Generating accurate gate level activity for power analysis, prior to back annotated timing simulation

Carsten Rau

Infineon Technologies AG

June 23rd, 2016

SNUG Germany

Agenda

Introduction

Traditional RTL Activity Propagation

Verdi Activity Generation/Propagation Flow

Results

Conclusion

Introduction

- Did you ever …
  - Try to compare two EDA power tools, using incomplete activity as input?
  - Find unexpected power consumption in high active combinatoric logic just at layout stage? Hence wished you had reasonable gate level activity a bit earlier?
- If you checked at least one of the above, you are probably looking for
  - A generic activity propagation engine that produces valid activity without need to be timing clean
- If you checked all of the above, you might be the one giving this presentation ☺
  - … and talk about some experiments using the Verdi/Siloti feature on a 65nm design layout netlist to see if that helps

Agenda

Introduction

Traditional RTL Activity Propagation

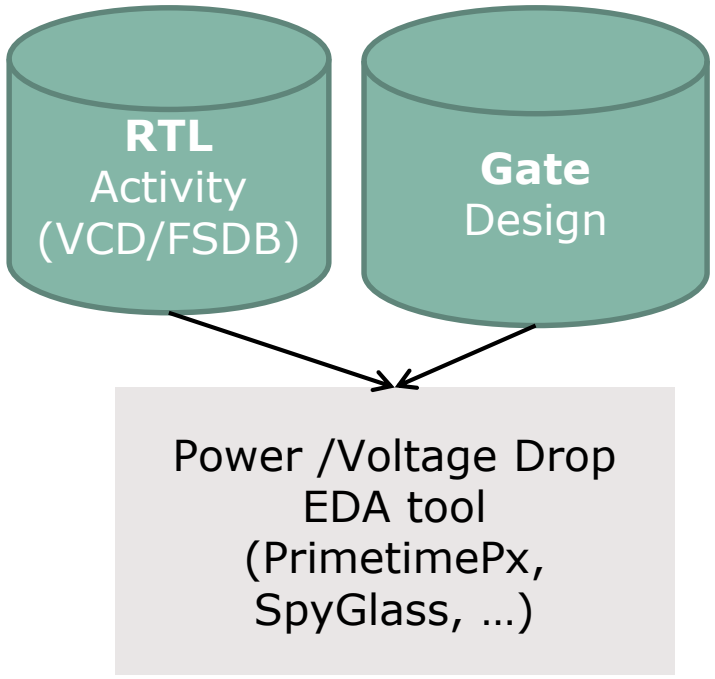
Verdi Activity Generation/Propagation Flow

Results

Conclusion

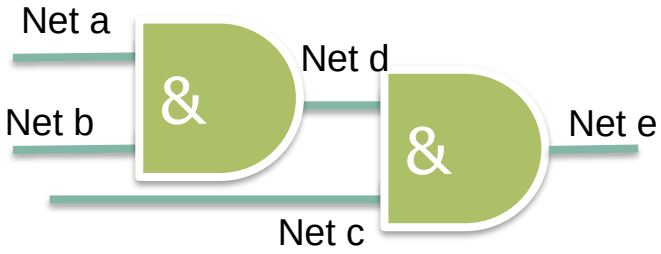
Traditional RTL Activity Propagation

- All power or voltage drop analysis tools do have algorithms to propagate missing activity through the design



Simple propagation example (actual EDA tools may differ)
Average Toggle Rates in activity file:

TR(a)= 0.5 (RTL simulated)
TR(b)= 0.5 (RTL simulated)
TR(c)= 0.25 (RTL simulated)
TR(d)= missing in simulation
TR(e)= missing in simulation



EDA tools propagate with own algorithms missing activity data:
TR(d)= 0.5(a) * 0.5(b) = 0.25 (propagated)
TR(e)= 0.25(c) * 0.25(d, propagated) = 0.0625 (propagated)

Traditional RTL Activity Propagation

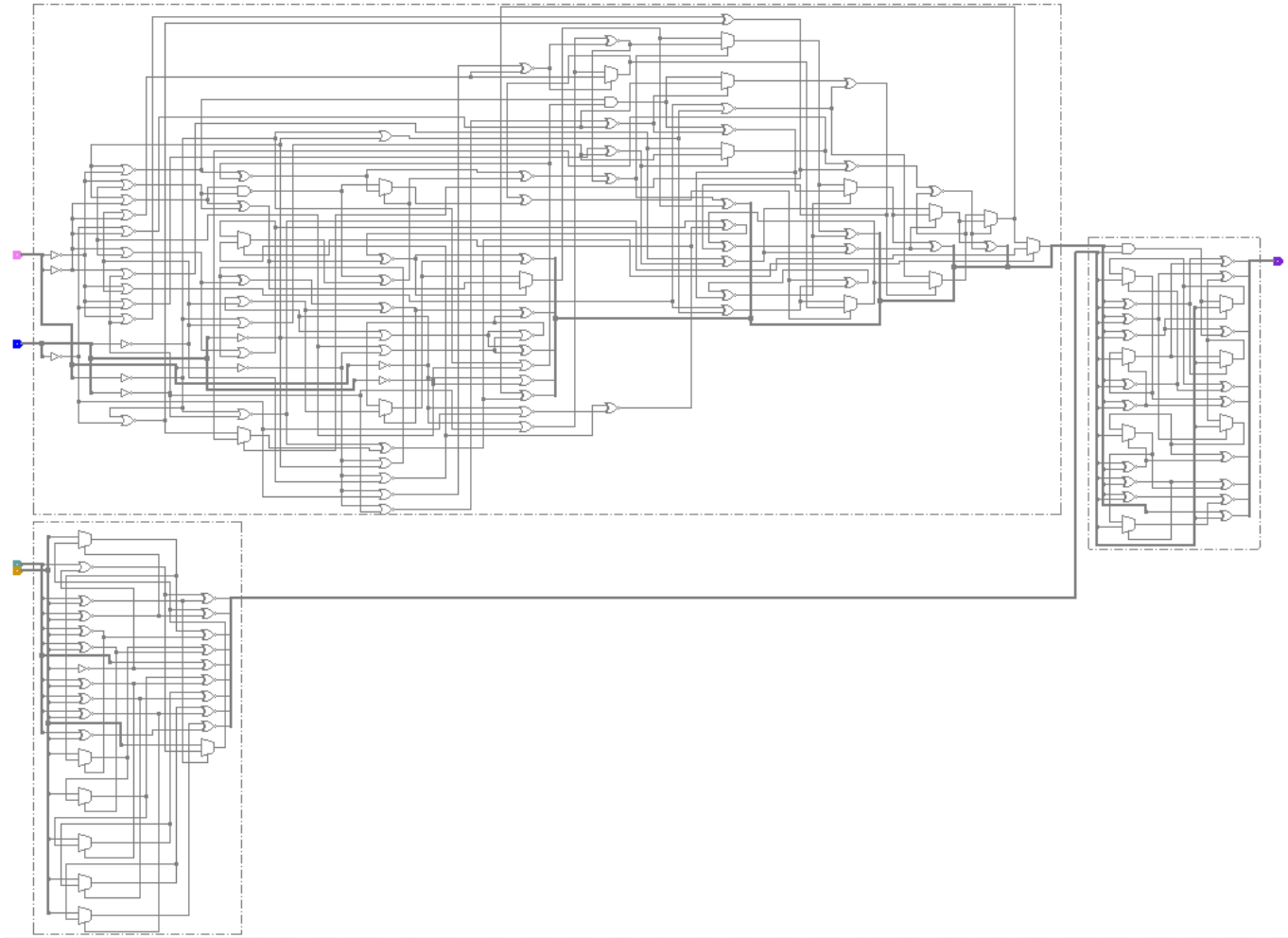
Main Issues with this approach

- RTL activity annotation may have issues with name mapping
- Each EDA tool internal propagation algorithm differs
- **All** EDA Vendor activity propagation engines have issues, especially with high active combinatoric (generated) clouds.
  - VHDL example concurrent assignment:
    A_signal64bit <= (B_signal32bit * C_signal32bit) + (D_signal64bit – E_signal64bit)
    - In RTL simulation A_signal64bit will change **once**, each time some input changes
    - In real gate simulation, A_signal64bit may change **multiple times**, as the inputs toggle through the generated combinatoric cloud needed for that single line of RTL
    - EDA tools will even honor original A_signal64bit activity from simulation activity
    - EDA tools propagation engine will not be accurate for the nets of the generated combinatoric cloud

Traditional RTL Activity Propagation

Detailed Example

- VHDL: A_signal8bit <= (B_signal4bit * C_signal4bit) + (D_signal8bit - E_signal8bit);
- Abstract Schematic:
- Detailed Schematic:



Agenda

Introduction

Traditional RTL Activity Propagation

Verdi Activity Generation/Propagation Flow

Results

Conclusion

Verdi Activity Generation/Propagation Flow

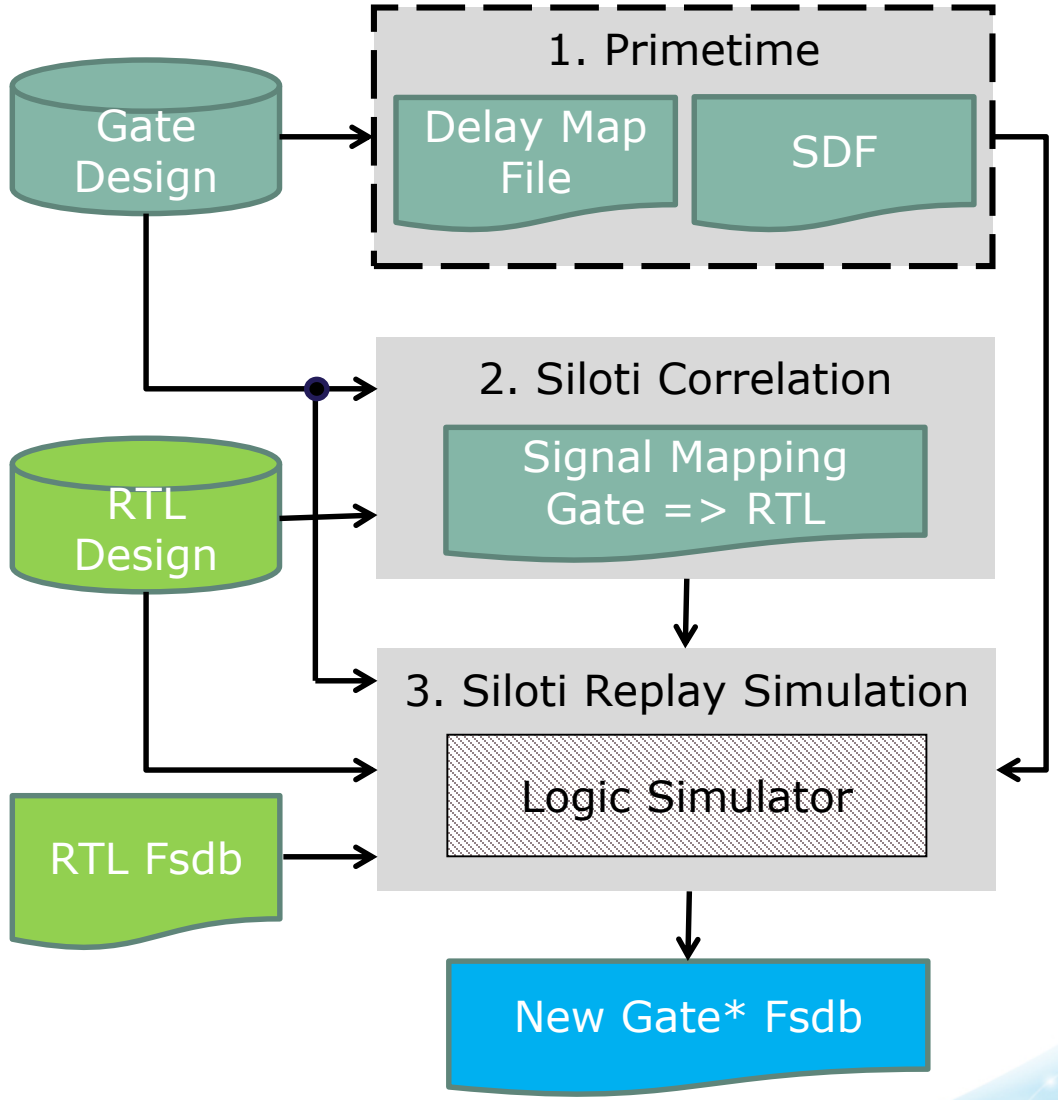
Overview

- The Trick to early valid gate activity:
  - Step 1: Create RTL-GATE mapping points only for special nets:
    - Primary inputs, registers, outputs of blackboxes/full custom macros
  - Step 2: Run standard gate level simulation, but instead of a standard gate testbench, use the RTL activity file to drive the simulation
    - Each time a mapped signal changes in RTL activity dump, force it in the gate level simulation
    - All other signals simply let the simulation run
  - Benefit:
    - Real gate simulation with real SDF, resulting in gate level like activity
    - Works even with netlist design state that has incomplete functionality or timing violations
    - Single propagation algorithm as activity file can be reused for all further tools

Verdi Activity Generation/Propagation Flow

Flow Description

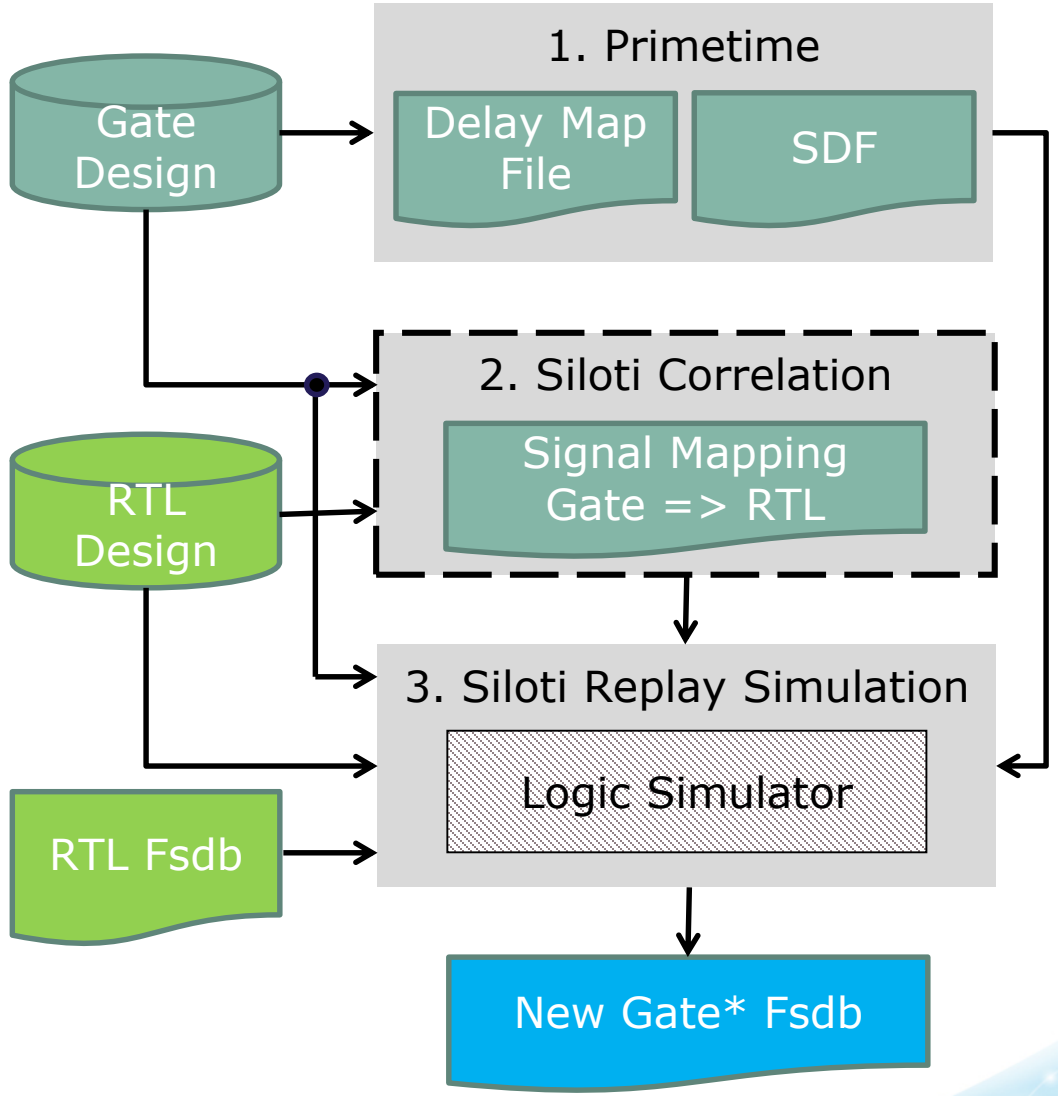
1. “Generate STA data” (with Primetime)
   - Generate SDF
   - Run a primetime tcl script to generate clock root to Q pin delay for all registers
   - Goal: To allow realistic delay of register activity as clock tree does not exist on RTL
   - Output: delay map file: “<register> #<clockroot_to_Q delay>“



Verdi Activity Generation/Propagation Flow

Flow Description (continued)

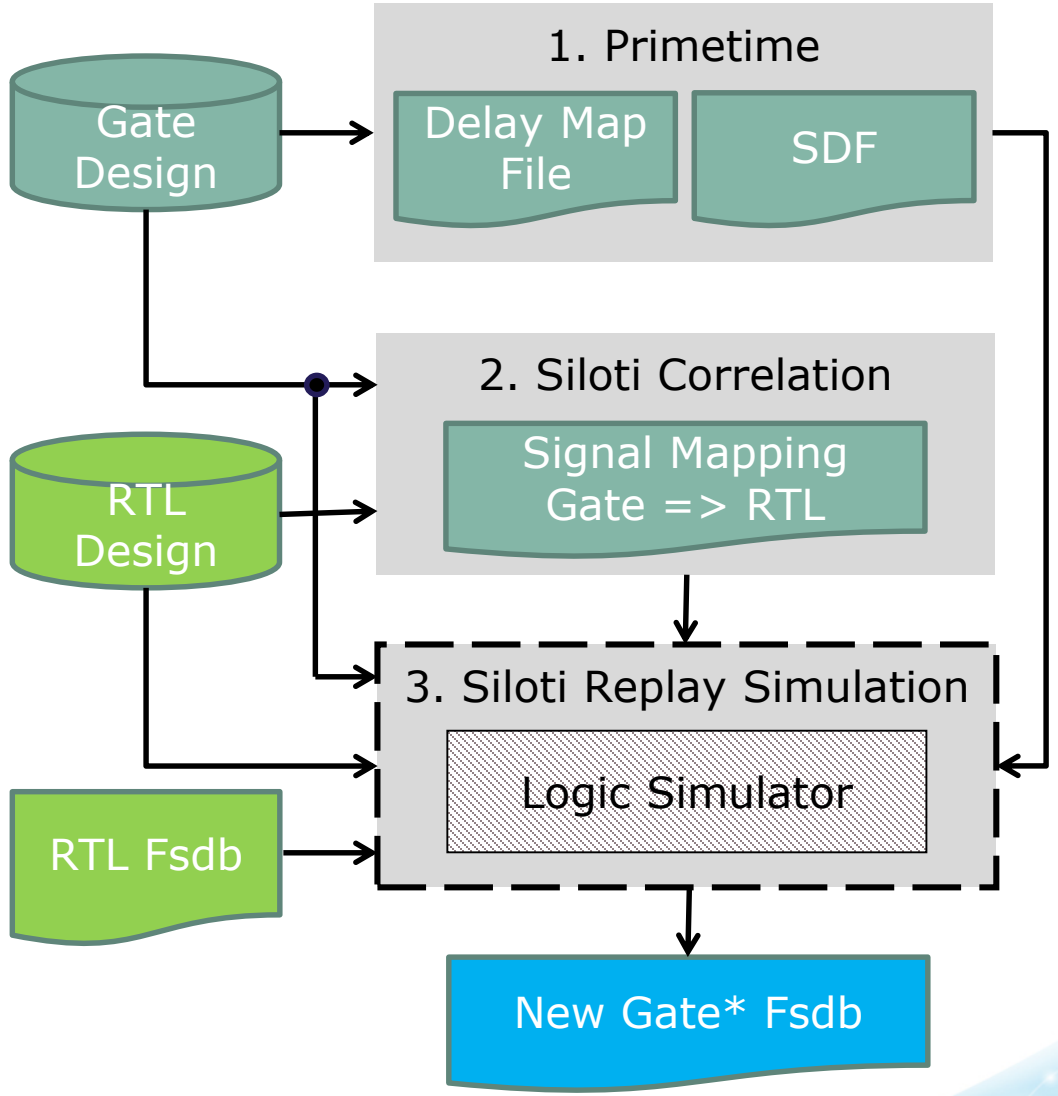
2. “Correlate GATE to RTL”
   - Read RTL + Gate Design into Verdi and run name mapping command crdb
   - Goal: Create for each register, primary input, black box output in the gate level design a signal mapping to its RTL equivalent
   - Output: signal mapping file:
     “<GATE signal> => <RTL signal>“



Verdi Activity Generation/Propagation Flow

Flow Description (continued)

3. “Run Replay simulation”
   - Verdi “cockpits” digital simulator via PLI interface
   - Goal: Simulate gate level design by forcing all signals in mapping file, whenever RTL equivalent signal toggles in RTL fsdb file
     => no need for timing clean design!
   - Time of force = RTL toggle time + delay from delay map
   - Output : new pseudo gate fsdb file



Verdi Activity Generation/Propagation Flow

Used flow for this exercise

- RTL FSDB file that contains same use case as real layout simulation
- Same layout netlist as used for real layout simulation
- Third party digital simulator as “propagation engine”
- SVF file for helping in name mapping was not available.
  To achieve 100% mapping of relevant signals,
  we used instead custom design specific Perl script
  to resolve name mapping issues

Agenda

Introduction

Traditional RTL Activity Propagation

Verdi Activity Generation/Propagation Flow

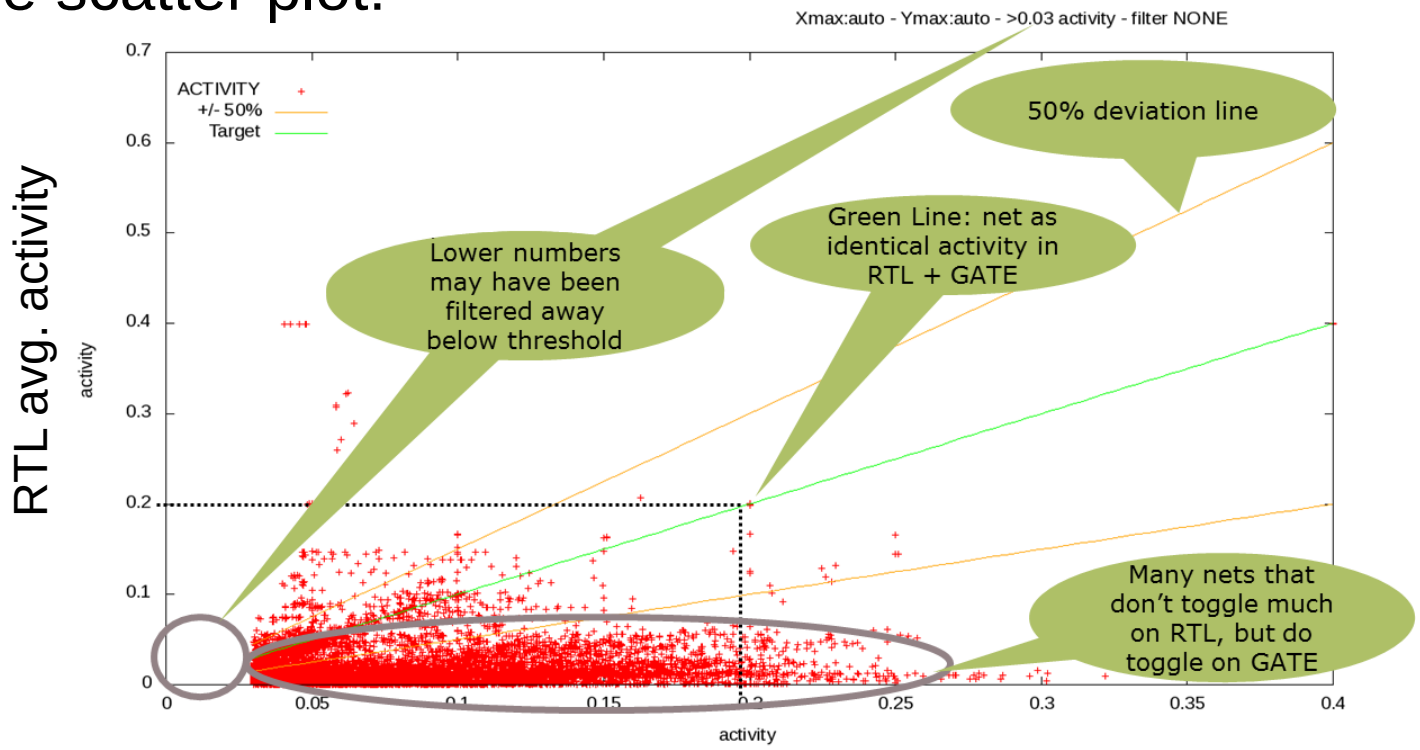
Results

Conclusion

Results

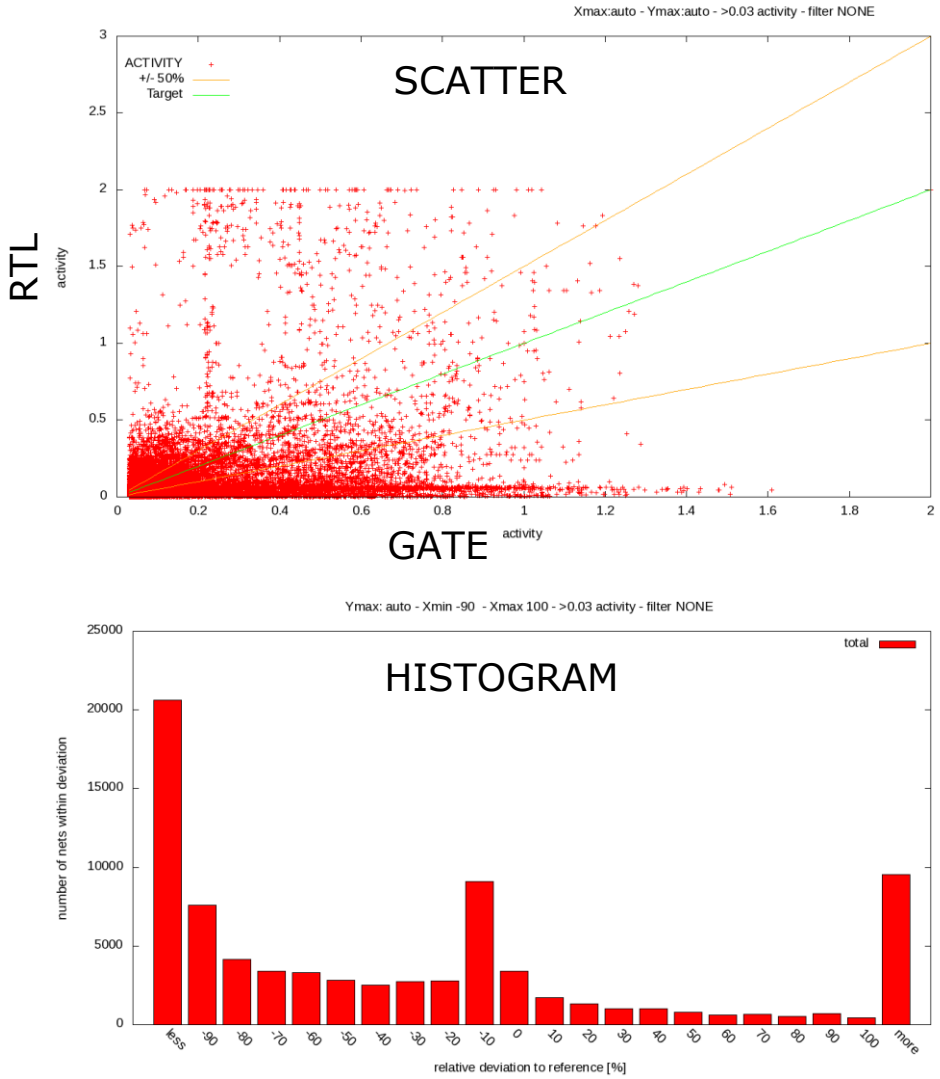
Overview

- Comparison of various tools running on same design layout netlist setup => only difference is input activity file (RTL, Gate, Verdi)
- Example scatter plot:

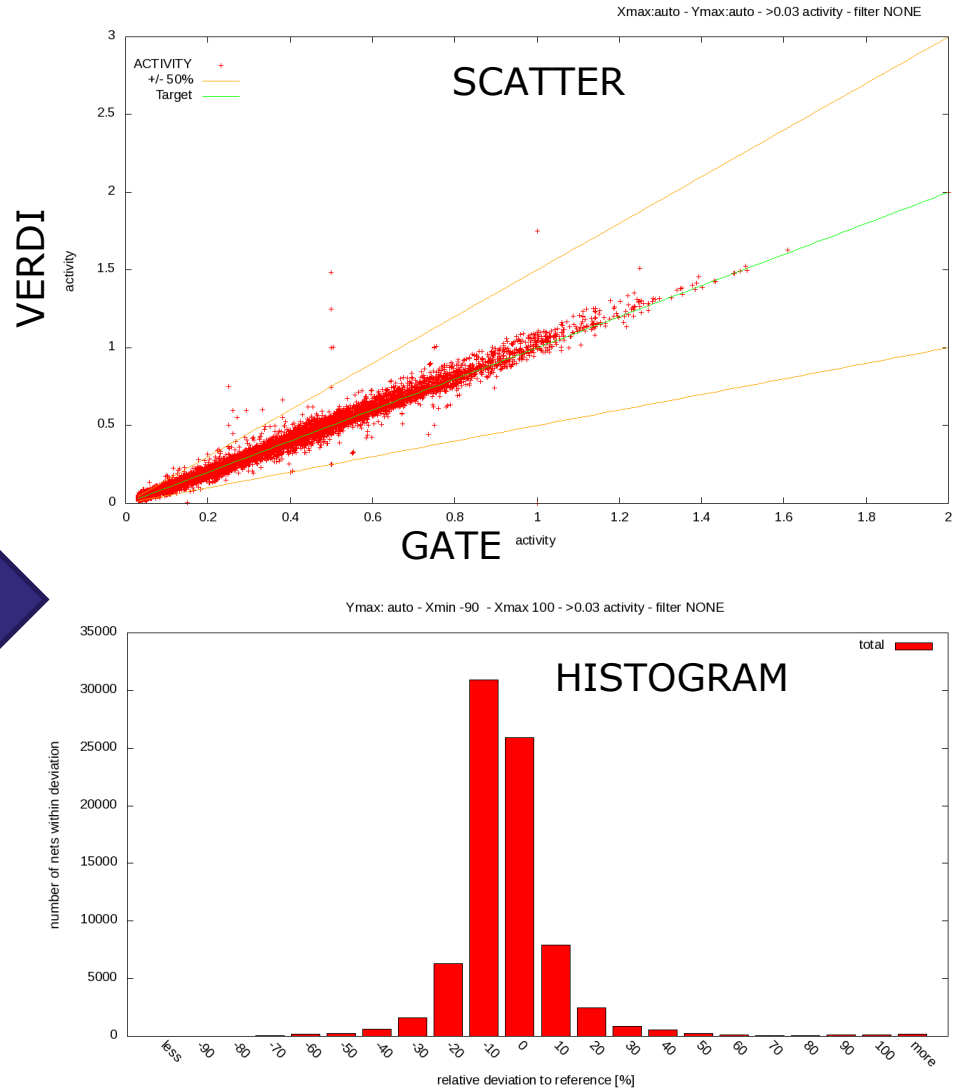


Results

SpyGlass Power – Net Activity

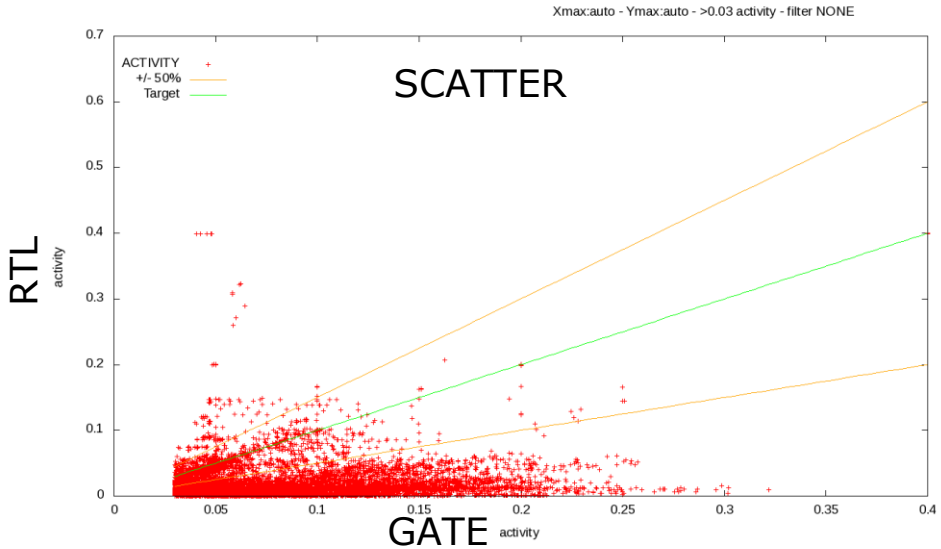


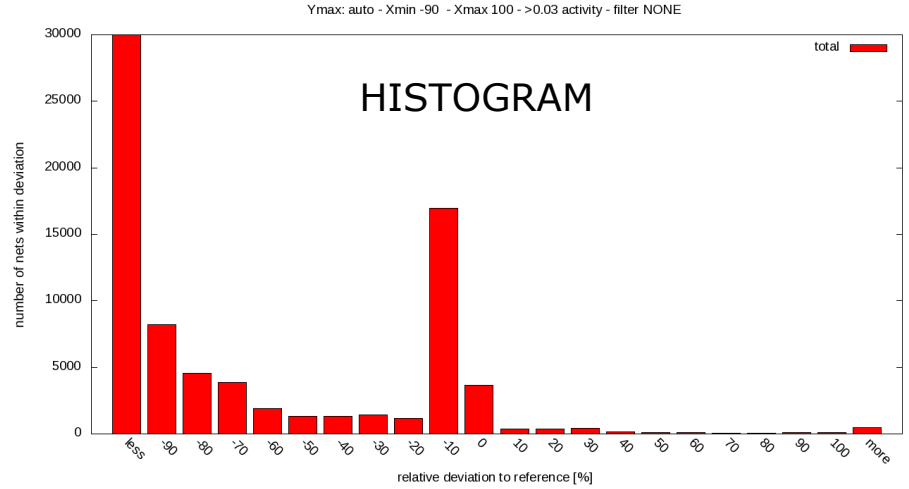
Improvement



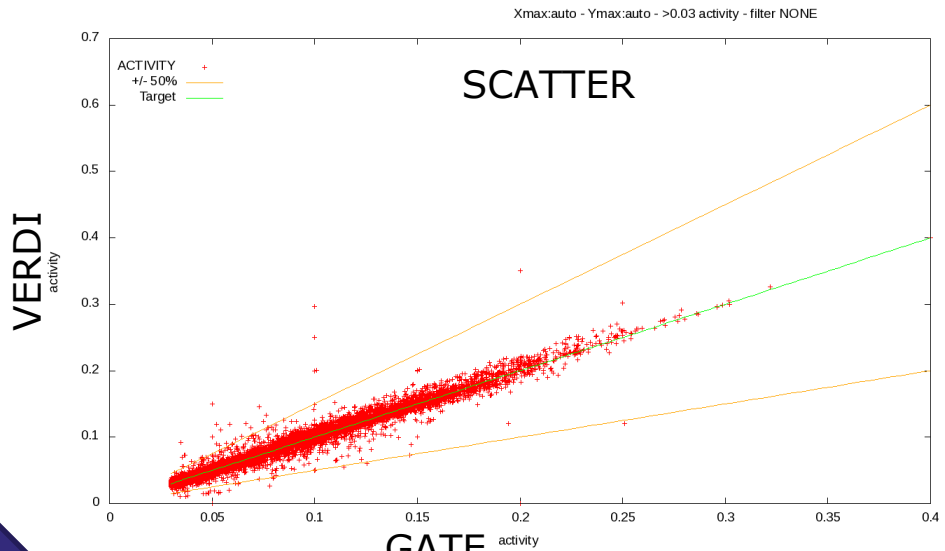
Results

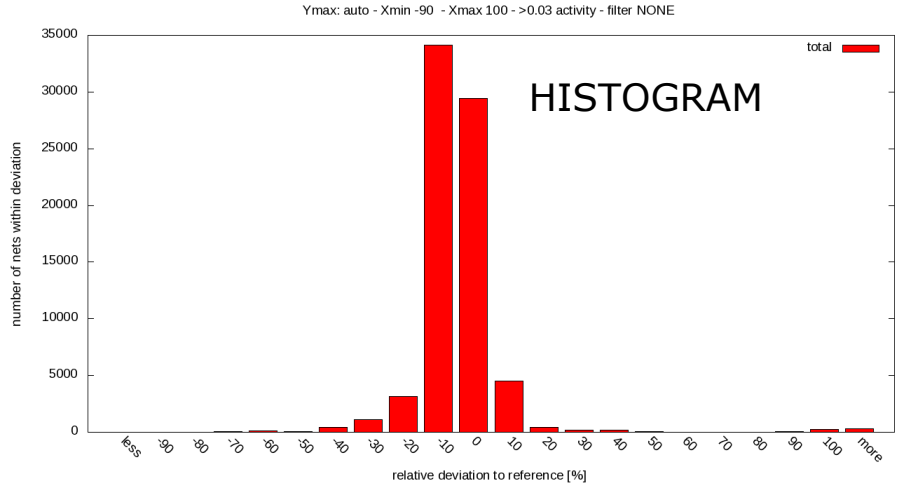
PrimetimePX – Net Activity





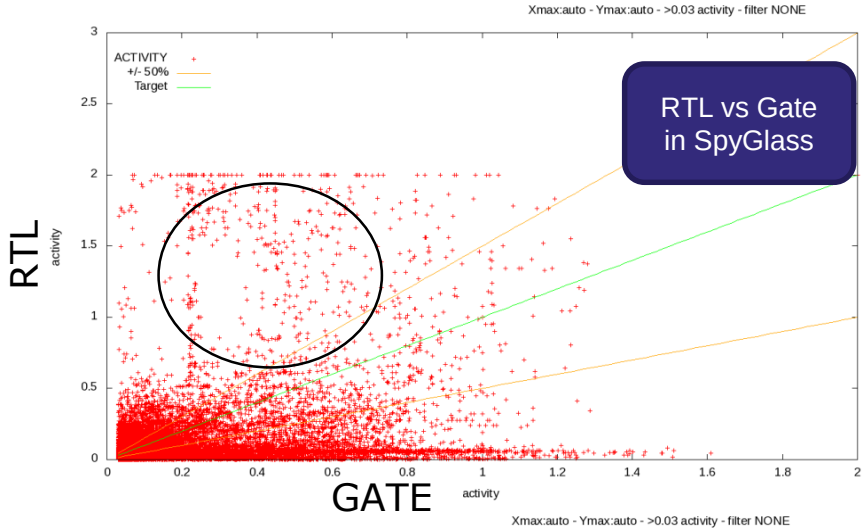
Improvement

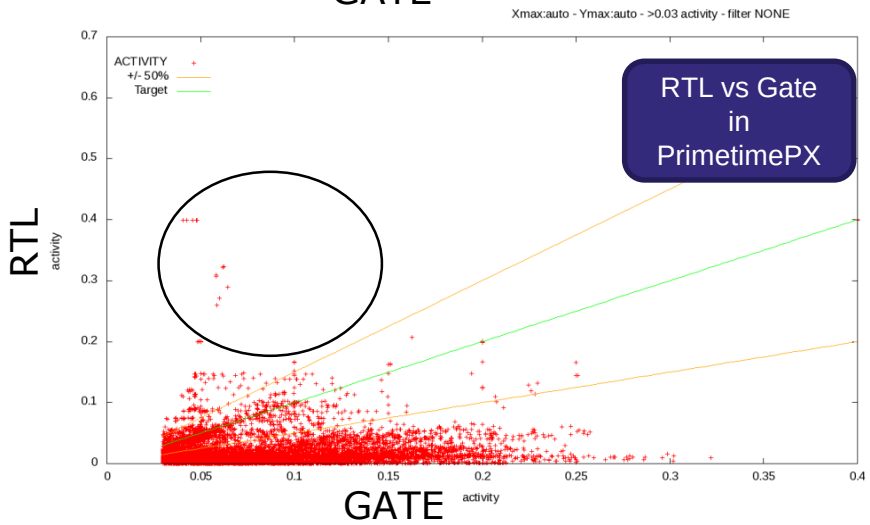




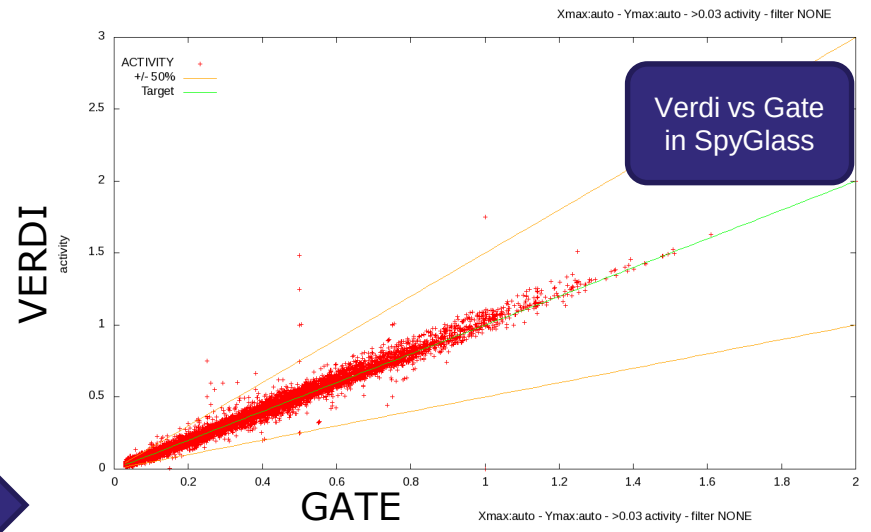
Results

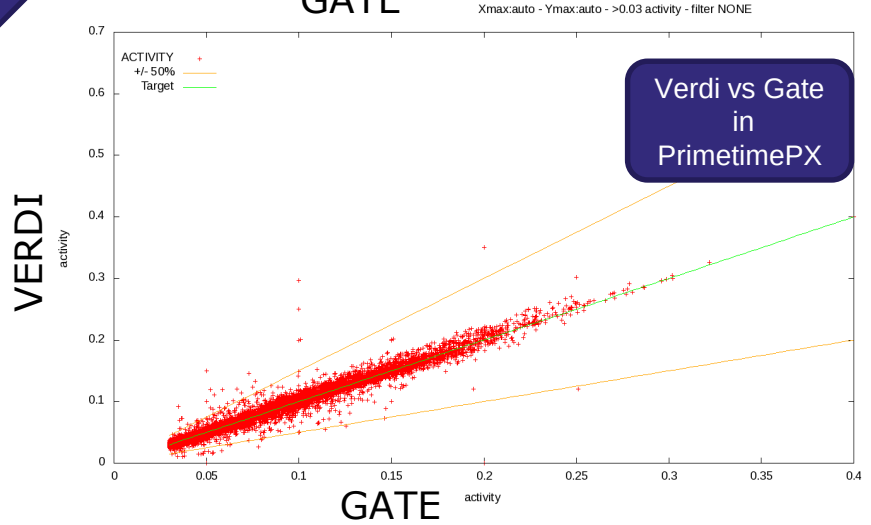
SpyGlass Power vs. PrimetimePX – Net Activity





Consistent behavior





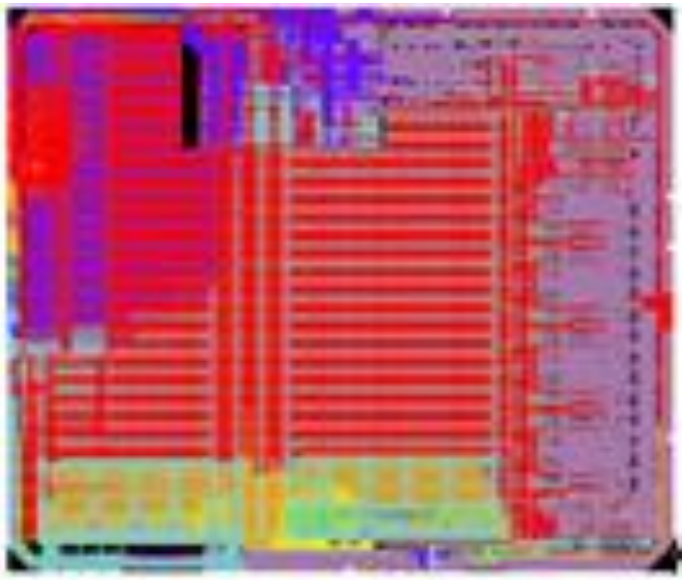
=> Removing any EDA tool specific activity propagation engine from the equation!

Results

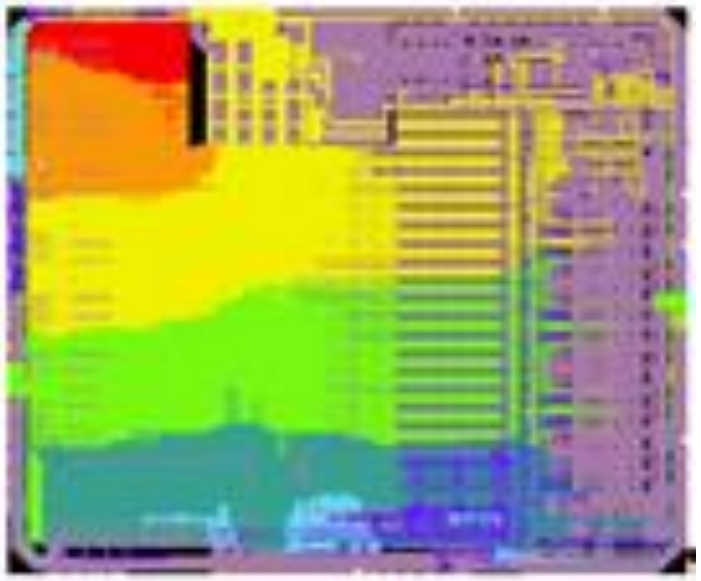
Voltage Drop Analysis Tool -Voltage drop map

- Average activity based static voltage drop analysis
  (color scaled automatically based on respective min/max voltage drop,
  so color not identical in all pictures)

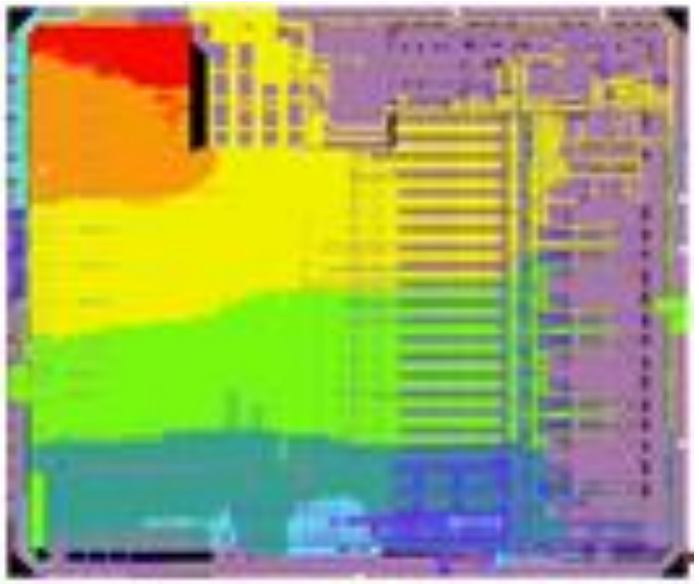
RTL Activity



Gate Activity



Verdi Activity



Agenda

Introduction

Traditional RTL Activity Propagation

Verdi Activity Generation/Propagation Flow

Results

Conclusion

Conclusion

- Described flow allows generation of valid gate level activity as shown in results
- Described flow ensures all EDA tools get identical activity as input data, as it substitutes their internal propagation engine.

- Acknowledgements
  - Infineon Technologies
    - Soenke Grimpen
    - Tim Bräuninger
  - Synopsys
    - Jens Dickel

Thank You