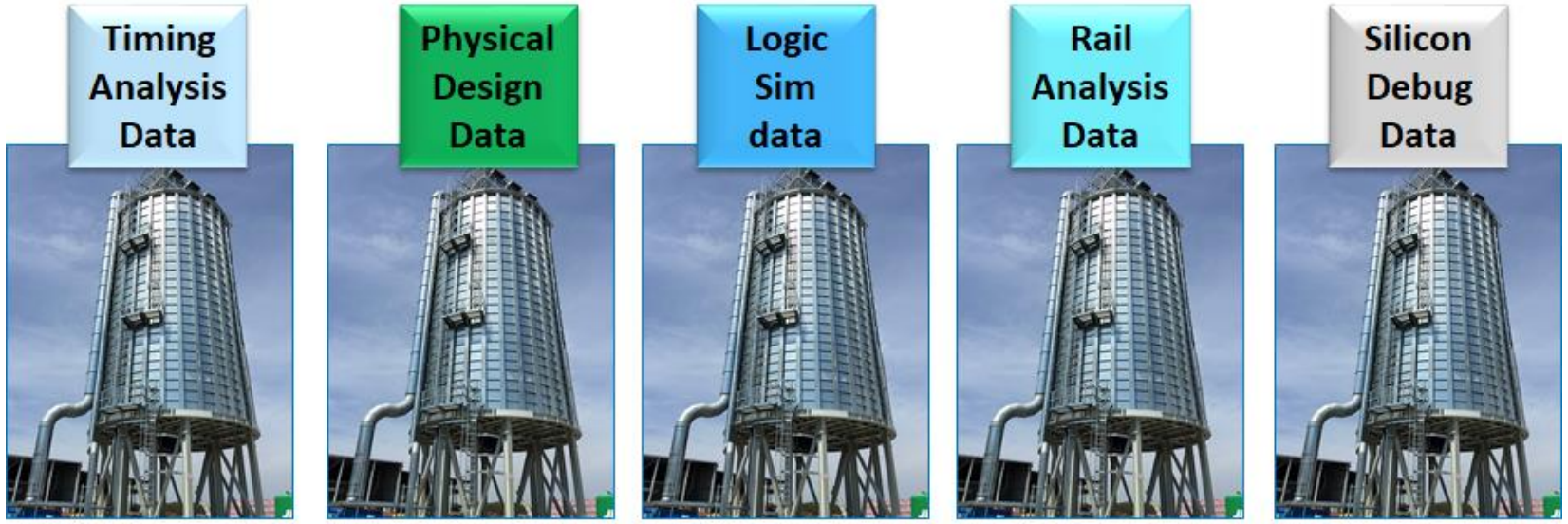
Ansys says :

Scheduling algorithms similar to MapReduce are the heart of **SeaScape®.** It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in **customer** cluster.

- **MapReduce** is a simplified model for processing large data sets with a parallel, distributed algorithm on a cluster in an automated, easy to write, fashion
- All big-data engines use MapReduce or its derived/augmented techniques
- Some popular big data engines below:



Traditional EDA Big Data Exists in Silos



Multi-Domain Chip-Level Analysis is not well supported by existing EDA tools and databases

Problem with Silos

Design margins too large

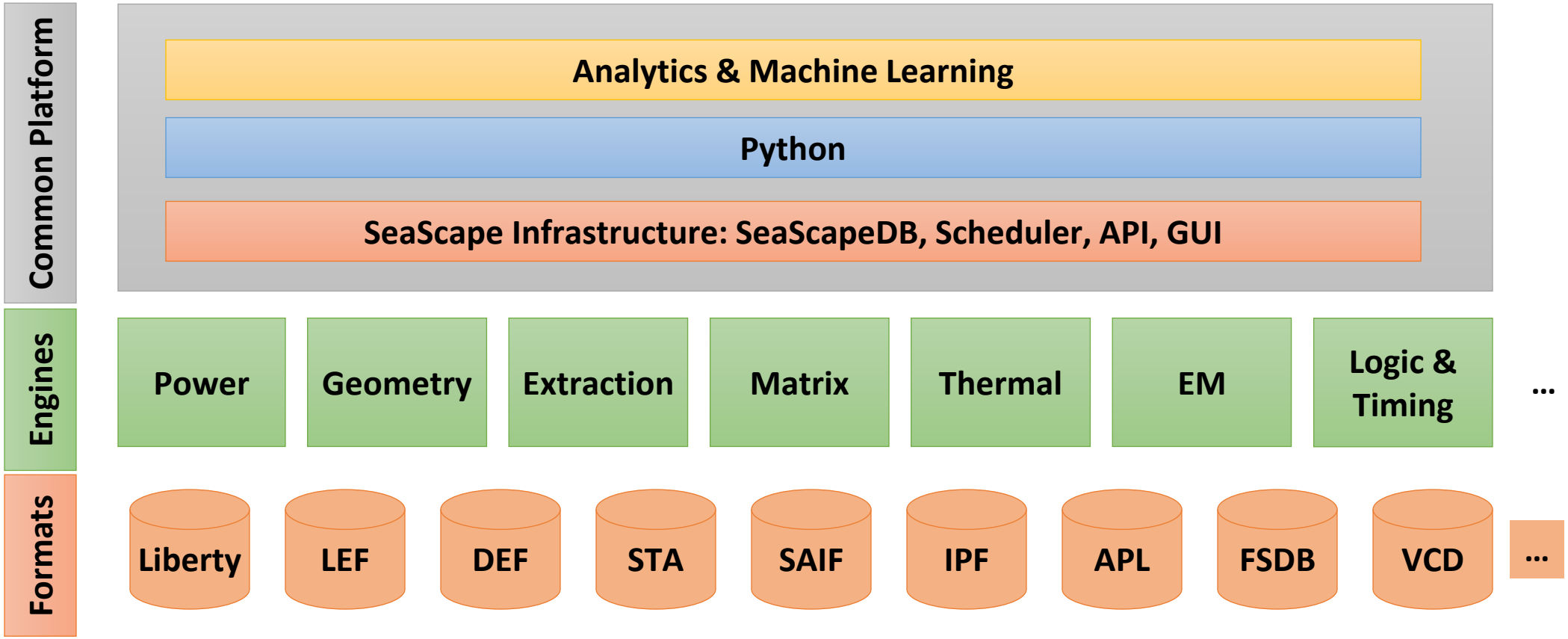
Coverage is poor

Results are not actionable

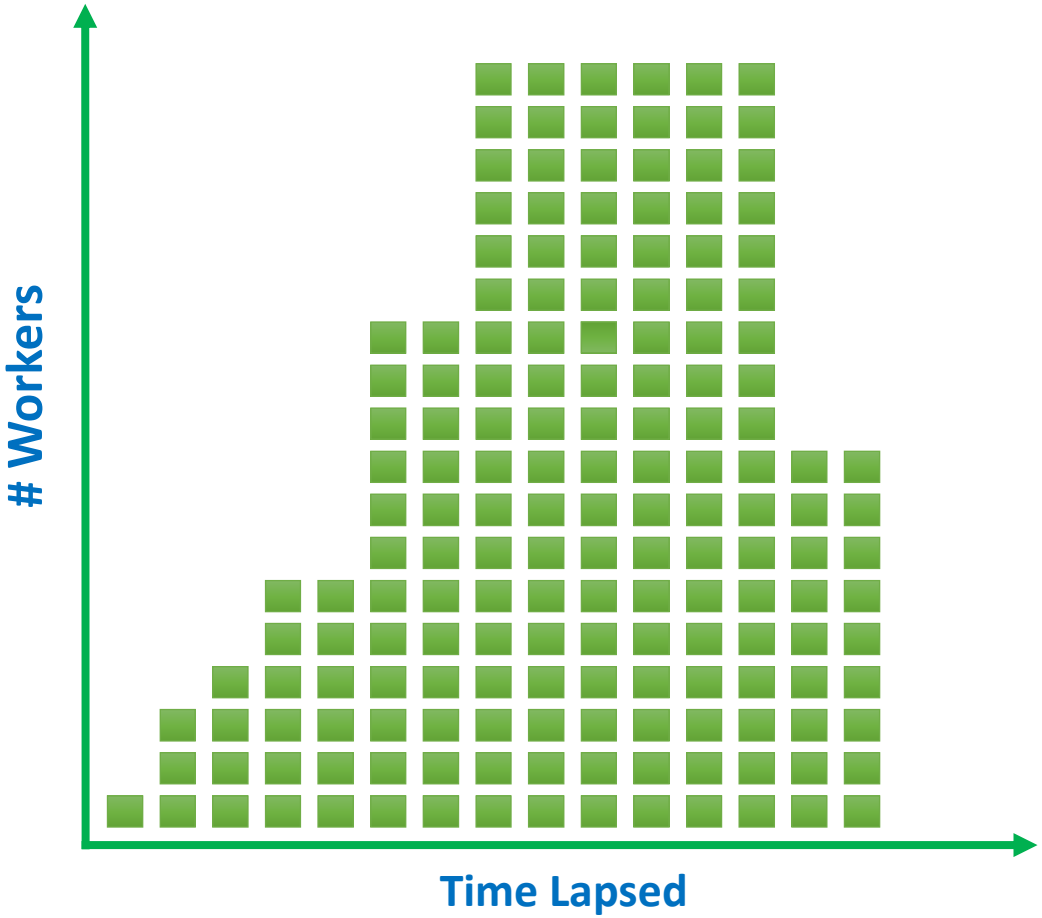
The SeaScape Platform

EDA-specific platform for distributed data and compute

1. SeaScape is the industry's first big data platform for Actionable analytics, using Big data and machine-learning apps
2. All products of ANSYS will work with the SeaScape platform.



SeaScape Elastic Compute



Launch runs immediately
- No waiting for full machines
- Instantly view results on single cores

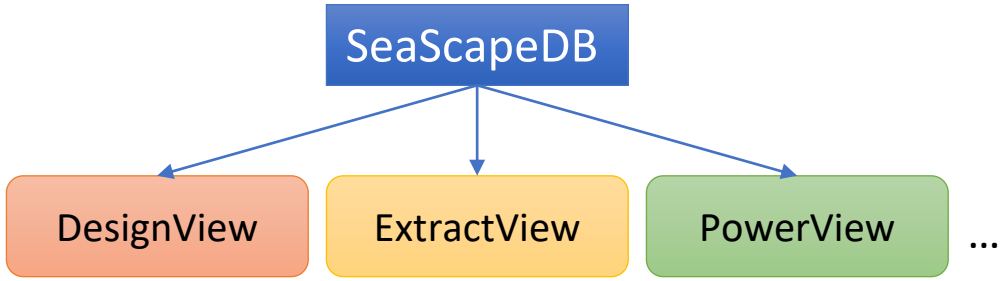
Auto checkpoint
- Reuse data from terminated runs
- Build multiple flows & reuse data

Resiliency
- Auto-recover from bad cores
- Or network issues

Parallelization
- All stages are scalable across cores
- Matrix solve > 1,000 cores

- Processing starts as soon as one worker is available
- Performance improves as more workers are added
- Tool will automatically launch more workers if needed for a specific view

The SeaScape Database



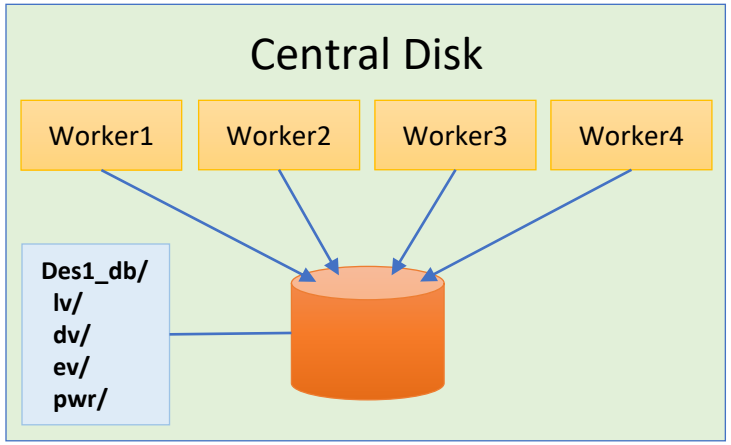
- SeaScapeDB holds all data used within SeaScape
- Data from external formats like DEF or FSDB
- Results of processing within the tool such as instance power

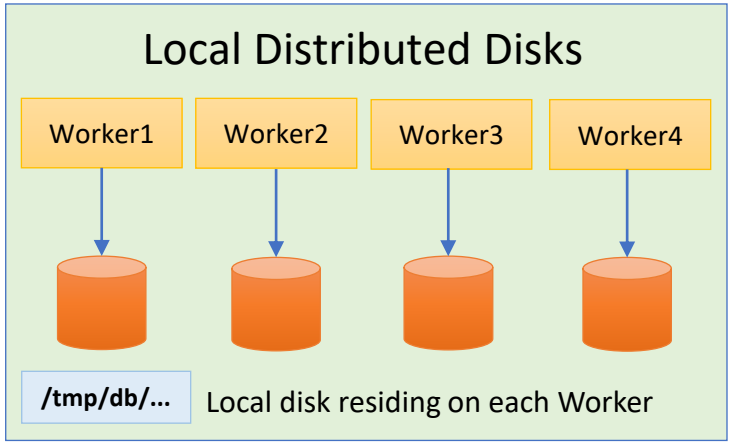
- One DB contains multiple views
- Each view holds related data
- All data is partitioned into chunks for efficient processing

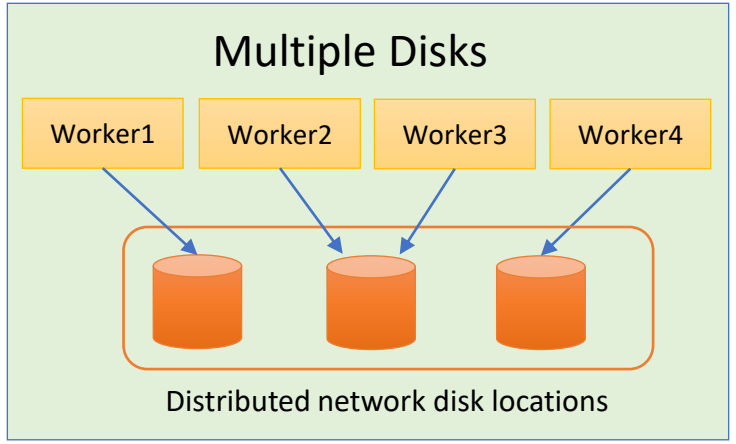
- Data is initially stored in each worker's memory, then later written to disk in the background

- Can read and/or write to multiple DBs in one tool session

DB Storage Options



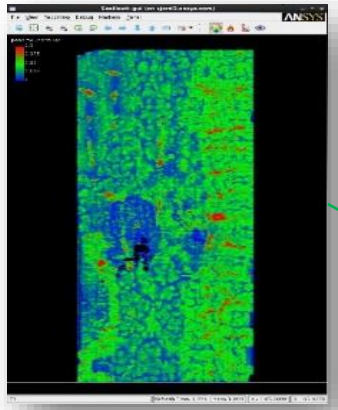




An example of big data analytics application using SeaScape platform

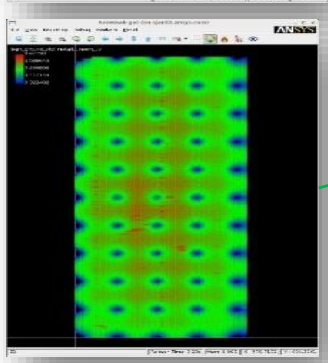
RAIL DATA

CURRENT CROWDING



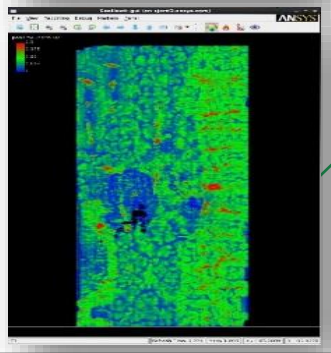
RAIL DATA

RESISTANCE



TIMING DATA

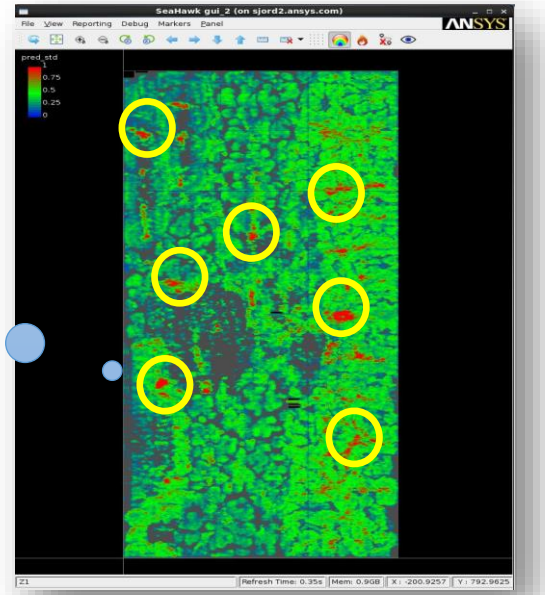
SLACKS

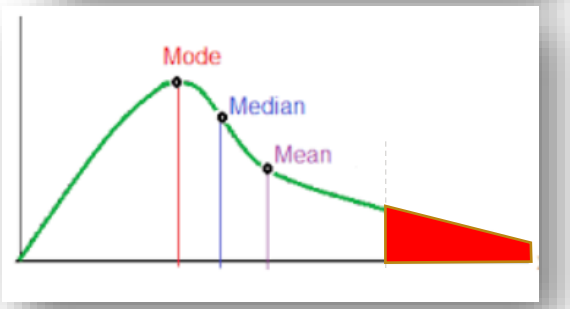


Example Analytics

highlight instances which are timing critical, with high PG resistance and placed in high current crowding area

Multi-metric Analytics





PHYSICAL DESIGN DATA

PER LAYER CONGESTION

LIBRARY DATA

CELL VOLTAGE SENSITIVITY

Statistical distribution of outliers from SC data-analytics can help in finding design weakness

ANSYS Technologies for Electronic Systems

POWER BUDGETING

PowerArtist™
RTL Power Analysis, Reduction

IP VALIDATION

Totem™:
- IR, EM and DvD
- Model for SoC

ON-CHIP ELECTROMAGNETICS

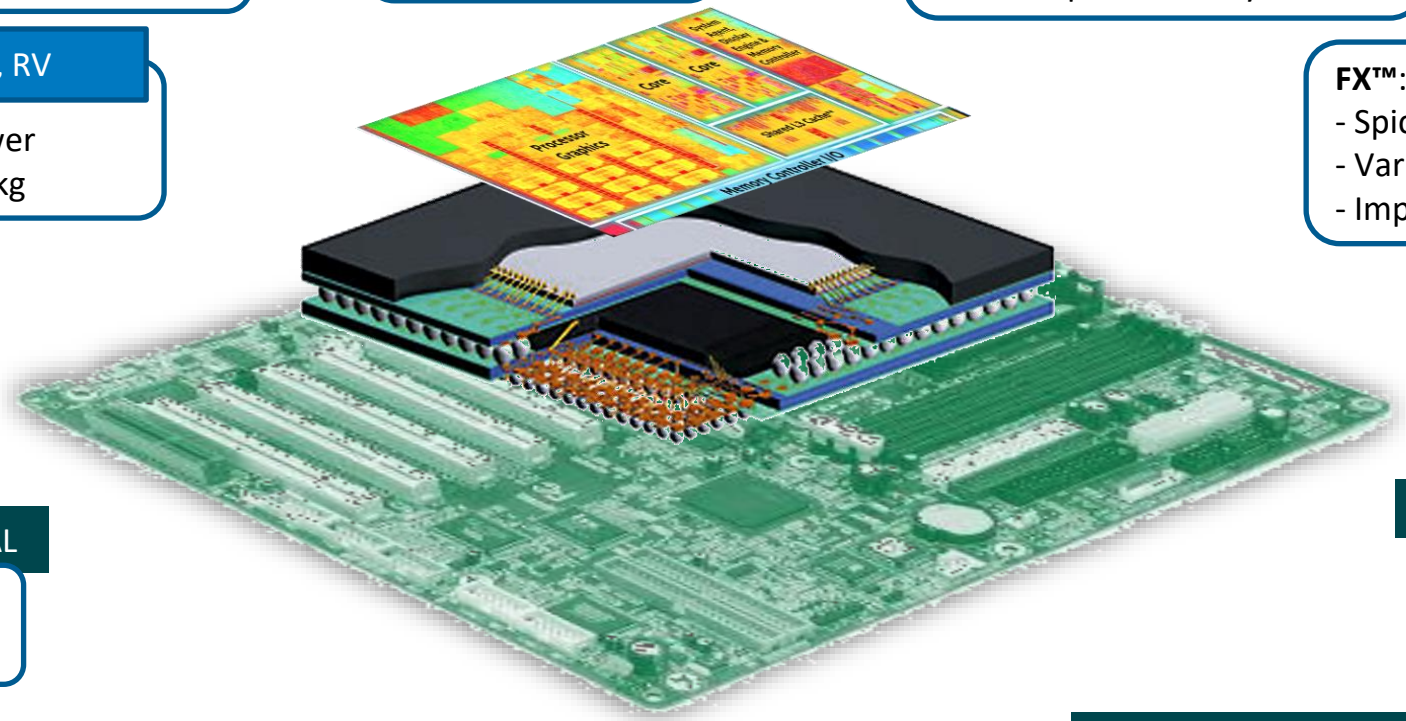
HELIC™:
- On Chip EM crosstalk for RF, High-speed digital
- On-chip Inductor Synthesis

PDN, RV

RedHawk-SC™
PDN planning, In-rush, Power Integrity, EM, Integrated pkg

TIMING/VARIABILITY

FX™:
- Spice-accurate timing
- Variability modeling
- Impact of DvD on timing



IC → SYSTEM

CPM™:
- System PDN
- System Thermal

ESD PROTECTION

PathFinder™: ESD sign-off

PACKAGE/PCB ELECTRICAL

SIwave/HFSS/Q3D:
Early to sign-off

MECHANICAL STRESS

ANSYS Mechanical
Stress, deformation

CONNECTORS

HFSS/Q3D

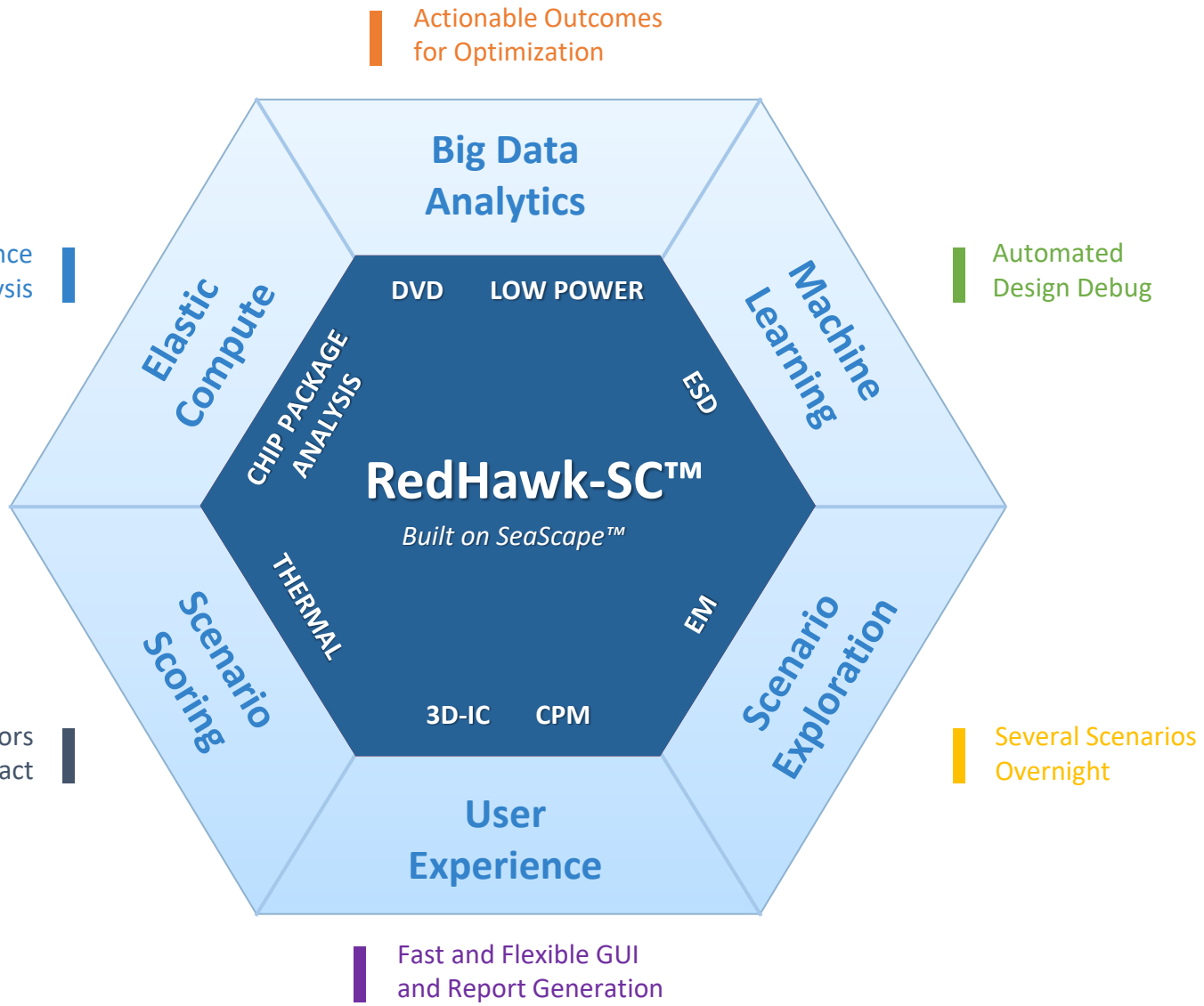
THERMAL PLANNING

Icepak
Chip-aware system thermal

IO DDR DESIGN

DesignerSI/SIwave/CSM™:
- IO ring verification
- System jitter prediction

RedHawk-SC™ Product Features



Challenges in Power Integrity Signoff and Methodology Overview

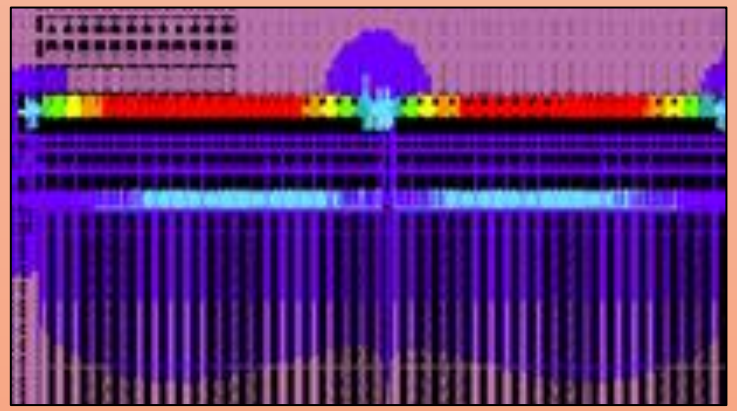
Ansys

What are Weaknesses of Power Delivery Network (PDN)?

PDN WEAKNESSES – Problem is Local *and* Global

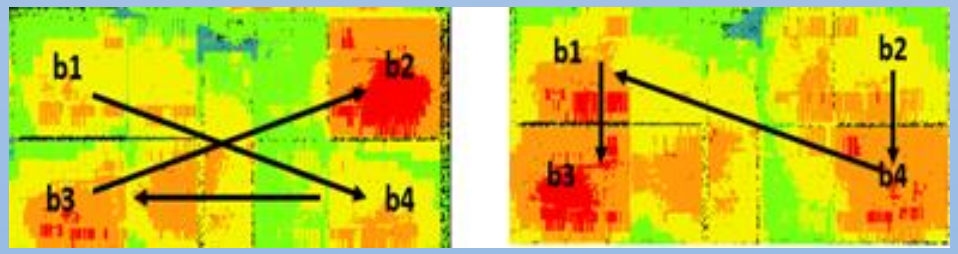
LOCAL DESIGN WEAKNESS

Instance level rail collapse happens from combination of localized simultaneous switching and power grid weakness (bump to transistor)



GLOBAL DESIGN WEAKNESS

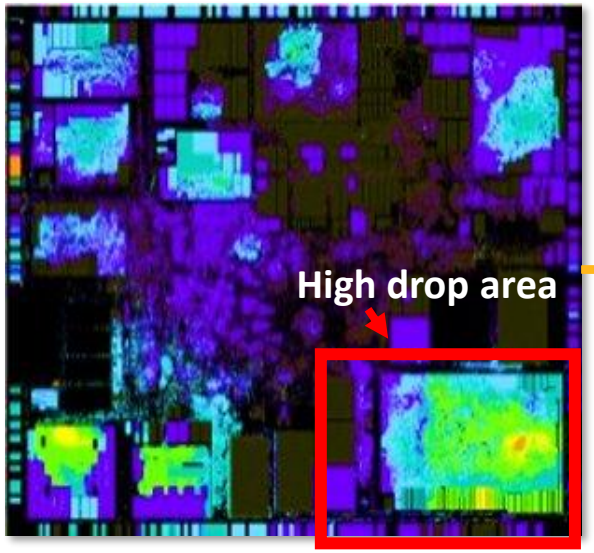
Global rail collapse happens when individual CPU cores fire at different times under different scenarios and generate “global” switching scenarios that resonate with the chip-package impedance



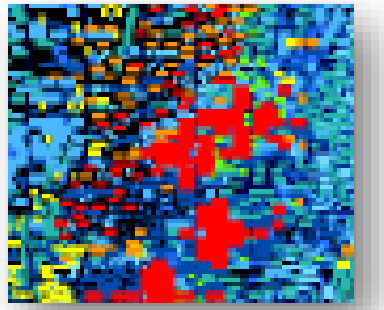
Simulation workflow needs to consider:

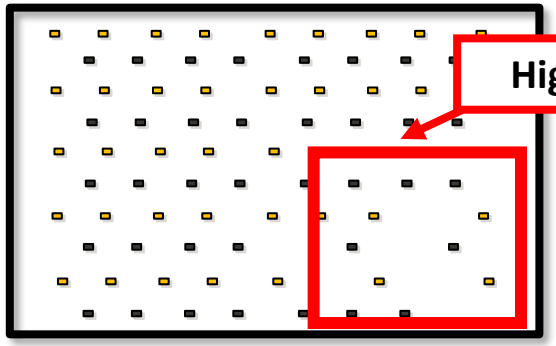
- Instance specific switching (Spice-accurate current, IR, decap, bump placement..)
- SoC + package/PCB interactions (block switching, pads, Pkg/PCB RLCK)

PDN issues are vast in nature



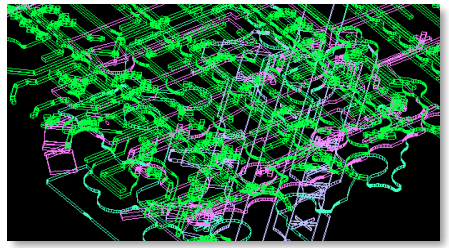
- Cluster of high-power cells
- Driving high load
- Firing at same time





Poor bump and RDL design

- Insufficient package layers
- Routing congestion
- Fragmented routing ~ high L



PDN issues are vast in nature (continued)

25+ design issues leading to weak PDN

HIGH POWER

- High C-Load
- High Fanout
- High Frequency
- High Toggle Rate

POWER GRID WEAKNESS

- Insufficient number of Layers
- Inadequate width-spacing
- Disconnected wire/via
- Disconnected instances
- Shorts
- Missing via
- Missing strap
- Unterminated strap
- Unconnected macro pins

PAD RELATED ISSUES

- Insufficient number of pads
- Uneven pad distribution
- Weakly connected pads

SWITCH RELATED ISSUES

- Insufficient number of switches
- Switch placement issues
- Weakly connected switches
- High switch resistance

PACKAGE RELATED ISSUES

- High Package Inductance
- Inadequate package routing
- Uneven Package Parasitic Distribution

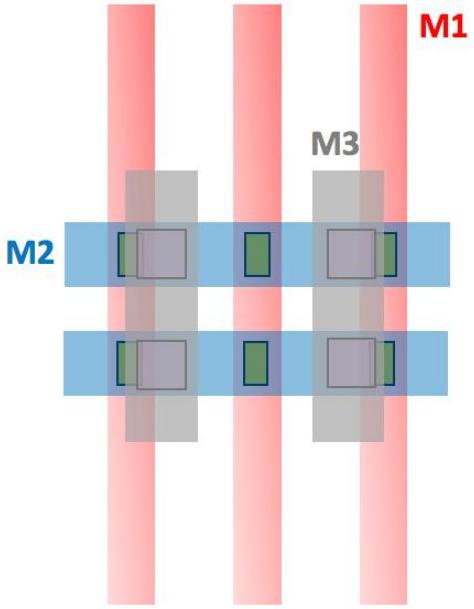
DECAP ISSUE

- Insufficient decap
- Insufficient lower metal routing
- Lower Tech Nodes

SIMULTANEOUS SWITCHING

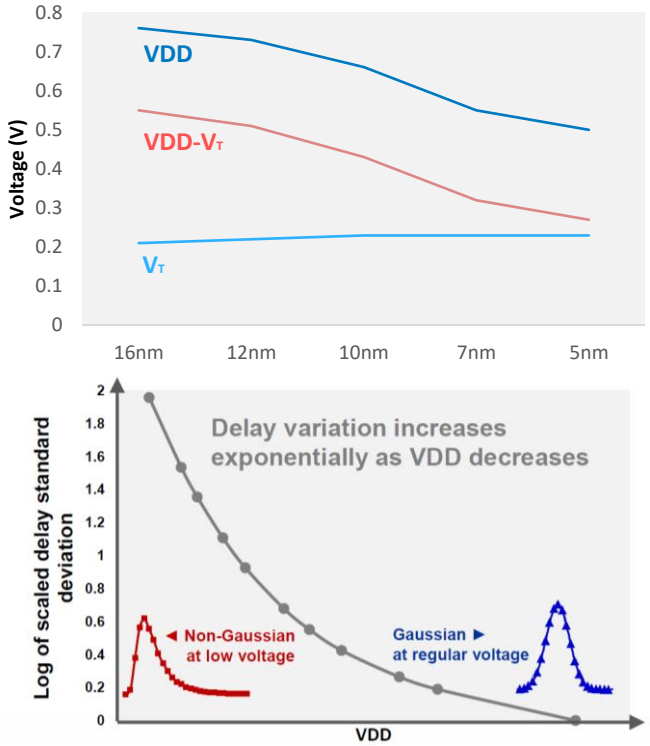
- TW clustering
- Clock tree clustering
- Scan-shift scenario
- Architectural issues

7nm Power Integrity Challenges



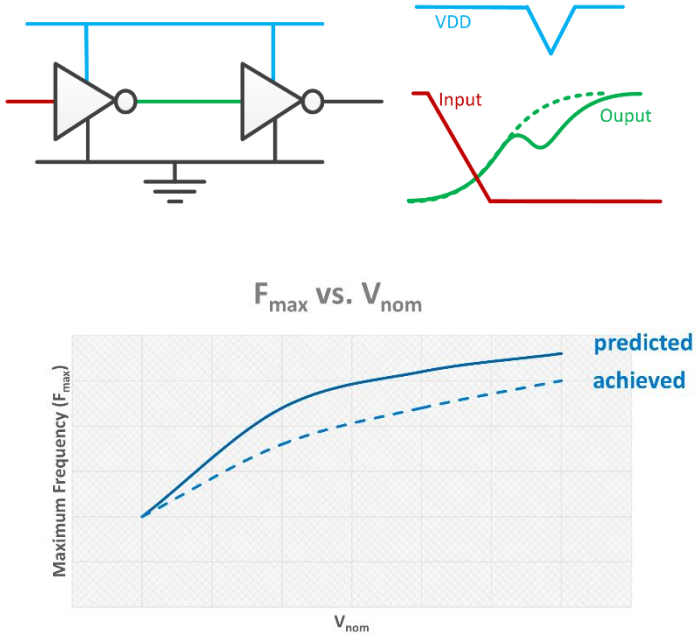
5x

Increase in grid complexity, compared to 16nm. Power grids have 10B+ nodes



600mV

Ultra low voltage computing means margins are razor thin, and variability is severe



10x

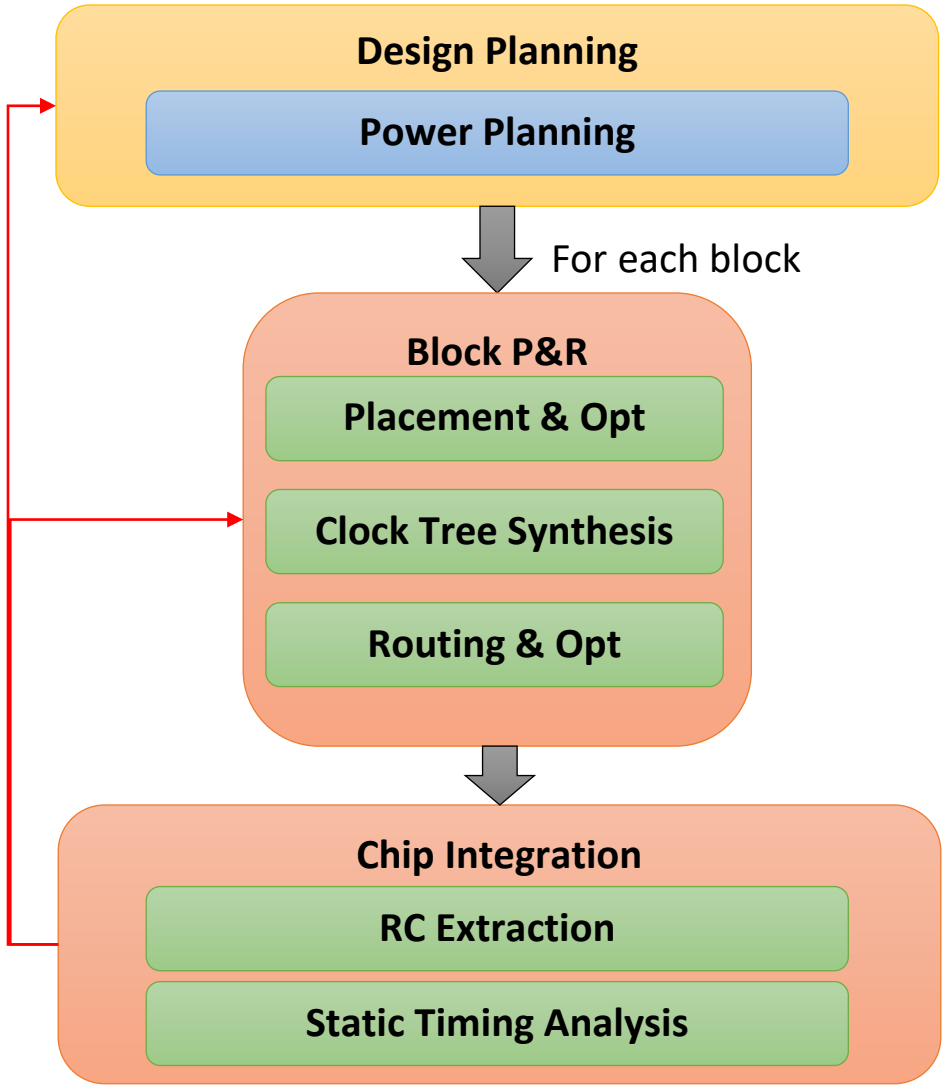
Need for increased scenario coverage, to ensure voltage and timing

Sources: ANSYS, Paolo Gargini, ITRS Past, Present and Future, TSMC: Physical Design Challenges and Innovations, ISPD 2017

7nm Implications

Signoff Methodology	• In older technology nodes, simple pass/fail thresholds were ok. This is no longer true
Silicon issues	• Silicon does not work or does not reach desired clock frequency • Critical for mobile processors and other low-voltage designs
What to do?	• Consider impact of dynamic voltage drop on timing as a key metric • Reorder paths with accurate voltage drop effect considered • Use full set of available analytics to identify and fix issues in grid and cell placement

Power Integrity in the Design Flow



Identify & Fix issues early

✓ **Early Grid Quality Checks**
- Grid robustness and construction integrity checks
- Build Quality Metric [BQM] based early analysis

Fix local effects at block level

✓ **Block-level Analysis & Signoff**
- ✓ Placement Check for Local simultaneous switching using Peak Timing Window Overlap [PeakTW] checks
- ✓ Static & dynamic voltage drop analysis
- ✓ Power/Signal EM analysis
- ✓ Power Network Optimization

✓ **Fullchip Analysis & System-level PI closure**
- ✓ Full-chip static & dynamic voltage drop analysis
- ✓ Power/Ground Electromigration (EM) analysis
- ✓ Chip Power Model (CPM) generation for package + die analysis

Static Analysis: Necessary But Not Sufficient

FUNCTION & TIMING

Gate-level Verilog simulation

+

Static Timing Analysis

+

RTL to Gate Equivalence Checking

VOLTAGE DROP

Static Rail Analysis

- Identifies weak grid issues
- Finds PDN issues due to Constant/Avg currents

+

Dynamic Rail Analysis

- Considers inductance and cap effects
- Considers dynamic current waveforms
- Models realistic switching behavior

It is not possible to bound the dynamic problem with worst-case static currents everywhere (would overwhelm the grid)

Dynamic Analysis: Challenges

Grid Complexity
- Geometries: 10B+
- Extracted circuit nodes: 10B+
- Sparse Matrix size: 10B x 10B +
- Memory needed: 2TB+
- Local + Global coupled system

Logic Complexity
- Instances: 2B+
- Flip-flops: 100M+
- Macros: 1M+
- Reachable states is infinite

Timing Complexity
- # of timing paths: huge
- # of arrival times combination is infinite
- Each instance has a range of arrival times (from logic and PVT, SI effects)

How to get high coverage in reasonable runtime with good accuracy?
- Identify the issues that are most likely to lead to a real silicon issue
- Help to determine the root cause
- Help to repair the issue

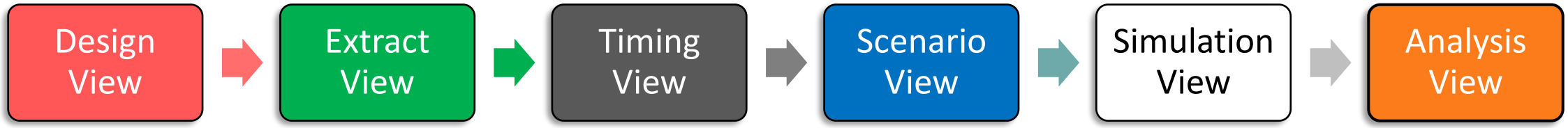
Setting up RedHawk-SC

SeaScapeDB – A View Oriented Data System

Seascape DB Overview

- A SeaScapeDB is simply a linux directory containing views
- The db is created through `'open_db'` command
- Directory name for the DB is set in the `'open_db'` command
- If db already exists, the db will be opened
- DB contains several views; each view is a sub-directory in the DB
- Views are created through `'create_<type>_view'` command
- Directory name for the view can be specified while creating the view
- Multiple SeaScapeDBs can be opened simultaneously
- A DB can be accessed simultaneously from multiple RH-SC sessions

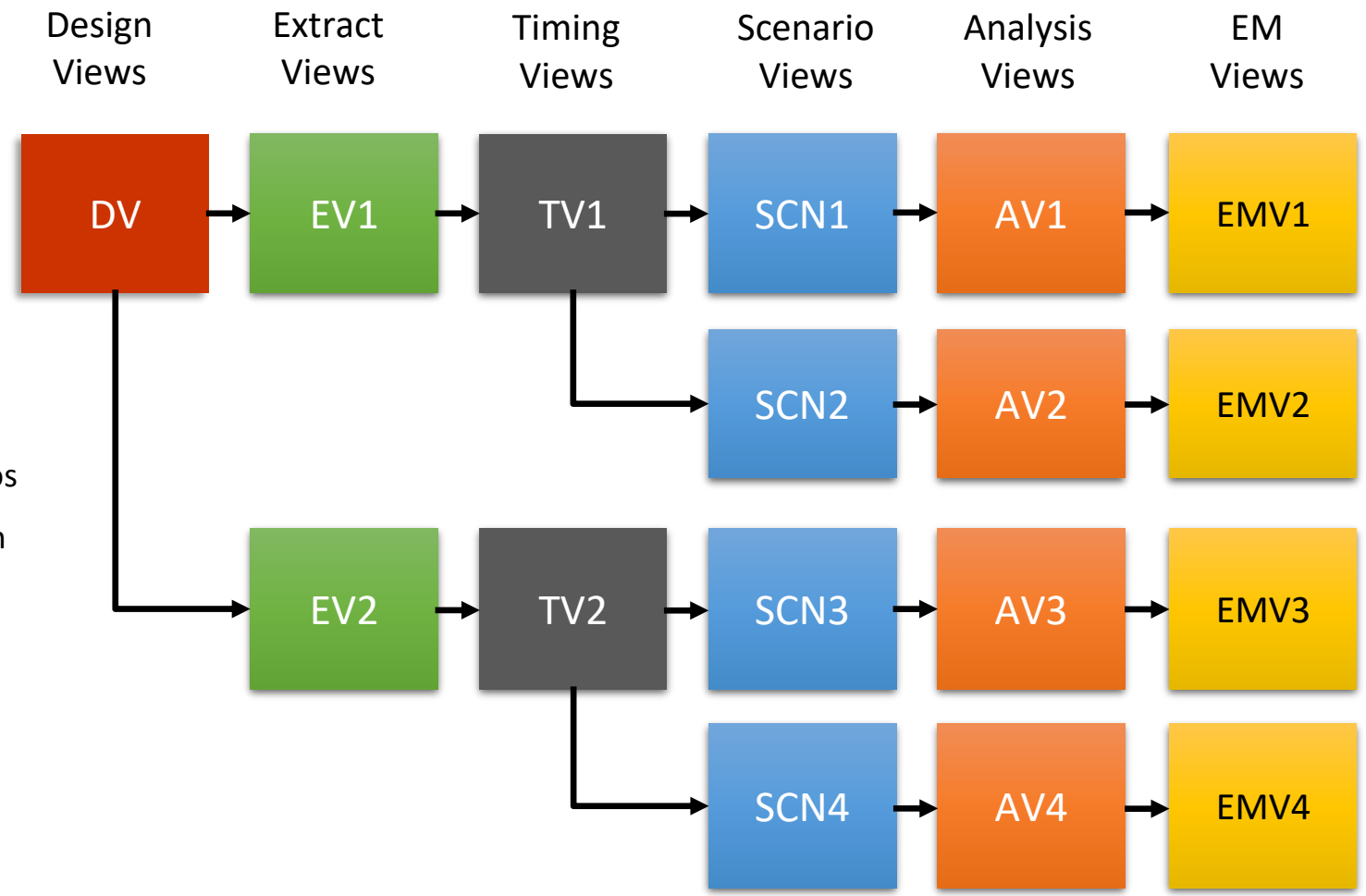
Views in SeaScape DB



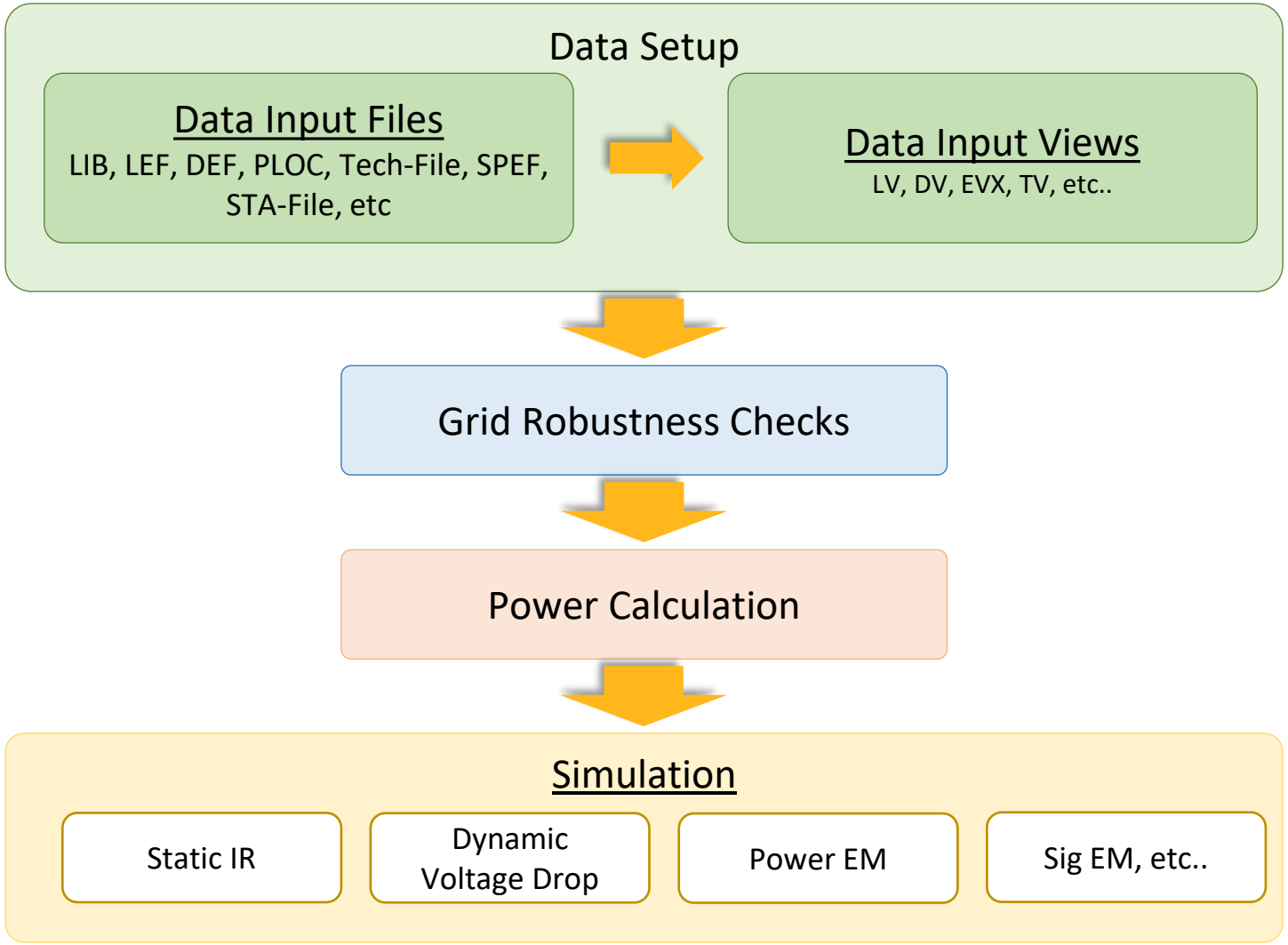
- Views are sub-directories inside SeaScape DB
- They are representations of a group of data
- Each view stores data from different stages of analysis
  - Write once, read many times
  - Automatic version is maintained when a view is over-written
  - Tag/directory name will have additional #v1, #v2 … appended
    - Example: tag='tv', on disk will be: ./<SeaScapeDB>/tv#v1
- It is possible to restart the run from any view for faster TAT
- Views are objects with several attributes which can be queried

Views - Example

- Single session can have views for:
  - Different corners (Liberty and Extraction)
  - Different conditions (voltage, temperature)
  - Different modes of operation
  - Different vector sets
  - Different vector-less settings
  - Different conditions for IR drop versus EM
- Example
  - SCN1, SCN2 represent VCD & V-less scenarios
  - EV1 and EV2 represent 2 different extraction corners
- Related views can be traced-back
  - EMV4 ◀ AV4 ◀ SCN4 ◀ TV2 ◀ EV2 ◀ DV



Main Steps in Analysis Flow



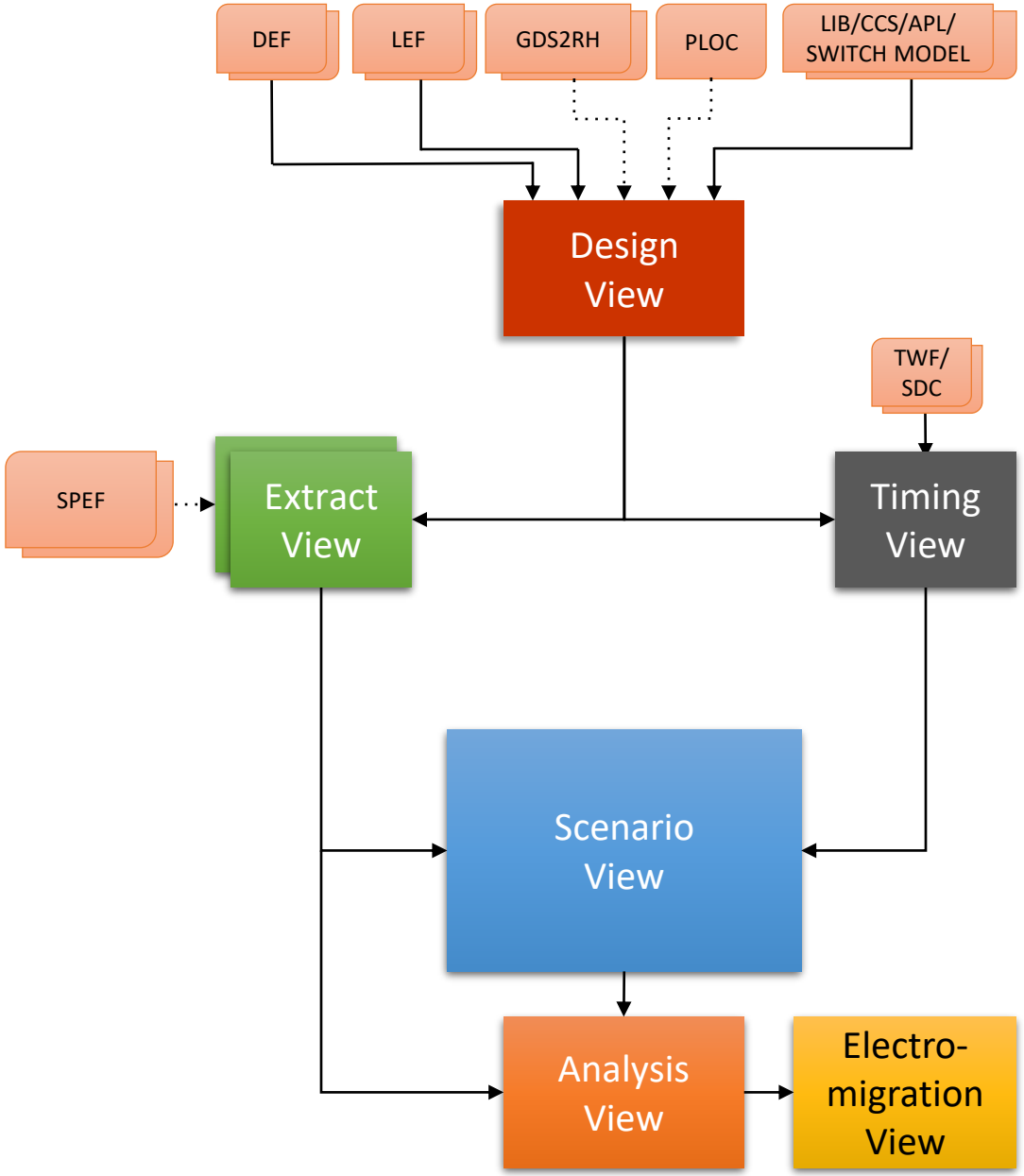
Flow Overview:

- The flow diagram shows various steps involved in Power Integrity Analysis
- Each step in the flow involves multiple view creation steps
- Though the steps are shown sequentially in the flow diagram, many view creations can happen in parallel
- Scheduler will manage the view dependencies & parallel execution

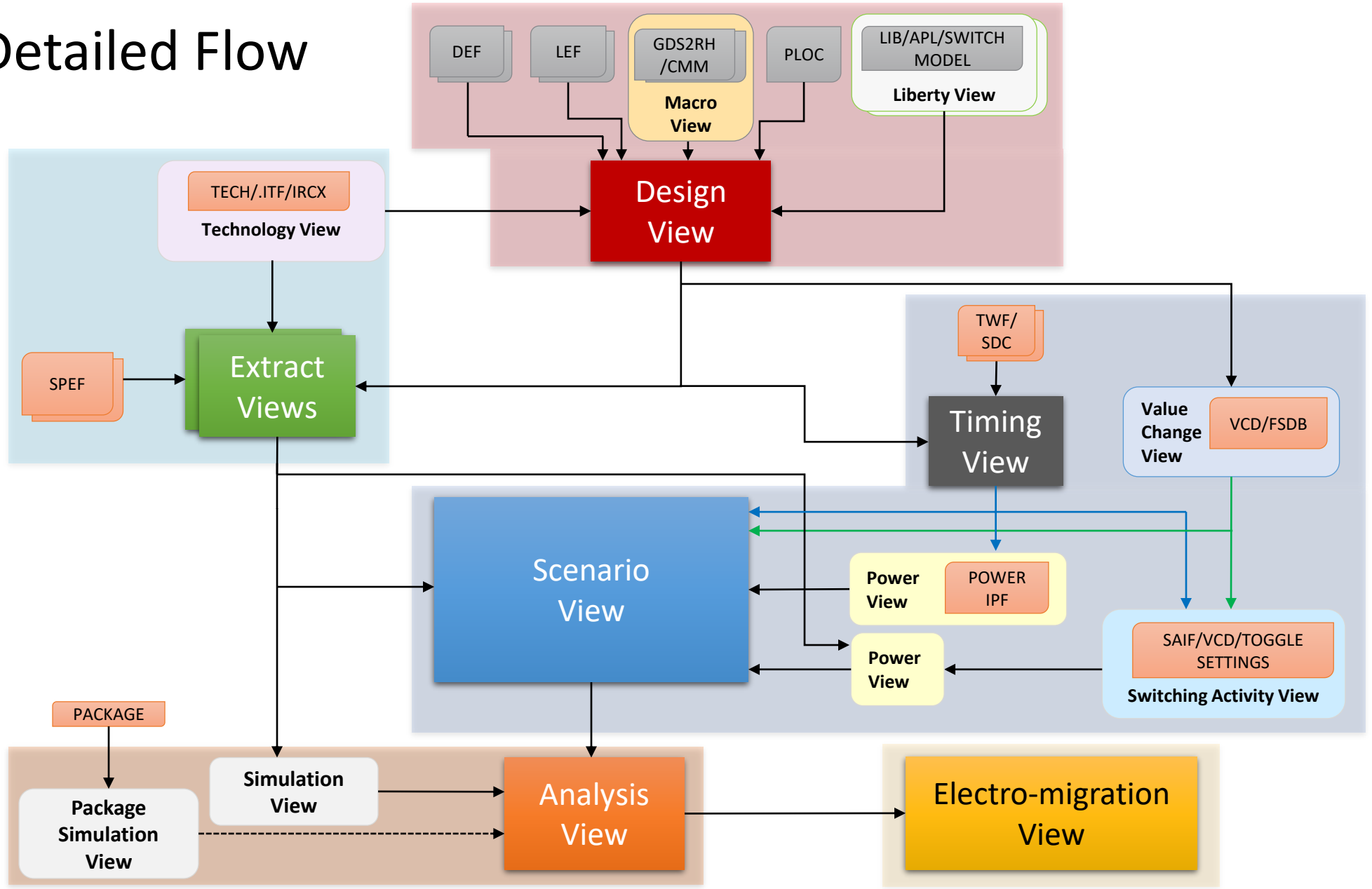
Flow Overview

Main SeaScape Views

View	Function
DesignView	• Stores netlist and layout information • Library information (through LibertyView)
ExtractView	• Stores RC extraction & SPR data • Imports and stores SPEF
TimingView	• Stores imported TWF and SDC constraints • Stores clock associations and slews • Computes timing graph
ScenarioView	• Calculates DC instance currents for static scenario • Stores logic events and transient instance currents for dynamic scenario
AnalysisView	• Stores result of simulation
ElectromigrationView	• Reads in EM rules and produces EM results on every wire/via segment



More Detailed Flow



Supplementary SeaScape Views

View	Function
LibertyView	• Imports and stores Liberty, CCS models and APL models
MacroView	• Read in macro models such as gds2rh, gds2db, Totem CMM models

TechView	• Stores extraction technology information
	• Below 16nm: Ansys tech file only
	• 16nm and above:
	  • Ansys tech file, ITF or iRCX
	  • NRC (derived from ITF or iRCX) includes detailed characterized Capacitance Modeling

ValueChangeView	• Stores imported FSDB/VCD
SwitchingActivityView	• Stores toggles – propagated/unpropagated • Reads in toggle rate settings: SAIF, VCD/FSDB, user settings
PowerView	• Stores imported instance power file • Processes power and toggle scaling • Computes power based on toggles – propagated or unpropagated from SwitchingActivityView
SimulationView	• Sets up for the analysis, static and dynamic
PackageSimulationView	• Stores package models

Why Views?

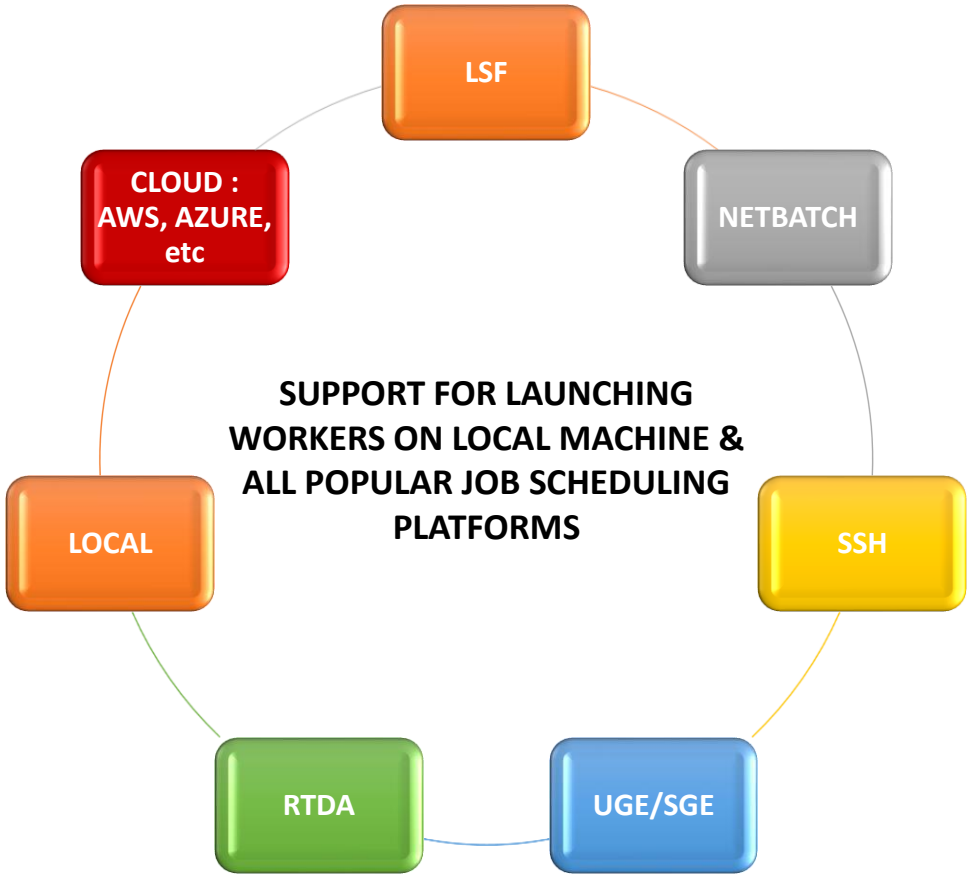
- Views help organize the vast variety of data and their access APIs into separate containers/classes
  - Enables much faster access to specific data associated with the type of view instead of having to query an entire database
- Well documented, view dependent APIs enable easy scripting
  - No need to learn internal format details of various files under hidden folders to get information
- Views provide the ability for multi-scenarios and unique IR/EM flow customization
- Views provide jump-starts (reuse of existing data), checkpoints and resilience
  - Enable fast, low effort, incremental experiments leveraging existing intermediate data
- You **do not have** to learn any Views or their python APIs to use this tool
  - Most standard usage through gui clicks

Setting up RedHawk-SC Environment

- Tool Installation/Binary:
  - RedHawk-SC build can be downloaded to any location and untarred. The typical directory structure is:
    - `<path_to_untar>/seascape_release/latest/linux_x86_64_rhel7/bin/redhawk_sc`
- License setup:
  - License can be specified through the ENV variable **`$LM_LICENSE_FILE`**
  - RedHawk-SC can be invoked directly using the installation path
  - There is no need to define any environment variables other than the license file
- Training Testcase Download:
  - The training testcase bundle is named **`RedHawk_SC_Galaxy2_Training.tgz`**
  - Registered customers can login to the Ansys customer portal to download the testcase (older version)
  - Alternatively, contact Ansys AE to send via SecureTransfer (latest version)
  - Untar the bundle , **`$cd Galaxy2/QuickStart_Training`**, and then look at the README file in the bundle

Launching Workers

- Definition of worker:
  - SC is a distributed platform running on a set of CPU cores
  - CPU cores used by RH-SC are called workers
  - SC can launch hundreds of thousands of workers to process jobs efficiently
  - 1 worker == 1 CPU core with certain amount of memory
- Where are workers located?
  - Workers can be on the same machine or different machines in a network
- How are workers launched?
  - SC supports all popular job scheduling platforms like LSF, UGE, RTDA etc.
  - SC can also run-on cloud-based services like AWS, Azure etc.



More On Workers (Launcher)

- Launcher commands syntax
  - **create_local_launcher**(name, num_workers_per_launch=None, …)

```
ll = gp.create_local_launcher('local')
```

  - **create_ssh_launcher**(name, submit_command, num_workers_per_launch=None, …)

```
ll = gp.create_ssh_launcher('sjo','sjozt11*3 sjozt7*3')
```

  - **create_grid_launcher**(name, submit_command, num_workers_per_launch=None, …)
    - A generic launcher supporting any load balancing software. A few examples:

LSF:

```
lll = gp.create_grid_launcher('lsf_launcher', 'bsub -R linux6 -R "mem>32" -n 1')
ll2 = gp.create_grid_launcher('SJO', '/tools/bin/bsub -q sjo_all -R "(os == CENT && ostype >= CENT7.3)" -R
    "rusage[mem=128000]" -R "span[hosts=1]" -n 4 -o lsf.log')
```

SGE:

```
ll3 = gp.create_grid_launcher('qsub','qsub -V -notify -b y -cwd -j y -q all_hosts -l mfree=32G')
```

SLURM:

```
ll4 = gp.create_grid_launcher('slurm_launcher', '/usr/bin/sbatch --export=All -p sjoall -c 1 --mem=32G -o slurm.log')
```

Netbatch:

```
ll5 = create_grid_launcher('nb', 'nbjob run --target zsc10_normal --qslot /appls/mtl/be/rv --class "SLES12&&64G"')
```

RTDA:

```
ll6 = create_grid_launcher('nc1', 'nc run -r redhat7 CORES/1 RAM/32000', num_workers_per_launch=5)
```

More On Workers (Launching)

- Launching a specific number of workers
  - After creating a launcher, launch a specified number of workers

```
ll.launch(32)
```

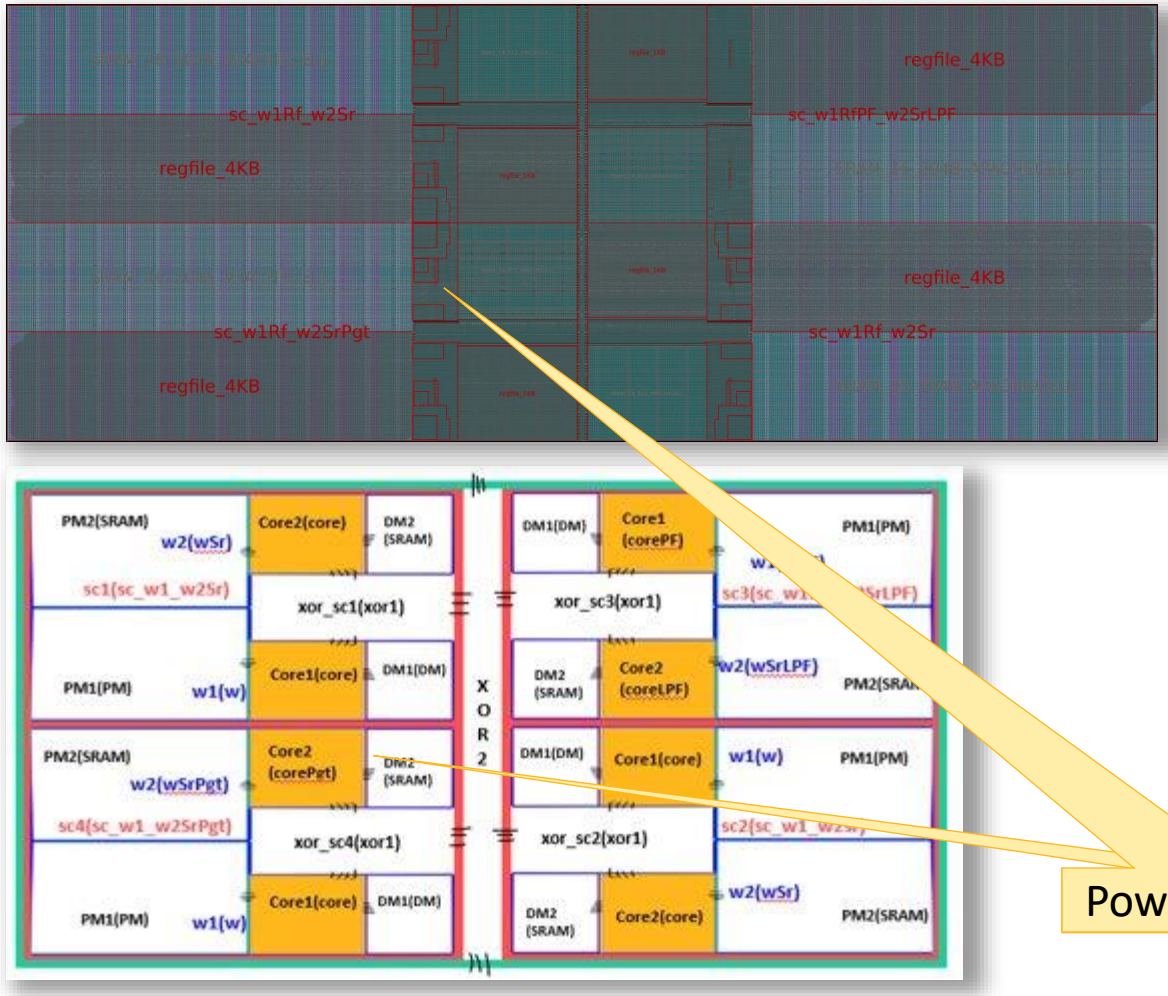
- Automatic Launcher
  - After creating a launcher, use "register_default_launcher" function for SeaScape to automatically launch as many workers as needed
  - This is the most common method
  - Gives SeaScape the flexibility to run in its most efficient manner

```
register_default_launcher(launcher=ll, min_num_workers=5, max_num_workers=50)
```

  - Increase minimum number if you want to push design through faster
  - Set maximum workers in case you don't have enough available workers and need to balance
    - Example:
      - Seascape wants ideally 800 workers
      - You only have 650 workers available
      - ➢ Set max_num_workers to 400 to balance evenly
      - **Extremely important for AnalysisView which will need to double up jobs on workers**

- **Total amount of memory is always the same for all runs**
- **When doubling up the jobs for AV transient runs, must request double the memory, as 2 AV parts will be running on each worker simultaneously**

Training Testcase (Galaxy2) Overview

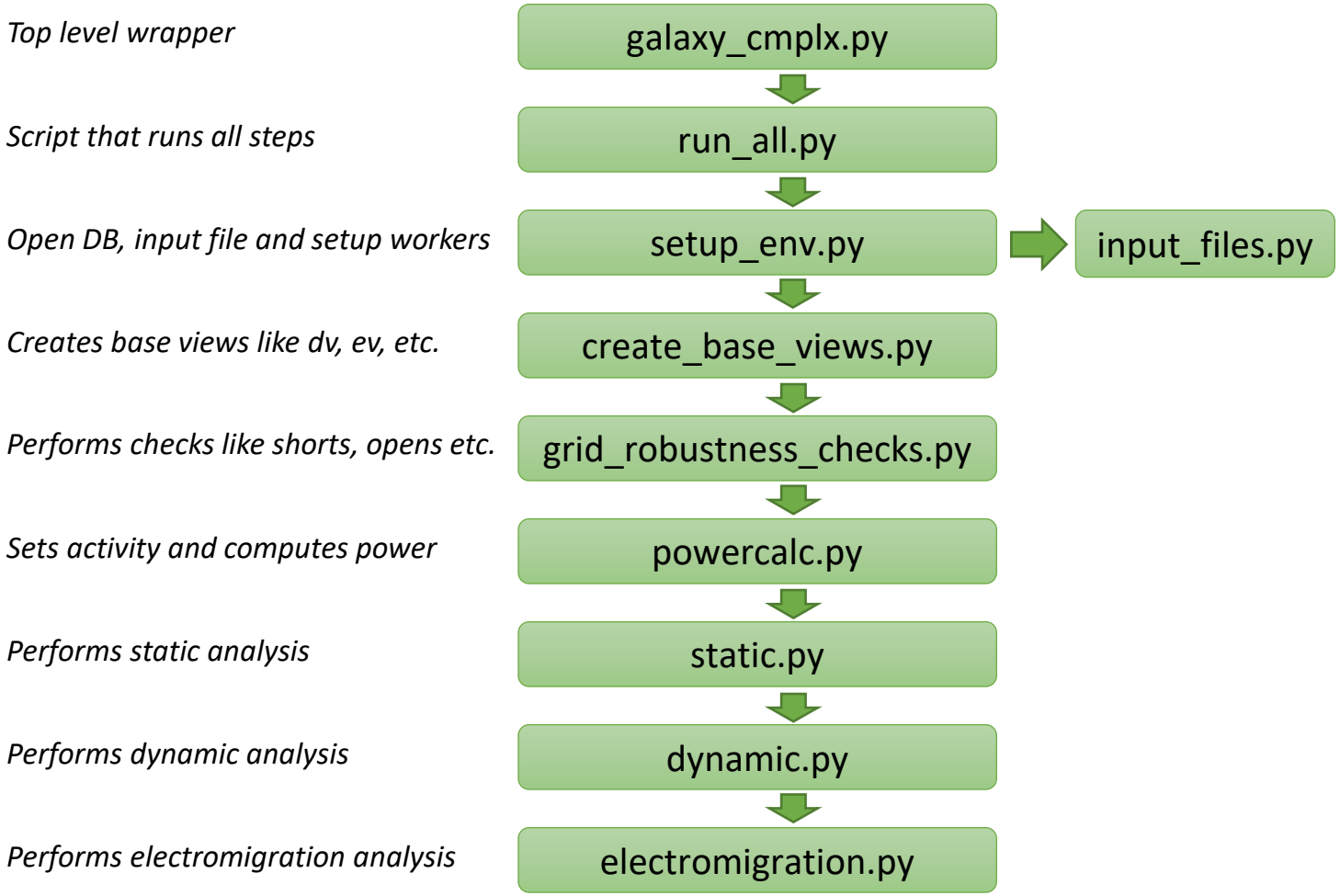


Power Gated Blocks

Galaxy2 Testcase Details	
Instance count	1.1M
Node count	6.2M
Domains	1 Always-ON Power domain 6 Power-gated Power domains 1 Ground domain
Cores	8 OpenMSP430 Microcontrollers
# Memories	8
Tech Node	7nm
# Layers	9

Galaxy2 testcase is created using Open PDK asap7 (7nm) and contains OpenCore OpenMSP430 (Microcontroller IP)

Getting Familiar with Training Lab Scripts



- Each step can be run individually too
- Inside the scripts directory:
  - run_all.py
  - run_create_base_views.py
  - run_grid_robustness_checks.py
  - run_power_calculation.py
  - run_static_analysis.py
  - run_dynamic_analysis.py
  - run_electromigration.py
- To run:
  - `redhawk_sc scripts/run_…`
- Beware of script dependencies
  - Create base views first
  - Robustness checks, powercalc and dynamic depend on the previous step, creating the base views
    - They do not depend on each other though
  - Static depends on powercalc
  - Electromigration depends on both static and dynamic

Getting Familiar with Training Lab Scripts

File: galaxy.py

```
test='galaxy_cmplx'
include('scripts/run_all.py')
```

Single command file that will invoke all steps

File: scripts/run_all.py

```
include('setup_env.py')
include('create_base_views.py')
include('grid_robustness_checks.py')
include('powercalc.py')
include('static.py')
include('dynamic.py')
include('electromigration.py')
```

Script runs all steps from scratch and completes static, dynamic and EM analysis

File: scripts/run_create_base_views.py

```
include('setup_env.py')
include('create_base_views.py')
```

File: scripts/run_grid_robustness_checks.py

```
include('setup_env.py')
include('grid_robustness_checks.py')
```

File: scripts/run_power_calculation.py

```
include('setup_env.py')
include('powercalc.py')
```

File: scripts/run_static_analysis.py

```
include('setup_env.py')
include('static.py')
```

File: scripts/run_dynamic_analysis.py

```
include('setup_env.py')
include('dynamic.py')
```

File: scripts/run_electromigration.py

```
include('setup_env.py')
include('electromigration.py')
```

Setting up the environment

File: scripts/setup_env.py

```
design_data_path = '../design_data/'


db = gp.open_db('db')

## Auto-load view tags from an existing db
## Needed only for incremental(jump-start) runs)
populate_view_tags()



include('input_files.py')



## Setup launcher.  Training is small so could use local launcher or setup LSF or whatever software you use
ll = create_local_launcher('local')
# ll = create_grid_launcher('slurm_launcher', '/usr/bin/sbatch --export=All -p sjoall -c 1 --mem=11G -o slurm.log')

register_default_launcher(ll, min_num_workers=16, max_num_workers=16)
open_scheduler_window()
```

Set *design_data_path* variable to central design data path area

Set the DB location settings using the open_db command

Auto view-tag loading for incremental (jump-start) runs

Include all design input file pointers

Setup your launcher and register with min/max workers

Setup and input files

File: scripts/setup_env.py

```
design_data_path = '../design_data/'

db = gp.open_db('db')

## Auto-load view tags from an existing db
## Needed only for incremental(jump-start) runs)
populate_view_tags()

include('input_files.py')

## Setup launcher.  Training is small so could use local launcher or setup
LSF or whatever software you use
ll = create_local_launcher('local')
# ll = create_grid_launcher('slurm_launcher', '/usr/bin/sbatch --export=All
-p sjoall -c 1 --mem=11G -o slurm.log')

register_default_launcher(ll, min_num_workers=16, max_num_workers=16)
open_scheduler_window()
```

Setup run and call input files

File: scripts/input_files.py

```
def_files = [
    design_data_path + 'def/omsp_dbg_uart_routed_fills_dbu4000_1x_PG2.def.gz',
    design_data_path + 'def/omsp_dbg_routed_fills_dbu4000_1x.def.gz',
...]
lef_files = [
    design_data_path + 'lef/asap7sc6t_26_SL_1x_210923b.lef.gz',
    design_data_path + 'lef/asap7sc6t_26_R_1x_210923b.lef.gz',
...
```

Specify all the design files in input_files.py

Launching RedHawk-SC - Get started with your labs

- Batch mode execution example:

```
<path_to_rhsc_installation>/bin/redhawk_sc galaxy_cmplx.py
```

- Interactive mode execution example:

```
<path_to_rhsc_installation>/bin/redhawk_sc -i galaxy_cmplx.py
```

  - When RH-SC has finished running the input script, the session stays active until the user enters in the Python shell: `exit()`

- Connecting to live RedHawk-SC run:
  - RedHawk-SC allows querying of data/results from an active session, by remotely attaching to the session
  - Multiple users can attach to the same session from multiple machines for querying/viewing results

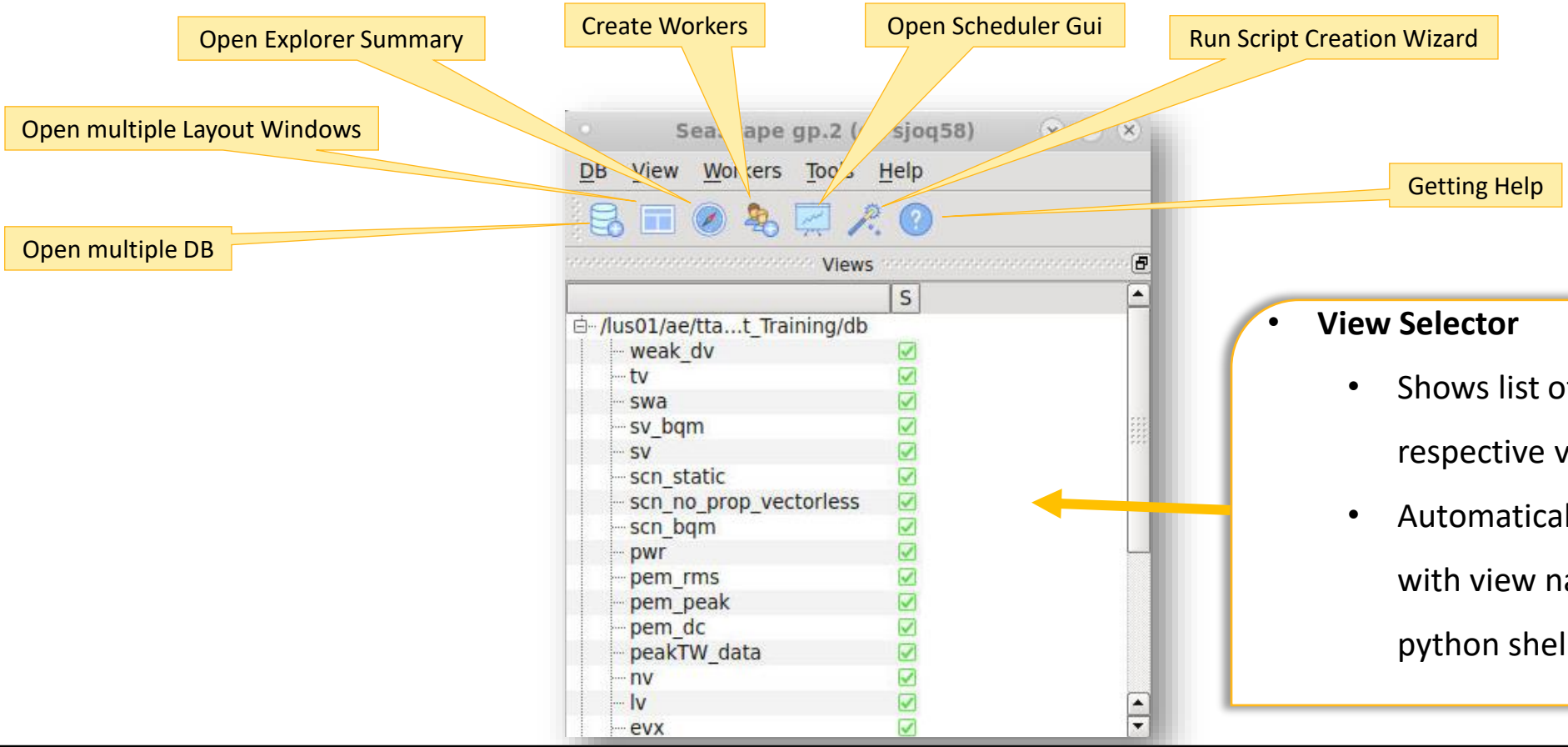
```
<path_to_rhsc_installation>/bin/redhawk_sc -r <gp_dir>
```

- Execution Log files:
  - All RHSC log files reside by default in gp<> folder
    - If the run is fired in the same directory, tool will create gp.1, gp.2 incrementally
    - "latest.gp" link will point to the most recent gp directory
  - Main log file for RedHawk-SC would be <gp_directory>/run.log file
  - Other important log files are run_v.log (more detailed log file for master) and Worker*.log files for individual workers

Launching RedHawk-SC Console

Run command: `<path_to_rhsc_installation>/bin/redhawk_sc <python_script> --console`

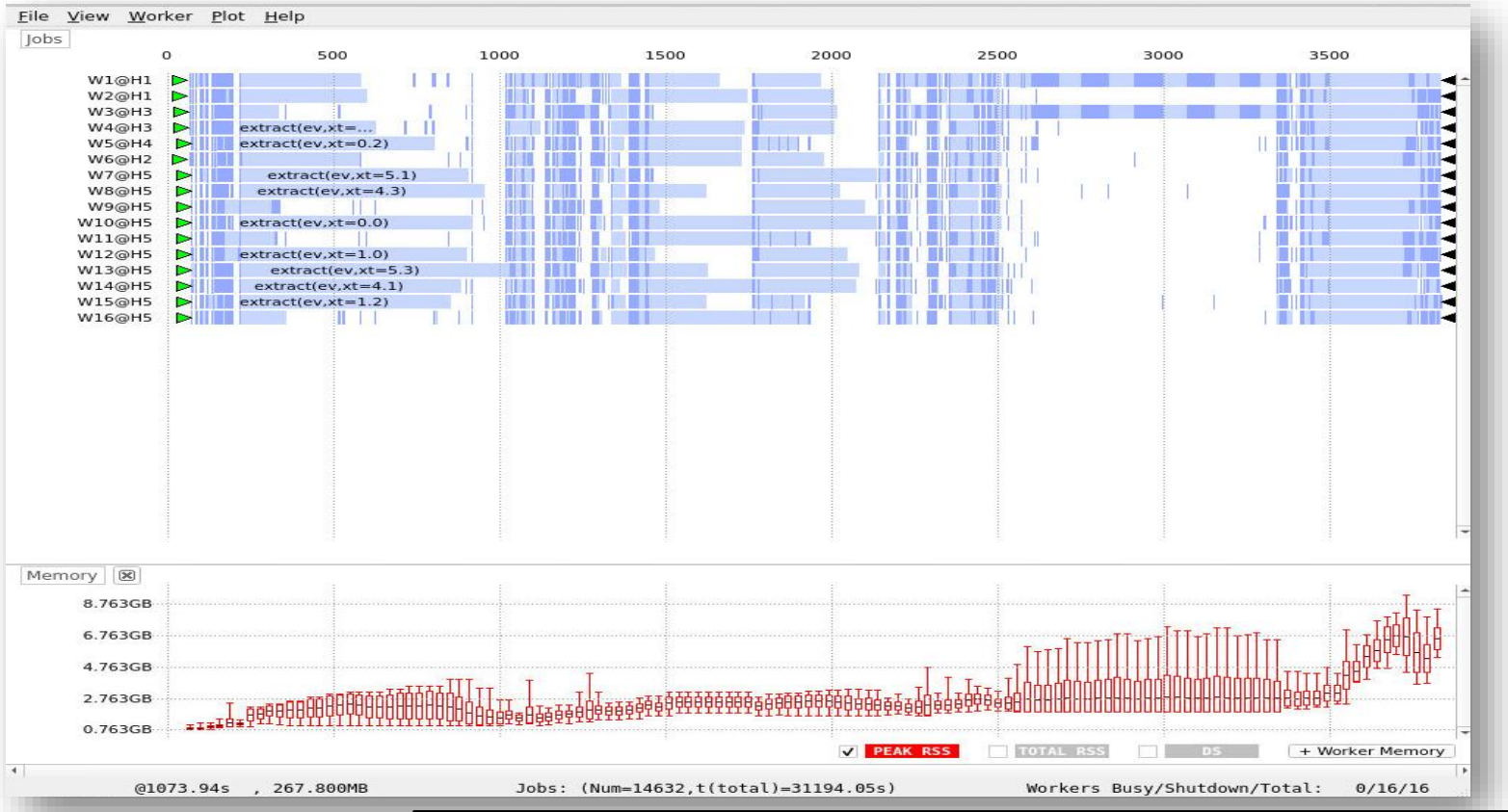
To invoke from RHSC python shell, type: `open_console_window()`



- **View Selector**
  - Shows list of open DBs and their respective views
  - Automatically create variables with view names in the RH-SC python shell

Launching Scheduler GUI

- Provides real time visualization of workers and their jobs
- Displays the jobs executed per worker (Y - axis) on the time axis (X - axis)
- Can be invoked from console or using the command: `open_scheduler_window`
- For an ongoing or completed run, can invoke the scheduler gui using the command: `redhawk_sc -g -l -r <path_to_gp_directory>`

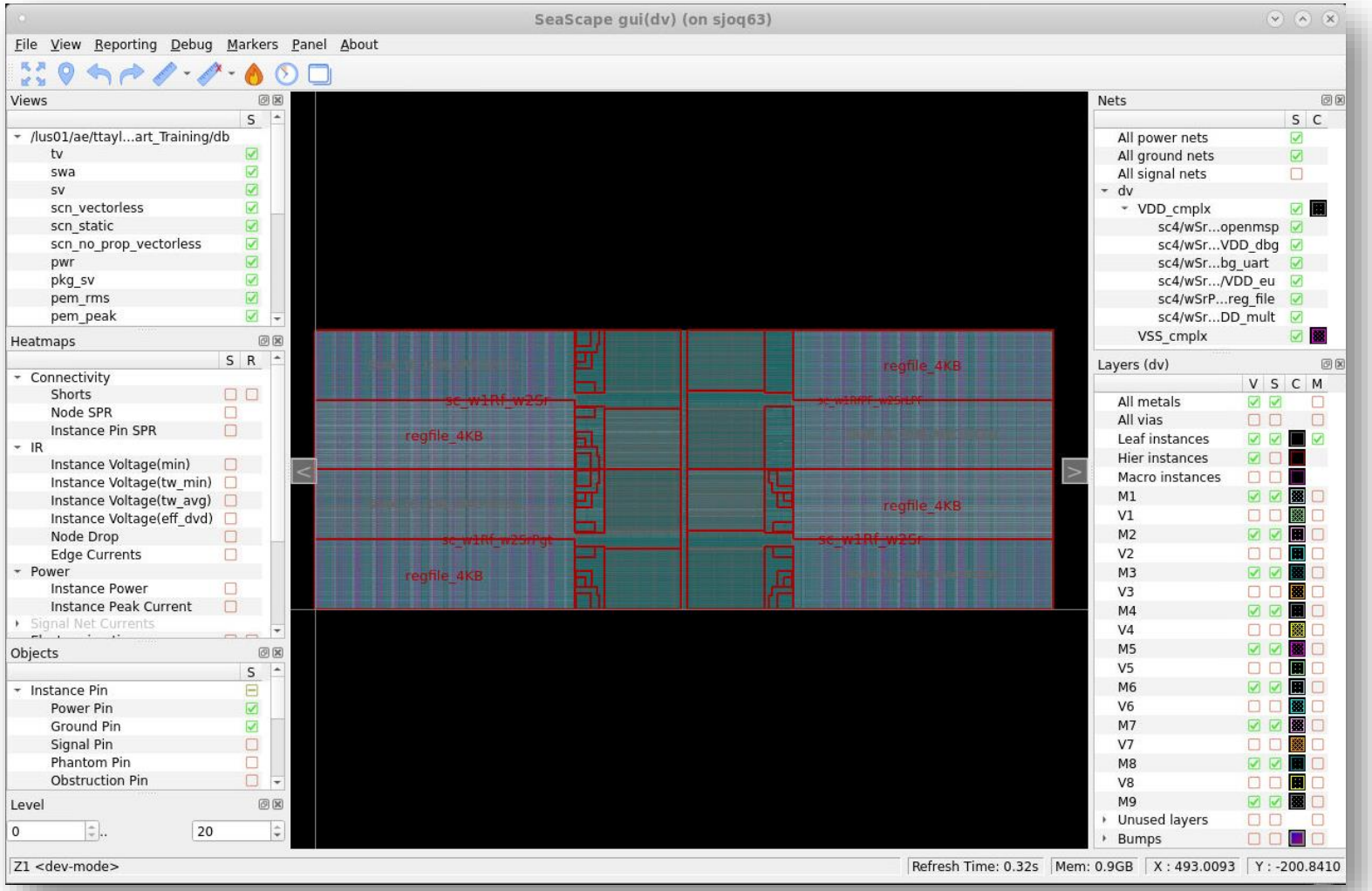


FEATURES OF SCHEDULER GUI

- **Worker info**
- **Host info**
- **Highlight Jobs using**
  - run-time
  - memory
- **Total number of workers, jobs**
  - name
- **Worker Stats Plots**
  - Worker memory
  - Machine load
  - Read ds rate
  - Total ds read
- **Identify network slowness/issues**
  - red boxes
  - using plots
- **Identify performance sticks ***

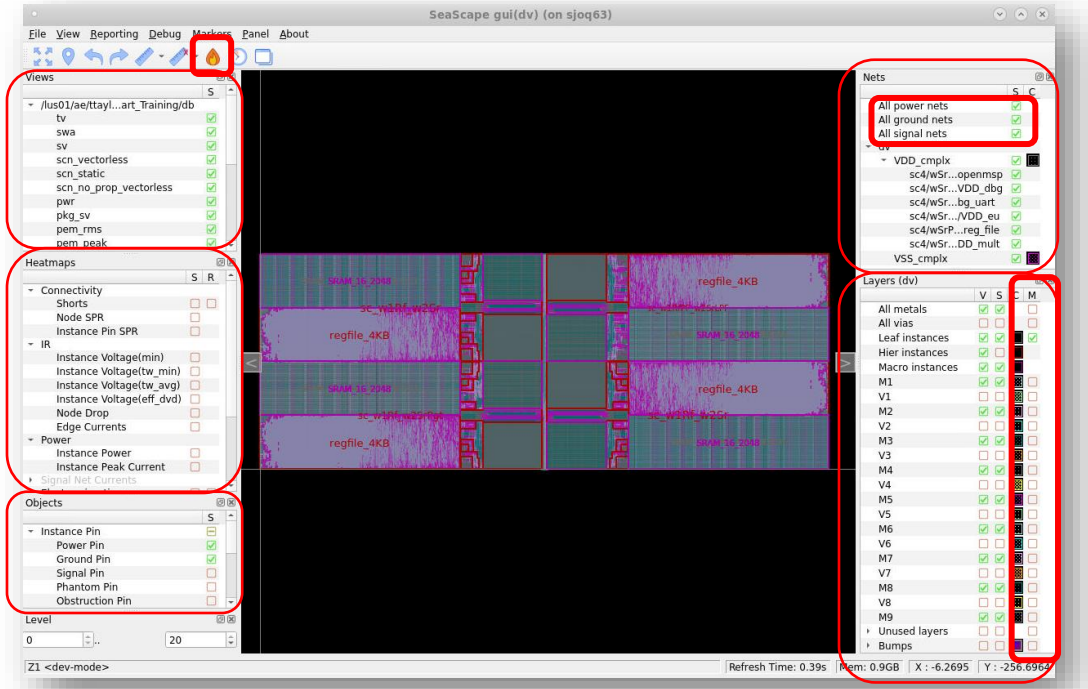
*** Performance Sticks : Long jobs where only a small number of workers are active**

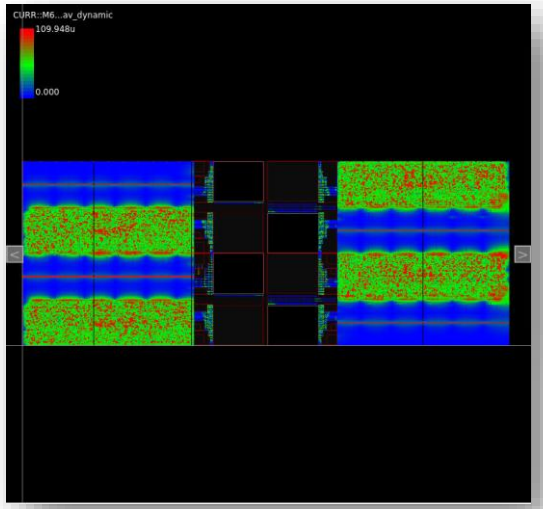
RedHawk-SC™ GUI - Thin Client

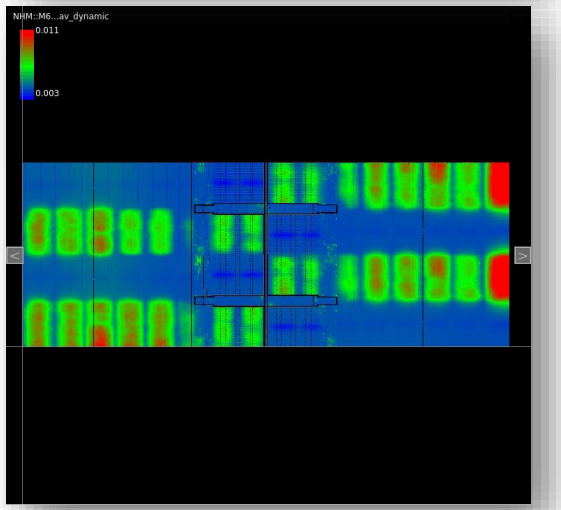


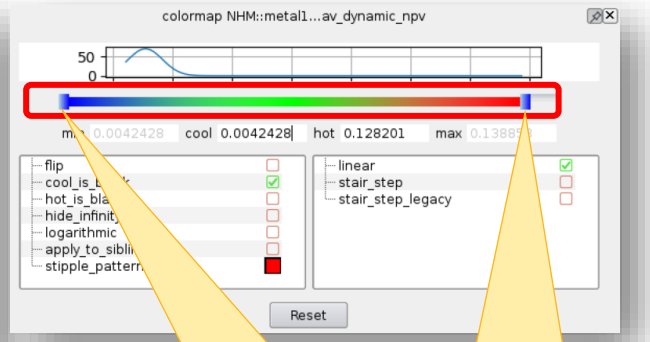
- GUI is a separate process that connects to master
- Multiple GUIs can be opened at same time with locked zoom
- It can run on a different host, or different network
- All data reads are on demand
- Very low memory consumption for viewing largest designs (~ 16 GB)

RedHawk-SC™ GUI - Thin Client



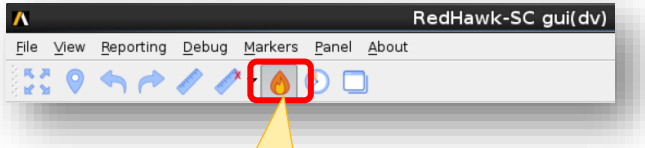






- ✓ **VIEW SELECTOR:** viewing results from multiple views in same GUI
- ✓ **HEATMAP SELECTOR:** enabling multiple heatmaps together
- ✓ **OBJECT SELECTOR:** for easy access to all PnR world objects – pins , vias , OBS etc
- ✓ **NET SELECTOR:** show PG and signal nets in single GUI
- ✓ **LAYER SELECTOR:** quick enabling/disabling layers
- ✓ **MAP SELECTOR:** (M button) for easy map access for instance based/layer based heatmaps

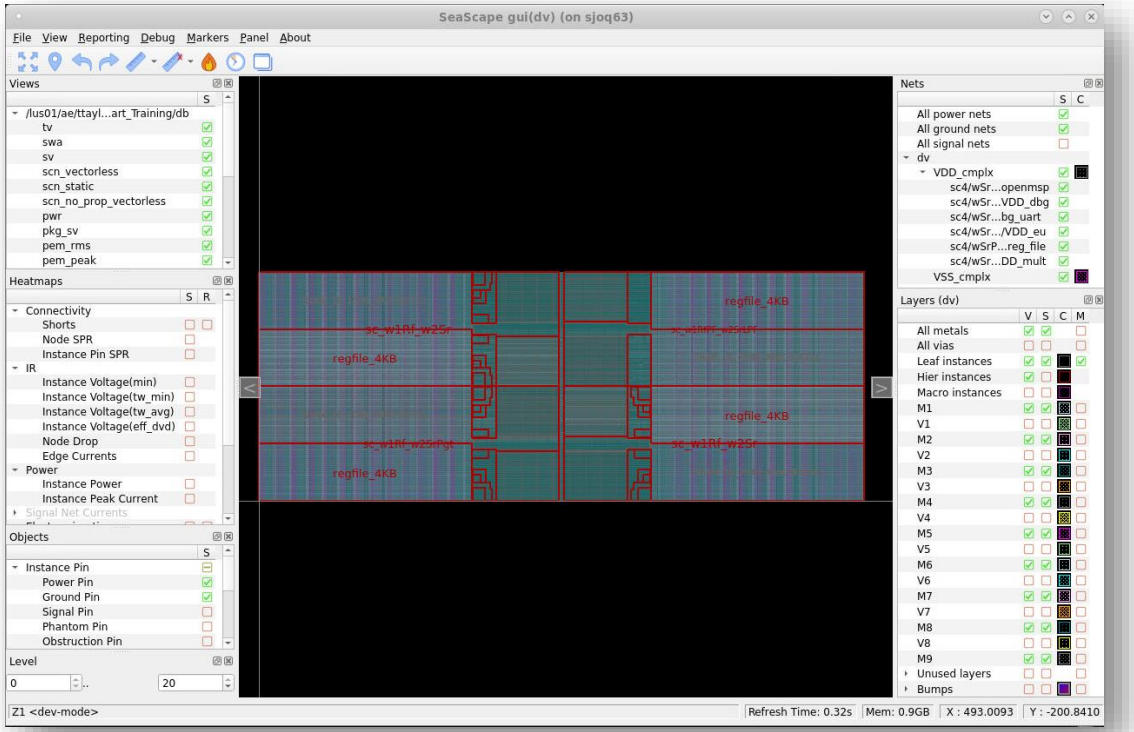
✓ **SLEEK SLIDER:** to instantly change map thresholds



✓ **FLAME BUTTON:** to quickly find hotspots above a threshold

Basic Gui Functions

- Selection – use left mouse button
  - Click to select single object
  - Click repeatedly to cycle through objects at same location
  - Multiple objects, hold down and draw box
    - Objects are selected that are completely within the box
- Zoom in
  - Right mouse button, hold down bottom left to top right (or top left to bottom right)
  - Mouse scroll button – rotate up
- Zoom out
  - Right mouse button, hold down bottom right to top left (or top right to bottom left)
  - Mouse scroll button – rotate down



Reading in input data and performing data Integrity checks

Ansys

Entering the input file pointers

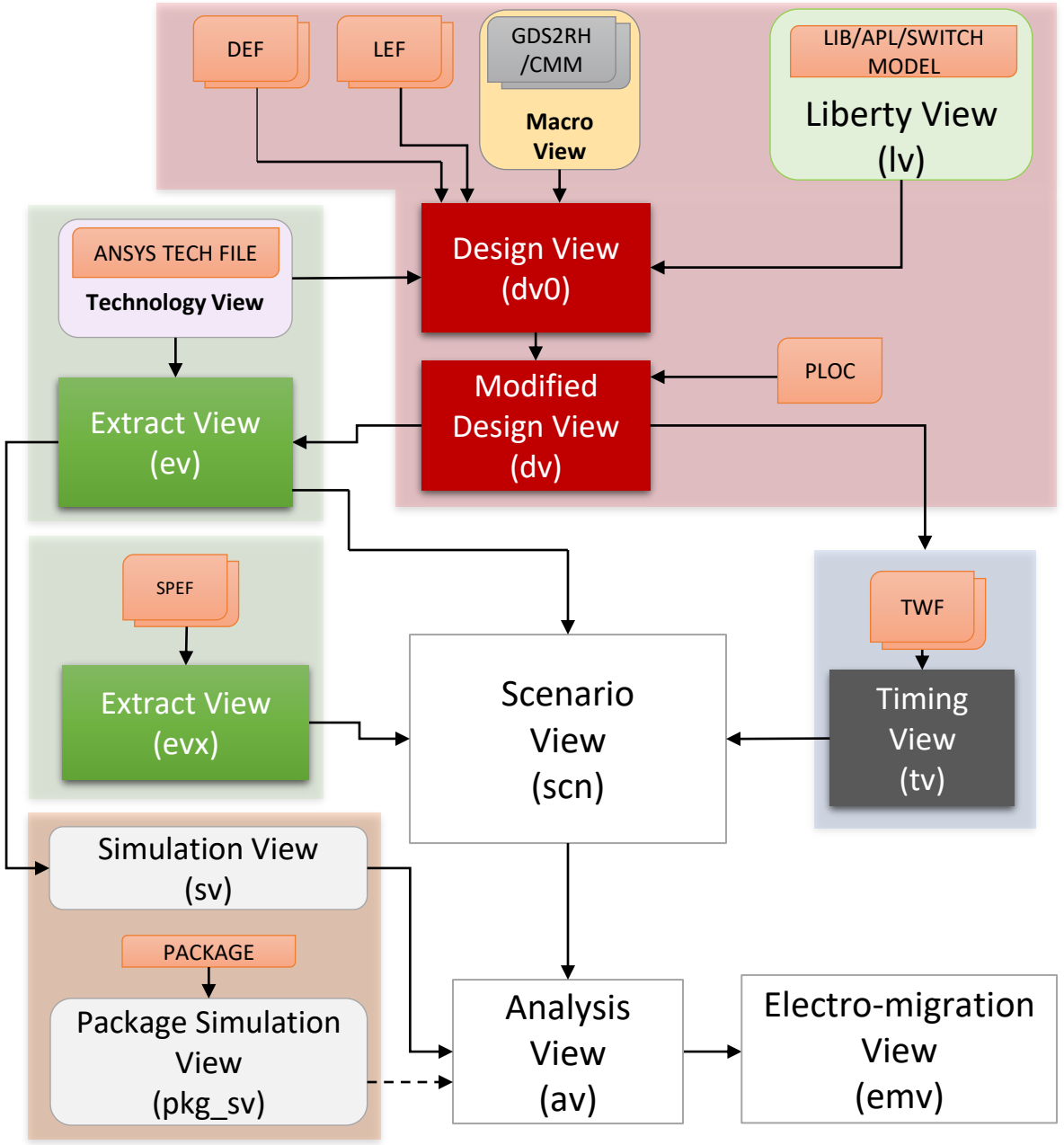
File: scripts/input_files.py

```
def_files = [design_data_path + 'def/omsp_dbg_uart_routed_fills_dbu4000_1x_PG2.def.gz', ...]

lef_files = [design_data_path + 'lef/asap7sc6t_26_SL_1x_210923b.lef.gz', ...]

lib_files_tt = [design_data_path + 'lib_std/asap7sc6t_AO_LVT_TT_ccs_211010.lib.gz', ...]

tech_file = design_data_path + 'tech/asap7_RHTech_wEM.rhtech'

tw_files = [{'instances': [Instance('')], 'file_name': design_data_path + 'timing/galaxy_cmplx_native_31May.timing.gz'}]

switch_files=[{'file_name': design_data_path + 'switch_files/HEADBUFx1_MD_ASAP7_6t_R_aplsw.out','temperature':110}]

ploc_file = design_data_path + 'ploc/galaxy_cmplx_M9_rh_50um_pitch.ploc'
pkg_file = design_data_path + 'pkg/cpa_rh_pkg_wrapper_ASCII.sp'
padfile = design_data_path + 'ploc/galaxy_complex_50um.padfile'

spef_files = [{'file_name': design_data_path + 'spef/galaxy_cmplx_starrc.spef.gz'}, ...]

apl_files = [ {'file_name': design_data_path + 'apl/asap7_TT.current'}, ...]
```

Creating Base Views

File: scripts/create_base_views.py

```
options = get_default_options()

focus_pg_nets = ['VDD_cmplx', 'VSS_cmplx']

extract_temperature = 110.0

lv_tt = db.create_liberty_view(liberty_files=lib_files_tt,
apl_switch_files=switch_files, apl_files=apl_files, tag='lv_tt', options=options)

libs = dict()
libs['tt'] = [lv_tt]

nv = db.create_technology_view(tech_file_name=tech_file, tag='nv', options=options)

model = db.create_macro_view(model_files, tag='macro', options=options)

settings_dv = {'focus_pg_nets': focus_pg_nets}

dv0 = db.create_design_view(tech_view=nv, lib_views=libs, def_files=def_files,
lef_files=lef_files, settings=settings_dv, top_cell_name='galaxy_cmplx', tag='dv0',
macro_views=[model], options=options)

dv = db.create_modified_design_view(dv0,
eco_commands=package.parse_ploc_func(ploc_file), tag='dv', options=options)

settings_ev = {
    'calculate_spr': True,
    'extract_temperature': extract_temperature,
    'detail_extraction_net_type': 'all'}

ev = db.create_extract_view(dv, nv, settings=settings_ev, tag='ev', options=options)

sv = db.create_simulation_view(ev, tag='sv', options=options)

pkg_sv = db.create_package_simulation_view(ev,
package=package.parse_ploc_spice_func(pkg_file, dv=dv, pad_file_name=padfile,
net_based=False), tag='pkg_sv', options=options)

evx = db.create_extract_view_from_files(dv, input_files=spef_files, tag='evx',
options=options)

tv = db.create_timing_view(dv, timing_window_files=tw_files, process_corner='tt',
tag='tv', options=options)
```



Importance Of Performing Data Integrity Checks

LIBRARY LEVEL DATA INTEGRITY CHECKS

APL/CCS Current Check	RedHawk-SC gets the cell P/G current waveforms from APL/CCS current data. APL is generated using a utility from Ansys called APL. In the absence of APL/CCS current data, RedHawk-SC creates a triangular current waveform using the LIB. Analysis using the LIB based current profiles will be normally more pessimistic than APL/CCS based analysis.
APL/CCS Cap Check	RedHawk-SC gets the cell intentional cap information from APL/CCS cap data. APL cap is generated using a tool called APL. APL/CCS Cap data contains Effective Series Cap (ESC) and Effective Series Resistance (ESR) values for each P/G pin of the cell. In the absence of APL/CCS Cap data, RedHawk-SC doesn't use any cell intrinsic cap values in the simulation. Analysis will give pessimistic drop values.
LIB Check	RedHawk-SC uses the LIB for calculating the leakage and internal power for the cell. In the absence of LIB, RedHawk-SC will not include these instances in power calculation. The timing, functionality, pgarc information of the cell will also be unavailable in absence of LIB files.
LEF Check	Instances without LEF cannot be hooked up to power grid, so RedHawk-SC excludes these cells in analysis. LEF coverage issue is very critical.
Macro Check	RedHawk-SC expects gds2rh/gds2db or Totem-CMM data for all hard-macros in the design. These models provides the physical representation of power grid geometries inside the macro. Without them, the macros will get black-boxed and will affect the accuracy of analysis.

Importance Of Performing Data Integrity Checks

DESIGN DATA INTEGRITY CHECKS

DEF Check	DEF data can have multiple issues: Shorts, Unconnected Instances, Unconnected Wires, Unconnected Vias and Missing Vias.
SPEF Check	In static analysis, RedHawk-SC uses SPEF for computing the switching/internal power. In dynamic analysis, RedHawk-SC needs the SPEF for picking the correct current waveform from the APL/CCS Look-up tables. Absence of SPEF leads to incorrect power or current waveform calculation.
STA Check	In static analysis, RedHawk-SC uses Frequency and Slew information from the STA/TWF file. In Dynamic, RedHawk-SC can also use timing window information from STA/TWF file. STA/TWF file is also used to identify the clock instances in the design. In the absence of STA/TWF File, RedHawk-SC uses 0 frequency or user specified global frequency. Can lead to incorrect static and dynamic results.
VCD checks	For VCD based runs – Gate or RTL, we need to make sure that we have full coverage for all nets (Gate VCD) or registers/primary inputs (RTL VCD).

Performing Data Integrity Checks

- Before proceeding with analysis, data integrity issues should be resolved
  - Otherwise, waste valuable time and effort

File: scripts/create_base_views.py

```
di_reports = data_integrity_reports

reports_dir = 'reports/data_integrity/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)

dv_data = di_reports.get_design_view_data_integrity_summary(dv, detailed_reports=True)
di_reports.write_design_view_data_integrity_reports(dv_data, dv, output_directory=reports_dir)

design_utils.report_detailed_shorts(ev,reports_dir+'shorts.rpt')

spef_data = di_reports.get_spef_annotation_summary(evx)
di_reports.write_spef_annotation_reports(spef_data, evx, summary_file=reports_dir +
'spef_annotation_summary.rpt', detail_file=reports_dir + 'spef_unannotated_details.rpt')

tv_data = di_reports.get_timing_view_data_integrity_summary(tv, detailed_reports=True)
di_reports.write_timing_view_data_integrity_reports(tv_data, tv, output_directory=reports_dir)
```

Data Integrity Reports available in "reports/data_integrity" directory

```
dv_apl_cap_check.rpt
dv_apl_current_check.rpt
dv_ccs_cap_check.rpt
dv_ccs_current_check.rpt
dv_ccs_timing_check.rpt
dv_di_check_summary.rpt
dv_lef_check.rpt
dv_lib_check.rpt
dv_lib_nldm.rpt
dv_lib_nlpm.rpt
dv_macro_model_check.rpt
inst_no_slew.rpt
inst_no_tw.rpt
spef_annotation_summary.rpt
spef_unannotated_details.rpt
tv_di_summary.rpt
```

Analyzing Data Integrity Reports

DATA INTEGRITY SUMMARY REPORTS

- **dv_di_check_summary.rpt**
- **tv_di_summary.rpt**
- **spef_annotation_summary.rpt**

```
# Total design view data integrity check summary:

# Check type         Cell Category       Covered        Total         Covered%
#-------------------------------------------------------------------------------
apl_cap_check        Combinational           282          284         99.2958%
apl_cap_check        Sequential               10           10        100.0000%
apl_cap_check        Macro                     2            2        100.0000%
apl_cap_check        Power_Gate                0            2          0.0000%
apl_cap_check        Decap                     2            2        100.0000%
apl_cap_check        Filler                    0            3          0.0000%
apl_cap_check        All                     296          303         97.6898%

# Check type         Cell Category       Covered        Total         Covered%
#-------------------------------------------------------------------------------
apl_current_check    Combinational           282          284         99.2958%
apl_current_check    Sequential                9           10         90.0000%
apl_current_check    Macro                     2            2        100.0000%
apl_current_check    Power_Gate                0            2          0.0000%
apl_current_check    Decap                     0            2          0.0000%
apl_current_check    Filler                    0            3          0.0000%
apl_current_check    All                     293          303         96.6997%

# Check type         Cell Category       Covered        Total         Covered%
#-------------------------------------------------------------------------------
apl_pwcap_check      Combinational             0          284          0.0000%
```

```
#timing view data integrity summary

transition time integrity check summary:
    Combinational        covered/total: 758661/759477 percent: 99.8926%
    Macro                covered/total: 8/8 percent: 100.0000%
    Sequential           covered/total: 169856/169856 percent: 100.0000%
    total cell           covered/total: 928525/929341 percent: 99.9122%

timing window integrity check summary:
    Combinational        covered/total: 759204/759477 percent: 99.9641%
    Macro                covered/total: 8/8 percent: 100.0000%
    Sequential           covered/total: 169856/169856 percent: 100.0000%
    total cell           covered/total: 929068/929341 percent: 99.9706%


Instances have no translation time detailed file: /lus01/ae/ttaylor/trainin
g/Galaxy2/QuickStart_Training/reports/data_integrity/inst_no_slew.rpt
Instances have no timing window detailed file: /lus01/ae/ttaylor/training/G
alaxy2/QuickStart_Training/reports/data_integrity/inst_no_tw.rpt
```

```
### SPEF annotation summary report
### Only report the connected nets

annotated_nets/total_nets: 933019/933019, covered percent: 100.0000%

Block "galaxy_cmplx" SPEF annotated summary:
input_port           annotated/total: 94/94, covered percent: 100.0000%
flip_flop            annotated/total: 169864/169864, covered percent: 100.0000%
latch                annotated/total: 272/272, covered percent: 100.0000%
combinational        annotated/total: 762883/762883, covered percent: 100.0000%
memory               annotated/total: 0/0, covered percent: 0.0000%
pad                  annotated/total: 0/0, covered percent: 0.0000%
sequential_clock     annotated/total: 0/0, covered percent: 0.0000%
combinational_clock  annotated/total: 0/0, covered percent: 0.0000%
macro                annotated/total: 0/0, covered percent: 0.0000%
unknown              annotated/total: 0/0, covered percent: 0.0000%
total                annotated/total: 933019/933019, covered percent: 100.0000%
```

DATA INTEGRITY DETAILED REPORTS

- **dv_apl_cap_check.rpt**
- **dv_apl_current_check.rpt**
- **dv_ccs_cap_check.rpt**
- **dv_ccs_current_check.rpt**
- **dv_ccs_timing_check.rpt**
- **dv_lef_check.rpt**
- **dv_lib_check.rpt**
- **dv_lib_nldm.rpt**
- **dv_lib_nlpm.rpt**
- **dv_macro_model_check.rpt**
- **inst_no_slew.rpt**
- **inst_no_tw.rpt**
- **spef_unannotated_details.rpt**

```
# tag can be any of
# "MTT" --- missing transition time
# "MTW" --- missing timing window
# "NCP" --- no clock pin
# "NIP" --- no input pin
# "NOP" --- no output pin

#Instance                                                Failing_pins  tag
sc4/_069_                                                ['Y']    MTW
sc4/_092_                                                ['Y']    MTW
sc4/HFSBUF_4_139                                         ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/HFSINV_59_1114              ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/HFSINV_89_1301    ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/HFSINV_47_1302    ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/HFSINV_4_747    ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/HFSINV_4_748    ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/HFSBUF_2_746    ['Y']    MTW
sc4/wRf_inst/openMSP430_inst/clock_module_0/_211_    ['Y']    MTW
```

File: inst_no_tw.rpt report file

```
# tag can be any of
# "MTT" --- missing transition time
# "MTW" --- missing timing window
# "NCP" --- no clock pin
# "NIP" --- no input pin
# "NOP" --- no output pin

#Instance                                               Failing_pins  tag
sc4/_041_                                               ['B', 'A']    MTT
sc4/_042_                                               ['B', 'A']    MTT
sc4/_043_                                               ['B', 'A']    MTT
sc4/_045_                                               ['B', 'A']    MTT
sc4/_046_                                               ['B', 'A']    MTT
sc4/_068_                                               ['B', 'A']    MTT
sc4/_069_                                               ['B', 'A']    MTT
sc4/_070_                                               ['B']    MTT
sc4/_072_                                               ['B', 'A']    MTT
sc4/wRf_inst/openMSP430_inst/optlc_1830                 []    NIP
```

File: inst_no_slew.rpt

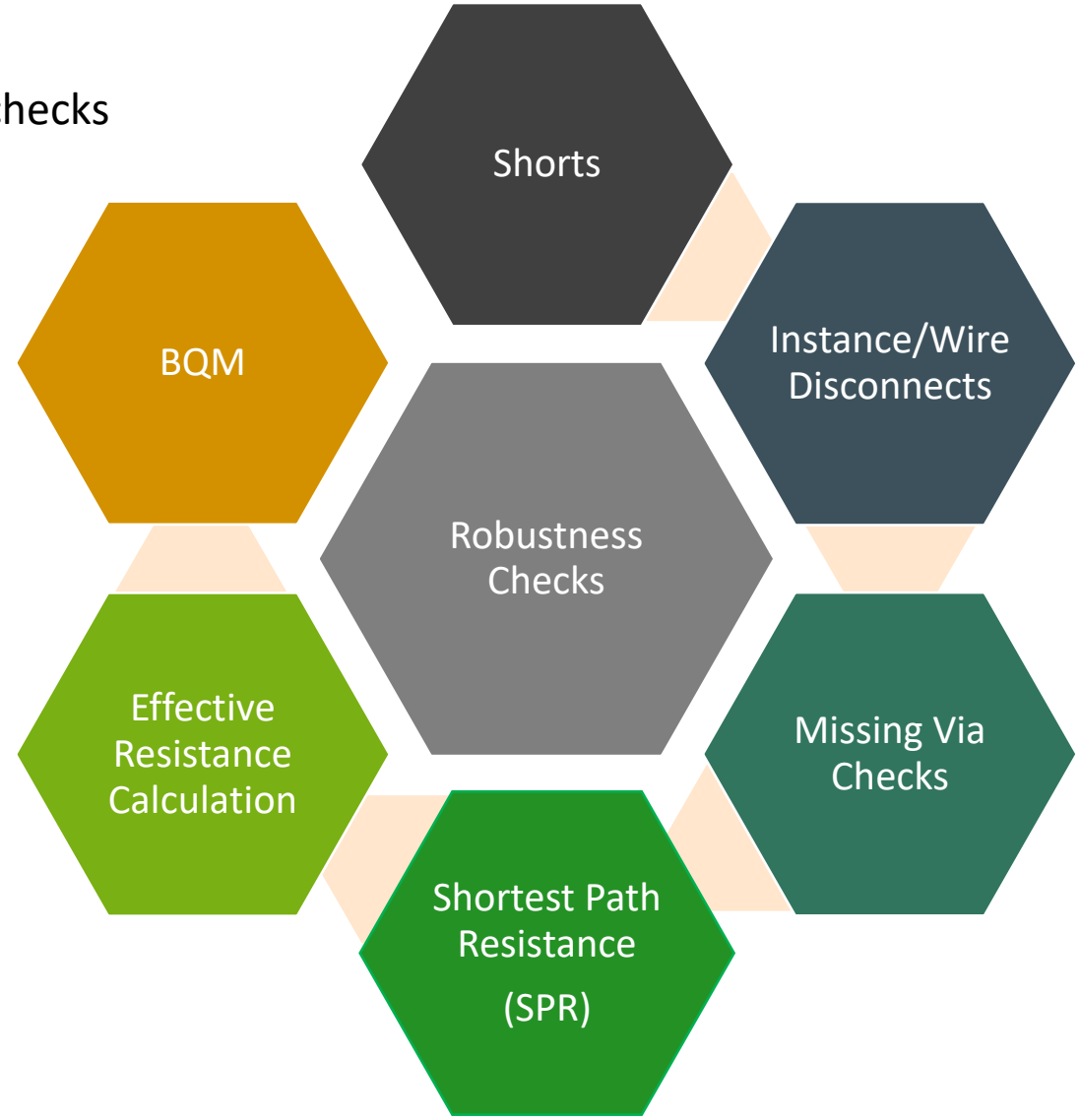
```
ccs_timing_check     covered/all summary: 290/298, covered percent: 97.3154%
#ccs_timing_check uncovered cells:
#cell_type           cell_name
Sequential           MB2DFFASHQNx1_ASAP7_6t_R
Sequential           MB4DFFASHQNx1_ASAP7_6t_R
Macro                SRAM_16_2048
Macro                SRAM_16_512
Combinational        TIEHIxp5_ASAP7_6t_L
Combinational        TIELOxp5_ASAP7_6t_L
Power_Gate           HEADBUFx1_MD_T2B_ASAP7_6t_R
Power_Gate           HEADBUFx1_MD_B2T_ASAP7_6t_R
```

File: dv_ccs_timing_check.rpt

Grid Robustness and Construction Integrity Checks

Different Types Of Grid Robustness Checks

- RedHawk-SC offers the following types of grid robustness checks
  - ✓ Shorts check
  - ✓ Instance and Wire Disconnect check
  - ✓ Missing Via check
  - ✓ Shortest Path Resistance (SPR)
  - ✓ Effective Resistance Calculation*
  - ✓ Build Quality Metric (BQM), PeakTW



* Effective resistance checks will not be part of condensed training.
For more details, refer to “Examining PG grid weakness in SC” application note

Looking at Shorts in GUI

```
>>> ev.get_shorts_summary()
{'total_num_shorts': 58, 'total_num_shorts_between_pg_nets': 6,
'pg_net_shorts': {(Net('VDD_cmplx'), Net('VSS_cmplx'), Layer('M1')): 6}}
```

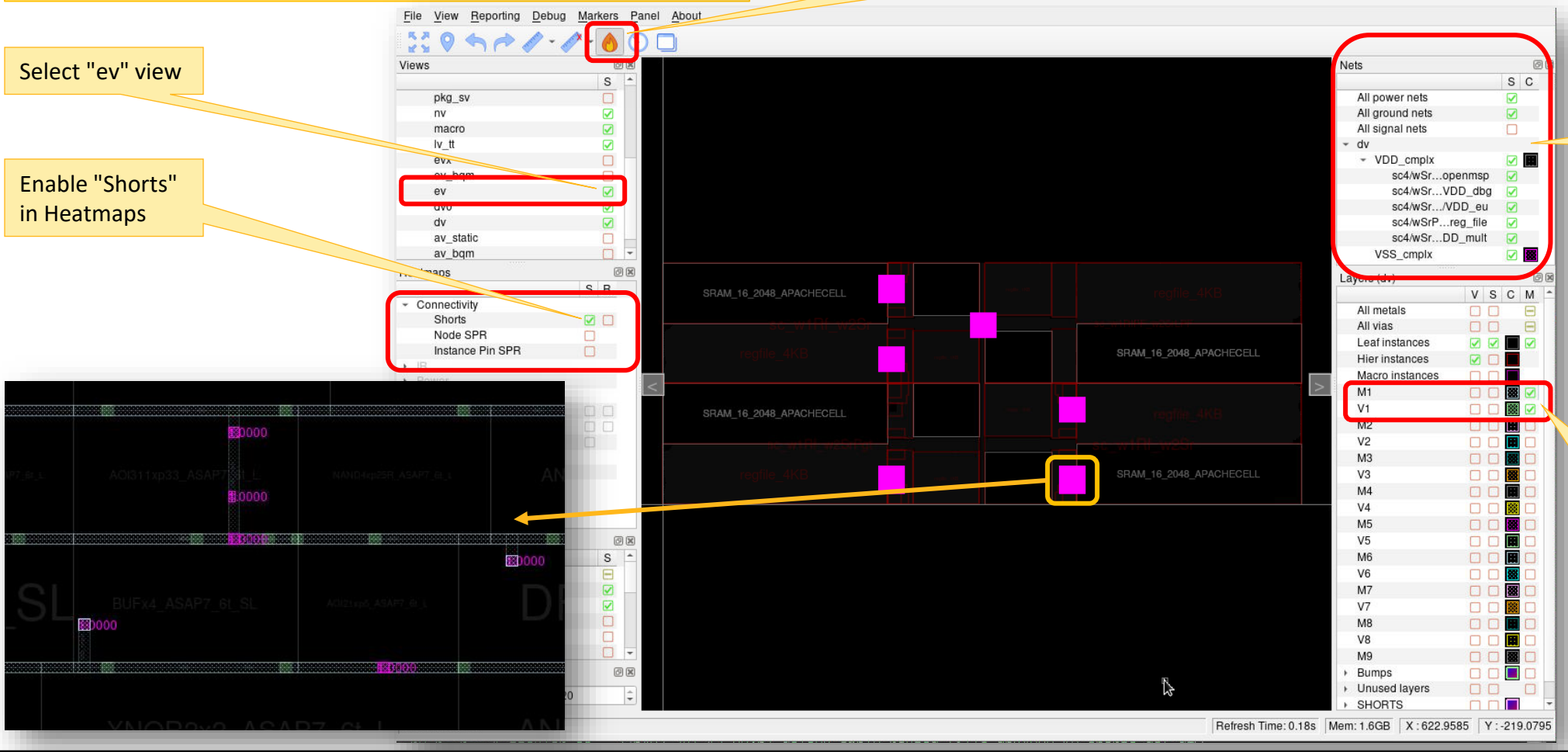
Enable the flame button if the shorts are not clearly visible from top level

Select "ev" view

Enable "Shorts" in Heatmaps

Can choose nets

Choose layer on which shorts need to be displayed



Missing Via Checks

Command: `mvc = db.perform_missing_via_check(design_view=dv, tag='mvc', options=options)`

Text Report Generation: `heatmap_utils.report_missing_vias(missing_via_view=mvc, file_name='./mvc.rpt', redhawk_style=True)`

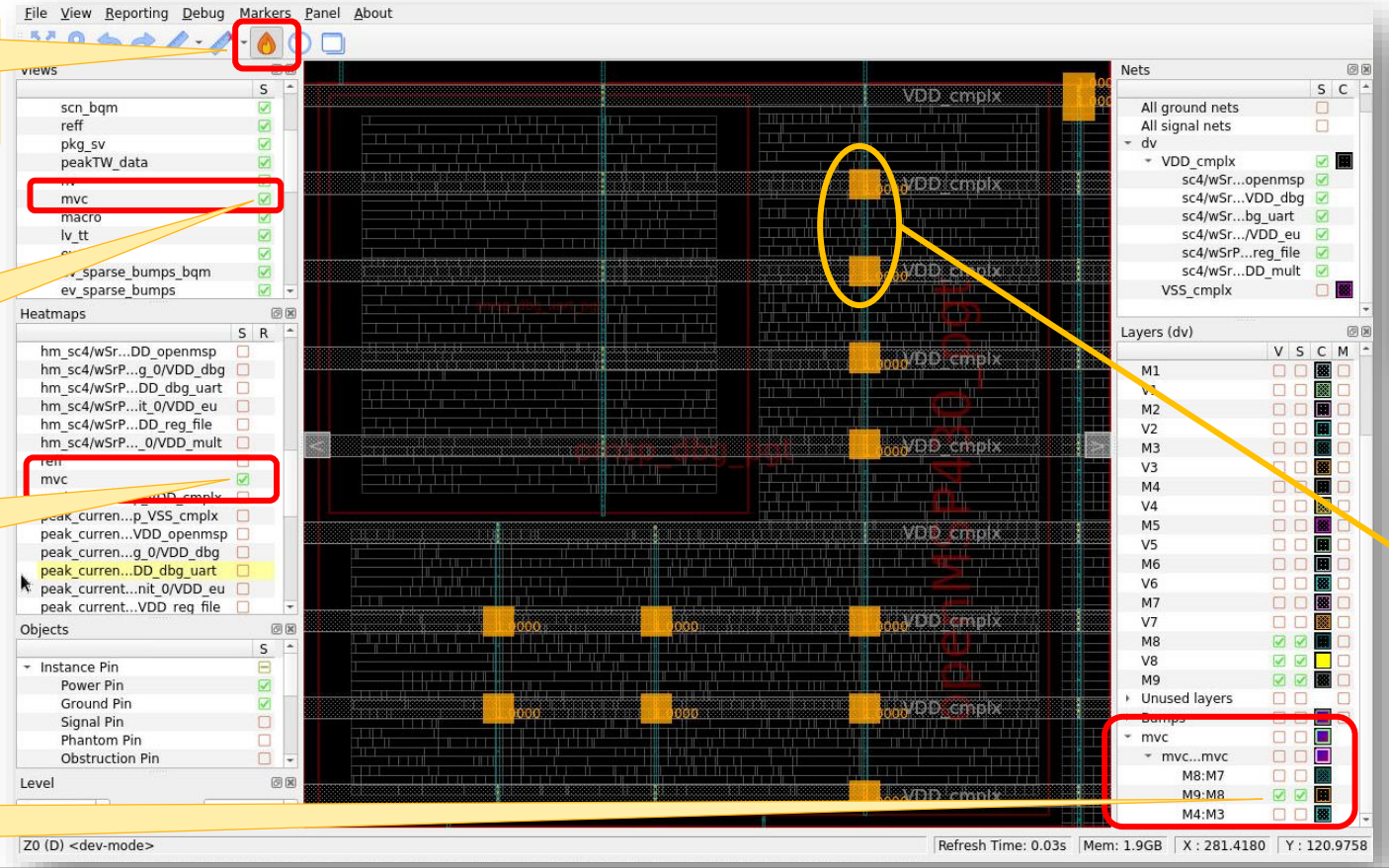
For more options, see command: `help(SeaScapeDB.perform_missing_via_check)`

Enable the flame as needed

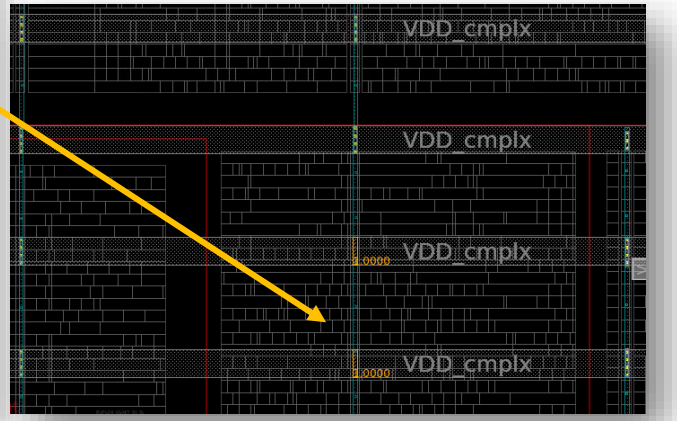
Select "mvc" view

Select "mvc" heatmap

Choose which mvc layers




```
# Net Layer1 Layer2 X,Y
VSS_cmplx M9 M8 303.129,15.75
VSS_cmplx M4 M3 303.129,15.876
VSS_cmplx M4 M3 303.129,19.332
VSS_cmplx M9 M8 301.929,92.75
VSS_cmplx M9 M8 303.129,92.75
VSS_cmplx M9 M8 303.129,134.75
VSS_cmplx M4 M3 301.929,92.72
VSS_cmplx M4 M3 303.129,92.72
VSS_cmplx M4 M3 303.129,131.168
VSS_cmplx M4 M3 303.129,134.624
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu M4 M3 283.557,132.862
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu M4 M3 285.957,132.862
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu M4 M3 288.356,132.862
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu M4 M3 289.557,132.862
VDD_cmplx M9 M8 284.757,111.75
VDD_cmplx M9 M8 284.757,113.75
```



Looking At Grid Weakness Reports

File: scripts/grid_robustness_checks.py

```
reports_dir = 'reports/grid_robustness_checks/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)

data_integrity_reports.write_short_report(ev, output_file=reports_dir + 'shorted_nodes.rpt')
data_integrity_reports.write_disconnected_pg_pins_report(ev, max_lines=None, output_file=reports_dir + 'disconnected_instance_pins.rpt')
data_integrity_reports.write_disconnected_node_report(ev, output_file=reports_dir + 'disconnected_nodes.rpt')

spr_data = emir_reports.get_instance_pin_spr_report(ev)
emir_reports.write_instance_pin_spr_report(spr_data, ev, output_file=reports_dir + 'instance_pin_spr.rpt')
```

```
disconnected_instance_pins.rpt
disconnected_nodes.rpt
instance_pin_spr.rpt
shorted_nodes.rpt
```

Grid Robustness Reports available in "reports/grid_robustness_checks" directory

Looking At Grid Weakness Reports

File: disconnected_instance_pins.rpt

# loc_x	loc_y	cell_name	pin	net	logical_disconnect	physical_disconnect	instance
303.939	148.09	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_7
303.939	148.306	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_6
303.939	148.522	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_5
303.939	148.738	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_4
303.939	148.954	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_3
303.939	149.17	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_2
303.939	149.386	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_1
303.939	136.426	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_61
303.939	136.642	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_60
303.939	136.858	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_59
303.939	137.074	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_58
303.939	137.29	HEADBUFx1_MD_B2T_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_57
303.939	137.506	HEADBUFx1_MD_T2B_ASAP7_6t_R	VDD	VDD_cmplx	0	1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/power_gates_execution_unit/sw_1_56

File: disconnected_nodes.rpt

#Total disconnected nodes = 1236 . Line limit used = 100000

# X	Y	Layer	Net
305.4515	0.9745	M5	VSS_cmplx
281.4515	0.6530	M4	VSS_cmplx
282.6515	0.6530	M4	VSS_cmplx
283.8515	0.6530	M4	VSS_cmplx
285.0515	0.6530	M4	VSS_cmplx
286.2515	0.6530	M4	VSS_cmplx
287.4515	0.6530	M4	VSS_cmplx
288.6515	0.6530	M4	VSS_cmplx

File: shorted_nodes.rpt

#Net_1	Net_2	Layer	Location
VDD_cmplx	VSS_cmplx	M1	284.8385, 31.4700
VDD_cmplx	VSS_cmplx	M1	284.8385, 181.4700
VDD_cmplx	VSS_cmplx	M1	284.8385, 269.0300
VDD_cmplx	VSS_cmplx	M1	508.9998, 31.4700
VDD_cmplx	VSS_cmplx	M1	508.9998, 119.0300
VDD_cmplx	VSS_cmplx	M1	397.3300, 222.9460

File: instance_pin_spr.rpt

#pin	spr_value	cell_name	x	y	layer	instance_name
VDD	1005.097	NAND3xp33_ASAP7_6t_L	304.641	142.798	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/ctmTdsLR_2_2872
VDD	1001.281	OAI21xp5b_ASAP7_6t_SL	304.641	145.39	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/_741_
VDD	992.267	BUFx3_ASAP7_6t_L	304.614	140.638	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/HFSBUF_70_859
VDD	991.793	BUFx6q_ASAP7_6t_SL	304.506	141.07	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/ZBUF_182_inst_3651
VDD	989.358	INVx1_ASAP7_6t_SL	281.7335	119.03	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/dbg_uart_0/ctmTdsLR_2_1887
VDD	988.328	INVx2_ASAP7_6t_L	304.668	140.206	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/HFSINV_4_750
VDD	988.328	INVx2_ASAP7_6t_L	304.668	140.206	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/HFSINV_4_749
VDD	987.628	INVx1_ASAP7_6t_SL	304.695	141.502	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/ctmTdsLR_1_2871
VDD	983.655	OR2x3R_ASAP7_6t_L	304.452	141.07	M1	sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/_432_

Shortest Path Tracing (SPR)

- SPR is the electrically shortest path from one of the voltage sources (PLOCs) to an instance pin
- SPR reported resistance is different from effective/real resistance in the design
- Shortest Path Resistance (SPR) data is not generated by default in the ExtractView
- User must specify additional setting called `'calculate_spr': True` in create_extract_view argument settings:

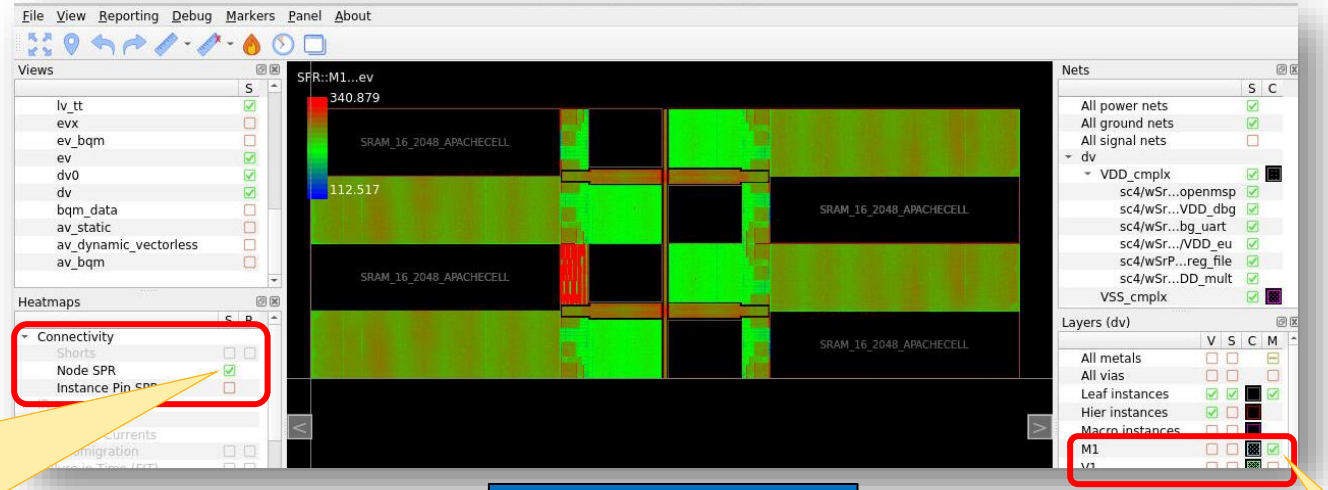
```
ev = db.create_extract_view(dv, nv, settings={'calculate_spr': True,
'extract_temperature' : 25.0}, tag='ev', options=options)
```

- If ExtractView is not generated with `'calculate_spr': True` user can generate SPR data using the command:

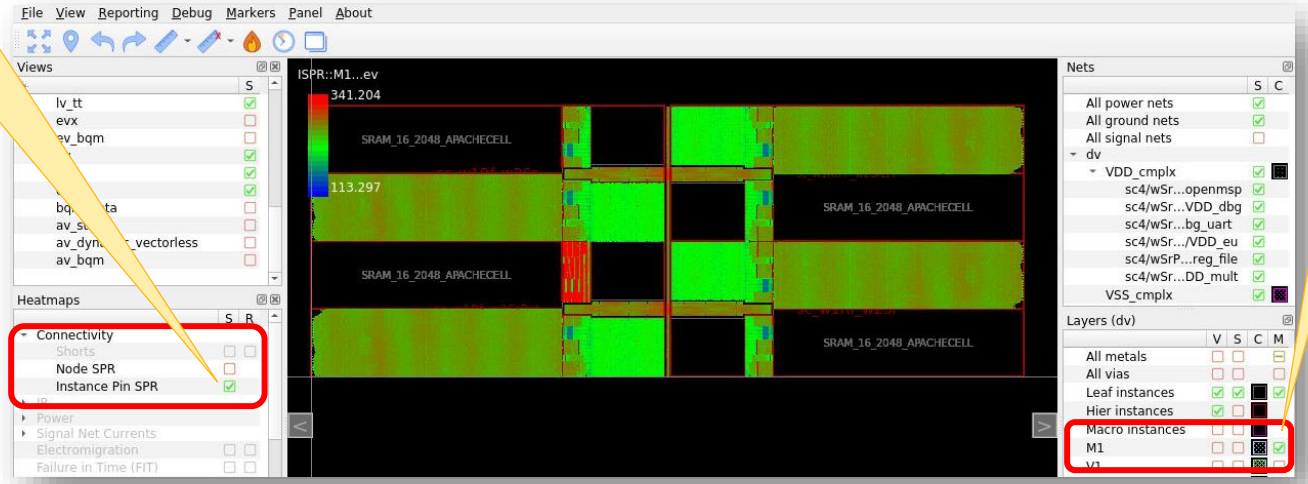
```
spr = db.perform_shortest_resistance_check(ev, tag='spr', options=options)
```

Looking at SPR in GUI

If SPR results are available in ExtractView, then we can choose Node SPR and Instance SPR maps

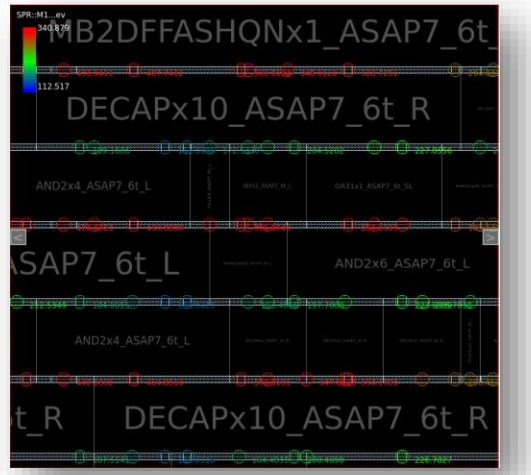


Node SPR Map (M1)

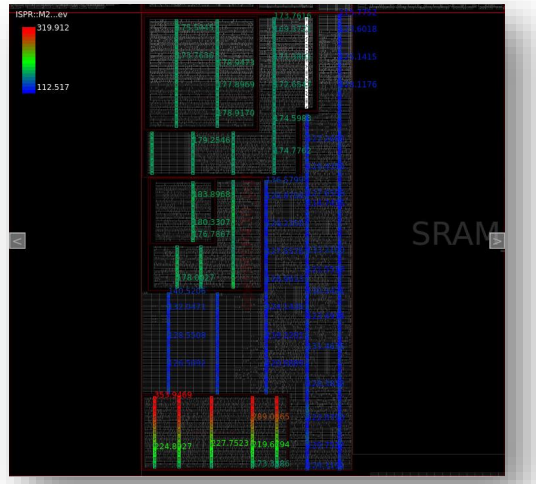


Instance Pin SPR Map (M1)

Choose which layers



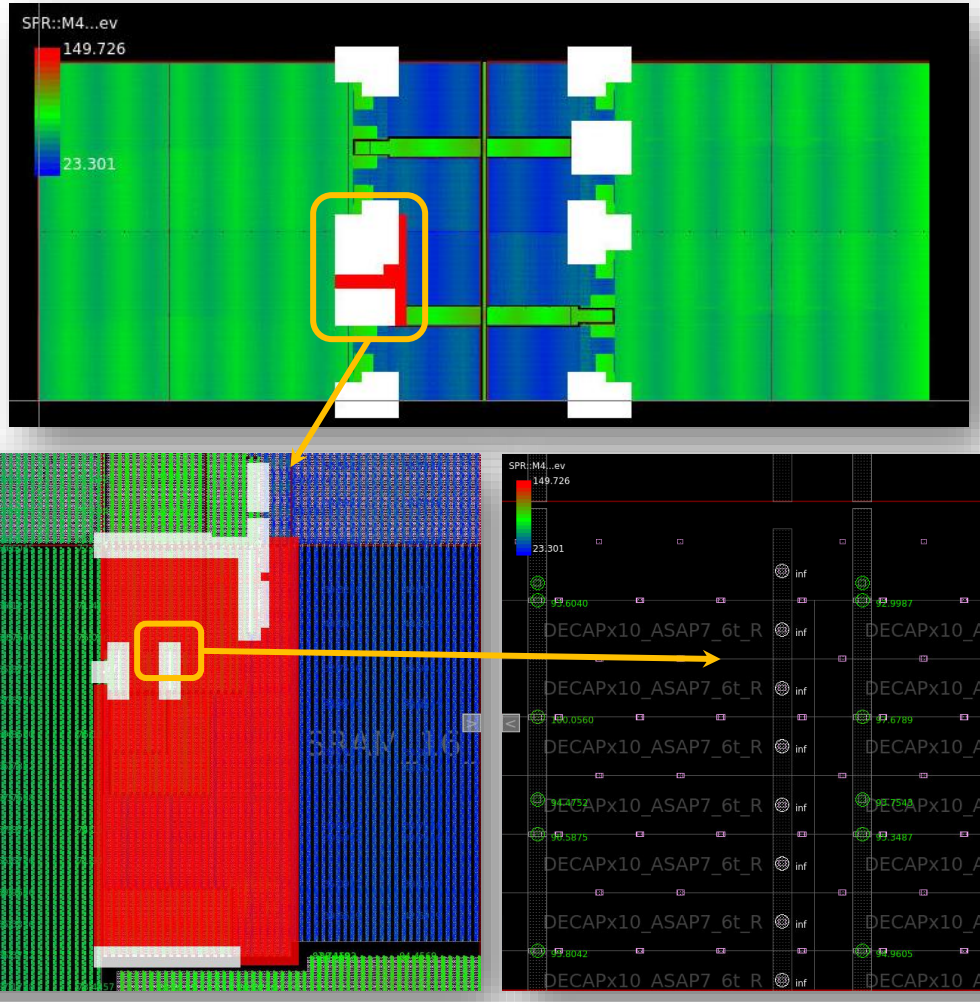
Node SPR Map (M1)

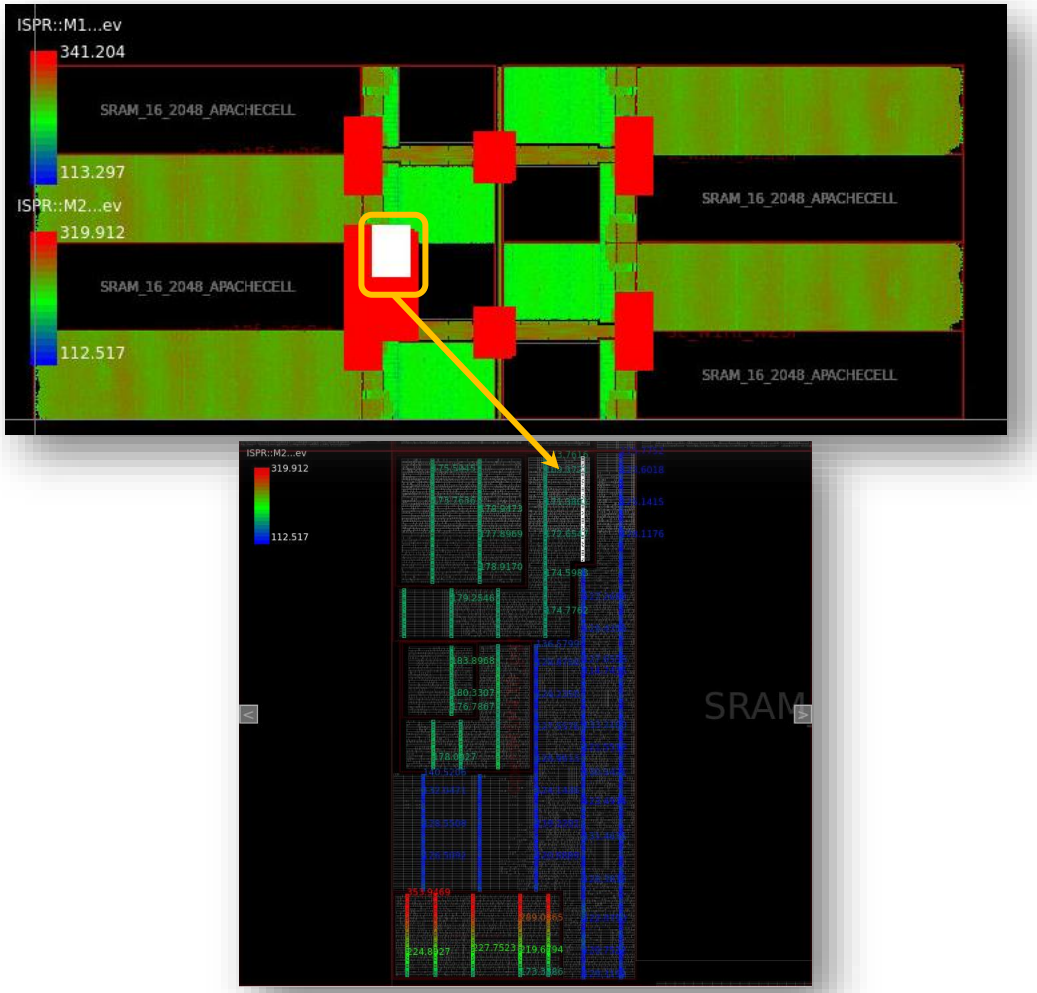


Instance Pin SPR Map (M2)

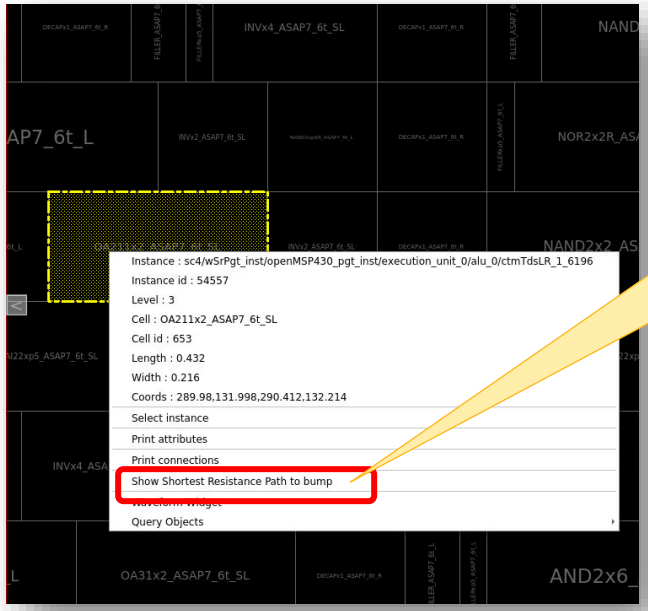
Looking at Unconnects through SPR heatmap

- Node SPR heatmap (left) and Instance Pin SPR heatmap (right) show disconnected nodes and instance pins in white color

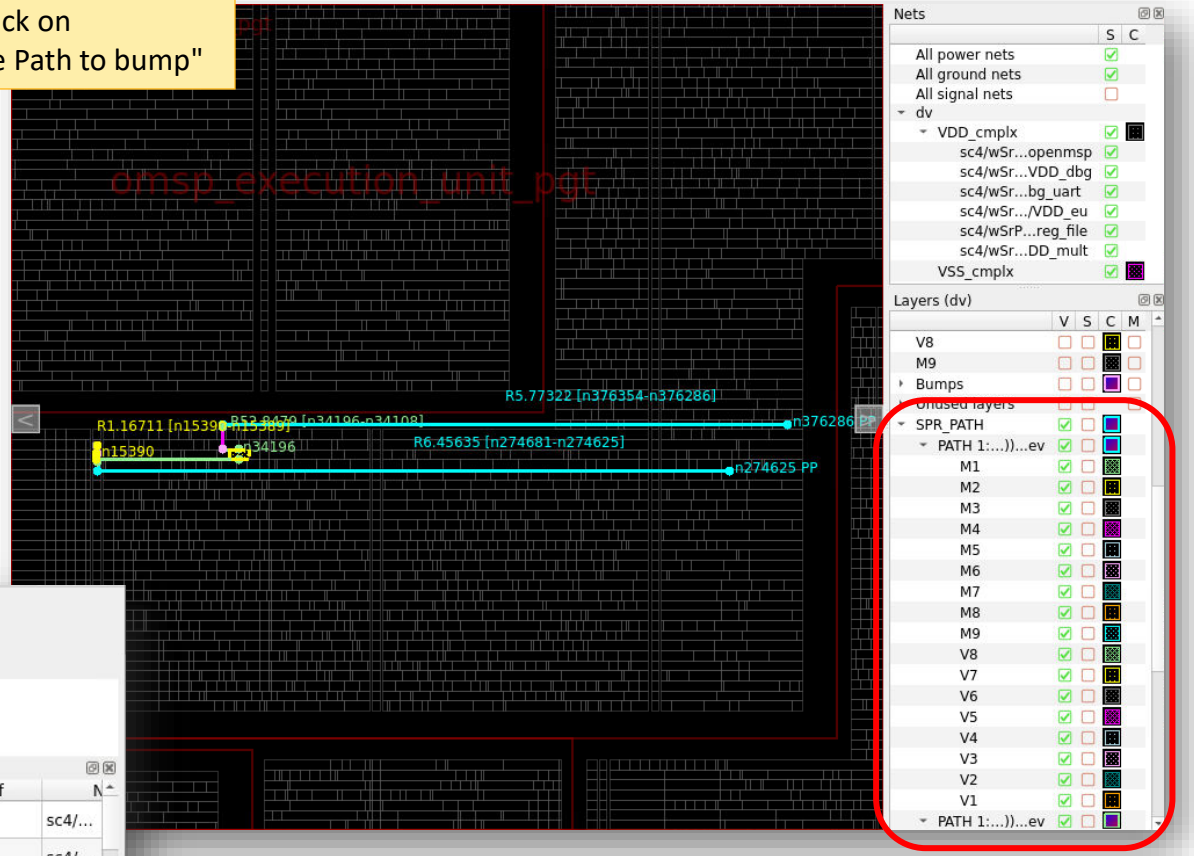


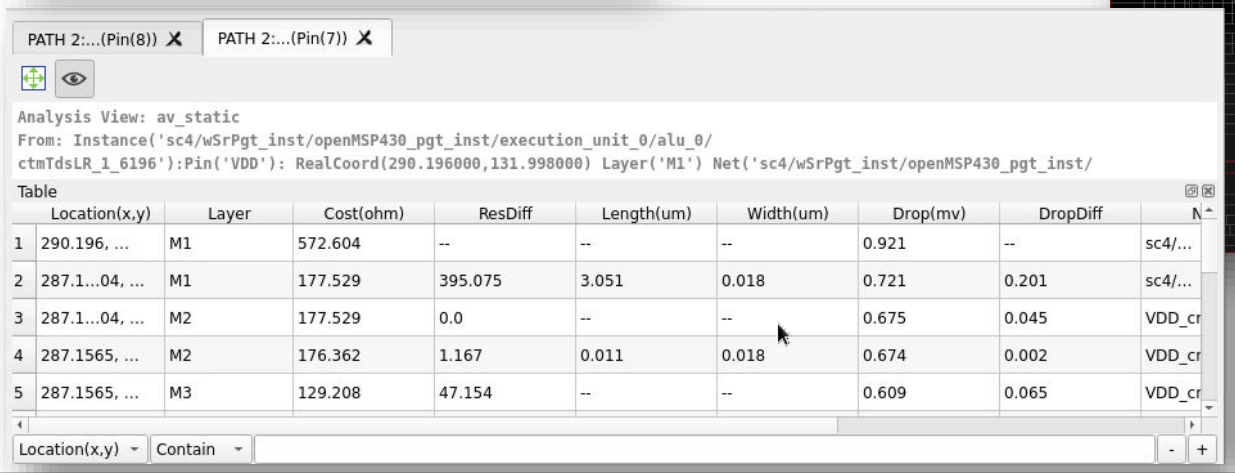


SPR Tracing For Single Instance



Left click to select instance, right click to bring up menu, and left click on "Show Shortest Resistance Path to bump"



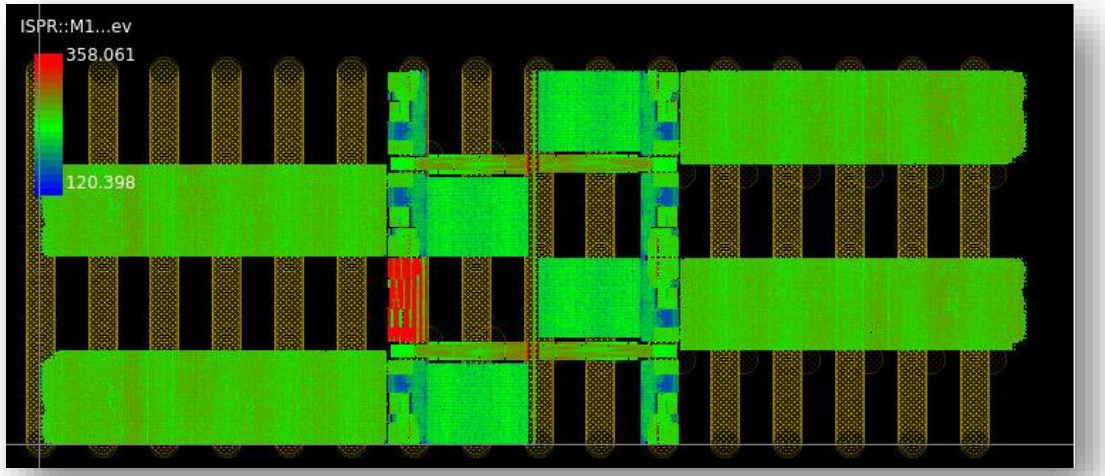


Limitations of SPR in catching real design weaknesses

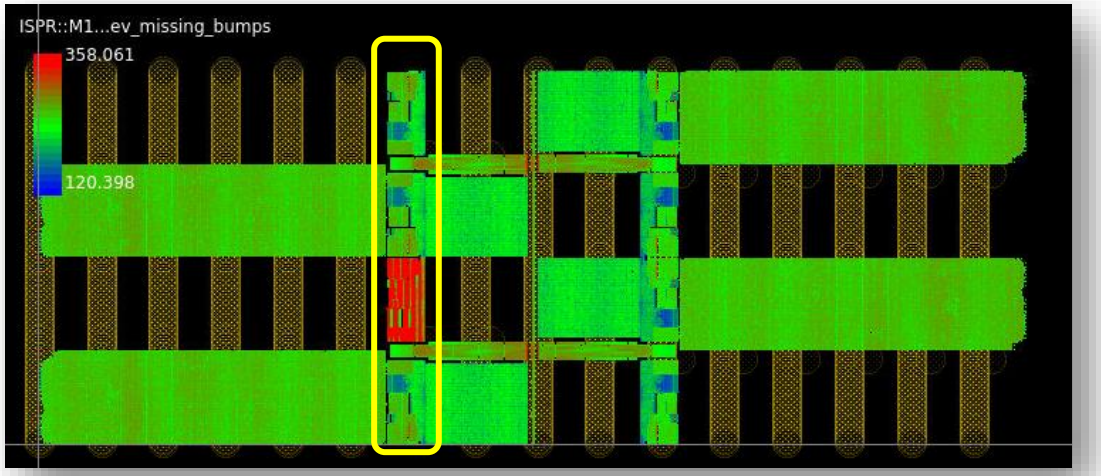
File: scripts/grid_robustness_checks.py

```
dv_missing_bumps = db.create_modified_design_view(dv0, eco_commands=package.parse_ploc_func(ploc_missing_file), tag='dv_missing_bumps', options=options)
```

- We removed some plocs and created a new design view called 'dv_missing_bumps'
- Instance Pin SPR map on metal1 of 'dv' and 'dv_missing_bumps' are almost identical
- **SPR fails to catch issue of missing bumps**



Complete bumps (dv)
Instance Pin SPR Map
(Power Domains)



Less bumps (dv_missing_bumps)
Instance Pin SPR Map
(Power Domains)

Effective Resistance Calculation

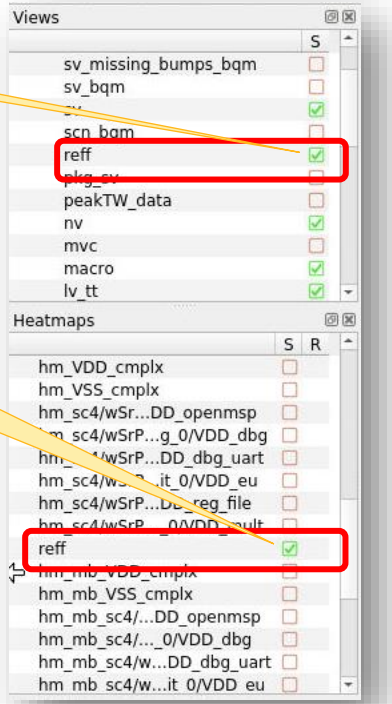
Command: `reff = db.create_effective_resistance_view(sv, tag='reff', options=options)`

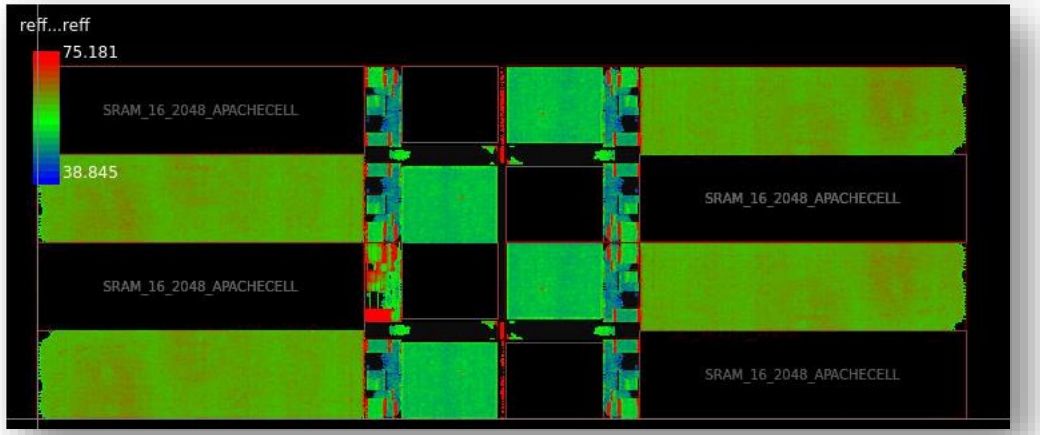
Text Report Generation: `emir_reports.report_effective_resistance(reff, file_name='effective_resistance.rpt')`

For more options, see command: `help(SeaScapeDB.create_effective_resistance_view)`

Select "reff" view

Enable "reff" heatmap





Effective Resistance Heatmap

File: effective_resistance.rpt

```
# instance:pin    reff_stats  pin_box
xofiller!FILLCELL_X1!x368600y1218000:VDD = Min: 9.90307 Max: 9.90307 Mean: 9.90307 Count: 1 (pin_box=36955000,12320000
0,36955000,123200000)
core2.regfile_data_memory.MUX2_X1_1132:VSS = Min: 7.76756 Max: 8.27185 Mean: 8.0197 Count: 2 (pin_box=21970000,1134000
00,22730000,113400000)
xofiller!FILLCELL_X8!x190000y1428000:VSS = Min: 9.75544 Max: 9.75544 Mean: 9.75544 Count: 1 (pin_box=19685000,14420000
0,19685000,144200000)
xofiller!FILLCELL_X1!x146300y938000:VSS = Min: 13.0931 Max: 13.0931 Mean: 13.0931 Count: 1 (pin_box=14725000,93800000,
14725000,93800000)
xofiller!FILLCELL_X1!x967100y42000:VSS = Min: 8.48604 Max: 8.48604 Mean: 8.48604 Count: 1 (pin_box=96795000,4200000,96
795000,4200000)
xofiller!FILLCELL_X4!x108300y1372000:VSS = Min: 15.3883 Max: 15.3883 Mean: 15.3883 Count: 1 (pin_box=11210000,13860000
0,11210000,138600000)
```

BQM: Build Quality Metric

A better approach for static grid checking

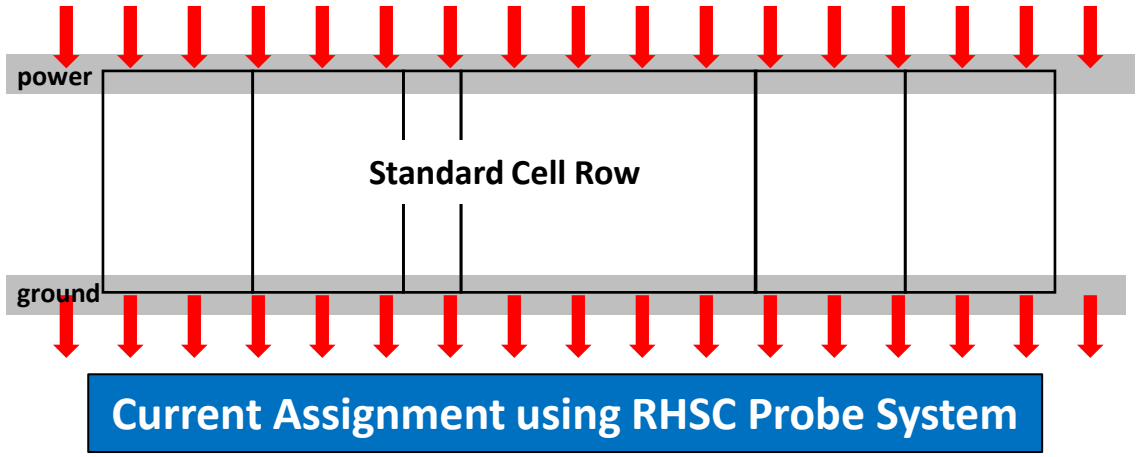
- BQM is a grid reliability check based on static simulation and presented as normalized voltage drop on every single node

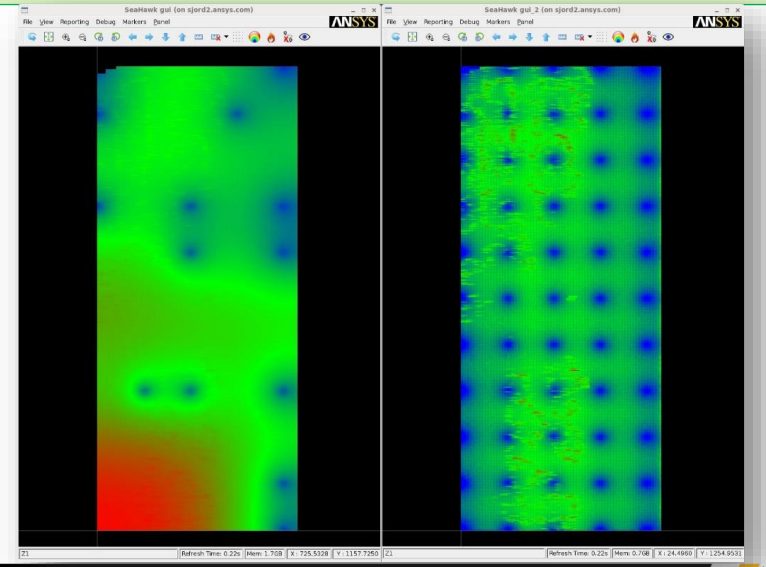
Approach

- Find all low metal PG segments where standard cells may connect
- Apply a constant current source at regular pitch on these segments
- Perform static voltage drop simulation
- Generate heatmaps to show relative weak areas of the grid

Benefits

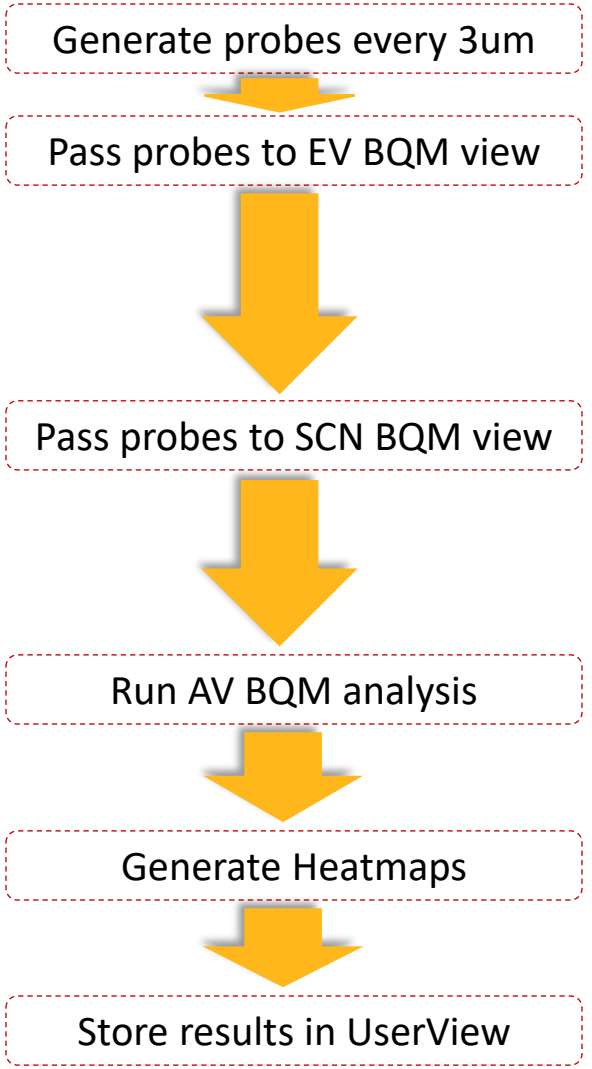
- Finds weakness issues in entire grid, even before instance placement
- More efficient than SPR or $R_{effective}$
- Finds problems that SPR or $R_{effective}$ can miss
- Flexible and extensible
- Analytics: BQM heatmap can be combined with other data



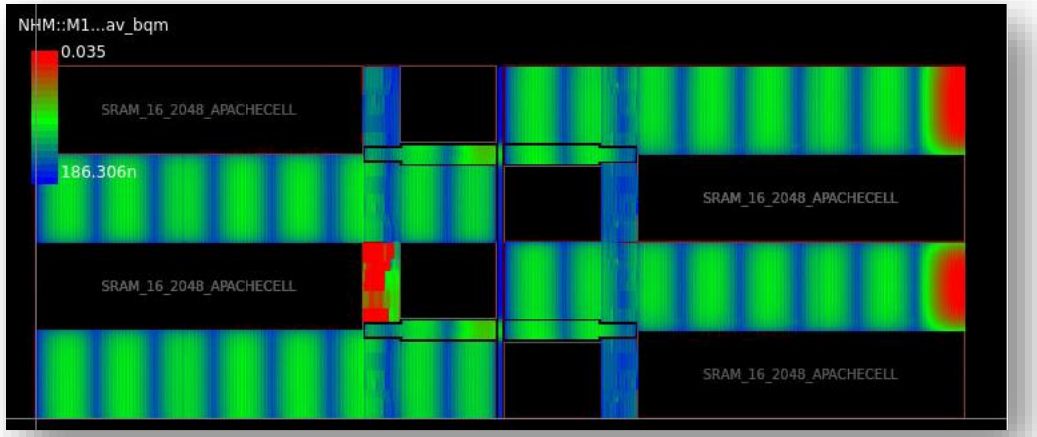


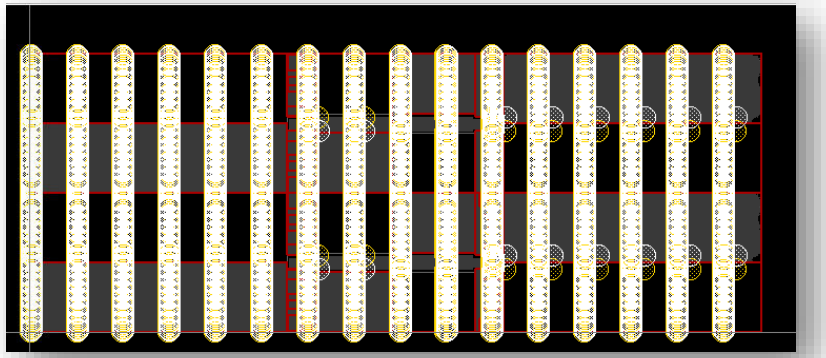
Running BQM analysis

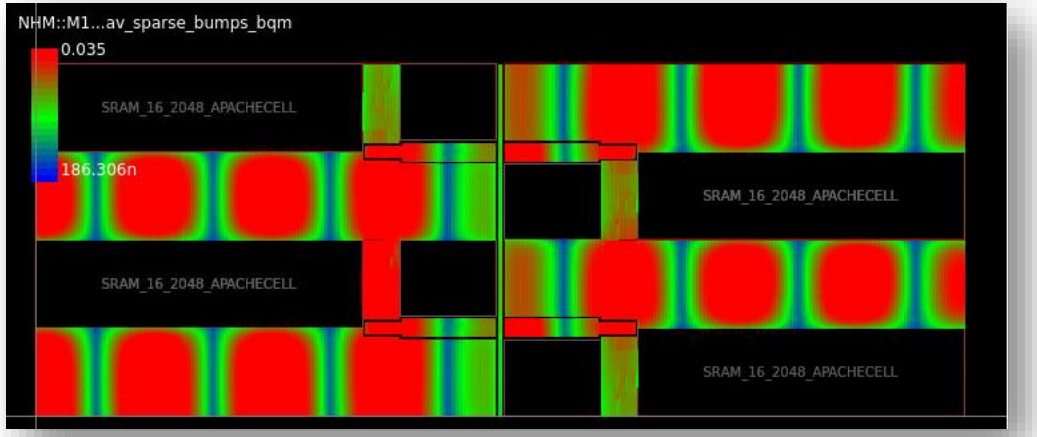
```
bqm_probes, bqm_currents = bqm_current.generate_bqm_current_probe_metal(dv, Layer('M1'), probe_distance=3.0)

ev_bqm = db.create_modified_extract_view(ev, probes=bqm_probes, tag='ev_bqm', options=options)

settings_bqm_scn = {
    'pvt': {'voltage_levels': []},
    'probe_sources': bqm_currents,
    'scenario_type': 'External'}

settings_bqm_av = {
    'duration': 0,
    'keep_stats_level': 'Bqm',
    'scheduler_barrier': False}

scn_bqm = db.create_scenario_view(design_view=dv, settings=settings_bqm_scn, tag='scn_bqm', options=options)

sv_bqm = db.create_simulation_view(ev_bqm, tag='sv_bqm', options=options)

av_bqm = db.create_analysis_view(sv_bqm, scn_bqm, settings=settings_bqm_av, tag='av_bqm', options=options)

pg_nets = dv.get_nets('pg')
bqm_inst = generate_drop_heatmaps.generate_pin_voltage_heatmaps(av_bqm, nets=pg_nets, base_level=True)

bqm_data = db.create_user_view(tag='bqm_data')
for kk in bqm_inst:
    bqm_data['hm_' + kk.get_name()] = bqm_inst[kk]
    bqm_data['stats_' + kk.get_name()]= normalize_heatmaps.get_heatmap_min_max_median(dv, bqm_inst[kk])
```

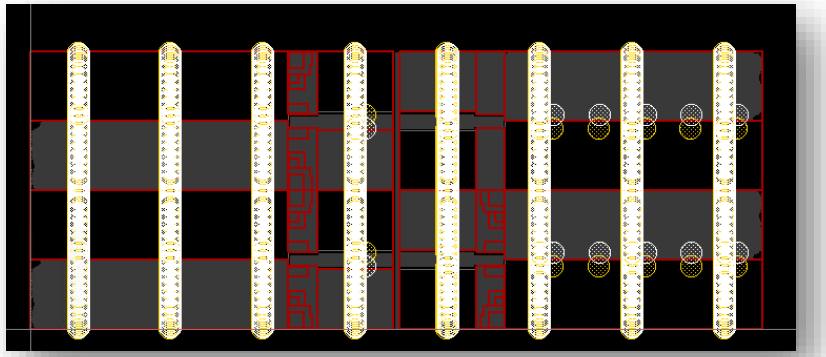


Advantage of running BQM over SPR









- BQM detects the poorly-connected area of the VSS grid which was missed entirely by SPR

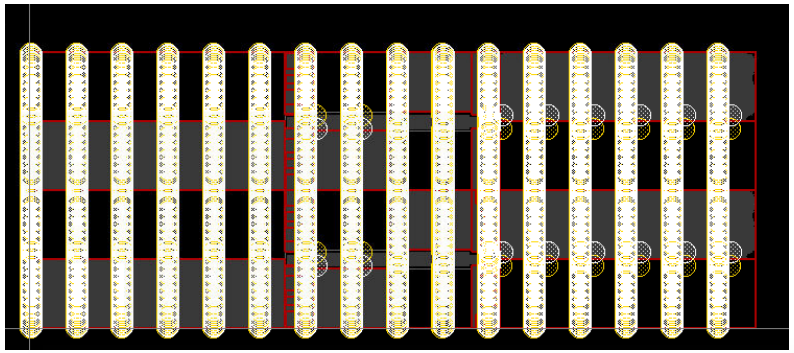
SPR vs R-eff vs BQM: Comparative Study

- SPR/R-eff/BQM comparison performed with 3 different bump configurations

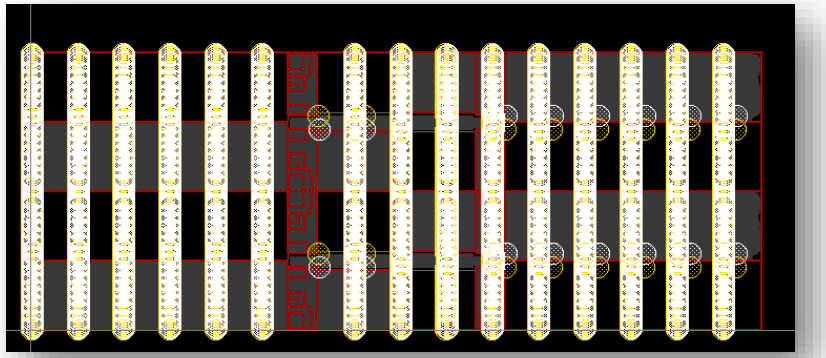
Run	Configuration / Setup
Run-1	Original bumps
Run-2	Removed column of bumps over some cores
Run-3	Removed multiple columns of bumps

- Metal1 Heatmaps are compared in the following section

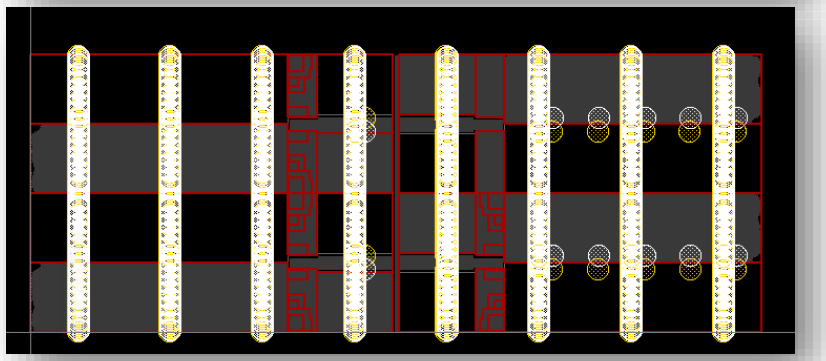
Run-1



Run-2

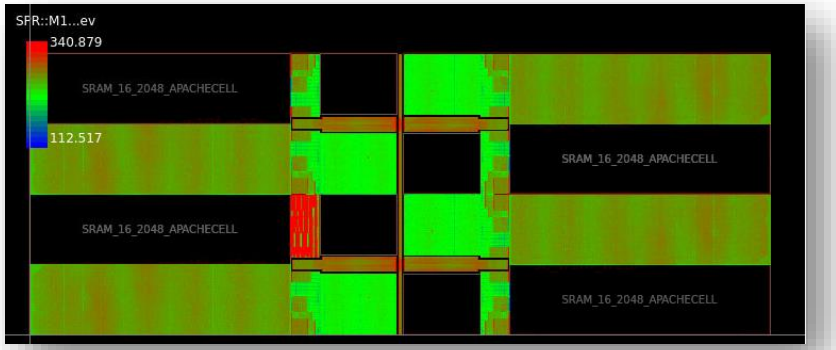


Run-3

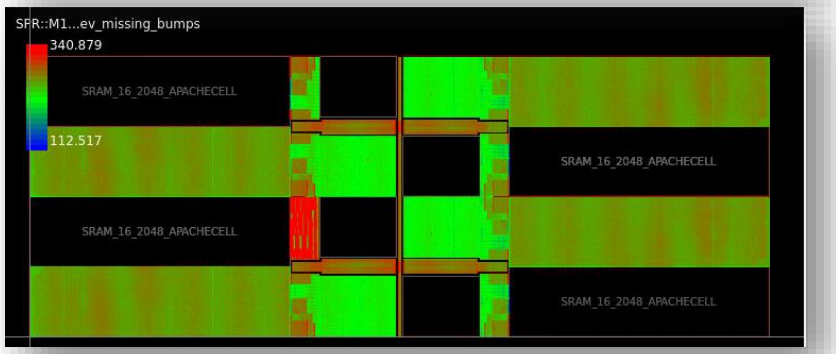


SPR Comparison

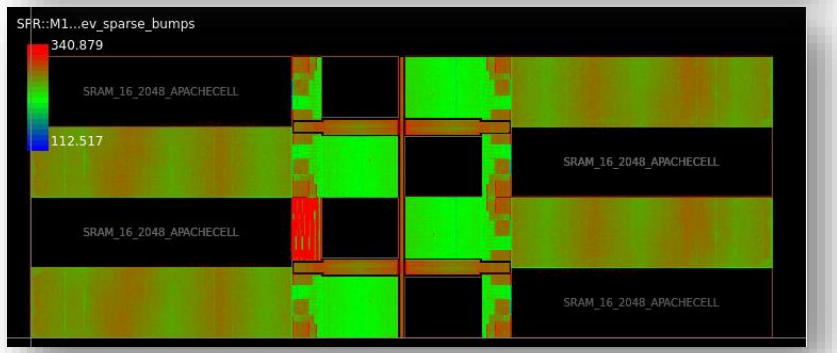
Run-1



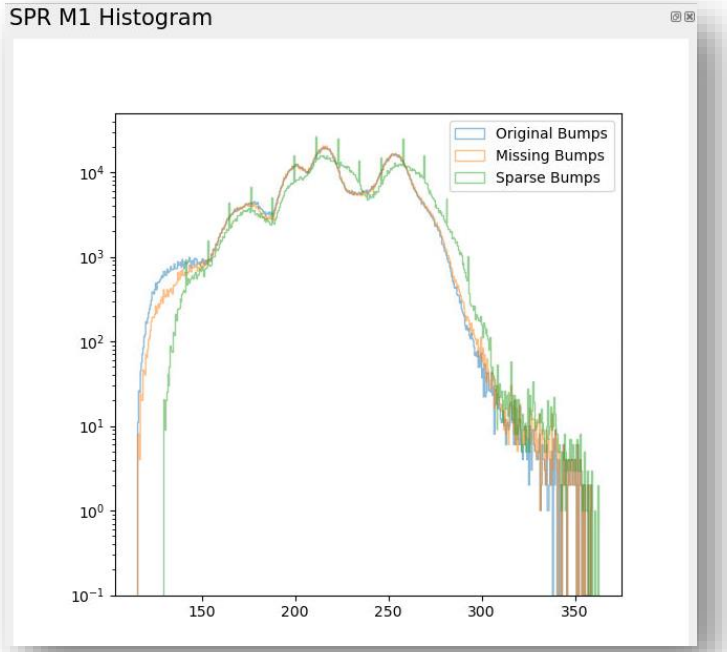
Run-2



Run-3



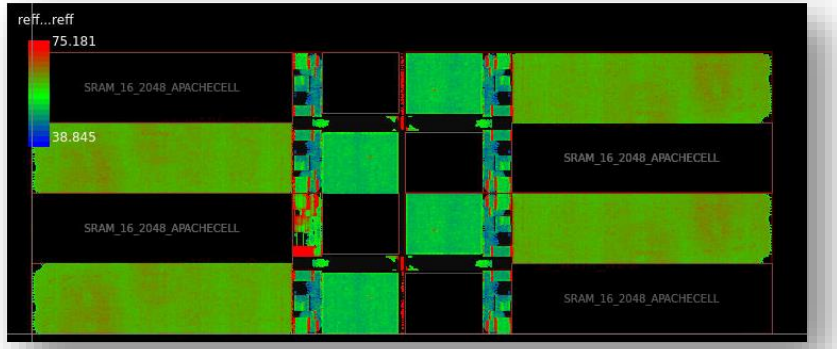

```
spr_hm = ev.get_spr_histograms()
plot(spr_hm[Net('VDD_cmplx')][Layer('M1')])
```



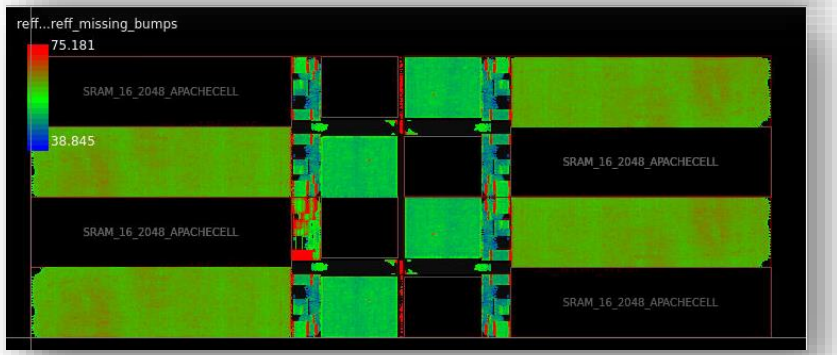
- All three grids show near identical SPR results (both heatmaps and histograms)
- SPR doesn't clearly capture the effect of removing bumps
- Since most of the SPR resistance is at Metal1, SPR fails to highlight problems above Metal1

Reff Comparison

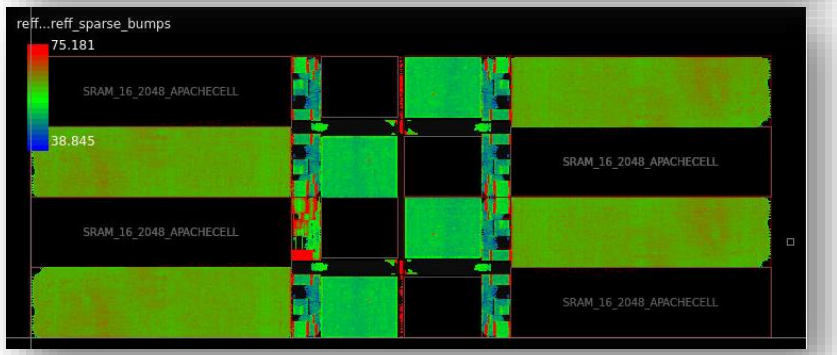
Run-1



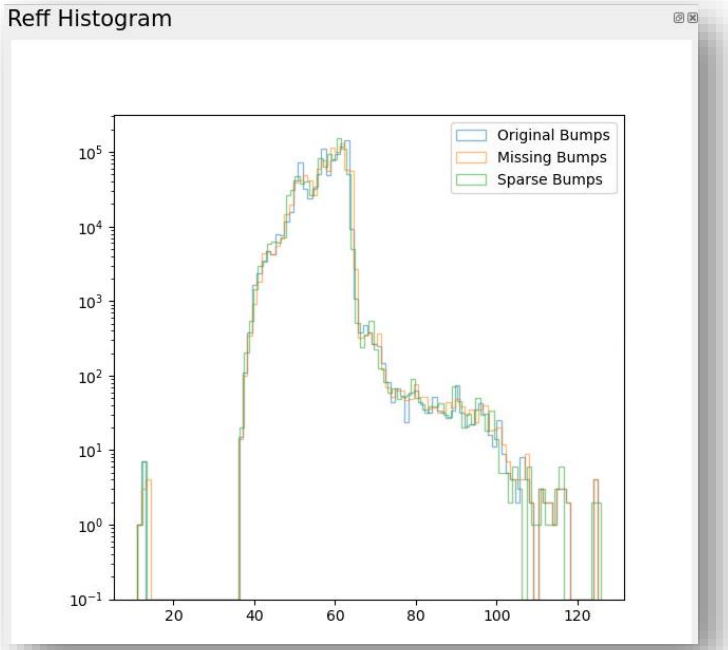
Run-2



Run-3



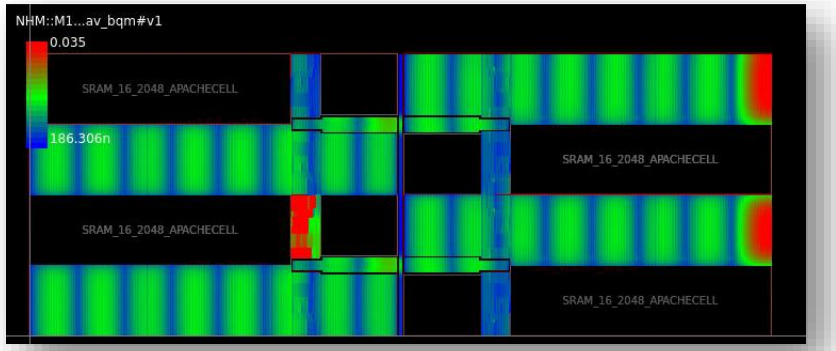

```
reff_hm = reff.get_effective_resistance_histograms()
plot(reff_hm[Net('VDD_cmplx')])
```



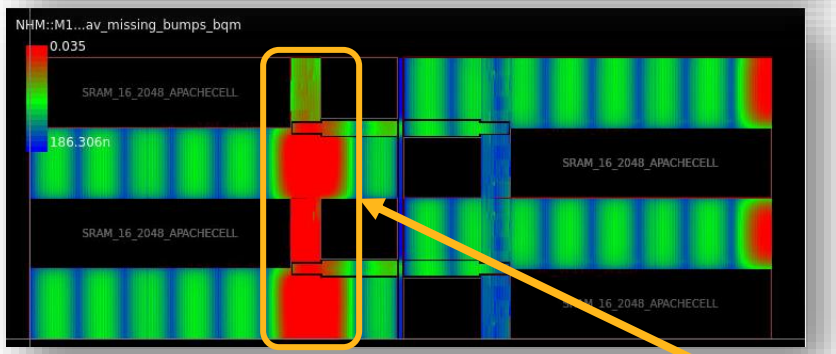
- All three grids show near identical Reff results (both heatmaps and histograms)
- Reff fails to capture the effect of removing the bumps
- Since Metal1 contributes most to effective resistance, Reff fails to highlight problems above Metal1

BQM Comparison

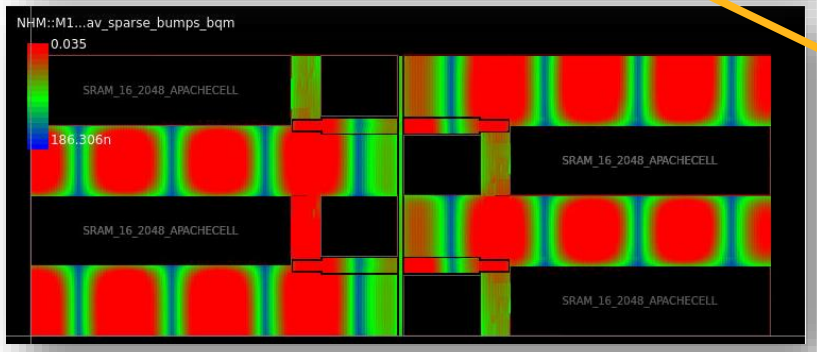
Run-1



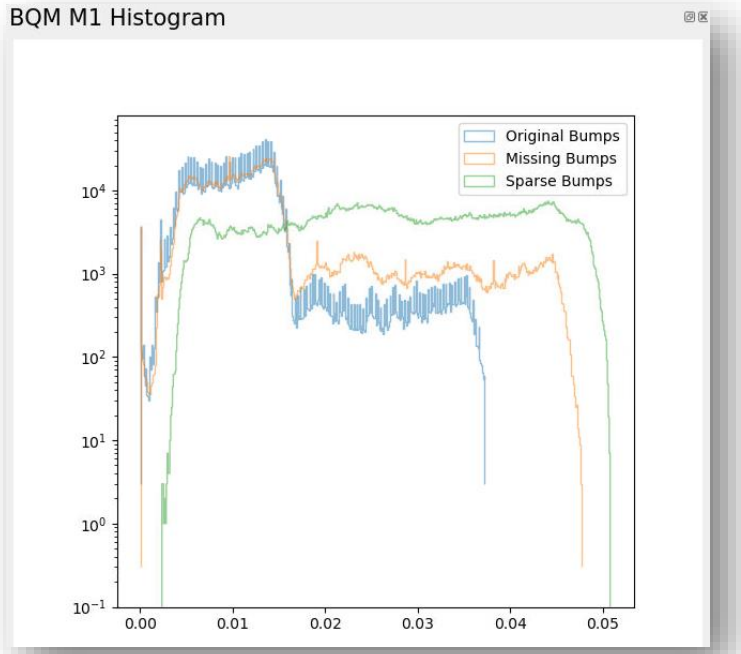
Run-2



Run-3




```
av_bqm_hm = av_bqm.get_effective_resistance_histograms()
plot(av_bqm_hm[Net('VDD_cmplx')][Layer('M1')])
```

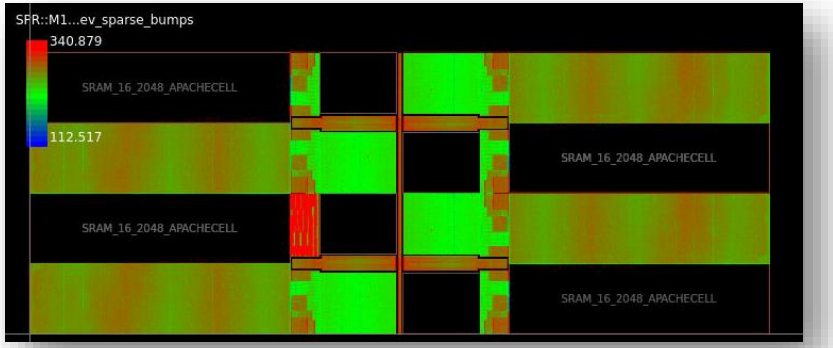


- BQM captures bump weakness more accurately
- BQM is able to catch the bump removal in Run-2
- The hot-ness of heatmap increases in the expected order (Run-3 > Run-2 > Run-1)
- Histograms show expected trends

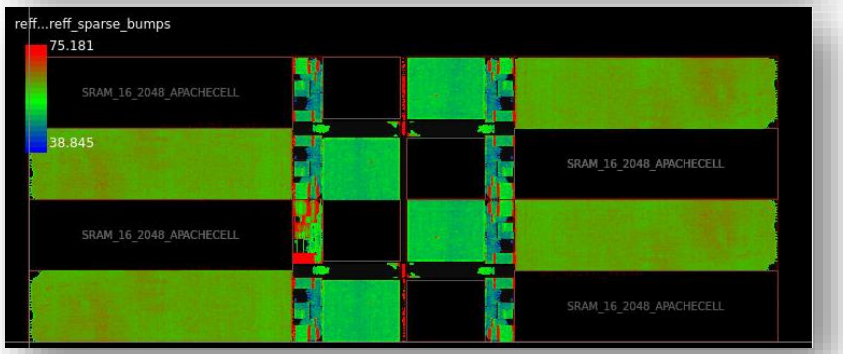
Comparison Summary & Conclusion

- In terms of value, BQM >> SPR >> Reff
- SPR can be used for manual debug of problem areas
  - Caution: good SPR does not mean PG-grid is good
- Reff can be used to understand relative strength of a PG-grid in early prototyping stages

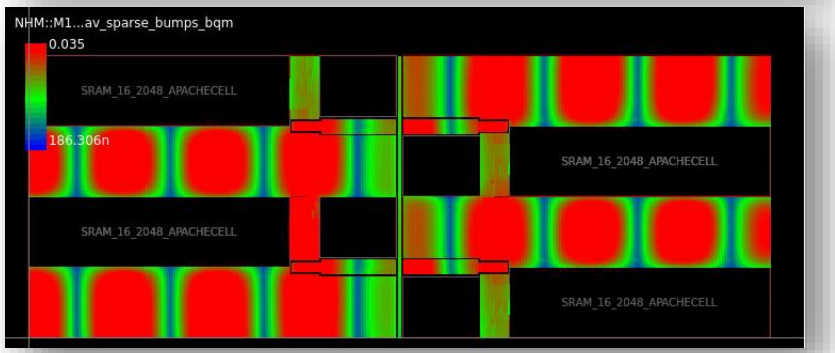
SPR



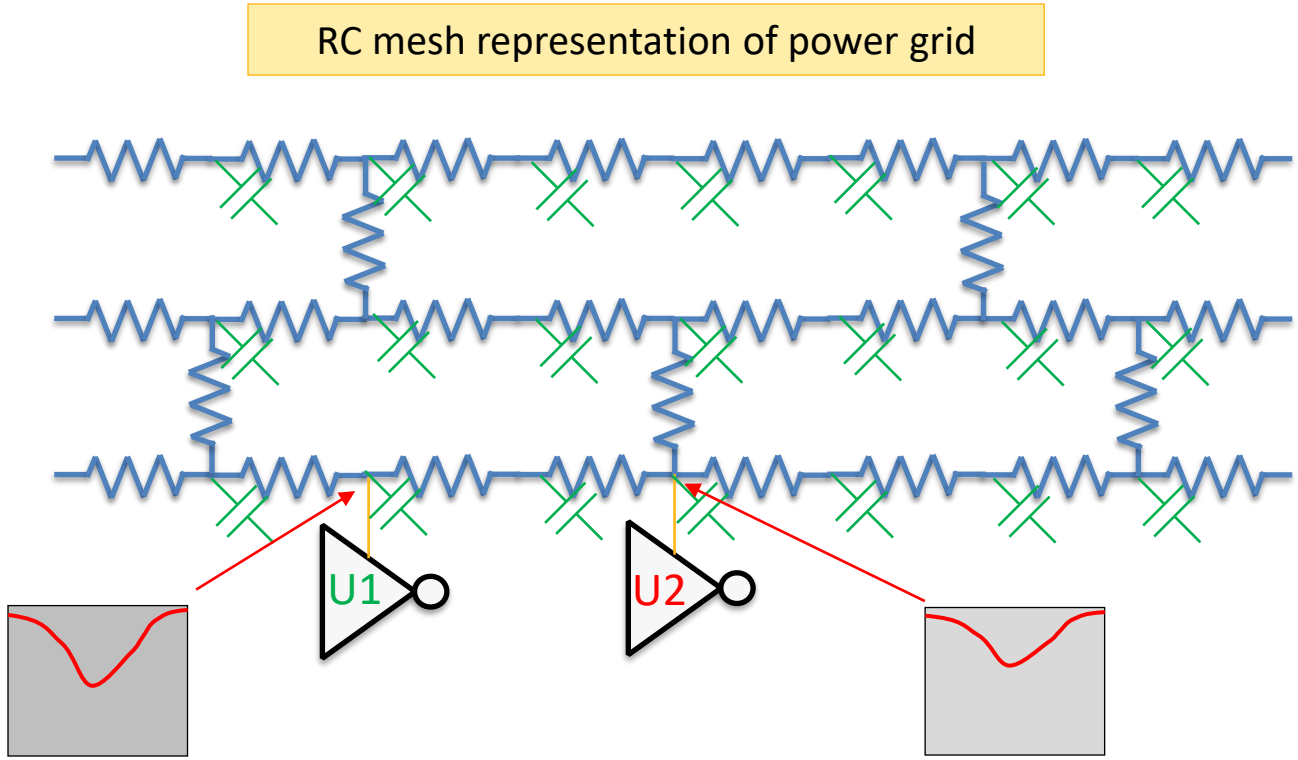
Reff



BQM



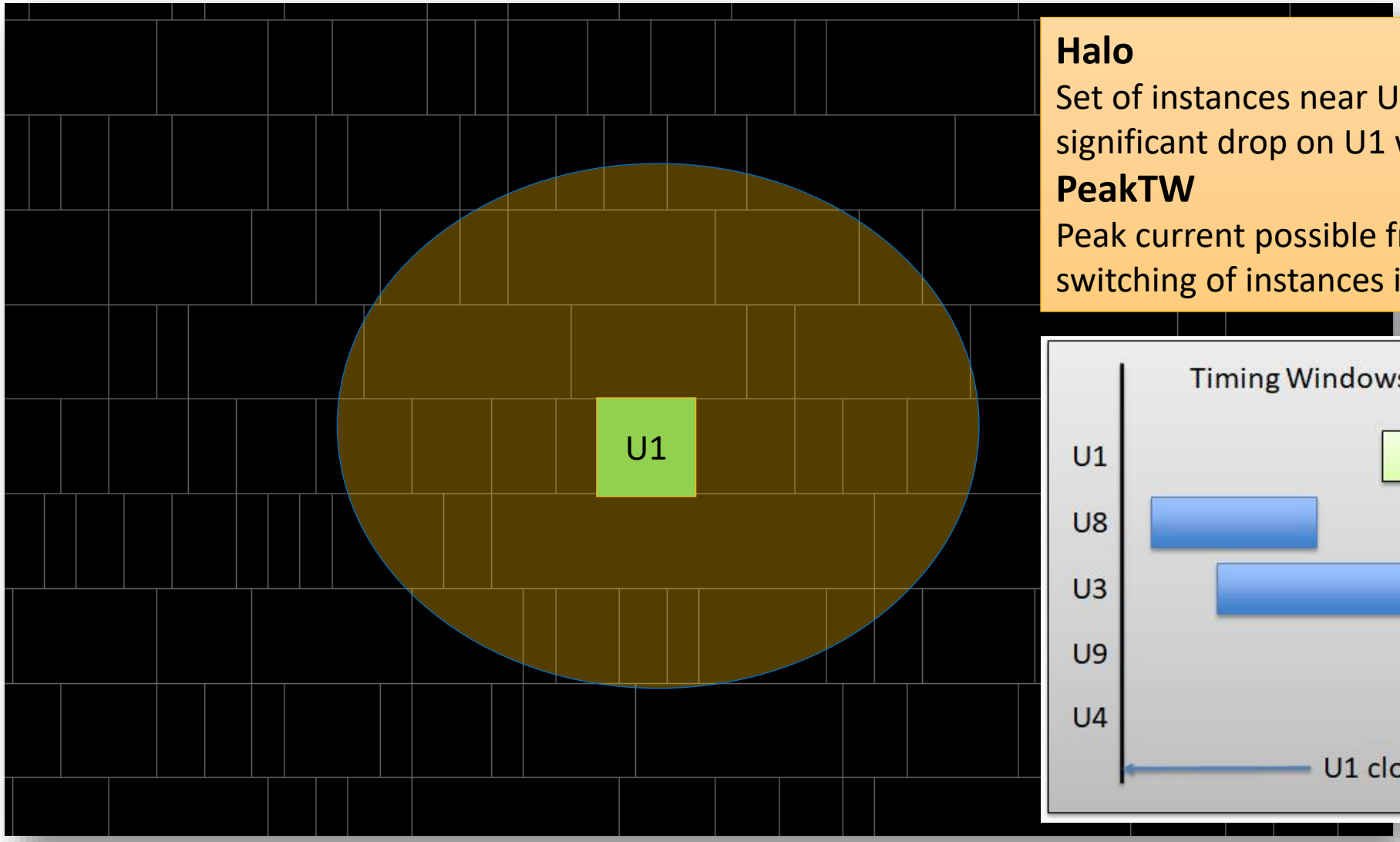
Voltage Drop From Self & Neighbor Switching



Effects on switching on instance voltages:

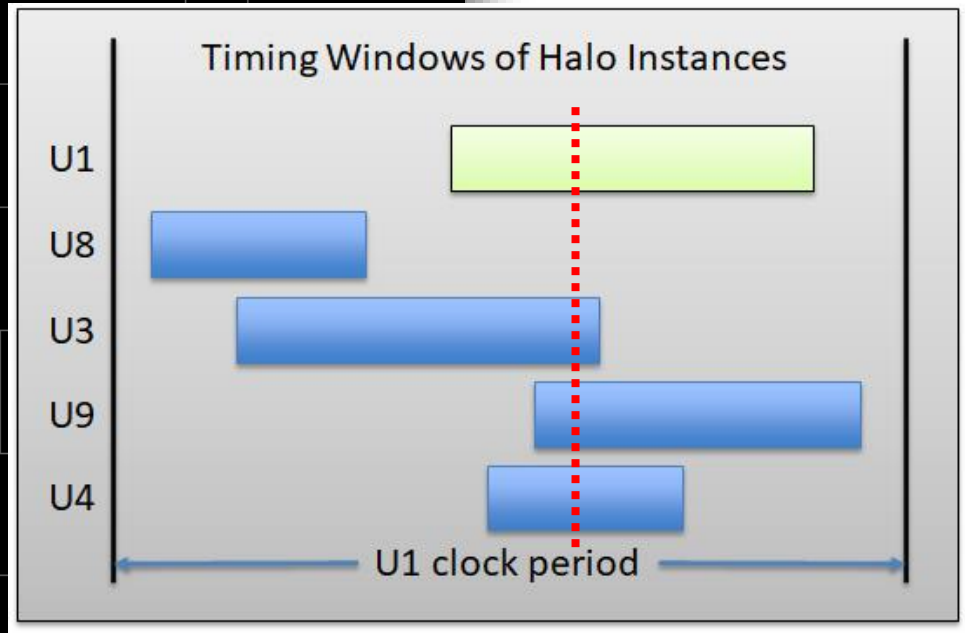
- **Self**: when U1 switches, it draws current from the power grid, reducing the voltage seen at U1
- **Neighbors**: when U1 switches, some voltage drop is also seen by nearby instances such as U2

PeakTW: Peak Current for Instance + Halo



Halo
Set of instances near U1 that can cause significant drop on U1 when it is switching

PeakTW
Peak current possible from simultaneous switching of instances in halo



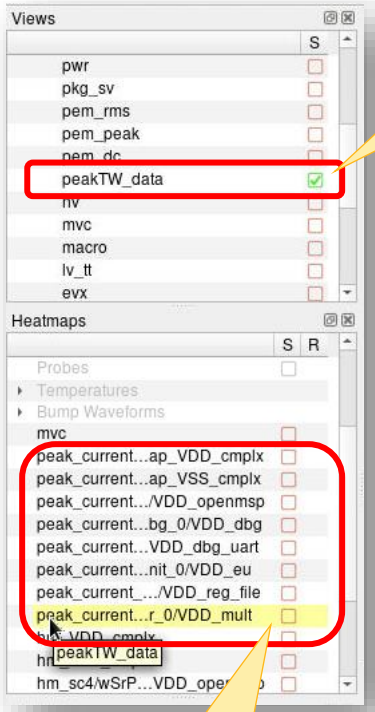
Running PeakTW Analysis

File: scripts/grid_robustness_checks.py

```
peaktw_dict = generate_peak_power_views.generate_peak_power_views(timing_view=tv,
external_parasitics=evx, extract_view=ev, per_domain_maps=True)

peaktw_data = db.create_user_view(tag='peakTW_data')
for map_type in peaktw_dict:
    for pgnet in peaktw_dict[map_type]:
        if map_type == 'peak_current_heatmap_TW_overlap':
            peaktw_data[map_type + '_' + pgnet.get_name()] = peaktw_dict[map_type][pgnet]
```

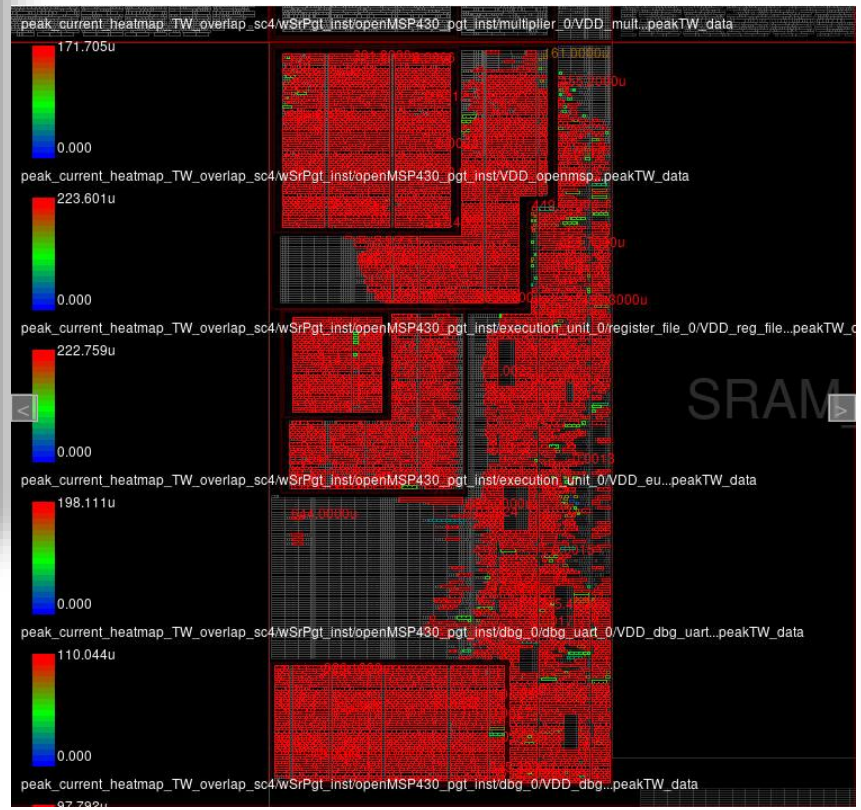
- Running the above script will produce heatmap which has the statistical halo current around every instance in the design
- The above heatmap is generated per domain and stored in the UserView called 'peakTW_data'

Looking at PeakTW Results

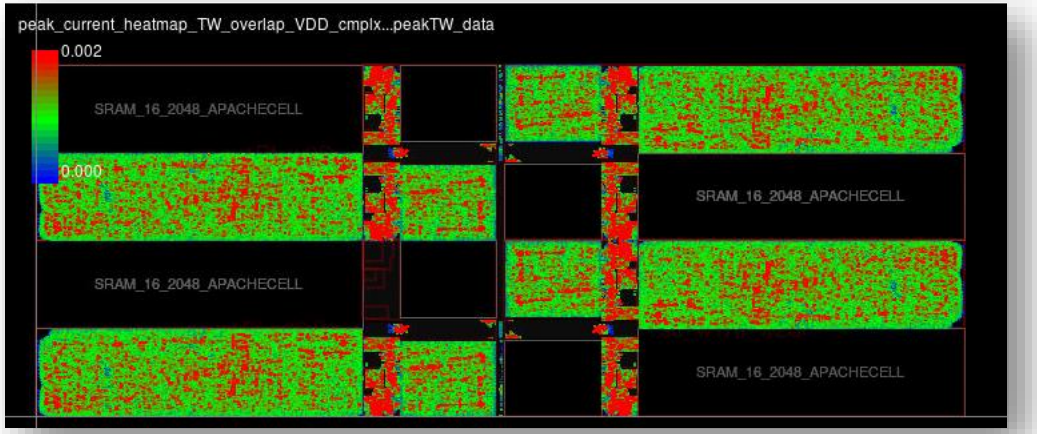


Select "peakTW_data" View

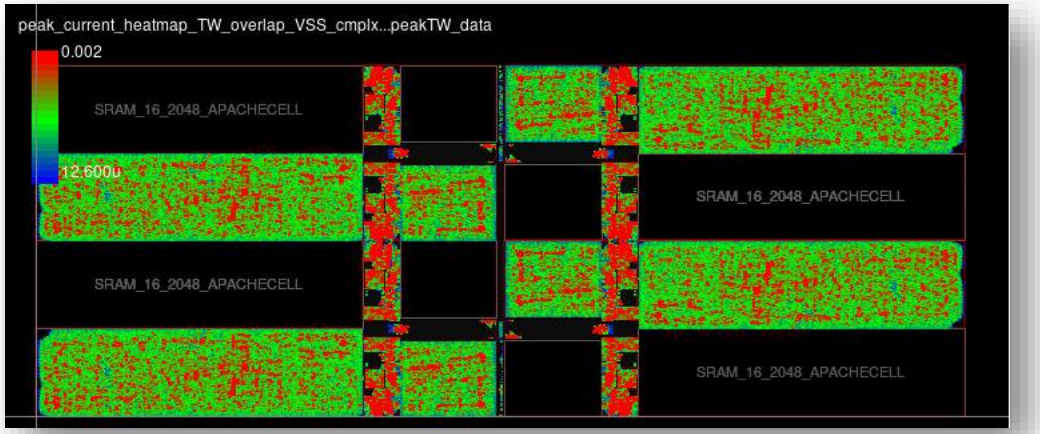
Enable each peak current heatmap domain by domain



Power Gated domains PeakTW map



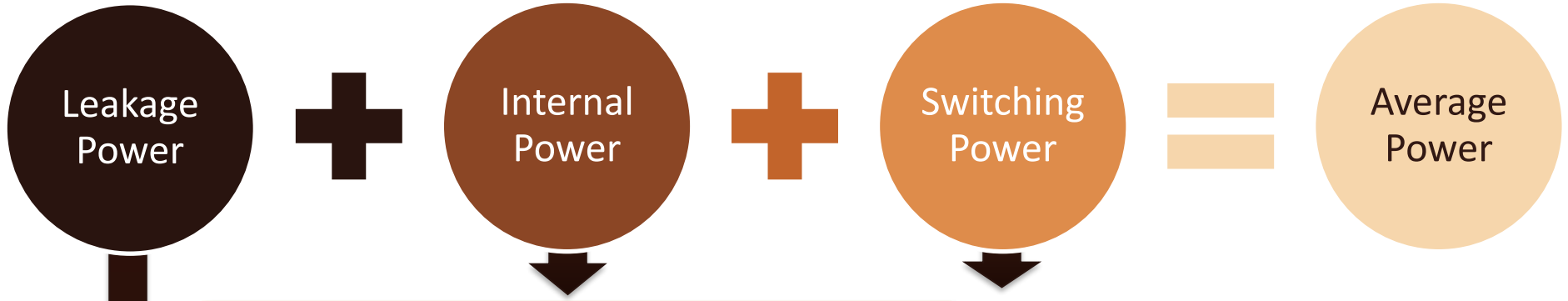
VDD_cmplx domain PeakTW map



VSS_cmplx domain PeakTW map

Power Calculation

Power Calculation Basics




```
leakage_power () {
  when            : "!A1 & !A2";
  value           : 20.324370;
}
leakage_power () {
  when            : "!A1 & A2";
  value           : 30.850688;
}
leakage_power () {
  when            : "A1 & !A2";
  value           : 20.622958;
}
leakage_power () {
  when            : "A1 & A2";
  value           : 28.466240;
}
```

From .lib

```
pin (ZN) {
  direction           : output;
  related_power_pin   : "VDD";
  related_ground_pin  : "VSS";
  max_capacitance     : 60.577400;
  function            : "(A1 & A2)";
…
  internal_power () {
    related_pin              : "A1";
    fall_power(Power_7_7) {
      index_1 ("0.00117378,0.00472397,0.0171859,0.0409838,0.0780596,0.130081,0.198535");
      index_2 ("0.365616,1.893040,3.786090,7.572170,15.144300,30.288700,60.577400");
      values ("2.707163,2.939134,3.111270,3.271119,3.366153,3.407657,3.420511", \
              …
              "4.526175,4.492292,4.510220,4.634217,4.814899,4.934862,5.047389");
    }
    rise_power(Power_7_7) {
      index_1 ("0.00117378,0.00472397,0.0171859,0.0409838,0.0780596,0.130081,0.198535");
      index_2 ("0.365616,1.893040,3.786090,7.572170,15.144300,30.288700,60.577400");
      values ("1.823439,1.926997,1.963153,2.028865,1.957837,2.123314,2.075262", \
              …
              "3.687718,3.756085,3.789394,3.792984,3.773583,3.593022,3.405552");
```

From .lib

Switching Power = ½ * Cap * V^2 * Freq * Toggle_Rate

From SPEF

From Timing File

Internal Power = Internal Energy(in .lib) * Freq * Toggle Rate

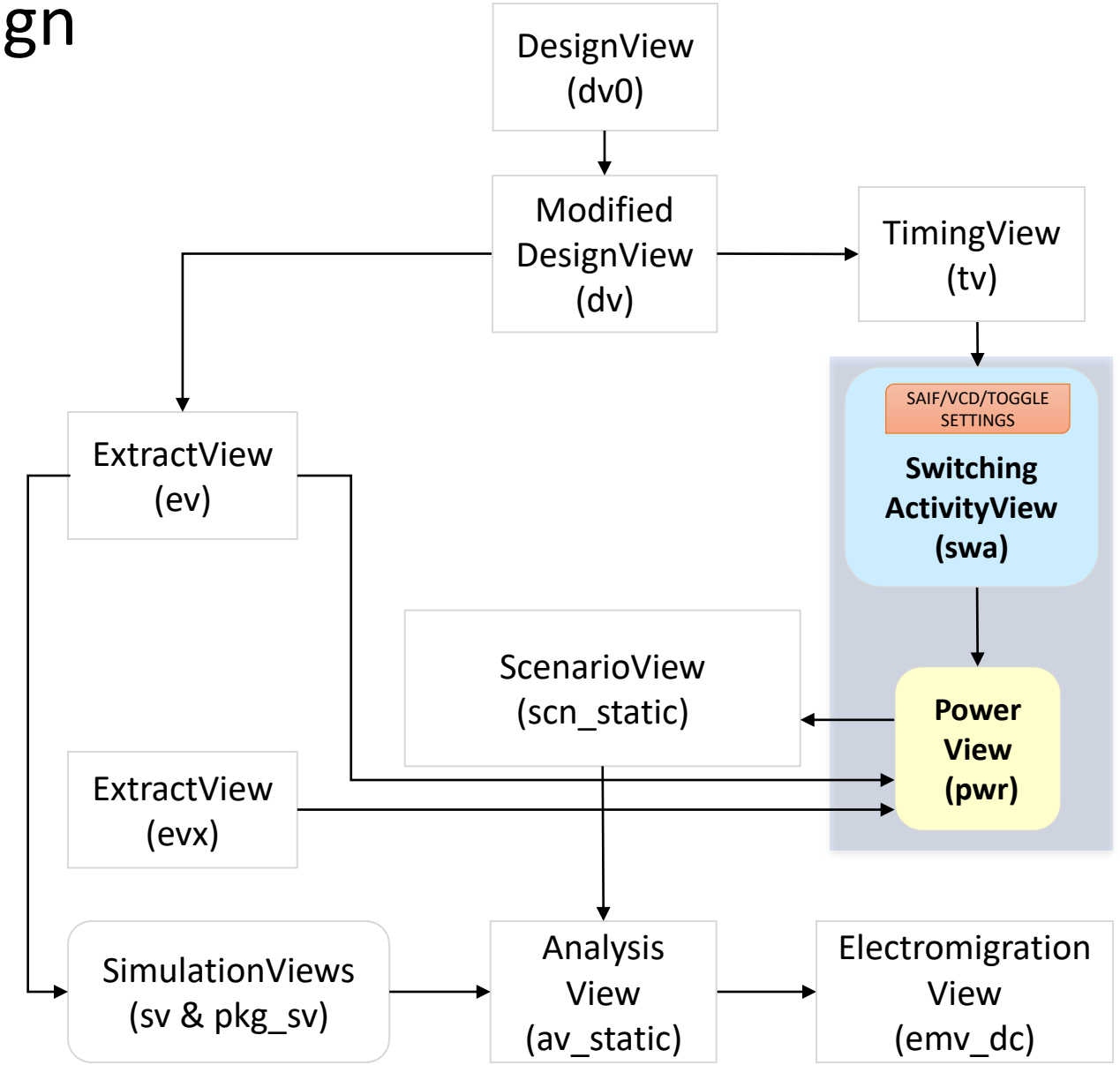
P(total) = P(Leakage) + ½ * (Internal-energy * Freq * TR) + ½ * C * V^2 * Freq * TR

Setting the activity in the design

- **SwitchingActivityView**
  - Switching activity can be propagated from registers and primary inputs
    - Computes the probability of signals being at 1 or 0 and how frequently (toggle rate) they can switch over a period of time.
  - Propagation is not needed if assigning uniform activity or SAIF/VCD/FSDB covers all instances
  - View contains the #transitions per second and probabilities of signal remaining high/low for every net
- **Required inputs**
  - DesignView
  - TimingView
- **Input activity set using one or more of the following methods:**
  - User-controlled settings
    - Default and block level Toggle Rates by cell types (clock, icg,
      sequential, macro, combinatorial)
    - Individual instance controls
  - Gate/RTL VCD/FSDB Files or ValueChangeView (VCV)
  - SAIF

For more details, see:

help(SeaScapeDB.create_switching_activity_view)

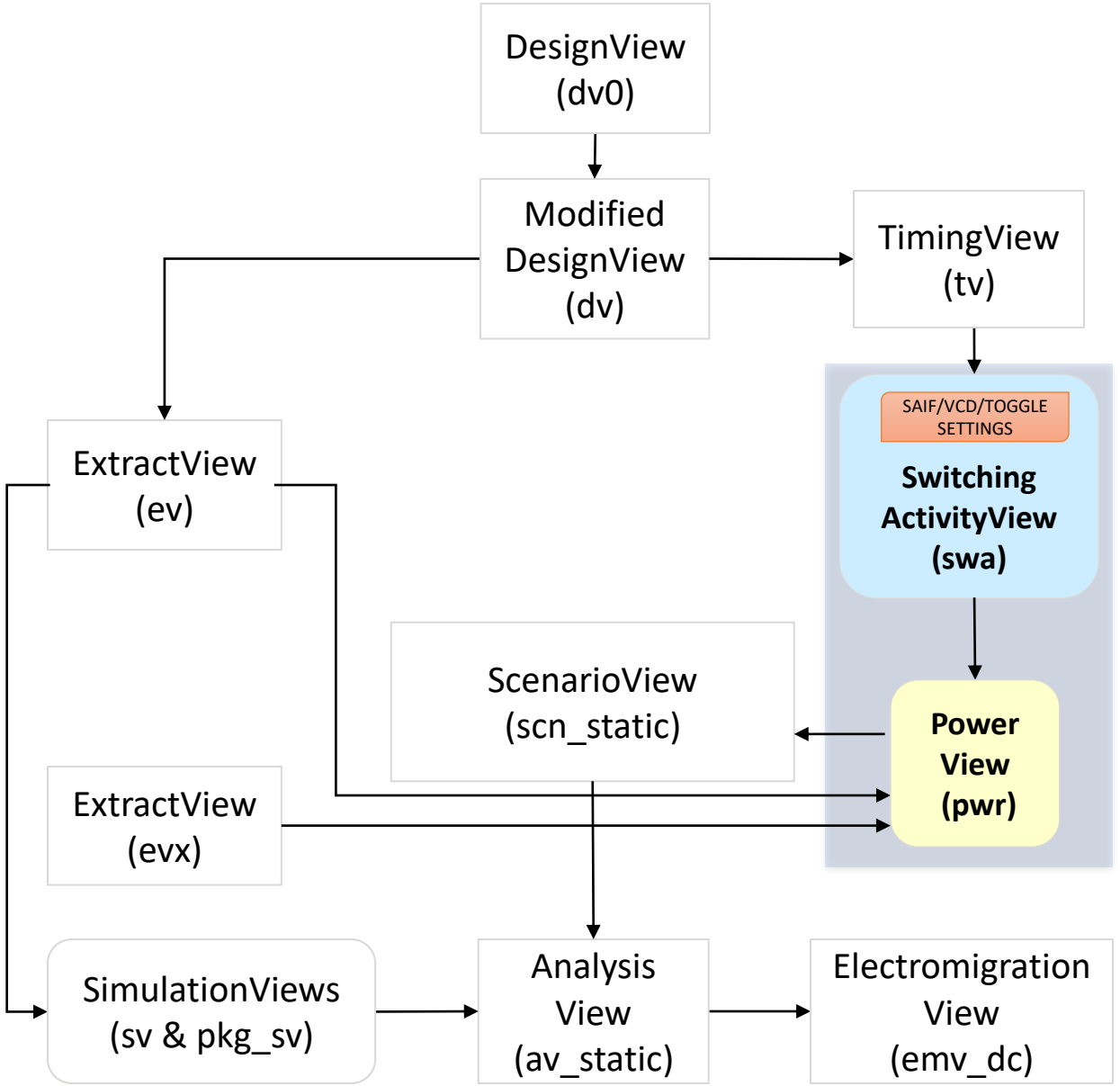


Performing Power Calculation

- **PowerView**
  - Contains the power per instance and its related info like load, slew, toggle rate, frequency, voltage levels, etc
- **Inputs for creating PowerView:**
  - Design View
  - Switching Activity View (Toggles)
  - Instance Power File, IPF (Direct power information)
  - ExtractView (SPEF based or RedHawk-SC native for load information)
  - Timing View (Timing)
  - Settings for macro power control through mode control

For more details, see:

help(SeaScapeDB.create_power_view)



Performing Power Calculation

File: scripts/powercalc.py

```
options = get_default_options()

voltage_levels = {'VDD_cmplx': 0.7, 'VSS_cmplx': 0.0}
default_period = 1.4e-09
clock_toggle_rate = 2.0
icg_toggle_rate = 1.4
data_toggle_rate = 0.1

object_settings_swa = {
    'design_values': {
        'clock_period': default_period,
        'clock_pin_toggle_rate': clock_toggle_rate,
        'icg_output_pin_toggle_rate': icg_toggle_rate,
        'macro_output_pin_toggle_rate': data_toggle_rate,
        'sequential_output_pin_toggle_rate': data_toggle_rate,
        'combinational_pin_toggle_rate': data_toggle_rate,
        'primary_input_port_toggle_rate': data_toggle_rate,
        'activity_precedence': ['default_activity']}}

settings_swa = {'object_settings': object_settings_swa}

swa = db.create_switching_activity_view(tv, settings=settings_swa, tag='swa', options=options)

object_settings_pwr = {
    'leaf_instance_values': [
        {'instances': Instance('sc1/wSr_inst/pmem'), 'mode_control': {'mode_sequence': ['read'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc2/wSr_inst/dmem'), 'mode_control': {'mode_sequence': ['write'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc3/wSrLPF_inst/dmem'), 'mode_control': {'mode_sequence': ['standby_trig'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc1/wSr_inst/dmem'), 'mode_control': {'mode_sequence': ['standby_trig'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc4/wSrPgt_inst/dmem'), 'mode_control': {'mode_sequence': ['read'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc2/wSr_inst/pmem'), 'mode_control': {'mode_sequence': ['standby_ntrig'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc3/wSrLPF_inst/pmem'), 'mode_control': {'mode_sequence': ['read'], 'mode_sequence_repeats': True}},
        {'instances': Instance('sc4/wSrPgt_inst/pmem'), 'mode_control': {'mode_sequence': ['write'], 'mode_sequence_repeats': True}}]}

settings_pwr = {
    'pvt': {'voltage_levels': voltage_levels, 'process_corner': 'tt'},
    'object_settings': object_settings_pwr}

pwr = db.create_power_view(switching_activity_view=swa, external_parasitics=evx, extract_view=ev, settings=settings_pwr, tag='pwr', options=options)

reports_dir = 'reports/powercalc/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)
emir_reports.write_instance_power_report_and_summary(pwr, reports_dir + 'instance_power.rpt', reports_dir + 'power_summary.rpt')
```

Variables

Setting up Toggle Rates

SwitchingActivityView

Controlling Macros for power calculation

PowerView

Power Reporting

Looking At Power Calculation Reports

File: power_summary.rpt

```
**** RedHawk-SC Power Summary Report ****
Created: Mon 15 Aug 2022 02:54:21 PM

----------------------------------------- Block "galaxy_cmplx" summary start -----------------------------------------
A total of 1057916 instances were summarized for this report while 126936 were omitted due to missing power data (88.00% coverage).

A total of 0 pins were omitted because they were not attached to a power domain.

***

grouping            clock_pin_power(W)  internal_power(W)  leakage_power(W)  switching_power(W)  total_power(W)  percent_power(%)  pin_count  instance_count

*** Power Domains:
VDD_cmplx                0.060979           0.077606          0.0030046          0.074075           0.15469           99.15         920972      920972
sc4/wSrPgt_inst/openMSP430_pgt_inst/VDD_openmsp    7.4558e-05         0.00017567         1.9962e-05         0.00024469         0.00044032          0.28          3205        3205
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/VDD_reg_file    7.0724e-05         0.00012533         1.1389e-05         0.00022706         0.00036377          0.23          1739        1739
sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult    2.2343e-05         6.0168e-05         1.0368e-05         0.00010174         0.00017228          0.11          1952        1952
sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu    9.5072e-06         4.5521e-05         1.1025e-05         0.0001011          0.00015765          0.10          1646        1646
sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/VDD_dbg     1.893e-05          4.0749e-05         7.1441e-06         5.4502e-05         0.0001024           0.07          871         871
sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/dbg_uart_0/VDD_dbg_uart    2.2345e-05         3.8464e-05         3.9508e-06         4.9692e-05         9.2106e-05          0.06          595         595
Total                    0.061198           0.078092          0.0030684          0.074853           0.15601           100.00        930980      930980

*** Frequency Domains:
7.143e+08                0.061198           0.078092          0.0020653          0.074853           0.15501           99.36         928628      928628
0                           0               1.1237e-08         0.0010031             0               0.0010032          0.64          2352        2352
Total                    0.061198           0.078092          0.0030684          0.074853           0.15601           100.00        930980      930980

*** User Defined Groups:
combinational logic      0.00032812         0.012402           0.003055          0.070783           0.086239          55.28         761124      761124
sequential logic          0.06087           0.065691          1.3422e-05         0.0040708          0.069775          44.72         169856      169856
Total                    0.061198           0.078092          0.0030684          0.074853           0.15601           100.00        930980      930980
----------------------------------------- Block "galaxy_cmplx" summary end -------------------------------------------

**** End Report ****
```

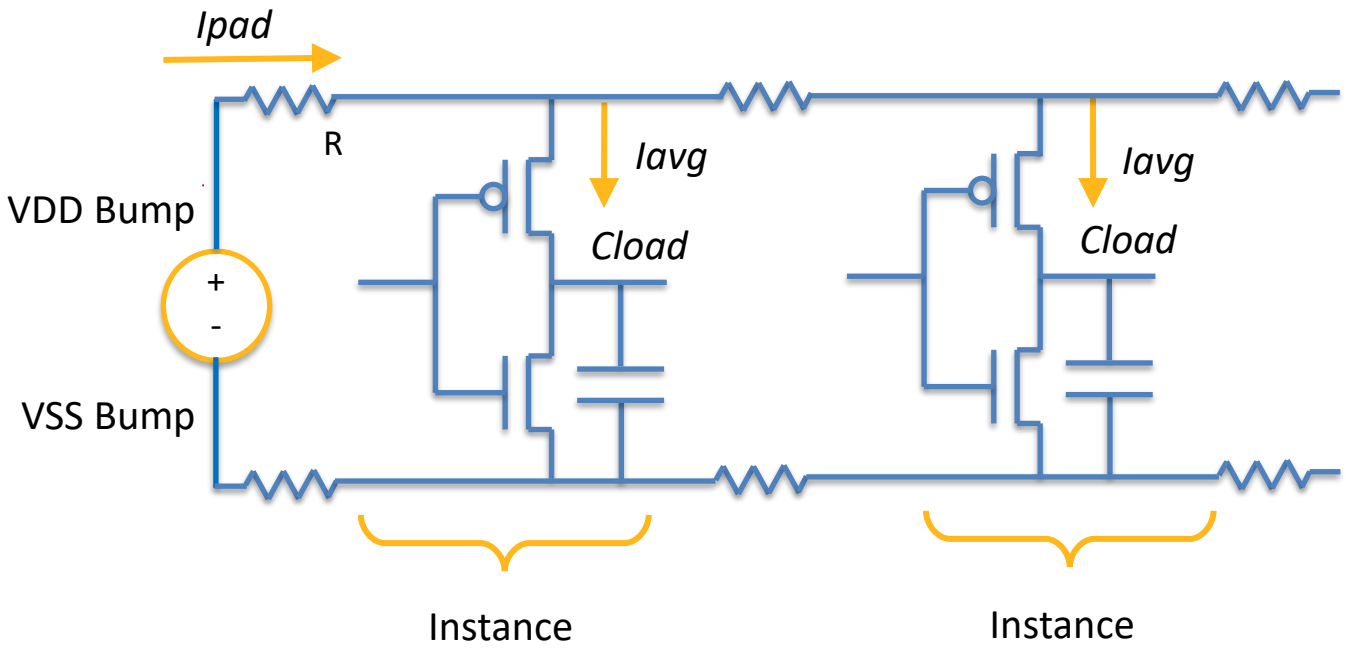
Power Calculation Reports available in "reports/powercalc" directory

File: instance_power.rpt

```
# clock_pin_power is included in internal_power
# pin     domain  frequency  toggle_rate  clock_pin_power  internal_power  leakage_power  switching_power  total_power  voltage  cell_name                instance
#                   (Hz)                       (W)              (W)             (W)             (W)            (W)        (V)
  VDD    VDD_cmplx 7.143e+08     0.10         9.201e-05        9.087e-05       3.346e-05        1.14e-06      0.0001255    0.70    SRAM_16_2048_APACHECELL   sc3/wSrLPF_inst/pmem/adsU1
  VDD    VDD_cmplx 7.143e+08     0.10         9.201e-05        9.088e-05       3.346e-05       1.125e-06      0.0001255    0.70    SRAM_16_2048_APACHECELL   sc1/wSr_inst/pmem/adsU1
  VDD    VDD_cmplx 7.143e+08     0.10         9.201e-05        9.175e-05       8.602e-06        2.61e-07      0.0001006    0.70    SRAM_16_512_APACHECELL    sc4/wSrPgt_inst/dmem/adsU1
  VDD    VDD_cmplx 7.143e+08     0.10         2.591e-05        2.537e-05       3.346e-05       5.332e-07      5.937e-05    0.70    SRAM_16_2048_APACHECELL   sc4/wSrPgt_inst/pmem/adsU1
  VDD    VDD_cmplx 7.143e+08     0.10          3.94e-08            0            3.346e-05       1.125e-06      3.459e-05    0.70    SRAM_16_2048_APACHECELL   sc2/wSr_inst/pmem/adsU1
  VDD    VDD_cmplx 7.143e+08     0.10         2.591e-05        2.515e-05       8.602e-06        7.52e-07      3.451e-05    0.70    SRAM_16_512_APACHECELL    sc2/wSr_inst/dmem/adsU1
  VDD    VDD_cmplx 7.143e+08     2.00             0            3.428e-06       9.531e-08       2.089e-05      2.441e-05    0.70    BUFx24_ASAP7_6t_SL        sc3/HFSBUF_4_137
  VDD    VDD_cmplx 7.143e+08     2.00             0             3.46e-06       9.531e-08       1.759e-05      2.115e-05    0.70    BUFx24_ASAP7_6t_SL        sc3/HFSBUF_4_164
  VDD    VDD_cmplx 7.143e+08     2.00             0            3.535e-06       9.531e-08       1.412e-05      1.775e-05    0.70    BUFx24_ASAP7_6t_SL        sc1/HFSBUF_4_137
  VDD    VDD_cmplx 7.143e+08     2.00             0            2.675e-07       8.527e-11       1.457e-05      1.483e-05    0.70    BUFx2_ASAP7_6t_R          sc1/wRf_inst/pmem/ClkBuf_inst_Lvl1_836_76_912_152
  VDD    VDD_cmplx 7.143e+08     2.00             0            2.675e-07       8.527e-11       1.457e-05      1.483e-05    0.70    BUFx2_ASAP7_6t_R          sc2/wRf_inst/pmem/ClkBuf_inst_Lvl1_836_76_912_152
  VDD    VDD_cmplx 7.143e+08     2.00             0            2.675e-07       8.527e-11       1.457e-05      1.483e-05    0.70    BUFx2_ASAP7_6t_R          sc3/wRfPF_inst/pmem/ClkBuf_inst_Lvl1_836_76_912_152
```

Static Voltage Drop Analysis

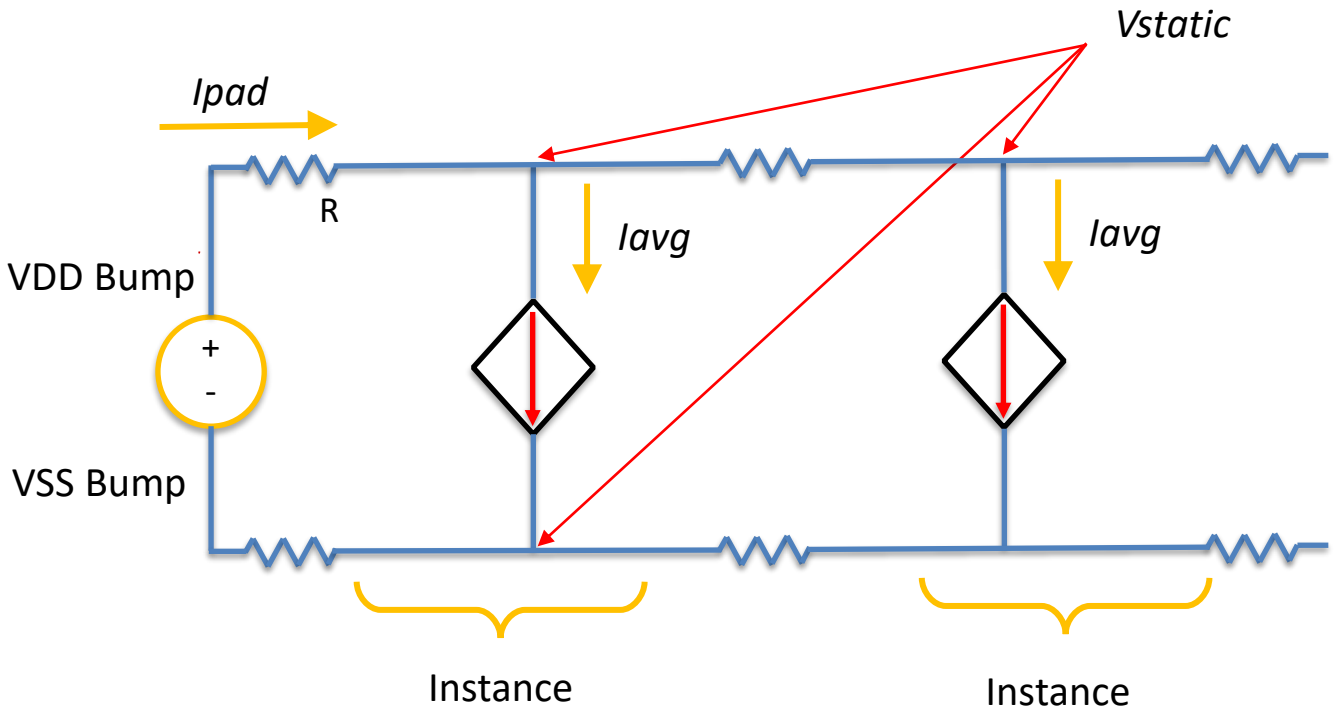
Static Voltage Drop Background



- On-chip power/ground network ➔ mesh of resistors
- Instances ➔ DC current sources

Static Voltage Drop on P/G Network

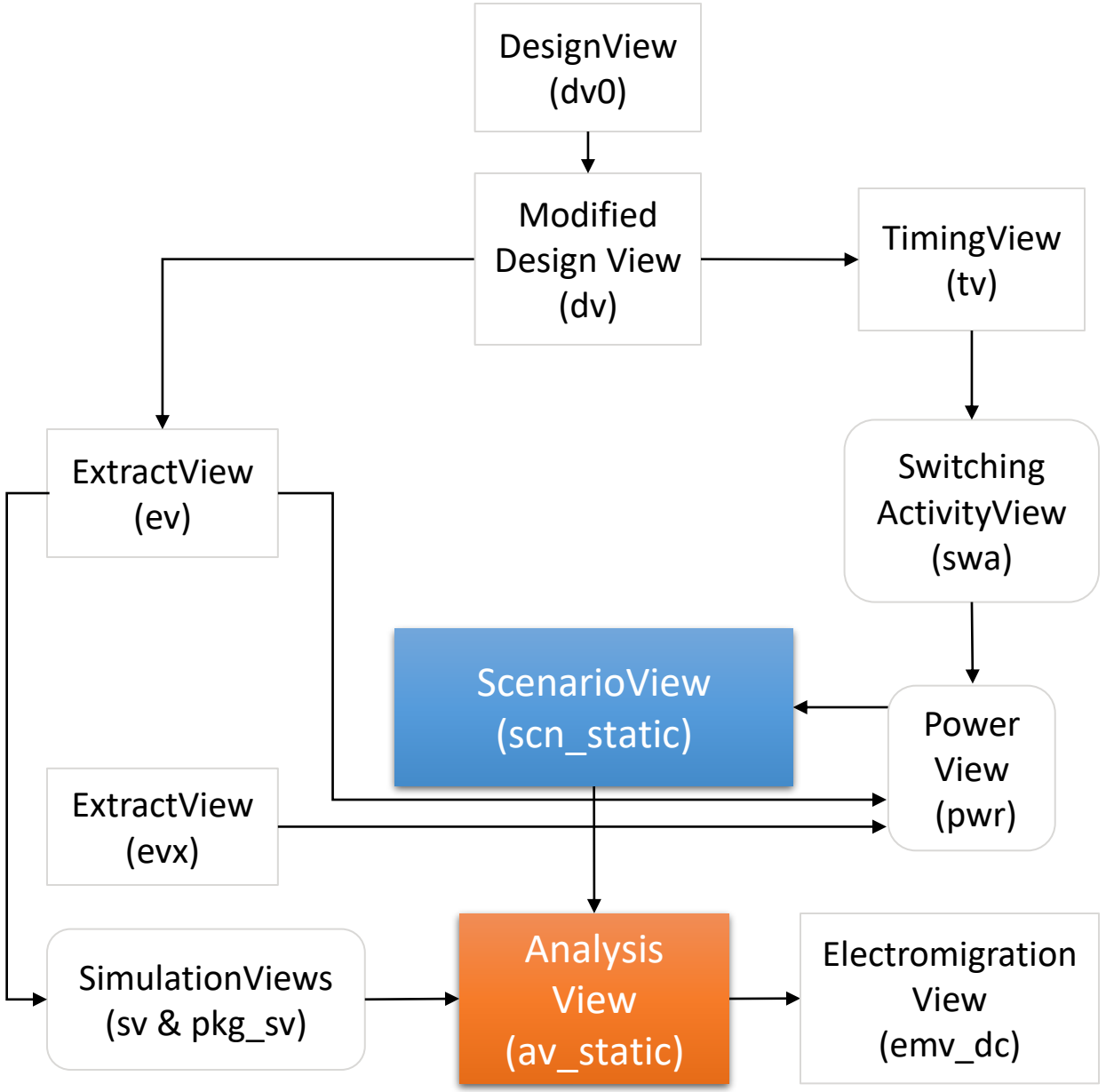
- Average current is calculated for each instance (using instance average power, $I_{avg} = P_{avg} / V_{supply}$)
- V_{static} is computed at every node (Ohm's law ...)
- Wire / via electromigration (*EM*) is post-processed from static current density



Performing Static Analysis

- **Static Scenario View**
  - Converts the power per instance to DC current per PG pin
  - For more details, see: **help(SeaScapeDB.create_scenario_view)**

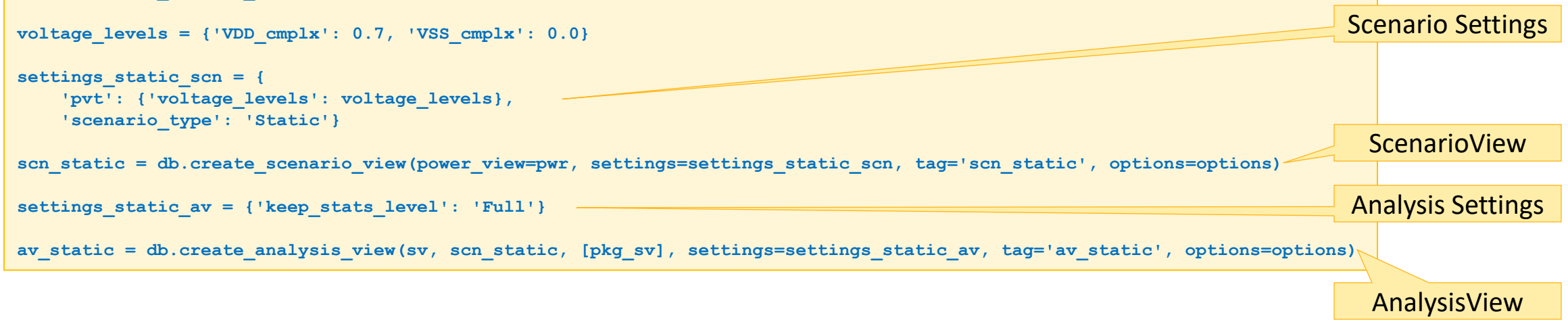
- **AnalysisView**
  - Holds results of simulation
  - Takes in SimulationView and ScenarioView as inputs
  - All voltage and current queries are available after simulation from AnalysisView
  - For more details, see: **help(SeaScapeDB.create_analysis_view)**



Preparing for static run

File: scripts/static.py

```
options = get_default_options()

voltage_levels = {'VDD_cmplx': 0.7, 'VSS_cmplx': 0.0}

settings_static_scn = {
    'pvt': {'voltage_levels': voltage_levels},
    'scenario_type': 'Static'}

scn_static = db.create_scenario_view(power_view=pwr, settings=settings_static_scn, tag='scn_static', options=options)

settings_static_av = {'keep_stats_level': 'Full'}

av_static = db.create_analysis_view(sv, scn_static, [pkg_sv], settings=settings_static_av, tag='av_static', options=options)
```



Looking At Static Analysis Reports

File: scripts/static.py

```
reports_dir = 'reports/static_analysis/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)

emir_reports.write_all_instance_voltages(av_static, reports_dir + 'inst_voltage.rpt')
emir_reports.write_bump_currents(av_static, reports_dir + 'bump_current.rpt')
emir_reports.write_bump_voltages(av_static, reports_dir + 'bump_voltage.rpt')
emir_reports.write_demand_currents(av_static, reports_dir + 'demand_current.rpt')
emir_reports.write_layer_voltage_report(av_static, reports_dir + 'layer_voltage.rpt')
emir_reports.write_node_voltage_report(av_static, reports_dir + 'node_voltage.rpt')
emir_reports.write_supply_currents(av_static, reports_dir + 'supply_currents.rpt')
emir_reports.write_switch_report(av_static, reports_dir + 'switch_report.rpt')
```

Static Analysis Reports available in "reports/static_analysis" directory

```
inst_voltage.rpt
demand_current.rpt
bump_current.rpt
bump_voltage.rpt
layer_voltage.rpt
node_voltage.rpt
supply_currents.rpt
switch_report.rpt
```

Looking At Static Analysis Reports

```
TitleText: Bump Currents
# Time (ps) I (A)
"pad_VDD_cmplx_1003.0_1007.0
         0.00     0.0000062

"pad_VDD_cmplx_1003.0_1015.0
         0.00     0.0000062

"pad_VDD_cmplx_1003.0_1023.0
         0.00     0.0000062

"pad_VDD_cmplx_1003.0_1031.0
         0.00     0.0000062

"pad_VDD_cmplx_1003.0_1039.0
         0.00     0.0000062
```

```
TitleText: Bump Voltages
# Time (ps) V (V)
"pad_VDD_cmplx_1003.0_1007.0
         0.00     0.6999955

"pad_VDD_cmplx_1003.0_1015.0
         0.00     0.6999955

"pad_VDD_cmplx_1003.0_1023.0
         0.00     0.6999955

"pad_VDD_cmplx_1003.0_1031.0
         0.00     0.6999955

"pad_VDD_cmplx_1003.0_1039.0
         0.00     0.6999955
```

File: bump_current.rpt / bump_voltage.rpt

```
# contents are sorted in descending order by layer, net, node_voltage
# loc_x      loc_y      layer      net      node_voltage
#  (u)        (u)                            (v)
 791.853    256.9555      M9     VSS_cmplx     0.0013
 791.853    256.8185      M9     VSS_cmplx     0.0013
 793.053    256.6815      M9     VSS_cmplx     0.0013
 791.853     256.75       M9     VSS_cmplx     0.0013
 793.053    256.5445      M9     VSS_cmplx     0.0013
 791.853    256.5445      M9     VSS_cmplx     0.0013
 793.053     256.75       M9     VSS_cmplx     0.0013
 793.053    256.8185      M9     VSS_cmplx     0.0013
 793.588     256.75       M9     VSS_cmplx     0.0013
```

File: node_voltage.rpt

```
TitleText: Demand Currents
# Time (ps) I (A)

"VDD_cmplx
         0.00     0.2221117

"VSS_cmplx
         0.00    -0.2221648

"sc4/wSrPgt_inst/openMSP430_pgt_inst/VDD_openmsp
         0.00     0.0006290

"sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/VDD_dbg
         0.00     0.0001463

"sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/dbg_uart_0/VDD_dbg_uart
         0.00     0.0001316

"sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/VDD_eu
         0.00     0.0002252

"sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/VDD_reg_file
         0.00     0.0005197

"sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult
         0.00     0.0002461
```

File: demand_currents.rpt

```
#  min_x     min_y    min_voltage    max_x     max_y    max_voltage   layer_drop   net          layer
#   (u)       (u)        (v)         (u)       (u)        (v)          (v)
 396.685   111.274      5e-06      285.093   93.294       0.01137      0.01136    VDD_cmplx    M2
  1.244    299.372      4e-06      285.102   93.294       0.01135      0.01134    VDD_cmplx    M3
   2.89    295.916      4e-06      285.102   93.51        0.01133      0.01133    VDD_cmplx    M4
   2.89    295.916      4e-06      285.102   83.05       0.004876      0.004872   VDD_cmplx    M5
   0.49     297.75      4e-06      285.102   83.05       0.003531      0.003527   VDD_cmplx    M6
 299.511    83.682     0.008979    284.391   91.458       0.01141      0.002436   sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult M1
 450.1533   82.21        7e-06     783.915   254.228     0.002414      0.002407   VSS_cmplx    M1
 396.685   111.274      5e-06      783.915   254.444     0.002406      0.002401   VDD_cmplx    M1
   0.49     299.75      4e-06      285.102   83.05       0.002188      0.002184   VDD_cmplx    M7
```

File: layer_voltage.rpt

```
TitleText: Supply Currents
# Time (ps) I (A)

"VDD_cmplx
         0.00     0.2221117

"VSS_cmplx
         0.00    -0.2221648
```

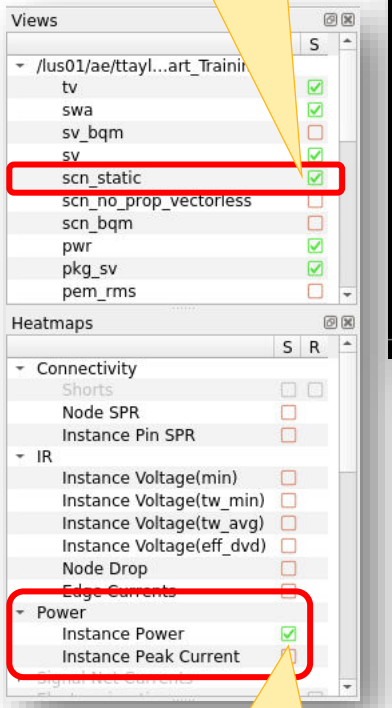
File: supply_currents.rpt

```
# loc_x      loc_y     min_instance_static_voltage   pg_arc        instance
#  (u)        (u)                 (v)               pwr/gnd
 284.958    91.242              0.6879             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/cto_inv_drc_5819
 284.958    91.674              0.6879             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/cto_inv_drc_5818
 286.362    91.674              0.688              sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/cto_inv_drc_5817
 284.742    91.674              0.688              sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/_1258_
 284.958    91.674              0.688              sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/HFSINV_34_396
 285.768    91.674              0.6881             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/_1397_
 285.552    91.242              0.6881             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/ZBUF_20_inst_3984
 284.958    92.106              0.6881             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/ZBUF_262_inst_721
 280.908    93.402              0.6881             sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/VDD_mult/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/multiplier_0/HFSBUF_5102_7
```

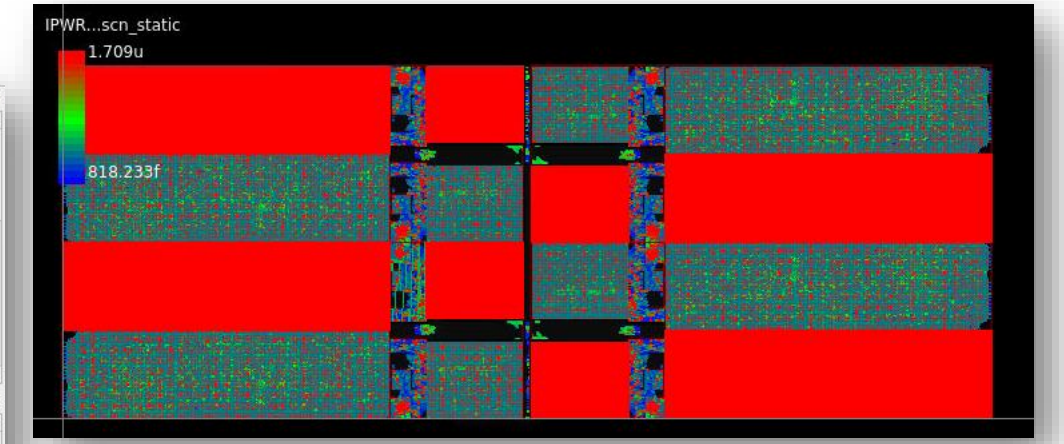
File: inst_voltage.rpt

GUI – Instance Power Map

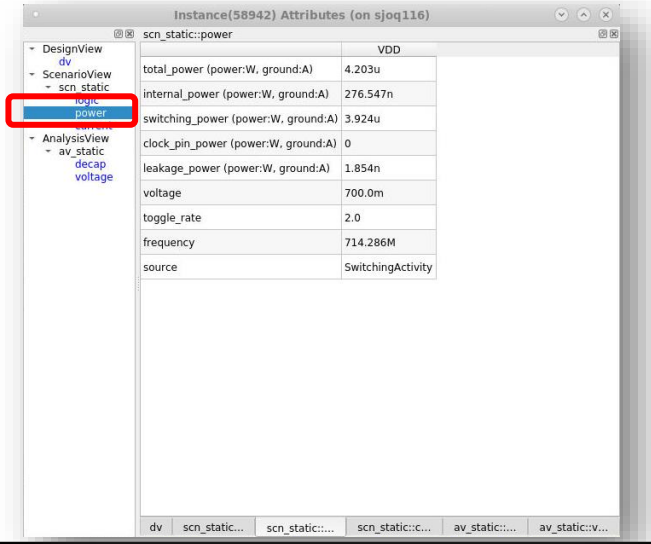
Select Static SCN view



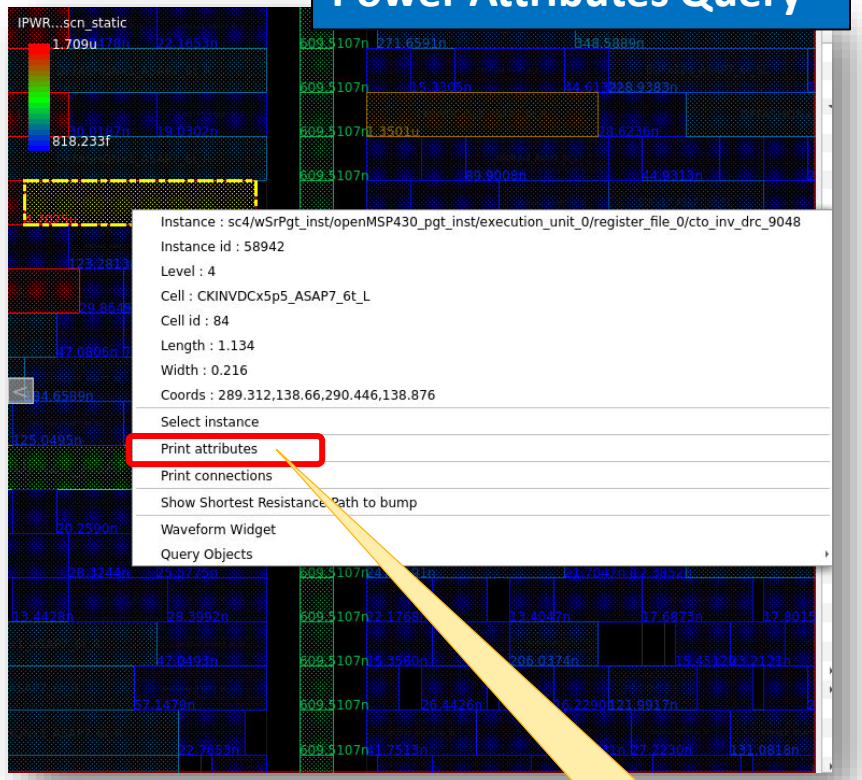
Enable "Instance Power"



Instance Power Map



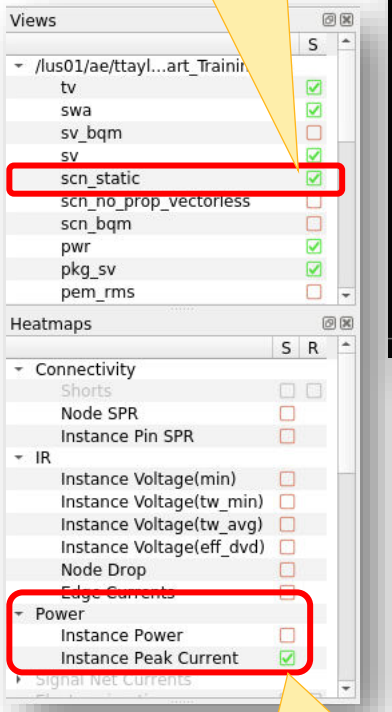
Power Attributes Query



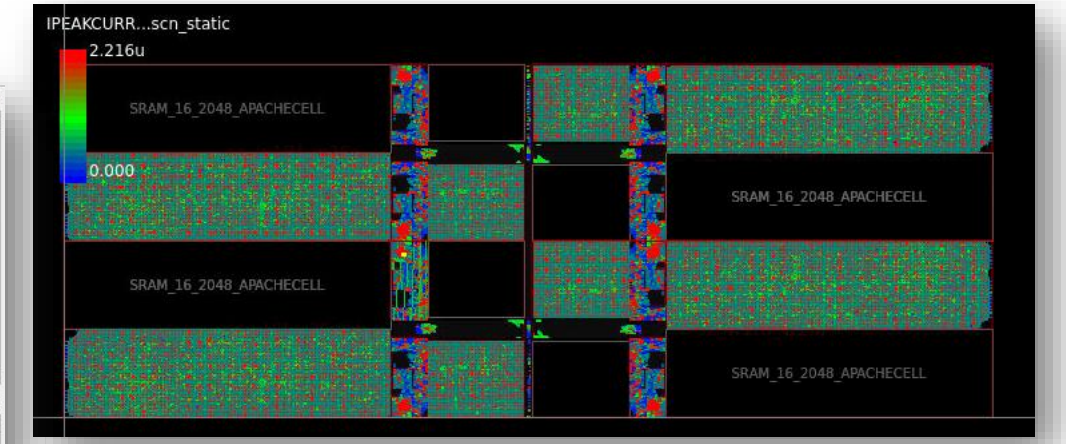
Left click to select instance, right click to bring up menu, and left click on "Print attributes"

GUI – Instance Peak Current Map

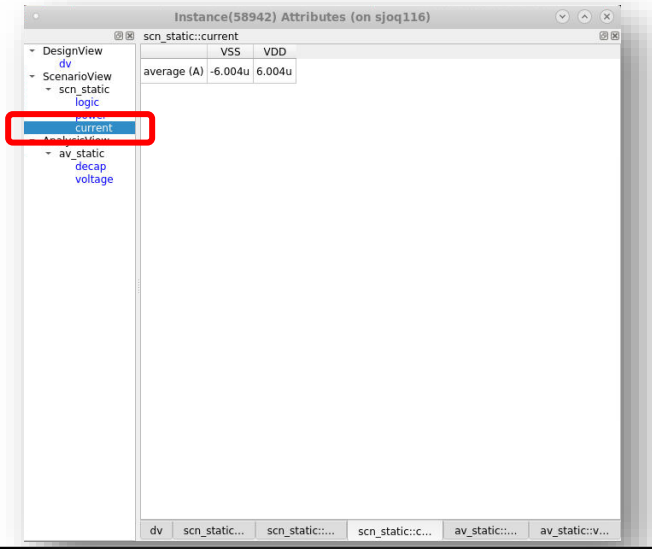
Select Static SCN view



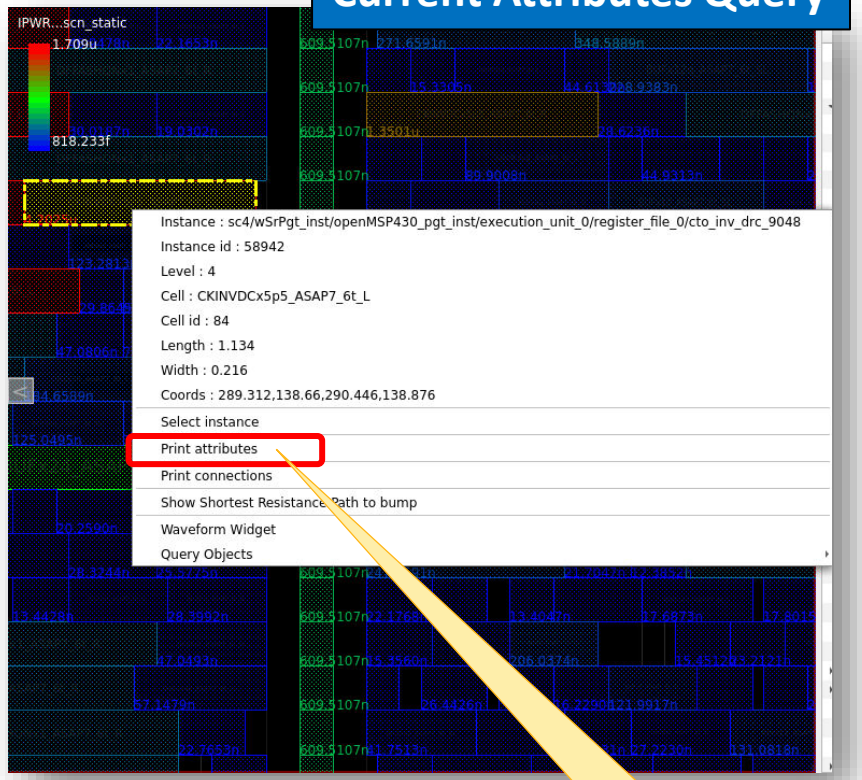
Enable "Instance Peak Current"



Instance Peak Current Map



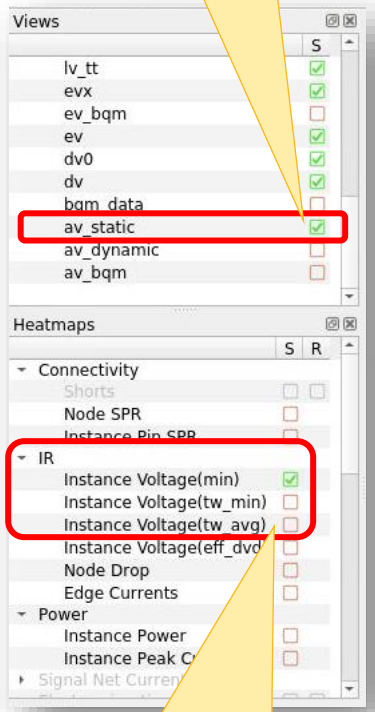
Current Attributes Query

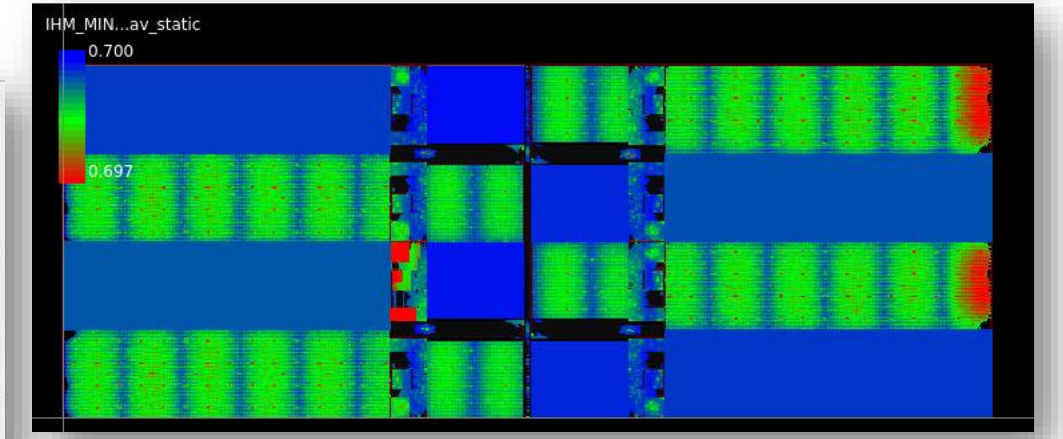


Left click to select instance, right click to bring up menu, and left click on "Print attributes"

GUI – Instance/Node Voltage Maps

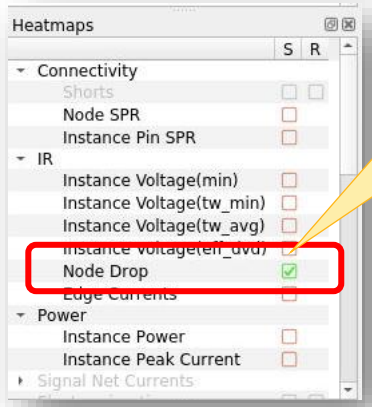
Select Static AV view



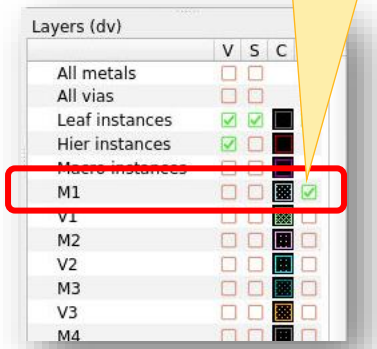


Instance Voltage Drop Map

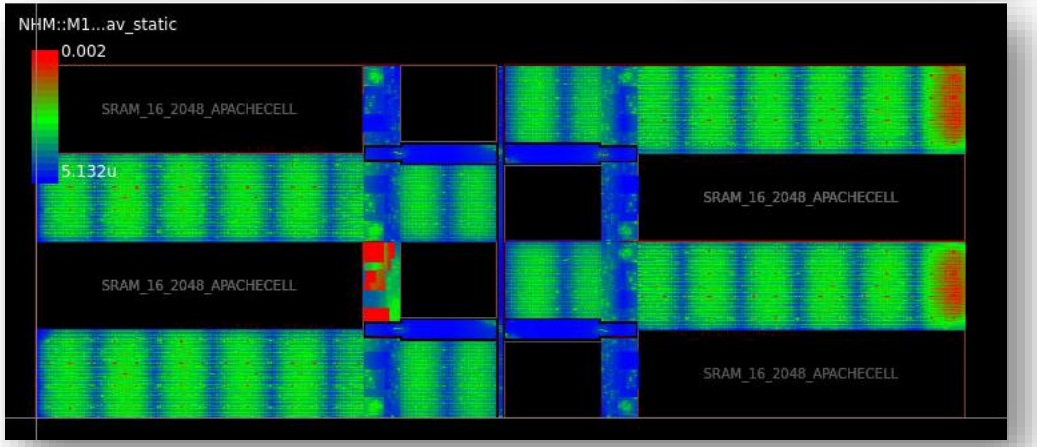
Enable "Node Voltage" Heatmap



Select metal layer



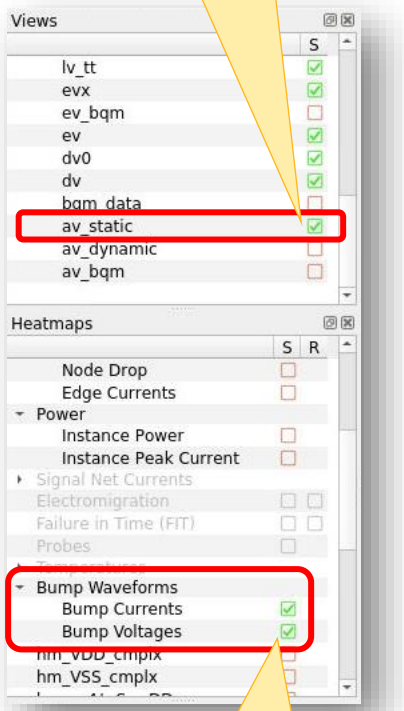
Enable one of
"Instance DVD (min/tw_min/tw_avg)"
[Do not select eff_dvd - empty]



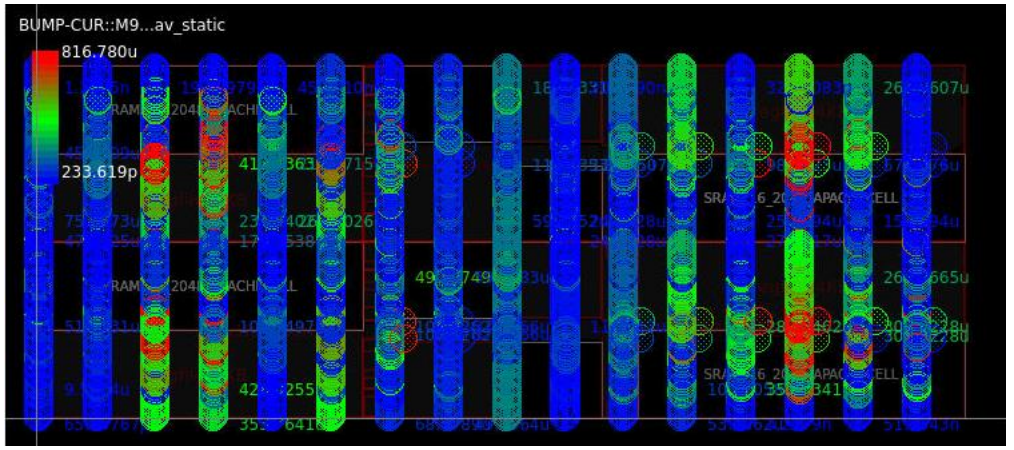
Node Voltage Drop Map (M1)

GUI – Bump Current/Voltage Maps

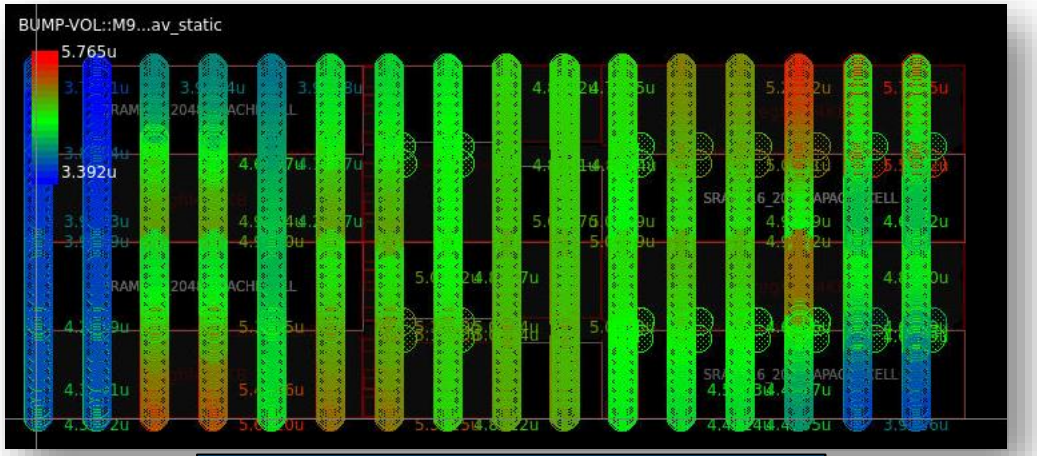
Select Static AV view



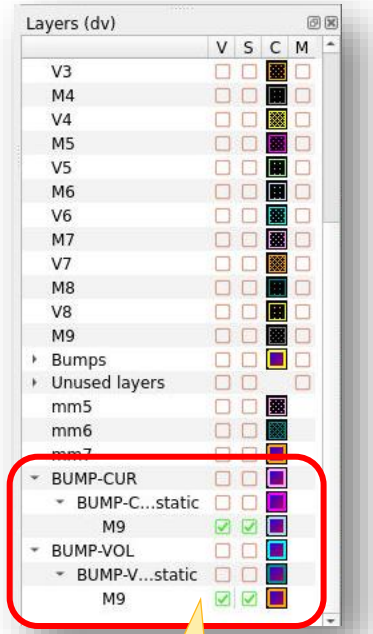
Enable "Bump Waveforms" Heatmaps



Bump Current Map



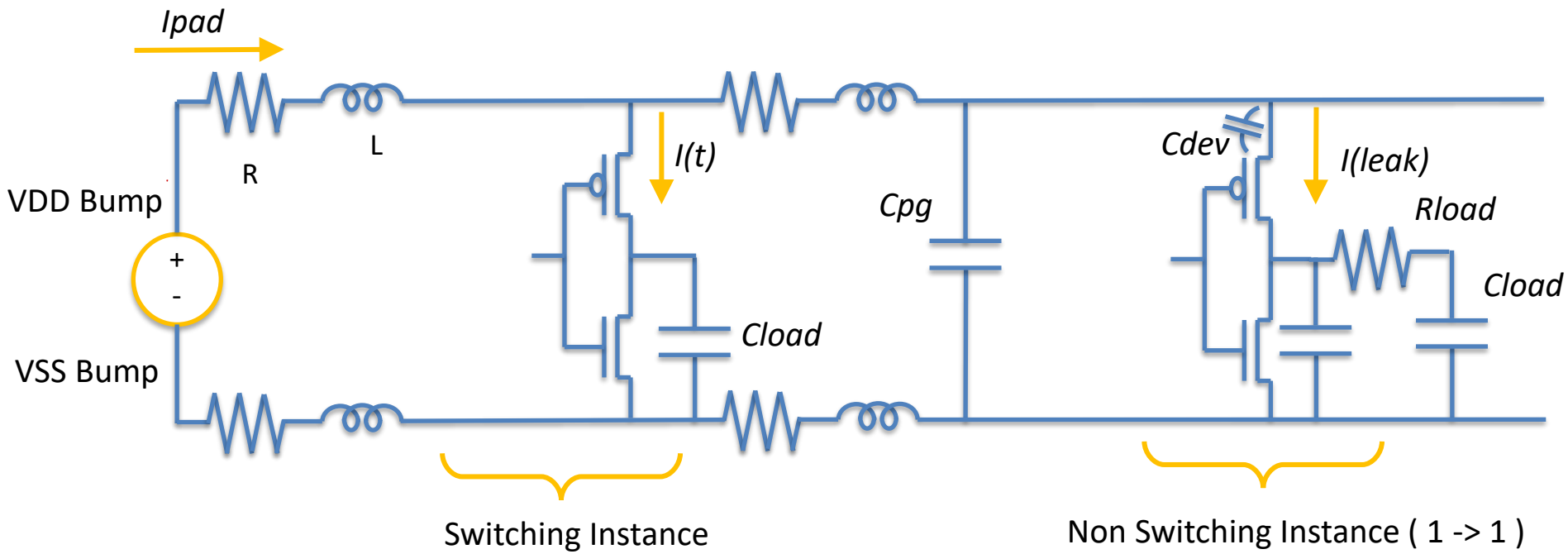
Bump Voltage Map



Select Bump Map (Voltage or Current)

Dynamic Voltage Drop Analysis

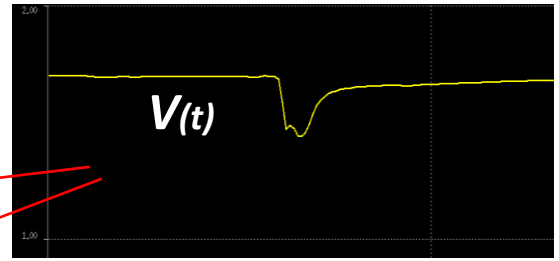
Dynamic Voltage Drop Problem Definition

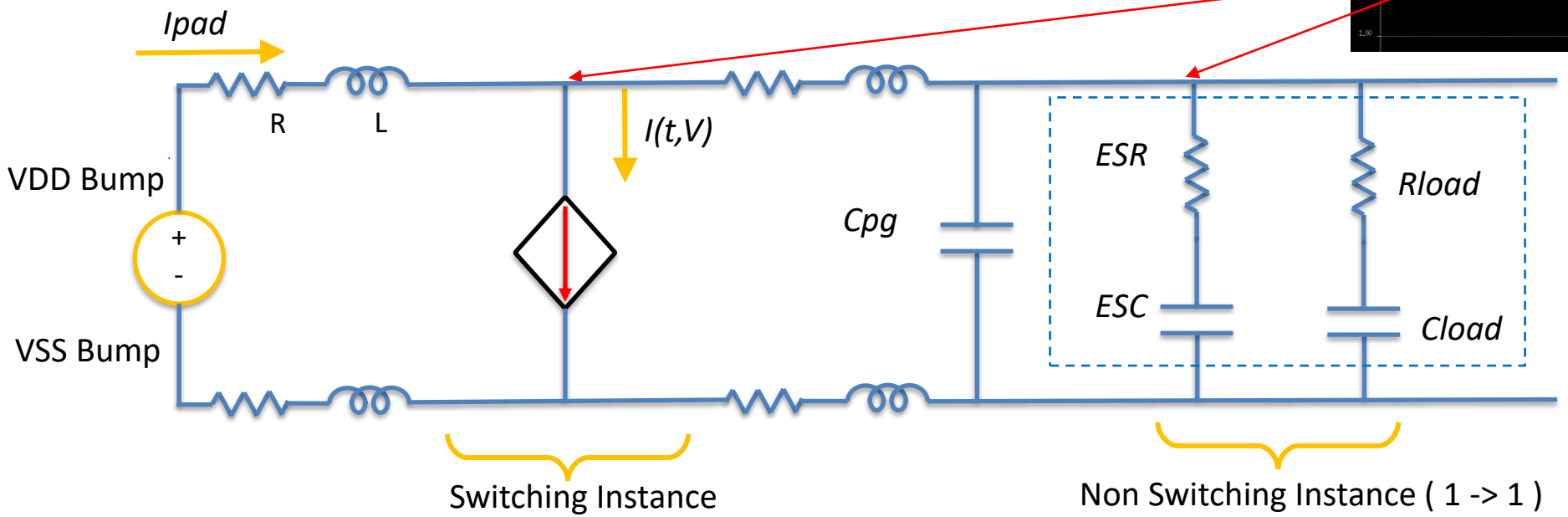


- On-chip power/ground network ➔ R,L,C mesh
- Switching instances ➔ i(t,V) sources
- Non-switching instances ➔ Effective decaps, ESR and leakage

Dynamic Voltage Drop Problem Definition

- PWL current for each instance
- $V_{dynamic}$ waveform is computed at every node by transient simulation
- Intrinsic resistance ESR and Intrinsic capacitance ESC from APL or CCS-P current / cap models



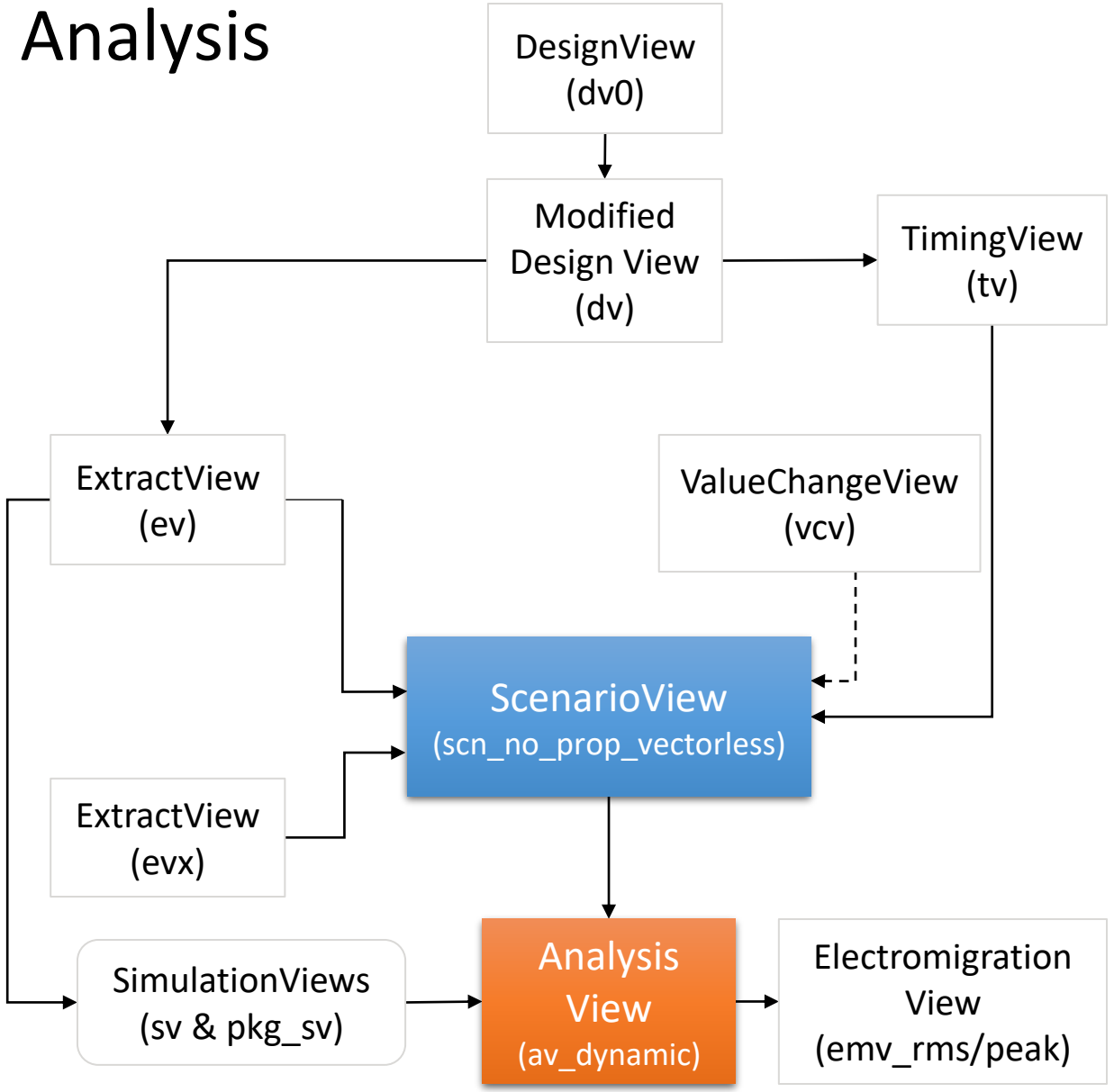


- On-chip power/ground network ➔ R,L,C mesh
- Switching instances ➔ i(t,V) sources
- Non-switching instances ➔ Effective decaps, ESR and leakage

Performing Dynamic/Transient Analysis

- **Dynamic Scenario View**
  - Calculates the currents per PG pin
  - Several algorithms available to support vectorless and vectored
  - For more details, see:
    help(SeaScapeDB.create_scenario_view)
    help(SeaScapeDB.create_no_prop_scenario_view)

- **AnalysisView**
  - Holds results of simulation
  - Takes in SimulationView and ScenarioView as inputs
  - All voltage and current queries are available after simulation from AnalysisView
  - For more details, see:
    help(SeaScapeDB.create_analysis_view)



Different Types of Dynamic Scenarios

Vector Input

- RTL VCDs/FSDB's: Some part of design populated by tool
- Gate VCDs – SDF annotated and unannotated

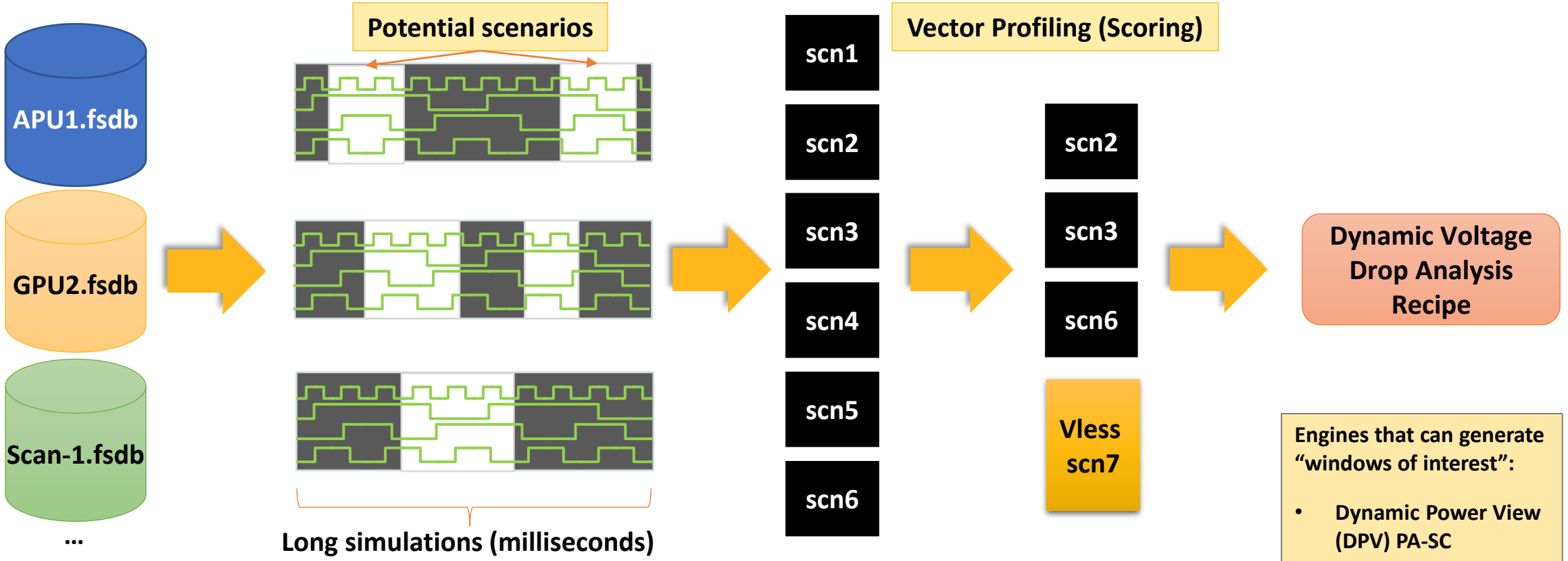
Vectorless

- Logic Propagation based, Power Controlled (PCVS)
- No Propagation Vectorless (NPV)
- Activity or Power driven

Special Flows

- Scan shift modes for BIST, MBIST, ATPG test operations
- Can be VCD based or Vectorless
- Power Transient mode in vectorless analysis for high DI/DT scenario

Choosing Scenarios – Mix Of Vector + Vectorless Approaches



Engines that can generate "windows of interest":

- Dynamic Power View (DPV) PA-SC
- Profile Power View (RH-SC)

Importance of Vectorless Approaches

Simulation vectors are often insufficient

- May be available too late or not at all
- Functional vectors may not be best to identify voltage drop problems
- Even good vectors will be incomplete and need to be augmented for full coverage

Can we "synthesize" vectors for use in dynamic analysis?

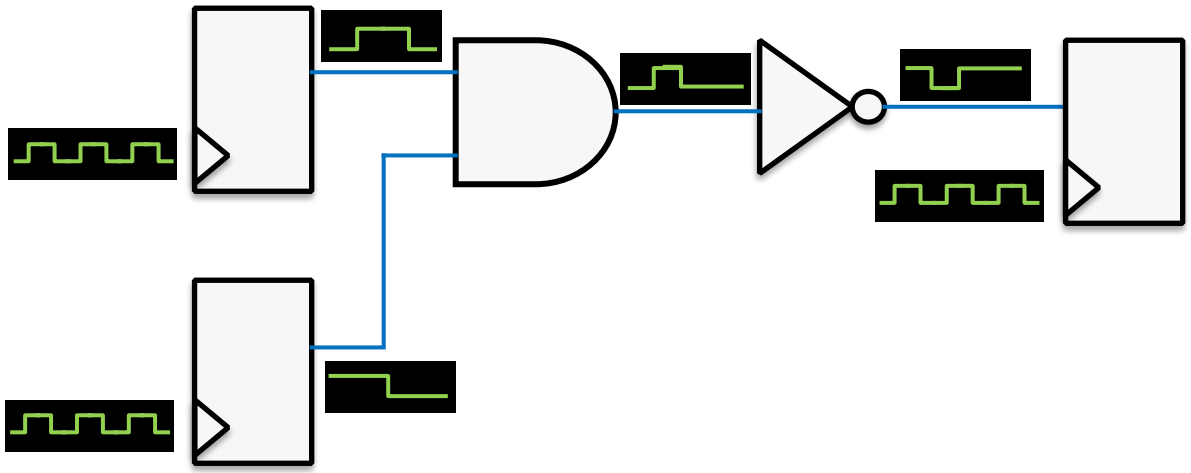
- This is known as vectorless analysis since the user doesn't need to supply vectors

What kind of vectorless approaches are available in SC ?

- Logic propagation based vectorless approaches
- Non-Logic based vectorless approaches (NPV)

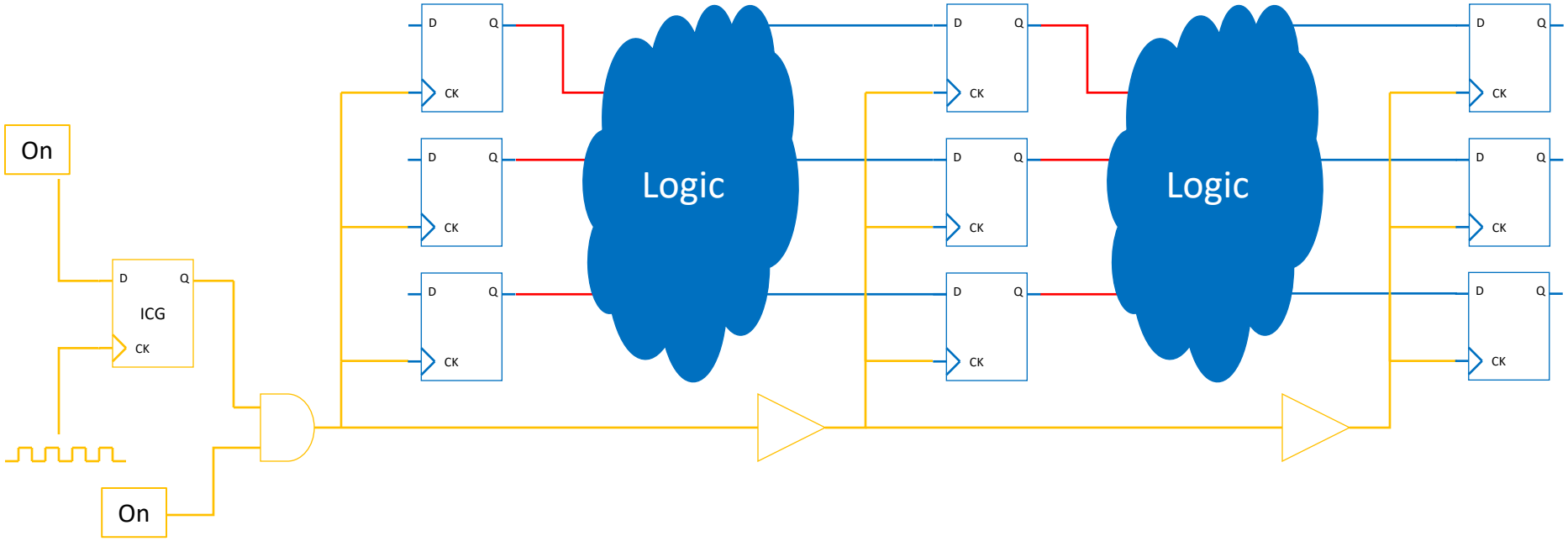
Logic (Event) Propagation: Vectorless

- Similar in concept to a gate-level logic simulator with full timing data
- Clock network is propagated first, followed by data paths and input ports
- Data paths are propagated from flip-flop (or macro) outputs where starting signal is derived from:
  - RTL VCD/fsdb data if available
  - Random switching based on user settings
  - Toggle Rate data in PowerView (on register/sequential elements only)
- Events propagate through combinational logic
- Uses cell delay and logic function information and net delay
  - This makes the activity more realistic and takes into account logical and temporal correlation.



Logic Propagation

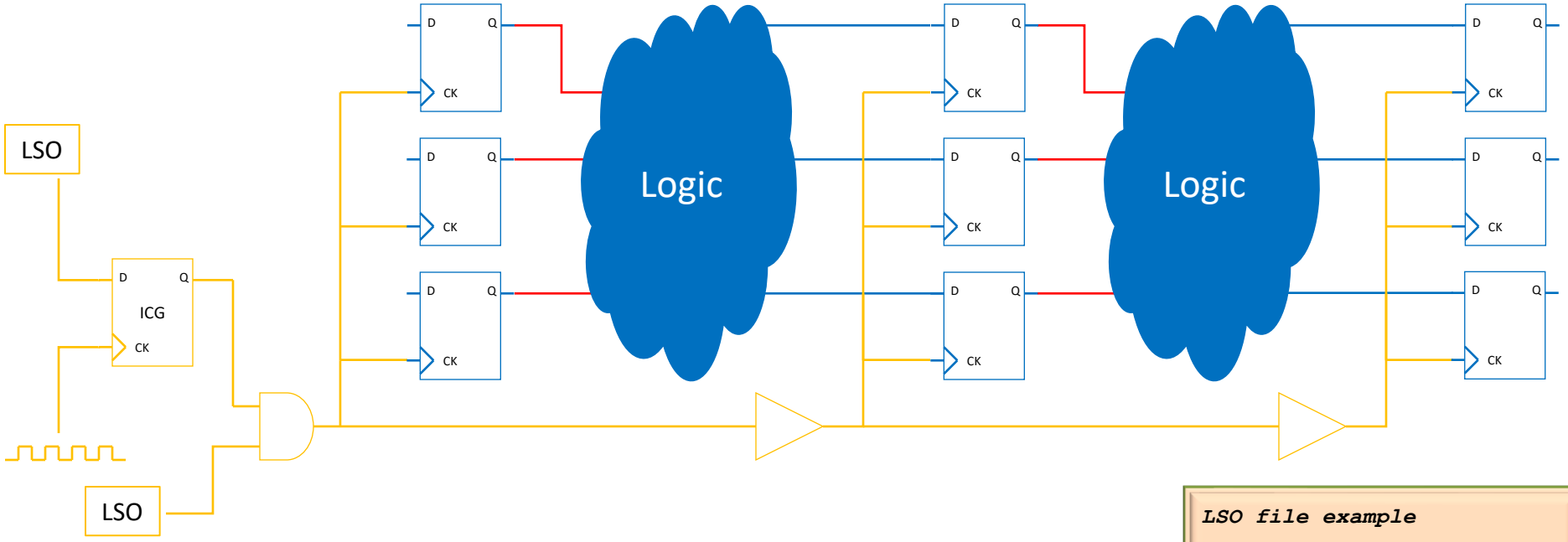
Example: Activity = 15% for data, 100% for clock



1. Seed all the clock start points and propagate the clock network
2. At start – completely randomize all state points (Q outputs).
3. Roll dice to see if state points will switch (every Q output has a 15% probability of switching each cycle).
4. When output switches, logic propagation occurs through the logic all the way to the D input of the next sequential element
5. Repeat 3 & 4 for each clock cycle for duration of simulation

Logic Propagation – ICGs

Example: Activity = 15% for data, 100% for clock

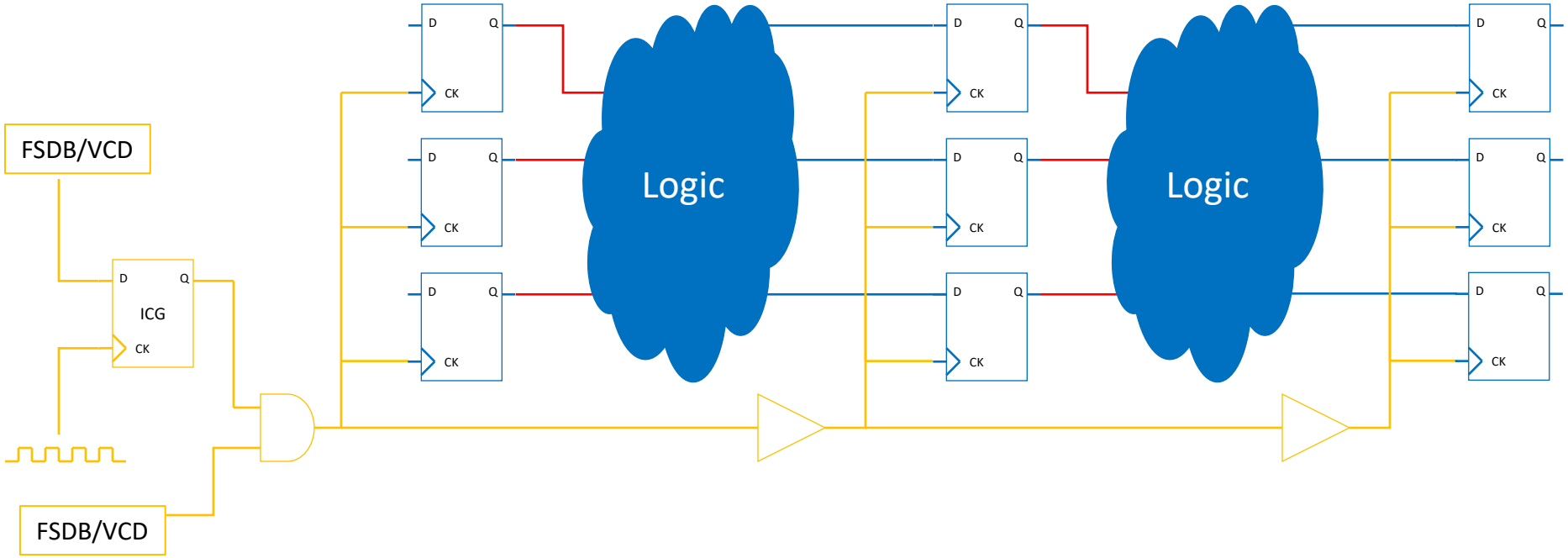


- ICGs and clock-gates are recognized and by default fully on
- User can override by supplying custom controls through LSO file

LSO file example

```
# Logic signal override
$version = 1.0
$time_unit = 1.0e-9

@ip U28/D {
   (0.0, 0)
   (0.251, 0)
   (0.279, 1)
}

@ip U29/B {
   (0.0, 0)
   (0.450, 0)
   (0.480, 1)
}
```

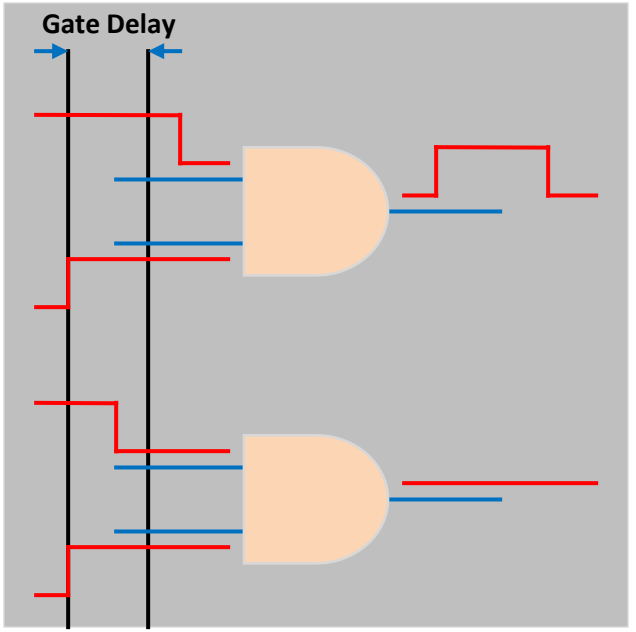
Logic Propagation with RTL FSDB/VCD



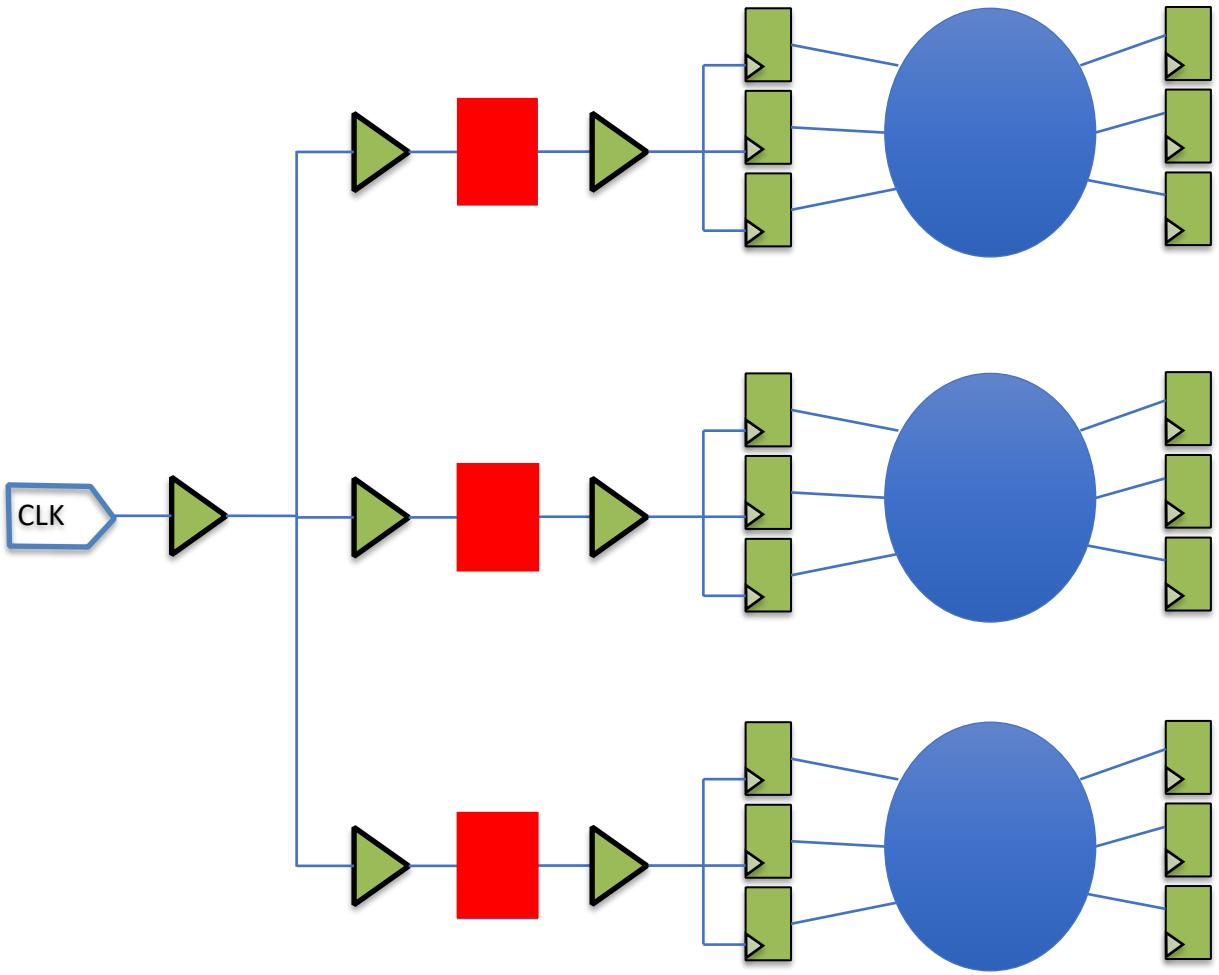
- FSDB/VCD applied to matching points
- For any sequential element not specified, the random assignment applies
- When point switches, logic propagation occurs through the logic all the way to the D input of the next sequential element
- RH-SC can infer ICG activity based on RTL states being constant for all elements controlled by a single ICG

Glitches

- Glitch events get propagated
- Will see logic and current waveforms reflect any glitch events
- What gets propagated?
  - A glitch is propagated if > cell delay
  - Suppressed if < cell delay
- If a glitch is not propagated:
  - If there are current tables (CCSP/APL) or internal_power tables (NLPM), currents are calculated for the relevant input_only events
  - Otherwise, no current is generated
  - Glitch control
    - There is an option to control
    - Will be replaced with an argument in a subsequent release
  - Glitch power
    - Dynamic Power View (DPV) provides breakdown of glitch power – separate training module



Power-Constrained Vectorless Scenario (PCVS)



- **Propagated, logically consistent scenario**
- **Delay-aware propagation**
- **User-specified target power**
  - Top or Top + Block
  - Meets power by enabling/disabling ICGs, then by flop toggle rate
  - Supports mixed flow where some blocks have gate-level VCD or FSDB
- **Supports Power Transient Analysis**
  - Allows different target power for different intervals of scenario_duration
- **Maximizes ICG coverage**
  - Different ICGs enabled per frame

PCVS is the only vectorless engine in industry that can perform the complex task of meeting logic and power together

Features of Power constrained Vectorless Scenario

- Generate dynamic scenarios based on target power
  - In a frame, a set of ICG are turned on such that the resulting power is close to the user given power target
  - Further, toggle rates of flop/memory can also be tweaked if controlling ICGs is not sufficient to achieve the target power (user controllable)
- Generated scenario is logically consistent
  - Create events on sequential cells w.r.t clock arrival time
  - Events generated at sequential cells are propagated through the downstream combinational logic
- Maximum ICG coverage
  - Select different sets of ICGs in different frames thereby maximizing the number of ICGs that are active
- Also honors user defined toggle rates at state points

No-Propagation Vectorless (NPV) Scenario

Overview

- Switching of each instance is directly determined without logic propagation
- Events are created per instance, rather than propagated from flip-flops

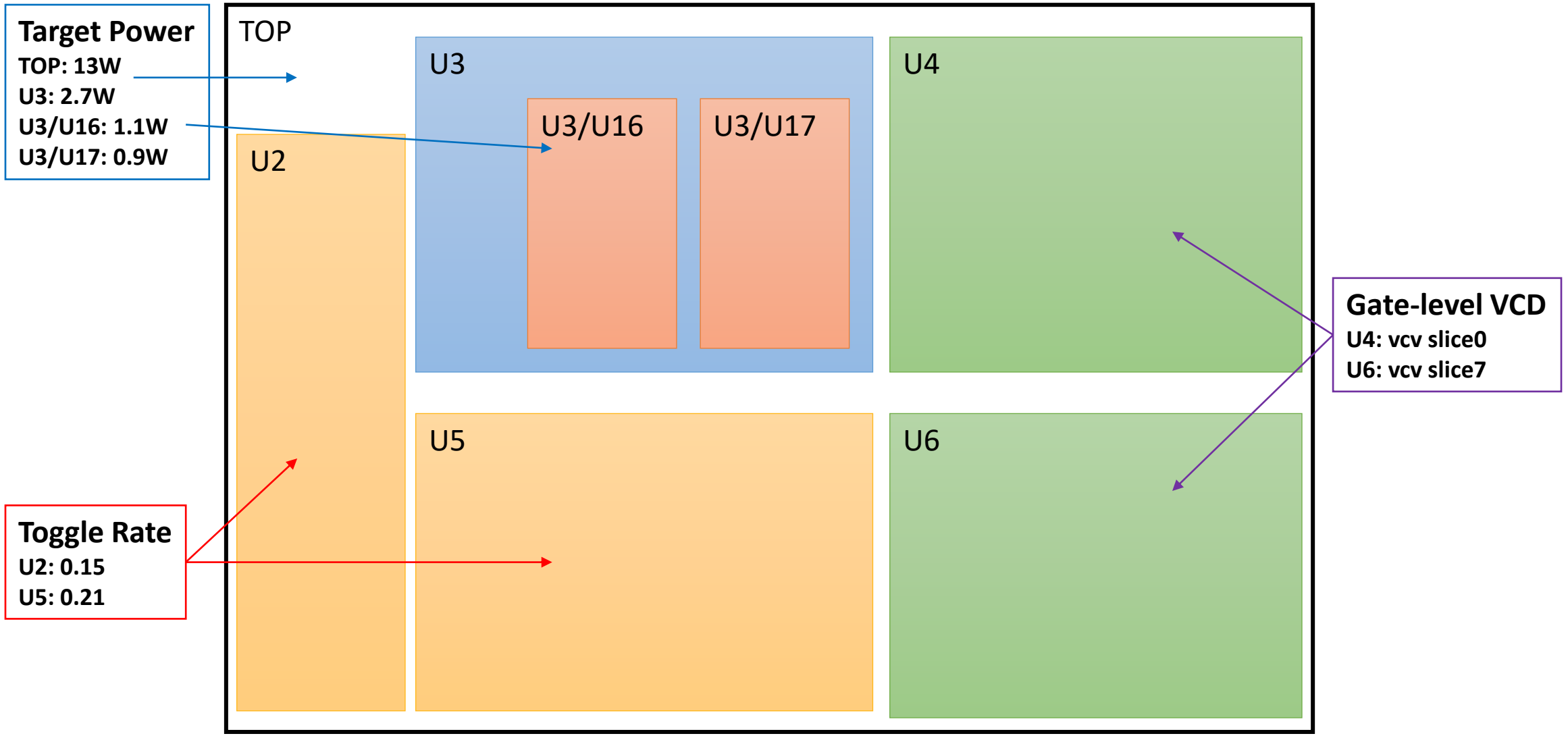
Benefits

- Very high coverage in short simulation duration
- Highly scalable/parallel - very efficient in memory and runtime
- Actual power is very close to the user-specified target power

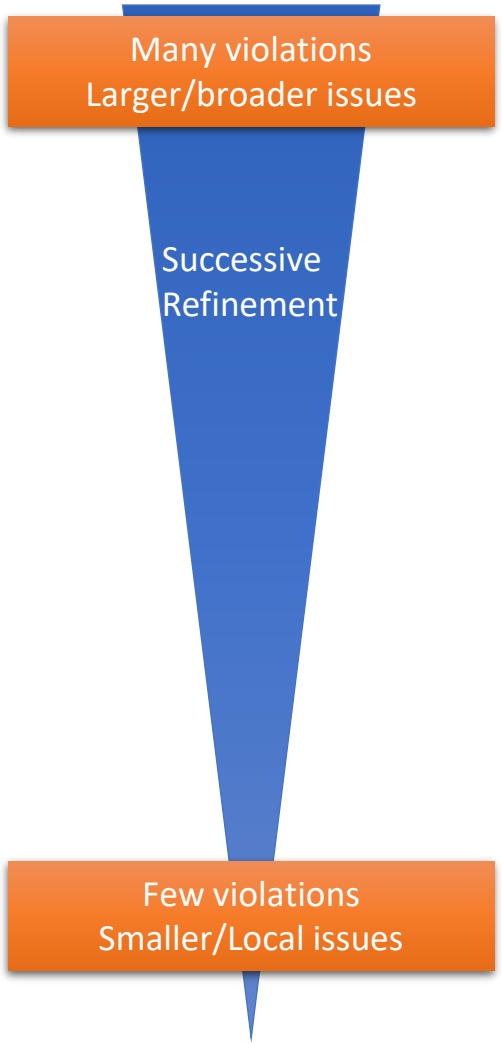
Features

- User can specify toggle rates or target power per block
- High-coverage mode to prioritize instances that haven’t switched yet
- Macro controls: user-specified mode probabilities or sequence
- Optional random event times within timing window

Hybrid Flow with NPV + Gate VCD



Convergent Prevention/Repair Flow for Voltage Drop



BQM: Build Quality Metric

- Measure relative weakness per PG domain across entire power grid
- Run from power planning through final implementation to ensure consistent resistance gradients as design matures

PeakTW

- Find placement issues by modeling possible peak current due to local simultaneous switching
- Run from early placement onward to identify issues as design is closed
- Merge with BQM results to determine a hazard metric for every instance PG pin

PCVS: Power-Constrained Vectorless Scenario

- Create scn for specific operating mode(s) with realistic propagated switching
- Run transient AnalysisView (typically 10 to 100ns duration each)
- Recommended for block level run's signoff
- Optional: analyze timing effect of voltage drop

No-Prop Vectorless Scenario

- Switch almost all instances in relatively few clock cycles
- Run transient AnalysisView with this scenario and fix outliers
- Provides excellent frequency domain power coverage for long duration simulations with full chip + package response

Preparing for Dynamic Run: No Propagation Vectorless Mode (NPV)

File: scripts/dynamic.py

```
options = get_default_options()

voltage_levels = {'VDD_cmplx': 0.7, 'VSS_cmplx': 0.0}
clock_period = 1.4e-09
clock_toggle_rate = 2.0
data_toggle_rate = 0.1
frame_length = 2 * clock_period
scenario_duration = 3 * frame_length
analysis_duration = scenario_duration - 0.1e-09

object_settings_npv = {
    'design_values': {
        'clock_pin_toggle_rate': clock_toggle_rate,
        'combinational_pin_toggle_rate': data_toggle_rate,
        'mode_control': {
            'mode_sequence': ['read', 'write', 'standby_trig']},
        'sequential_output_pin_toggle_rate': data_toggle_rate}}

settings_npv = {
    'pvt': {'voltage_levels': voltage_levels},
    'object_settings': object_settings_npv,
    'frame_length': frame_length,
    'event': {'default_clock': {'policy': 'custom', 'period': clock_period}}}

scn_no_prop_vectorless = db.create_no_prop_scenario_view(power_view=pwr, timing_view=tv, external_parasitics=evx, extract_view=ev, scenario_duration=scenario_duration,
settings=settings_npv, tag='scn_no_prop_vectorless', options=options)

settings_dynamic_av = {
    'duration': analysis_duration,
    'step_size': 7e-12,
    'keep_stats_level': KeepStats('Full', timing_window_stats_mode='sliding_window')}

av_dynamic = db.create_analysis_view(sv, scn_no_prop_vectorless, [pkg_sv], settings=settings_dynamic_av, tag='av_dynamic', options=options)
```

Variables

No Propagation Scenario Object Settings

No Propagation Scenario Settings

ScenarioView

Analysis View Settings

Analysis View

Dynamic Analysis Reports

File: scripts/dynamic.py

```
reports_dir = 'reports/dynamic_analysis/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)

emir_reports.write_instance_power_report_and_summary(scn_no_prop_vectorless, reports_dir + 'instance_power.rpt', reports_dir +
'power_summary.rpt')
emir_reports.write_all_instance_voltages(av_dynamic, output_file = reports_dir + '/inst_voltage_rpt', sort_order='ascending')
emir_reports.write_bump_currents(av_dynamic, reports_dir + 'bump_current.rpt')
emir_reports.write_bump_voltages(av_dynamic, reports_dir + 'bump_voltage.rpt')
emir_reports.write_demand_currents(av_dynamic, reports_dir + 'demand_current.rpt')
emir_reports.write_layer_voltage_report(av_dynamic, reports_dir + 'layer_voltage.rpt')
emir_reports.write_node_voltage_report(av_dynamic, reports_dir + 'node_voltage.rpt')
emir_reports.write_supply_currents(av_dynamic, reports_dir + 'supply_currents.rpt')
emir_reports.write_switch_report(av_dynamic, reports_dir + 'switch_report.rpt')
```

Dynamic Analysis Reports available in "reports/dynamic_analysis" directory

```
instance_power.rpt
power_summary.rpt
inst_voltage.rpt
demand_current.rpt
bump_voltage.rpt
bump_current.rpt
layer_voltage.rpt
node_voltage.rpt
supply_currents.rpt
switch_report.rpt
```

Looking At Dynamic Analysis Reports

```
TitleText: Bump Currents
# Time (ps) I (A)
"pad_VDD_cmplx_1003.0_1007.0
         0.00     -0.0000771
         7.00     -0.0000795
        14.00     -0.0000846
        21.00     -0.0000922
        28.00     -0.0001008
        35.00     -0.0001144
        42.00     -0.0001283
        49.00     -0.0001432
        56.00     -0.0001582
        63.00     -0.0001702
```

```
TitleText: Bump Voltages
# Time (ps) V (V)
"pad_VDD_cmplx_1003.0_1007.0
         0.00      0.6999937
         7.00      0.6999761
        14.00      0.6999369
        21.00      0.6998568
        28.00      0.6997347
        35.00      0.6995986
        42.00      0.6994500
        49.00      0.6992877
        56.00      0.6991166
        63.00      0.6989490
```

File: bump_current.rpt / bump_voltage.rpt

```
# contents are sorted in descending order by layer, net, node_voltage
# loc_x      loc_y      layer      net      node_voltage
#  (u)        (u)                           (v)
 791.853    260.9555      M9     VSS_cmplx    0.01498
 791.853    260.8185      M9     VSS_cmplx    0.01498
 793.053    260.6815      M9     VSS_cmplx    0.01498
 791.853     260.75       M9     VSS_cmplx    0.01498
 791.853    260.5445      M9     VSS_cmplx    0.01498
 793.053    260.5445      M9     VSS_cmplx    0.01497
 793.053     260.75       M9     VSS_cmplx    0.01497
 791.853    260.6815      M9     VSS_cmplx    0.01497
 793.053    260.8185      M9     VSS_cmplx    0.01497
 793.588     260.75       M9     VSS_cmplx    0.01497
```

File: node_voltage.rpt

File: demand_currents.rpt

```
TitleText: Supply Currents
# Time (ps) I (A)

"VDD_cmplx
         0.00      0.2130301
         7.00      0.2130344
        14.00      0.2130558
        21.00      0.2131195
        28.00      0.2132674
        35.00      0.2135616
        42.00      0.2140842
        49.00      0.2149364
        56.00      0.2162336
        63.00      0.2180994
```

File: supply_currents.rpt

```
TitleText: Demand Currents
# Time (ps) I (A)

"VDD_cmplx
         0.00      0.2130301
         7.00      0.2406948
        14.00      0.2718291
        21.00      0.3149632
        28.00      0.3670722
        42.00      0.4604040
        49.00      0.5129123
        56.00      0.5591626
        63.00      0.5947831
```

```
#  min_x     min_y     min_voltage    max_x     max_y     max_voltage   layer_drop   net          layer
#   (u)       (u)         (v)          (u)       (u)         (v)          (v)
 310.412   267.344      0.00302     512.9813   83.466      0.07283      0.06981    VSS_cmplx    M1
 310.326   267.344      0.003016    512.9813   83.466      0.06022      0.05721    VSS_cmplx    M2
  483.32    32.94       0.003373    512.9843   83.682      0.05906      0.05568    VDD_cmplx    M1
 308.819   114.968      0.003637    298.964    108.056     0.05764      0.05401    sc4/wSrPgt_inst/openMSP430_pgt_inst/VDD_openmsp M1
 293.1475  116.688      0.003962    282.2125    114.96     0.05751      0.05355    sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/VDD_dbg M1
 290.743   133.908      0.01125     281.698    135.204     0.06467      0.05342    sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/VDD_reg_file M1
 328.105   267.684      0.002408    513.5813   83.466      0.0555        0.0531    VSS_cmplx    M3
```

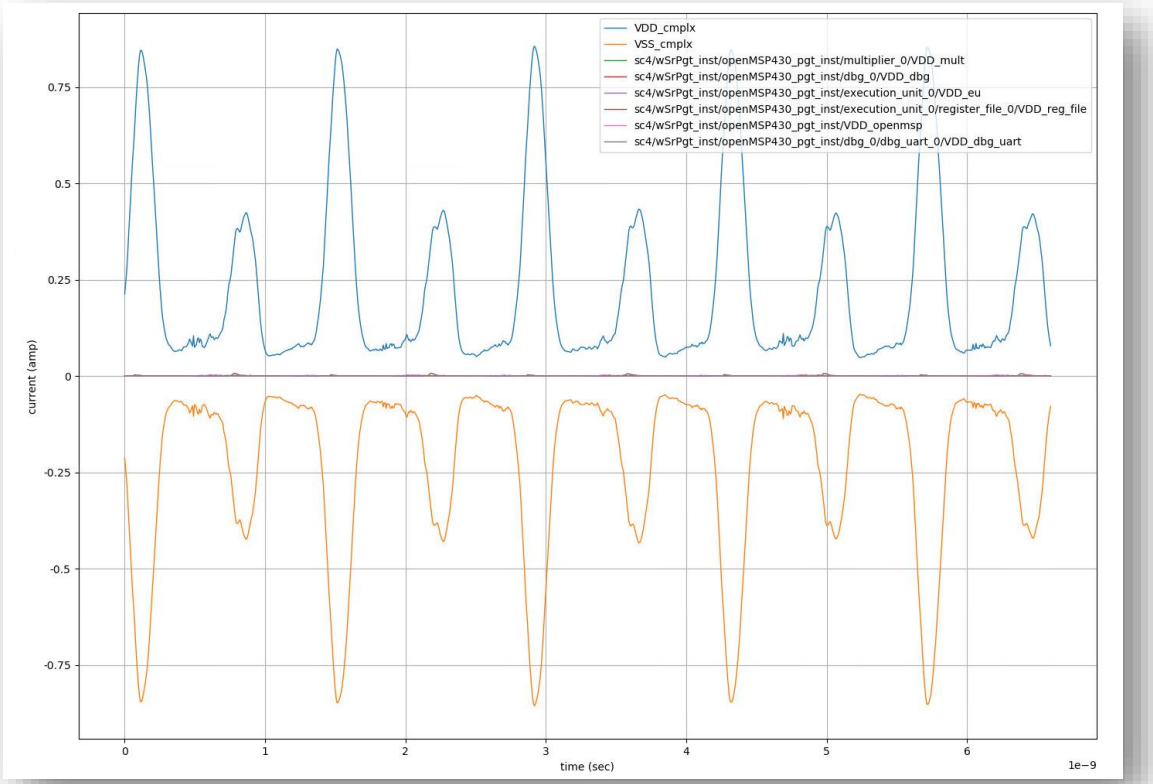
File: layer_voltage.rpt

```
# loc_x     loc_y     eff_Vdd   max_pg_tw  min_pg_tw  min_pg_sim  max_pg_sim  min_vdd_tw  max_vss_tw   pg_arc      instance
#  (u)       (u)       (v)       (v)        (v)        (v)         (v)         (v)         (v)        pwr/gnd
 512.1743   83.466    0.6613    0.6989     0.6385     0.5681      0.7064      0.6731      0.03456    VDD_cmplx/VSS_cmplx sc2/wRf_inst/openMSP430_inst/multiplier_0/HFSBUF_365_130
 282.508   134.988    0.6585    0.7056     0.6397     0.5752      0.7057      0.6678      0.02817    sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/VDD_reg_file/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/HFSBUF_797_303
 513.0583  134.556    0.6539    0.7062     0.6407     0.579       0.7063      0.6723      0.03156    VDD_cmplx/VSS_cmplx sc2/wRf_inst/openMSP430_inst/execution_unit_0/register_file_0/HFSBUF_854_239
 510.412   166.808    0.6683    0.7061     0.6394     0.5801      0.7062      0.67         0.0306    VDD_cmplx/VSS_cmplx sc3/wSrLPF_inst/openMSP430_LPF_inst/execution_unit_0/register_file_0/HFSBUF_806_247
  280.78    15.944    0.6578    0.7063     0.6491     0.5829      0.7064      0.6767      0.02761    VDD_cmplx/VSS_cmplx sc4/wRf_inst/openMSP430_inst/execution_unit_0/register_file_0/HFSBUF_854_239
 290.986   299.244    0.6696    0.6999     0.6461     0.5844      0.7064      0.6733      0.02715    VDD_cmplx/VSS_cmplx sc1/wSr_inst/openMSP430_inst/execution_unit_0/register_file_0/HFSBUF_2259_324
 298.154   108.272    0.6731    0.6988     0.6513     0.5883      0.7066      0.6748      0.02353    sc4/wSrPgt_inst/openMSP430_pgt_inst/VDD_openmsp/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/mem_backbone_0/HFSBUF_129_1602
  497.83   163.352    0.6833    0.7061     0.6536     0.5907      0.7062      0.6779      0.02424    VDD_cmplx/VSS_cmplx sc3/wSrLPF_inst/openMSP430_LPF_inst/execution_unit_0/register_file_0/HFSBUF_90_238
 286.612   133.692    0.6681    0.6992     0.6531     0.5914      0.7057      0.6748      0.02167    sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/VDD_reg_file/VSS_cmplx sc4/wSrPgt_inst/openMSP430_pgt_inst/execution_unit_0/register_file_0/HFSBUF_19048_199
  282.4    284.988    0.6666    0.7062     0.6522     0.5914      0.7063      0.6778      0.02556    VDD_cmplx/VSS_cmplx sc1/wSr_inst/openMSP430_inst/execution_unit_0/register_file_0/HFSBUF_790_244
```

File: inst_voltage.rpt

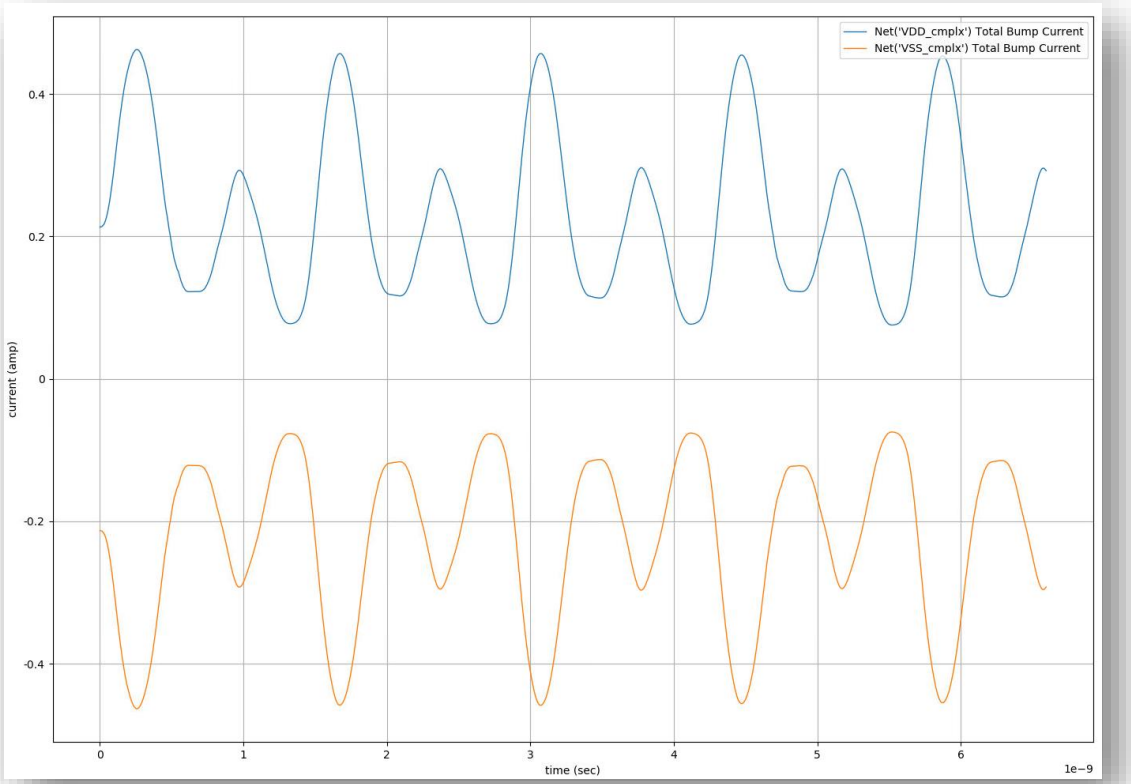
Looking At Chip Currents

```
dem_curr = av_dynamic.get_total_used_demand_currents()
plot(dem_curr)
```



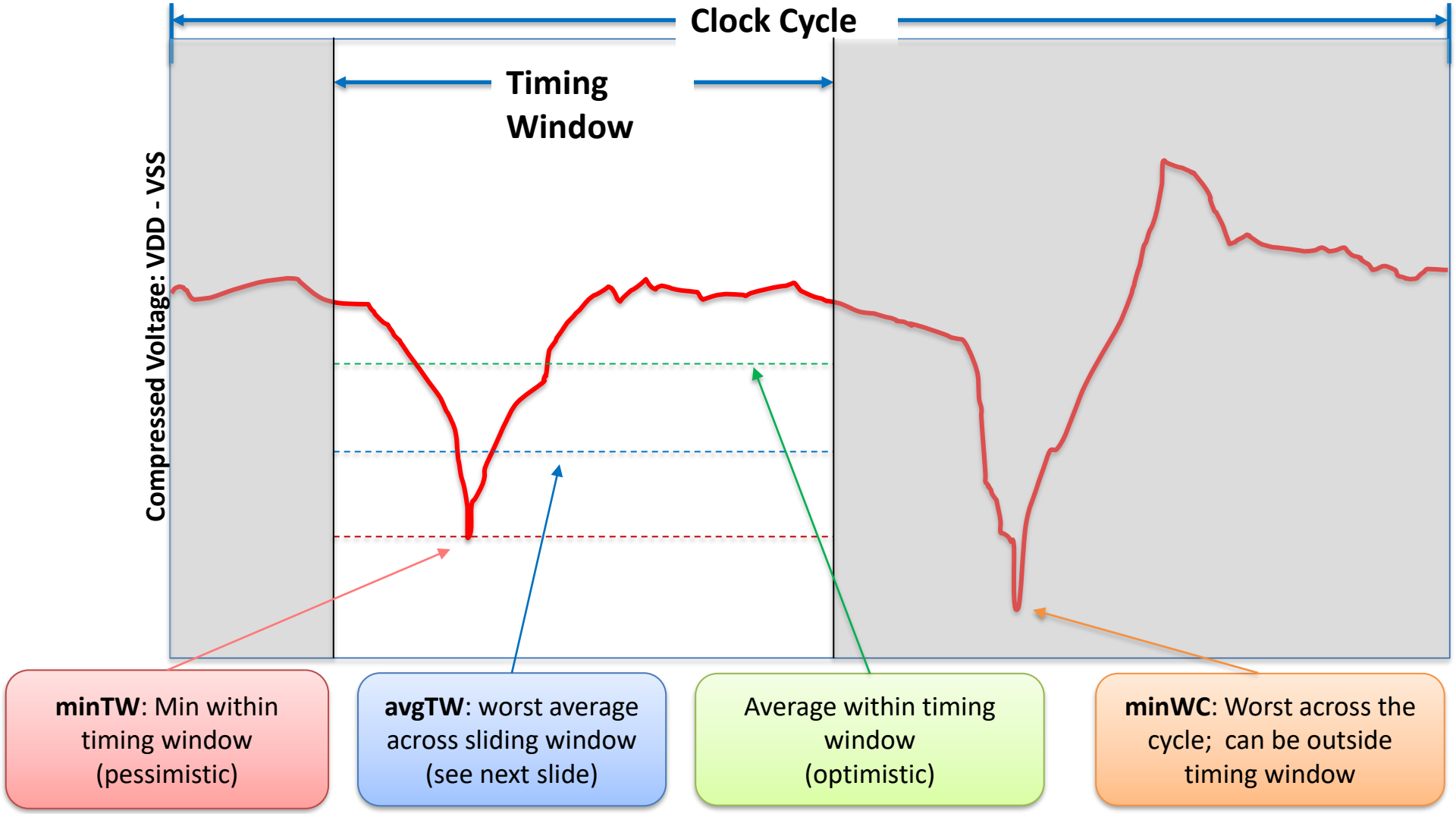
Used Demand Currents in Simulation

```
bump_curr = av_dynamic.get_total_bump_currents()
plot(bump_curr)
```

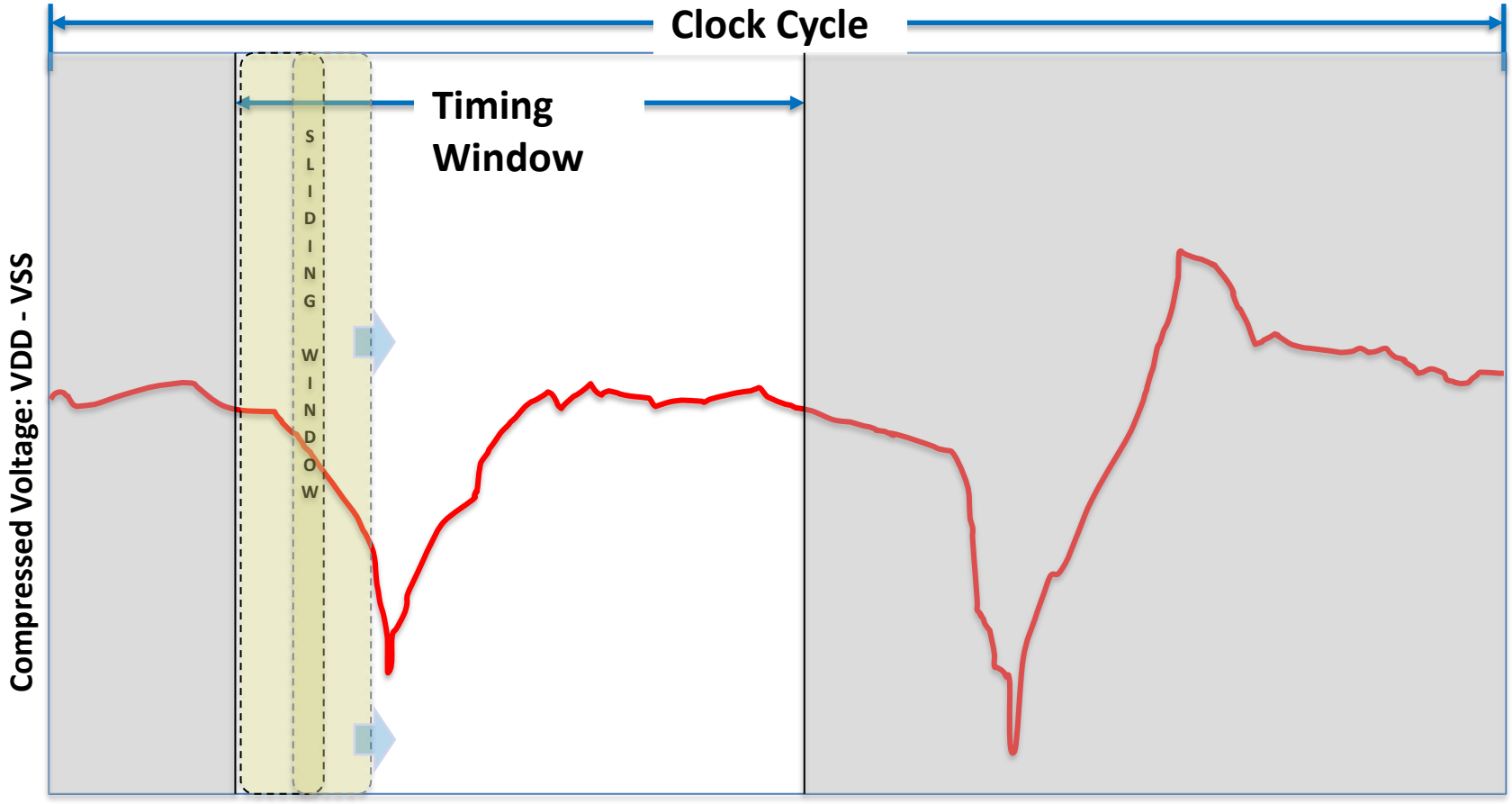


Total Bump Currents in Simulation

Dynamic Voltage Drop Parameters

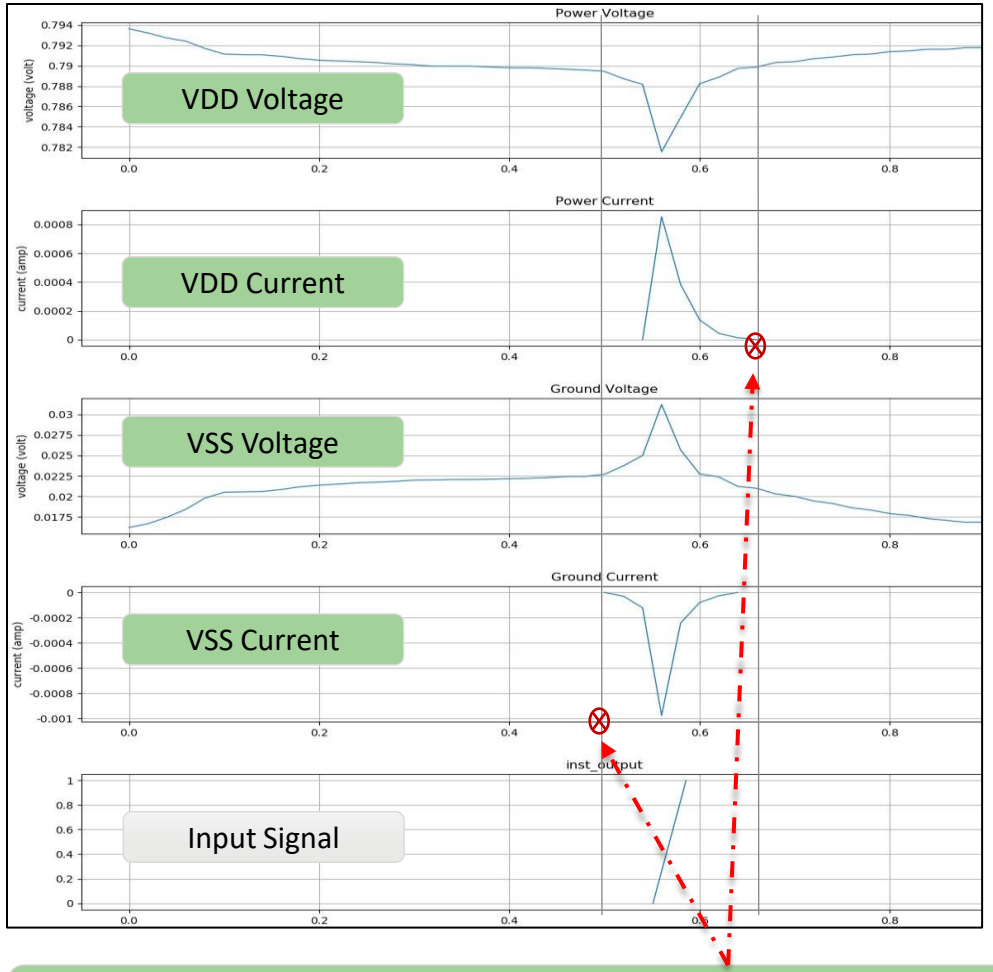


Moving Average Approach (Sliding Window)



- A sliding window is created representing the switching duration (time when instance is sensitive to voltage drop)
- This window is slowly moved across the Timing window and the average voltage drop (across sliding window) is measured
- AvgTW is measured as the worst among sliding window average voltage drop values

Effective DVD measurement

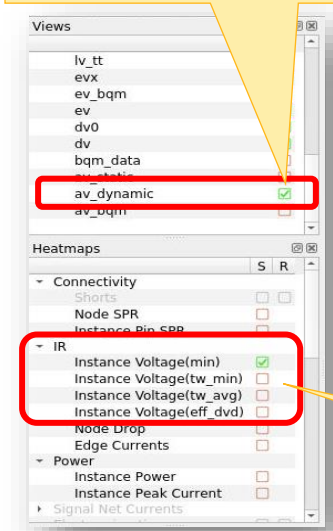


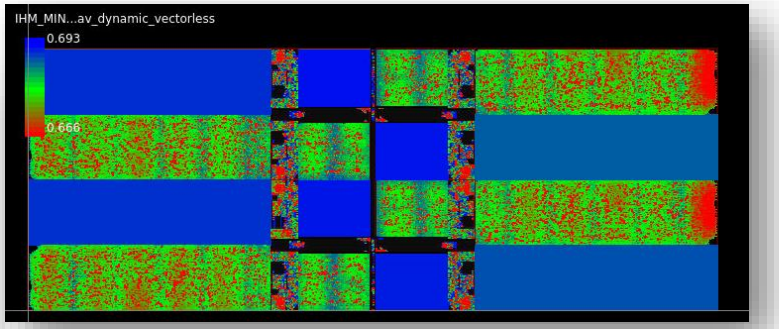
Parameter	Description
Effective DVD	Average VDD-VSS voltage in a current stamping window
Rise effdvd	*Effdvd* when any of the output signals rises
Fall effdvd	*Effdvd* when any of the output signals falls
in_only effdvd	*Effdvd* when there is no transition on output signals
fullsim effdvd	Minimum of the *Effdvd* for each type across all cycles
fullsim avgTW	Minimum of the cycle *AvgTW* rise/fall data across all cycles

"Effective DvD Window ": overlap window of currents of all PG pins of instance

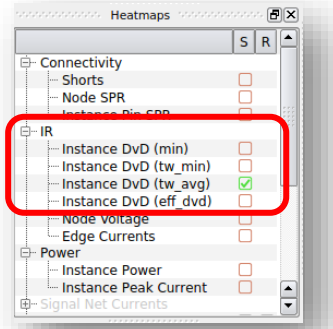
GUI – Instance IR Voltage Maps

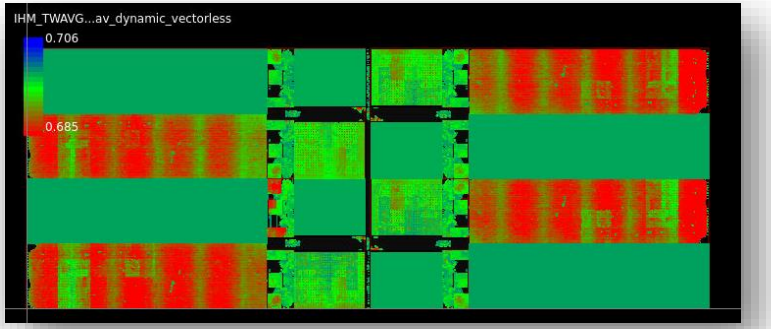
Select Dynamic AV view





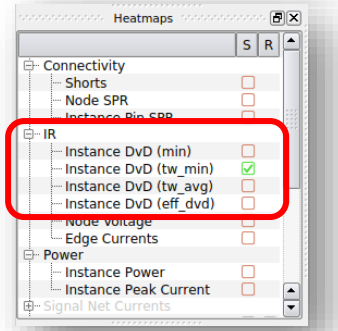
Min DVD compression Over Whole Cycle

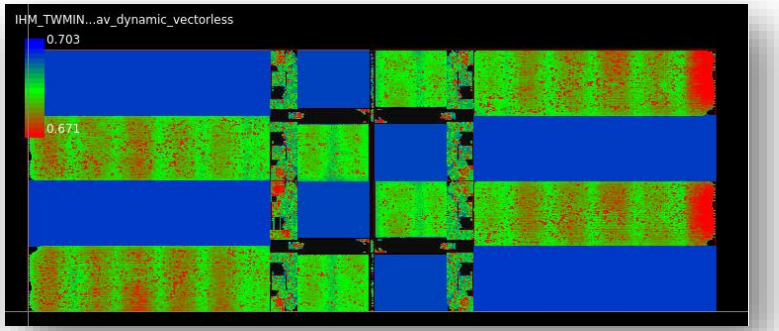




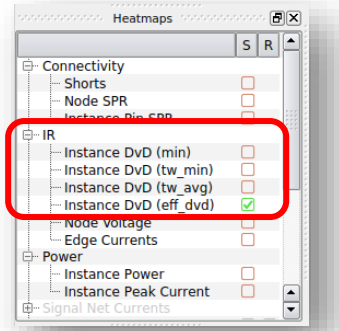
Average DVD compression Over Timing Window (using Sliding Window)

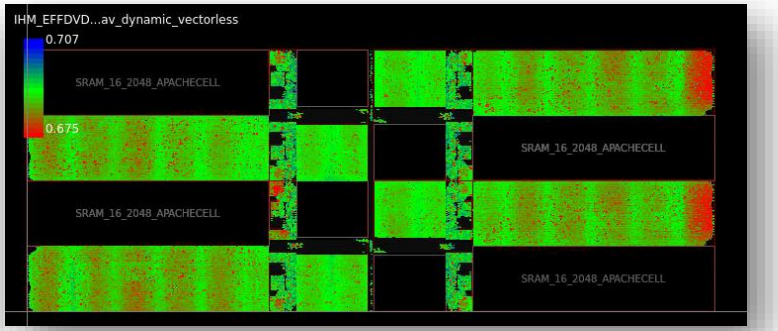
Enable desired "Instance DVD" Heatmap





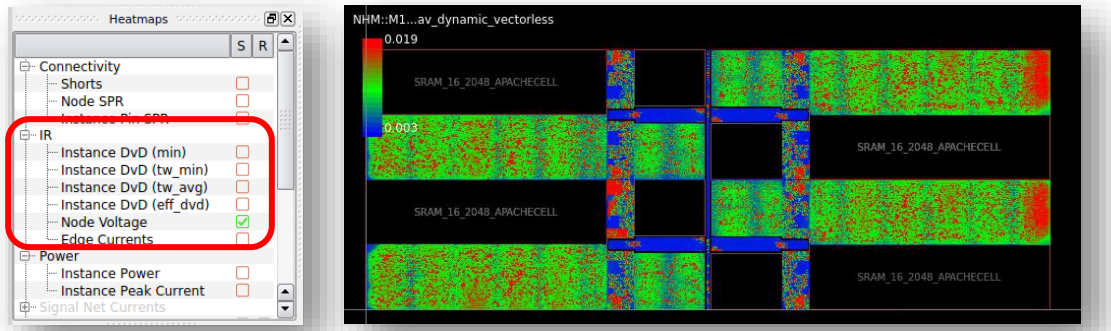
Min DVD compression Over Timing Window



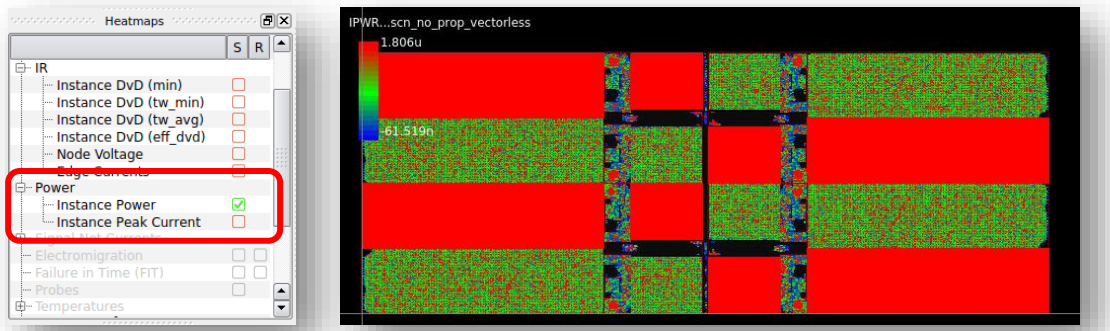


Effective DVD compression Measured over switching window

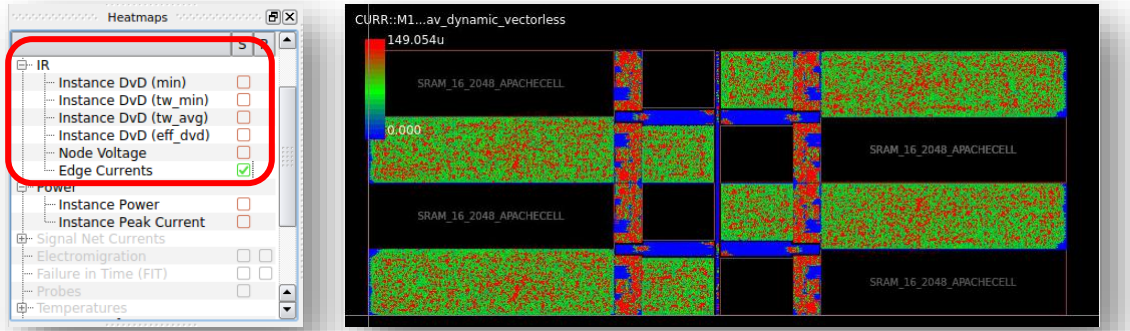
GUI – Node Voltage/Current & Instance Power/Current Maps



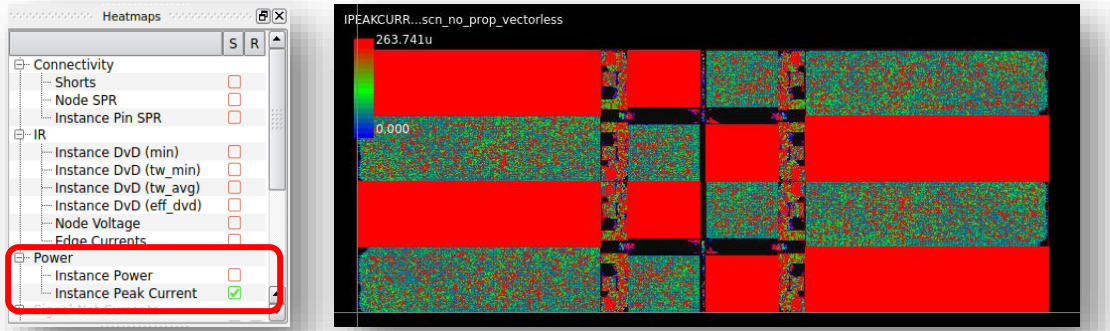
Node Voltage (M1) Map



Instance Power Map



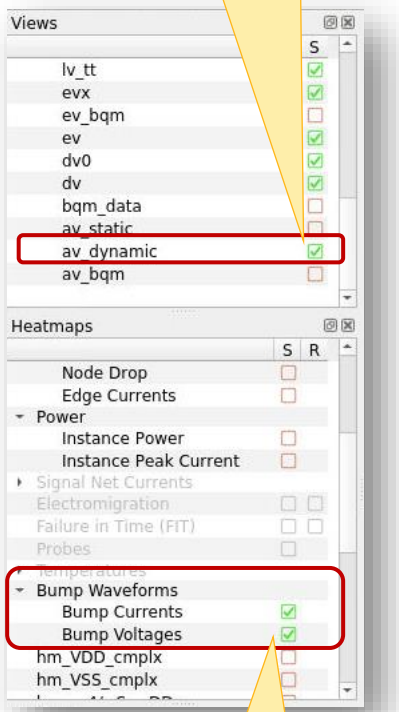
Wire/Edge Current (M1) Map



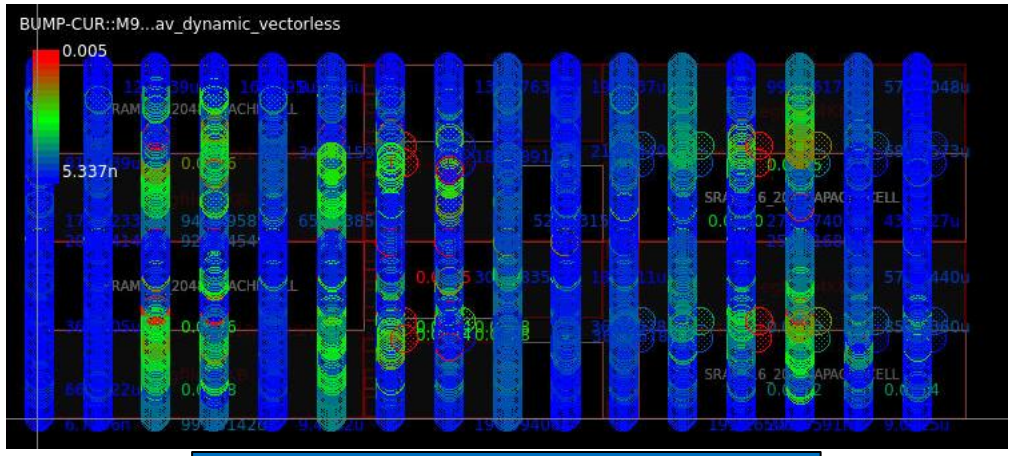
Instance Peak Current Map

GUI – Bump Current/Voltage Maps

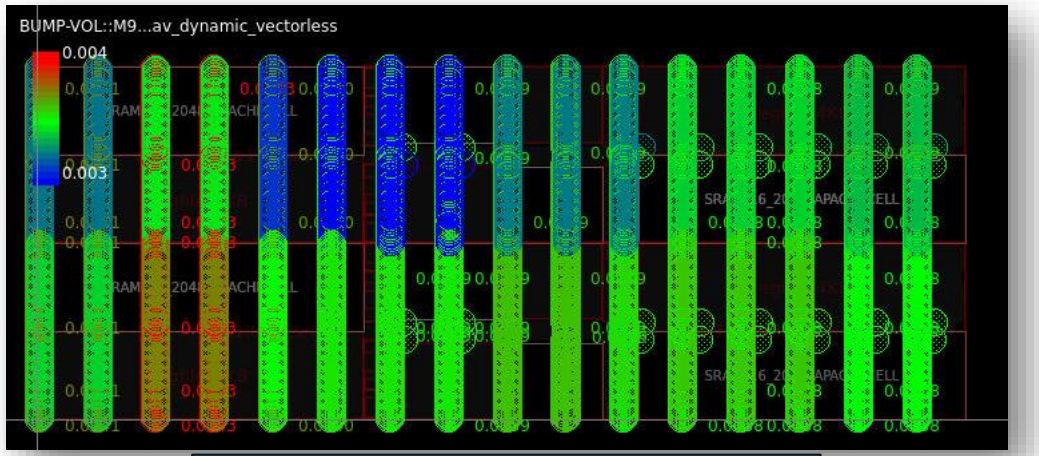
Select Dynamic AV view



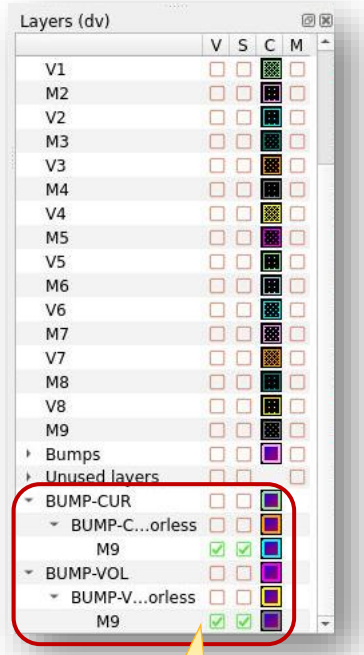
Enable "Bump Waveforms" Heatmaps



Bump Current Map

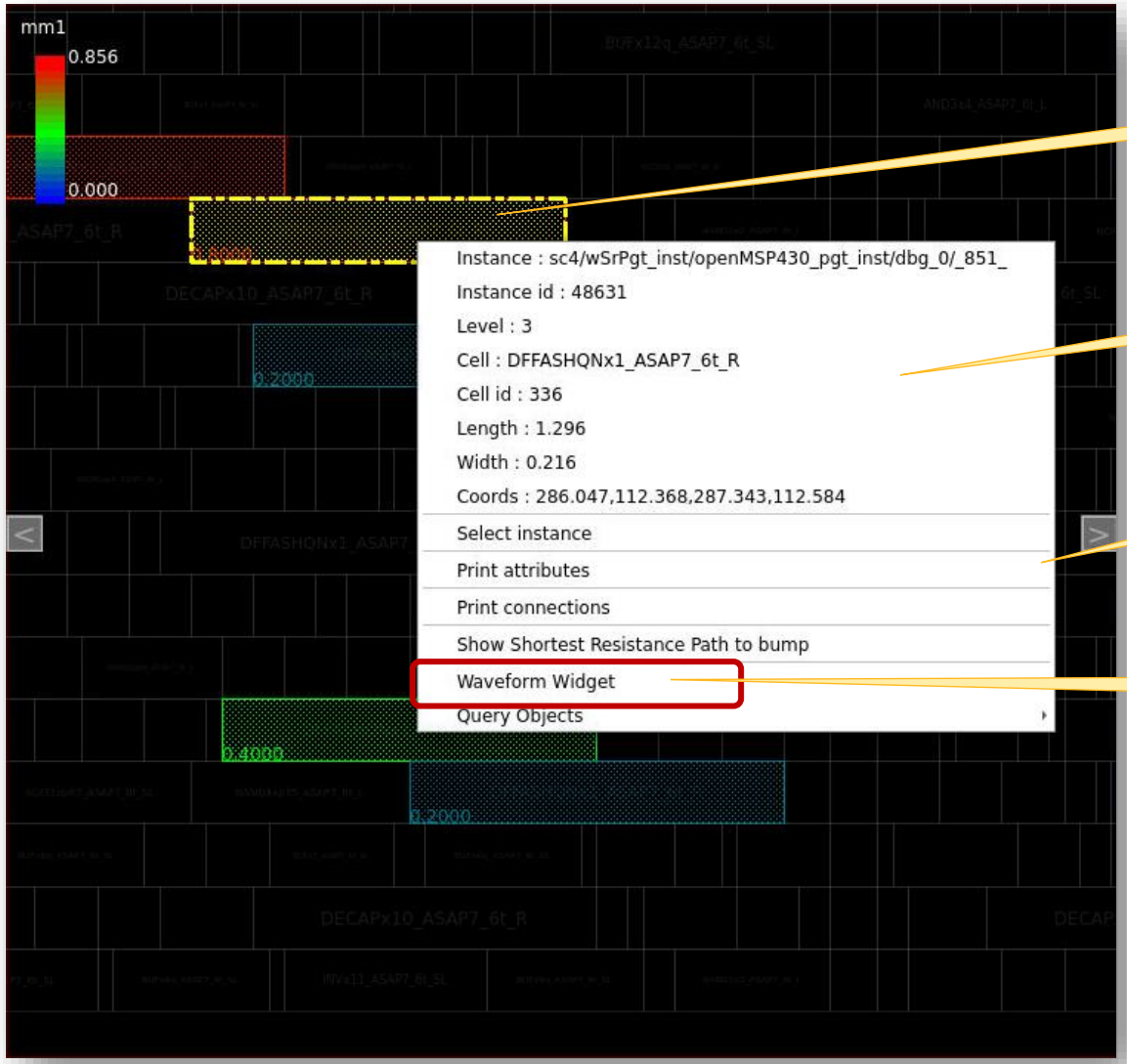


Bump Voltage Map



Select Bump Map (Voltage or Current)

Viewing Instance level voltage/current waveforms



Right click on selected instance to bring up menu

See instance information

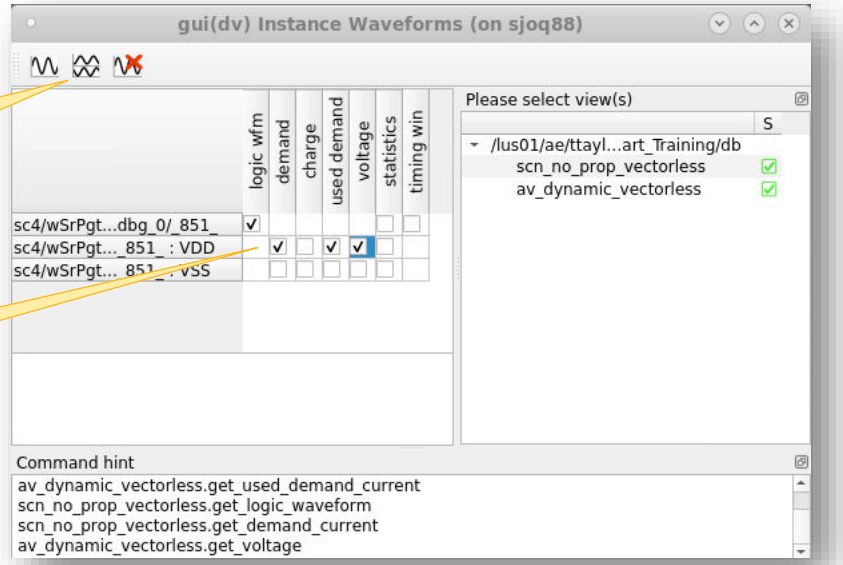
Available commands

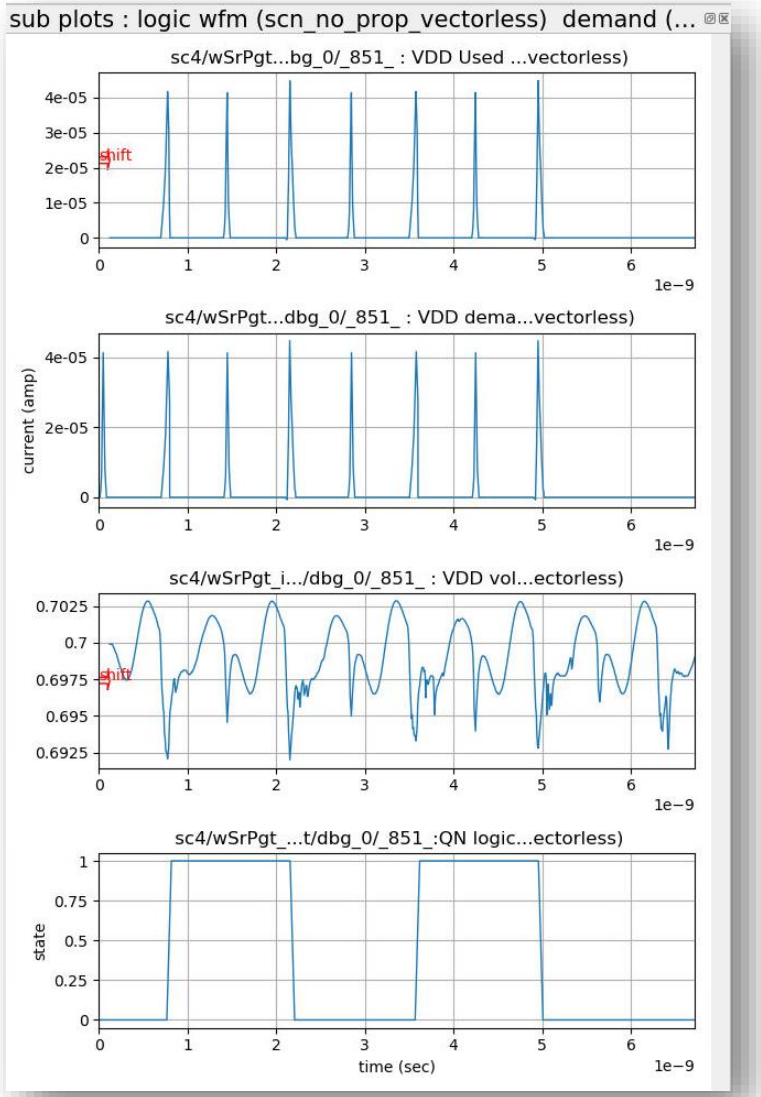
Left click on "Waveform Widget"

Viewing Instance level voltage/current waveforms

Choose whether overlayed or sub_plots (example shows sub_plots)

Select what waveforms you want to plot



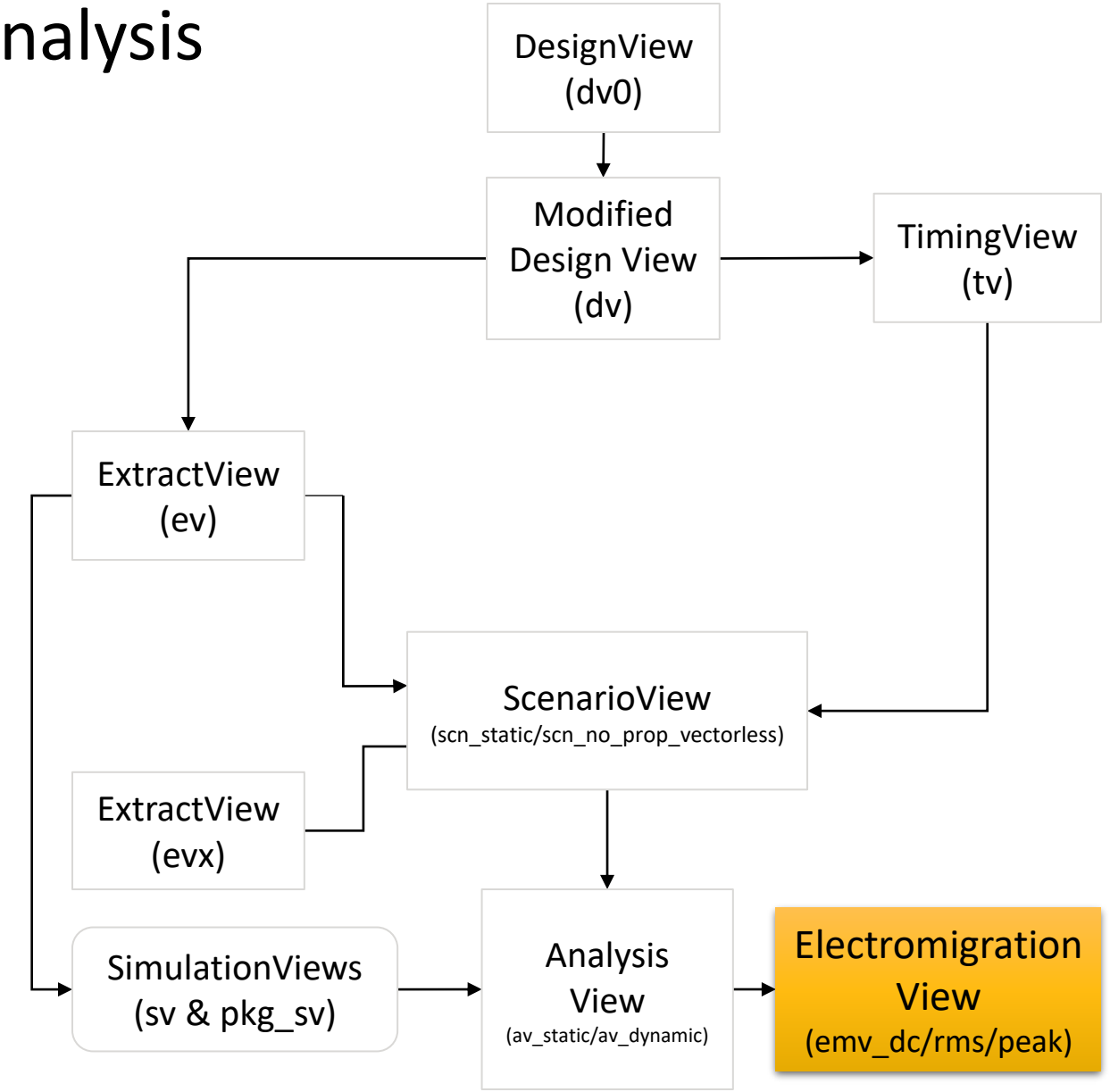


- To get instance level voltage waveforms, require **`keep_stats_level='Full'`** in create analysis_view
  - Default is 'Medium' which will provide voltage stats like minTW, effDVD, etc. but doesn't save waveforms
  - Saving waveforms in mode 'High' or 'Full' will consume disk space
- "Demand Current" comes from ScenarioView. "Used Demand Current" comes from AnalysisView. Important differences:
  - "Used Demand Current" is sampled by time step, whereas the Scenario "Demand Current" is raw PWL without sampling
  - The "Used Demand Current" is shifted
    - Find the average current value of the dominant PG net in the ScenarioView
    - Starting point is at the time when first crossing that average value
    - Achieves quicker stability before dynamic simulation starts and helps shorten presim time

Electromigration Analysis

Performing Electromigration Analysis

- ElectromigrationView overview
  - A specific view performing EM limit checks
  - Can perform the following checks: Avg, RMS and Peak on PG nets
    - **For DC EM (Avg), feed in a static AnalysisView**
    - **For RMS or Peak, feed in a transient AnalysisView**
  - EM rules come from the technology information stored in the TechnologyView
  - For more details, see:
    help(SeaScapeDB.create_electromigration_view)



Preparing for Electromigration run

File: scripts/electromigration.py

```
options = get_default_options()

temperature_em = 110
delta_t = 10

settings_dc = {
    'mode': 'dc',
    'temperature_em': temperature_em}

settings_rms = {
    'mode': 'rms',
    'delta_t_rms': delta_t}

settings_peak = {'mode': 'peak'}

pem_dc = db.create_electromigration_view(av_static, settings=settings_dc, tag='pem_dc', options=options)

pem_rms = db.create_electromigration_view(av_dynamic, settings=settings_rms, tag='pem_rms', options=options)

pem_peak = db.create_electromigration_view(av_dynamic, settings=settings_peak, tag='pem_peak', options=options)

reports_dir = 'reports/powerem_analysis/'
import os
if not os.path.exists(reports_dir):
    os.makedirs(reports_dir)

for em_type, em_view in {'DC_EM': pem_dc, 'RMS_EM': pem_rms, 'PEAK_EM': pem_peak}.items():
    emir_reports.write_em_metal_report(em_view, reports_dir + em_type + '_metal_report.rpt')
    emir_reports.write_em_via_report(em_view, reports_dir + em_type + '_via_report.rpt')
```

Different EM settings for DC/RMS/Peak

Separate EM views for DC/RMS/Peak

Report Generation

Looking At EM Analysis Reports

```
# em_type = DC, ignore_sliver = True, 'length' column is blech length
#  layer            from                  to              length     width      current    constraint violation  status net
#                    (u)                  (u)               (u)       (u)        (A)        (A)        (%)
    M5       (287.1565,117.968)    (287.1565,117.984)      0.06      0.024    1.445e-05  6.413e-05    22.53     PASS  VDD_cmplx
    M8       (299.502,83.5445)      (299.502,83.05)       10.77       0.07    9.314e-05  0.0004985    18.68     PASS  VDD_cmplx
    M8       (295.902,83.5445)      (295.902,83.05)       10.77       0.07    9.211e-05  0.0004985    18.48     PASS  VDD_cmplx
    M8       (289.902,83.5445)      (289.902,83.05)       10.77       0.07    9.058e-05  0.0004985    18.17     PASS  VDD_cmplx
    M8       (285.102,83.5445)      (285.102,83.05)       10.77       0.07    9.007e-05  0.0004985    18.07     PASS  VDD_cmplx
    M8       (281.502,83.5445)      (281.502,83.05)       10.77       0.07    8.918e-05  0.0004985    17.89     PASS  VDD_cmplx
    M7       (287.1565,121.984)    (287.1245,121.984)     0.091      0.032     1.7e-05   0.0001047    16.25     PASS  VDD_cmplx
    M7       (287.1565,123.584)    (287.1245,123.584)     0.091      0.032    1.69e-05   0.0001047    16.15     PASS  VDD_cmplx
    M7       (287.1565,121.152)    (287.1245,121.152)     0.091      0.032    1.607e-05  0.0001047    15.36     PASS  VDD_cmplx
    M7       (287.1565,124.352)    (287.1245,124.352)     0.091      0.032    1.577e-05  0.0001047    15.07     PASS  VDD_cmplx
```

File: DC_EM_metal_report.rpt

```
# em_type = RMS, ignore_sliver = True, 'length' column is blech length
#  layer            from                  to              length     width      current    constraint violation  status net
#                    (u)                  (u)               (u)       (u)        (A)        (A)        (%)
    M1       (486.753,224.002)      (486.76,224.002)      110.8      0.018    7.524e-05  0.0002792    26.94     PASS  VSS_cmplx
    M1       (486.7533,74.002)     (486.7603,74.002)      110.8      0.018    6.477e-05  0.0002792     23.2     PASS  VSS_cmplx
    M1       (486.753,224.002)      (486.598,224.002)     110.8      0.018    5.535e-05  0.0002792    19.82     PASS  VSS_cmplx
    M1       (483.153,224.866)      (483.196,224.866)     110.8      0.018    5.508e-05  0.0002792    19.72     PASS  VSS_cmplx
    M1       (483.196,224.866)      (483.304,224.866)     110.8      0.018    5.507e-05  0.0002792    19.72     PASS  VSS_cmplx
    M1       (483.304,224.866)      (483.358,224.866)     110.8      0.018     5.3e-05   0.0002792    18.98     PASS  VSS_cmplx
    M1       (483.453,224.866)      (483.358,224.866)     110.8      0.018    5.295e-05  0.0002792    18.96     PASS  VSS_cmplx
    M1        (307.085,76.498)       (307.078,76.498)     109.1      0.018    5.055e-05  0.0002792     18.1     PASS  VSS_cmplx
    M1       (486.453,224.002)      (486.544,224.002)     110.8      0.018    4.918e-05  0.0002792    17.61     PASS  VSS_cmplx
    M1       (486.544,224.002)      (486.598,224.002)     110.8      0.018    4.913e-05  0.0002792     17.6     PASS  VSS_cmplx
```

File: RMS_EM_metal_report.rpt

```
# em_type = PEAK, ignore_sliver = True, 'length' column is blech length
#  layer            from                  to              length     width      current    constraint violation  status net
#                    (u)                  (u)               (u)       (u)        (A)        (A)        (%)
    M1        (304.31,168.468)      (304.317,168.468)     7.476      0.018     0.00077    0.001525    50.48     PASS  VSS_cmplx
    M1        (304.31,18.468)        (304.317,18.468)     7.476      0.018    0.0007584   0.001525    49.72     PASS  VSS_cmplx
    M1        (283.729,12.272)       (283.75,12.272)       16.2      0.018    0.0007228   0.001525    47.38     PASS  VDD_cmplx
    M1        (300.78,296.47)        (300.729,296.47)     8.518      0.018    0.0007201   0.001525    47.21     PASS  VSS_cmplx
    M1       (511.5868,114.96)      (511.6258,114.96)      16.2      0.018     0.000718   0.001525    47.07     PASS  VDD_cmplx
    M1       (493.0583,146.47)      (493.1093,146.47)     8.518      0.018    0.0007118   0.001525    46.66     PASS  VSS_cmplx
    M1         (303.23,64.26)         (303.179,64.26)     9.328      0.018    0.0007098   0.001525    46.53     PASS  VSS_cmplx
    M1       (299.504,198.276)      (299.517,198.276)     30.39      0.018     0.000707   0.001525    46.35     PASS  VSS_cmplx
    M1       (489.6093,130.952)     (489.5823,130.952)    7.485      0.018     0.000697   0.001525    45.69     PASS  VDD_cmplx
    M1       (489.609,280.952)      (489.582,280.952)     7.485      0.018    0.0006969   0.001525    45.68     PASS  VDD_cmplx
```

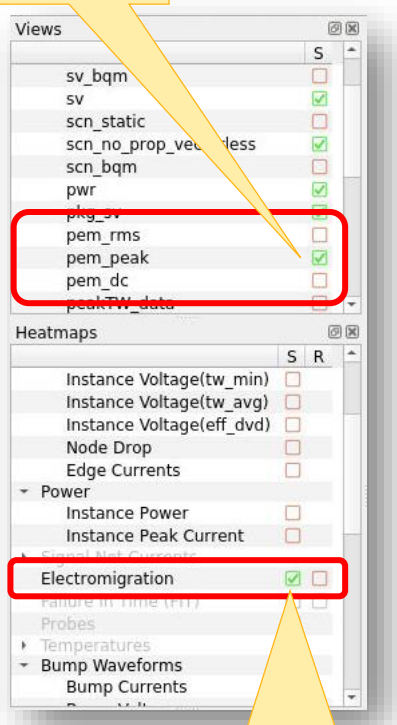
File: PEAK_EM_metal_report.rpt

ElectroMigration Analysis Reports available in "reports/powerem_analysis" directory

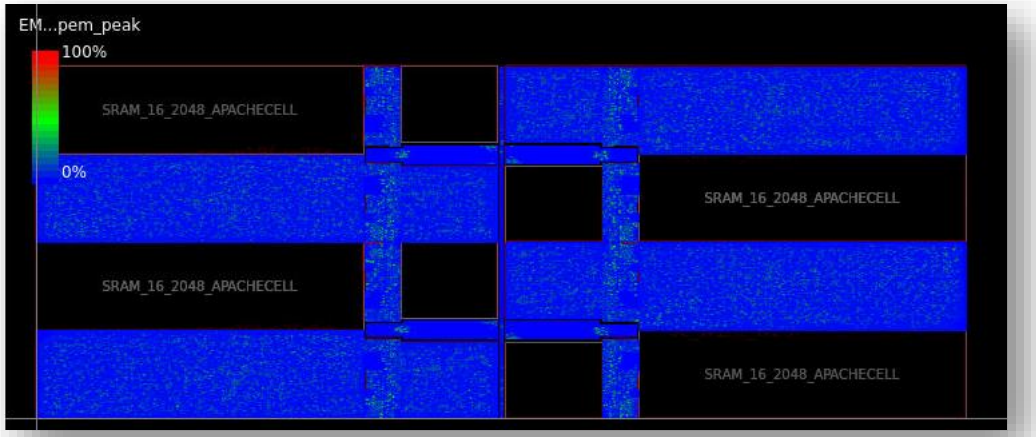
```
PEAK_EM_via_report.rpt
PEAK_EM_metal_report.rpt
RMS_EM_metal_report.rpt
DC_EM_metal_report.rpt
RMS_EM_via_report.rpt
DC_EM_via_report.rpt
```

GUI – EM Violation Maps

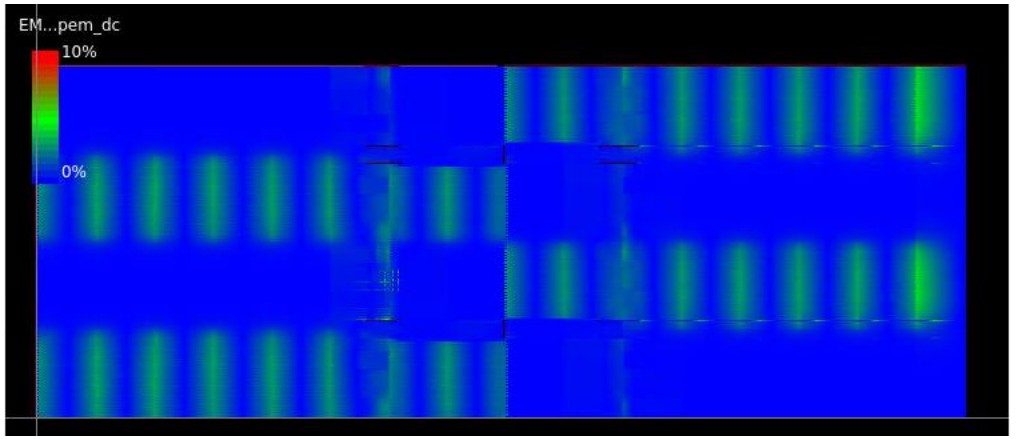
Select EM view



Enable "Electromigration" Heatmap

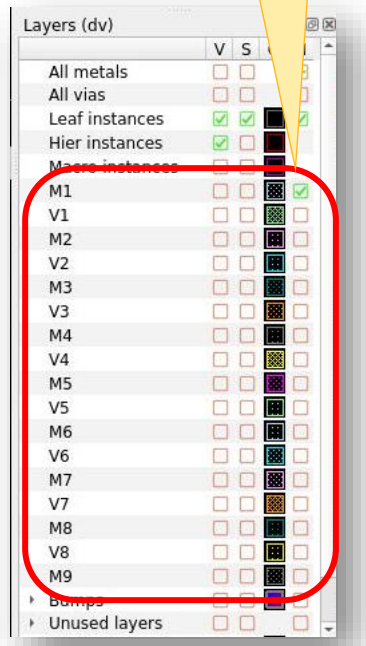


EM Peak M1

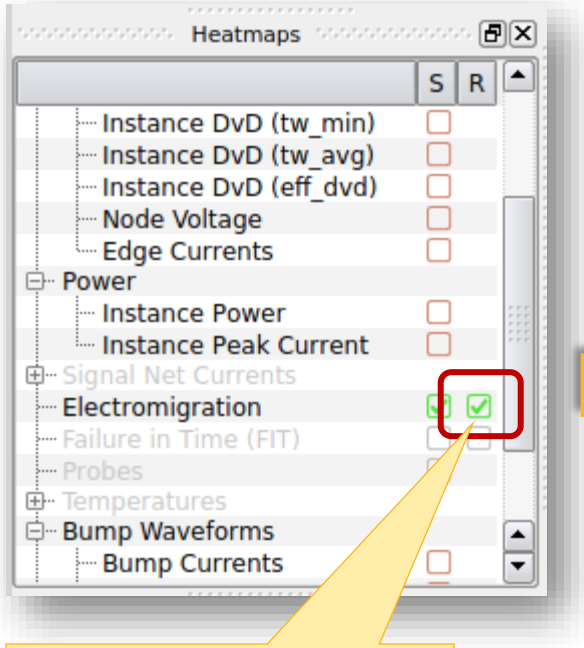


EM DC M9

Select metal/via layer Map

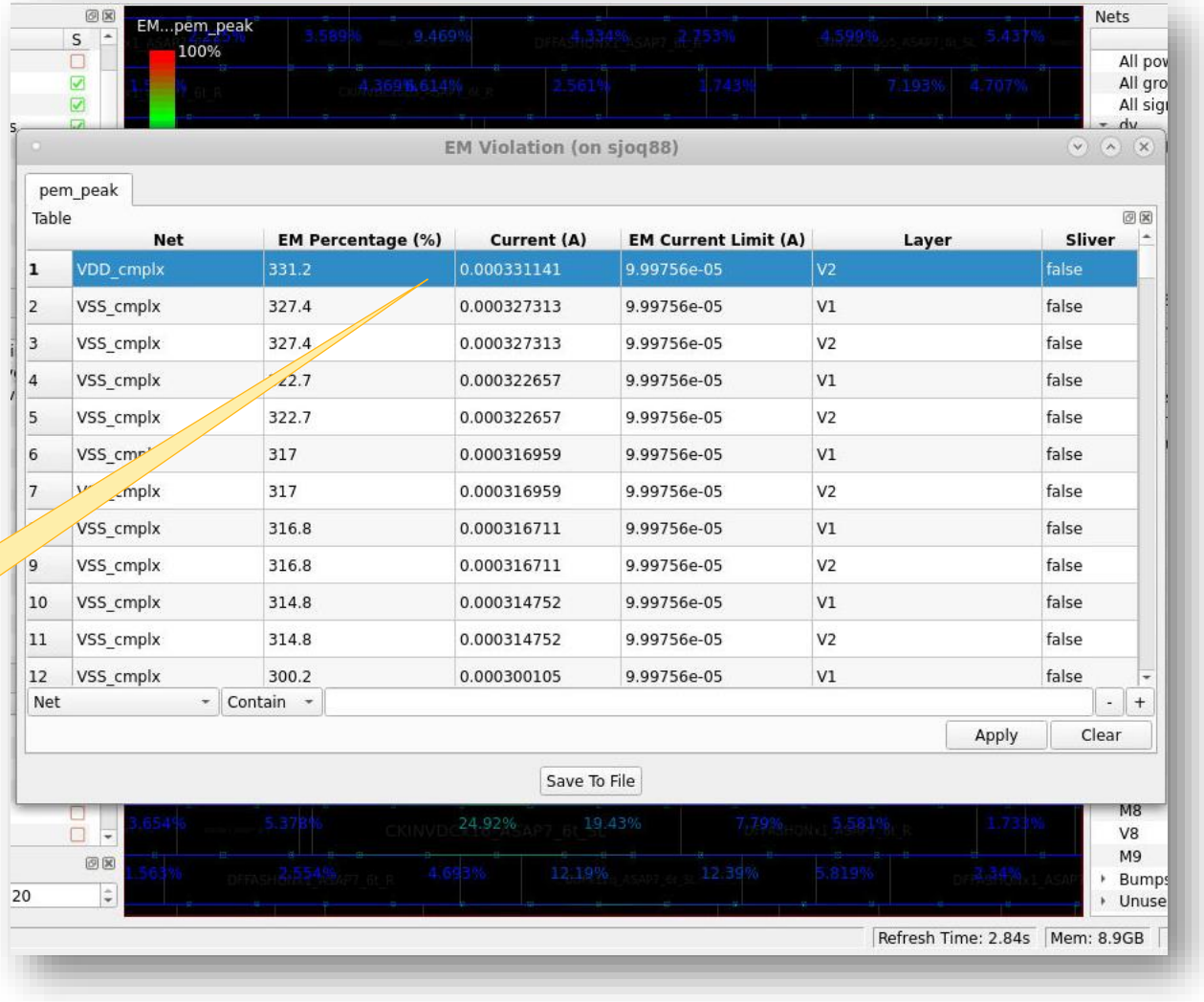


GUI – EM Violation Browser

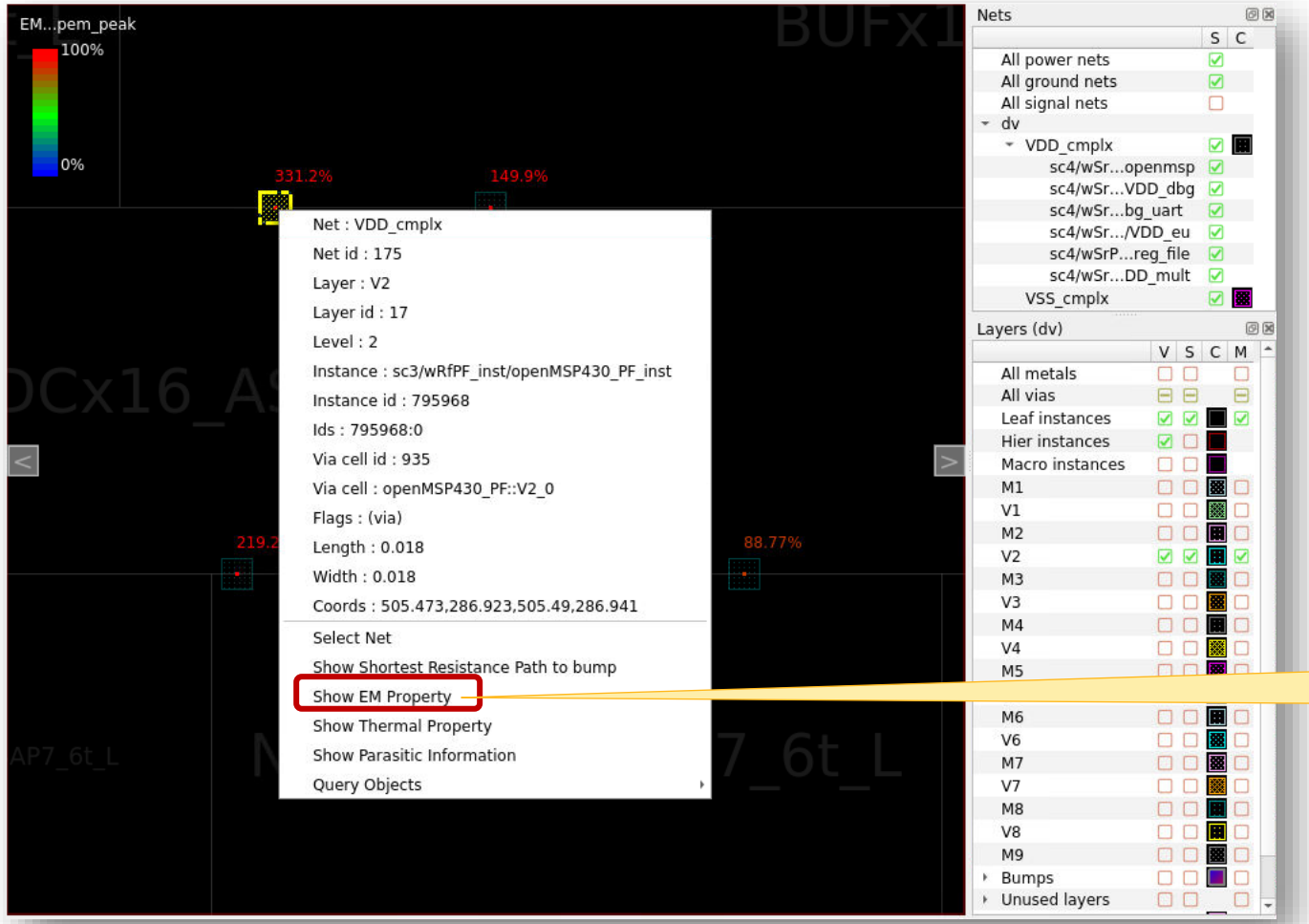


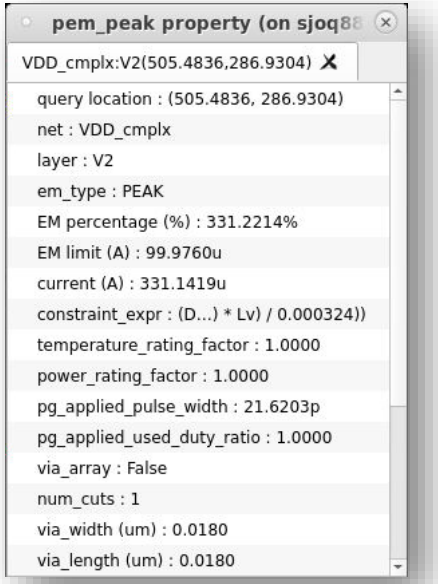
Click on box under letter "R" in Electromigration heatmap to open up EM Violation browser

Double click on the violation to zoom into the location in GUI



GUI – EM Properties for Wire/Via





Right click on selected wire segment or via and left click on "Show EM Property"

Introduction to SC Help System & APIs

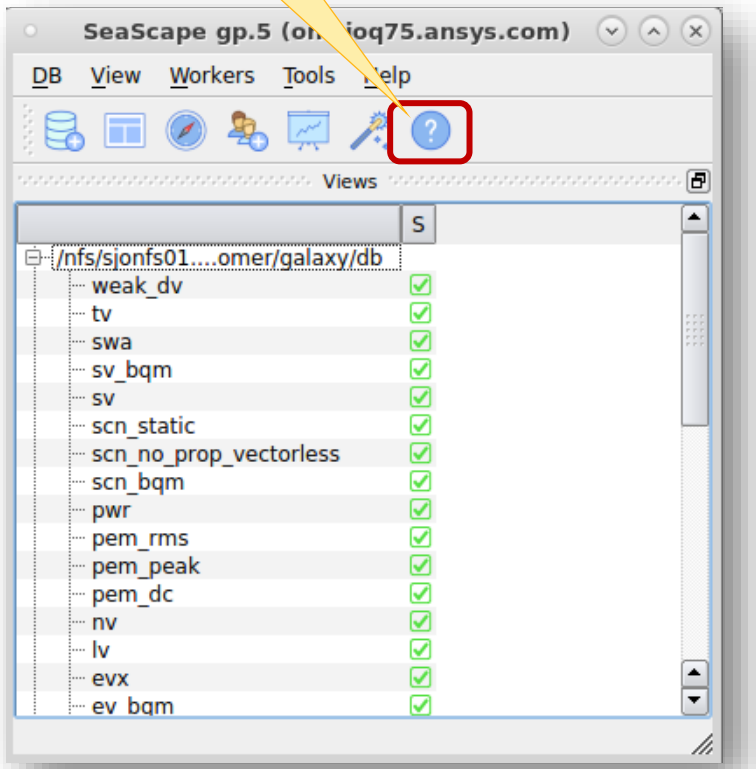
Ansys

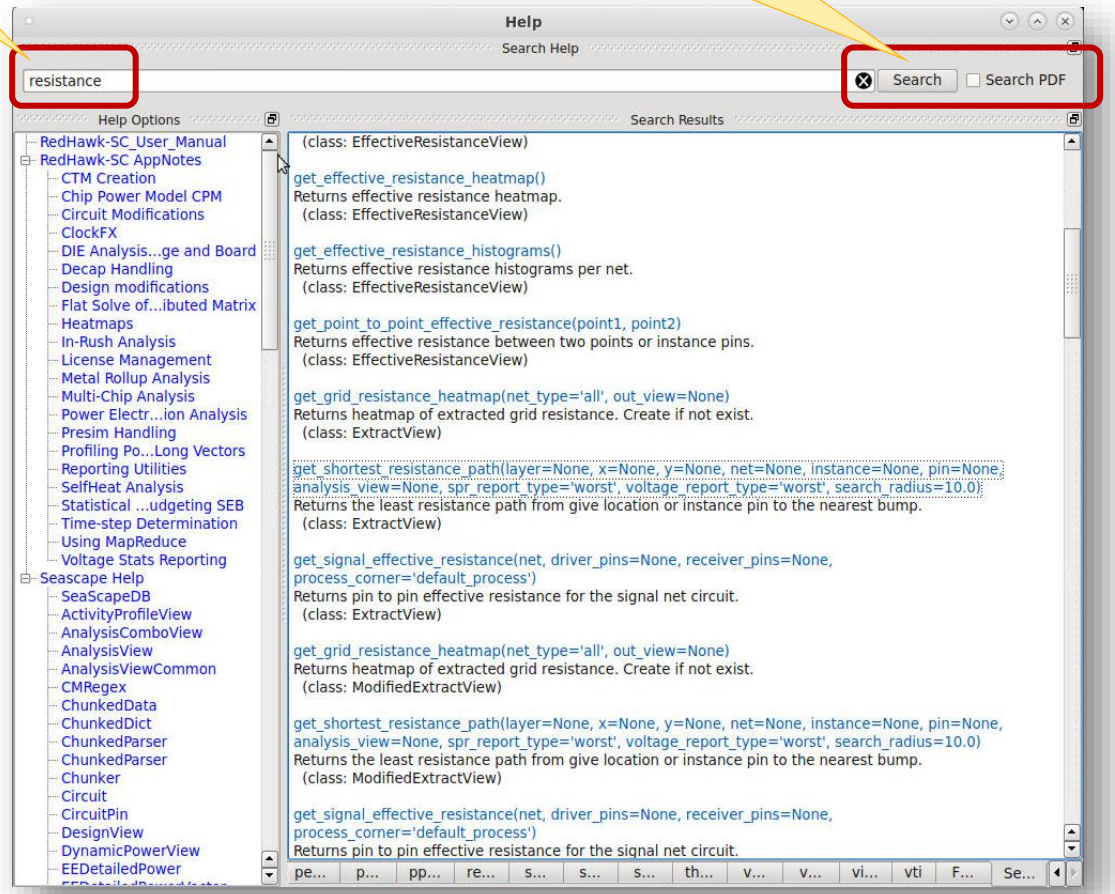
Accessing Help From RedHawk-SC Console

Open console and click on help button

Enter search item in box, for example: resistance

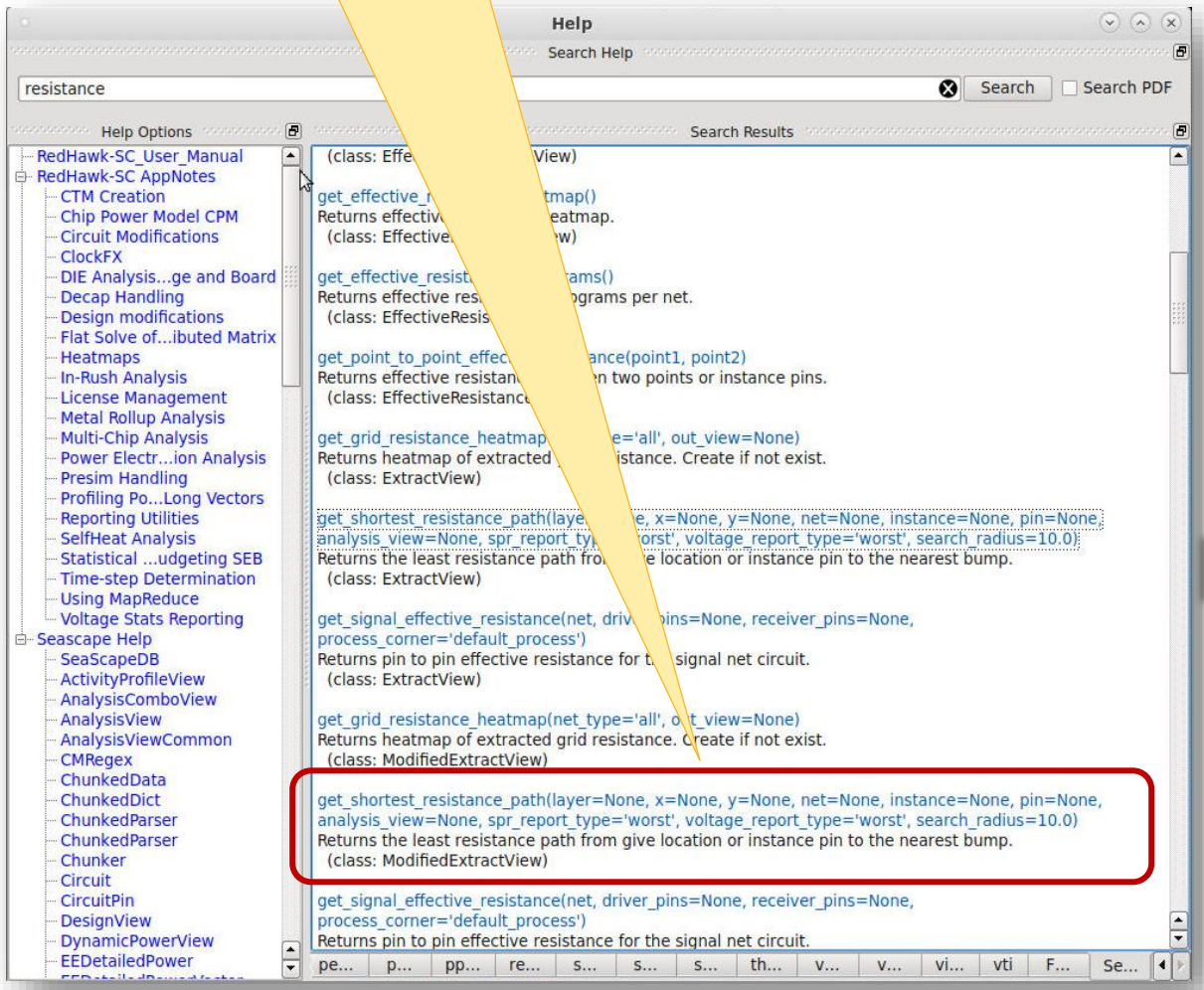
Click on "Search" to get all results (Click on "Search PDF" if you want to search App Notes or User Manual)

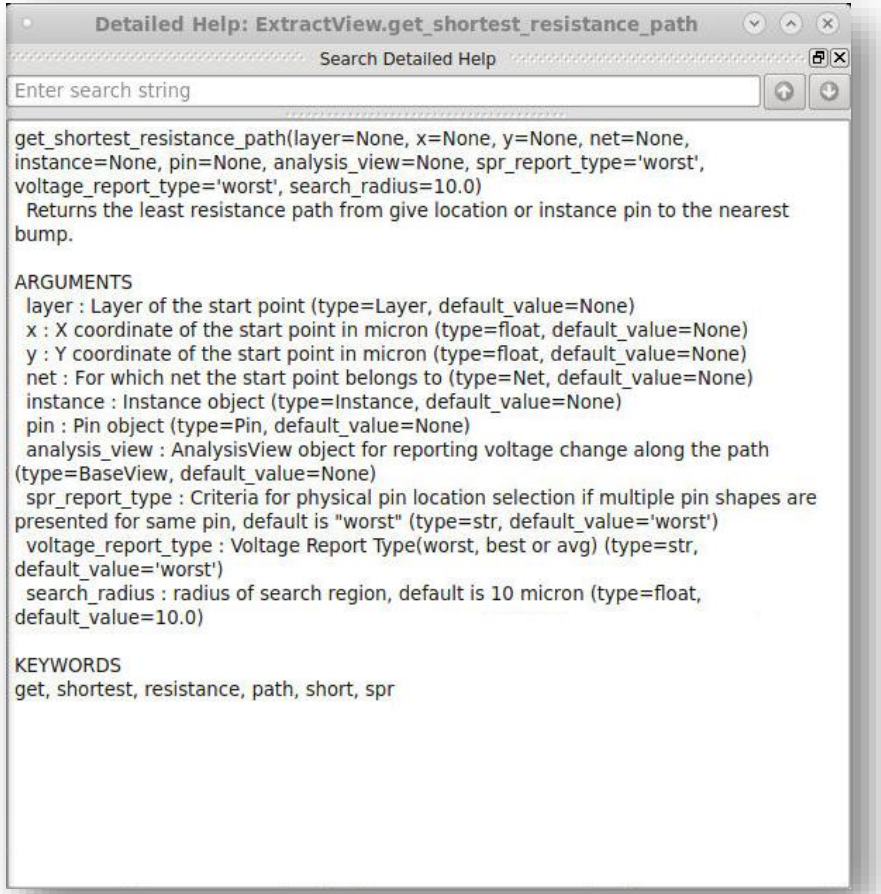




Accessing Help From RedHawk-SC Console

Click on any search result to get more details



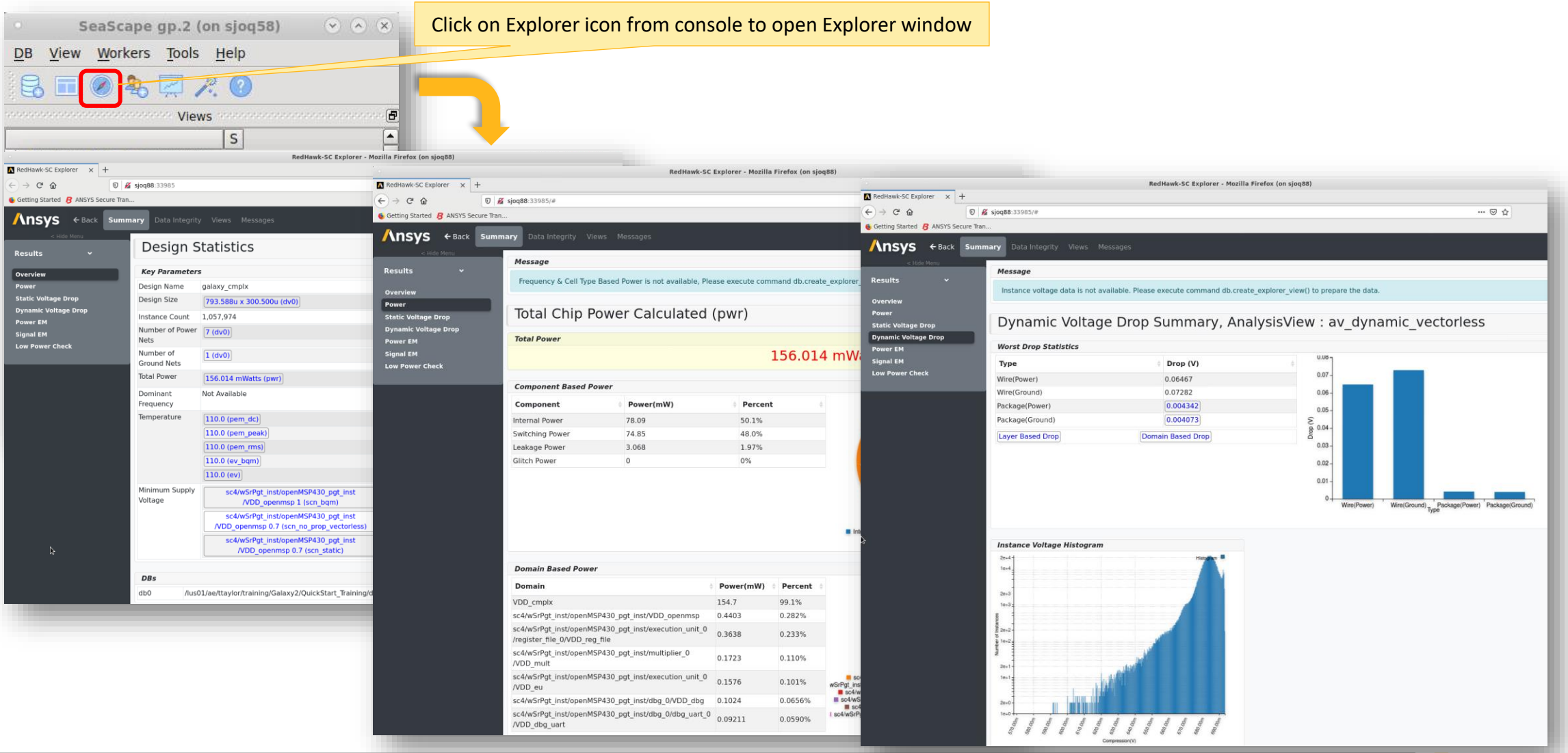


Accessing Help From RedHawk-SC Python Shell

```
>>> help(DesignView)
CLASS
DesignView
   View for design data including layout and connectivity.

SEE ALSO
SeaScapeDB.create_design_view

FUNCTIONS
convert_to_id(obj)
  Converts name-based object or iterable to id-based object or list.

convert_to_name(obj)
  Converts id-based object or iterable to name-based object or list.

get_attributes(obj, include_pin_geoms=None, include_phantoms=None, include_clock_gate_pins=None,
include_di_checks_stats=None, only_data_types=None, include_data_types=None)
  Returns a dict of DesignView attributes for the specified object.

get_cell_data_coverage()
  Returns a dict of DesignView DI Summary.
.. .. ..
```

Looking at Explorer Results

Click on Explorer icon from console to open Explorer window



Understanding the API – SeaScapeDB Objects

- SeaScape organizes data by SeaScapeDB objects, the most common of which are:
  - **Layer, Cell, Instance, Pin, Net** , etc
- Each of these objects has both a string and integer representation
- In distributed processing and MapReduce, strings are highly inefficient
- Conversion between string and integer can be done through :
  - <DesignView>.convert_to_name(<object>)
  - <DesignView>.convert_to_id(<object>)

```
>>> dv.convert_to_id(Instance("sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/_898_"))
Instance(48678)
>>> dv.convert_to_name(Instance(48678))
Instance('sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/_898_'))
>>> >>> dv.convert_to_id(Pin('VSS_cmplx'))
Pin(267)
```

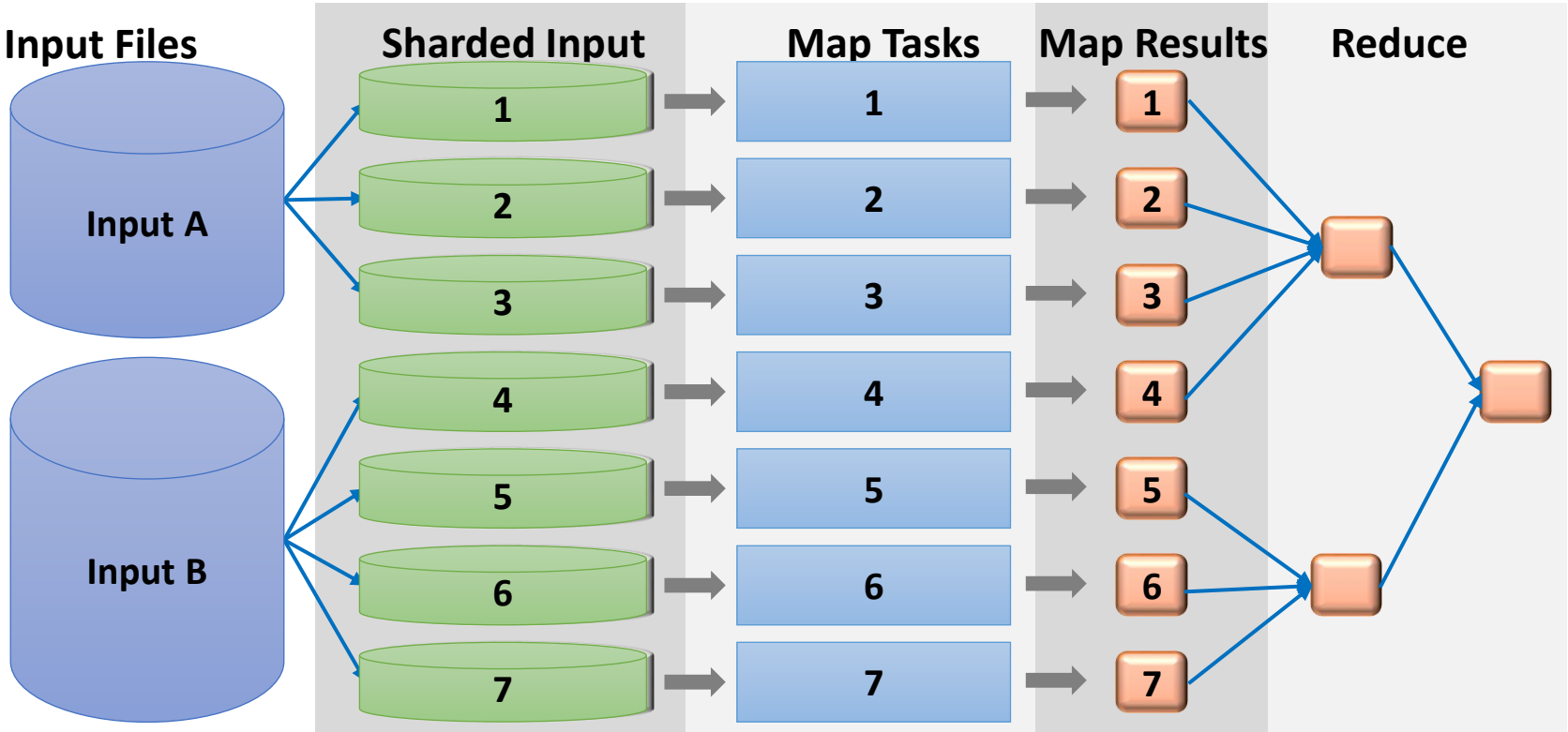
Object Attribute Query Examples

```
>>> dv.get_attributes(Instance("sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/_898_"))
{'cell': Cell('DFFASHQNx1_ASAP7_6t_R'),
'cell_id': 336,
'coord': RealCoord(293.714500,123.816000),
'instance_id': 48678,
'is_leaf': True,
'rotation': Rotation(RotationType.N)}
```

```
>>> scn_no_prop_vectorless.get_attributes(Instance("sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/_898_"))
{'logic': {'clock_instance': False,
         'clock_tree_level': None,
         'clocks_reached': ['dco_clk'],
         'frequency': 714285696.0,
         'logic_level': 0,
         'output_pins': {Pin('QN'): {'load_cap': 5.394114389978646e-16,
                                     'logic_signal': {'initial_value' : 1, 'events' : [],},
                                     'net': Net('sc4/wSrPgt_inst/openMSP430_pgt_inst/dbg_0/_025_'),
                                     'num_events': 0,
                                     'pi_model': (2.368292870866693e-17,
                                                  132.97044372558594,
                                                  5.157510559239734e-16),
                                     'toggle_rate': 0.0,
                                     'tw_file': {'tw_clock': 'dco_clk'}}}},
'power': {Pin('VDD'): {'clock_pin_power': 0.0,
                       'frequency': 714285696.0,
                       'internal_power': 0.0,
                       'leakage_power': 6.870469471120799e-11,
                       'source': 'Scenario',
                       'switching_power': 0.0,
                       'toggle_rate': 0.0,
                       'total_power': 6.870469471120799e-11,
                       'voltage': 0.699999988079071}}}
```

Introduction to MapReduce (MR)

MapReduce (MR) is a simplified model for processing and generating large data sets with a parallel, distributed algorithm on a cluster in an automated, easy to write, fashion.



Disclaimer: This section is going to cover only basics of MapReduce usage in RedHawk-SC.
For detailed training, please refer to RedHawk-SC Modular Training on "Python APIs & MapReduce"

Hands on MapReduce in SeaScape

- **What type of data am I going to analyze in MapReduce?**
  - Most common data which can be analyzed using MapReduce are sets of **Instances**
  - Also available for MapReduce are:
    - **Circuits** :: the actual extracted circuit
    - **Shapes** :: the geometries in the design
- **What type of data am I going to return?**
  - Most common data set returned from MapReduce is **list** and **dict**
  - Reduce operations are built-in within SeaScape
  - You can write your own custom reduce function also
  - Support for statistics, Waveforms, and others are available
- **What type of reduction will I need for my return?**
  - Any Python object that has '+' operator uses **reduce_add**
  - For reducing dict, **reduce_add_dict** is available
  - Reduce_add_dict is the default reducer for MapReduce (when not specified)

MapReduce Example: Find my Clock Instances in Design

```
def get_clk_inst_select(instances, scn):
    dv = scn.get_related_views(DesignView)[0]
    out = list()
    for inst in instances:
        is_clock = scn.get_attributes(inst)['logic'].get('clock_instance')
        if is_clock:
            out.append(inst)
    return out
```

Serial Execution:

```
data = get_clk_inst_select(dv.get_instances('*'), scn=scn_no_prop_vectorless)
```

Parallel Execution using MapReduce:

```
mm = MapReduce(dv)
mm.map_reduce(dv.get_mr_instances(), partial(get_clk_inst_select, scn=scn_no_prop_vectorless))
data = mm.get()
gp_print("Found ", len(data), 'clock instances')
clki_selection = InstanceSelection(dv,data)
gui.add_layer(clki_selection,'clock_instance_map')
```

Anatomy of a SeaScape MapReduce function

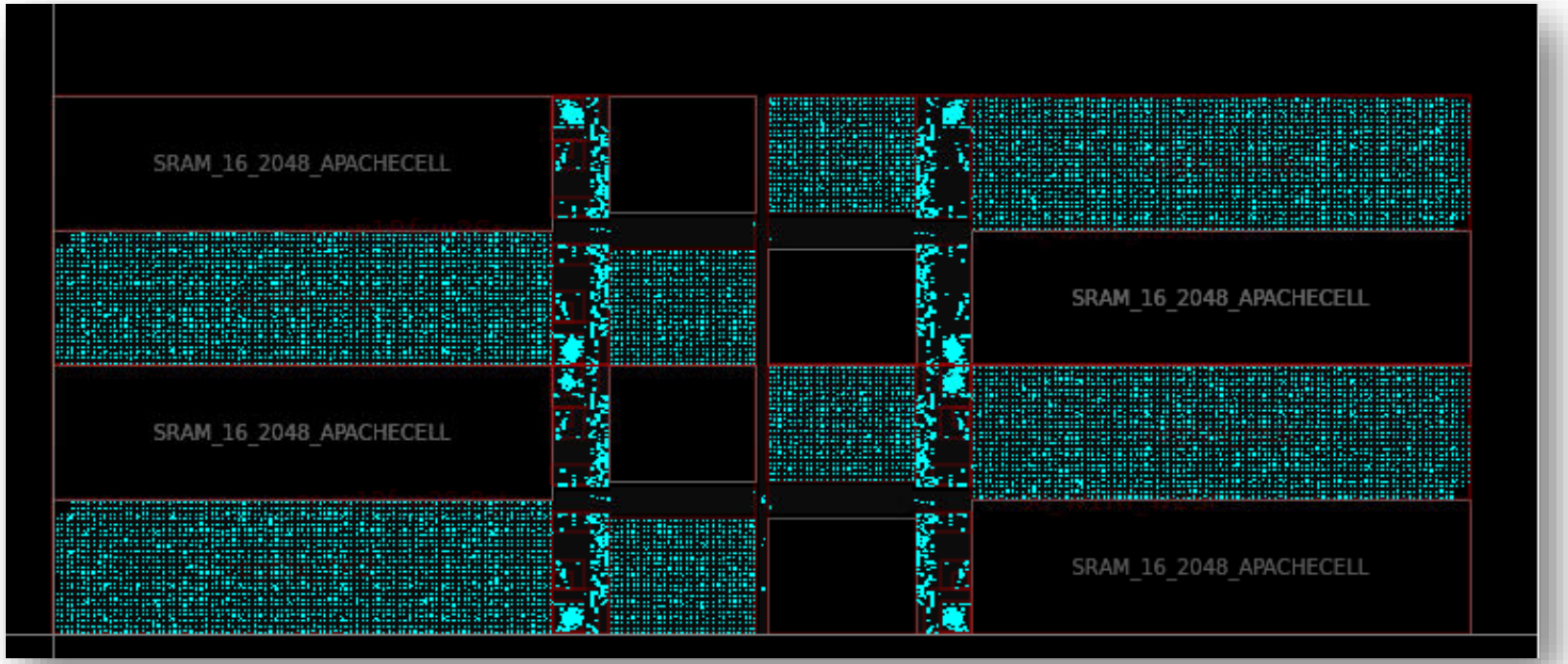
```
def map_func(<mapped data>, <arguments>):
    return <something>
    # map_func is the python job that will run in every worker
    # map_func will do some operations with a return data object (list, dict, etc.)
instances = dv.get_mr_instances()
# Get the sharded or mapped data, here gets sharded instances
# Sharded data can be Instances, Nets, Circuits etc
mm = MapReduce(dv)
# Informs the system what sharded data (here Instances) will be used in MapReduce.
# SeaScape internally keeps the instance data in sharded way for MapReduce operations
mm.map_reduce(<mapped data>, partial(<map_func>, <arguments by name>))
# Start the map and reduce operations .
data = mm.get()
# Do Something with the returned data from MapReduce
```

MapReduce Output

```
INFO<USER.000>  Found  14822 clock instances.
```

The parallel execution using MapReduce operations is much faster than serial execution on a large design

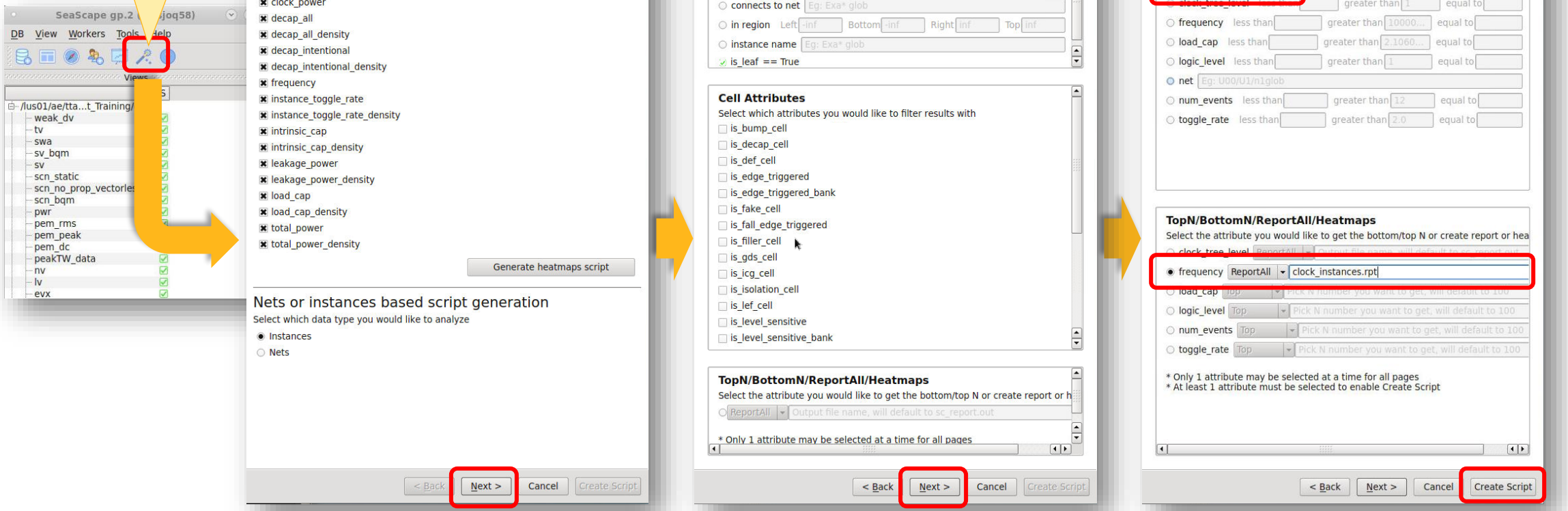
The clock instances from MapReduce are displayed in the GUI



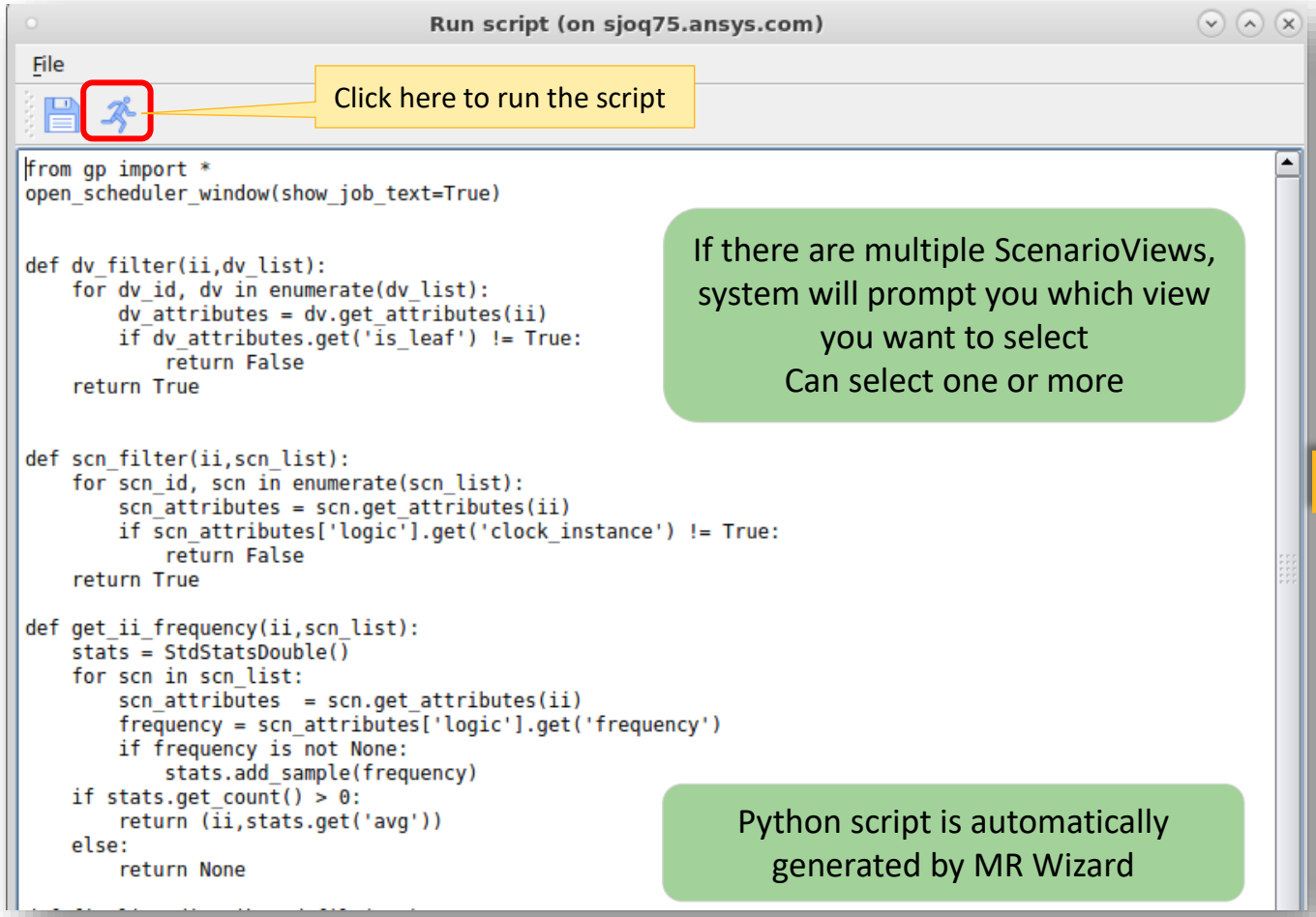
Using MapReduce Wizard To Run Python Scripts

- You don't need to be a Python expert to run MapReduce script in SC
- Use the MapReduce wizard to generate your Python script for most common use cases

Click on Magic Wand icon from console to open Map Reduce Wizard



Using MapReduce Wizard To Run Python Scripts




```
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_36_12_48 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_48_12_60 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_60_12_72 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_72_12_84 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_84_12_96 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_96_12_108 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_108_12_120 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_120_12_132 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_132_12_144 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_144_12_156 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_156_12_168 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_204_12_216 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_216_12_228 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_228_12_240 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_240_12_252 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_252_12_264 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_264_12_276 714285696.0
sc4/wRf_inst/pmem/ClkBuf_inst_Lvl0_0_276_12_288 714285696.0
```

File: clock_instances.rpt

Thank You

Ansys