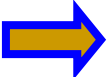


Challenges, Possible Solutions and Flow Changes for Low Power Design Verification of Complex Chips

**Narasimha Karunakar
AMD India Ltd**

Agenda

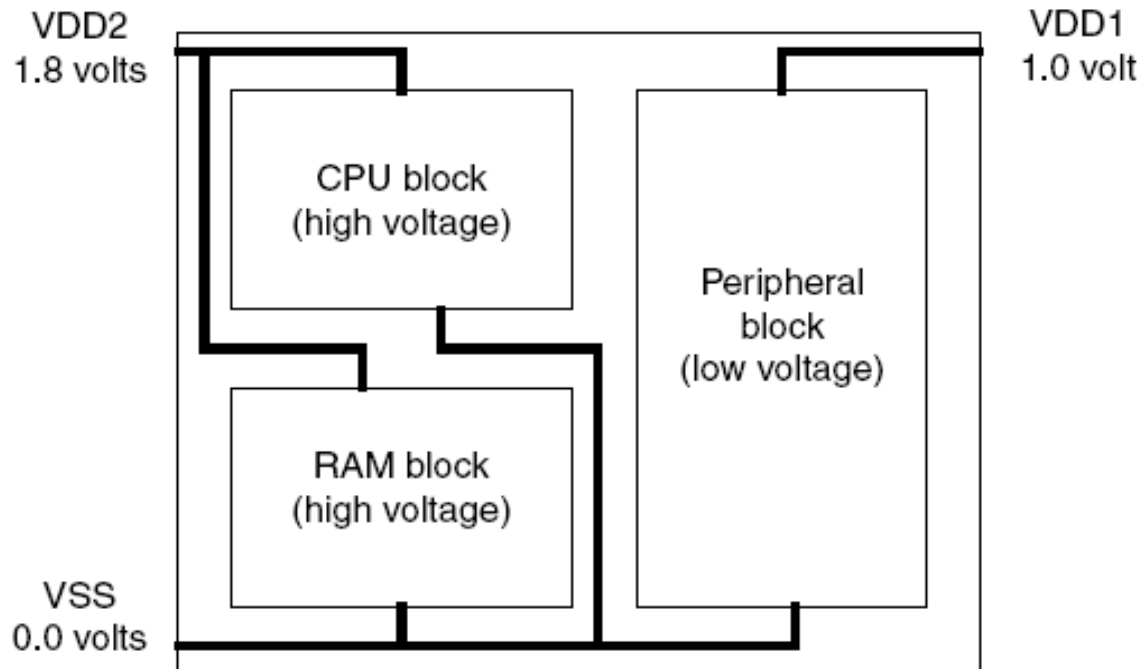
- 
- Low power design techniques overview
 - Power Aware Verification (PAV) challenges
 - Possible solutions
 - Design guidelines
 - Verification guidelines

Low Power Design Techniques: Requirement than Improvement

- Higher clock frequencies + Higher device density
=> High power dissipation
- Conflicts with increasing importance on battery life and power savings
- Exponential growth in usage of Handhelds/Laptops/Mobiles
- Performance/Watt is the new mantra!!

Low Power Techniques Overview

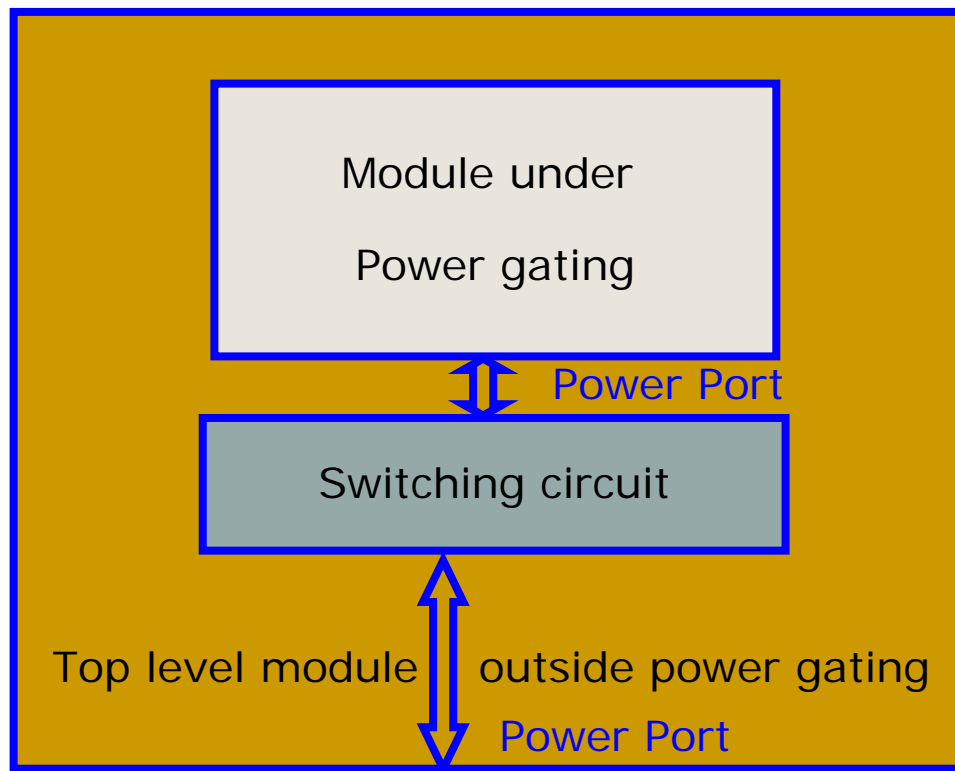
- Supply voltage reduction
 - Based on speed requirement



- Multi-Vt (Threshold Voltage) Library cells
 - High Vt transistors for low speed blocks => Low static leakage current

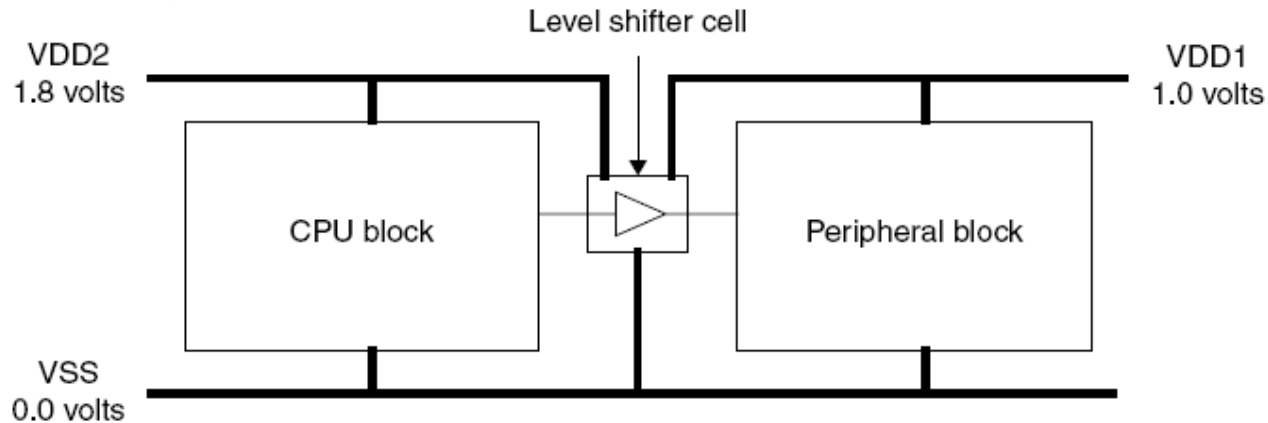
Low Power Techniques Overview

- Power switching
 - Shut down when idle.
 - Leakage power & Switching power is lowered.

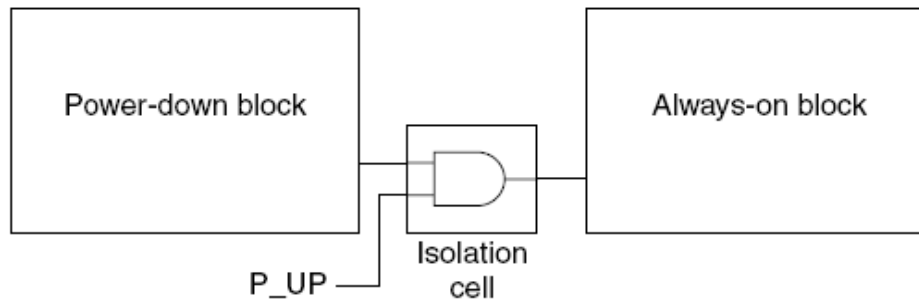


Low Power Techniques Overview

- Level shifter



- Isolation cell



Next...

- Low power design techniques overview
- ➔ • Power Aware Verification (PAV) challenges
- Possible solutions
- Design guidelines
- Verification guidelines

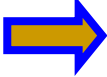
Power Aware Verification challenges

- Conventional verification infrastructure is not PAV ready
 - No notion of voltage/power
 - Protection gates modeling absent in RTL
 - Power Ports & Power Switches not modeled
 - Checks on missing protection gates not done

Power Aware Verification challenges

- Power up and Recovery sequence checks inadequate
 - All voltage/power planes tied to 1'b1 or 1'b0 – no sequencing
 - Reset asserted at time 0 allows signals to be initialized at the beginning itself
 - Flops retain value even when domain is powered off

Next...

- Low power design techniques overview
- Power Aware Verification (PAV) challenges
-  • Possible solutions
- Design guidelines
- Verification guidelines

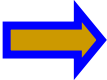
Possible solutions

- Make Verification environment voltage/power aware
 - Able to Simulate Real silicon behavior ('Z'/'X' when power down)
 - Model power switches & protection gates
 - Check for illegal power state transitions
 - Support for recovery sequences
 - Bring all state points to 'X' at power-up
 - Initialization sequence should get them back to defined state

Possible solutions

- Some Tool vendors that address the above challenges
 - Synopsys
 - MVtools (MVSIm & MVRC)
 - UPF (Unified Power Format)
 - Cadence
 - Incisive verification flow
 - CPF (Common Power format)

Next...

- Low power design techniques overview
- Power Aware Verification (PAV) challenges
- Possible solutions
-  • Design guidelines
- Verification guidelines

Why Guidelines (RTL & PAV)

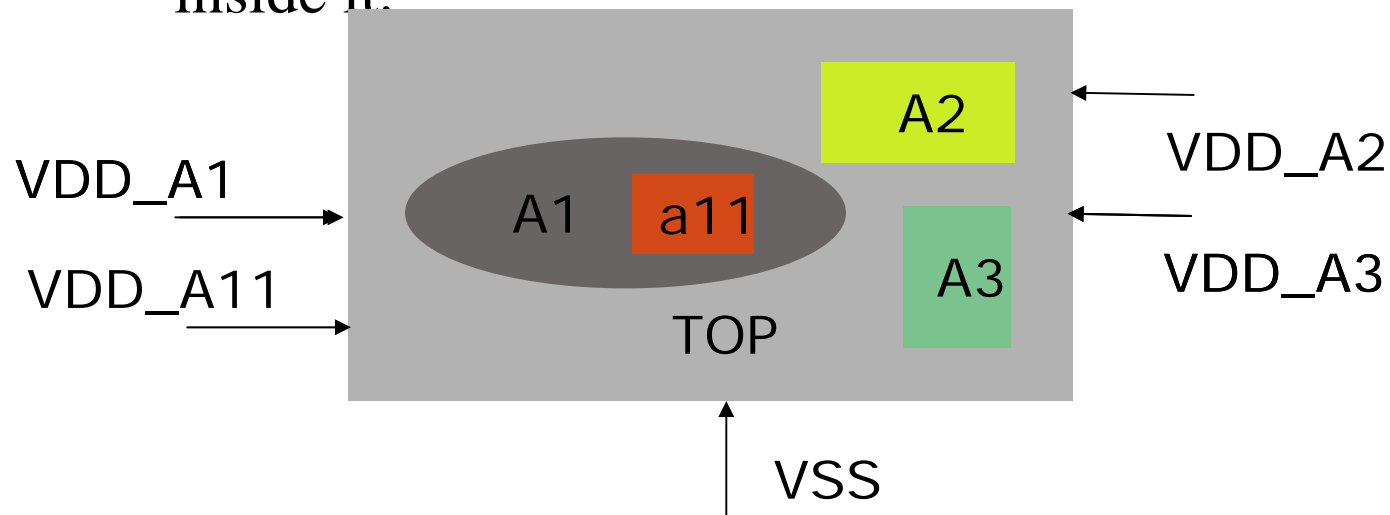
- Support power-aware tools/infrastructure
- Avoid hang issues
- Avoid false fails
- Minimize debug time
- Catch issues early in design cycle

RTL design guidelines

- Modeling the protection gates in RTL
 - Instantiation in RTL
 - For SAPR flow & Custom flows
 - Find bugs early in the game
- Firewall the ports when chip 'PwrOk' is not set
 - Correct Protection gates to be implemented between power domains

RTL design guidelines

- Partition & Boundary
 - Voltage island boundaries alignment
 - No Multiple Power domain in a final leaf module
 - Below Ex: a11 shouldn't have multiple power domains inside it.



RTL design guidelines

- Initialization of states/regs & Memories
 - Only after power ON
 - At least one power port in the port list.

```
/** Wrong Initialization **/  
initial begin
```

```
    num_phases = 5'b11100;
```

```
end
```

```
/** Correct Initialization **/  
always @(posedge VDD_IP) begin
```

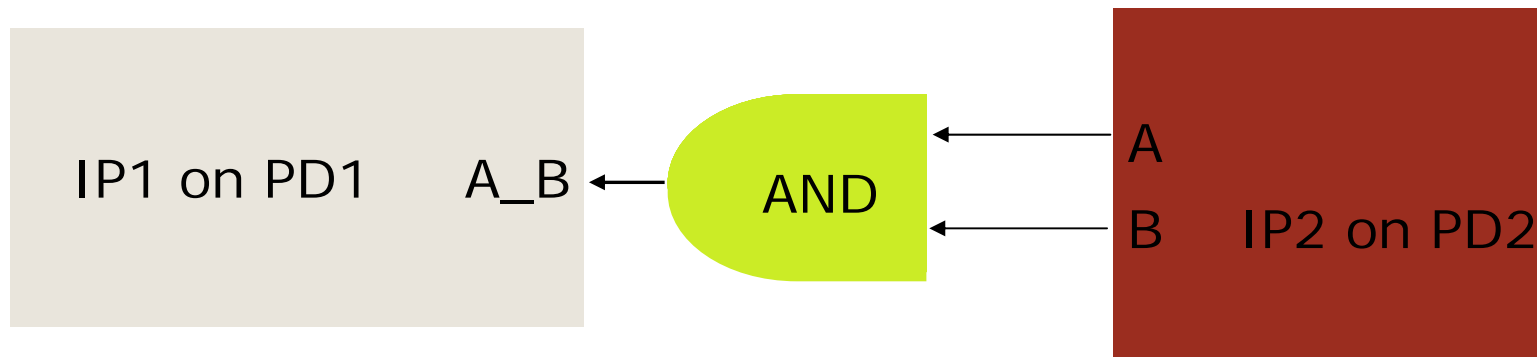
```
    num_phases = 5'b11100;
```

```
end
```

RTL design guidelines

- Expression in instantiation
 - Wrong/No power domain assigned to 'AND'

```
IP1 instance_a (  
    .A_B ( A & B ),  
    ....  
);
```

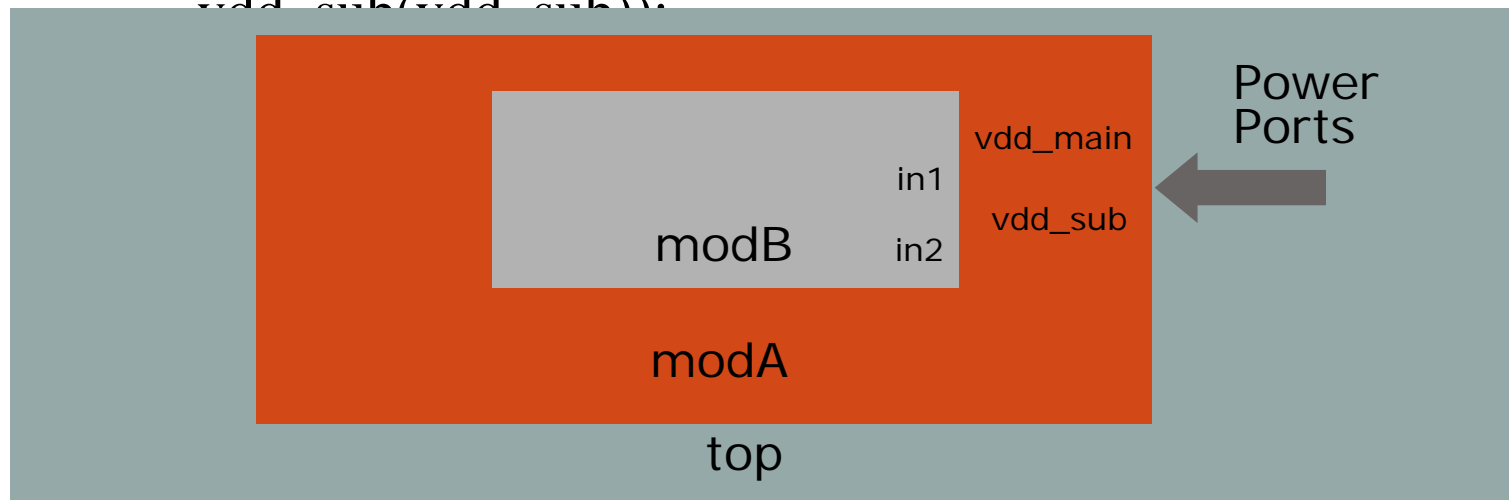


RTL design guidelines

- Avoid Constants in Port mapping
 - Avoid constants (1'b1/1'b0) & shouldn't cross domain boundaries

```
module modA;  
  modB inst_modB (.in1(abc), .in2(1'b0)); //DOMAIN_B  
endmodule
```

```
module top;  
  modA inst_modA (.vdd_main(vdd_main),  
                 vdd_sub(vdd_sub));
```



RTL design guidelines

- Correct coding practice: Use Power rails

```
module modA;  
    modB inst_modB (.in1(abc), .in2(vdd_sub)); //DOMAIN_B  
endmodule
```

```
module top;  
    modA inst_modA (.vdd_main(vdd_main),  
        .vdd_sub(vdd_sub));  
endmodule
```

Next...

- Low power design techniques overview
- Power Aware Verification (PAV) challenges
- Possible solutions
- Design guidelines
- ➔ • Verification guidelines

Verification guidelines

- Checkers should be aware of:
 - Current power state of the module to which it is associated
 - Should know that signal is 'Z'/'X' during power down.
 - Qualify the checks with PwrOk or Reset of the module

```
void Monitor() {  
    if(RTL_IP_IReset != RTL_IP_IReset[-1]) {  
        return();  
    }  
}
```

OR

```
void Monitor() {  
    If (RTL_PwrOk.IsX ==1 or RTL_PwrOk == 1'b0) {  
        return();  
    }  
}
```

Verification guidelines

- Always assume 4-state values for RTL signals:
 - Qualify the control signal to see if it is 'Z'/'X'
 - Then check for correct value of the signal (1'b1 or 1'b0)
 - In C/C++ Domain 'X' is seen as 1'b1 & 'Z' is seen as 1'b0

Verification guidelines

```
// *** Wrong coding ***//
```

```
void Monitor() {  
    if(RTL_BIST_Complete == 1) {  
        ...  
        // checks//  
    }  
}
```

```
// *** Correct coding ***//
```

```
void Monitor() {  
    if (RTL_BIST_Complete == 1 && RTL_BIST_Complete.IsXorZ()  
        != 1) {  
        ...  
        // checks//  
    }  
}
```

Verification guidelines

- Test plan is to check if it is not 'X' or 'Z'

```
// *** Correct coding ***//  
void Monitor() {  
  if (IntState == SLEEP) {  
    if(RTL_DramReset.IsXorZ() == 1 || RTL_DramReset == 1)  
    {  
      Error (“Un-expected X/Z”);  
    } else {}  
  }  
}
```

References

- Diagrams on “Power reduction techniques”
 - Synopsys Low-Power Flow User Guide (Version A-2007.12, December 2007)

Good Bye

THANKS FOR YOUR TIME