



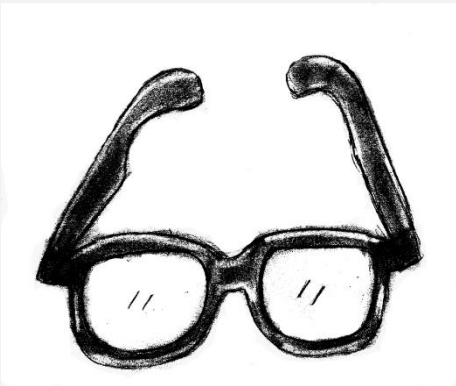
VANIKA HANS INTERNSHIP EXPERIENCE

Mentor: Stefanie (Quiyi) Wang

Manager: Chris Smithhisler

Team: MPP/Xprop Validation

AGENDA

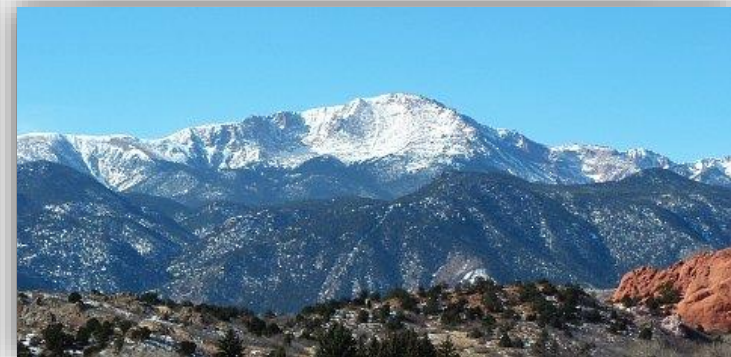
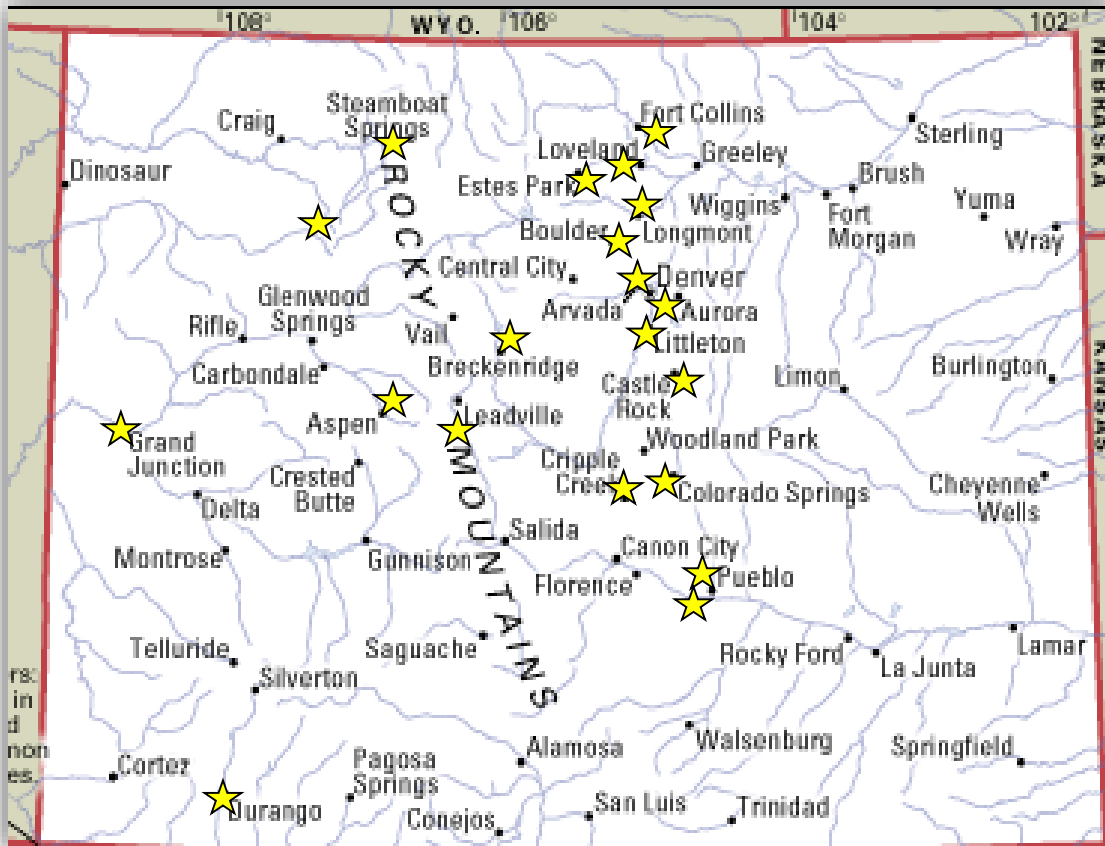


- Introduction
- MPP Validation
- X-Propagation
- Experiences at Intel
- Outside of Work
- Special Thanks



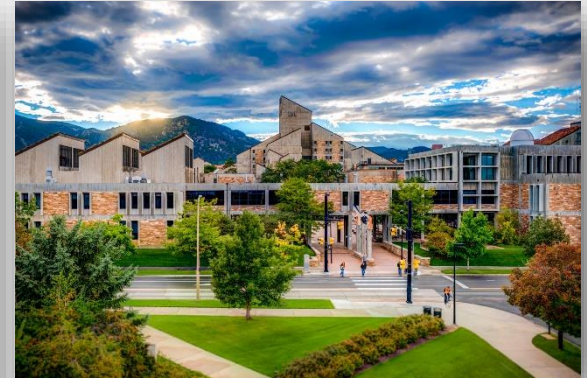
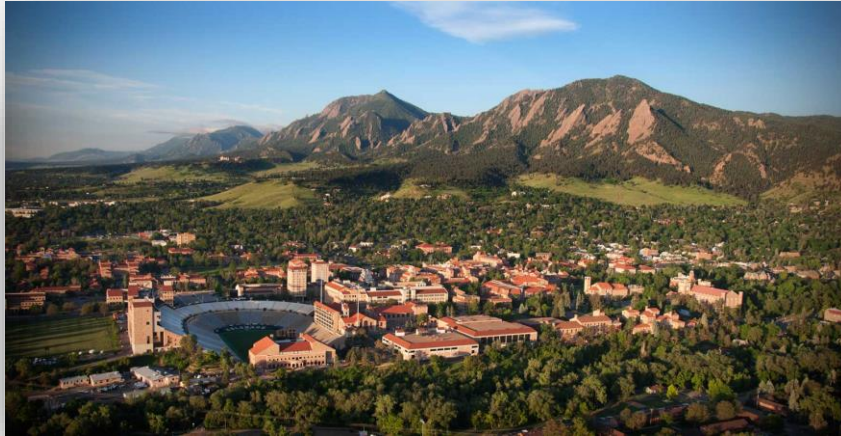
ABOUT ME!

- Born in New Delhi, India
- Grew up in Colorado Springs
- First time in Fort Collins!



ABOUT ME!

- School: University of Colorado Boulder
- Electrical and Computer Engineering, B.S.
- Minor in Computer Science, Leadership Certificate



DANCE LIFE



SOCIETY OF WOMEN ENGINEERS



- **2015-2016:** Director of K-12 Outreach
- **2016-2017:** Society Representative
- **2017-2018:** Vice President
- HUGE part of college career
- Conferences in Boulder, Nashville, Philadelphia, Oklahoma City, and Austin!

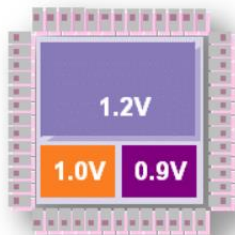




PROJECT #1 – MPP ISOLATION COVERAGE

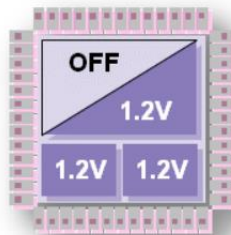
BACKGROUND

- UPF: Unified Power Format
 - Allows for advanced Low Power Techniques
 - IEEE standard language
 - Extension to RTL Design
 - Provides consistent power behavior interpretation

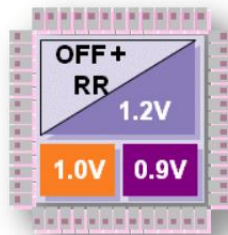


Multi-Voltage (MV)

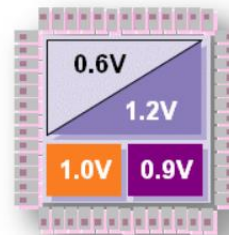
©Synopsys 2013



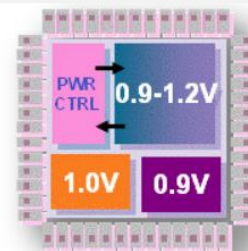
MTCMOS Pwr Gating
(Shutdown)



MV & Pwr Gating w/
State Retention



Low-VDD Standby



Dynamic or Adaptive
Voltage & Frequency Scaling
(DVS, DVFS, AVS, AVFS)

Functional Intent vs. Power Intent

What is the difference?

Functional intent specifies

- **Architecture**
 - Design hierarchy
 - Data path
 - Custom blocks
- **Application**
 - State machines
 - Combinatorial logic
 - I/Os
 - ex: CPU, DSP, Cache
- **Usage of IP**
 - Industry-standard interfaces
 - Memories
 - etc

Captured in RTL

Power intent specifies

- **Power distribution architecture**
 - Power domains
 - Supply rails
 - Shutdown control
- **Power strategy**
 - Power state tables
 - Operating voltages
- **Usage of special cells**
 - Isolation cells
 - Level shifters
 - Power switches
 - Retention registers

Captured in UPF

DETAILED EXAMPLES OF NEED

- **Power Gating**
 - Switches off inactive blocks
 - **Advantage:** Significantly reduces leakage power
- **Multi-Voltage**
 - Allows different blocks to operate at different voltages
 - **Advantage:** Significantly reduces dynamic and leakage power when high performance is not needed
- **Memory Retention**
 - during times of limited power, can retain and restore info
- **Level Shifters**
 - Allows different voltage power domains
- **Power Switches**
 - Turn PDs on and off

TERMINOLOGY

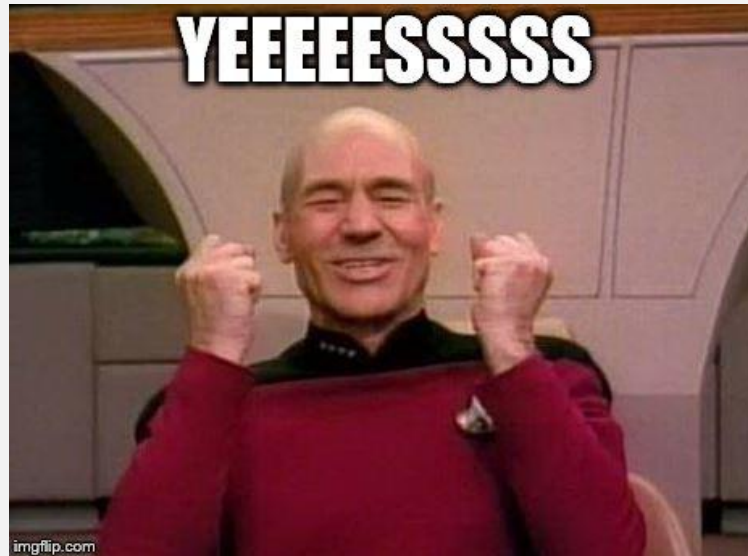
- **VCS**: Verilog Compiler Simulator
 - Advanced Simulator Techniques
- **“Power aware”**
 - UPF enabled
 - Some logic running at lower voltage, some logic turned off, protections (isolation) in between
 - **VCS+NLP**: VCS+Native Low Power
 - Power aware simulator
 - Can verify RTL and advanced voltage techniques

MPP Validation:

Includes everything involving UPF (power intent)!

MPP = Multiple Power Plane

WE NOW HAVE A POWER
AWARE SIMULATOR (VCS+NLP)
THAT ALLOWS ADVANCED LOW
POWER TECHNIQUES.

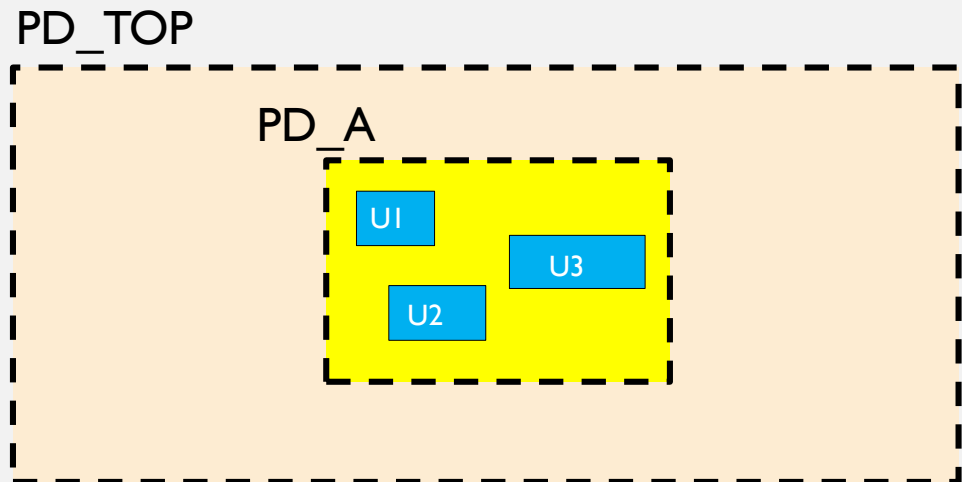


TERMINOLOGY (CONTINUED)

- **Power Domain:** Sub-region of design. Need to define regions that have different power characteristics than other parts of design.

```
create_power_domain PD_TOP
```

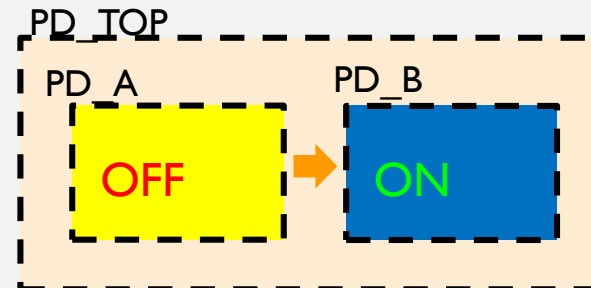
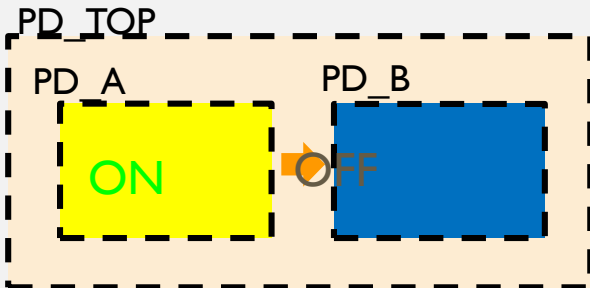
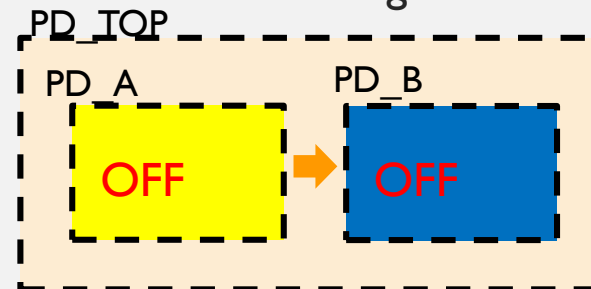
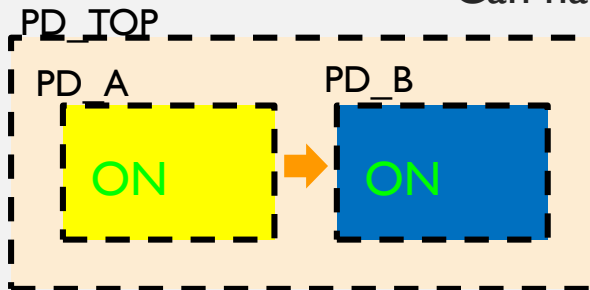
```
create_power_domain PD_A  
-elements {U1 U2 U3}
```



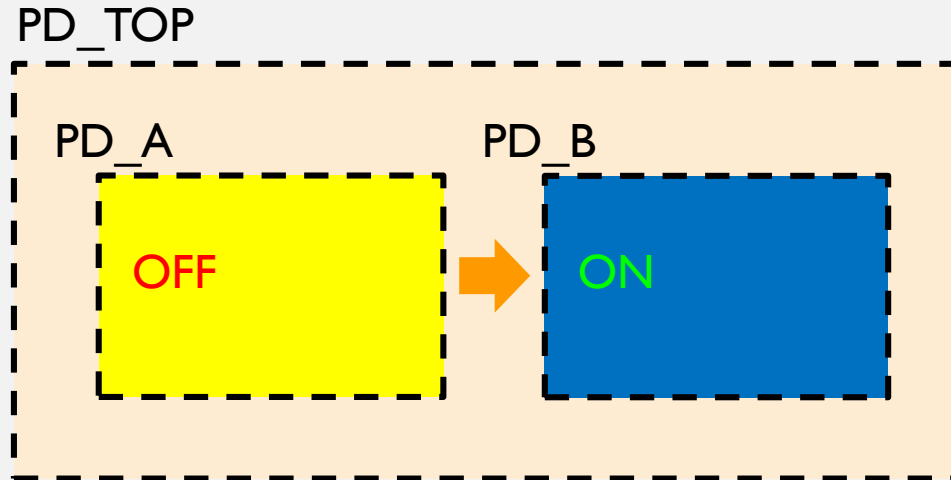
- Elements in PD_A are now separate from PD_TOP
- Can have different power characteristics

POWER DOMAINS

- Can have multiple power domains in design

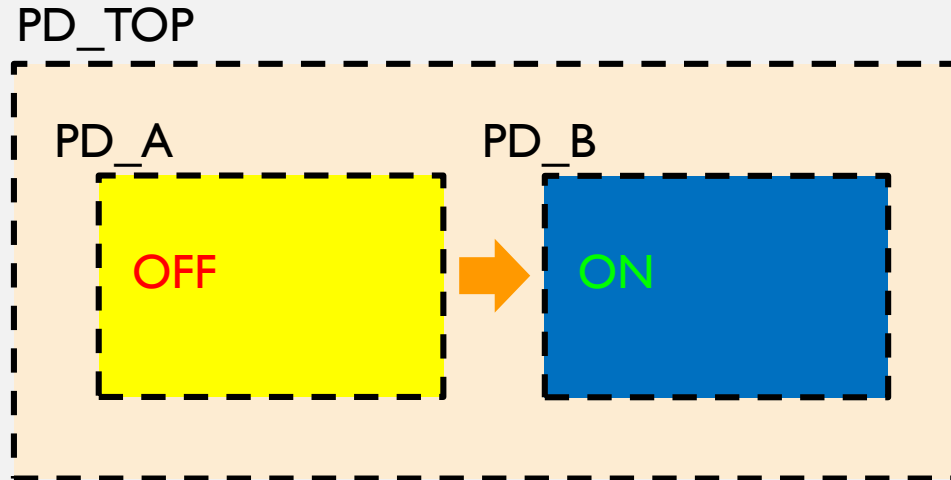


WHAT'S THE ISSUE?



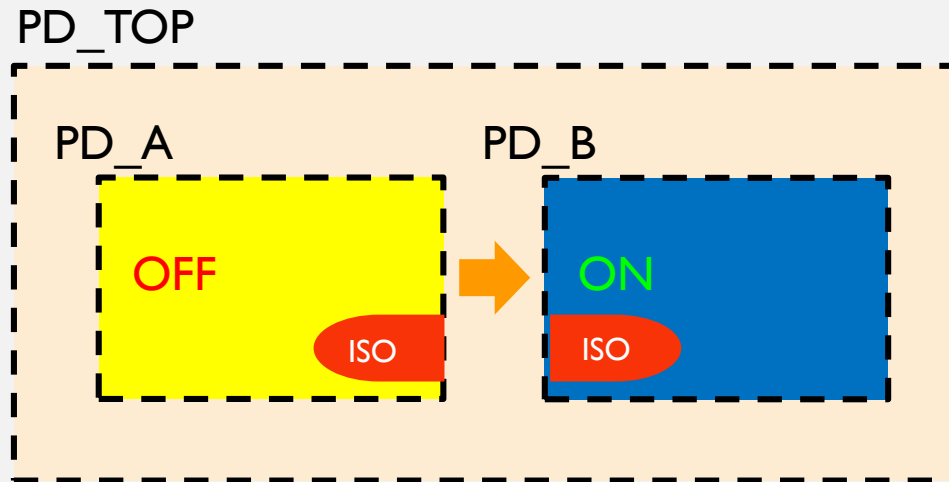
- Outputs from shutdown elements (PD_A) connected to active logic (PD_B) causes issues:
 - In documentation: “Spurious signal propagation and crow bar current”
 - In reality: Unpredictable outputs, i.e. “X” outputs.

WHAT'S THE ISSUE?



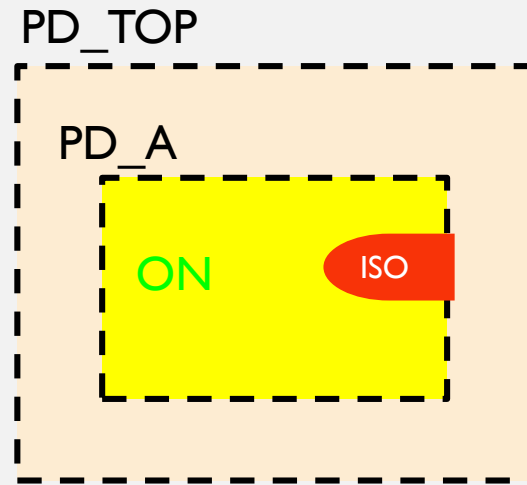
- Need an element to “protect” logic
 - Ensure that an OFF PD doesn't have any extra outputs
 - No leakage paths from powered down logic

ISOLATION



- PD_B is protected when either:
 - Isolation on input of PD_B is enabled
 - Isolation on output of PD_A is enabled

OPTIONS FOR ISOLATION: CAN SPECIFY IN UPF

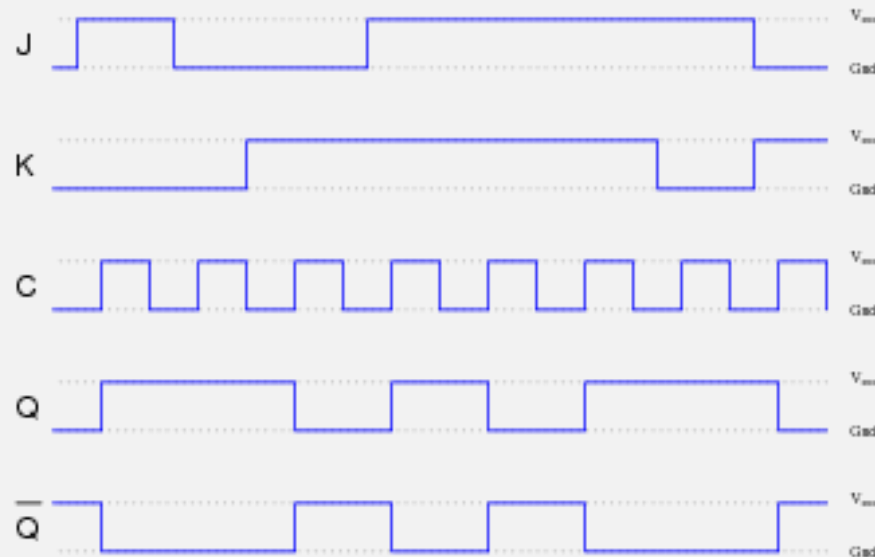


- What's powering isolation strategy?
- What controls isolation?
- Is it active high or low?
- What value should we clamp to when isolation occurs?
- Where should it occur? Input or output?
- Is it inside or outside PD?

ON DVE:

I was given all the partitions in CDF which have isolation strategies.






- Each partition has anywhere from 1 to 53 strategies
- Was to validate these strategies relative to behavior of PD








COVERAGE GROUPS

Low Coverage:

25%






	Status	Bin Name	At Least	Size	Hit Count
		ACTIVE_LEVEL	1	1	1
		INACTIVE_LEVEL	1	1	0
		ACTIVE_TO_INACTIVE	1	1	0
		INACTIVE_TO_ACTIVE	1	1	0

75%

	Status	Bin Name	At Least	Size	Hit Count
		INACTIVE_LEVEL	1	1	1
		ACTIVE_LEVEL	1	1	1
		ACTIVE_TO_INACTIVE	1	1	1
		INACTIVE_TO_ACTIVE	1	1	0




High
Coverage:

100%

	Status	Bin Name	At Least	Size	Hit Count
		INACTIVE_LEVEL	1	1	2
		ACTIVE_LEVEL	1	1	2
		ACTIVE_TO_INACTIVE	1	1	2
		INACTIVE_TO_ACTIVE	1	1	1

COVERAGE GROUPS

- Contain different active/inactive levels

Behavior	Isolation Control Signal
0 or 1	 25%
Toggle	 75%
2 Toggles	 100%

GOAL: 100% COVERAGE

100

- Process is to keep adding on tests to improve coverage
 - We want test cases that will bring us here
- Spyglass catches issues after 100% coverage
 - Get isolation coverage to this point, rest will be spyglass errors

100



25% EXAMPLE

- Isolation Strategy:

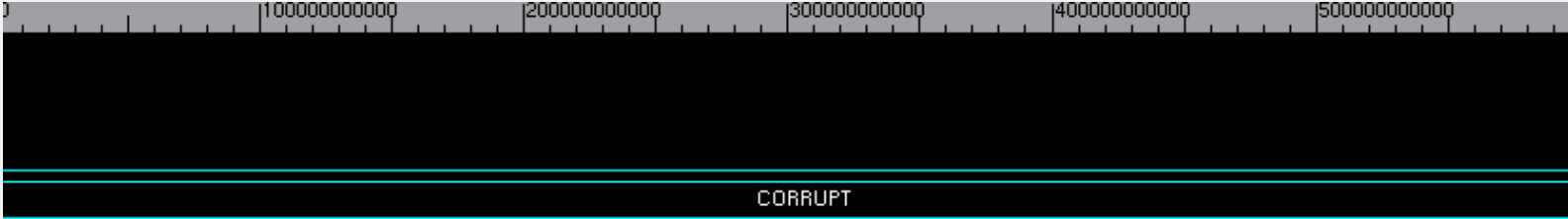
Iso_example

- From UPF code, we learned:

✖	Status	Bin Name	At Least	Size	Hit Count
	✓	ACTIVE_LEVEL	1	1	1
	✗	INACTIVE_LEVEL	1	1	0

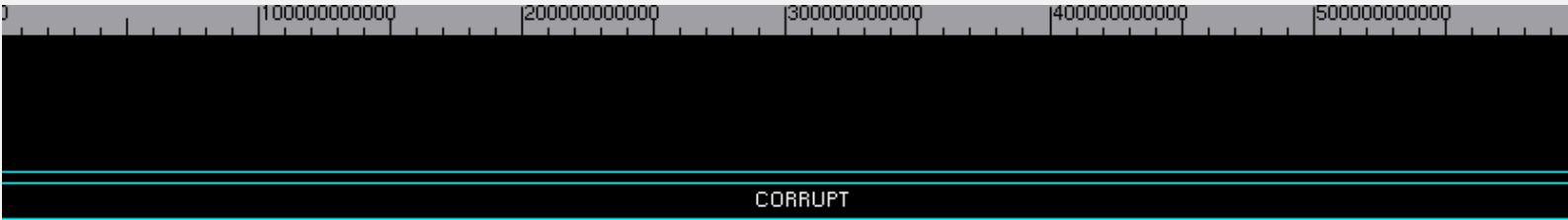
Iso Signal	PD	Active	Location	Control Signal	Clamp:	I/O
Iso_example	PD_Example	low	self	ISO_ctrl	0	output

ISO_ctrl
PD_Example

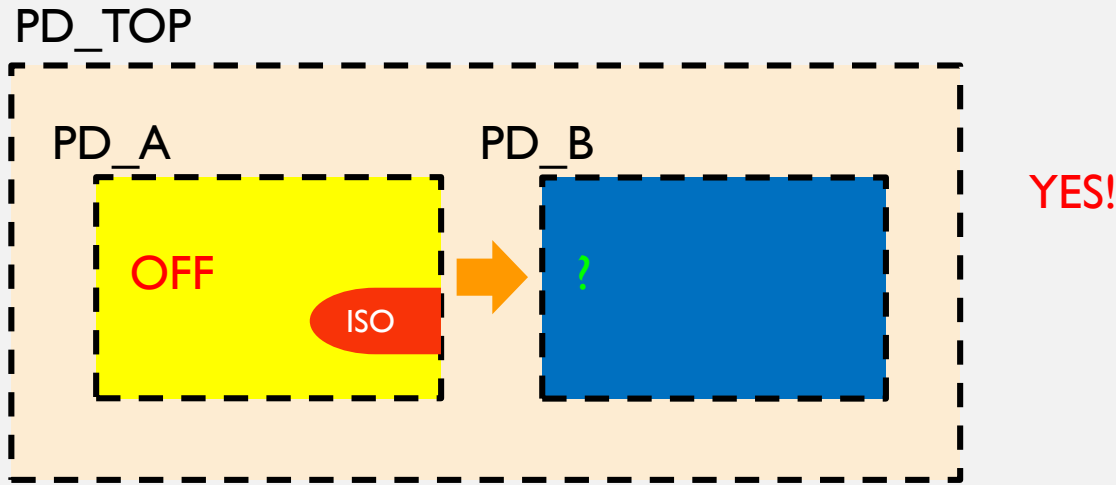


Isolation control signal is always enabled, as PD is always corrupt. Is this expected?

Iso Signal	PD	Active	Location	Control Signal	Clamp:	I/O
Iso_example	PD_Example	low	self	ISO_ctrl	0	output



Isolation control signal is always enabled, as PD is always corrupt. Is this expected?



MY JOB SUMMARIZED:

- Validate isolation strategies implemented in UPF by pulling waveforms on DVE
- Critically analyze strategies, and pessimistically question whether what we see is “expected”
- Communicate with IP Owner when things aren’t expected
- Implement more tests to get to 100% coverage

PROJECT #2 – X PROPAGATION

BACKGROUND

- Synopsys Tool
- Addition to RTL simulations
- VCS (can think of as Verilog and VHDL) doesn't accurately model uninitialized registers or power-on reset values
 - Overly optimistic
 - Xprop attacks ambiguity of VCS
- X's used to represent unknown values of elements

A WORLD WITHOUT XPROP

- Errors in connection signals
 - Verilog optimism is not reliable
- Most relevant/valued when using NLP (power-aware) models, especially cold-boot /cold-reset
- Sanity Measure
- Historically has solved issues that were found through other means
 - *Faster, more efficient way*

EXAMPLE OF VCS OPTIMISM VS. VCS+XPROP

- **Verilog semantics state:**
 - If the condition of an if-else statement is X, it will be interpreted as false

Let “a” be an X-valued control signal.

VCS Simulation (Verilog)

```
if (a)
  b = 0; Is a ambiguous(X)?
else
  b = 1;
  Yes. So b=1
```

X-prop Simulation

```
if (a)
  b = 0;
else
  b = 1;
```


EXAMPLE OF VCS OPTIMISM VS. VCS+XPROP

- **When xprop is enabled:**
 - Any ambiguity in condition results in output of X

Let “a” be an X-valued control signal.

VCS Simulation (Verilog)

```
if (a)
  b = 0;
else
  b = 1;
```

X-prop Simulation

```
if (a)
  b = 0;
else
  b = 1;
```

Is a ambiguous(X)?
Yes. So b=X

EXAMPLE OF VCS VS. VCS+XPROP

This might be important when:

- *Let “a” represent a decoder signal fetched from memory*
- *Memory is X starting out (in simulation and in real silicon)*
- *When $a=X$ the else condition is forced, firing an interrupt signal*

Let “a” be an X-valued control signal.

VCS Simulation (Verilog)

```
if (a)  Is a ambiguous(X)?  
    b = 0;  
else    Yes. So b=1  
    b = 1;
```

X-prop Simulation

```
if (a)  Is a ambiguous(X)?  
    b = 0;  
else    Yes. So b=X  
    b = 1;
```

SECOND EXAMPLE:

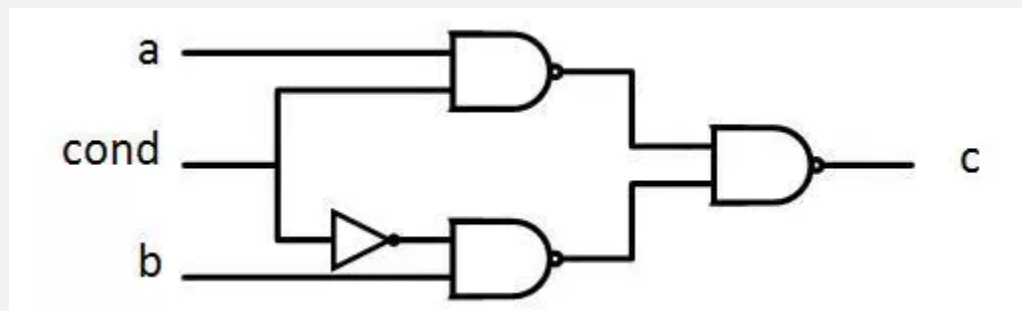
- The following are all valid transitions in Verilog and SystemVerilog that will trigger a posedge operator
 - 0 -> 1
 - 0 -> X
 - 0 -> Z
 - X -> 1
 - Z -> 1
- In simulation the simulator will behave the same for all of these

REALITY:

- In real silicon the X's are very real, as on startup values could be anything
- Traditional VCS will not recognize these subtle differences
- But can be catastrophic later on

TMERGE SCHEME

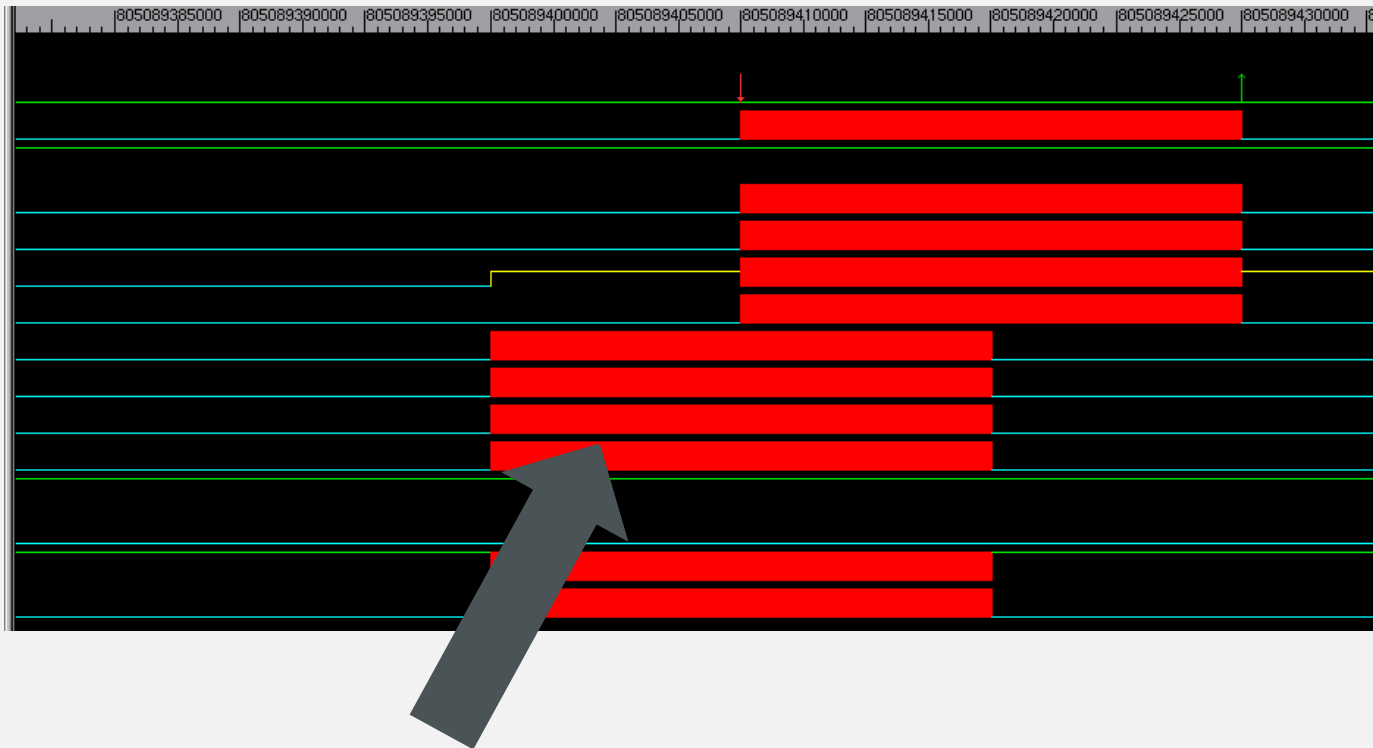
- Different schemes by which this works
 - 1) **V-merge**: classic Verilog optimism (**overly optimistic**)
 - 2) **X-merge**: Any ambiguity in control signals outputs X (**overly pessimistic**)
 - 3) **T-merge**: Traverses both code paths with 0 and 1; if there is a difference between outputs, X will be propagated



MY JOB

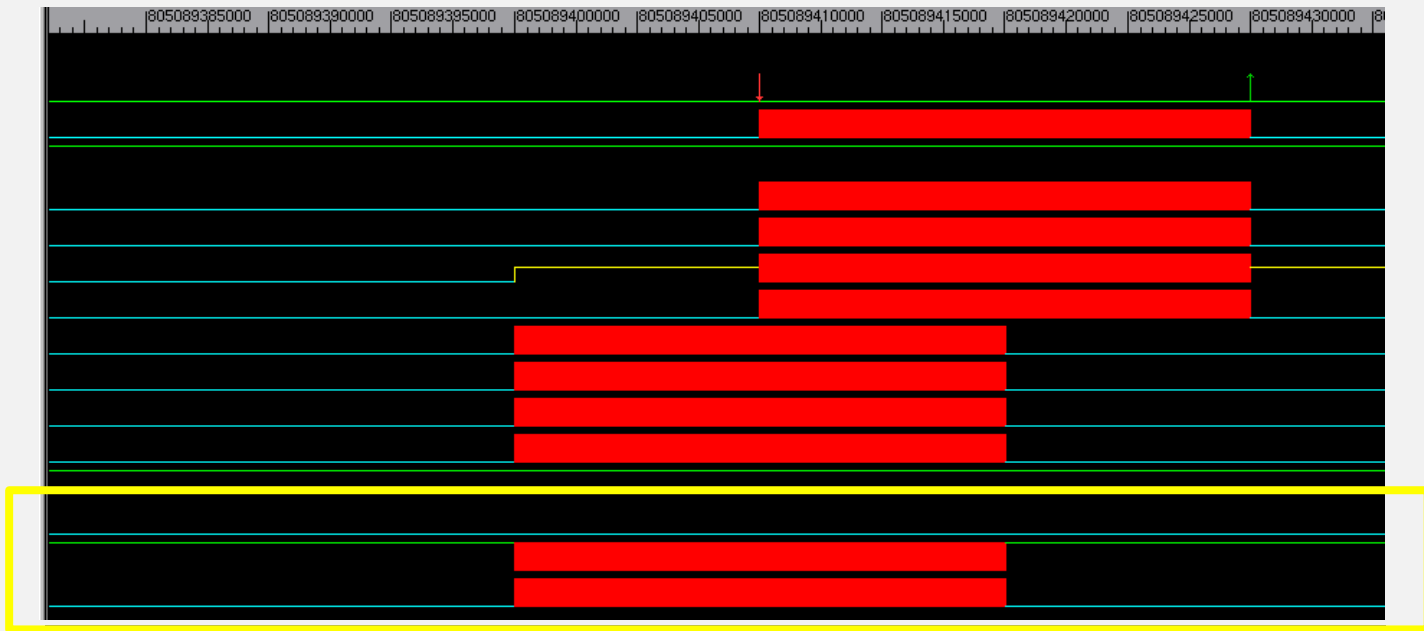
- Given an assertion failures error log:
 - Compare errors in Xprop vs. Non-Xprop
 - Find an assertion failure unique to Xprop
 - Go to DVE and find this assertion failure
 - Trace drivers back to find root cause

MY JOB (CONT)



Red means 'X' propagated

Xprop



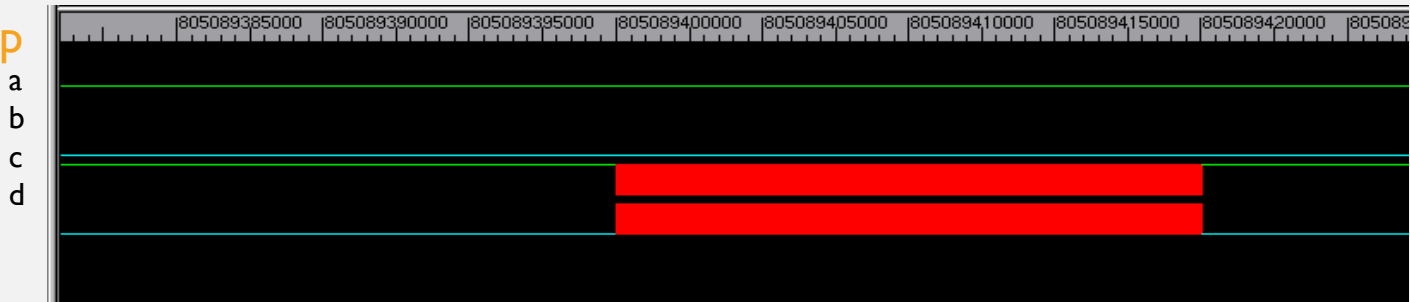
Non-Xprop



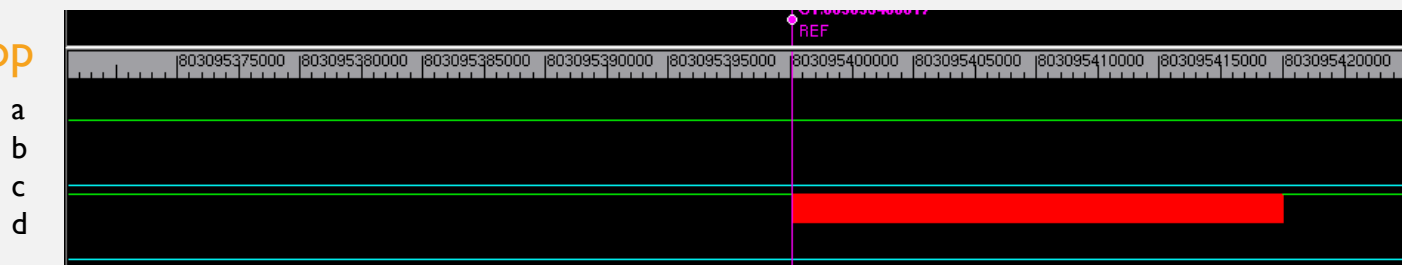
MY JOB (CONT)

- Important signals obtained from previous slide

Xprop



Non-Xprop



Which tells us that something in a, b, or c causes d to propagate an X.

MY JOB (CONT)

- Find module of interest that utilizes this signal.
- In this case might be something like:

```
module example(a,b,c,d);  
  input a,b,c;  
  output d;  
  always @ (a or b or c)  
  begin:  
    if (c)  
      d = a;  
    else  
      d = b;  
    end  
  end module
```

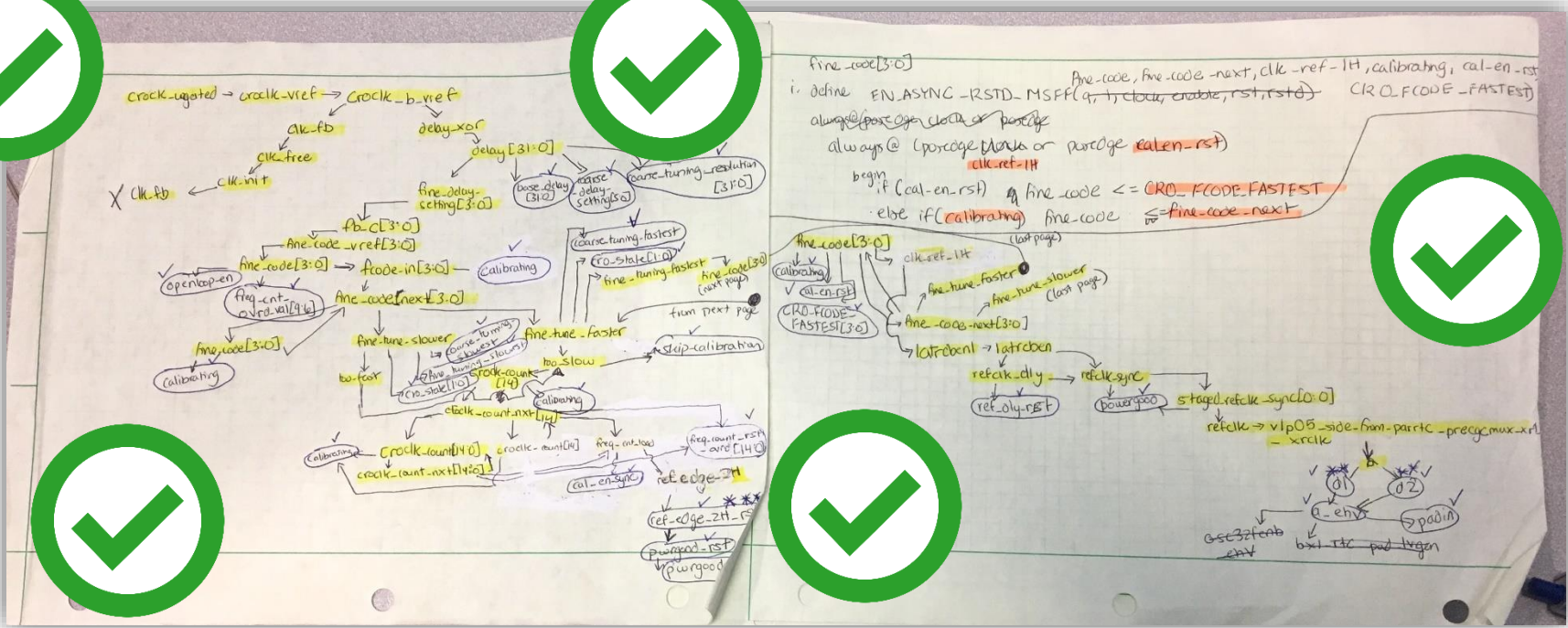
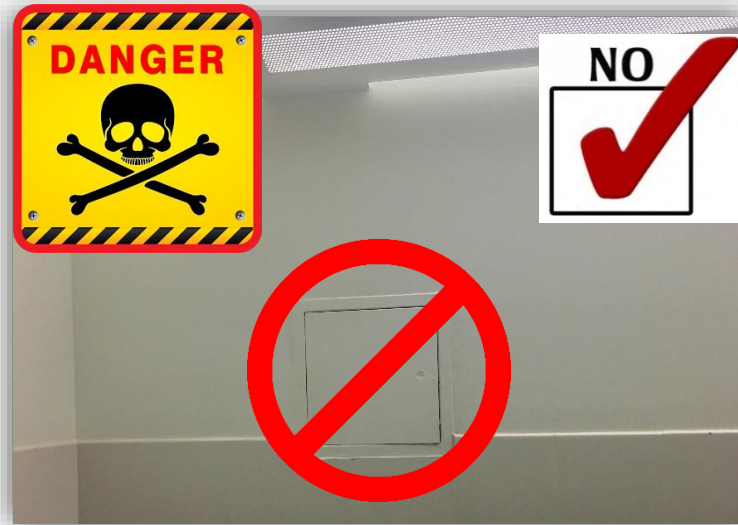
SUMMARY OF MY JOB:

- Implement a workaround to remove this module's X-prop and see if this removes assertion failure
 - Given that module is "example"
 - Add: module {example}{xpropOff}; to configuration file
- Temporary fix for progression
- THEN: Check workaround to see if error is gone
- If so, let other people know so they can determine the issue!

OVER THE SUMMER, I LEARNED:

- UPF
- Low Power Techniques
- UNIX
- VIM
- DVE
- Verdi
- Assertions in SystemVerilog
- Verilog Optimizations
- Architecture
- Validation – pre and post
- Intel Test Flows

- Curiosity is definitely my strongest suit
- Ask questions effectively
- That I learn better by discussing
- Debugging and critical thinking skills
- Networking skills



EXPERIENCES AT INTEL

Events:

- Galactic Girls Outreach!
- NextGen
- Community Day at Rocky Mountain Raptor Program Service
- WIN Q2 Networking Event
- Annual Picnic
- FCDC Lab Walkthrough

EXPERIENCES AT INTEL

Meetings:

- Intern Series by Career Zone
- Open Forum, All Hands Meeting
- EXOS counseling sessions w/ Campy
- Weekly Spanish Time in Cafeteria

- Most if not all free popcorn and ice cream events
- Met many people from different backgrounds!

OUTSIDE OF WORK

- Here are some of the adventures I had around Fort Collins this summer!
- <https://quik.gopro.com/v/y7hheOSZuy/>



SPECIAL THANKS

- **Mentors:**

- Stefanie Wang
- Ben Ruter

- **Managers:**

- Chris Smithhisler
- Eduardo Iturbe

- **For taking the time to meet with me:**

- Patrick Mahoney
- Eric Fernandez
- Keerthi Ravella
- Stephanie Wang
- Tamara Wesley
- Sharvani Mukkula
- Sarah Kay
- Keri Sibley

- **Intel Events/Outside of Work:**

- Jerri Van Zanten
- Campy
- WIN
- Kristina Santiago
- Chirag Patil
- Kevin Virgen
- FCDC Interns!

- **Advisors on Career Connections:**

- Joe Huber (Hudson)
- Michael S Bair (Hillsboro)

- **Career Zone**

- Daman Oberoi for ***Tips and Tricks for Navigating Intel, How to Talk to your Manager, Formel Intern Panel: Key Lessons Learned***

WHAT'S NEXT?

- Senior Year!!!!
- This semester:
 - Classes related to Computer Architecture / Val (ish)
 - Senior Design Project: Smart Blind Stick for the Visually Impaired
- Hoping to learn more about:
 - Careers in Val
 - RTL code
 - Python
 - SystemVerilog
 - OVM/UVM
 - Computer architecture/microarchitecture in general

ANY QUESTIONS?

FURTHER INFO:

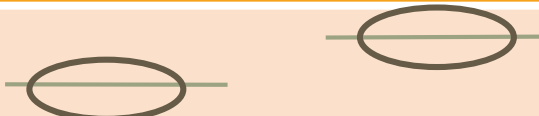
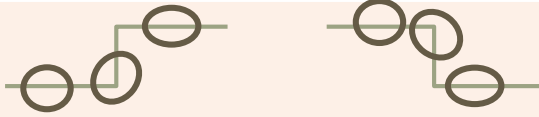

DETAILED EXAMPLES OF NEED

- **Power Gating**
 - Switches off inactive blocks
 - Mostly used when parts of design are dormant for long periods of time
 - **Advantage:** Significantly reduces leakage power
- **Multi-Voltage**
 - Allows different blocks to operate at different voltages
 - **Advantage:** Significantly reduces dynamic and leakage power
 - High performance levels only needed for timing critical components. Useful when max performance levels unnecessary
- **Memory Retention** during times of limited power
- **Level Shifters, Power Switches**

COVERAGE EXPLANATIONS

- Can break down validation of isolation strategies according to percent coverage

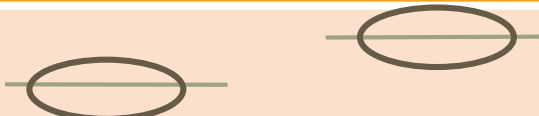
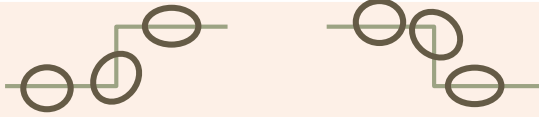

Single State
= 25%

Behavior	Isolation Control Signal
0 or 1	 25%
Toggle	 75%
2 Toggles	 100%

COVERAGE EXPLANATIONS

- Can break down validation of isolation strategies according to percent coverage

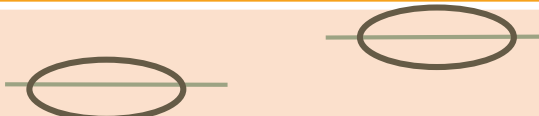
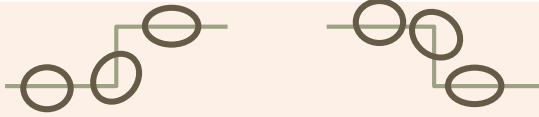

Starting state + transition
state + ending state =
 $25+25+25 = 75\%$

Behavior	Isolation Control Signal
0 or 1	 25%
Toggle	 75%
2 Toggles	 100%

COVERAGE EXPLANATIONS

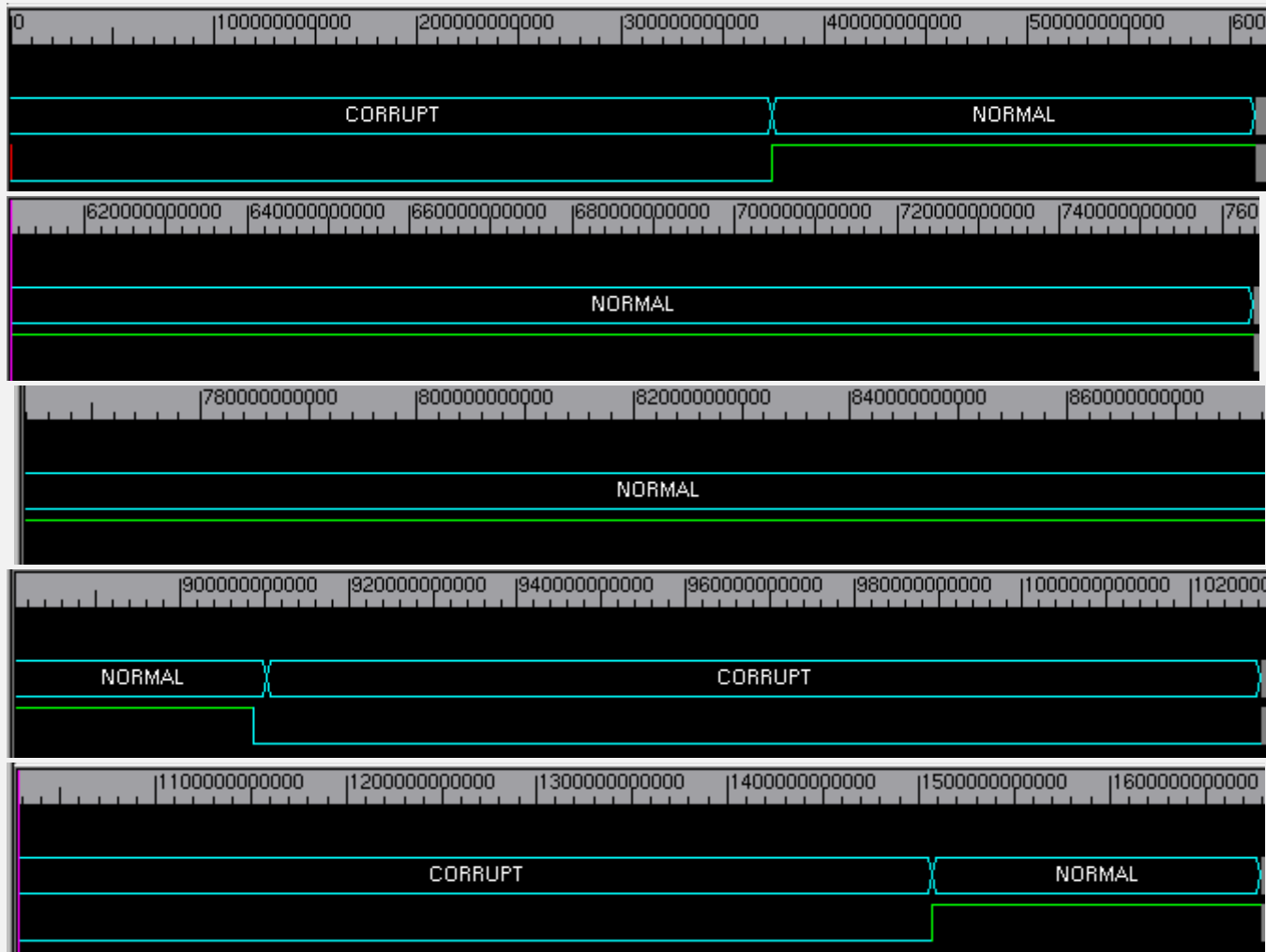
- Can break down validation of isolation strategies according to percent coverage

Starting state + transition
state + ending state +
transition state =
 $25+25+25+25 = 100\%$

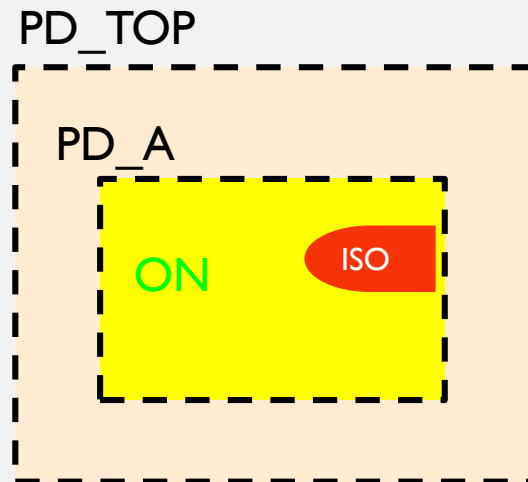
Behavior	Isolation Control Signal
0 or 1	 25%
Toggle	 75%
2 Toggles	 100%

100% EXAMPLE

✖	Status ▾	Bin Name	At Least	Size	Hit Count
	✓	ACTIVE_LEVEL	1	1	2
	✓	INACTIVE_LEVEL	1	1	2



OPTIONS FOR ISOLATION



- What's powering isolation strategy?
- What controls isolation?
- Is it active high or low?
- What value should we clamp to when isolation occurs?
- Where should it occur? Input or output?
- Is it inside or outside PD?

ISOLATION UPF CODE

set_isolation <isolation_strategy_name>

-domain <power_domain>

-isolation_power_net <isolation_power_net>

-isolation_ground_net <isolation_ground_net>

[-clamp_value 0 | 1 | latch]

[-no_isolation]

[-applies_to inputs | outputs | both]

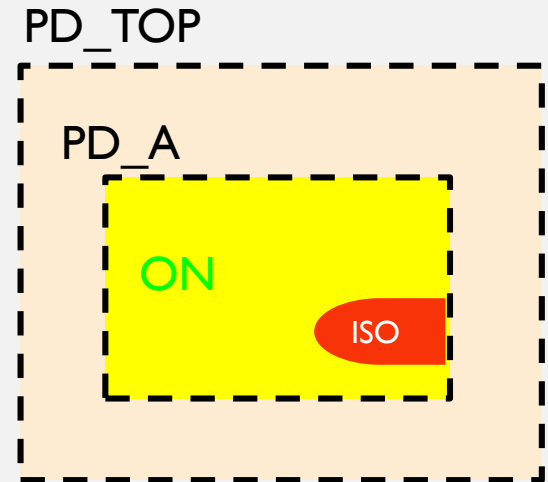
set_isolation_control <isolation_strategy_name>

-domain <power_domain>

-isolation_signal <isolation_signal>

[-isolation_sense low | high]

[-location self | parent]



EXAMPLE OF ISOLATION STRATEGY

set_isolation ISO_EXAMPLE

-domain PD_A

-isolation_power_net VDD

-isolation_ground_net VSS

-clamp_value 0

-applies_to outputs

set_isolation_control ISO_EXAMPLE

-domain PD_A

-isolation_signal ISO_CTRL

-isolation_sense low

-location self

