Early power analysis flow using RTLA and RTL-PrimePower, with a focus on glitch power

Nicola Imperato – Hardware designer
Felice Tecce– Hardware designer

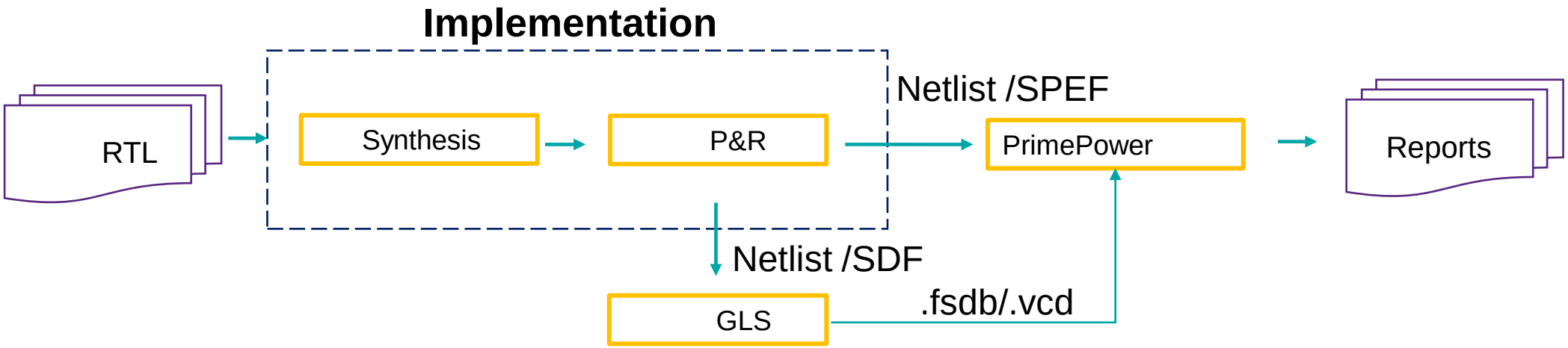
STMicroelectronics

AGENDA

1. Power estimation flow: classic flow
2. Power estimation flow with RTLA
3. Results
4. Glitch power analysis & optimization
5. Conclusion

1. Power estimation flow: classic flow
2. Power estimation flow with RTLA
3. Results
4. Glitch power analysis & optimization
5. Conclusion

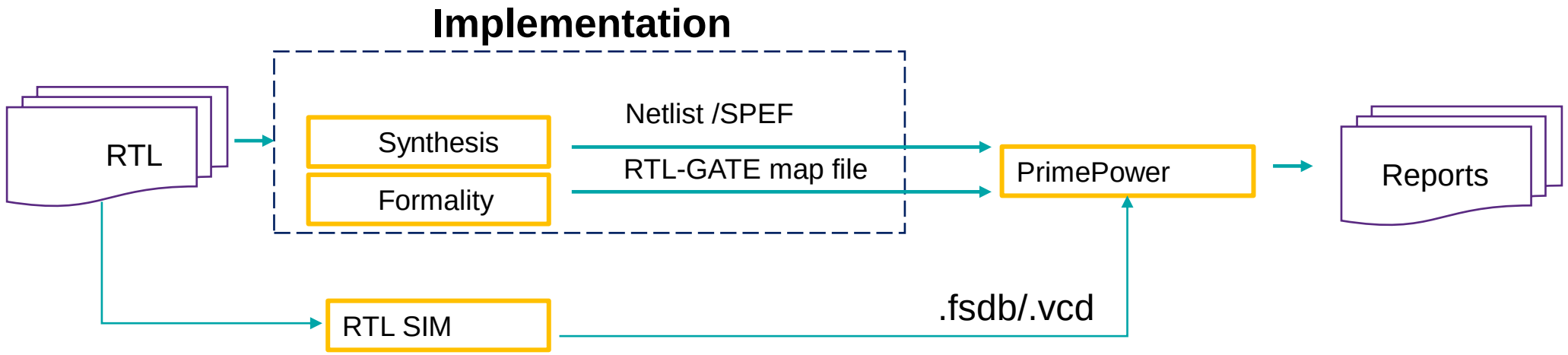
Power estimation flow: postlayout flow



Signoff Power analysys implies :

- GLS activity files (FSDB/VCD) → complete switching activities
- SPEF → real capacitance estimation
- Postlayout netlist → final netlist topology
- Final clock tree structure → accurate clock network power

Power estimation flow: postsyn flow

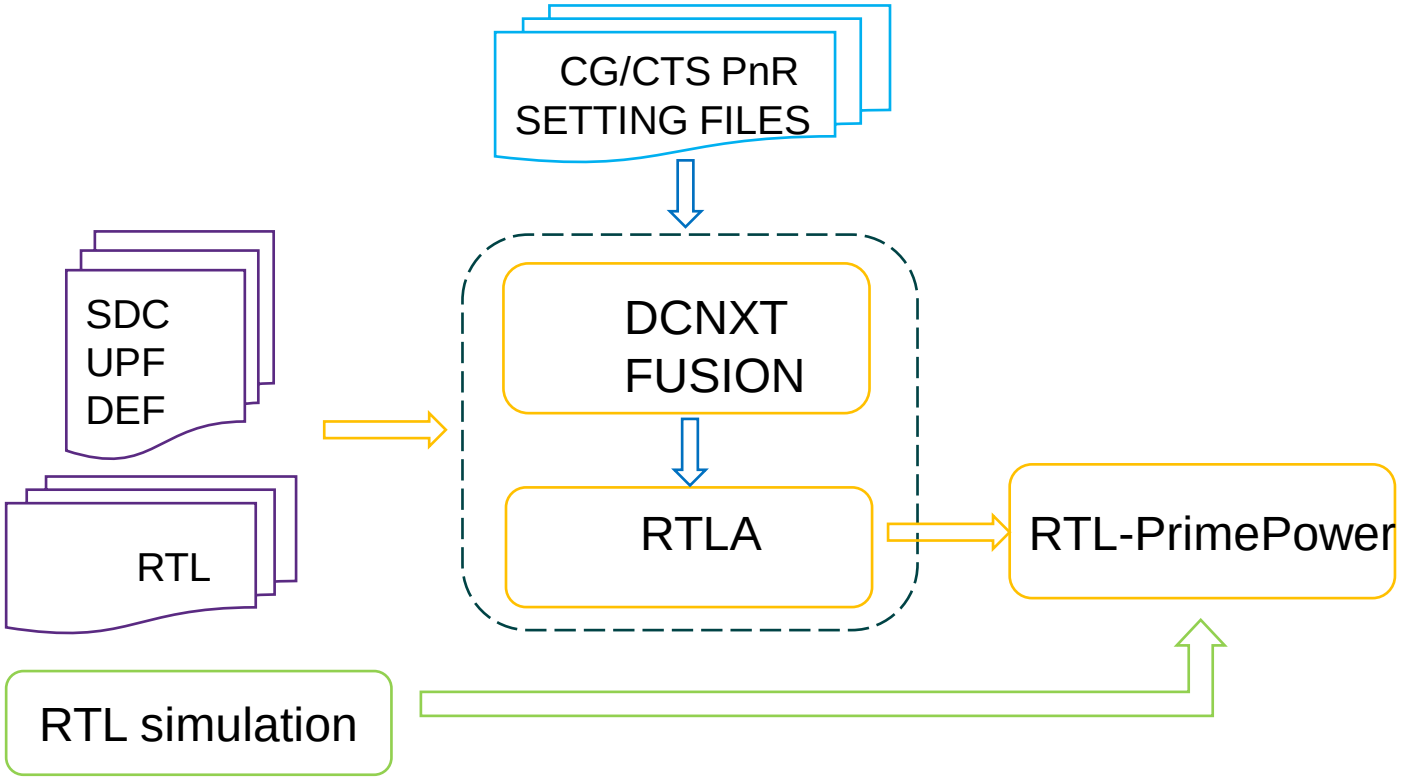


Early PA is inaccurate because :

- Only RTL s.a. is available → postsyn netlist
  - Use map file to import RTL activity (only invariant point are mapped)
- CTS power very dependent by clock gating strategy and placed cells →
  - Clock gating is aligned to postlayout (**CTS is not present but can be estimated**)
- Glitch Power very dependent by synthesized netlist and interconnect delay →
  - Glitch Power can be estimated enabling Delay Shifted Analysis

1. Power estimation flow: classic flow
2. **Power estimation flow with RTLA**
3. Results
4. Glitch power analysis & optimization
5. Conclusion

Power estimation flow with RTLA



Accurate early PA implies:

- Use same postsyn netlist to match topology and clock gating strategy → RTLA shares synthesis setting
- For CTS power use same PnR setting (cell, max_cap …) → RTLA share FUSION physical setting
- Accurate RTL s.a. mapping on postsyn netlist

Power estimation flow with RTLA

compile_setup.tcl

```
set BOUNDARY_OPT                    "FALSE"
set AUTOUNGROUP                     "FALSE"
set ENABLE_OUTPUT_INVERSION         "FALSE"
set DISABLE_QN_USAGE                "FALSE"
……"
```

clock_setup.tcl

```
set CLOCK_GATING_MAX_FANOUT    "32"
set CLOCK_GATING_NUM_STAGES    "2"
set CLOCK_GATING_MIN_BITWIDTH "4"
set ICC2_CTS_MAX_FANOUT        "8"
set ICC2_CTS_MAX_CAPACITANCE   "0.15"
set ICC2_CTS_LAYER_LIST        "metal3 metal4"
```

scenarios.csv

```
mode,libset,pex,corner_name,scenario_name,
single_sce,WorstPower,best,WorstPower_best,single_sce_WorstPower_best
single_sce,WorstPower,Typ,WorstPower_Typ,single_sce_WorstPower_Typ
```

RTLA

```
analyze
rtl_opt
compute_metrics -power
export_power_data
```

RTL-PP

Pprtl_setup.tcl

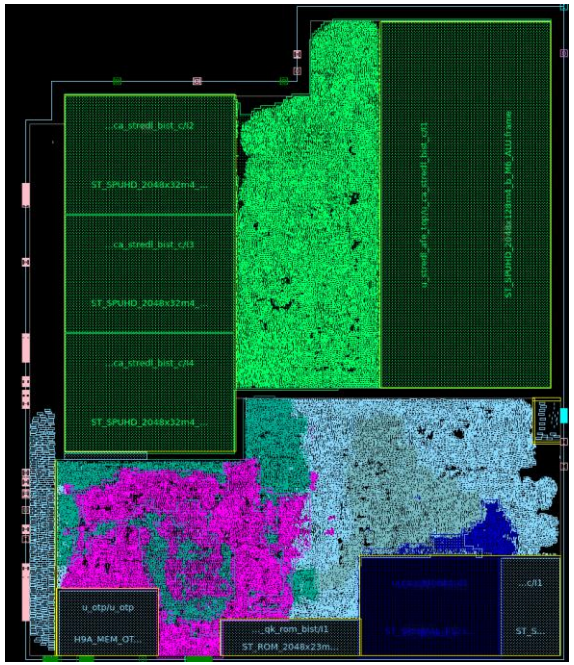
```
set_app_var power_enable_rtl_analysis                true
set power_analysis_mode                              time_based
set_app_var power_enable_delay_shifted_event_analysis true
set_app_var power_enable_clock_cycle_based_glitch    true
```

1. Power estimation flow: classic flow
2. Power estimation flow with RTLA
3. **Results**
4. Glitch power analysis & optimization
5. Conclusion

Results

ID CARD

Technology node: 130nm
Area: 180 Kgate
Max frequency: 32MHz
MACROs: 6xRAM , 2xROM
n. register ~10K
3 power domain : 1 switchable



- PL (Postlayout Power) : DCNXT (SNPS) + PnR (no SNPS) + FSDB (GLS)
- RTLA : same DEF/SDC/UPF + FSDB (RTL)

Power Group	postlayout Total [W]	RTLA Total [W]	Delta %
clock_network	4.59E-04	4.24E-04	-8
register	1.58E-04	1.58E-04	0
combinational	4.74E-04	4.54E-04	-4
memory	6.19E-04	6.18E-04	0
Total	1.71E-03	1.65E-03	-3.5

Major slack on **clock tree** while **register** and **memory** match

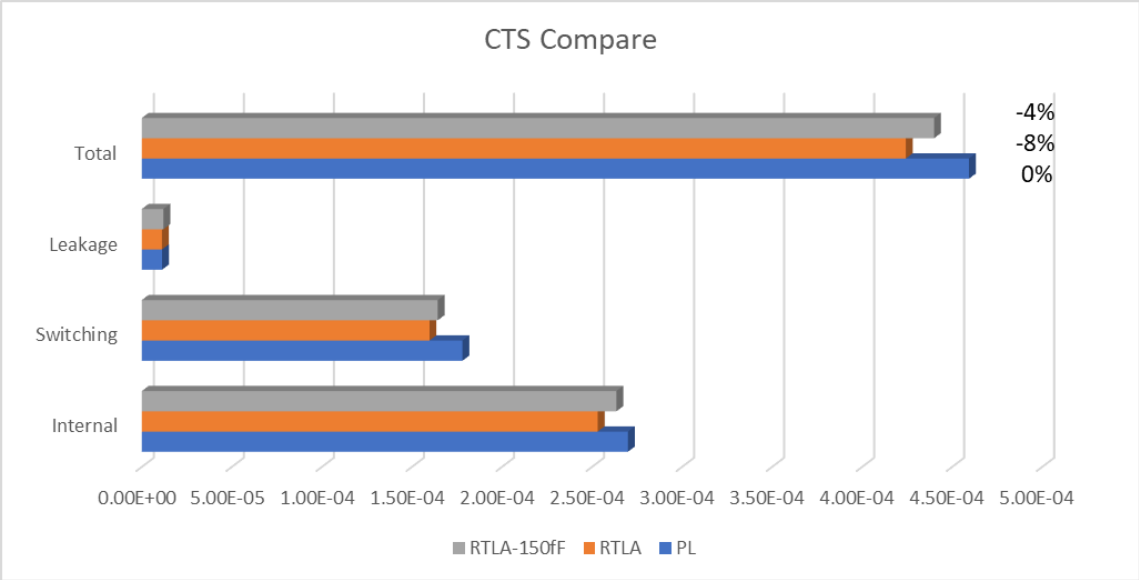
Results: Clock tree

CTS mismatch → align CTS_MAX_CAP to PL value

- ✓ Clock gating : 946 (RTLA) vs 737 (PL)
- ✓ Ct cells : 101 (RTLA) vs 486 (PL)

```
set ICC2_CTS_MAX_TRANSITION                    ""
set ICC2_CTS_TARGET_SKEW                       ""
set ICC2_CTS_MAX_FANOUT                        ""
set ICC2_CTS_MAX_CAPACITANCE                   "0.15"
set ICC2_CTS_LAYER_LIST                        "metal3 metal4"
```

- ✓ Clock gating : 946 (RTLA) vs 737 (PL)
- ✓ Ct cells : 421 (RTLA) vs 486 (PL)



Results: Register

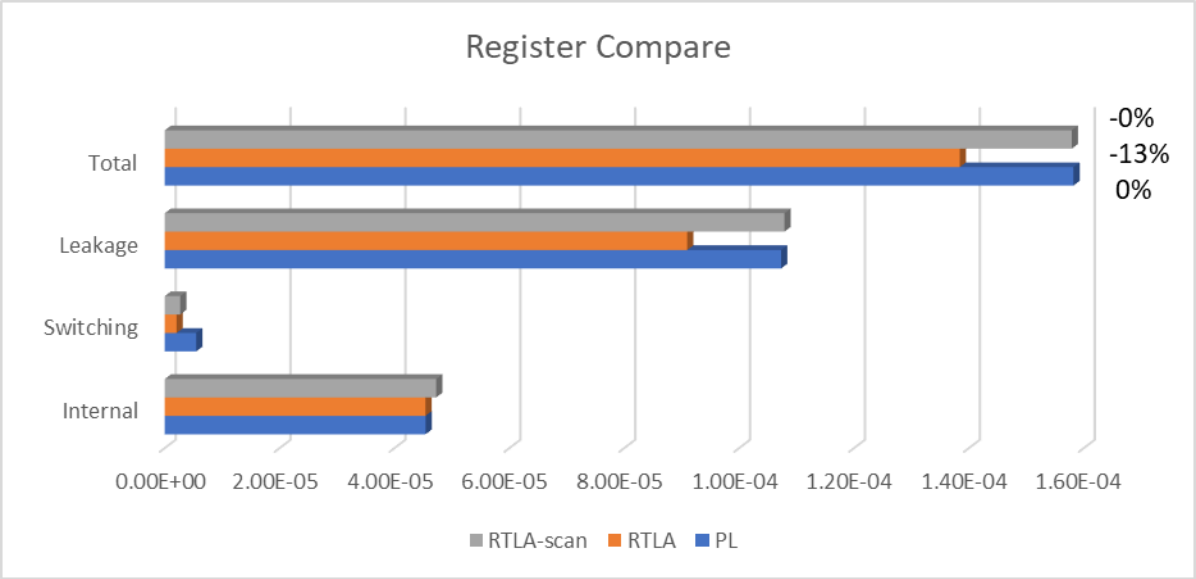
- Main difference RTLA vs DCNXT is register optimization :

  RTLA 62 FFs less than DCNXT due to different compile options (FUSION vs DC)

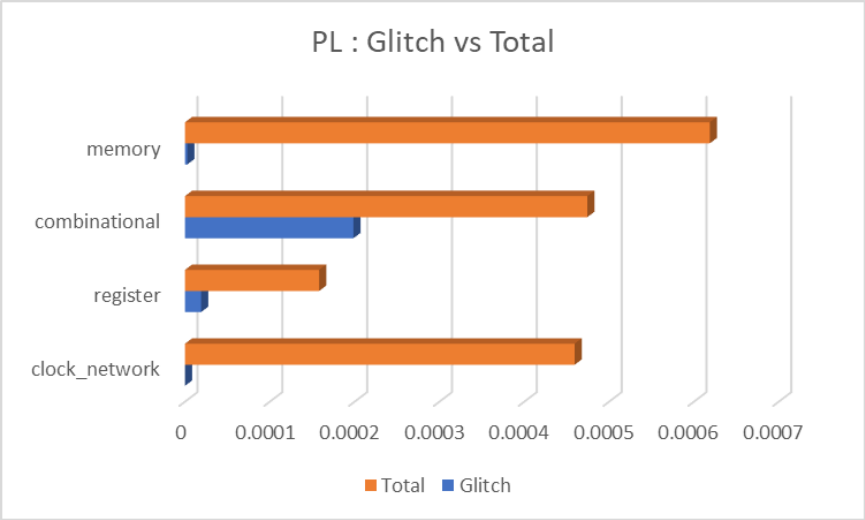
- Use scan ready register make the differences in case RTLA is not DFT ready

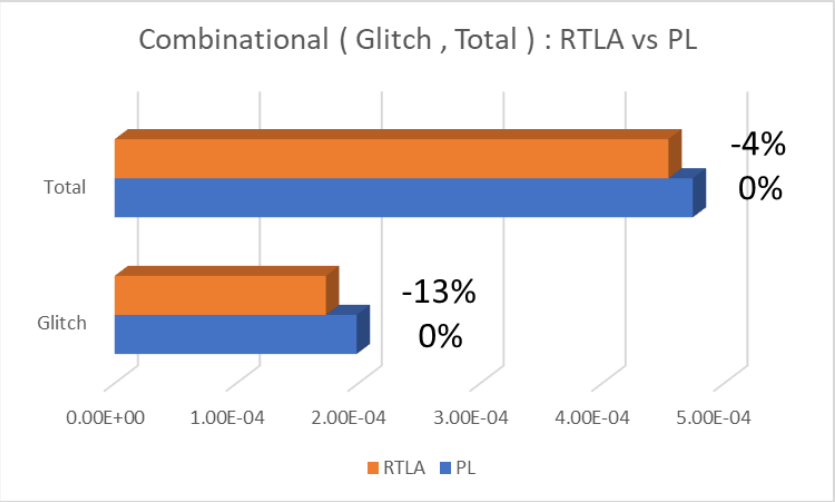
```
set_app_options -name compile.seqmap.scan -value true
```

- Switching activity annotation from RTL FSDB about 90% on invariant points



Results: Glitches





- Very important to have accurate estimation of glitch power in system with arithmetic
- Glitches power almost in combinational because of long datapath

- The major discrepancies of power is on combinational because of different tool used for synthesis
- Inside the combinational we see a big delta on the glitches power

1. Power estimation flow: classic flow
2. Power estimation flow with RTLA
3. Results
4. **Glitch power analysis & optimization**
5. Conclusion

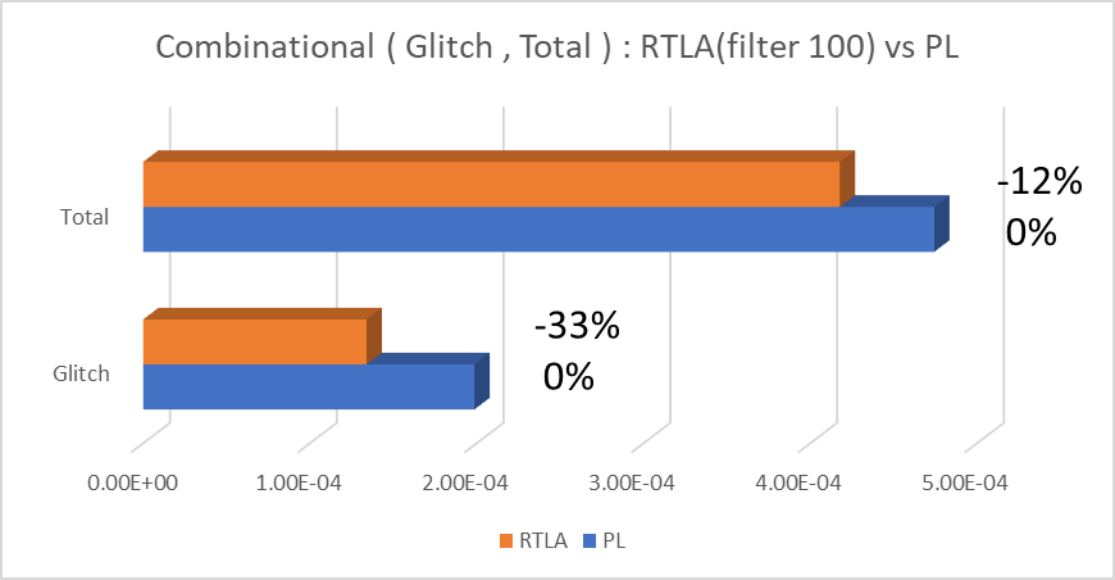
Glitch power analysis

Glitches: pulse rejection setting

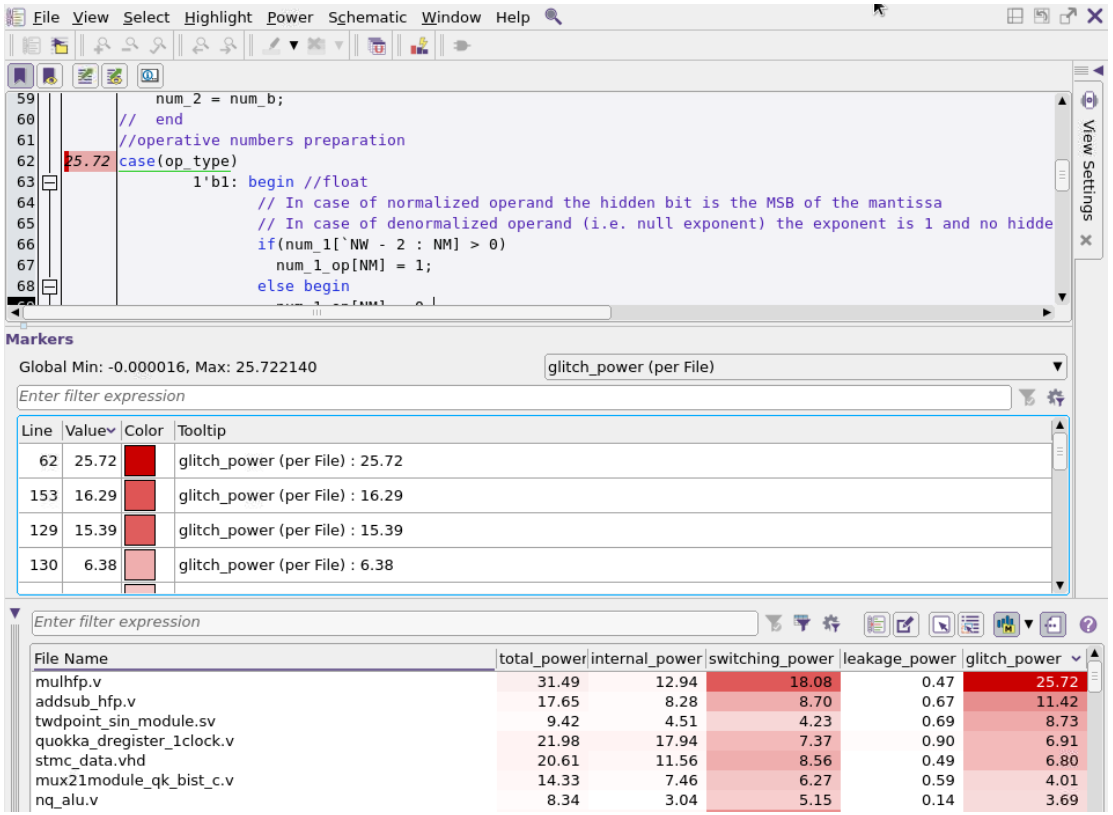
- Glitches performed enabling the DSA
- Very important to align the glitch_pulse_r option to GLS value

```
set_power_delay_shifted_event_analysis_options -glitch_pulse_r 70
```

- Default pulses with width > is 100% of cell delay are propagated : not realistic and lead to glitch power underestimation .



Glitch power analysis : the Gui



cross probe the power on source code

rank the power by line

report glitches power contributor by file

- Multiplier in DSP unit is the main glitch power contributor

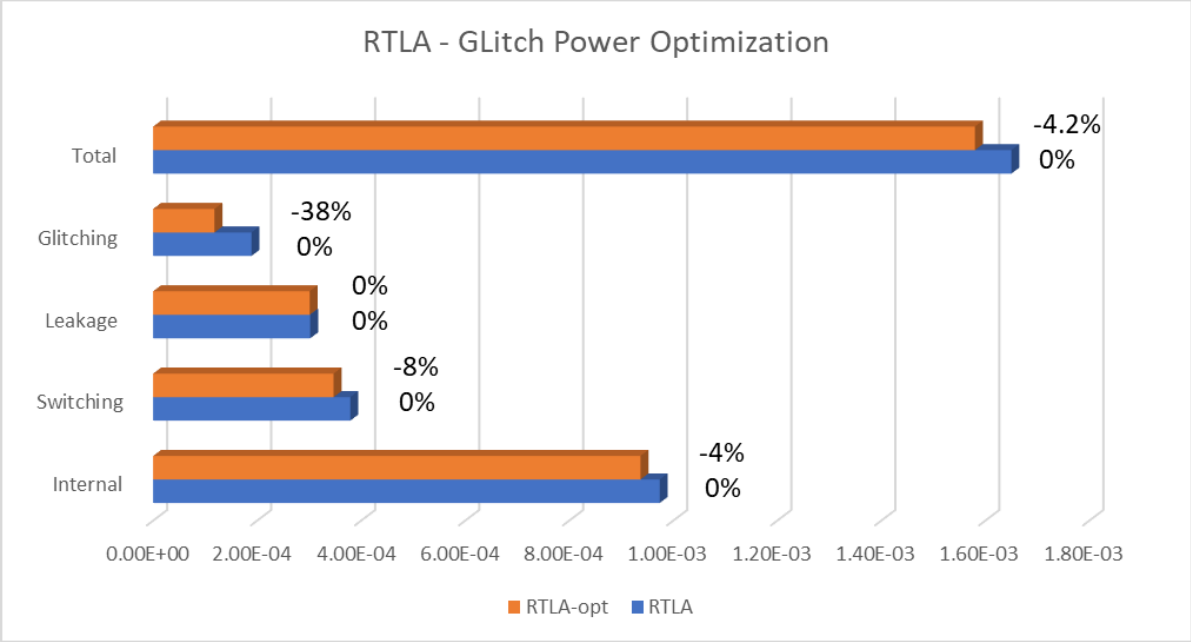
Glitch power optimization

- Possible solution on multiplier block:
  - RTL changes: add pipeline
  - Physical: equalize delays by cell swap
  - **Synthesis: enable datapath optimization**

```
Module                    :  DW_mult_uns_J46671_P0_D1
Parameters                :  a_width=11
                             b_width=11
Current Implementation:  pparch(area,speed)
Contained Operations  :  mult_117(mulhfp.v:117)
Multiplier Arch       :  benc radix4
```

- Configure new RTLA-opt changing mult. architecture

```
set_datapath_architecture_options  -mult_arch   nand \
                              [get_cells top/u_dsp/mul_module_inst]
```



- RTLA-opt (nand mult) vs RTLA (Booth-Encoder mult)
  - Glitch power reduced by 38%
  - Total Power reduction is by 4.2%

1. Power estimation flow: classic flow
2. Power estimation flow with RTLA
3. Results
4. Glitch power analysis & optimization
5. Conclusion

Conclusion

- Reduced runtime (x20)
  - ✓Avoid full implementation flow SYN → PNR → PA
  - ✓Avoid GLS, extracting activity from RTL simulation

- Shared Frontend setting improves power estimation accuracy
  - < 5% for total power
  - < 10% for clock network and <5% with CT cell load setting
  - ~ 0% for register mainly for perfect activity annotation

- Capability to explore different design options within one tool

Acknowledgments

We would like to thank

Claudio Mucci (ST) for RTLA integration in Frontend Kit

Alberto Baldi (Synopsys) for supporting on the RTLA

Matt Karsten (Globalfoundries) , Roberto Anelli (Nordicsemi),

Frank De Meersman (Synopsys) for reviewing this works.

life.augmented

snug

THANK YOU

Our
Technology,
Your
Innovation™