SYNOPSYS®
Silicon to Software™

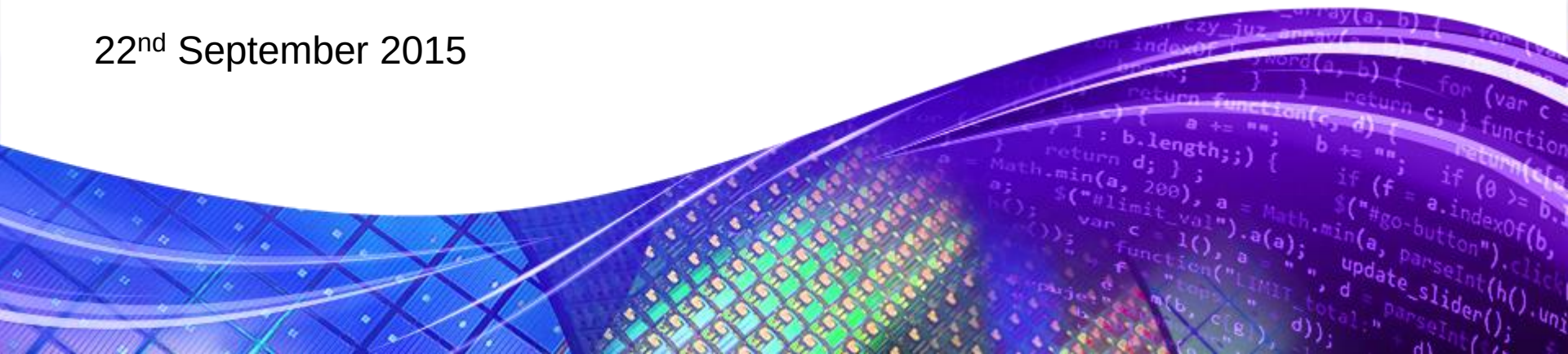
Static Power Intent Verification of Power State Switching Expressions

Srobona Mitra

Senior R&D Engineer, Synopsys India Pvt. Ltd.

Co-authors: Bhaskar Pal, Soumen Ghosh, Rajarshi Mukherjee, Kaushik De

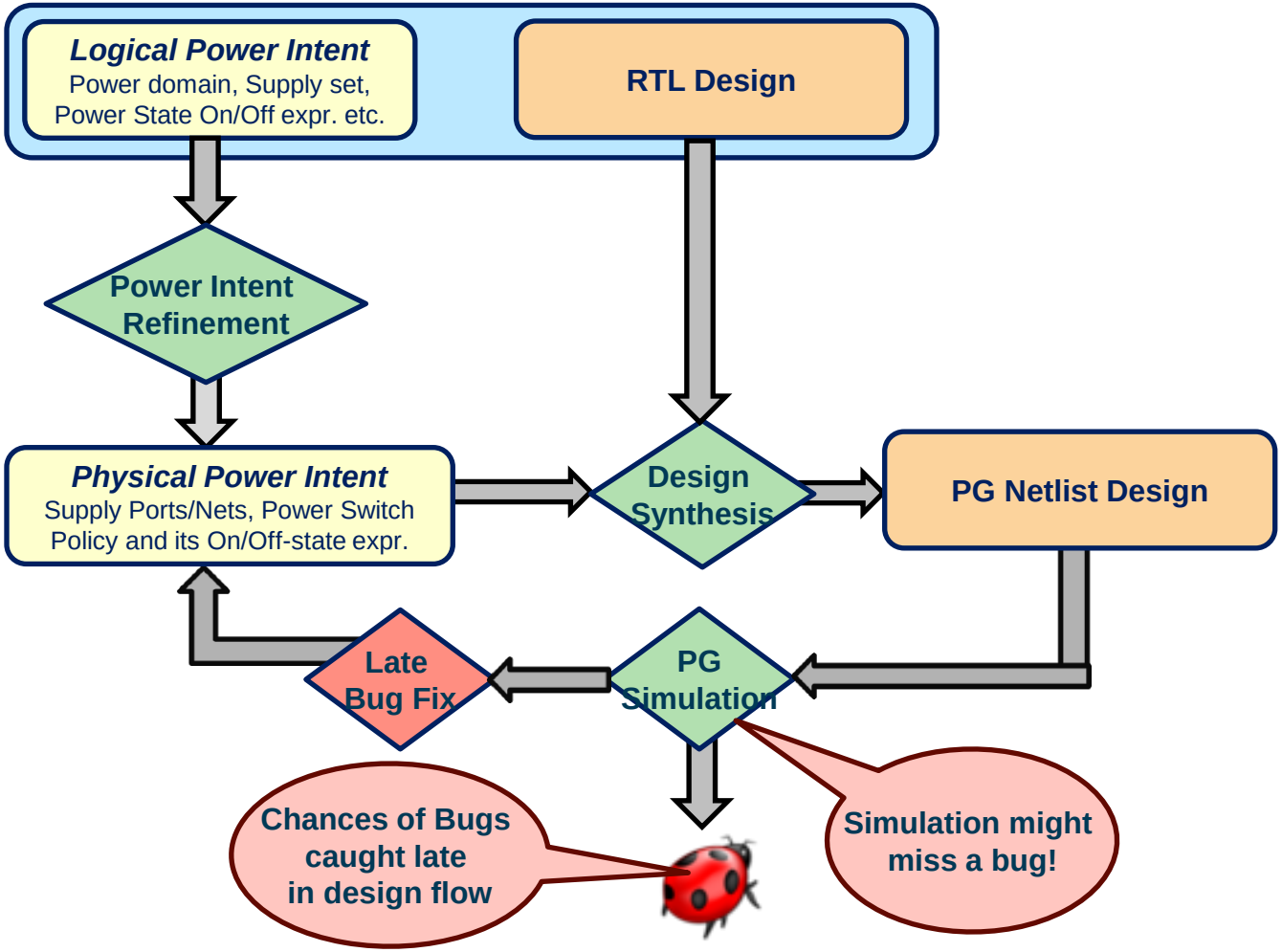
22nd September 2015



Background and Objectives

- Low power architectural intent (**UPF/CPF**) undergoes successive refinements during design cycle
- Logical Power Intent (High Level)
  - Power Domains, Supply Sets, Power State On/Off Expressions
- Physical Power Intent (Low Level Details)
  - Supply Nets, Supply ports, Supply Port/Net connections, Power Switches, Power Switch On/Off expressions
- **Requirement:** ***Initial power intent remains intact through refinements (logical to physical)***
- Erroneous refinement steps lead to ***late functional bugs***
- **Goal:** ***Static checking methodology for verifying early that the power state switching expressions through refinement steps are equivalent***

Current Low Power Design/Verification Flow



SYNOPSYS®

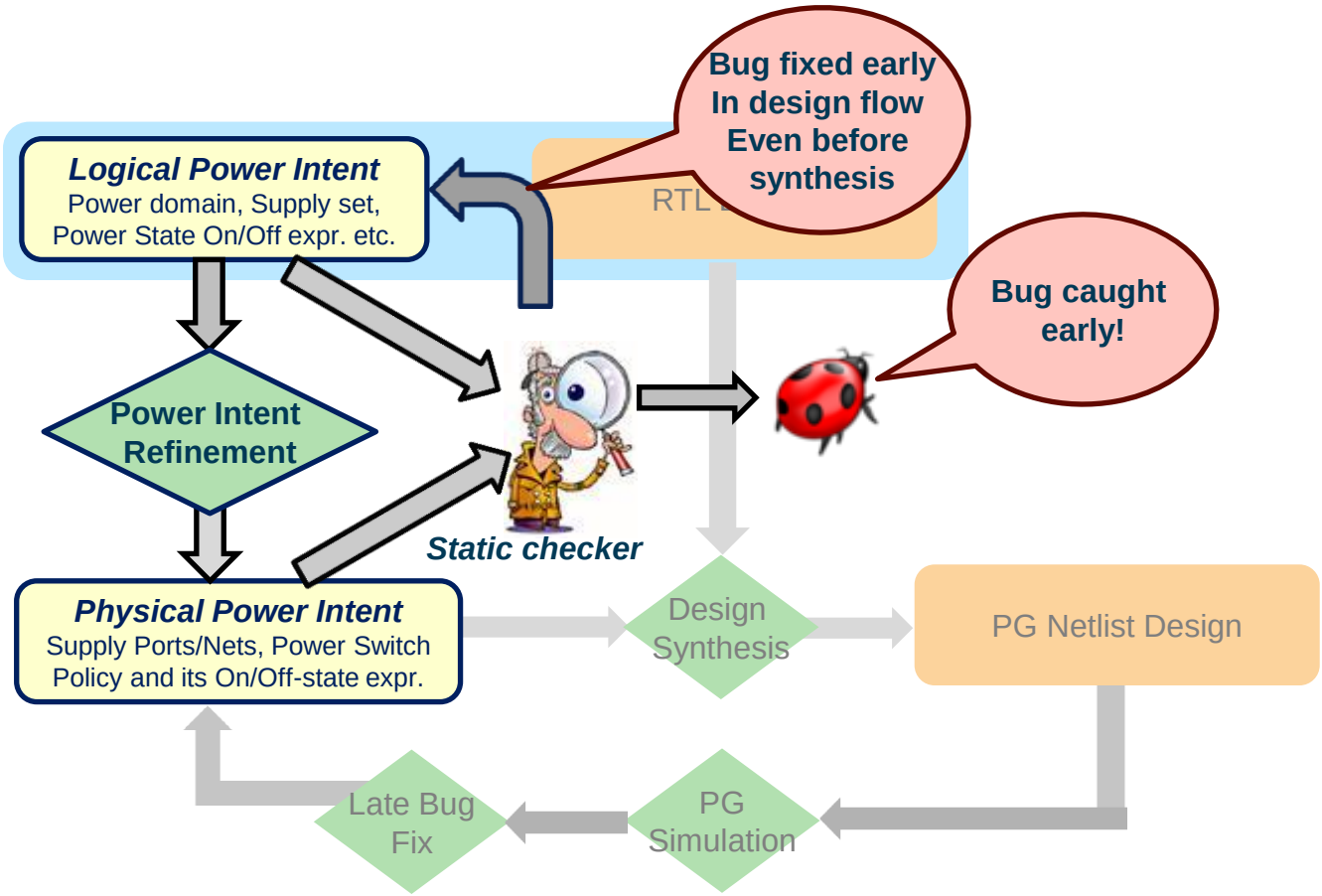
add_power_state logic expression Vs. create_power_switch logic expression

- *add_power_state* logic expression (logical UPF):
  - Uses logic expressions to specify conditions under which the power/ground net of supply set will be in a particular power state
  - Specifies the on/off conditions for these supply nets
  - Turns on/off the power domain whose primary supply is this supply set
- *create_power_switch* logic expression (physical UPF):
  - Specifies the on-state/off-state conditions for a particular power switch policy through logical expressions on logic nets.
  - Specifies logical condition under which its output supply net turns on/off
  - Turns on/off the power domain powered by this output supply net, controlling the on/off state of the domain.

Problem Definition

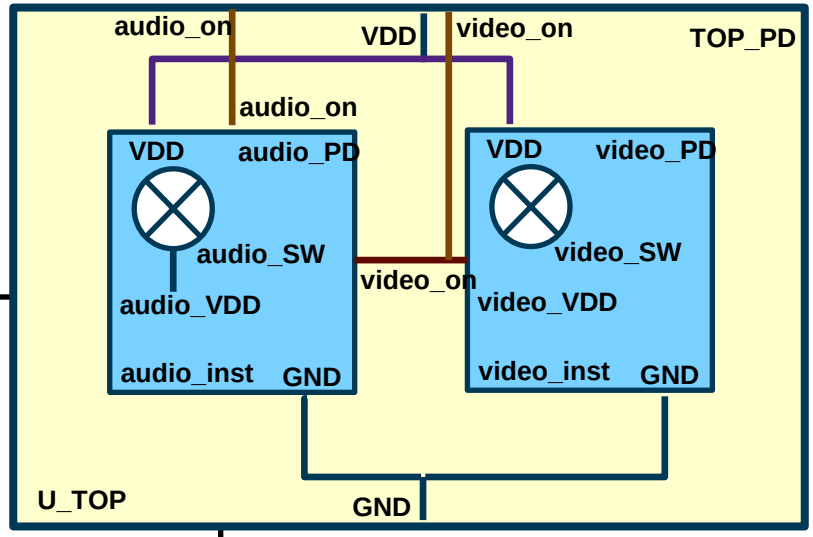
- **Given:**
  - ***add_power_state*** logic expressions
  - ***create_power_switch*** on/off state logic expressions
- **Goal:**
  - To prove by static checking of power intent that the *add_power_state on/off logical expressions* are equivalent to the corresponding *create_power_switch on/off logical expressions* for each supply net
  - If the equivalence cannot be proved, catch and report the corresponding violations early in the design cycle, as soon as the physical power intent is available

Design Flow with Proposed Early Power Intent Static Checking



SYNOPSYS®

Illustration through Example



LOGICAL UPF

```
create_power_domain TOP_PD  \
                  -include_scope
create_supply_set   video_SS  \
                   -function {ground GND}
create_supply_set   audio_SS \
                   -function {ground GND}
create_power_domain  video_PD \
        -elements {video_inst} \
        -supply {primary video_SS}
create_power_domain  audio_PD \
        -elements  {audio_inst} \
        -supply {primary audio_SS}

add_power_state  video_SS \
   -state video_PWR_ON { \
   -supply_expr {power==`{FULL_ON,1.2}} \
   -logic_expr {video_inst/video_on}}

add_power_state  audio_SS \
   -state audio_PWR_ON { \
   -supply_expr {power==`{FULL_ON,1.2}} \
   -logic_expr {video_inst/video_on || \
                audio_inst/audio_on}}
```

PHYSICAL UPF

```
create_supply_port  VDD
create_supply_net    VDD
create_supply_port  GND
create_supply_net   GND
connect_supply_net  VDD -ports {VDD}
connect_supply_net  GND -ports {GND}

create_supply_net audio_VDD

create_supply_set   audio_SS \
   -function {power audio_VDD} \
   -update

create_power_switch audio_SW \
   -domain audio_PD \
   -input_supply_port {VDD VDD} \
   -output_supply_port {VDDV audio_VDD} \
   -control_port {ENB audio_inst/audio_on} \
   -on_state      {SW_ON VDD ENB}
```

Inconsistency! leads to low power bug!

Scenario:
video_inst/video_on *ON*,
audio_inst/audio_on *OFF*

=> ***audio_SW*** *OFF*
=> ***audio_PD*** *OFF*

Functional bug!

SYNOPSYS®

Results

- Proposed static checking implemented in existing static checker flow
- Enhanced static checker run on industrial benchmark designs of sizes ranging from around ***20 thousand sequential elements and around 3 lakhs nets*** to very large benchmark designs with approximately ***2 million sequential elements and 15 million nets***
- Demonstrated ***negligible overhead*** in terms of time and memory to the existing static checking runtime
- Able to ***report inconsistencies*** in logic expressions in these designs at physical UPF level, demonstrating the viability of our approach

Design	Time Without(s)	Memory Without(MB)	Time With(s)	Memory With(MB)
Design1	427.93	5435	433.49	5435
Design2	1138.23	2969	1161.24	2975
Design3	2927.33	22501	3014.76	22720

Conclusions

- Inconsistency between on/off-state conditions between logical and physical power intents results in a **functional problem** in the design
- Results in functional violations in power intent, manifests itself very late in the cycle as **low power design bug**
- For this problem to be detected in simulation, simulation must exercise the required triggering condition for this problem
  - **Problem: inherent incompleteness of simulation**
- The functional bug would get detected in simulation very late in the design flow, when **bug fixing at both physical power intent and physical design level becomes prohibitively expensive**
- **Our main contributions:**
  - ***Completeness:*** Guaranteed to find any inconsistency, if exists
  - ***Early Catching of Bugs:*** We catch the bugs early at power intent level, without having to wait for synthesized design

References

[1] Synopsys-MVRC

www.synopsys.com/tools/Verification/lowpowerverification/pages/mvrc.aspx

[2] VCS with MVSIM

www.synopsys.com/Tools/Verification/.../Pages/MVSIM.aspx

[3] *Unified Power Format (UPF 2.0) Standard [Draft Version]*, IEEE P1801/D18,23rd October, 2008

[4] *Common Power Format (CPF 1.0) Standard [Draft Version]*, Silicon Integration Initiative Inc., 2*nd* January, 2007

[5] Synopsys-Design Compiler

www.synopsys.com/Tools/Implementation/RTLSynthesis/DesignCompiler/Pages/default.aspx

[6] Synopsys-IC Compiler

www.synopsys.com/Tools/Implementation/PhysicalImplementation/Pages/ICCompiler.aspx

Thank you!

Email: {srobona, bpal, gsoumen,rmukherj, kaushikd}@synopsys.com

