2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Automatic Investigation of Power Inefficiencies

~ Activity Analysis in RTL Simulation ~

Kuo-Kai Hsieh, Li-C. Wang

University of California, Santa Barbara

Wen Chen, Monica Farkash, Jayanta Bhadra

NXP Semiconductors

UCSB

NXP

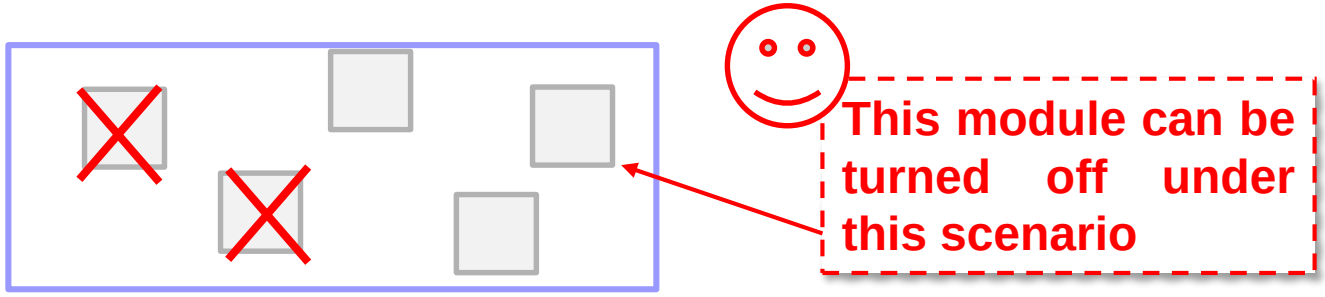
Power Reduction

- Power consumption – increasing issue
- Different methods to reduce consumption by design
  - Power gating
  - Clock gating
  - Dynamic voltage and frequency scaling
- Difficult to identify missed opportunities to decrease power consumption

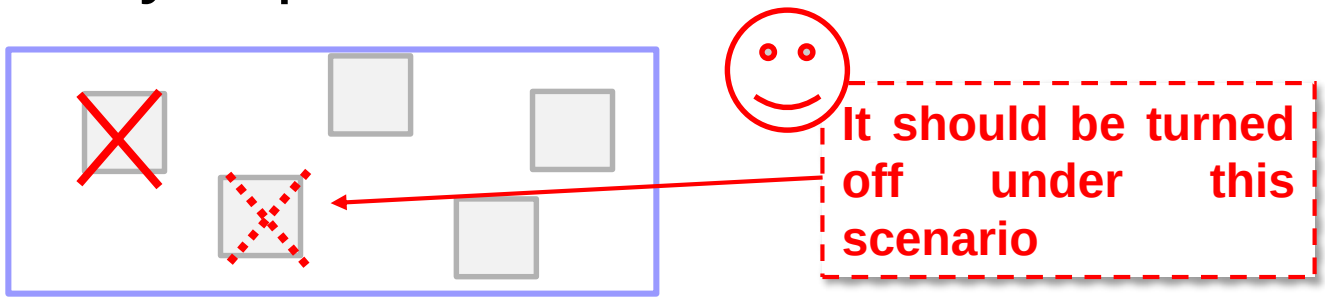
This work presents an automatic process which identifies such missed opportunities and reveals them to designers for improvement.

Questions – at SoC Level …

- How do we know if there are missed power optimization opportunities in the design?



- How do we know if the power reduction mechanisms are correctly implemented?



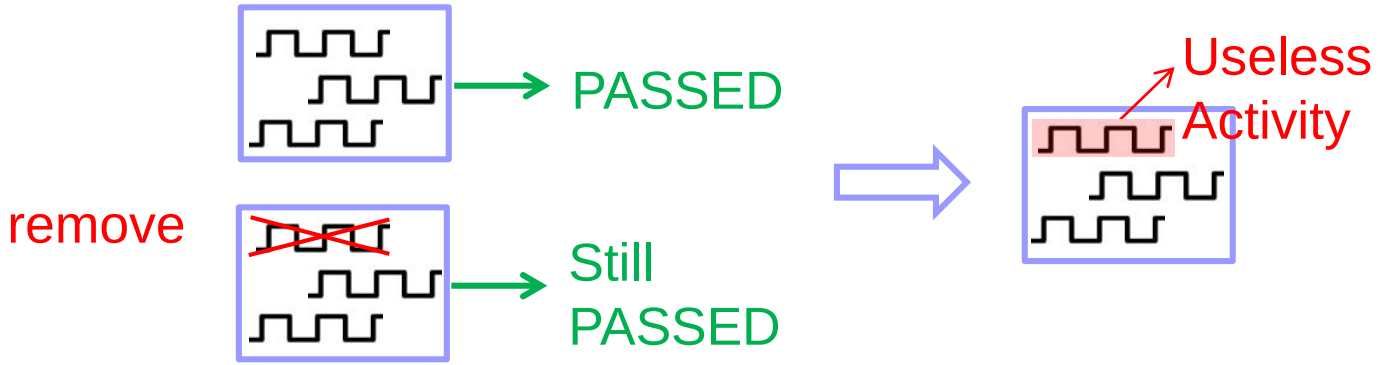
2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Features of the Proposed Method

- Automatic methodology
- Leveraging functional regression tests
- Toggle coverage analysis
- Can be easily integrated to design environments without much efforts

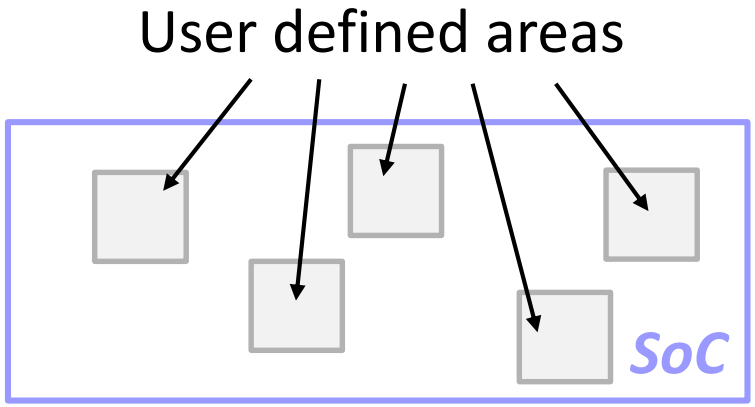
Basic Idea

- A power aware design will only consume power which is required by its task(s)
- Removing any activities, a power aware design should fail its tasks
- There is no useless activity in a power aware design



Area-based Activity Removal

- Focusing on a single signal is infeasible
- We defined several **areas** in our SoC
  - Some modules may be power-inefficient, e.g. prefetch buffer, and it will be excluded
  - We investigate power optimization opportunity in these area



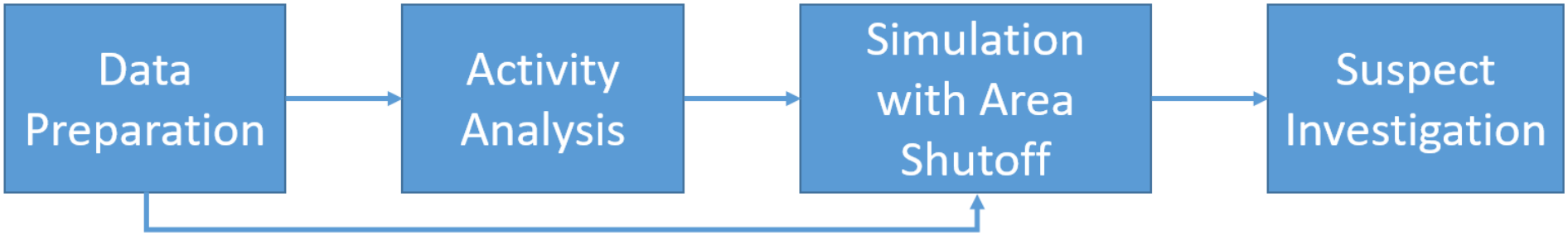
When to Remove Activity?

- The time to remove activity can be customized to each area and test scenario
- We decided to remove the activity of an area from power-on reset de-assert
- It takes less effort to setup the environment
  - But we may miss potential power optimization opportunity

Available Test Scenarios to Activate All The Areas?

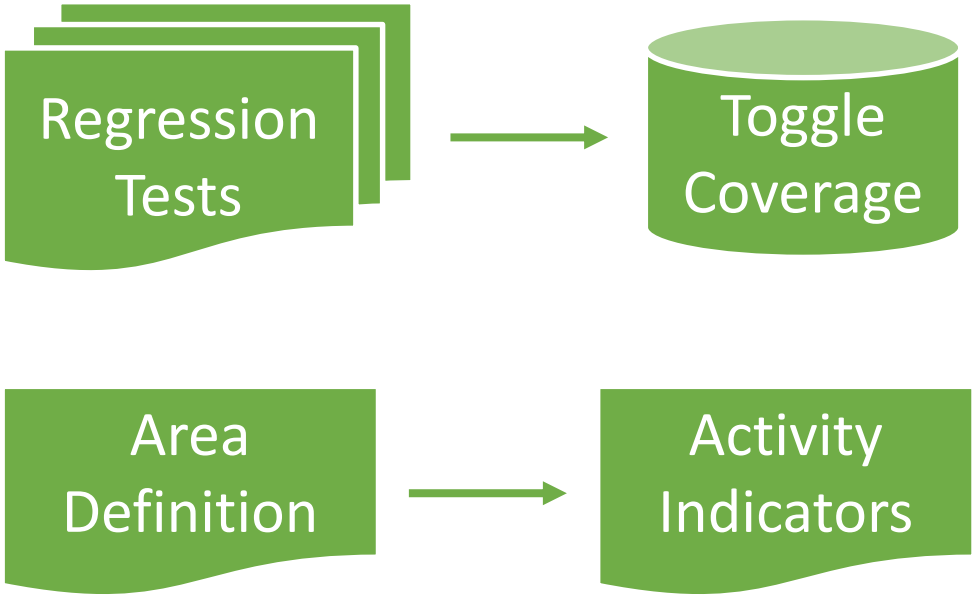
- Leveraging functional regression tests
- They are **diverse** and cover all the basic scenarios
- They are **available** in the verification process so we can re-use them
- Also, the proposed method can be built upon the existing verification environment

Methodology Overview



Data Preparation

- To start with, we need
  - Functional regression tests
  - Defining the areas (based on users' choice)



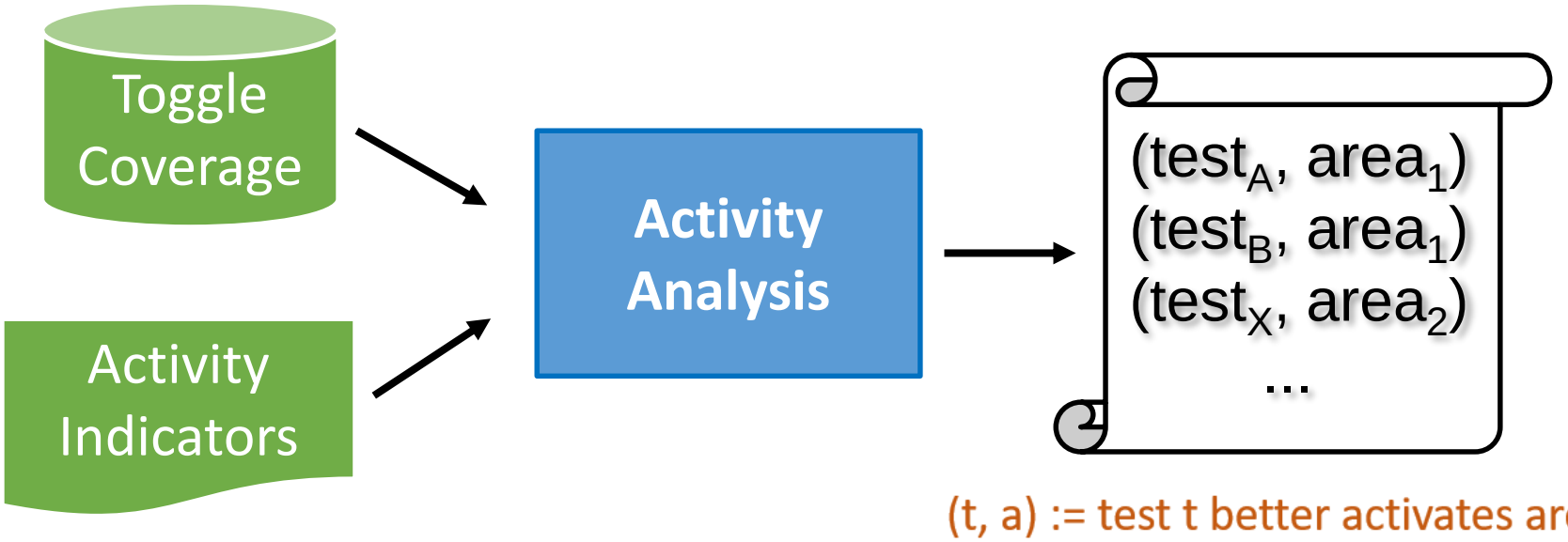
Area Definition Example

```
<root>
<region>
    <name>Area_1</name>
    <hierarchy>testbench.top.foo</hierarchy>
</region>
<region>
    <name>Area_2</name>
    <hierarchy>testbench.top.bar.xxx</hierarchy>
</region>

...
</root>
```

Activity Analysis

- Running all the pairs of (test, area) is too expensive.
- For each area, find tests that can better activate it.



Activity Analysis

- Not all the signals are good indicators
  - For example, primary inputs can toggle all the time but the area is idle.
- Primary outputs are good indicators
  - Also, they can be obtained automatically.
- Activity is obtained by the sum of toggling of activity indicators in that area.
  - Toggle coverage record the ***number of switches*** for each signal.

Obtaining Primary Outputs

- For example, using simvision

process to report selected signals of a given path

Open shm database

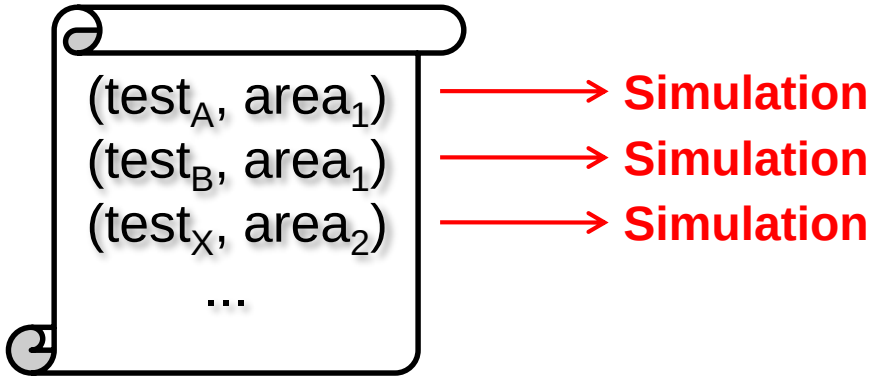
Set output signals

```
proc retrieveSignals {fp name path} {
    dbfind set -scope $path
    dbfind search
    set signals [dbfind find -result %n]
    set line [concat "$name $path" $signals]
    puts $fp $line
}

database open test.shm
dbfind open -database simple_shm_cov -type signal \
    -searchinputs False -searchoutputs True       \
    -searchinouts False -searchinternals False    \
    -searchfibers False -searcherrors False       \
    -searchassertions False -signalformat path    \
    -scopeformat path
```

Simulation with Area Shutoff

- Given a pair (test, area), run the test with area disabling



- Disabling an area is achieved by freeze all the output signals in the area
  - Using TCL, or
  - Using HDL and plusargs options

Example TCL for Area Disabling

- Example TCL to IUS

```
proc freeze_sig {sig} { force $sig #$sig }

proc freeze_area {
   freeze_sig <sig_1>
   freeze_sig <sig_2>
   ...
   freeze_sig <sig_N>
}

run 100 ns
# Set a break point for freezing the given area
stop -name break_rst -delbreak 1 \
        -condition {#<end_of_reset_indicator> == 1'b1} \
        -continue -execute {freeze_area}
run
```

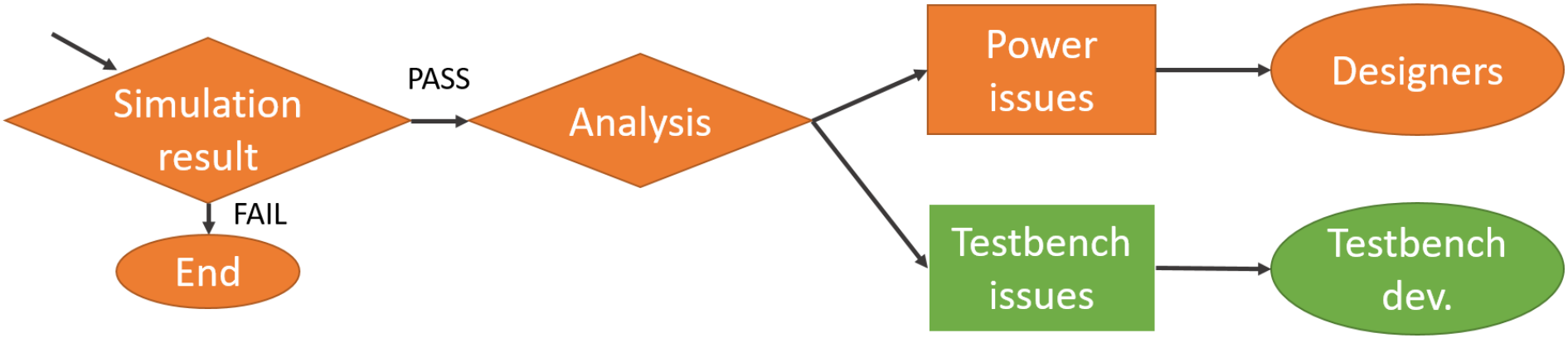
Force signal to its current value

Force an area (generated by scripts)

Disable and area after reset

Suspect Investigation

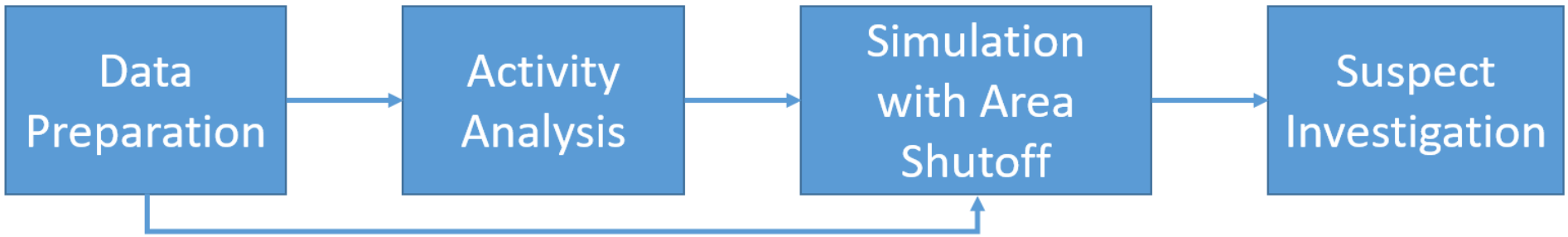
- Analysis starts with the activity which was disabled but the test still passed
- Issues can be a true power issues
- Issues can be testbench issues such as missing checkers



Short Review

- What is the effort required to use the proposed methodology
  1. Area definition
  2. End-of-reset indicator

- Others can be done with scripts



2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

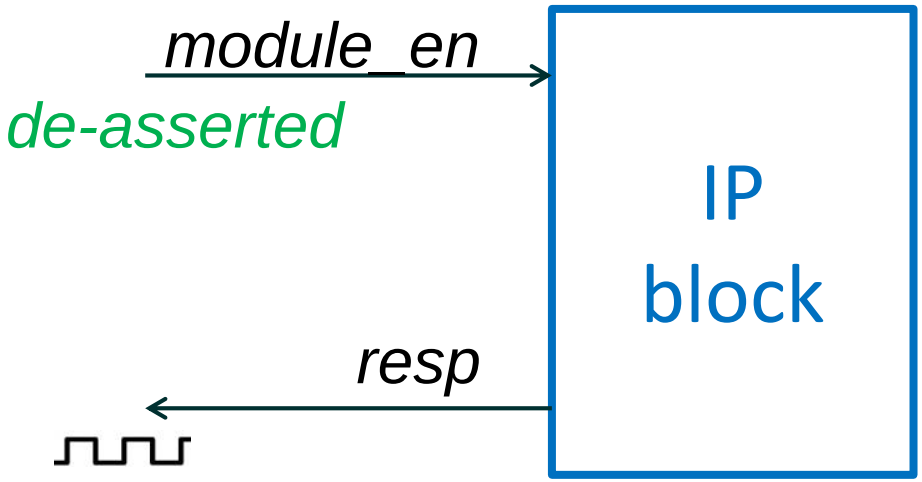
Experiments

- Real life experiment
  - Commercial dual-core microcontroller SoC
  - 1400 regression tests
  - 100 areas
  - Ran at most 200 high-toggle tests for each area

- Findings
  - 9 tests having useless activity
  - 2 design issues, 3 testbench issues

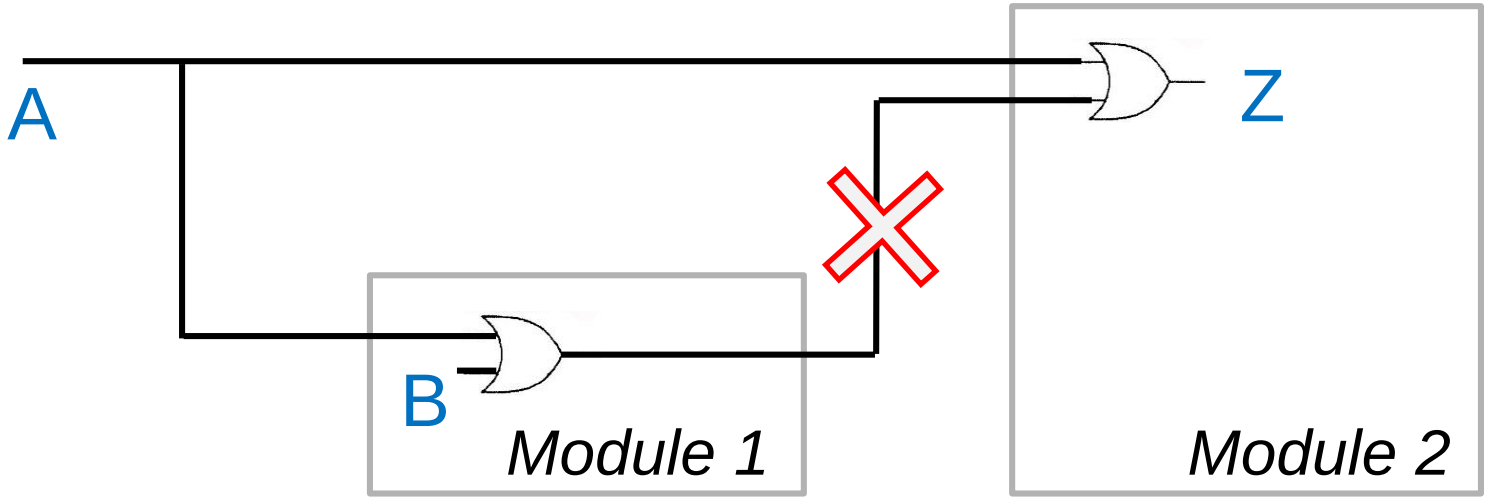
Design Issue 1

- Unnecessary switching at primary outputs of IPs.
  - resp is a primary output and a bus signal.
  - It is used only when module_en is high, but it toggled frequently



Design Issue 2

- Redundant logics
  - Signal A is ORed twice, i.e. Z = A|B|A
  - Hard to find the issue because the two modules are not close to each other in RTL hierarchy
  - It leads to more power, more area and worse timing



Testbench Issues

- Missing checkers: Design didn't enter stop mode but no checker detected it.
  - This test try to verify a functionality under stop mode
  - It failed to detect the failure of entering stop mode
- Missing checkers: Bus protection signals were not checked.
- Redundant configuration
  - Set configuration to its default multiple times

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Conclusion

- An automatic flow to detect missed power optimization opportunities.
- Leveraging regression suites. Able to be seamlessly integrated to the design environments.
- Able to point out where the testbench needs improvement. It is a testbench certification approach in the context of low-power verification.

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank you