

Unveil the Mystery of Code Coverage in Low Power Designs (Achieving Power Aware Verification Closure)

Madhur Bhargava, Mentor, A Siemens Business

(madhur_bhargava@mentor.com)

Durgesh Prasad, Mentor, A Siemens Business

(durgesh_prasad@mentor.com)

Pavan Rangudu, Mentor, A Siemens Business

(rangudu_pavan@mentor.com),



Agenda

- Introduction
- Motivation for paper
- Challenges in code coverage of low-power designs
- Case studies and examples
 - Addressing the challenges
- Conclusion

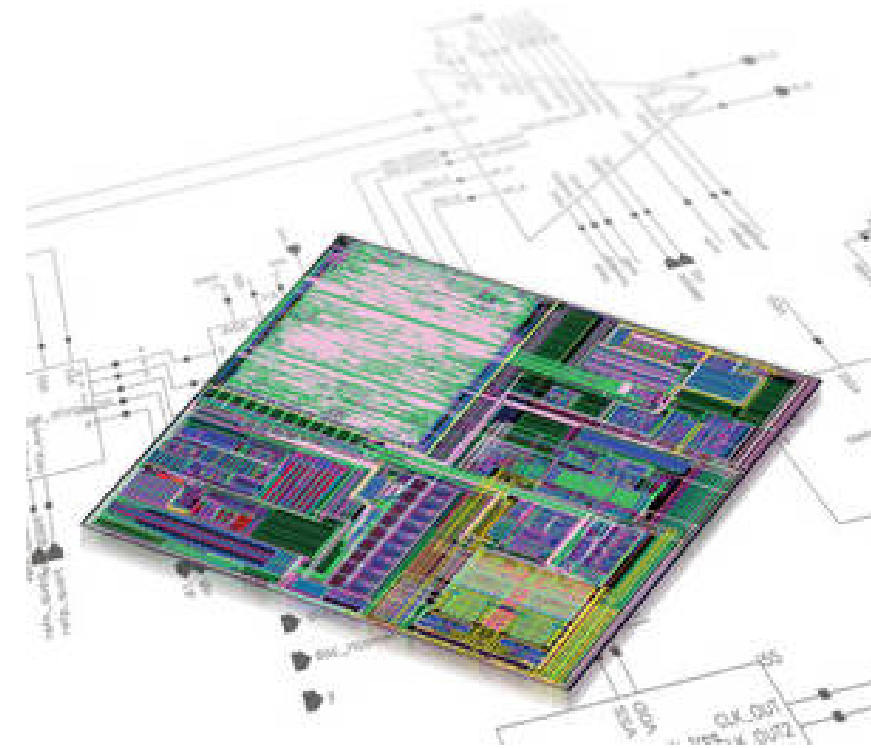
Introduction

Today's SoCs

- Are incredibly Complex
- Have sophisticated power management strategies for highly power efficient design
- Make use of various coding styles and have complex power aware and non-power aware macro models
- Integrate variety of implementation cells
 - Isolation, retention, buffers etc.

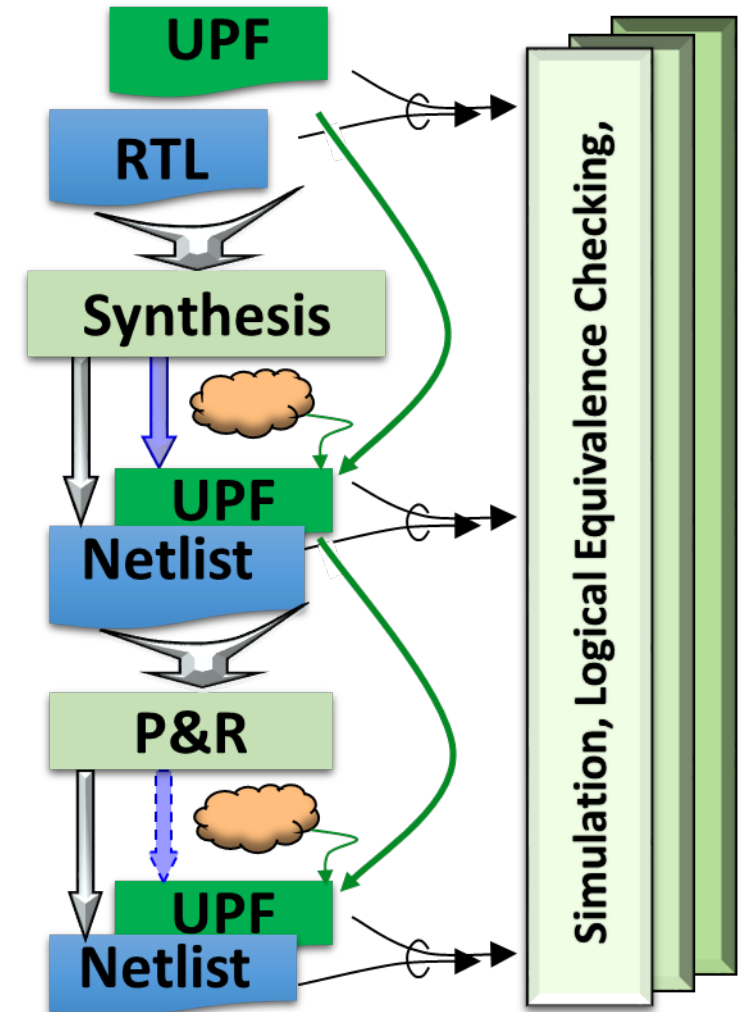
They Must

- Verify the power management
 - Make sure 100% code coverage and low-power coverage



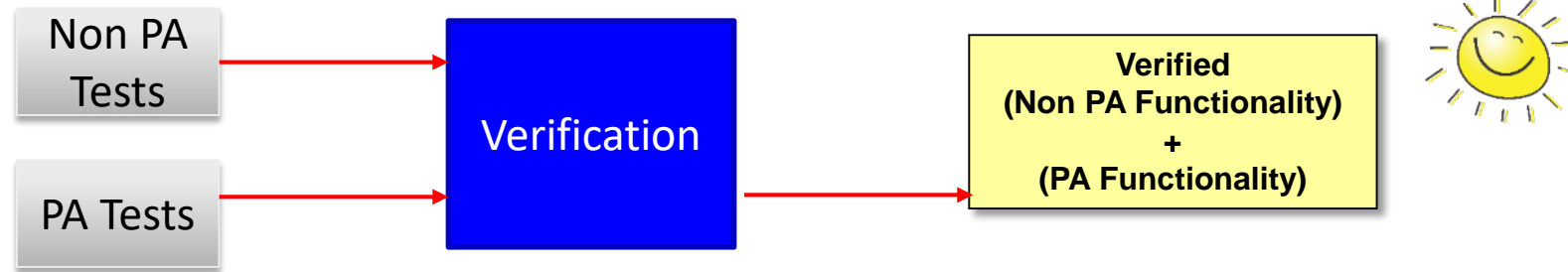
Unified Power Format(UPF)

- RTL is augmented with a UPF specification
 - To define the power architecture for a given implementation
- RTL + UPF drives implementation tools
 - Synthesis, place & route, etc.
- RTL + UPF also drives power-aware verification
 - Ensures that verification matches implementation



Motivation

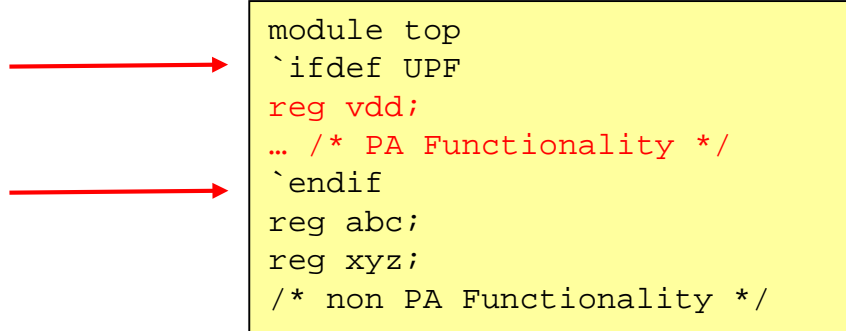
- Low-Power is now de-facto
- Low-Power design RTL is changed by simulator to make it power aware
- A typical low-power regression setup



- **No standard for modeling of code coverage in low-power designs**
 - UCIS has no support for low-power
- Need for verification plan to achieve closure for low-power verification
 - Coverage of power objects (all possible states and transitions)
 - 100% code coverage of user RTL

Challenges - Code Coverage of Low-Power Instrumented Design

- PA logic inside “ifdef PA”
 - Block gets activated/covered only when low-power simulations are run
 - Guideline: Guard the low-power functionality and avoid enabling the PA code in Non-PA runs
- Functional Coverage
 - Coverage of assertions (checker logic):
 - Non PA assertions can get triggered during power-off : False alarms
 - Simulation tools generally disable assertions during the power down period
 - Disable the coverage of these assertions during the power down period.
 - Covergroup based coverage
 - Functional coverage done using covergroups does not have any impact and can be easily achieved in low-power designs.



```
module top
`ifdef UPF
reg vdd;
... /* PA Functionality */
`endif
reg abc;
reg xyz;
/* non PA Functionality */
```

Challenges Contd..

- Power Controlling Logic: PA Coverage
 - Verification closure plan requires coverage of power objects
 - Low-power coverage is handled separately by the verification tools
- Low-Power Designs having “Hard Macros: PA Behavior Model”
 - Power-aware behavioral model: power behavior is modeled inside the model itself
 - Visible only when the UPF connections are made

```
module ana_mac(... ip1, ..)
wire vdd = 1; wire vss = 0;
always @(vdd, vss, clk)
begin
if (vdd === 1'b1 && vss === 1'b0)
d = clk & a1;
else
d = 1'bx;
end
```

- **UPF connections to the supply pins vdd and vss**
- **No PA logic inserted**
- **RTL code coverage does not pose any challenge**
- **Toggle coverage of supply pins (vdd, vss) is not considered**
- **Code coverage numbers (RTL) differs in a Non PA simulation (always on) Vs PA Simulations (Supplies going on/off)**

Challenges Contd..

- Soft Macros

- Verification tool inserts the isolation, level shifter cells and other pa cells
- Insert some power logic into the design in order to do power aware

```
always @(*)  
  out = in1 & in2;
```

- Whenever 'in1' or 'in2' changes, the statement gets hit
- if the power of this part of design is OFF,
 - then this statement will not get triggered;
 - signal 'out' will get a value 'x'
- When power is enabled
 - the assign statement will get triggered
- Moreover, the number of times this statement gets triggered now also depends on power along with 'in1' and 'in2'

Types of code coverage (Challenges & How to address them)

- Line Coverage

Actual D-FlipFlop RTL logic

```

1. always @(posedge clk, posedge reset,
   posedge set) begin
2.     if (reset)
3.         q<=1'b0;
4.     else if(set)
5.         q<=1'b1;
6.     else if(clk)
7.         q <= d;
8. end

```

PA Instrumented D-FlipFlop RTL logic

```

1. always @(posedge clk, posedge reset,
   posedge set, posedge PWR) begin
2.     if (~(PWR))
3.         q <= 1'hx;
4.     else
5.         if (reset)
6.             q <= 1'h0;
7.         else if (set)
8.             q <= 1'h1;
9.         else if (clk)
10.            q <= d;
11. end

```

- New Lines get introduced**
- Code coverage on this PA instrumented RTL logic will not give proper results**

Solution

- Exclude the coverage of new lines/statements**
- Original RTL line number remain same**
- Set new lines numbers as "0"**

Types of code coverage (Challenges & How to address them) contd..

- Conditional/Expression Coverage

Actual RTL Expression	PA-Instrumented RTL Expression
assign c = a&b;	assign c = (PWR) ? (a&b) : 1'bx;

- Challenges

- Input terms for Expression coverage will be **PWR**, a and b
- Increase in FEC Expression input terms
- coverage results will not give proper results as expected on a non-pa RTL logic

- Solution

- exclude the input terms that have been additionally added

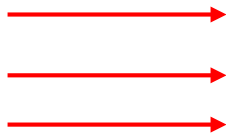
Types of code coverage (Challenges & How to address them) contd..

- Branch Coverage.

```
1. always @(posedge clk, posedge  
   reset, posedge set, posedge PWR)  
   begin
```

```
2.   if (~(PWR))  
3.     q <= 1'hx;  
4.   else
```

```
5.     if (reset)  
6.       q <= 1'h0;  
7.     else if (set)  
8.       q <= 1'h1;  
9.     else if (clk)  
10.      q <= d;  
11.  end
```

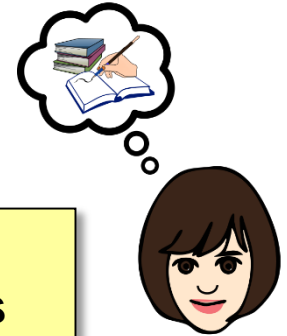


**New Branches
Introduced**



**Exclude
Extra Branches**

(Does not capture
activity when power
goes down) – PA
Coverage



Types of code coverage (Challenges & How to address them) contd..

- Toggle Coverage
 - Report how many times signals and ports are toggled during a simulation run
 - Insertion of power logic into the RTL logic, toggling of signals and ports may increase
 - Always different results in toggle activity of RTL signals in PA & Non PA (always on) runs

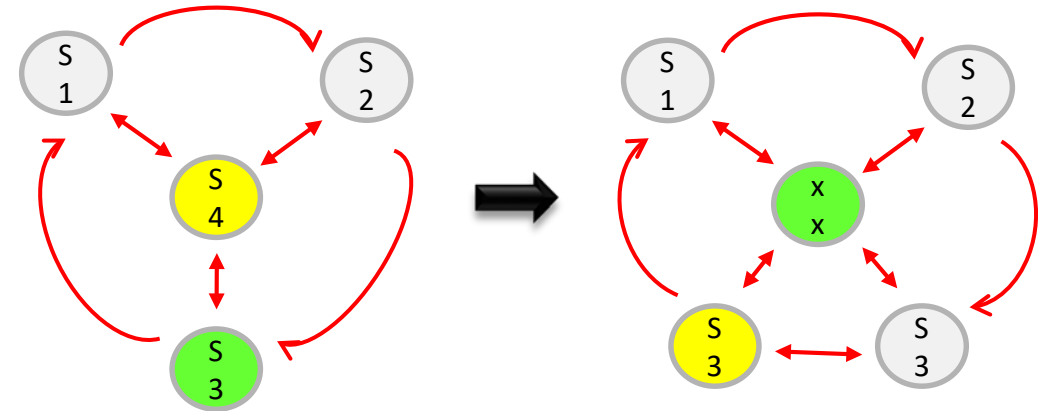
Actual D-FlipFlop RTL logic	PA Instrumented D-FlipFlop RTL logic
<pre> 1. always @(a) begin 2. t = 1'b1; 3. #1 t = 1'b0; 4. end </pre>	<pre> 1. always @(a, PWR) begin 2. if (~(PWR)) 3. t <= 1'hx; 4. else 5. begin 6. t = 1'b1; 7. #1 t = 1'b0; 8. end </pre>

't' toggles when 'a' changes

't' additionally toggles at "PWR" off->on

Types of code coverage (Challenges & How to address them) contd..

- State/FSM Coverage
 - States defined assuming design is always powered-up
 - During a low-power simulation, when the power goes off
 - Object enters undefined (verification tool added states)
 - Introduction of a new state will not give proper coverage results
- Solution
 - Exclude this extra state
 - Powered down state captured in PA Coverage

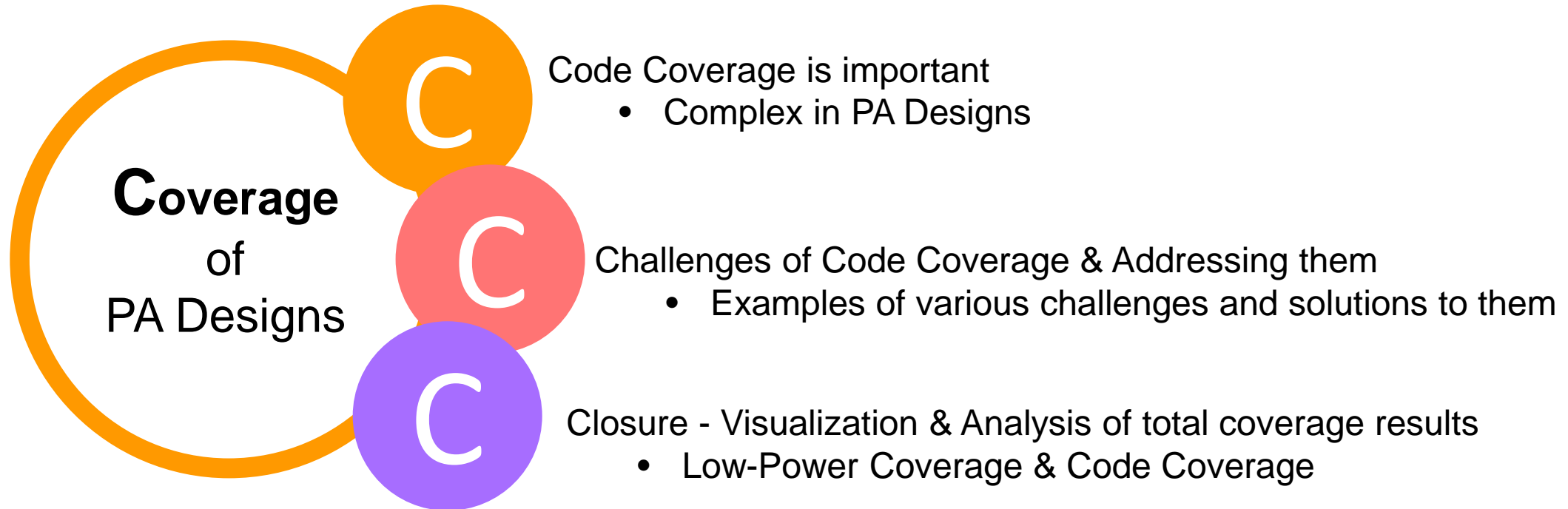


Low-Power Coverage

- Low-Power coverage
 - Together with code coverage leads to verification closure
 - Ensure that adequate testing of power aware elements of the design
- How ?
 - Tool defined low-power coverage
 - User defined low-power coverage using covergroups
 - Using bind_checker calls
 - Random directed coverage methodology



Conclusion



References

- [1] IEEE Std 1801™-2015 for Design and Verification of Low Power Integrated Circuits. IEEE Computer Society, 05 Dec 2015.
- [2] “Awashesh Kumar, Madhur Bhargava”, Unleashing the Power of UPF 3.0: An innovative approach for faster and robust Low-power coverage, DVCon India 2017
- [3] “Pankaj Kumar Dwivedi, Amit Srivastava, Veeresh Vikram Singh”, Let’s DisCOVER Power States, DVCon USA 2015
- [4] “Mohit Jain, Amit Singh, JSS Bharat, Amit Srivastava, Bharti Jain”, Overcoming barriers in Power Aware Simulation, DVCon Europe 2014



Q&A

Thank You!