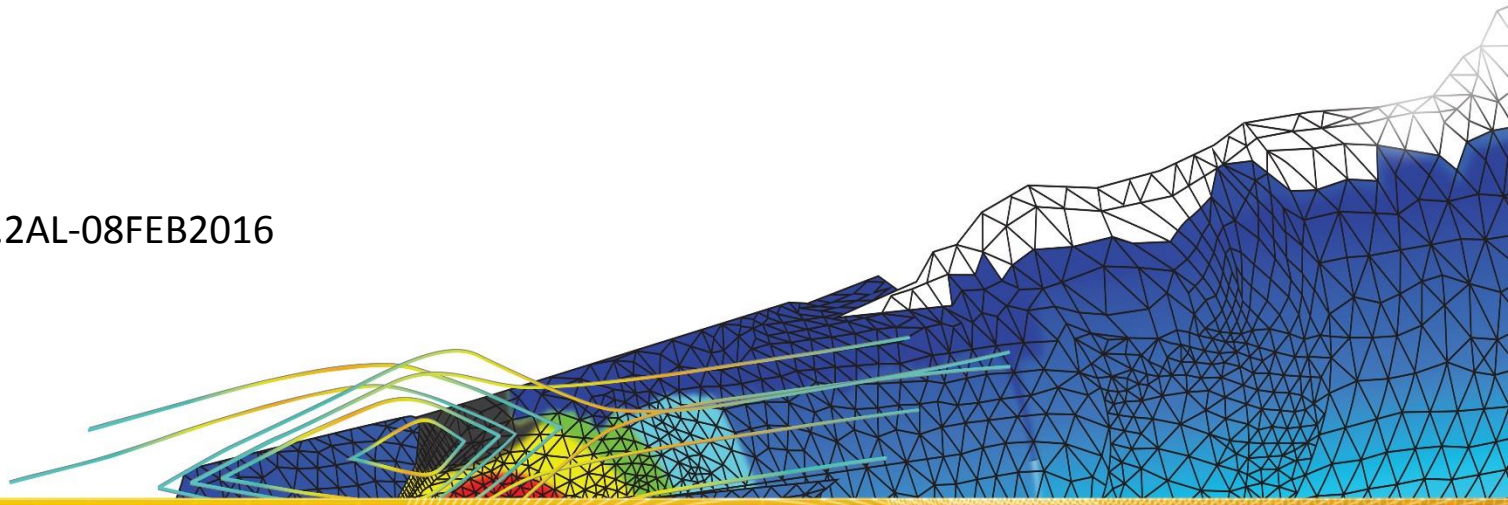
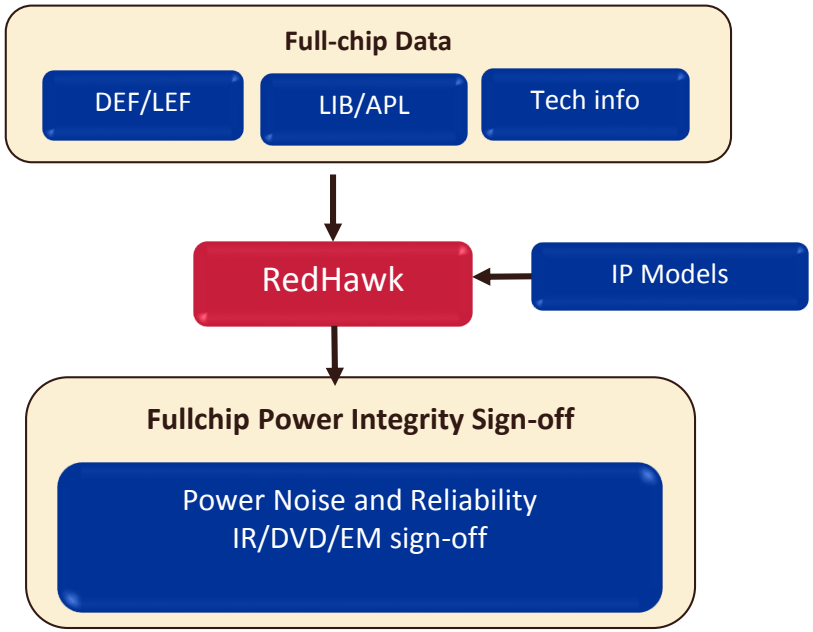
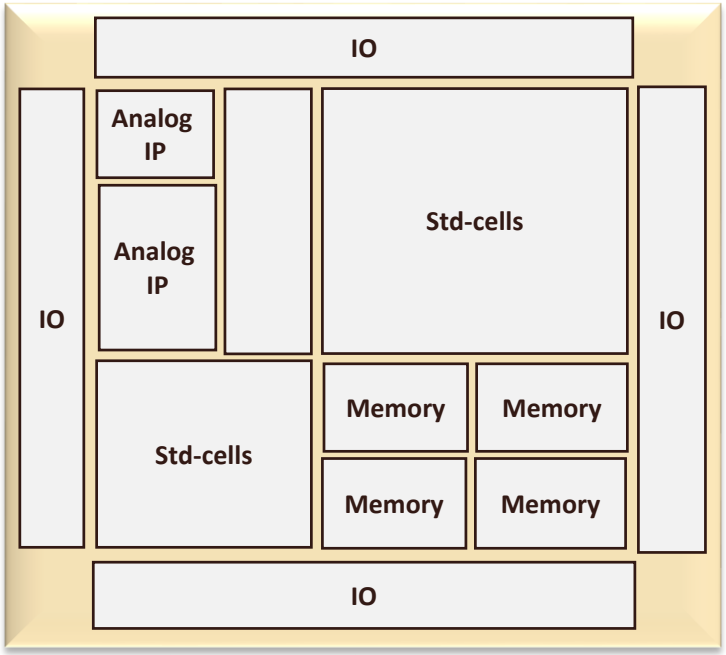
IP Modeling in RedHawk

VERSION: V2.2AL-08FEB2016



Need for IP Modeling

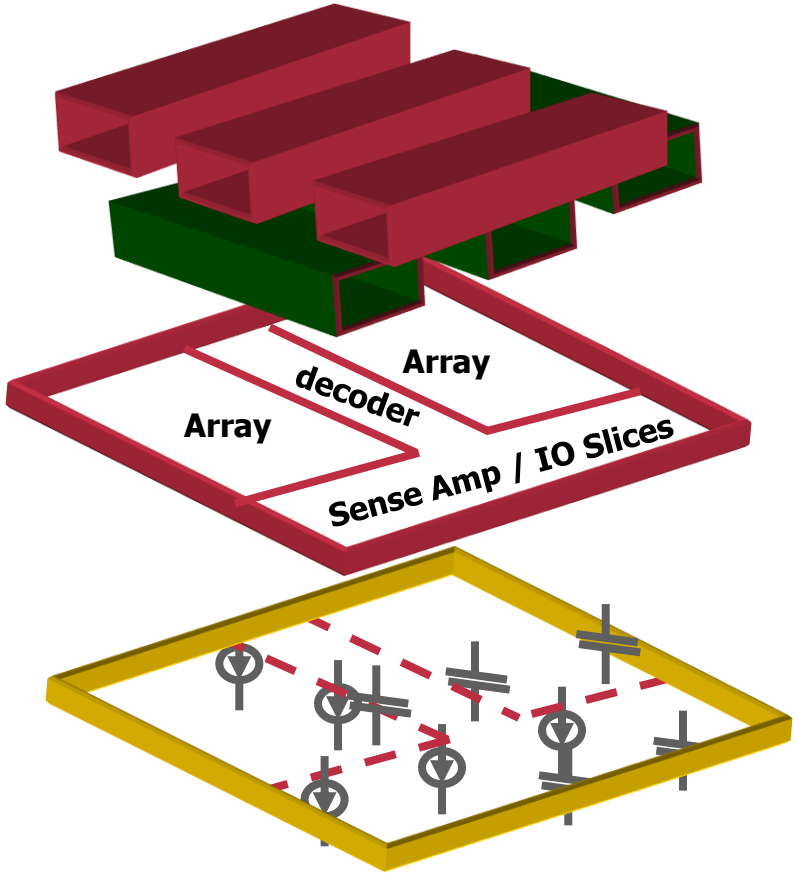




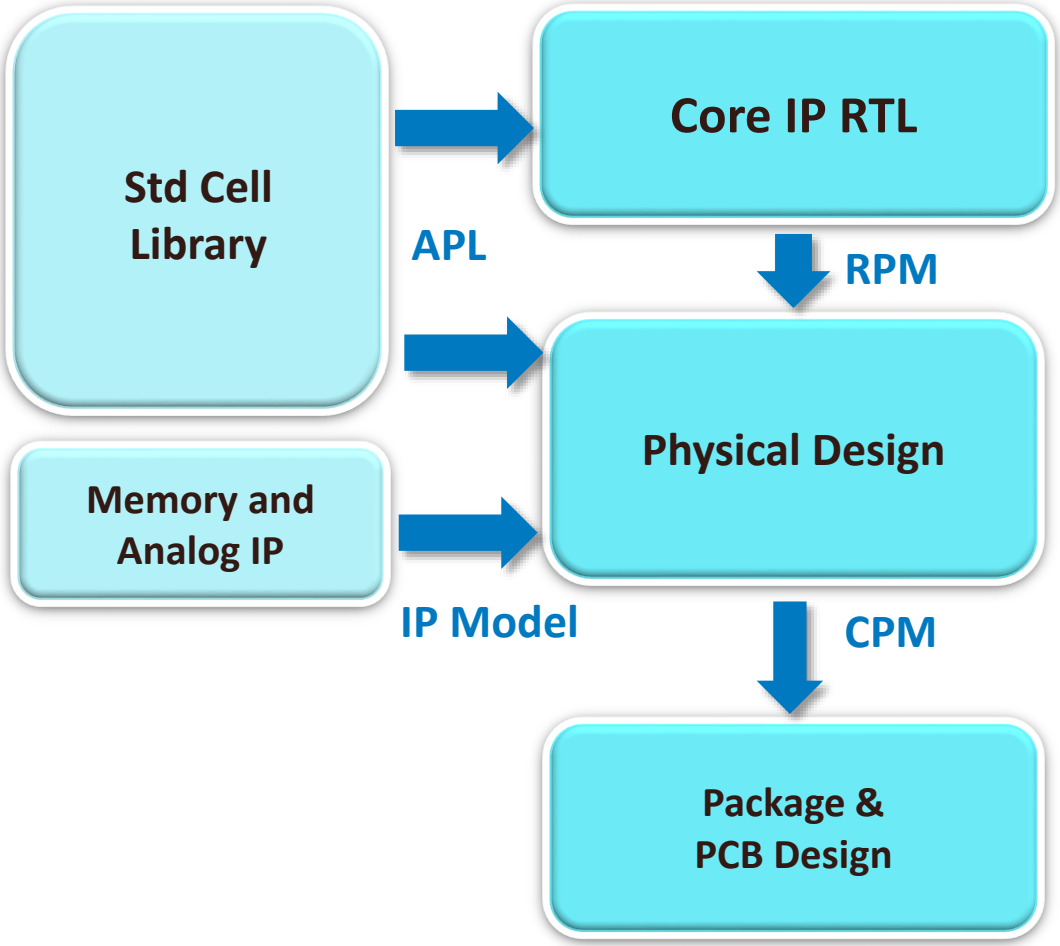
- Increased sharing of core and IP power supply domains
- IP models significantly affect the accuracy of full-chip analysis

IP Modeling Challenges

- How to model power grid inside IP?
- How to characterize current signatures & decaps?
- How to capture the temporal and spatial behaviour of switching activity?

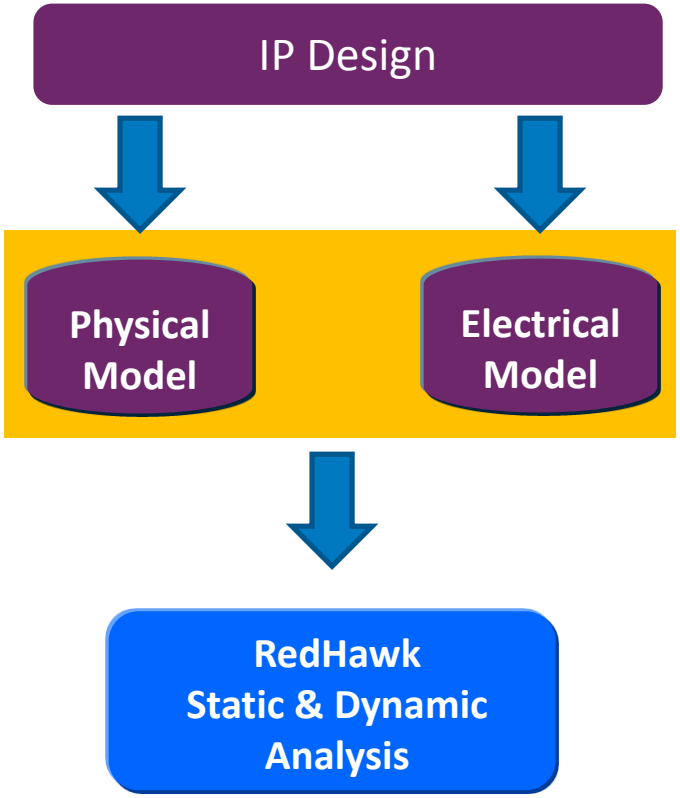


Apache's IP Model Exchange EcoSystem



IP Models for RedHawk

- LEF
- GDS
- DEF/LEF/PRATIO



- .lib
- Switching current profiles
- ESR/ESC
- Leakage current

Physical Modeling Options

Approach	Feature	Typical design application
Lef view	➢ LEF shapes for Power grid ➢ Current/decap at pins	
gds2rh	➢ Full GDSII power grid ➢ Current/decap at pins	➢ RDL layers ➢ IO cells
gds2rh –m	➢ Full GDSII power grid ➢ Spatial distribution of current/decap on lowest metal layer	➢ Memories
Totem (cell_view)	➢ Xtor based spice char ➢ Reduced distribution	➢ CAM/TCAMS/Complex IP blocks/Ios
Totem (mmx_view)	➢ Xtor based spice char ➢ Xtor based distribution	➢ Critical CAMS/TCAMS/RF/IP

Electrical Modeling Options

Approach	Feature	Typical design application
.lib	➢ Triangular current profile ➢ No ESR/ESC	➢ Small/General purpose IOs
AVM/lib2avm	➢ Current profile, cap based on lib/data-sheet tables	➢ Small, Single VDD, RF/SRAM
Sim2iprof + Ace	➢ IP level current profile from spice char ➢ Cap char through ace	➢ Memories/IPs/IOs
Totem (cell_view)	➢ Xtor based spice char ➢ Reduced distribution	➢ CAM/TCAMS/Complex IP blocks/IOs
Totem (mmx_view)	➢ Xtor based spice char ➢ Xtor based distribution	➢ Critical CAM/TCAMS/RF/IP

PHYSICAL MODELING (NON TOTEM)

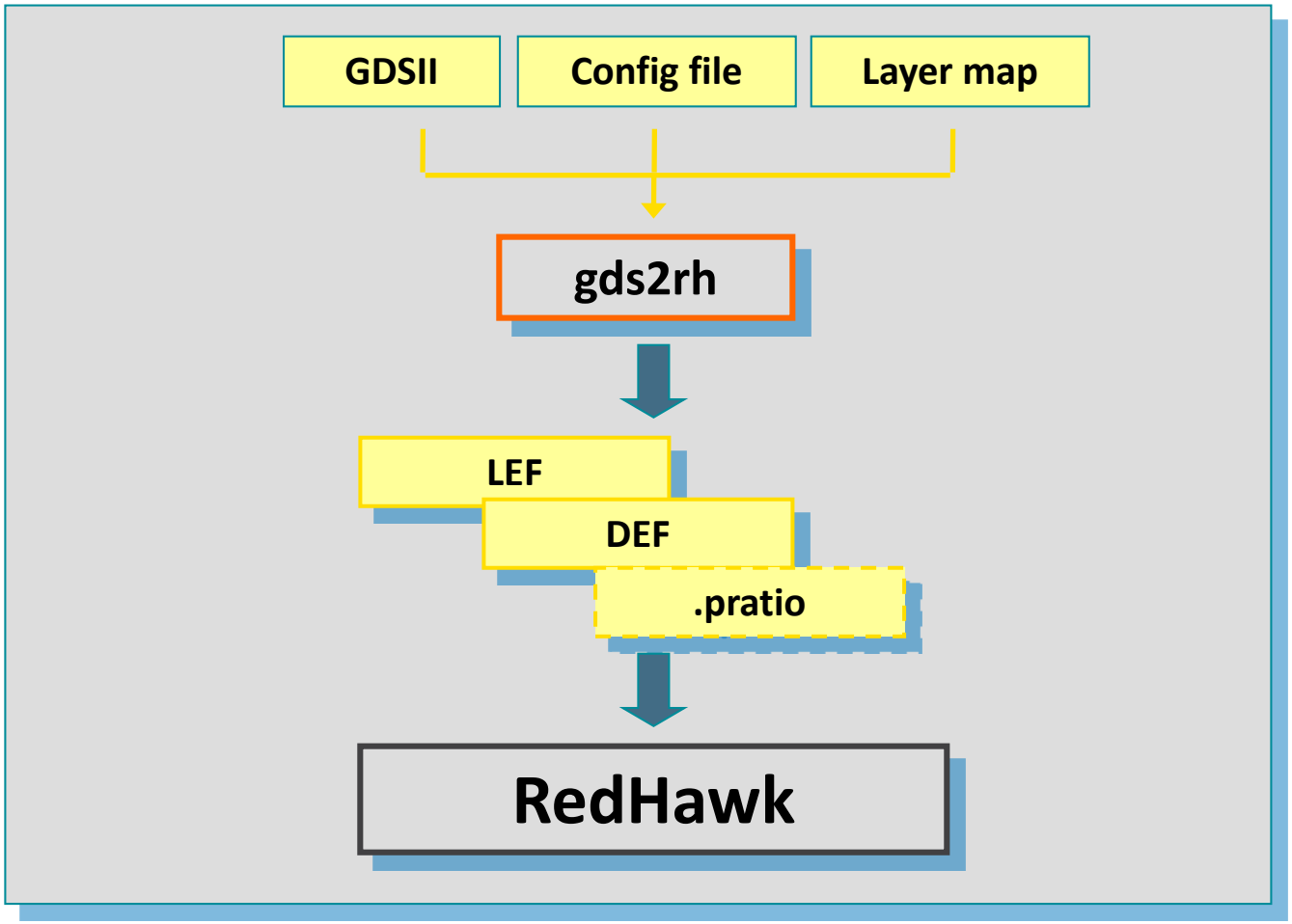
Physical Model Creation

- **RedHawk is primarily a LEF/DEF cell-based flow**
- **Can take in GDS (Graphical Database System) data as input**
- **Translates to DEF format using Apache utilities (gds2rh)**
- **GDS to DEF translations done for:**
  - RDL layers
  - I/O cells
  - memory blocks
  - Custom macros and hard IPs

Physical Modeling (non-Totem) Options

Approach	Features	Typical design application
Lef view	➢ LEF shapes for Power grid ➢ Current/decap at pins	
gds2rh	➢ Full GDSII power grid ➢ Current/decap at pins	➢ RDL layers ➢ IO cells ➢ IPs/Macros
gds2rh –m	➢ Full GDSII power grid ➢ Spatial distribution of current/decap on lower-most metal layer	➢ Memories

gds2rh Input/Output



Usage: *gds2rh [-m] <gds2rh_configuration_file>*

gds2rh Methods

- **gds2rh <configuration file>**

  Use for:

  – RDL layers

  – I/O cells

- **gds2rh –m <configuration file>**

  Use for:

  – memory blocks

  – Custom macros and hard IPs

gds2rh
Input and Output Data

- **Input data**
  - Configuration file
  - GDS file (individual or combined)
  - LEF file
  - Layer mapping file

- **Output data [create in a central area for all groups to use]**
  - <block_name>.def
  - <block_name>_adsgds.lef

gds2rh Configuration File

Extract for one cell

TOP_CELL top_cell

GDS_MAP_FILE <layermap>

LEF_FILE <lef>
If specified, creates a LEF file along with DEF

VDD_NETS {
VDD
}
GND_NETS {
VSS
}

Extract for many cells

TOP_CELL {
<cell 1>
<cell 2>
..
}

GDS_MAP_FILE <layermap>

LEF_FILE <lef>
If specified, creates a LEF file along with DEF

VDD_NETS {
VDD
}
GND_NETS {
VSS
}

Creating GDSII Layer Map File

Syntax

<Layer_name> <Layer_type> <Layer_number> <Text_layer_number>

If no text layer number is available, it can be replaced by "-". An example layer map file is shown below. Please refer to manual for more details.

Example

```
#Layer_name   Layer_type   Layer_number   Text_layer_number
 METAL1         m             1:5;10:0        11:1;13
 VIA1           v             2;12            - (10 10)
 METAL2         m             13:0;15:50      -
 VIA2           v             3:12;3:43       -
```

In the above specification,

- **Layer_type can be 'm' for metal, 'v' for via**
- **Layer_number and Text_layer_number can be specified for several layer-datatype pairs and for several layer numbers separated by ';' (semi-colon).**
- **Layer and datatype are separated by ':' (colon).**
- **Text_layer_number can be ignored by inserting a dash '-'.**

ANSYS

gds2rh –m
Input and Output Data

- **Input data**
  - Configuration file
  - GDS file (individual or combined)
  - LEF file
  - Layer mapping file

- **Output data [create in a central area for all groups to use]**
  - <block_name>_adsgds.def
  - <block_name>_adsgds.lef
  - <block_name>_adsgds.pratio

gds2rh -m Configuration File

TOP_CELL top_cell
GDS_MAP_FILE <layermap>

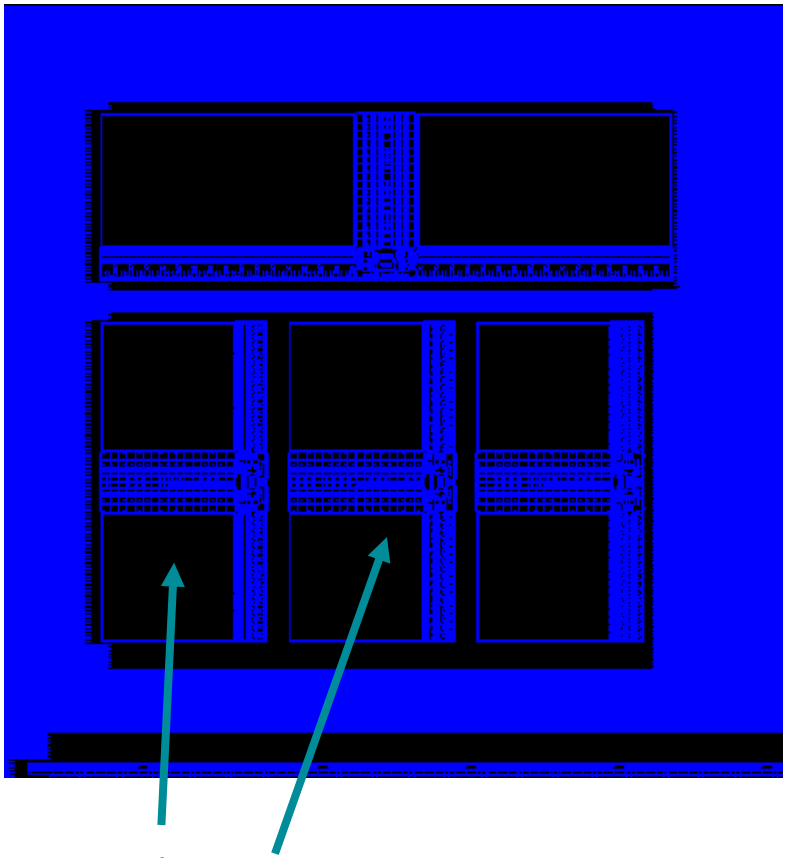
LEF_FILE <lef>
If specified, creates a LEF file along with DEF

VDD_NETS {
VDD
}
GND_NETS {
VSS
}

MEMORY_BIT_CELL

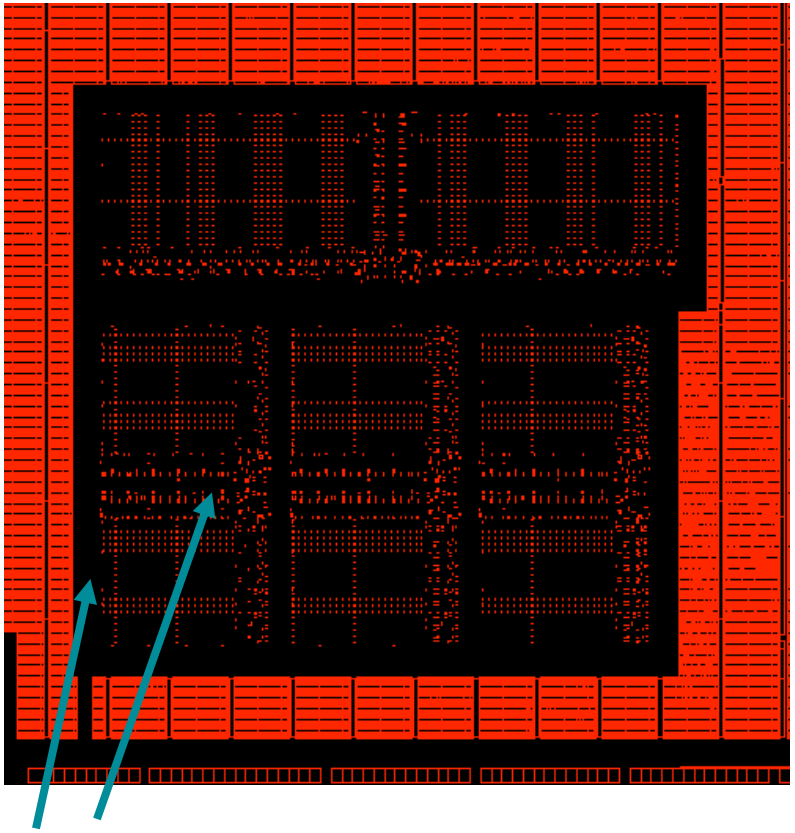
Physical Model Creation

Controlled Power Grid Extraction



No met1/2 inside array region but present elsewhere

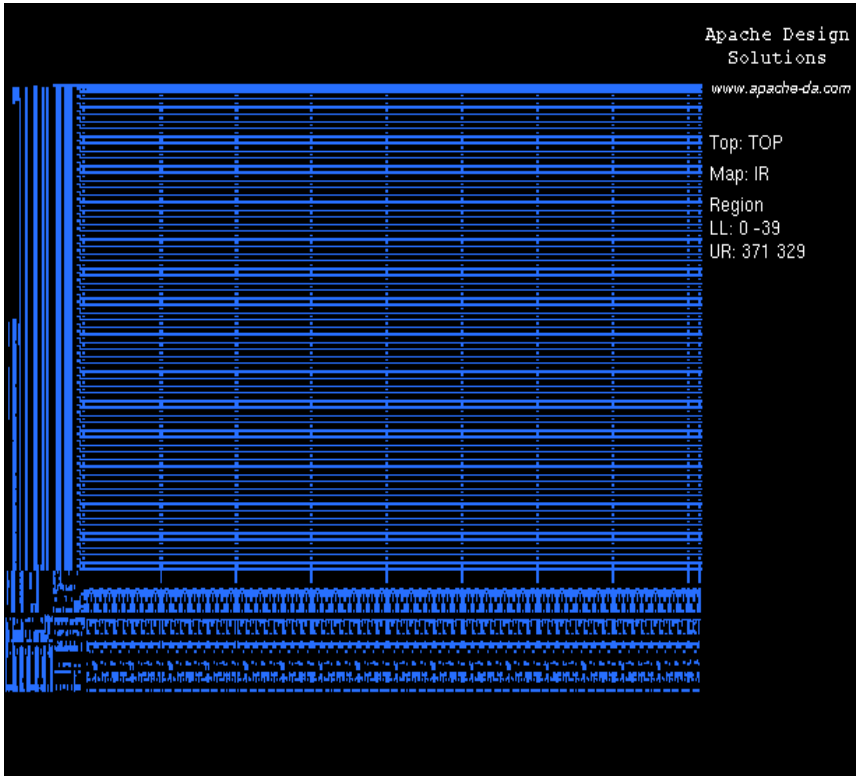
Current source / decap insertion



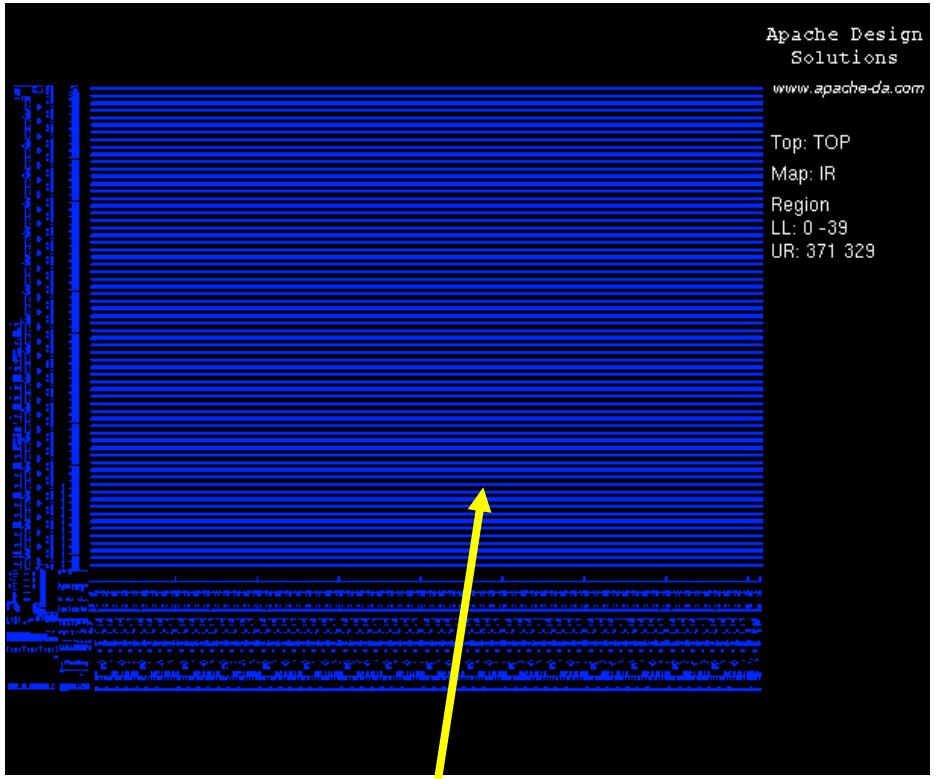
Instances (decaps and current sources) inserted in the memory block

Physical Model Creation

Metal2



Metal1



Consider Metal1 and above everywhere

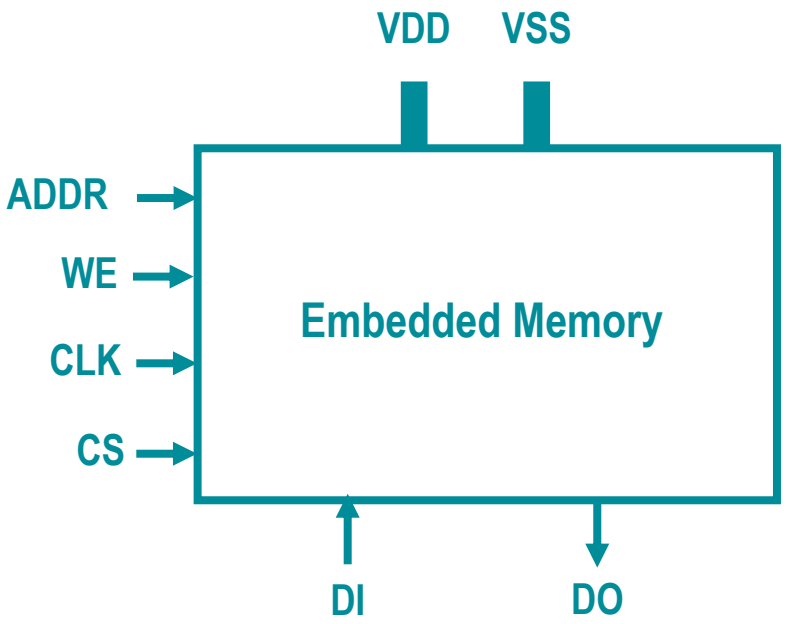
Consider Metal2 and above in memory array regions => metal1 has straps over array

ELECTRICAL MODELING (NON TOTEM)

Electrical Models (non-Totem) Options

Approach	Feature	Typical design application
.lib	➢ Triangular current profile ➢ No ESR/ESC	➢ Small/General purpose IOs
AVM/lib2avm	➢ Current profile, cap based on lib/data-sheet tables	➢ Small, Single VDD, RF/SRAM
Sim2iprof + Ace	➢ IP level current profile from spice char ➢ Cap char through ace	➢ Memories/IPs/IOs

Memory Power Model Basics



Current waveform VDD

Current waveform VSS

One current waveform per power port
(i.e. total current profile for the memory macro)

Memory Power Model

- **For each memory master, each voltage, and each vector input subset:**
  - Current profile
  - Intrinsic decap data
  - Leakage current value
- **Typical vector subset**
  - Write operation
  - Read operation
  - NOP (stand-by) operation

Memory Characterization

Characterization for current profile creation

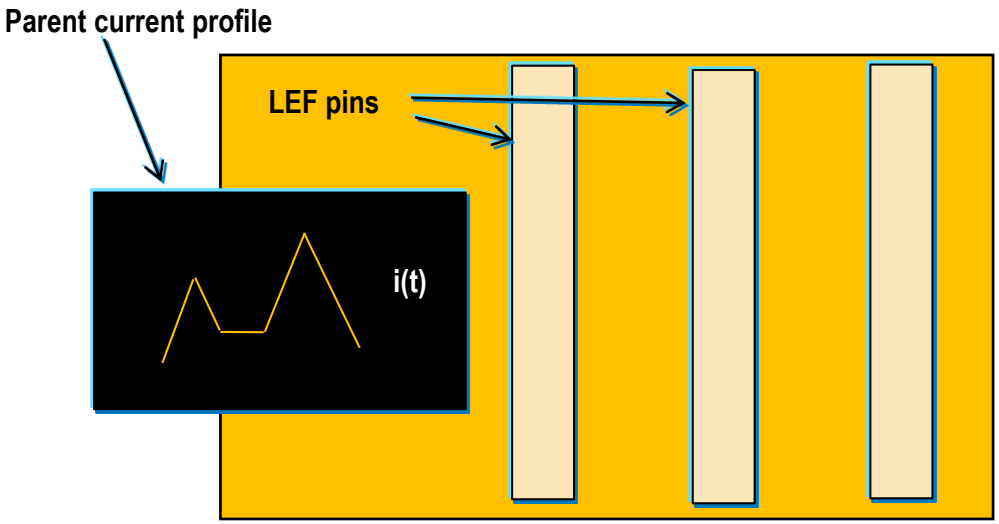
Triangular profiles	Lib or user specified power
Data sheet based (AVM)	Triangular / Trapezoidal profiles
Translation of user data (sim2iprof)	Functional statement based translation of pre-characterized current profiles

Apache Virtual Model (AVM)

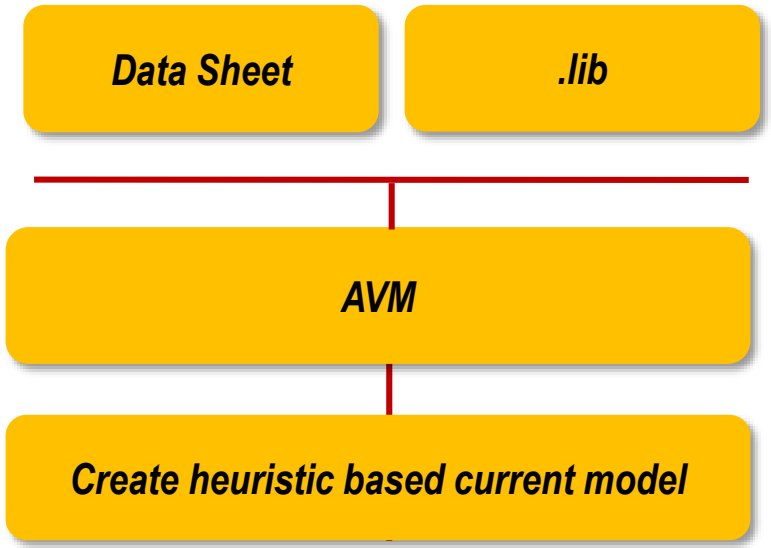
- **Apache Virtual Model (AVM) utility offers a rule based approach for generating current profiles for memories within the RedHawk flow**
- **AVM offers a desirable compromise between the detailed accuracy of sim2iprof and Liberty based power**
- **AVM uses the timing and power information from data sheets or LIB**
  - For LIB, use “LIB2AVM 1” in GSR (Default ON)
- **Generates current waveforms that contain two triangular waveforms - one for memory decode and the second waveform for memory read/write**
- **Waveform shape can optionally be trapezoidal too**

AVM Model

- **AVM (Apache Virtual Model)**
  - Current profile generated based on datasheet
  - Trapezoidal or dual-triangle waveforms for different families of memories
  - Current profile controlled by switching charge, timing data and memory type
  - Current is assigned to LEF pins of macro cell



Electrical model creation in AVM Flow



AVM *(Cont'd)*

- **Sample AVM Configuration file:**

```
<name of memory | register file | block | IP>
{
EQUIV_GATE_COUNT <integer number>
MEMORY_TYPE      <SRAM|IP|RegFile|..>
VDD              <voltage value>
Cpd_read         <effective charging cap for read operation>
Cpd_write        <effective charging cap for write operation>
Cpd_standby      <effective charging cap for standby operation>
tsu              <setup time>
ck2q_delay       <access time>
tr_q             <average input rise time>
tf_q             <average input fall time>
Cload            <average output load>
PROCESS   [ BC | WC | TC | SS | FF | TT ]
Leakage_I        [user specified leakage current in A]

WAVEFORM_TYPE    [trapezoidal | triangular (default)]

Peak_I_write/read/standby
                 [user specified peak current value]
Peak_T_write/read/standby
                 [user specified time at which peak happens]

C_decap          [user specified decap]
}
```

AVM *(Cont'd)*

- **AVM config file generation**
  - RedHawk automatically generates AVM config file for memories using the timing and power info from their .libs (Path: adsPower/avm.conf in run directory)
  - User can choose to provide his own config file for some or all memories in design in which case they take precedence. GSR Keyword:

```
APL_FILES {
<path/to/avm/config/file> avm
}
```

  - Besides, GSR keyword LIB2AVM can be set to '0' to turn-off automatic generation of AVM config file by RedHawk.
    - **LIB2AVM [0/OFF | 1/TRIANGULAR |2 /TRAPEZOIDAL]**

AVM *(Cont'd)*

- **AVM usage**
  - RedHawk internally generates current profile (vmemory.current) and cdev (vmemory.cdev) for memories by running utility called avm on the AVM config file.
  - AVM can also be run outside (avm <avm_configuration_file>) to create current and cdev files which can be imported in RedHawk through GSR

```
   APL_FILES {
<path to vmemory.current> current_avm
<path to vmemory.cdev> cap_avm
}
```

sim2iprof

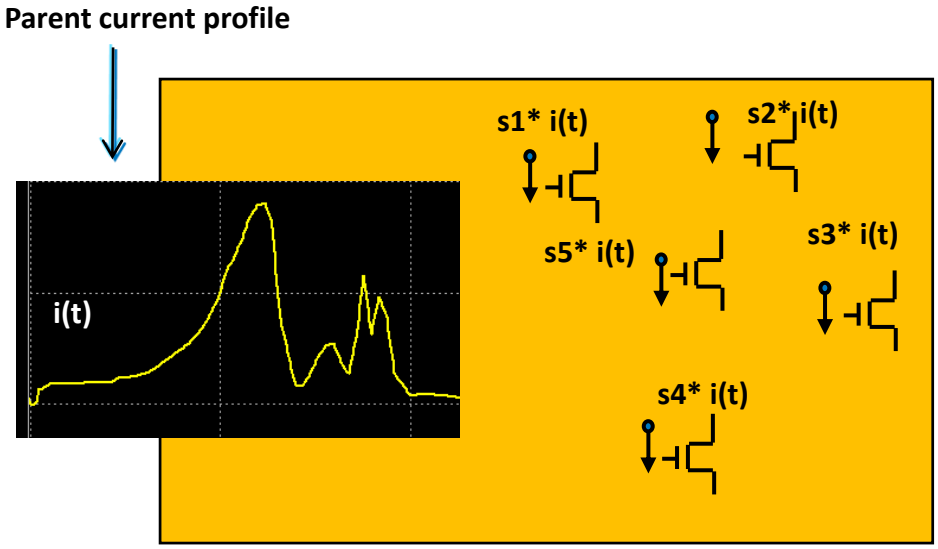
- **sim2iprof can be used to obtain characterization data for memory with a faster run time**

- **sim2iprof utility uses third-party simulation output files, such as *fsdb, .out, tr0,* or *pwl* format files as inputs to obtain Read/Write/Standby mode data for memories**

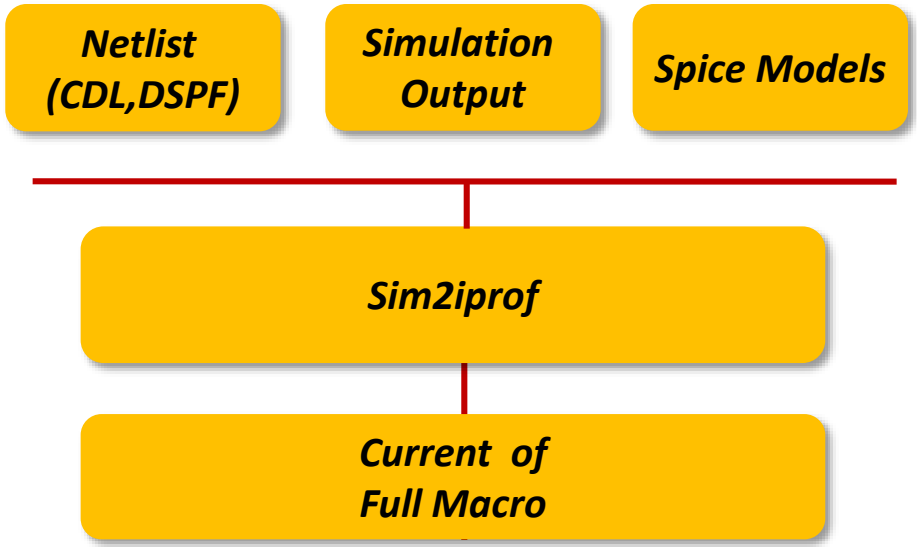
- **Generates current profiles in *a <cell>.current* file for RedHawk power analysis.**

sim2iprof

- **SIM2IPROF**
  - Accurate current profile of macro is captured from spice simulation
  - Current distribution is based on via/contact densities in design
  - Scaled currents i(t) is modeled at each pin locations



Sim2iprof Flow



Running sim2iprof

- **To run the sim2iprof utility, use the following invocation:**
  - sim2iprof <config_file> [-o <current_output_file>] [-avm <avm_config_file>]

- **The Vdd current waveform is extracted and converted to RedHawk *<cell>.current* format.**

- **To get decap information, you can specify the '-avm' option, which runs the AVM utility to get decap values, ESR and leakage power and write the data to the *<cell>.cdev* file**

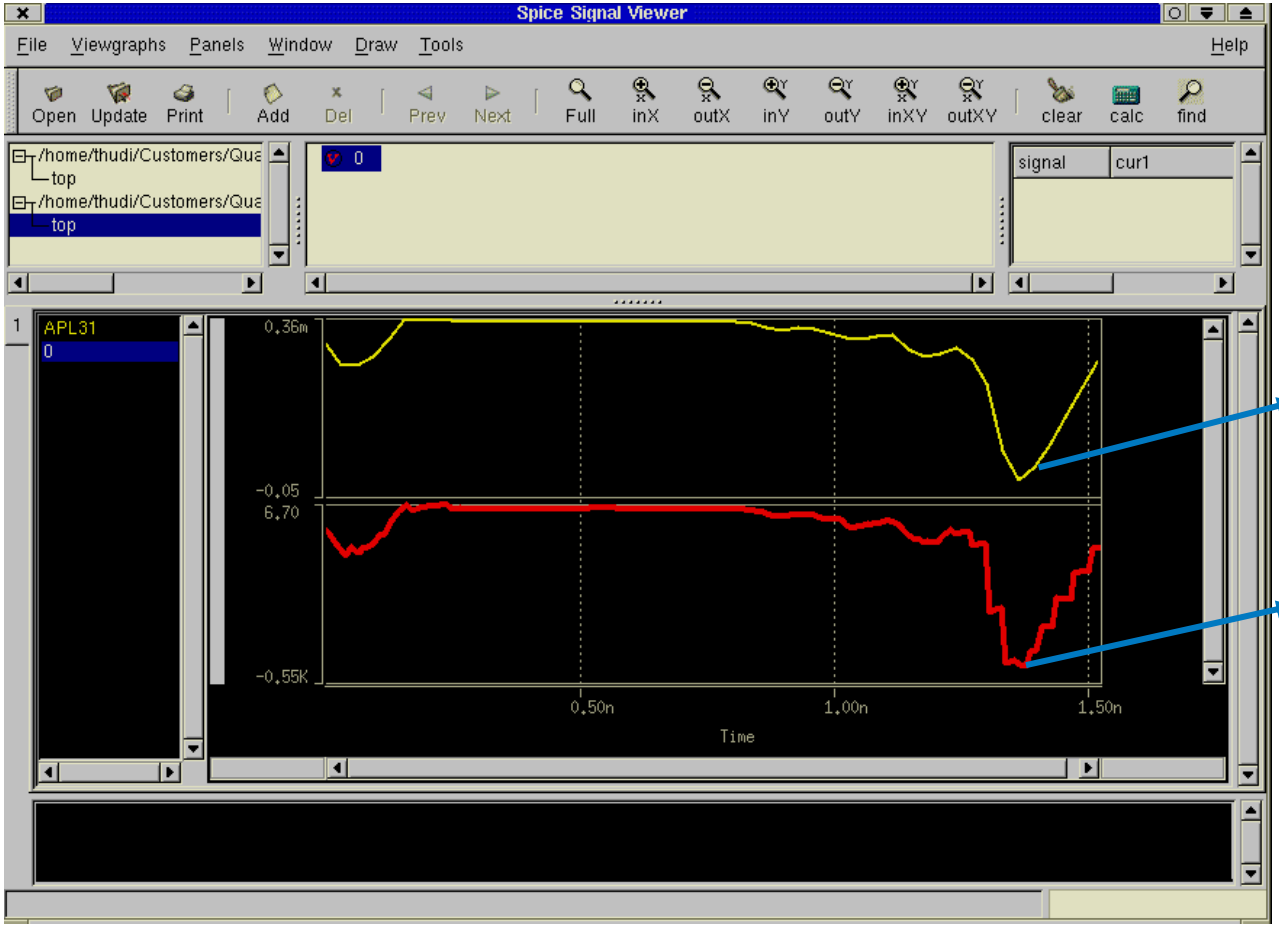
sim2iprof

- **Sample configuration file**

```
#-- sim2iprof configuration file generated by
sim2iprof_setup.pl on Thu Feb 1 15:23:41 PST 2007
CELL (sram128x32) {
FILENAME {
# <filename> [<vdd1>=<vdd1_v2> <vdd2>=<vdd2_v2>...]
dram.s0.fsdb vdd=1.15 vddo=1.05
dram.s1.fsdb vdd=1.1 vddo=1.0
}
SLEW {
11ps
}
LOAD {
15fF
}
CDEV {
vpwr vgnd {
C0 = 1.0p
C1 = 2.0p
R0 = 400.21
R1 = 395
LEAK0 = 1.0e-6
LEAK1 = 4.2uA
}
ivdd vgnd {
C0 = 3.0p
C1 = 5.0p
R0 = 621
R1 = 385
LEAK0 = 2.0e-6
LEAK1 = 4.2uA
}
} // end of CDEV
} // end of cell
SIM_TIME {
READ 1e-12
WRITE 1e-12
STANDBY-H 1e-12
STANDBY-L 2e-12
}
VDD_PIN vpwr ivdd
VSS_PIN vgnd
DURATION 500e-12
```

Optional

Sample current profile

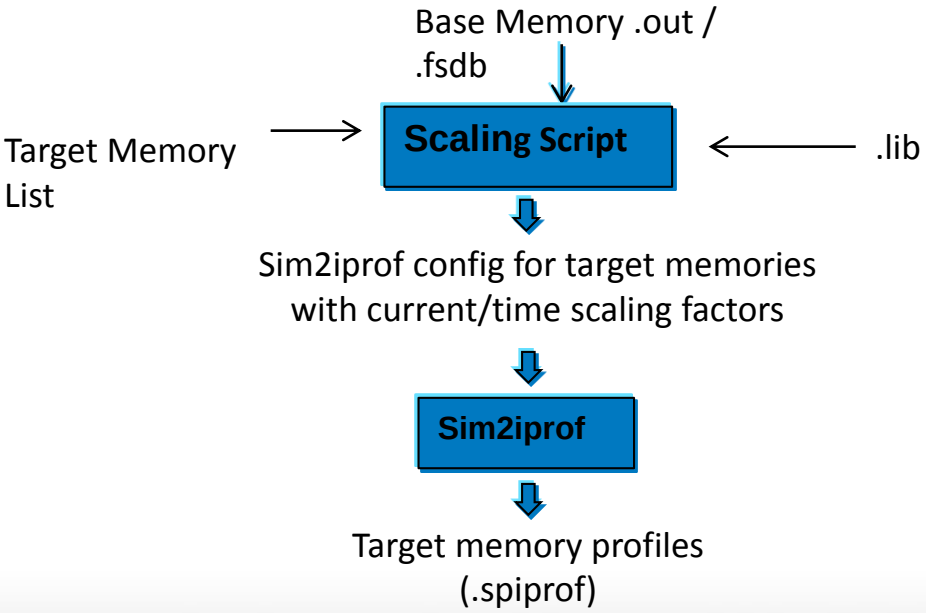


From third party tool – Input to sim2iprof

From SIM2IPROF's cell.current

Sim2iprof: Energy/Time scaling based Current profile

- **Simulation data may be available for few memories**
- **Sim2iprof allows scaling available profile in time/magnitude to create current profile for other memories with similar architecture**
- **Current (magnitude) and Time scaling factors can be derived from lib energy and delay values using script**



Sim2iprof: Scaling Flow contd..

- **Script usage:**

perl scale_mem_profile.pl -baseName <memoryName> -fsdbFile <fileName.fsdb> -targetMemList <MemListFile> -blib <bmemlib> -tlib <tmemlib> -configDataFile <file>

-baseName : Name of memory that correspons to base current waveforms (.fsdb/.out/.ta0 file).
-targetMemList : File containing target memory list
-blib : Base memory liberty file for target memory
-tlib : Target memory liberty file for target memory
-fsdbFile : Simulation output file (fsdb/out) containing base PG currrent waveforms.
-configDataFile : Filename containing sim2iprof config data (optional)

Sim2iprof: Scaling Flow contd..

The config data file can contain following :

```
CUSTOM_STATE_SIM_TIME {
c00 "(!cs_n & we_n & scan_n & clk)" clk 220n 230n
c01 "(!cs_n & !we_n & scan_n & clk)" clk  100n 110n
c10 "(!scan_n & clk)" clk  620n 630n
standby_trig "(cs_n & scan_n & clk)" clk 60n 70n
}

SPICE2LEF_PIN_MAP {
vvddx_sram vddx_sram
vvssx_sram vssx_sram
}

VDD_PIN  vvddx_sram
VSS_PIN  vvssx_sram

IPROF_SAMPLING_MODE accurate
LEAKAGE 0
DATAVERSION 7v1_pwl
peak_tolerance 0.05
width_tolerance 0.1
RATIO 0.1
```

Sim2iprof: Scaling Flow contd..

- **Example:**

```
perl  scale_mem_profile.pl   \
        -baseName   sram1111 \
        -fsdbFile   base_mem/power_scan_capture.out    \
        -targetMemList   tmem.list    \
        -blib   ../base_lib_link  \
        -tlib    ../target_lib_link   \
        -configDataFile   ../../simconfig.data
```

tmem.list:

```
sram1111_12
sram1111_13
sram1111_14
sram1111_15
```

Multi-State support

- Multi-state provides the ability to have more than four current profile modes
- AVM/sim2iprof/APLMMX provide support for multi-state current profile capturing for custom and/or memory blocks
- Power calculation and Voltage drop analysis honors both the state in GSC (Global Switching Constraint)or the state in VCD-based mode

Multi-state current profile generation

- **Following are the new keywords required for multi-state current profile generation:**
  - Sim2iprof: CUSTOM_STATE_SIM_FUNC or
    CUSTOM_STAE_SIM_TIME
  - AVM: CUSTOM_STATE_FILE
- **Sample config files in next few slides**

Sample multi-state AVM config file

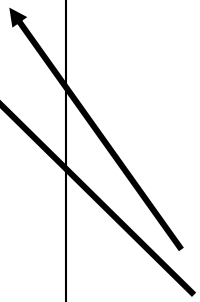
```
ram64x8
{
EQUIV_GATE_COUNT  400
MEMORY_TYPE MEMORY
PROCESS XX
VDD_PIN VDD
GND_PIN VSS
VDD   1.1
WAVEFORM_TYPE triangular
CUSTOM_STATE_FILE ram64x8.state
Cpd M0 {
VDD VSS 15pF
}
Cpd M1 {
VDD VSS 20pF
}
Cpd M2 {
VDD VSS 10pF
}
Cpd M3 {
VDD VSS 10pF
}
C_decap {
VDD VSS 1pF
}
Leakage_i  {
VDD VSS 0.1mA
}
tsu   400ps
ck2q_delay  1.37613ns
tr_q   0.688334ns
tf_q   0.71125ns
Cload  0.104pF
DATAVERSION 7v1
}
```

Specify custom state file (see next slide)

Specify Cpd data for multiple states in avm config file

Sample multi-state AVM config file *(Cont'd)*

```
cell ram64x8 {
M0 "(((CEN & !WEN)) & CLK)" "CLK" "NA"
M1 "(((!CEN & WEN)) & CLK)" "CLK" "NA"
M2 "(((CEN & WEN)) & CLK)" "CLK" "NA"
M3 "(((!CEN & !WEN)) & CLK)" "CLK" "NA"
}
cell dup_ram64x8 {
M0 "(((CEN & !WEN)) & CLK)" "CLK" "NA"
M1 "(((!CEN & WEN)) & CLK)" "CLK" "NA"
M2 "(((CEN & WEN)) & CLK)" "CLK" "NA"
M3 "(((!CEN & !WEN)) & CLK)" "CLK" "NA"
}
```



Specify Custom states and corresponding equations in custom state file

Sample Multi-State sim2iprof configuration file

```
CELL REGION1 {
        FILENAME {
        ./input_REGION1.pwl vdd_i=1.08
        }
        CDEV {
                vdd_i vss {
                C0 = 400p
                C1 = 400p
                R0 = 450
                R1 = 450
                LEAK0 = 0.001
                LEAK1 = 0.001
                }
        }
}
CUSTOM_STATE_SIM_TIME {
M0 A&B CLK 0e-12 5140e-12
M1 !A&B CLK 5160e-12 10280e-12
M2 A&!B CLK 10300e-12 15420e-12
M3 !A&!B CLK 15440e-12 20580e-12
}
VDD_PIN vdd_i
VSS_PIN vss
CURRENT_SCALING_FACTOR 2
```

Multi-state current profile Usage

- **For Vector less analysis: One can specify an instance to be switching in any of the valid states (state for which current profile is generated) using GSC file. Sample GSC_FILE contents:**
  - #<inst_name> <state symbol>
  - Instance1 WRITE
  - Instance2 M0

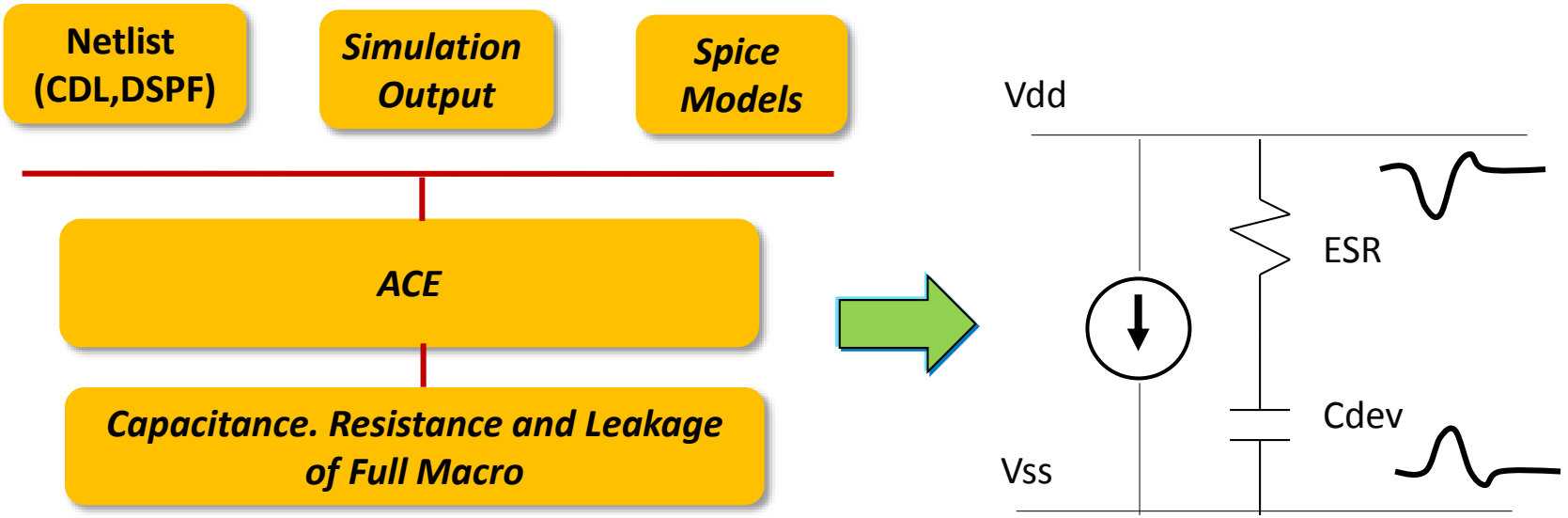
- **For VCD based analysis, activity on all control pins associated with states are automatically tracked (from VCD) inside RH; Whenever a valid event (state) occurs as read/write/etc, corresponding current profile is automatically used.**

ACE

- **Estimates the P/G pin intrinsic and intentional decap values for all types of memory through spice simulation**
- **uses a unique tracing algorithm to find the P/G network in a cell and generates various types of capacitance and effective resistance values for the cell in APL format, as well as an estimate of leakage current**
- **also traces R elements in power/ground nets using DSPF (Detailed Standard Parasitic Format) information (*|NET, *|I ...) to identify relevant capacitance**
- **can recognize header/footer switches embedded inside a memory or a functional block**
  - especially useful for low-power design

ACE Flow

Electrical model creation in ACE Flow



ACE config File

- **Sample ACE configuration file settings are shown below:**

```
EXTERN_POWER_NETS vdd
INTERNAL_POWER_NET  Xblock1.net_int Xblock2.net_int
VP_PAIRING (Xblock1.net_int  vdd) (Xblock2.net_int  vdd)
EXTERN_GROUND_NETS  vss
VDDVALUE vdd 0.85
switch_sub SWITCH
```

```
INCLUDE ./input_data/models.inc
SUBCKT              ./input_data/spi
```

Running ACE Characterization

- **The command for invoking ACE from a UNIX command line is:**
  - ace [-d] [-o <outfile>] [-toggle_rate <On_fraction>] [-pwc] <cellname>.smin

- **ACE generates three files in the APL result directory:**
  - <cellname>.mcap - APL's CDEV file format (pin-based capacitance) to be imported into RedHawk
  - <cellname>.ace.mmx - more detailed pin capacitance data (intentional, intrinsic, parasitic) to be imported into MMX
  - <cellname>.ace.decap - detailed intentional decap information

Summary of Library Creation Activities (Non -Totem)

Memory Physical Views

Creation of DEF views of memory and other custom blocks

Utility	Input Data	Output Data
gds2rh -m	Configuration file GDS file Layer mapping file LEF view	<macro_name>_adsgds.def <macro_name>_adsgds.lef <macro_name>_adsgds.pratio

I/O Cell Physical Views

Creation of DEF views of I/O cells

Utility	Input Data	Output Data
gds2rh	Configuration file GDS file Layer mapping file LEF view	<macro_name>.def <macro_name>_adsgds.lef

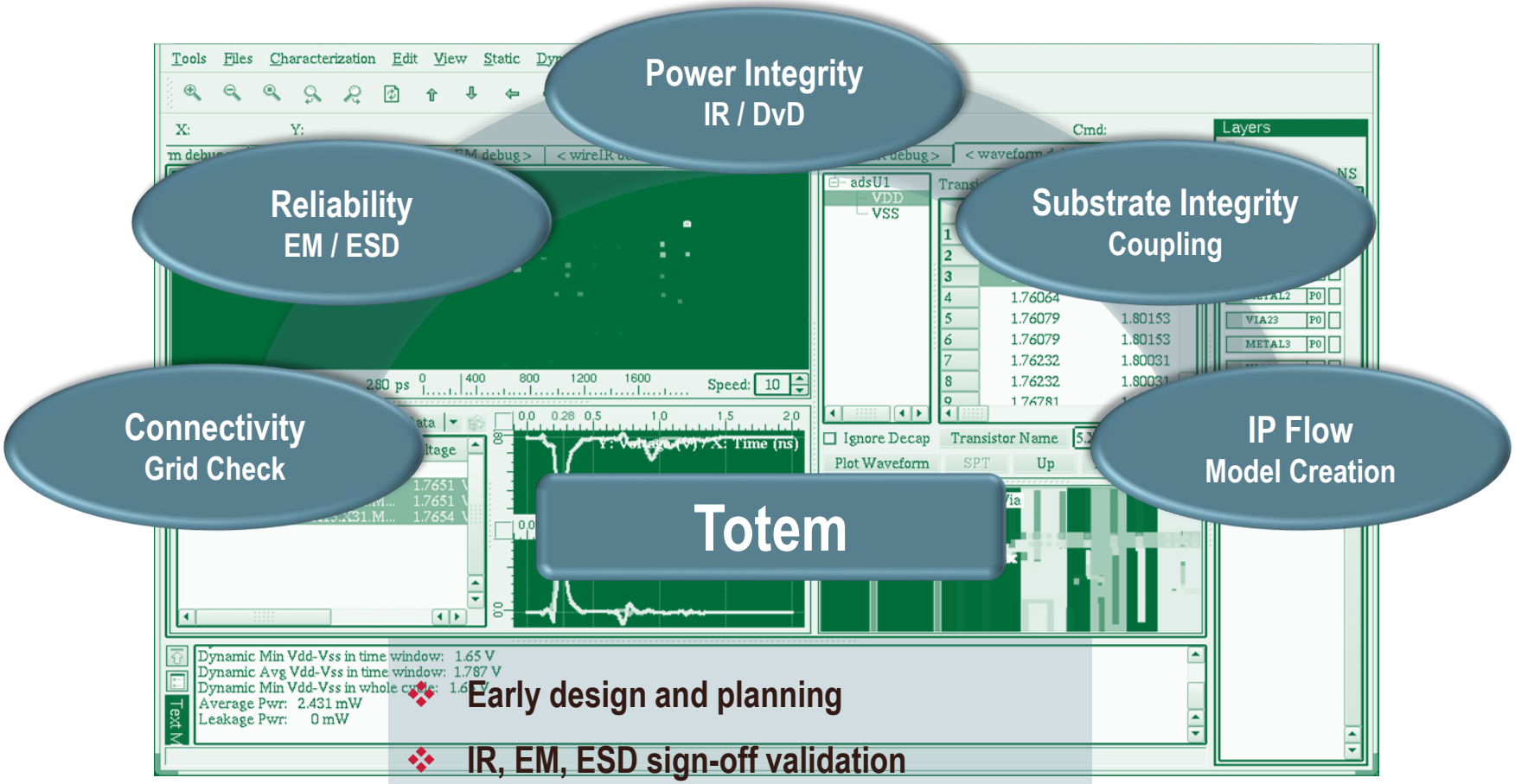
Electrical Models for Memory cells

APL Library Characterization

Utility	Input Data	Output Data
sim2iprof	Functional statements (.lib) Spice characterization data Decap/Leakage estimates	cell.spcurrent cell.cdev
AVM	Datasheet parameters avm.conf	None for user
ACE	Netlist (CDL,DSPF) Simulation Output Spice Models	cell.mcap cell.ace.mmx cell.ace.decap

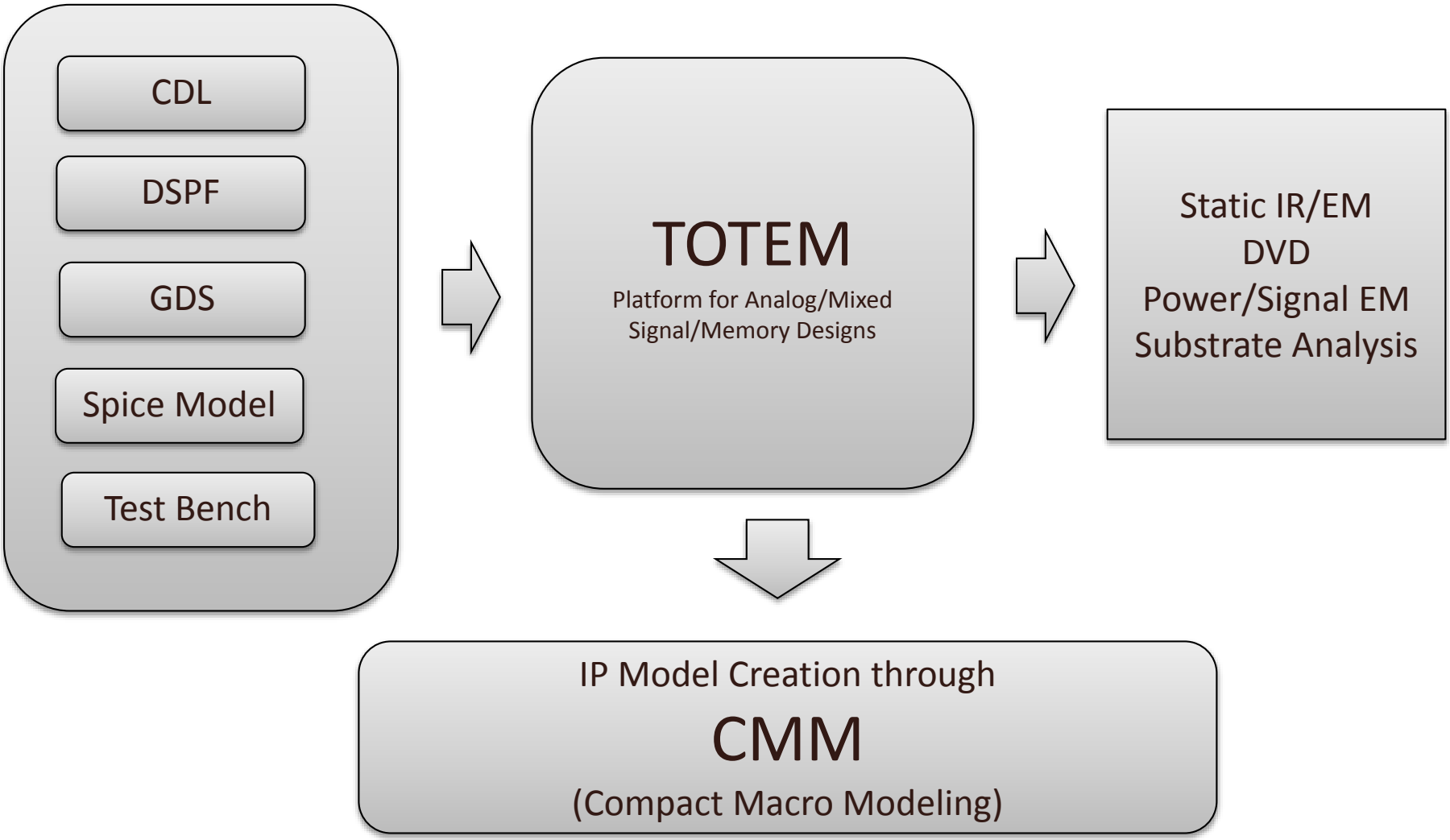
TOTEM CMM
(Transistor level Integrated "Electrical + Physical" Model)

Totem™: Power Noise and Reliability Analysis for Custom and Mixed Signal Designs



- Early design and planning
- IR, EM, ESD sign-off validation
- Substrate noise impact modeling

Totem Transistor Level IP Modeling



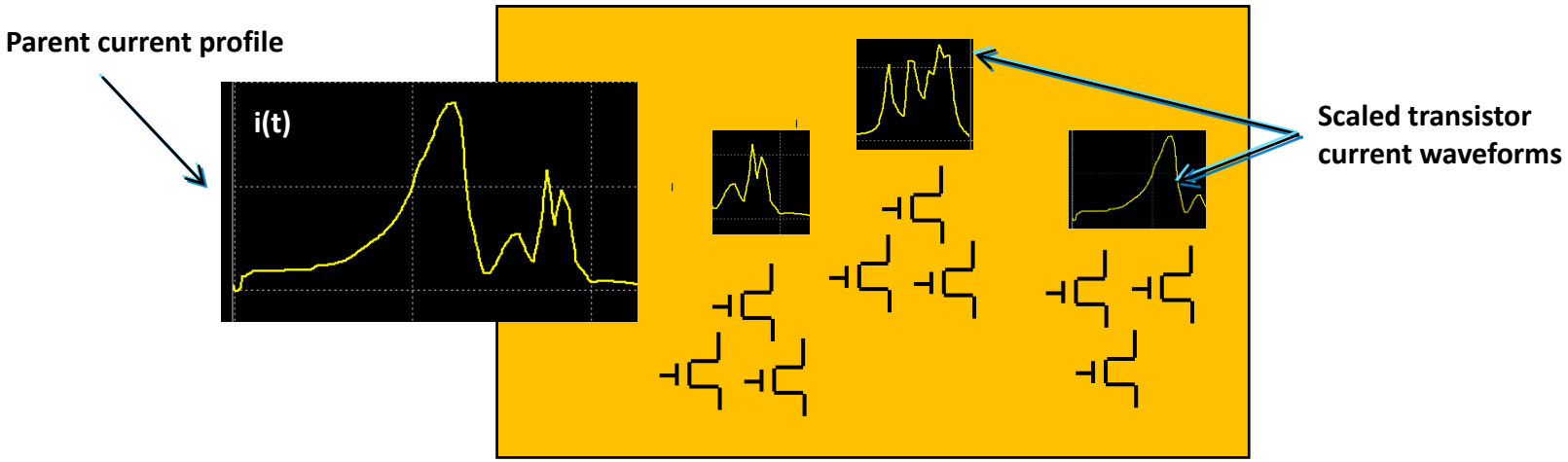
Totem CMM

- **Totem CMM or 'Custom Macro Model': A compact optimized power model that contains both electrical and physical data**
- **Two types of CMM models can be generated**
  - Totem – cell_view
  - Totem – mmx_view
- **Totem 'cell_view' preserves transistor switching/toggling information based on peak current demand of each transistor during simulation**
  - Optimized for runtime for full-chip SoC analysis
  - Improved runtime and good accuracy at SoC level analysis
- **The Totem 'mmx_view' preserves detailed transistor-level switching information for each device**
  - Optimized for accuracy for IP validation and selective SoC analysis
  - Runtime penalty on SoCs due to detailed transistor behavior

ANSYS®

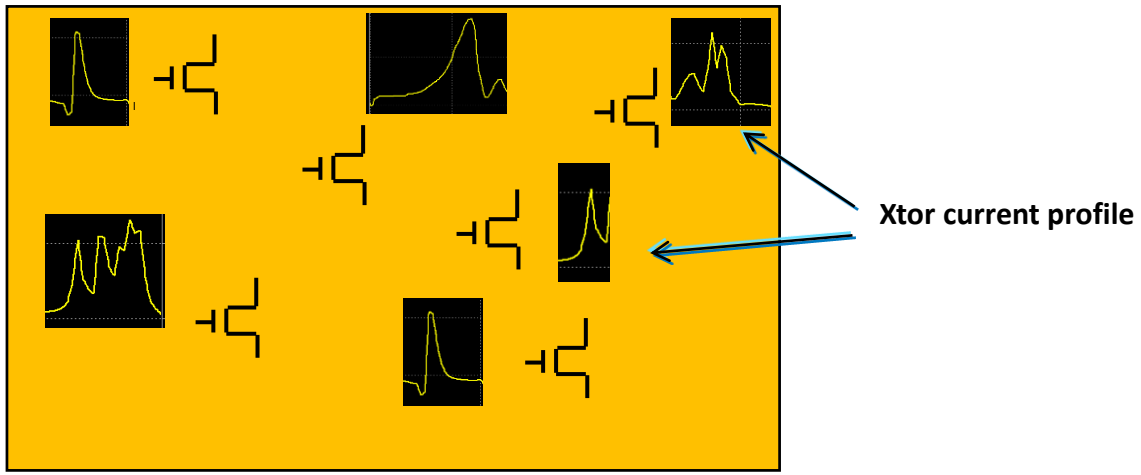
Totem CMM – cell_view

- **Totem CMM (cell_view)**
  - Spice simulation based current profile for macro
  - Single (cell level) current profile for entire macro is captured
  - Current sinks created at transistor or higher level metal with weights based on switching state
  - Capacitance of intrinsic and intentional decaps are modeled



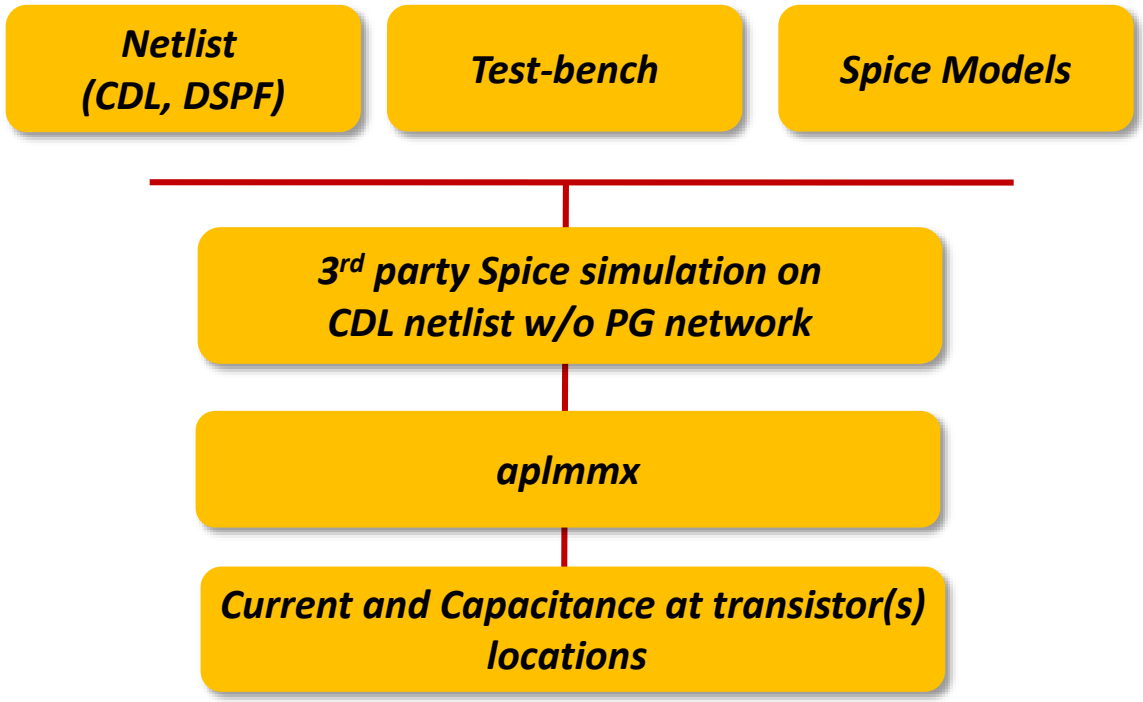
Totem CMM – mmx_view

- **Totem CMM (mmx_view)**
  - Spice simulation based current profile for macro
  - Individual transistor currents are captured during simulation
  - True transistor currents i(t) is modeled at transistor locations
  - Capacitance of intrinsic and intentional decaps are modeled

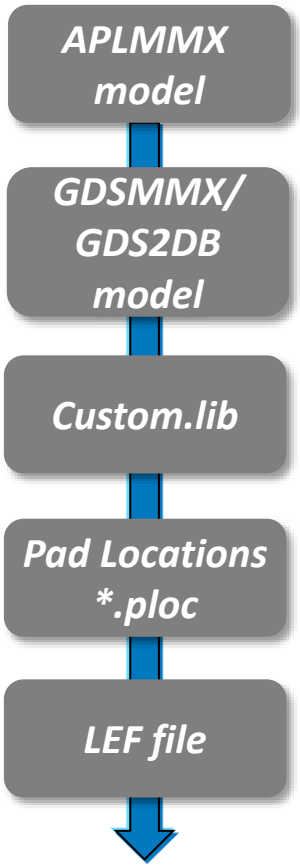


Current Modeling
MMX Flow

Electrical model creation in MMX Flow



Required Input Data



- *Electrical design information from APLMMX (previous slide)*
- **.spcurrent and *.cdev files*

-GDS to extract Power grid view using GDSMMX/GDS2DB

- *Defines the power and ground arcs needed for the IP*

- *Define the ideal voltage source locations on top metal*

- *Original LEF file for the macro that is used in PnR environment*

- **Refer Totem Training Slides for Details**

RedHawk Analysis Flow: Using Totem Models

- **The model pointers are specified inside the top level gsr file using the CMM_CELLS construct:**

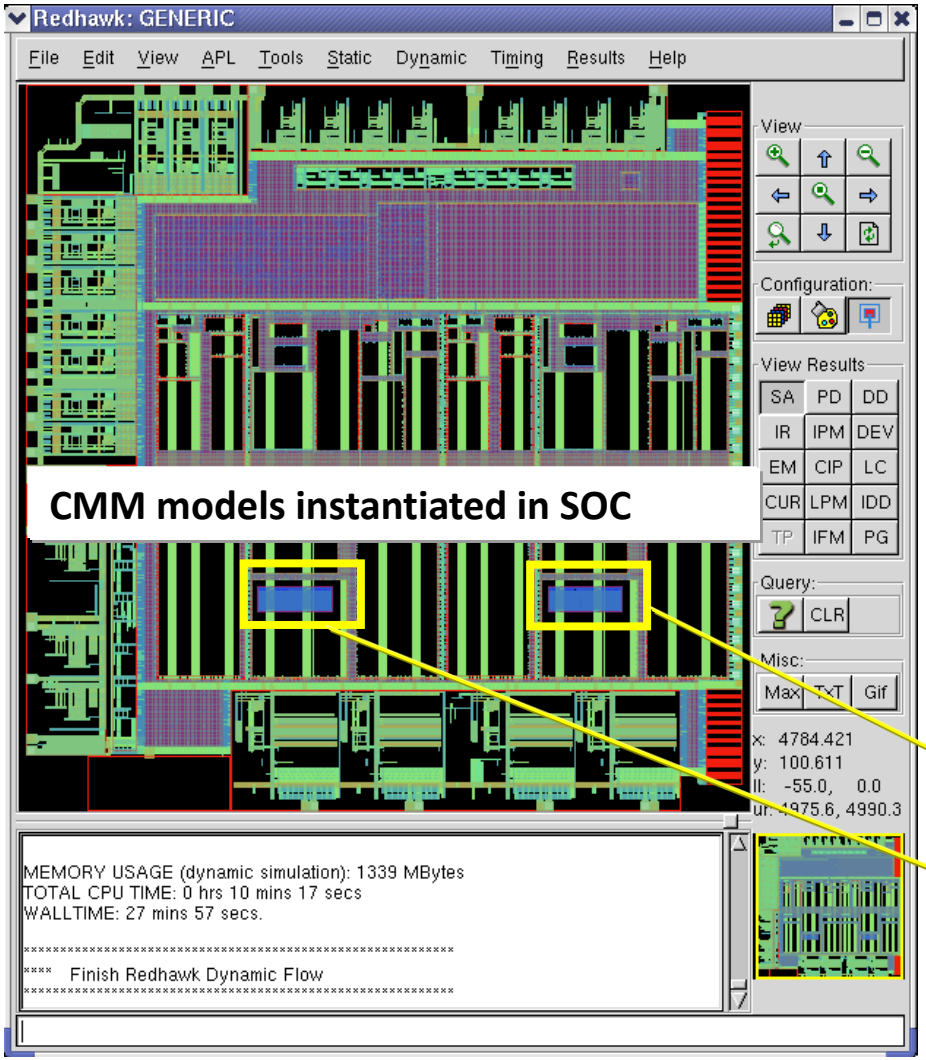
```
CMM_CELLS {
    <cell_name1>  <macro_model_path1> [original | reduced | compact]
    <cell_name2>  <macro_model_path2> [original | reduced | compact]
}
```

- **Example:**

```
CMM_CELLS {
    MEM_A    MEM_A_model/cell_view/MEM_A_cmm original
    MEM_B    MEM_B_model/mmx_view/MEM_B_cmm original
}
```

  - In the above example, MEM_A is using the cell_view and MEM_B is using the mmx_view mode.
  - Original/Reduced/Compact controls model reduction for capacity and accuracy tradeoff with reduced being default (reasonably good accuracy)
  - MEM_A and MEM_B should not be defined under the GDS_CELLS and APL_FILES section in the gsr

SoC Integration Example



Instance level voltage drop report

Errors/Warnings Summary | CPU/Memory Usage | Setup Design | Power | Results

```
Worst Dynamic Voltage Drop:
-----------------------------------------------------------
Type   value    net  ideal_volt   location      name
-----------------------------------------------------------
WIRE  180.60mV  VDD  1.200  (2046.174,1506.945) METAL1
WIRE  215.60mV  VSS  0.000  (2012.474,1511.485) METAL1
avgTW 284.90mV  VDD  1.200  (2718.110,2804.189) inst_129422/inst_7730
maxTW 272.50mV  VDD  1.200  (2718.110,2804.189) inst_129422/inst_7730
minTW 290.30mV  VDD  1.200  (2718.110,2804.189) inst_129422/inst_7730
minWC 328.90mV  VDD  1.200  (2448.545,2722.895) inst_129422/inst_7778

Worst Transistor Pin Dynamic Voltage Variation:
-------------------------------------------------------------------------
Type      value    net  ideal_volt  location          name (xtor_name)
-------------------------------------------------------------------------
XtorMinWc 1.0194  VDD  1.200       (2046.194,1505.915)  inst_129422/inst_7780/inst_mmx/
adsU1:VDD.gds15847 (X29.X4.X2.X0.X0.X14.M73)
XtorMinWc -0.0009  VSS  0.000        (3945.729,1388.275)  inst_129425/inst_7780/inst_mmx
/adsU1:VSS.gds10441 (X0.X23.X1.X0.X3.X0.X1.X1.X0.M2)
XtorMaxWc 1.1809  VDD  1.200       (3914.579,1386.870)  inst_129425/inst_7780/inst_mmx/
adsU1:VDD.gds7275 (X0.X23.X1.X0.X2.X3.X1.M7)
XtorMaxWc 0.2156  VSS  0.000       (2008.394,1511.485)  inst_129422/inst_7780/inst_mmx/
adsU1:VSS.gds22472 (X112.X2.X0.X14.M9)
```

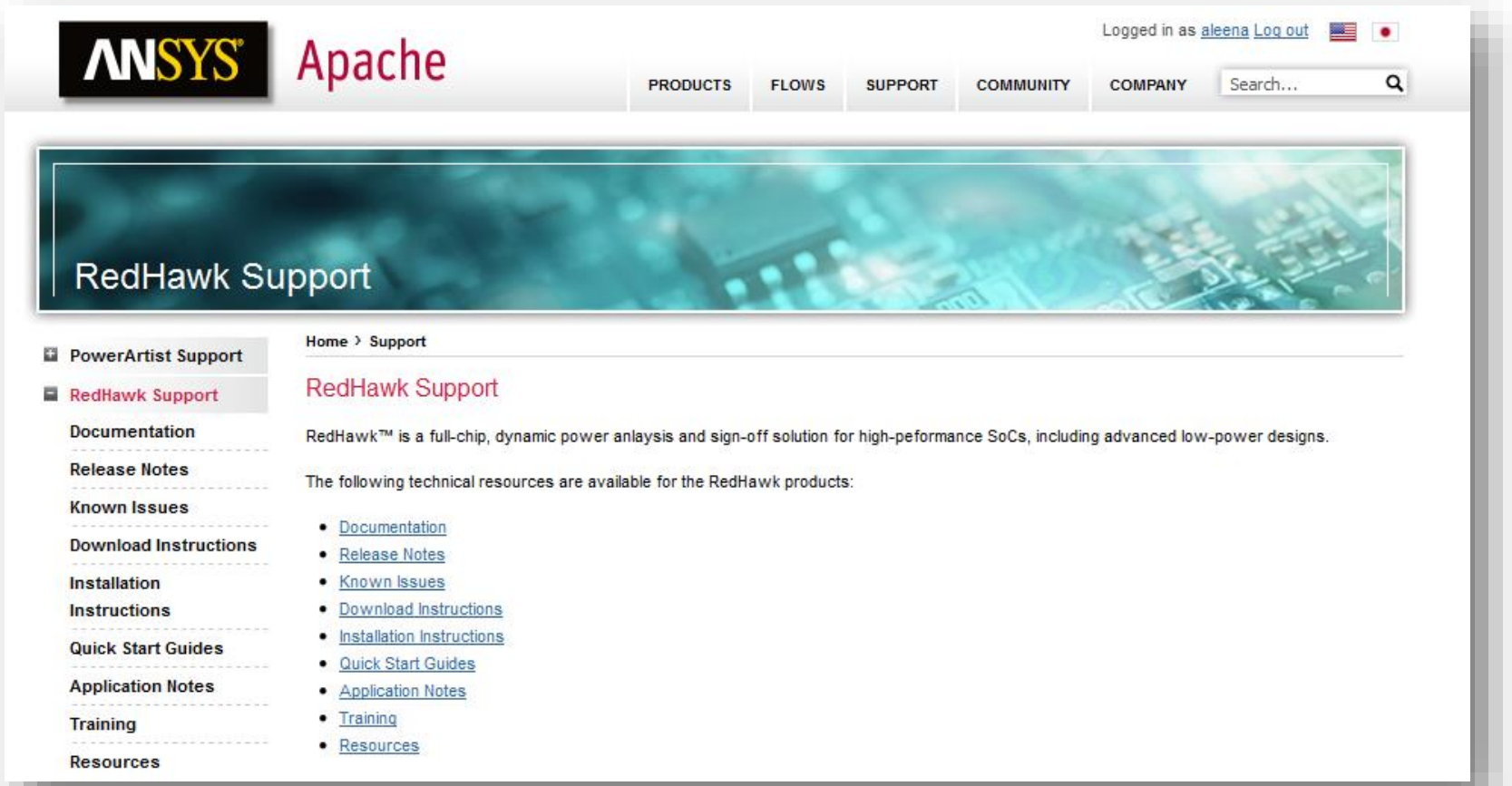
Xtor level voltage drop report inside MMX block

MEM_A → cell_view model

MEM_B → mmx_view model

How to get Help!!

- **Apache Online Customer Support Center**
  - http://support.apache-da.com
  - **Email:** support@apache-da.com



Thank You!!!