# Managing Highly Configurable Design and Verification

Jeremy Ridgeway
Broadcom, Inc.

October 23, 2018
Austin

# Agenda

Root of the problem

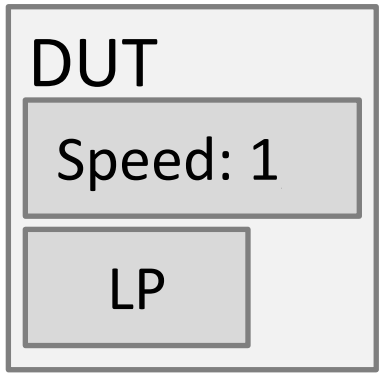
What I’m selling today to fix it

Conclusions

# PCIe Express Subsystem

- Composed of multiple IP deliveries

- Each IP includes specific capabilities that are customizable
  - Specify a subset of capability
  - Exclude capability
  - Dependency is OK
  - Mutual exclusivity is OK

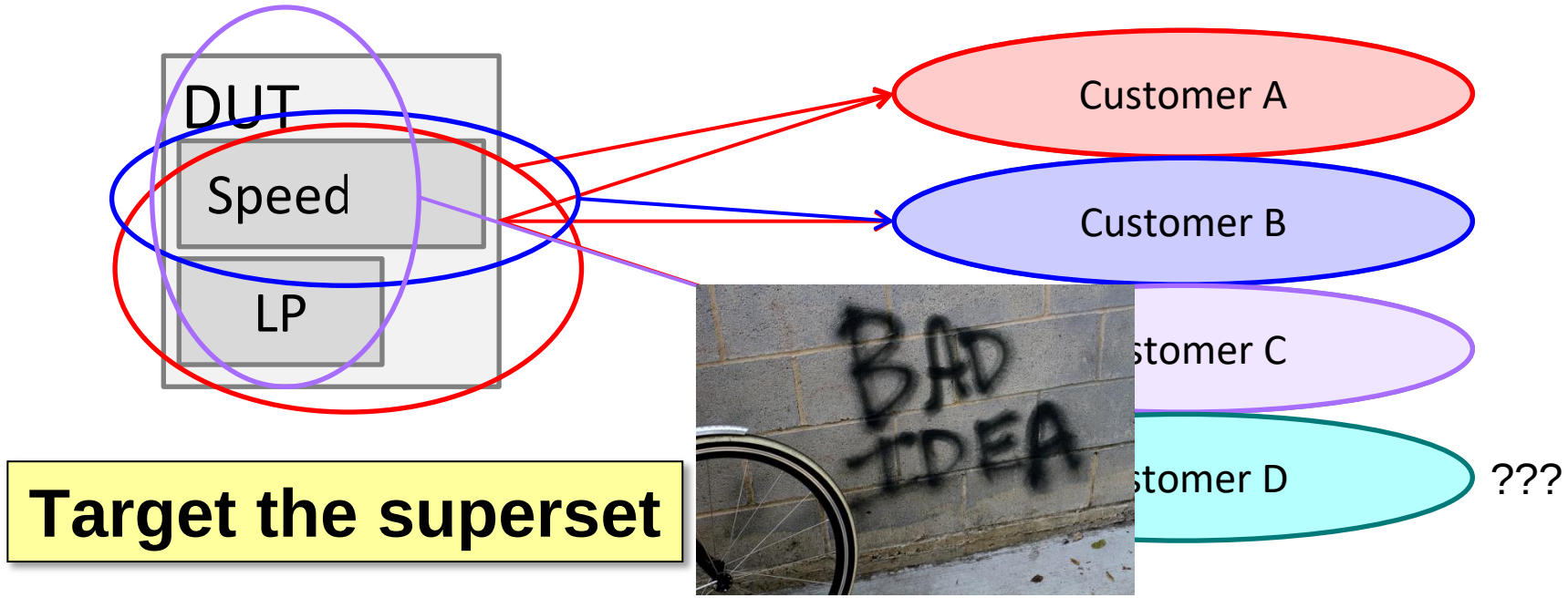



**Pretty Darn Complex**

**LP = Low Power**

# Hardware Configurations

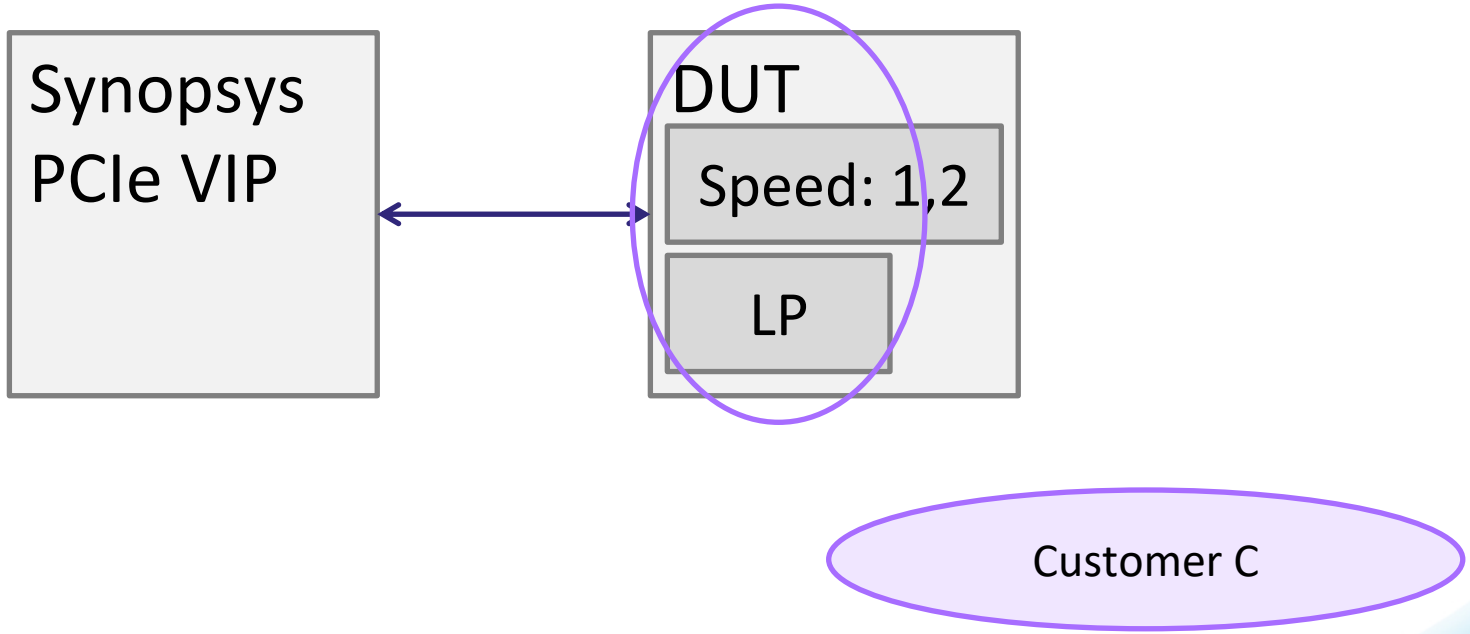
- One RTL set to rule multiple customers



# Superset Testbench

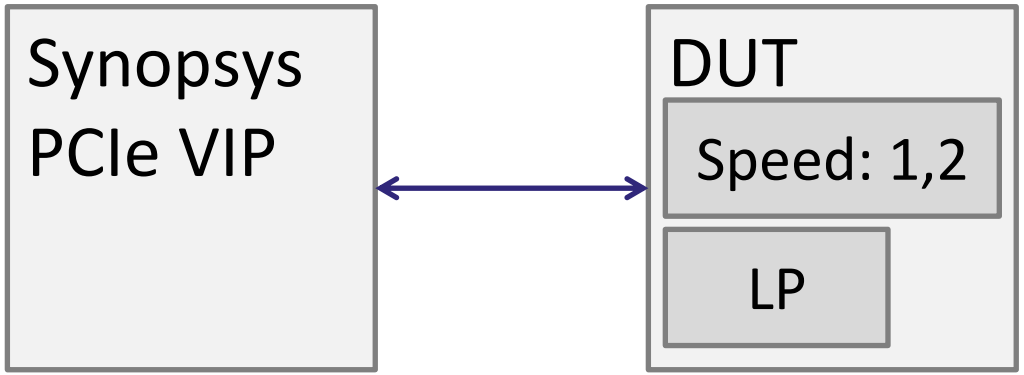
- *"Yeah we've covered that because it is a subset"*
  - It's possible to miss a mode or feature
  - Was the feature *actually* covered?
  - Show me the feature covered *during* the customer-specific mode-of-operation

**Functional coverage!**

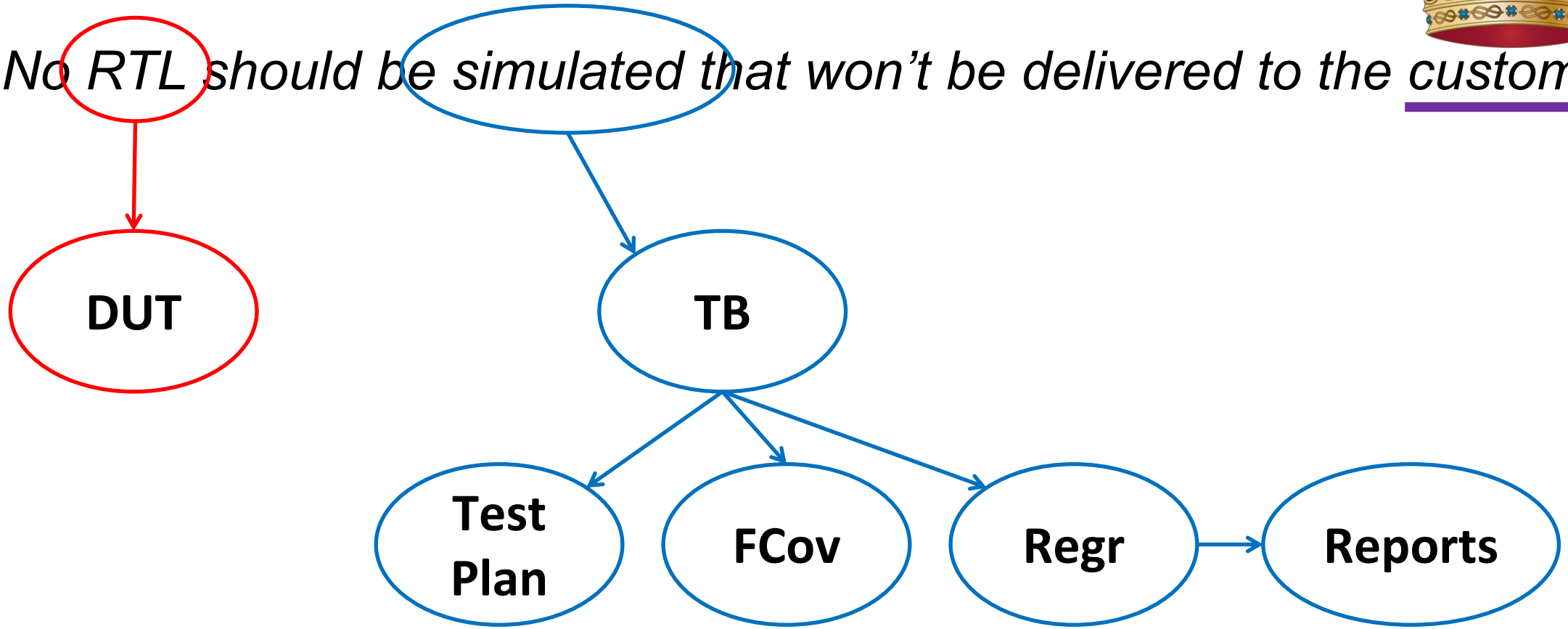
# Superset Testbench

- Strict superset may not be possible
  - Capability mutual exclusivity is OK
- Could be missing unknown customer-specific scenarios

# Mantra

*No RTL should be simulated that won't be delivered to the customer*



# Agenda

Root of the problem

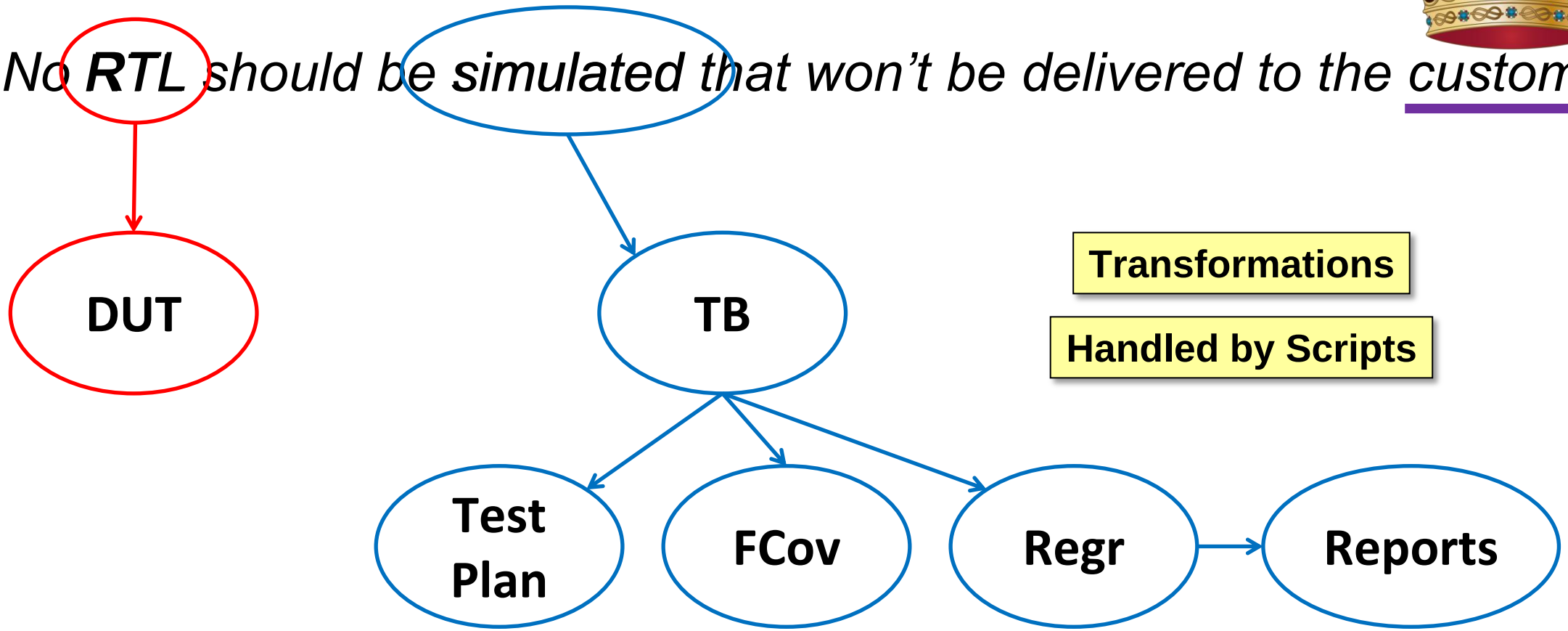
What I’m selling today to fix it

    Cohesive scripting ecosystem

Conclusions

# Ecosystem

*No **RTL** should be **simulated** that won't be delivered to the customer*



# Configuration Object

- Glue binding the ecosystem together
- Must be something that can be reused in all tranformations

- We chose Perl5
  - Perl Packages support inheritance and polymorphism natively
  - General understanding of Perl in Engineering
  - Broad support for Perl

- Any OOP programming language would suffice

# Configuration Matrix and Object

BROADCOM

snug Synopsys Users Group Austin

| Configuration | Values | Customer Selection |
|---|---|---|
| Speed | 1, 2 | ? |
| Low Power -- L0s | Y / N | ? |
| … | | |

**CUST_A.pm**

```
SPEED => [ 1, 2 ]
LP => 1
…
```

**CUST_B.pm**

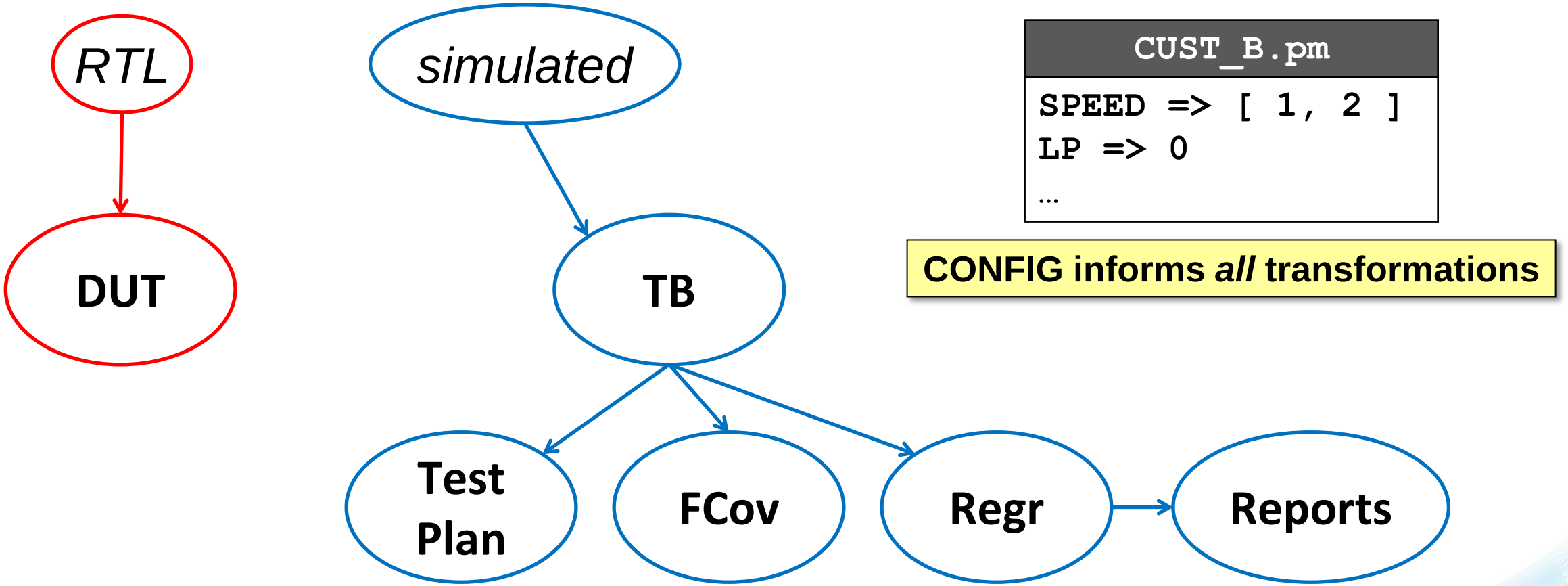
```
SPEED => [ 1, 2 ]
LP => 0
…
```

**CUST_C.pm**

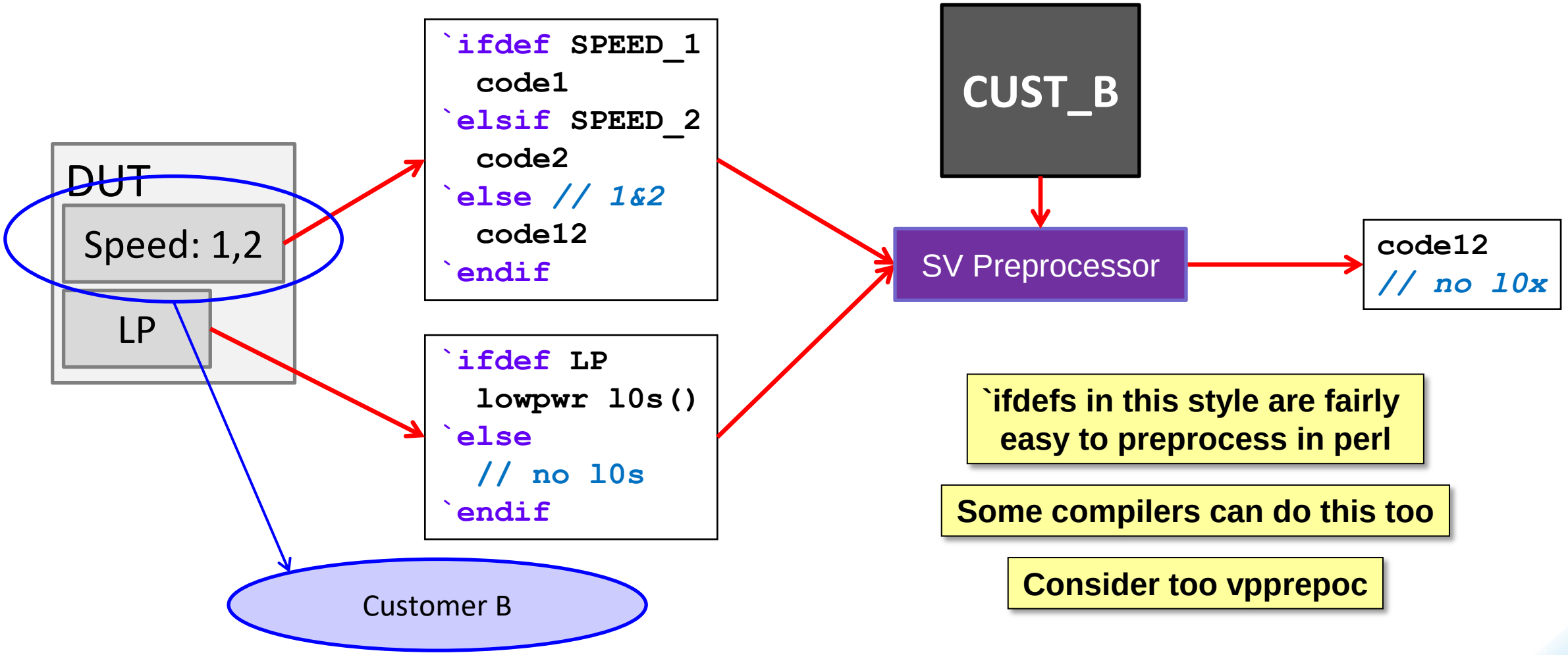
```
SPEED => [ 1 ]
LP => 1
…
```

- Encode the customer selection in a CONFIG.pm file

# Ecosystem


RTL
DUT
simulated
TB
Test Plan
FCov
Regr
Reports
CUST_B.pm
SPEED => [ 1, 2 ]
LP => 0
…
CONFIG informs *all* transformations

# DUT Transformation



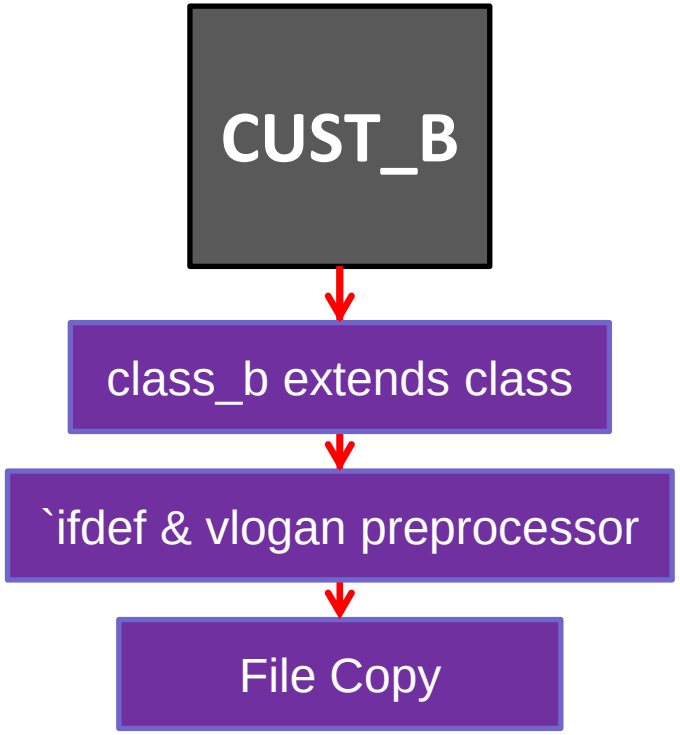

**`ifdefs in this style are fairly easy to preprocess in perl**

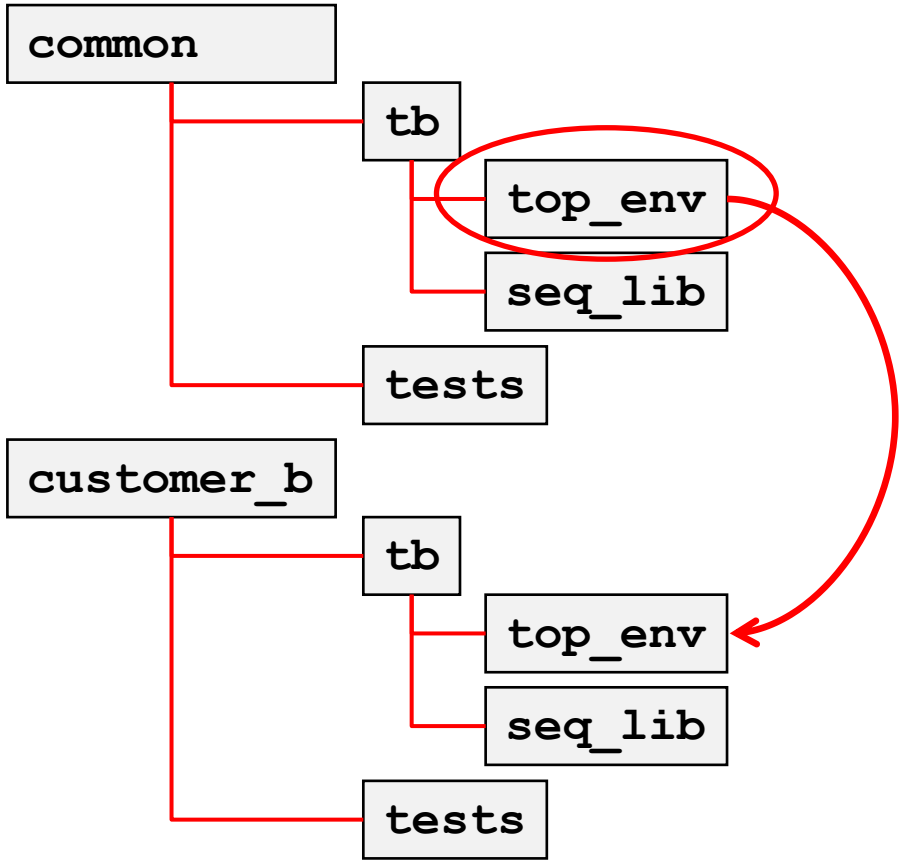
**Some compilers can do this too**

**Consider too vpprepoc**

# Testbench Transformation

- Common components / objects
  - Inherit to customer-specific
  - Copy to customer-specific



# Agenda

- Root of the problem
- What I'm selling today to fix it
  - Cohesive scripting ecosystem
    - Testplan
    - Functional Coverage
    - Reporting Structure
- Conclusions

# Testplan: A Live Document

BROADCOM

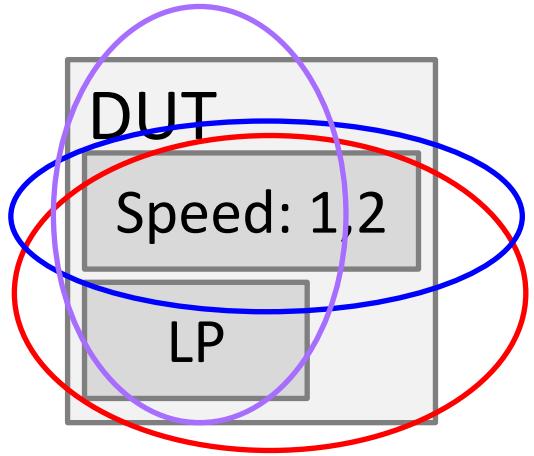
snug
Synopsys Users Group
Austin

| Scenario ID | Feature | Subfeature | Generate | Expect |
|---|---|---|---|---|
| LP.DATA.1 | LP Good Enter | Normal | …stimulus… | …verification… |
| ERR.LP.1 | LP Error Enter | Unexpected | … | … |
| SPEED.MODE_1.1 | Speed Gen1 | Autoselect | … | … |
| SPEED.MODE_1.2 | Speed Gen1 | FW Override | … | … |
| SPEED.MODE_2.1 | Speed Gen2 | Autoselect 2 | … | … |
| ERR.SPEED.1 | Speed 1 Bad | Unsuccessful | … | … |
| **Excel.xlsx** | | | | |

- ALL testing scenarios uniquely identified: the superset case

# Testplan: A Live Document

| Scenario ID | CUST_A | CUST_B | CUST_C |
|---|---|---|---|
| LP.DATA.1 | Y | | Y |
| ERR.LP.1 | Y | | Y |
| SPEED.MODE_1.1 | Y | Y | Y |
| SPEED.MODE_1.2 | Y | Y | Y |
| SPEED.MODE_2.1 | Y | Y | |
| ERR.SPEED.1 | Y | Y | Y |

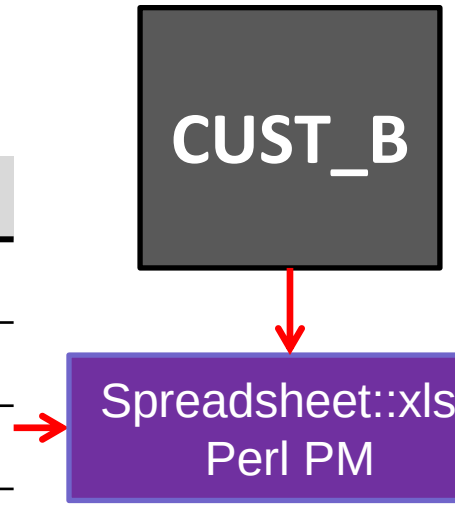



**LP = Low Power**

- Customer-specific scenarios are indicated in the testplan

# Testplan: A Live Document

| Scenario ID | CUST_A | CUST_B | CUST_C |
|---|---|---|---|
| LP.DATA.1 | Y | | Y |
| ERR.LP.1 | Y | | Y |
| SPEED.MODE_1.1 | Y | Y | Y |
| SPEED.MODE_1.2 | Y | Y | Y |
| SPEED.MODE_2.1 | Y | Y | |
| ERR.SPEED.1 | Y | Y | Y |




| Scenario ID | CUST_B |
|---|---|
| SPEED.MODE_1.1 | Y |
| SPEED.MODE_1.2 | Y |
| SPEED.MODE_2.1 | Y |
| ERR.SPEED.1 | Y |

**Can Review with Customer**

- Transform the testplan with customer CONFIG object
- Result informs testbench required for customer-specific simulation

# Agenda

- Root of the problem
- What I'm selling today to fix it
  - Cohesive scripting ecosystem
    - Testplan
    - Functional Coverage
    - Reporting Structure
- Conclusions

# Testing Scenarios

- Directed tests
- Directed-random tests
- Constrained-random test bench

# Directed Testing

- Scenario ID not tested when tests not run
- Scenario ID verified when tests run & pass
- Scenario ID fails when tests run & fail

| Scenario ID | Test | Regr |
|---|---|---|
| SPEED.MODE_1.1 | Test1 | Y |
| SPEED.MODE_1.2 | Test2 | Y |
| SPEED.MODE_2.1 | Test3 | |
| ERR.SPEED.1 | Test4 | |

**not run**

**not tested**

- No functional coverage required
- Regression report sufficient for verification status
- No scenarios outside testplan expected

**No Unknown Scenario Coverage**

# Directed-Random Testing

- Scenario ID not tested when tests not run
- Scenario ID verified when tests run & pass and functional coverage hit
- Scenario ID fails when tests run & fail

| Scenario ID | Test | Regr |
|---|---|---|
| SPEED.MODE_1.1 | Test1 | Y |
| SPEED.MODE_1.2 | Test1 | Y |
| SPEED.MODE_2.1 | Test1 | |
| ERR.SPEED.1 | Test2 | |

- Functional coverage required
- Regression report + functional coverage sufficient for verification status

**Some Unknown Scenario Coverage**

# Constrained-Random Testbench

- Scenario ID not tested when functional coverage not hit
- Scenario ID verified when functional coverage hit & tests pass
- Scenario ID fails when functional coverage hit & tests fail

| Scenario ID | Coverage | Test | Regr |
|---|---|---|---|
| SPEED.MODE_1.1 | group: a | Base | Y |
| SPEED.MODE_1.2 | group: b | Base | Y |
| SPEED.MODE_2.1 | group: c | Base | |
| ERR.SPEED.1 | group: d | EnErrs | |

**Unknown Scenarios Likely**
***In the Customer Config***

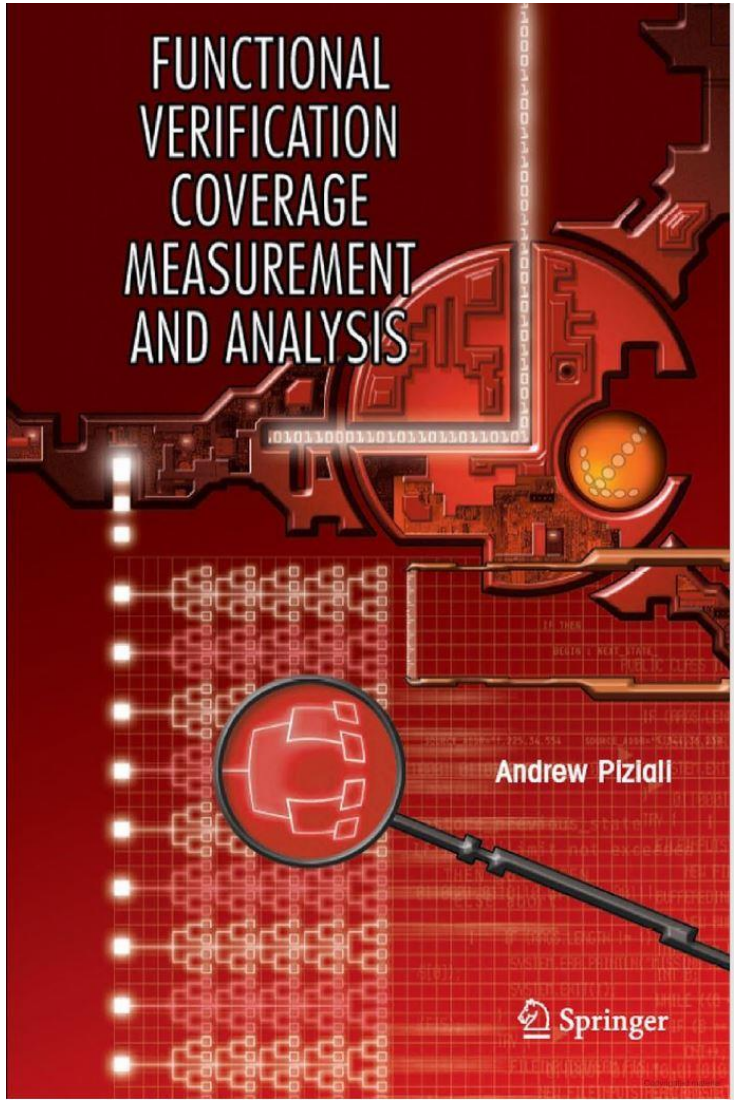
- Functional coverage required and is focus
- Regression report + functional coverage required for verification status

# Functional Coverage Model

- Internal process
- Model developed independent of testbench
- Metrics per scenario indicated in the Testplan

| Scenario ID | Coverage | Test | Regr |
|---|---|---|---|
| SPEED.MODE_1.1 | group: a | Base | Y |
| SPEED.MODE_1.2 | group: b | Base | Y |
| SPEED.MODE_2.1 | group: c | Base | |
| ERR.SPEED.1 | group: d | EnErrs | |

# Define coverage in Spreadsheet




Andrew Piziali,

*Functional Verification Coverage Measurement and Analysis*,

Springer Science+Business Media,

2008.

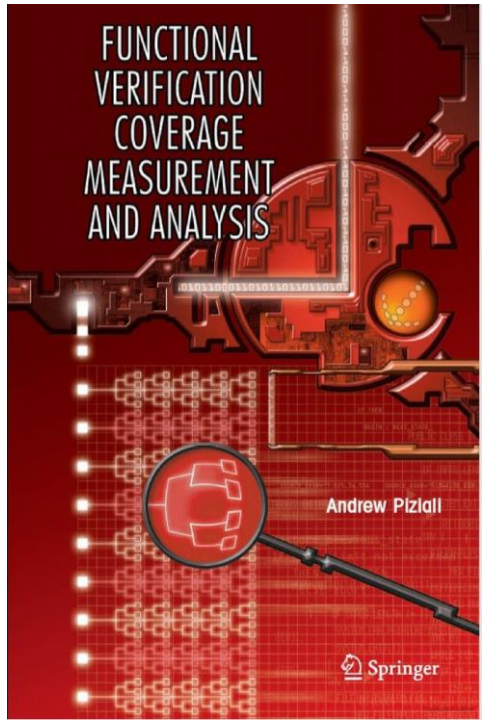
# Define coverage in Spreadsheet

- *Can* specify functional coverage as a table

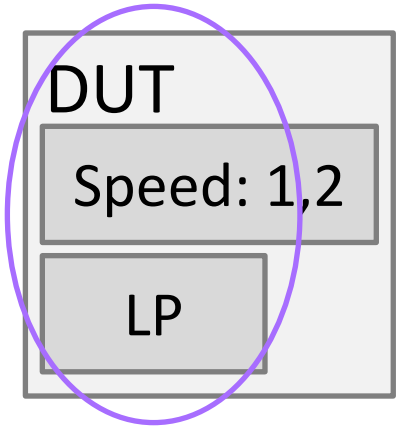
| Name | a | | |
|---|---|---|---|
| Points | M_SPEED | FW_SEL | SEL_INVALID |
| *autosel_gen1* | 1 | 0 | 0 |
| *fwsel_gen1* | 1 | 1 | 0 |
| *autosel_gen2* | 2 | 0 | 0 |

```
covergroup a;
  coverpoint M_SPEED { bins B0 = { 1 };
                       bins B1 = { 2 }; }

  cross M_SPEED, FW_SELECT, SEL_INALID {
    autosel_gen1: binsof(M_SPEED.B0) && … }
endgroup
```



# Generate coverage for instance




**CUST_C**

**LP = Low Power**

| Name | a | | |
|---|---|---|---|
| **Points** | M_SPEED | FW_SEL | SEL_INVALID |
| ***autosel_gen1*** | 1 | 0 | 0 |
| ***fwsel_gen1*** | 1 | 1 | 0 |
| ***autosel_gen2*** | 2 | 0 | 0 |

***not applicable***

generate_cg.pl | Spreadsheet::xlsx Perl PM

```
covergroup a;
  coverpoint M_SPEED { bins B0 = { 1 };
                       bins B1 = { 2 }; }

  cross M_SPEED, FW_SELECT, SEL_INALID {
    autosel_gen1: binsof(M_SPEED.B0) && … }
endgroup
```

# Use CONFIG.pm in Spreadsheet

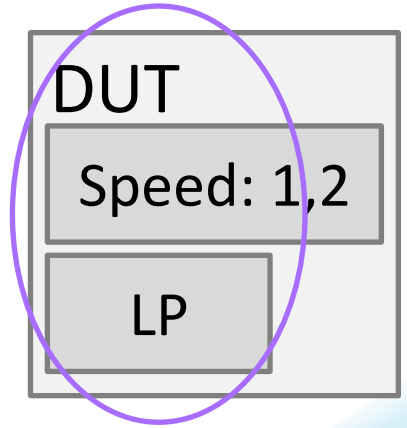
- Extended to use “config variables” to indicate applicable crosses

| Name | a | | | |
|---|---|---|---|---|
| Points | M_SPEED | FW_SEL | SEL_INVALID | C_SPEED |
| *autosel_gen1* | 1 | 0 | 0 | 1 |
| *fwsel_gen1* | 1 | 1 | 0 | 1 |
| *autosel_gen2* | 2 | 0 | 0 | 2 |

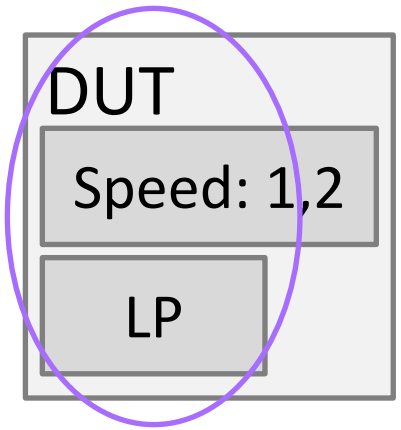
```
covergroup a;
  coverpoint M_SPEED { bins B0 = { 1 };
                       bins B1 = { 2 }; }

  cross M_SPEED, FW_SELECT, SEL_INALID { …
    autosel_gen2: binsof(M_SPEED.B1) && … }
endgroup
```

*don’t want these*



# Generate coverage for instance




| Name | a | | | |
|---|---|---|---|---|
| Points | M_SPEED | FW_SEL | SEL_INVALID | C_SPEED |
| *autosel_gen1* | 1 | 0 | 0 | 1 |
| *fwsel_gen1* | 1 | 1 | 0 | 1 |
| *autosel_gen2* | 2 | 0 | 0 | 2 |

CUST_C

generate_cg.pl | Spreadsheet::xlsx Perl PM

```
covergroup a;
  coverpoint M_SPEED { bins B0 = { 1 }; }

  cross M_SPEED, FW_SELECT, SEL_INALID {
    autosel_gen1: binsof(M_SPEED.B0) && … }
endgroup
```

# Agenda

- Root of the problem
- What I'm selling today to fix it
  - Cohesive scripting ecosystem
    - Testplan
    - Functional Coverage
    - Reporting Structure
- Conclusions

# Reporting

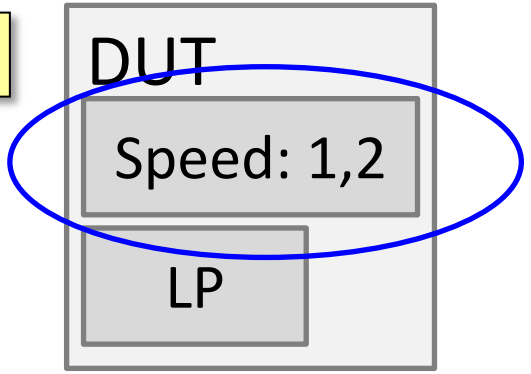
- Must report customer-specific metric
- Consider the Regression itself (pass/fail)
- Consider the Regression with respect to Testplan (pass/fail)
- Consider the Functional Coverage itself (score)
- Consider the Functional Coverage AND Regression with respect to Testplan

**This is our Customer-specific Verification Status**

# Regression Report vs Testplan

- In this regression, only Base test is run
  - Base test may be run 1000s of times
- Assume Base test passes 965 / 1000 times:

**LP = Low Power**




**Regression score = (pass_rate * covered_scenarios) / total_scenarios**

**Regression score = (96.50% * 3) / 4 = 72.375%**

| Scenario ID | Subfeature | Coverage | Test | Regr |
|---|---|---|---|---|
| SPEED.MODE_1.1 | Autosel G1 | cross: tb::a.autosel_gen1 | Base | Y |
| SPEED.MODE_1.2 | FW sel G1 | cross: tb::a.fwsel_gen1 | Base | Y |
| SPEED.MODE_2.1 | Autosel G2 | cross: tb::a.autosel_gen2 | Base | Y |
| ERR.SPEED.1 | Sel Failed | cross: tb::a.sel_err | EnErrs | |

# Testplan Report

- Each scenario indicates pass/fail with simulation & coverage

| Scenario ID/Metric | | Instances | Rate | Result |
|---|---|---|---|---|
| SPEED.MODE_1.1 | | | | fail fcov_met |
| 1 | Base | 1000 | 96.50% | fail |
| 1 | cross: tb::a.autosel_gen1 | | 100% | fcov_met |

| Scenario ID | Subfeature | Coverage | Test | Regr |
|---|---|---|---|---|
| SPEED.MODE_1.1 | Autosel G1 | cross: tb::a.autosel_gen1 | Base | Y |
| SPEED.MODE_1.2 | FW sel G1 | cross: tb::a.fwsel_gen1 | Base | Y |
| SPEED.MODE_2.1 | Autosel G2 | cross: tb::a.autosel_gen2 | Base | Y |
| ERR.SPEED.1 | Sel Failed | cross: tb::a.sel_err | EnErrs | |

# Testplan Report with FCov



- Testplan score combines functional coverage with regression score
- Assume 96.50% pass rate and SPEED.MODE_2.1 not covered

**Regression score = (pass_rate * covered_scenarios) / total_scenarios)**

**Testplan score = Regression score * functional_coverage_score**

**Testplan score = ((96.50% * 3) / 4) * (2 / 3) = 48.25%**

| Scenario ID | Subfeature | Coverage | Test | Regr |
|---|---|---|---|---|
| SPEED.MODE_1.1 | Autosel G1 | cross: tb::a.autosel_gen1 | Base | Y |
| SPEED.MODE_1.2 | FW sel G1 | cross: tb::a.fwsel_gen1 | Base | Y |
| SPEED.MODE_2.1 | Autosel G2 | cross: tb::a.autosel_gen2 | Base | Y |
| ERR.SPEED.1 | Sel Failed | cross: tb::a.sel_err | EnErrs | |

# Reporting Results

- Customer specific view
- Snapshot regression passing rate
- Snapshot regression score
- Snapshot testplan score
- History graph indicates where we are and where we need to go

- Can isolate individual testing scenarios in a Constrained Random Verification Environment

# Reporting Results the Customer

- *"Yeah we've covered that because it's reported in the testplan"*
  - It's possible to miss a mode or feature

    **Tie mode/feature directly to Testplan**
    **A lot harder to miss**

  - Was the feature *actually* covered?

    **Yes or no, explicitly**

  - Show me the feature covered *during* the customer-specific mode-of-operation

    **This built-in to the architecture**

    **No hand-waving; no heursitics**

    **No waivers**

| Scenario ID/Metric | | Instances | Rate | Result |
|---|---|---|---|---|
| SPEED.MODE_1.1 | | | | fail fcov_met |
| 1 | Base | 1000 | 96.50% | fail |
| 1 | cross: tb::a.autosel_gen1 | | 100% | fcov_met |

# Conclusions

- Used this methodology in a PCIe Express Subsystem Project
  - >8 simultaneous sub-projects
  - Able to isolate verification status per sub-project for customer
  - Able to review customer-specific views with the customer
- Is this the only/best way of doing things?
  - No, this is the way we chose
  - What's the best?  I don't know, this is a conversation we must have.
- Cohesive and comprehensive general purpose verification architecture in order to manage a highly configurable design
- See Paper for more Information

# Agenda

- Root of the problem
- What I'm selling today to fix it
  - Cohesive scripting ecosystem
    - Testplan
    - Functional Coverage
    - Reporting Structure
- Conclusions

BROADCOM®

snug
Synopsys Users Group
Austin

# Thank You