Expediting the Code Coverage Closure Using Static Formal Techniques – A Proven Approach at Block and SoC Levels

Questa Formal team and friends

2015 DVCon India

D1A2.1-DV

2015
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
INDIA

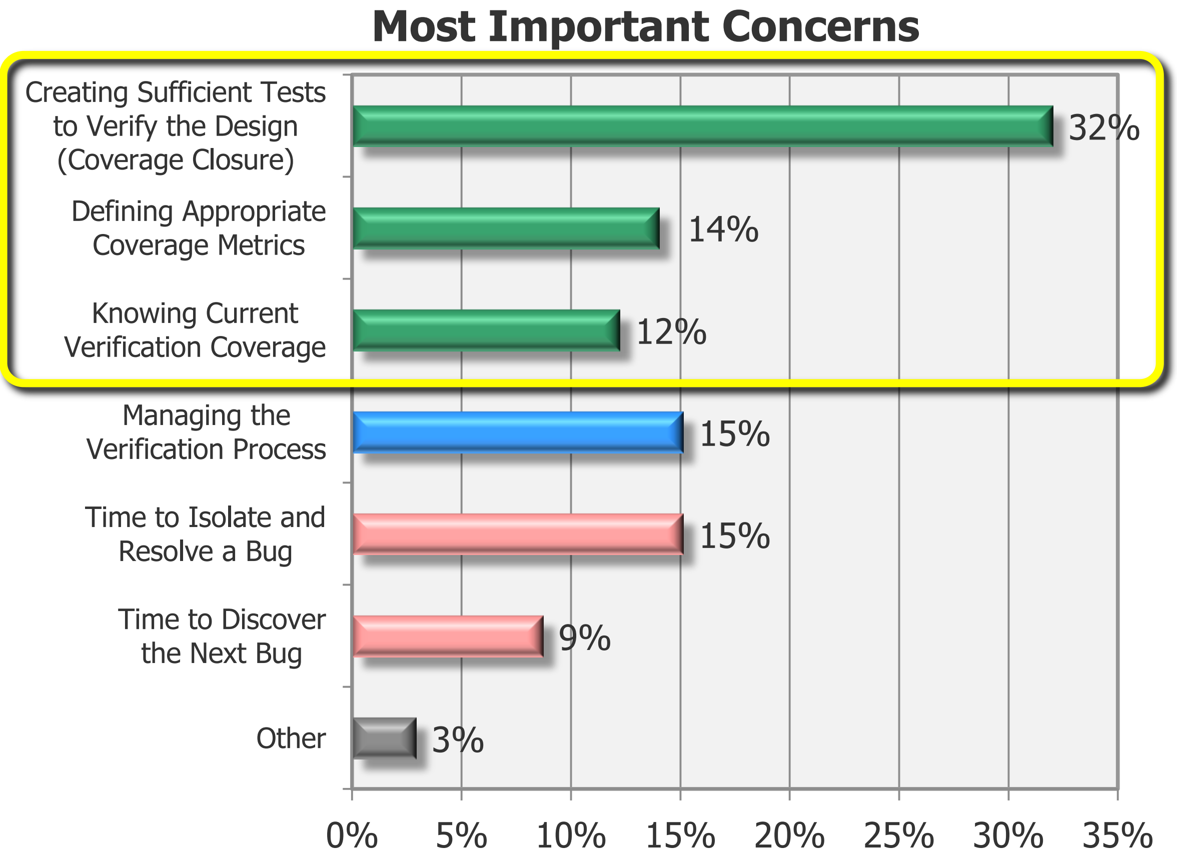
Mentor Graphics®

Agenda

- Introduction
- Coverage Backgrounder
- Targeting Unreachable Coverage with Formal
- Reaching Coverage Closure Faster
- Conclusion

Functional Verification Challenges

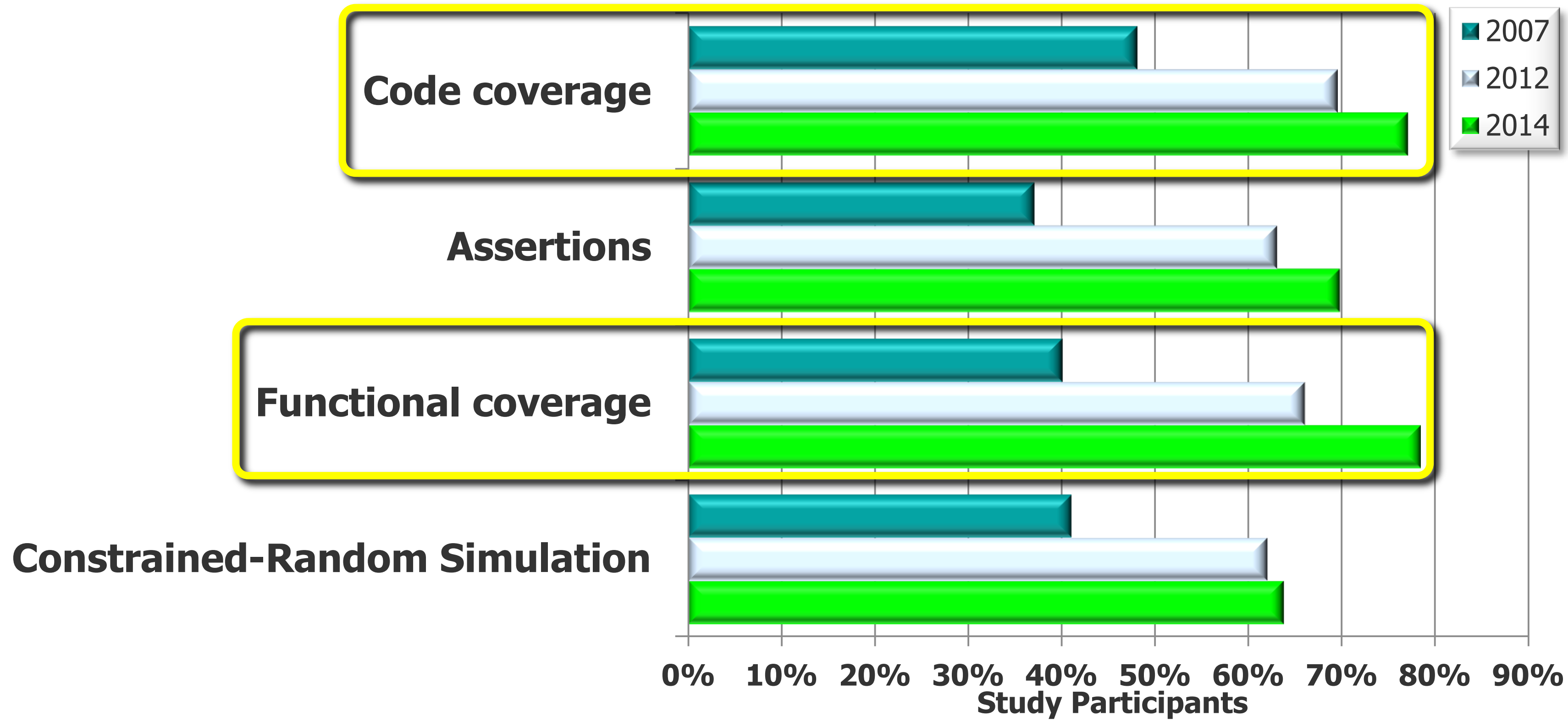
Coverage Ranks at the Top of Project Management's Concerns



Wilson Research Group
2010 Functional Verification Study

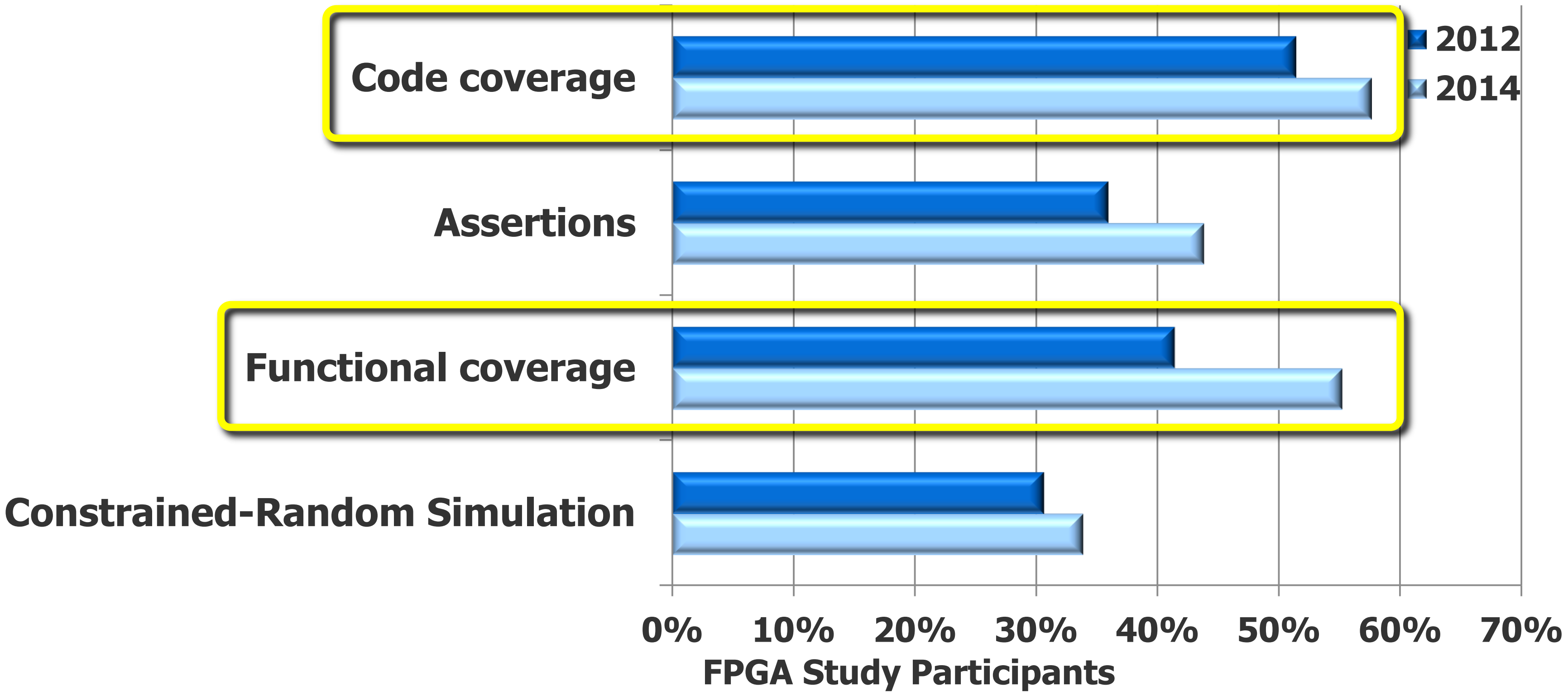
Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

Industry Trends: ASIC D&V Use of Code Coverage



Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

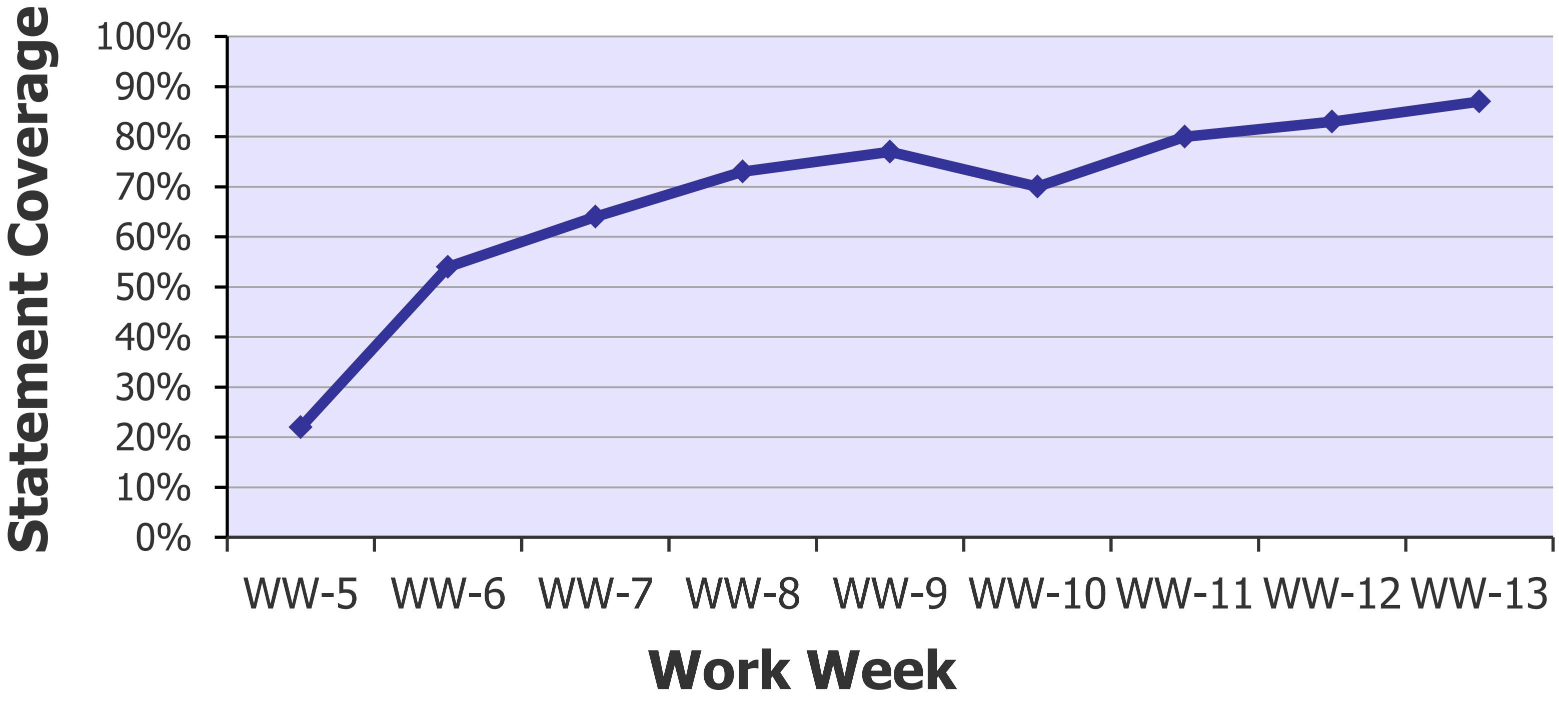
FPGA Verification Technique Trends



Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

Code Coverage Challenge

Are we there yet?



Data & graph from 2011 customer paper

- ✓ 270 man weeks to do coverage waiver analysis for one design
- ✓ 180 man weeks to write missing tests
- ➢ That’s almost 9 man-years!

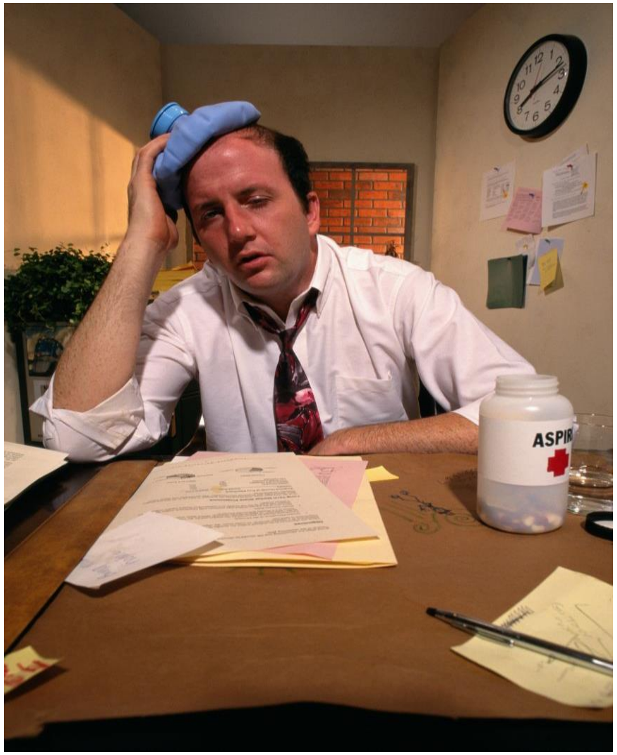
Verification Management Challenge

What's been covered?

What needs to be covered?

How long before we are done?

How can I improve on my processes



What You Will Learn Today

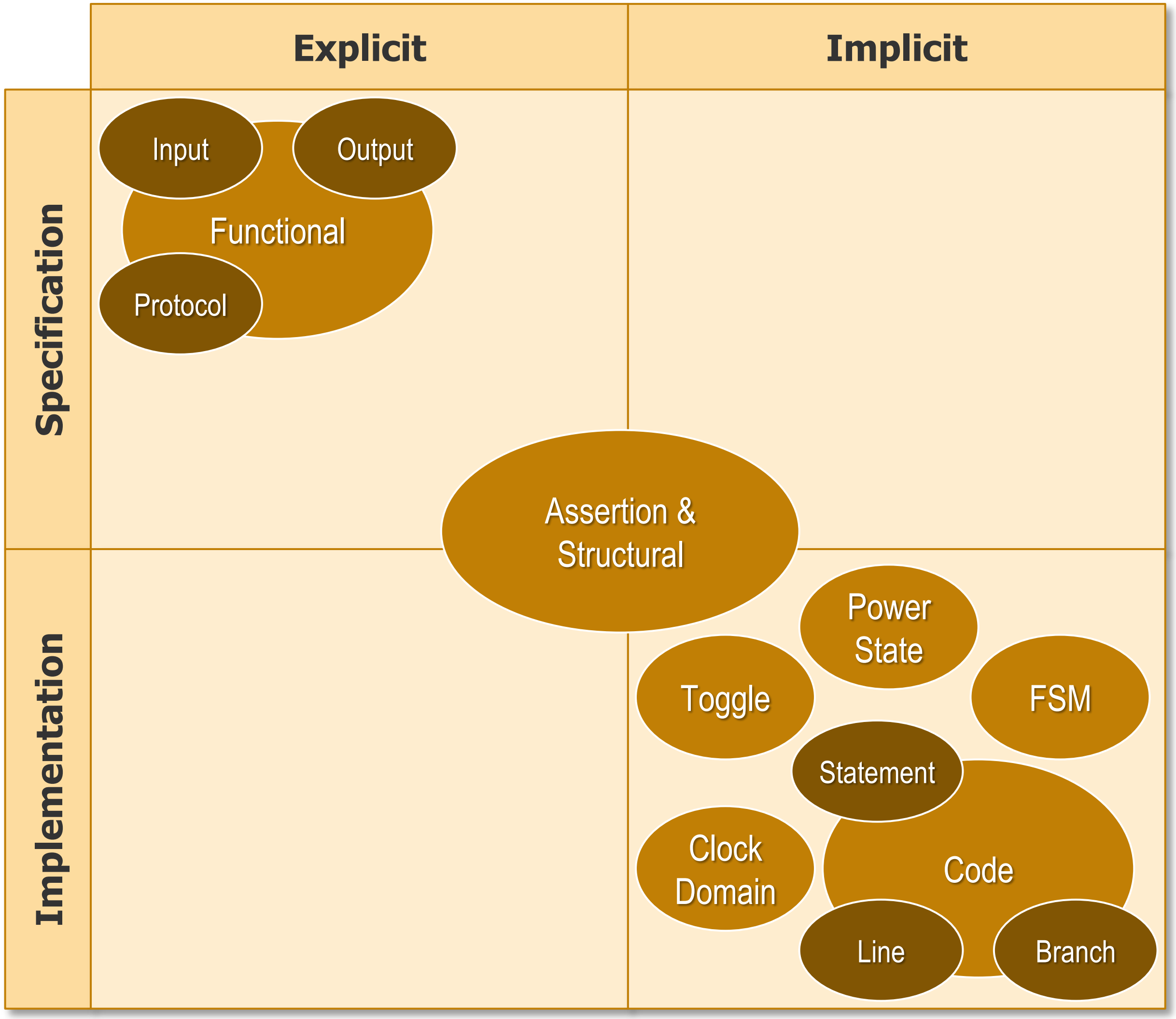
- Primer on coverage types and how coverage is recorded
- How to rapidly identify “unreachable” coverage areas
- How to reach your coverage goals faster

Agenda

- Introduction
- Coverage Backgrounder
- Targeting Unreachable Coverage with Formal
- Reaching Coverage Closure Faster
- Conclusion

Types of Coverage

- Origin of Source
  - Specification
  - Implementation
- Method of Creation
  - Explicit
  - Implicit
- Both are Required
  - Functional
  - Structural



Coverage: Implementation

What areas of the design have been touched by verification

- Code Coverage
  - — Did these lines/branches of code get exercised?
  - — Automated in the simulation environment
  - — One of the basic design verification signoff metrics
  - — A basic measure with little correlation to functionality

- FSM Coverage
  - — Did all the states and transitions get exercised?
  - — Automated in the simulation environment
  - — One of the basic design verification signoff metrics
  - — Typically included with code coverage

Code Coverage: Statement (s)

- Counts the execution of each statement on a line
  - Even if multiple statements
- Example:

```
always @(posedge clk or negedge rstn)

…

reg <= dat;

…

C <= A && B;
```

Count the statements and the number of times each one is hit

- Report style based on number of Statements

```
Enabled Coverage      Active    Hits    Misses   % Covered
----------------      ------    ----    ------   ---------
Stmts                    415     387        28        93.2
```

Code Coverage: Branch (b)

- Counts the execution of each conditional “if/then/else” and case statement
  - All true and false branches are considered
  - Each (if/else if/else | case) element counts as a branch
- Example (if statement):

```
if (!rstn)
    q <= 1’b0;
 else
    q <= d;
```

Count total coming into if statement, count times each branch taken

- Report style based on number of Branches

```
Enabled Coverage      Active      Hits   Misses % Covered
----------------      ------      ----   ------ ---------
Branches                  47        45        2      95.7
```

Code Coverage: Condition (c)

- Analyzes the decision made in “if” and ternary statements
  - Considered extension of branch coverage
- Example:

```
if (ce && we)
     1    0/1
```

All FEC conditions must be hit:

ce = 0,1; we = 0,1

ce is uncovered:

Never hit 0

- Report style based on Focused Expression Coverage

```
Enabled Coverage      Active   Hits  Misses   % Covered
----------------      ------   ----  ------   ---------
FEC Condition Terms       16     13       3        81.2
```

Code Coverage: Expression (e)

- Analyzes expressions on the right hand side of an assignment
- Example:

```
wire C = A && B
         1   0/1
```

All FEC conditions must be hit:

A = 0,1; B = 0,1

A is uncovered:

Never hit 0

- Report style based on Focused Expression Coverage

```
Enabled Coverage       Active   Hits  Misses   % Covered
----------------       ------   ----  ------   ---------
FEC Condition Terms        25     14      11        56.0
```

Code Coverage: Toggle (t)

- Counts each time a logic node transitions one state to another
- Example:

```
reg FF_A;
always @(posedge clk)
          FF_A <= din;
```

To be covered
FF_A must toggle:

0 to 1 and 1 to 0
bin bin

- Report style based on Toggle Bins

```
Enabled Coverage      Active    Hits  Misses    % Covered
----------------      ------    ----  ------    ---------
Toggle Bins              356     351       5         98.5
```

Code Coverage: FSM (f)

- Counts the states and transitions of a FSM
- Example:

```
FSM States: S1; S2; S3
FSM Transitions: S1 -> S1; S1 -> S2;
S2 -> S3; S2 -> S1; S3 -> S1
```

All States and Transitions must be hit

This Transition not exercised (uncovered)

- Report style based on FSM States and Transitions

```
Enabled Coverage    Active    Hits  Misses   % Covered
----------------    ------    ----  ------   ---------
States                   3       3       0     100.0
Transitions              5       4       1      80.0
```

Coverage: Structural

How much verification has stressed the design

- Assertion Coverage
  - How many times did the assertion get evaluated, pass, fail
  - Automated in the simulation environment
  - Doesn't answer the questions:
    - *Is the assertion implemented correctly? (check anything of value)*
    - *Are there enough assertions?*

- Structural Coverage
  - Measures corner case type activity
    - How many times was my FIFO empty, full, hit high water mark
  - Implementation specific, can be automated with assertions
  - How well is the TB environment stressing the design?

Cover Statements/Properties

- Properties and Sequences can be “covered”

- Useful for checking temporal behavior of your design
  - SVA/PSL designed for describing temporal behavior
  - Cover statements typically target a sequence of events
    - Can also target single cycle events
    - Simulation will count the number of occurances
    - Formal will tell you if it’s reachable or unreachable

- Examples:

```
cov_sm_trans: cover property (@(posedge clk) cstate == TRANS );
cov_ddr_wr: cover property (@(posedge clk) ddr_act ##[1:20] ddr_wr);
sequence apb_wr;
    pselx && pwrite && !penable ##1 pselx && pwrite && penable;
endsequence
cov_b2b_wr: cover property (@(posedge clk) apb_wr ##1 apb_wr);
cov_seq: cover property (@(posedge clk) a ##2 b ##[1:3] c[*4] ##1 a );
```

Coverage: Functional

What features of the design have been tested

- Transactional Coverage
  - Measures interface type transactions
    - *Have I covered all my AHB/AXI transactions?*
  - Typically implemented with cover groups/points
  - Often used with complex TB environments (TLA, CR, iTBA)

- Functional Coverage
  - Measures occurrence of functional events
    - *Did my design do back-to-back writes?*
  - Typically implemented with cover groups and cover directives
  - Used in complex TB environments, correlate function to spec

Functional Coverage

- Must be specified by the user and cannot be automatically inferred from design
- Validates actual functionality
- Formal specification of verification plan
  - Direct correlation between requirements and verification
- Measures verification completeness against specification
  - *Have I verified all functional requirements?*
  - *Have I covered the entire verification plan?*
  - *Are my tests adding values to my verification goal?*
  - *Have I exercised all corner cases in my design?*
  - *Am I done?*
- Counts how many times “interesting” things occur

CoverGroups

- System Verilog CoverGroups
  - coverpoints and coverbins used to categorize/display data
  - Must be instantiated

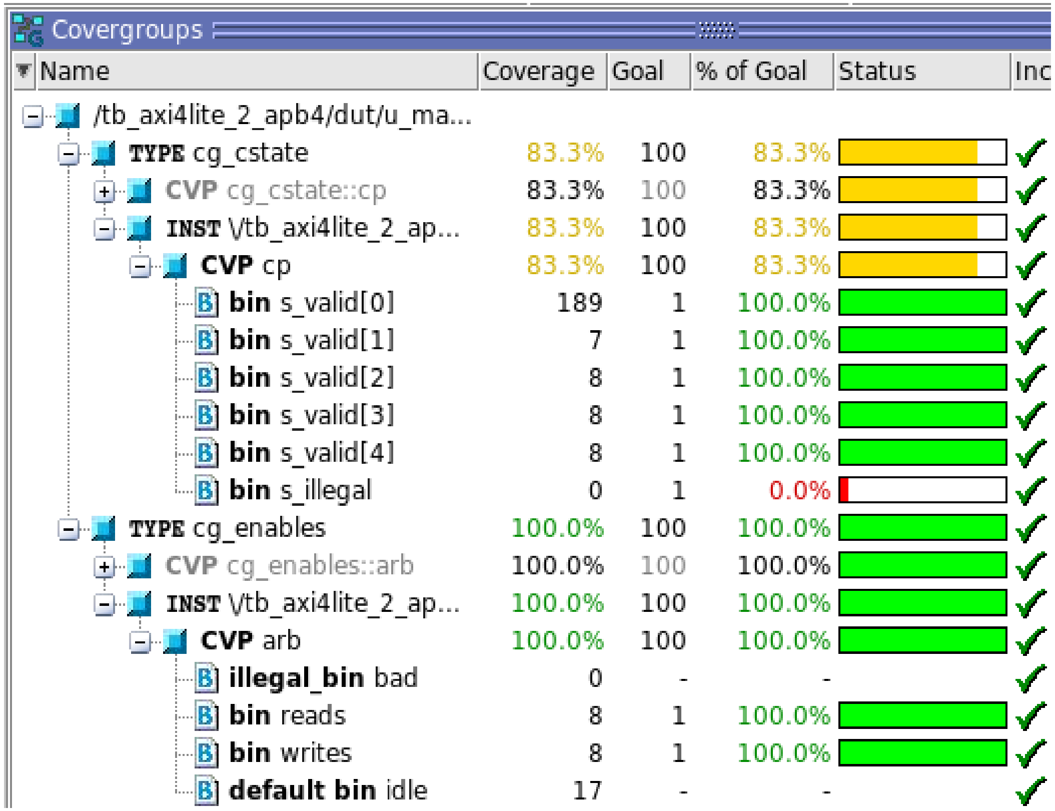
- Example: CG in module

```
module cover_pci_master32_sm ( input clk_in, input [3:0] cur_state);
covergroup cg_cur_state @ (posedge clk_in);
  cp: coverpoint cur_state {
        bins s_idle = {1};
        bins s_addr = {2};
        bins s_tran = {4};
        bins s_end  = {8};   }
endgroup : cg_cur_state;
cg_cur_state cg_cur_state_inst = new;
endmodule
```

Are coverbins reachable?

CoverGroups Example: FSM, Arbiters

```
covergroup cg_cstate @ (posedge clk);
  cp: coverpoint cstate {
        bins s_valid [5] = {1,2,4,8,16};
        bins s_illegal   = {0,3,5,6,7,[9:15]}; }
endgroup : cg_cstate;
cg_cstate cg_cstate_inst = new;

wire [1:0] enables = {wr_en,rd_en};
wire en = $changed(enables);
reg len;
always @*
if (!clk) len <= en;
wire gclk = clk & len;
covergroup cg_enables @ (posedge gclk);
  cp: coverpoint enables {
        bins reads        = {1};
        bins writes       = {2};
        illegal_bins bad  = {3};
        bins idle         = default; }
endgroup : cg_enables;
cg_enables cg_enables_inst = new;
```



Covergroups

Name	Coverage	Goal	% of Goal	Status	Inc
/tb_axi4lite_2_apb4/dut/u_ma...					
TYPE cg_cstate	83.3%	100	83.3%		✓
CVP cg_cstate::cp	83.3%	100	83.3%		✓
INST \/tb_axi4lite_2_ap...	83.3%	100	83.3%		✓
CVP cp	83.3%	100	83.3%		✓
bin s_valid[0]	189	1	100.0%		✓
bin s_valid[1]	7	1	100.0%		✓
bin s_valid[2]	8	1	100.0%		✓
bin s_valid[3]	8	1	100.0%		✓
bin s_valid[4]	8	1	100.0%		✓
bin s_illegal	0	1	0.0%		✓
TYPE cg_enables	100.0%	100	100.0%		✓
CVP cg_enables::arb	100.0%	100	100.0%		✓
INST \/tb_axi4lite_2_ap...	100.0%	100	100.0%		✓
CVP arb	100.0%	100	100.0%		✓
illegal_bin bad	0	-	-		✓
bin reads	8	1	100.0%		✓
bin writes	8	1	100.0%		✓
default bin idle	17	-	-		✓

Coverage: Metrics

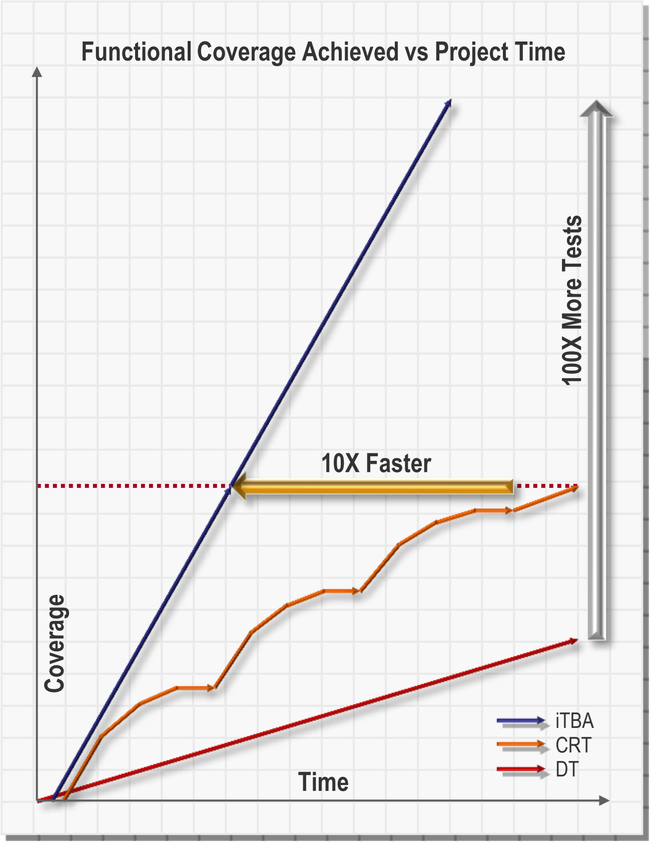
- Basic: Code/FSM/Assertion Coverage
  - Checks that all RTL has been exercised
  - All assertions have been exercised
- Semi-Automated: Transaction/Structural Coverage
  - Checks that all types of transactions have occurred
  - Ensures that the tests have sufficiently stressed the design
- Advanced: Functional Coverage
  - Checks that all the requirements for the design have been tested
  - Does the design work in all scenarios?
- All these coverage types are measured and tracked to determine when verification is complete and the chip can tape out

Coverage: Metric Holes

- Code/FSM/Assertion Coverage
  - Functional dead code and unreachable FSM states/transitions
  - Modes of the design that create dead code
  - Time can be wasted trying to hit these holes!
- Transaction/Structural Coverage
  - TB doesn’t stress the design enough
  - Incomplete models don’t exercise all transactions
- Functional Coverage
  - Incomplete spec or planning, lack of knowledge/time
- Proper test planning can mitigate some of these challenges
- Making use of automated formal techniques such as Questa CoverCheck can minimize time to closure

Common Methods to Achieve Coverage

- Directed Tests
  - Can target specific areas
  - Less setup typically
- Constrained Random Tests
  - More sophisticated setup
  - More automated to coverage
- Intelligent Testbench Automation
  - >10X Faster Coverage Than CRT
  - >100X More Tests Than DT
- Goals
  - Achieve total coverage faster
  - With fewer resources
  - In less time



Typical Coverage Closure Methods

- Fix design issues that prevent code coverage from being achieved

- Run more vectors to hit missing code coverage
  - Directed tests
  - Constrained random
  - Intelligent test bench generation
  - Spend a lot of time analyzing and applying new vectors

- Apply formal methods to determine coverage reachability

- Add exclusions by hand
  - Sometimes the simulator can add automated exclusions

- Use an automated flow to generate exclusions for unreachable coverage elements

Coverage Backgrounder Summary

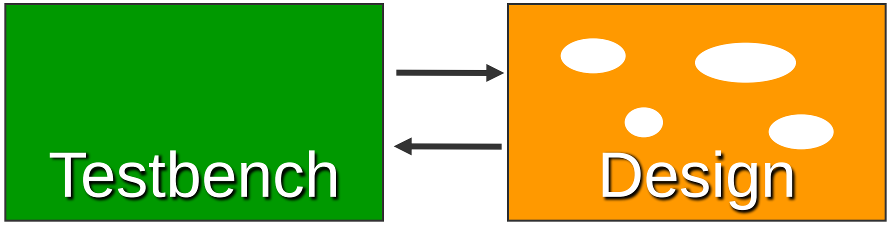
- There is no “silver bullet” structural and functional coverage methodology or metric

- Multifaceted simulation and formal-based automation, guided by the D&V engineer’s judgment, is required

Agenda

- Introduction
- Coverage Backgrounder
- Targeting Unreachable Coverage with Formal
- Reaching Coverage Closure Faster
- Conclusion

Coverage Closure Challenges



Today coverage-driven verification is a well established methodology

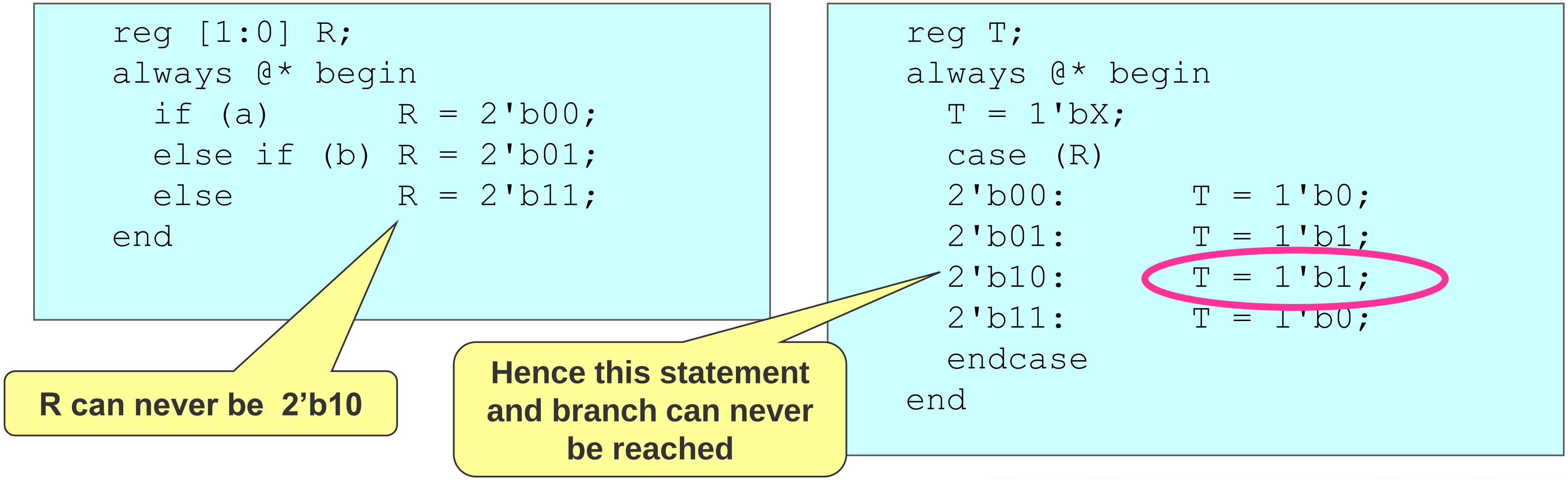
Question: *What if certain parts of the design simply cannot be reached?*

Answer: You will run extra constrained-random tests to try to cover these parts

This can lead to a lot of wasted effort!

Example: Branch/Statement Coverage

- Dead code easily slips into the design
  - Especially after changes are made
- Dead code often identifies incorrect assumptions
  - Leading to critical bugs due to differing interpretation of design requirements
- Possibly synthesizes into logic that is not needed



```
reg [1:0] R;
always @* begin
  if (a)         R = 2'b00;
  else if (b)  R = 2'b01;
  else           R = 2'b11;
end
```

```
reg T;
always @* begin
  T = 1'bX;
  case (R)
  2'b00:      T = 1'b0;
  2'b01:      T = 1'b1;
  2'b10:      T = 1'b1;
  2'b11:      T = 1'b0;
  endcase
end
```

R can never be 2’b10

Hence this statement and branch can never be reached

Example: Condition/Expression Coverage

- Design configuration can have a large impact here
  - Has every combination of signals been exercised?
  - Is every combination of signals possible?

If “use_conf” is tied to 0 in the design, the condition “1-” of the two signals isn’t reachable

```
always @*
case (cstate)
            0
IDLE: if (!use_conf && !data_empty)
                nstate <= RR_RESP;
        else
                nstate <= RR_IDLE;
…
endcase
```

Example: Toggle Coverage

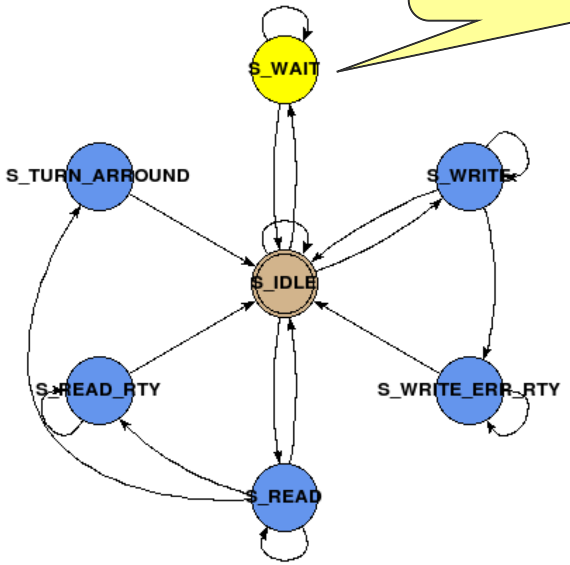
- Typically registers and signals can’t toggle due to configuration or some other constraint/bug in the design

```
always @(posedge pclk or negedge prstn)
if (!prstn)
        b_active <= 1'b0;
else
               0
        if (apb_wr && bready)
                b_active <= 1'b0;
   else if (bready && b_active)
        b_active <= 1'b0;

                    0
        else if (apb_wr && !bready)
                b_active <= 1'b1;
        else
                b_active <= b_active;
```

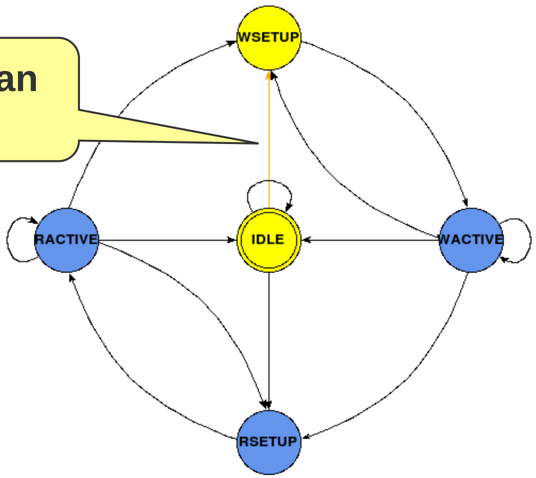
Some signals which are stuck due to either configuration or a bug in the design prevent the b_active signal from toggling 0 => 1

Example: FSM Coverage

- Indicates an over specified state machine
  - May lead to unused logic
- Easily overlooked in simulation
  - Info is passed to simulation for exclusion in the set of coverage goals

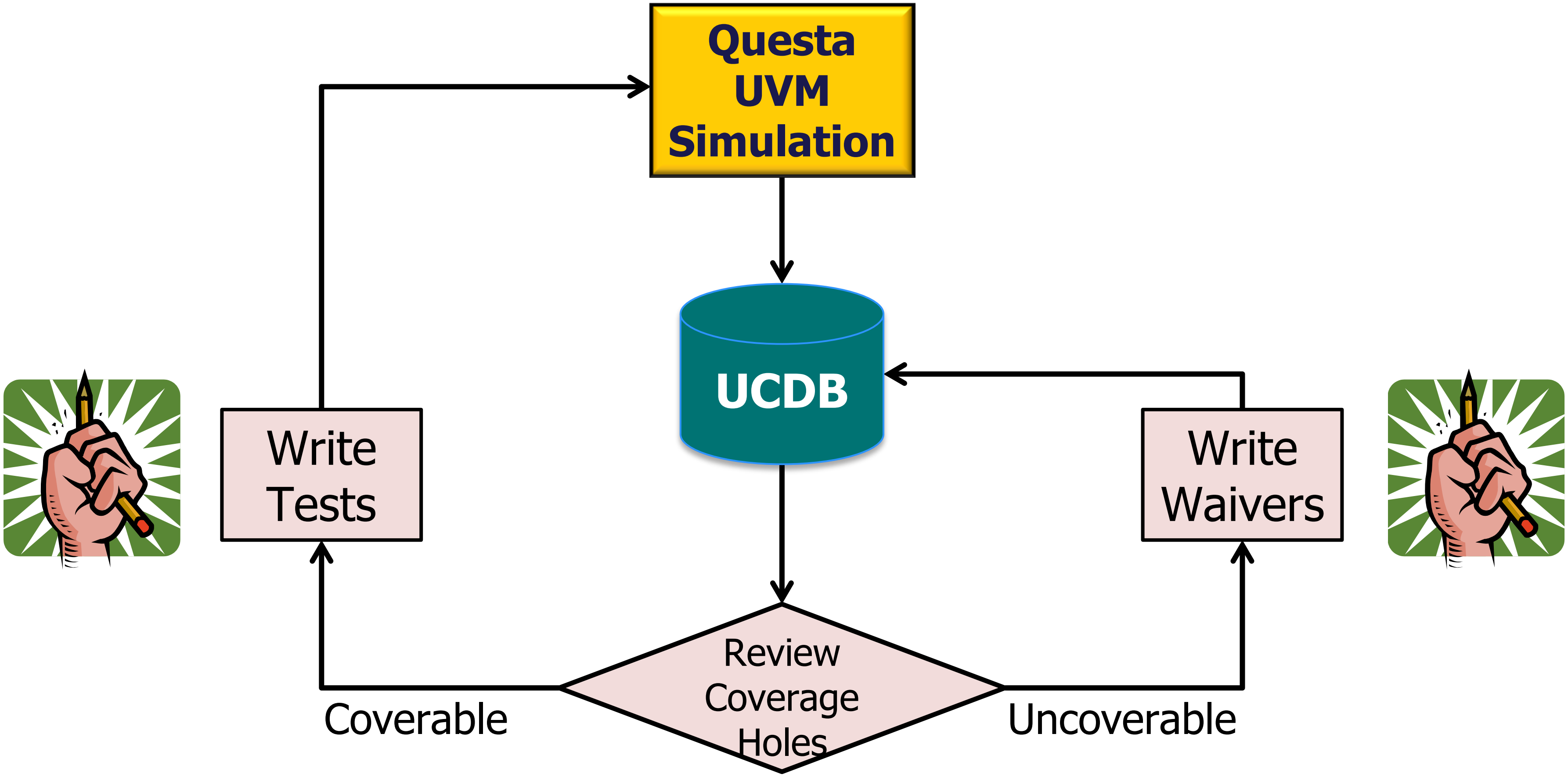


This state is deadlocked



This transition can never happen

Traditional Coverage Closure



Questa UVM Simulation

UCDB

Write Tests

Write Waivers

Review Coverage Holes

Coverable

Uncoverable

Questa CoverCheck

Automatic code coverage enhancement solution

Goal: 100% Code Coverage

Difficult to achieve:
1. Some coverage items cannot be reached
2. Other coverage items are difficult to hit

Problem: Wasted Time

- Engineers waste time manually identifying unreachable coverage & justifying waivers

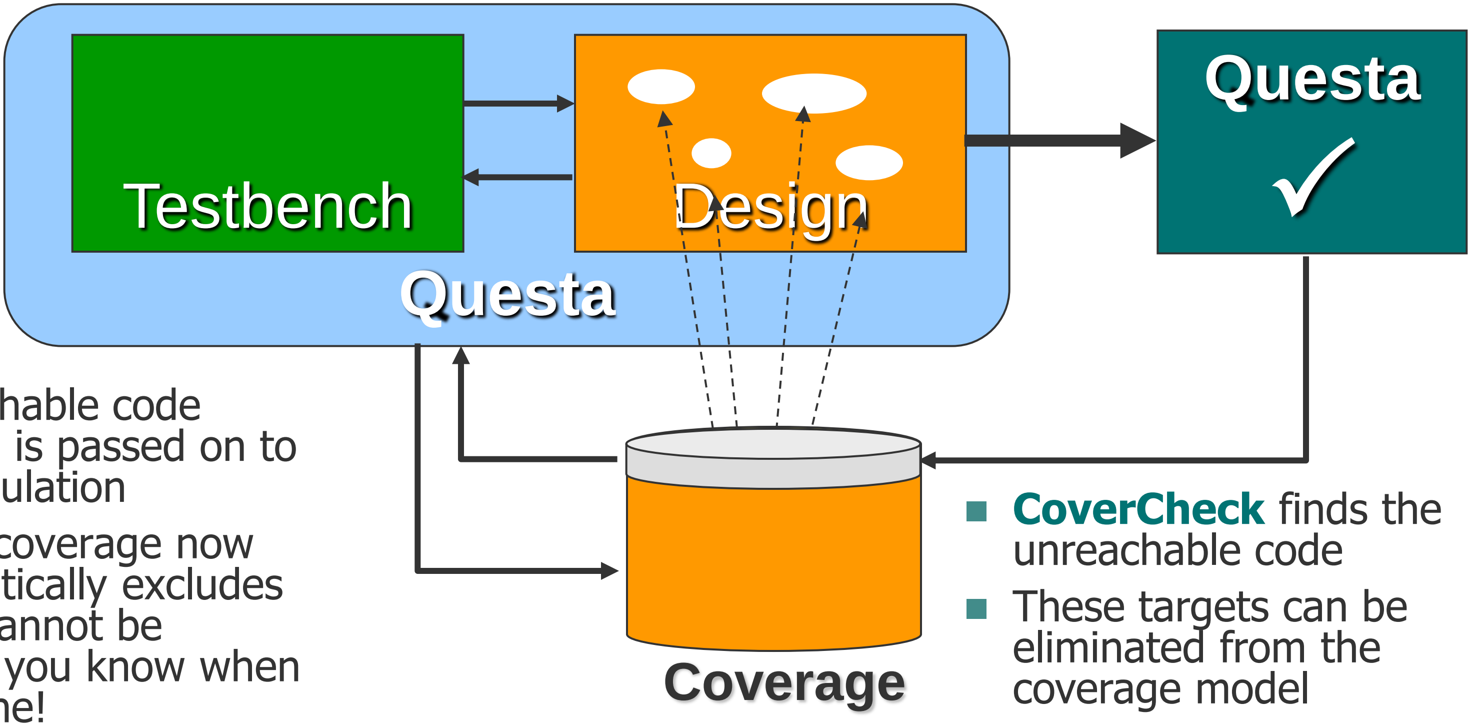
Problem: High Effort

- Requires significant manual effort to create complex test scenarios

Questa CoverCheck

- Identifies and prunes unreachable goals
- Guide test generation for reachable goals

The Questa CoverCheck Methodology



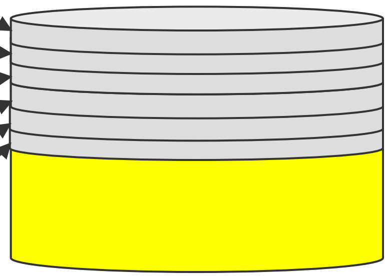
- The unreachable code information is passed on to Questa Simulation
- Measuring coverage now will automatically excludes code that cannot be reached so you know when you are done!

- **CoverCheck** finds the unreachable code
- These targets can be eliminated from the coverage model

No more is time wasted to try to cover unreachable code

Checks for Coverage Exclusions

- Branch
  - Unreachable if/else and case branches
- Condition/Expression
  - Unreachable FEC conditions
- Statement
  - Unreachable lines of code
- Toggle
  - Unreachable register transitions
- FSM
  - Unreachable FSM states and transitions
- Covergroups
  - Unreachable covergroup bins



Unreachable items are automatically excluded from your coverage model

The Coverage Improvement Process

1. Generate Final UCDB file from simulations

2. Run Questa CoverCheck reading final UCDB
   — Target uncovered code coverage elements
   — Run major blocks

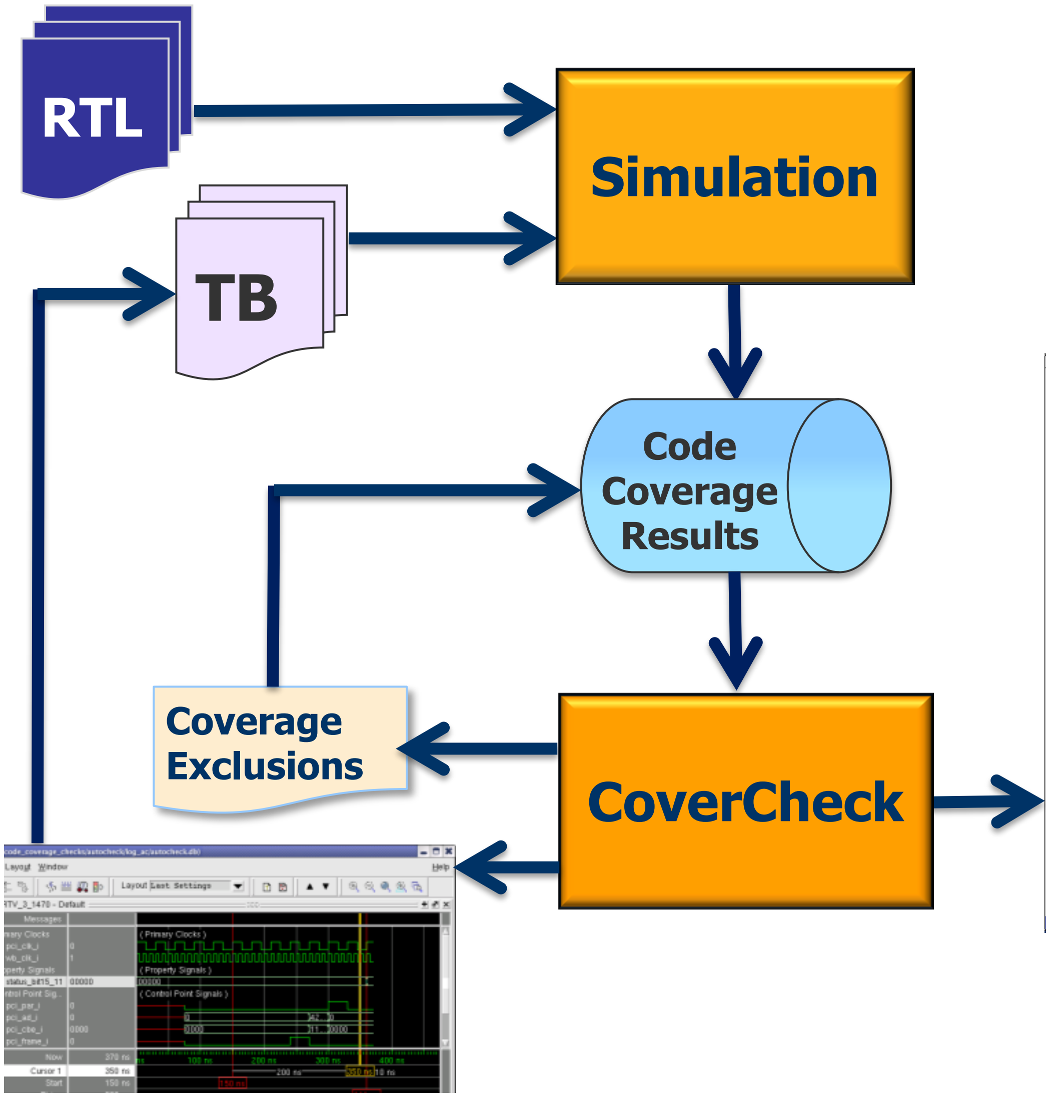
3. Generate the exclude file

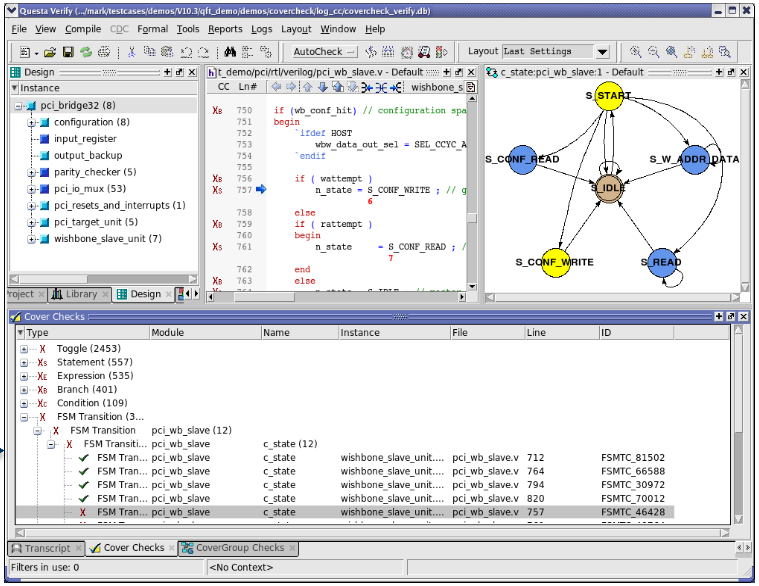
4. Apply exclusions to your simulation results
   — Update existing .ucdb file with exclude file

5. Report coverage
   — Track and manage coverage data

Questa CoverCheck Verification Flow

Use static analysis to improve simulation results!




Questa Verify (.../mark/testcases/demos/V10.3/qft_demo/demos/covercheck/log_cc/covercheck_verify.db)
File View Compile CDC Formal Tools Reports Logs Layout Window Help
AutoCheck
Layout Last Settings
Design
Instance
t_demo/pci/rtl/verilog/pci_wb_slave.v - Default
CC Ln#
wishbone_s
c_state:pci_wb_slave:1 - Default
S_START
S_CONF_READ
S_W_ADDR_DATA
S_IDLE
S_CONF_WRITE
S_READ
Project Library Design
Cover Checks
Transcript Cover Checks CoverGroup Checks

- pci_bridge32 (8)
  - configuration (8)
  - input_register
  - output_backup
  - parity_checker (5)
  - pci_io_mux (53)
  - pci_resets_and_interrupts (1)
  - pci_target_unit (5)
  - wishbone_slave_unit (7)

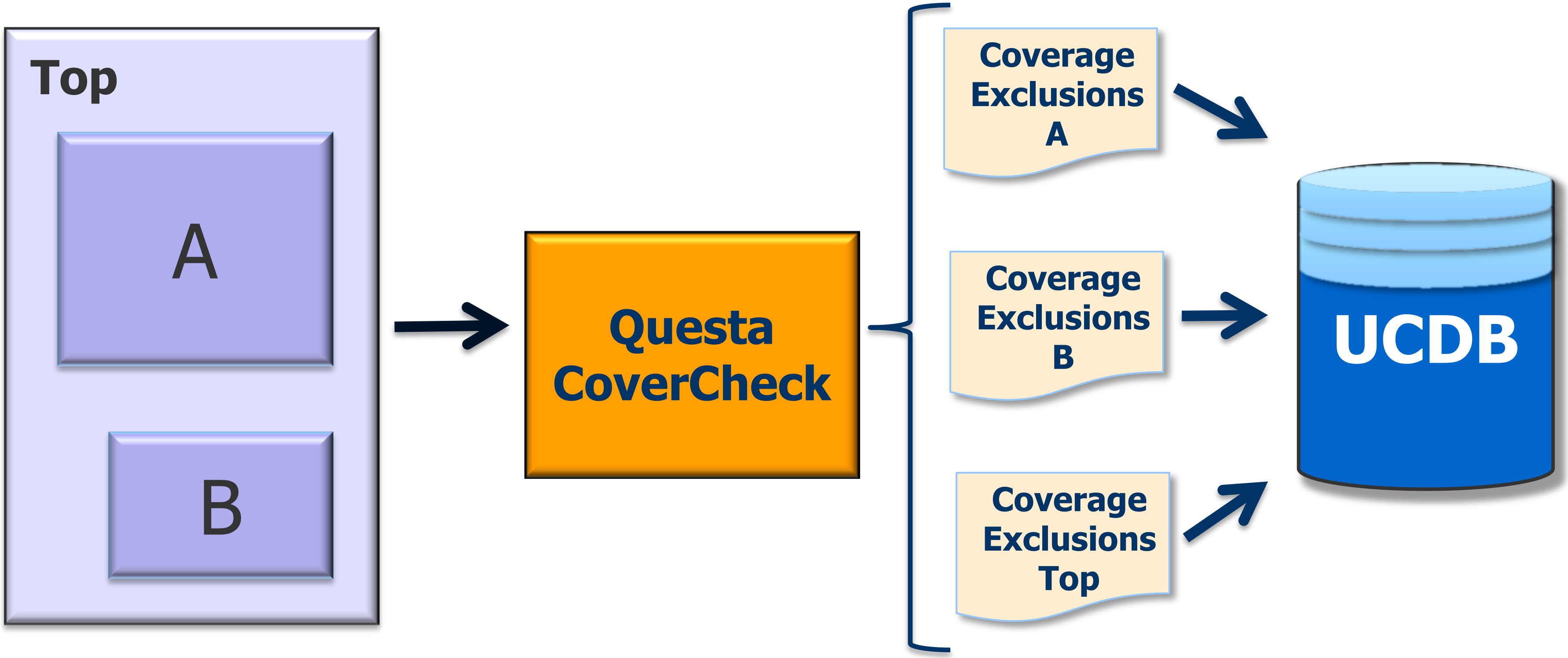
```
XB     if (wb_conf_hit) // configuration spa
       begin
           `ifdef HOST
               wbw_data_out_sel = SEL_CCYC_A
           `endif

XB         if ( wattempt )
XS             n_state = S_CONF_WRITE ; // g
                         6
           else
XB         if ( rattempt )
           begin
XS             n_state      = S_CONF_READ ; /
                              7
           end
XB         else
```

Type	Module	Name	Instance	File	Line	ID
X Toggle (2453)						
XS Statement (557)						
XE Expression (535)						
XB Branch (401)						
XC Condition (109)						
X FSM Transition (3...						
X FSM Transition	pci_wb_slave (12)					
X FSM Transiti...	pci_wb_slave	c_state (12)				
✓ FSM Tran...	pci_wb_slave	c_state	wishbone_slave_unit....	pci_wb_slave.v	712	FSMTC_81502
✓ FSM Tran...	pci_wb_slave	c_state	wishbone_slave_unit....	pci_wb_slave.v	764	FSMTC_66588
✓ FSM Tran...	pci_wb_slave	c_state	wishbone_slave_unit....	pci_wb_slave.v	794	FSMTC_30972
✓ FSM Tran...	pci_wb_slave	c_state	wishbone_slave_unit....	pci_wb_slave.v	820	FSMTC_70012
X FSM Tran...	pci_wb_slave	c_state	wishbone_slave_unit....	pci_wb_slave.v	757	FSMTC_46428

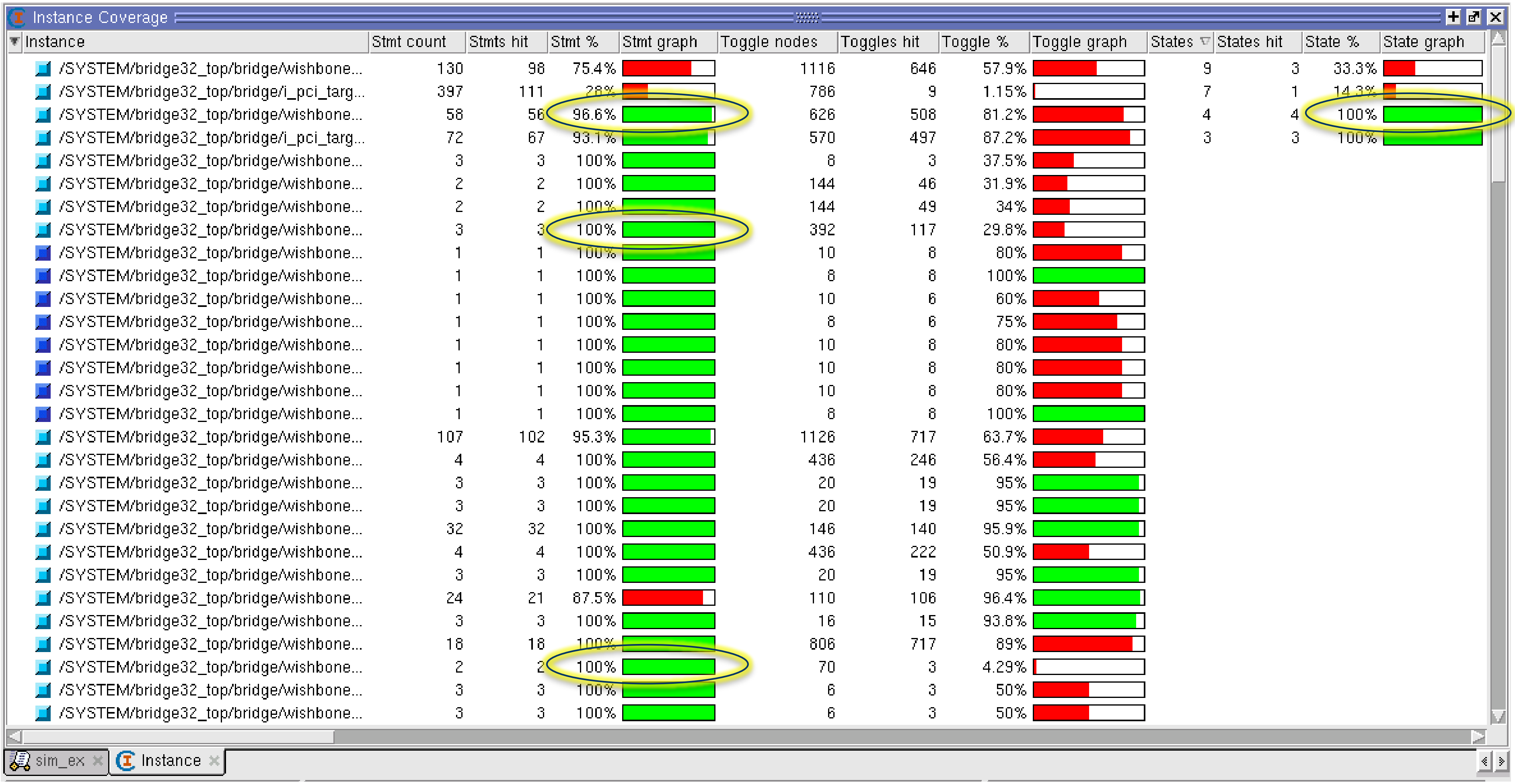
Filters in use: 0

<No Context>

Scaling Unreachable Analysis to the SoC Level



Improved Code Coverage Scores



Instance Coverage

Instance	Stmt count	Stmts hit	Stmt %	Stmt graph	Toggle nodes	Toggles hit	Toggle %	Toggle graph	States	States hit	State %	State graph
/SYSTEM/bridge32_top/bridge/wishbone...	130	98	75.4%	🟥	1116	646	57.9%	🟥	9	3	33.3%	🟥
/SYSTEM/bridge32_top/bridge/i_pci_targ...	397	111	28%	🟥	786	9	1.15%	🟥	7	1	14.3%	🟥
/SYSTEM/bridge32_top/bridge/wishbone...	58	56	96.6%	🟩	626	508	81.2%	🟥	4	4	100%	🟩
/SYSTEM/bridge32_top/bridge/i_pci_targ...	72	67	93.1%	🟩	570	497	87.2%	🟥	3	3	100%	🟩
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	8	3	37.5%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	2	2	100%	🟩	144	46	31.9%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	2	2	100%	🟩	144	49	34%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	392	117	29.8%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	10	8	80%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	8	8	100%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	10	6	60%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	8	6	75%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	10	8	80%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	10	8	80%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	10	8	80%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	1	1	100%	🟩	8	8	100%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	107	102	95.3%	🟩	1126	717	63.7%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	4	4	100%	🟩	436	246	56.4%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	20	19	95%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	20	19	95%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	32	32	100%	🟩	146	140	95.9%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	4	4	100%	🟩	436	222	50.9%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	20	19	95%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	24	21	87.5%	🟥	110	106	96.4%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	16	15	93.8%	🟩				
/SYSTEM/bridge32_top/bridge/wishbone...	18	18	100%	🟩	806	717	89%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	2	2	100%	🟩	70	3	4.29%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	6	3	50%	🟥				
/SYSTEM/bridge32_top/bridge/wishbone...	3	3	100%	🟩	6	3	50%	🟥				

Simulation Coverage Before/After Exclusions

Original Run

```
Coverage Report Summary Data by file

File: ../../pci/rtl/verilog/pci_target32_sm.v
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           98        93         5      94.8
    Branches                        22        21         1      95.4
    FEC Condition Terms              0         0         0     100.0
    FEC Expression Terms           186       127        59      68.2
    FSMs                                                         90.0
        States                       3         3         0     100.0
        Transitions                  5         4         1      80.0
    Toggle Bins                    106       100         6      94.3
```

With exclusions

```
File: ../../pci/rtl/verilog/pci_target32_sm.v
    Enabled Coverage            Active      Hits    Misses % Covered
    ----------------            ------      ----    ------ ---------
    Stmts                           93        93         0     100.0
    Branches                        22        21         1      95.4
    FEC Condition Terms              0         0         0     100.0
    FEC Expression Terms           186       128        58      68.8
    FSMs                                                        100.0
        States                       3         3         0     100.0
        Transitions                  4         4         0     100.0
    Toggle Bins                    106       100         6      94.3
```

CoverGroup Coverage Before/After Exclusion

Original Run

```
Coverage Report Summary


TOTAL COVERGROUP COVERAGE: 28.1%  COVERGROUP TYPES: 4

TOTAL ASSERTION COVERAGE: 80.0%  ASSERTIONS: 5

Total Coverage By File (code coverage only, filtered view): 39.1%
```

With exclusions

```
TOTAL COVERGROUP COVERAGE: 48.2%  COVERGROUP TYPES: 4

TOTAL ASSERTION COVERAGE: 80.0%  ASSERTIONS: 5

Total Coverage By File (code coverage only, filtered view): 42.3%
```

Calculating Your ROI from Using CoverCheck

- Calculating the amount of time saved in your coverage closure flow by using CoverCheck is fairly easy:
  - — N = the number of unreachable coverage elements
  - — T = the time it would have taken you to manually analyze it
  - — ROI = total amount of time saved automating your exclusion flow
  - — **ROI = N x T**

- Example: In one of the above examples there were over 3000 unreachable coverage elements in the design
  - — Let's be generous and estimate it would have taken 15 minutes on average to analyze each unreachable item and exclude it
  - — ROI = 3000 X 15 min
  - — ROI = 45000 min (750 hr)
  - — **ROI = ~4.5 man months of effort saved**

Summary: CoverCheck Benefits

Schedule predictability

- Save project time that would have been spent manually reviewing the coverage holes

Improved metrics

- Automatically eliminate code that’s never meant to be exercised
- Tune measurement to the relevant modes of operation

Elimination of waiver rot

- Manually generated waivers have to be maintained as the code changes

Improved design quality

- Witness waves eliminate danger of ignoring coverage holes that are reachable
- Guides design for verification

Using Questa CoverCheck To Speed Up RTL Freeze of PCIe IP

Sundararajan Haran

Engineering Manager

Microsemi

2015
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
INDIA

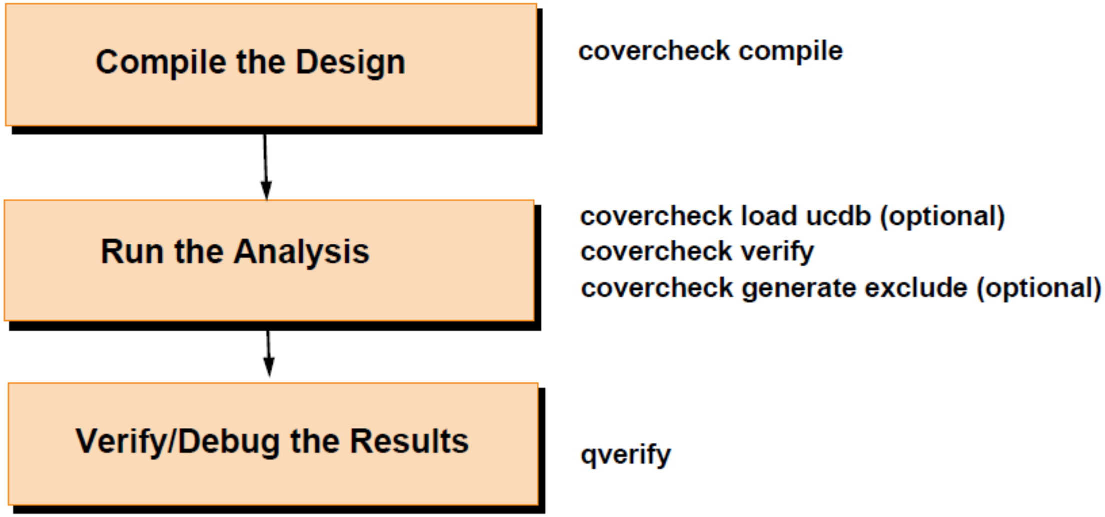
Microsemi

Section Agenda

- Questa Covercheck
- PCIe evaluation bench approach
- Results
- Benefits
- References

Questa CoverCheck Details

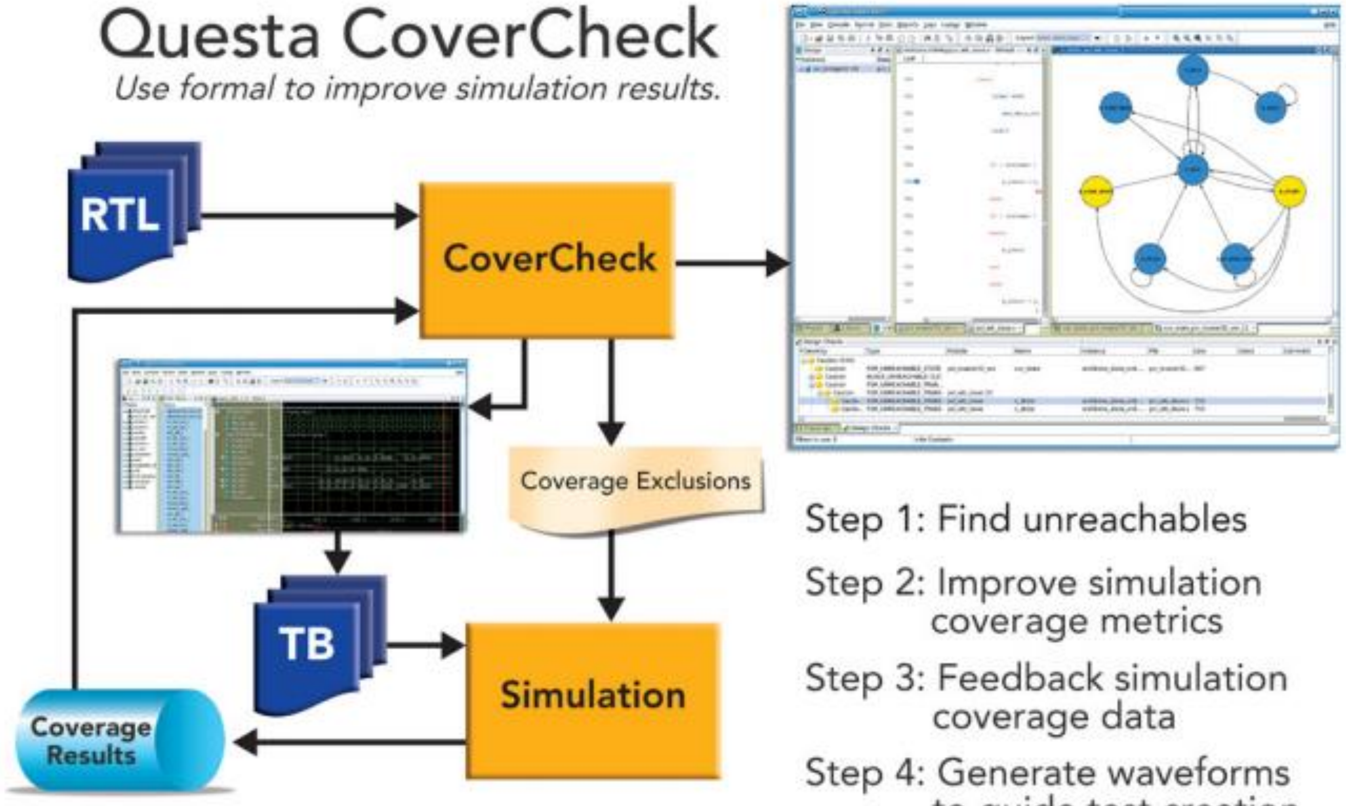
- CoverCheck analyzes coverage items that are found to be:
  - Unreachable through simulation using a QuestaSim Universal Coverage Database (UCDB)
  - Or through a formal analysis
- CoverCheck can run without a simulation UCDB
  - App automatically runs formal analysis on the entire design to determine and analyze the unreachable items
  - Downside: this takes a long time (several hours).



Out-of-the-Box CoverCheck Flow

Questa CoverCheck

Use formal to improve simulation results.



Step 1: Find unreachables

Step 2: Improve simulation coverage metrics

Step 3: Feedback simulation coverage data

Step 4: Generate waveforms to guide test creation

Questa CoverCheck methodology. The tool applies formal methods to target code that's unreached by the simulator.

Section Agenda

- Questa Covercheck
- PCIe evaluation bench approach
- Results
- Benefits
- References

PCIe Evaluation Bench Approach

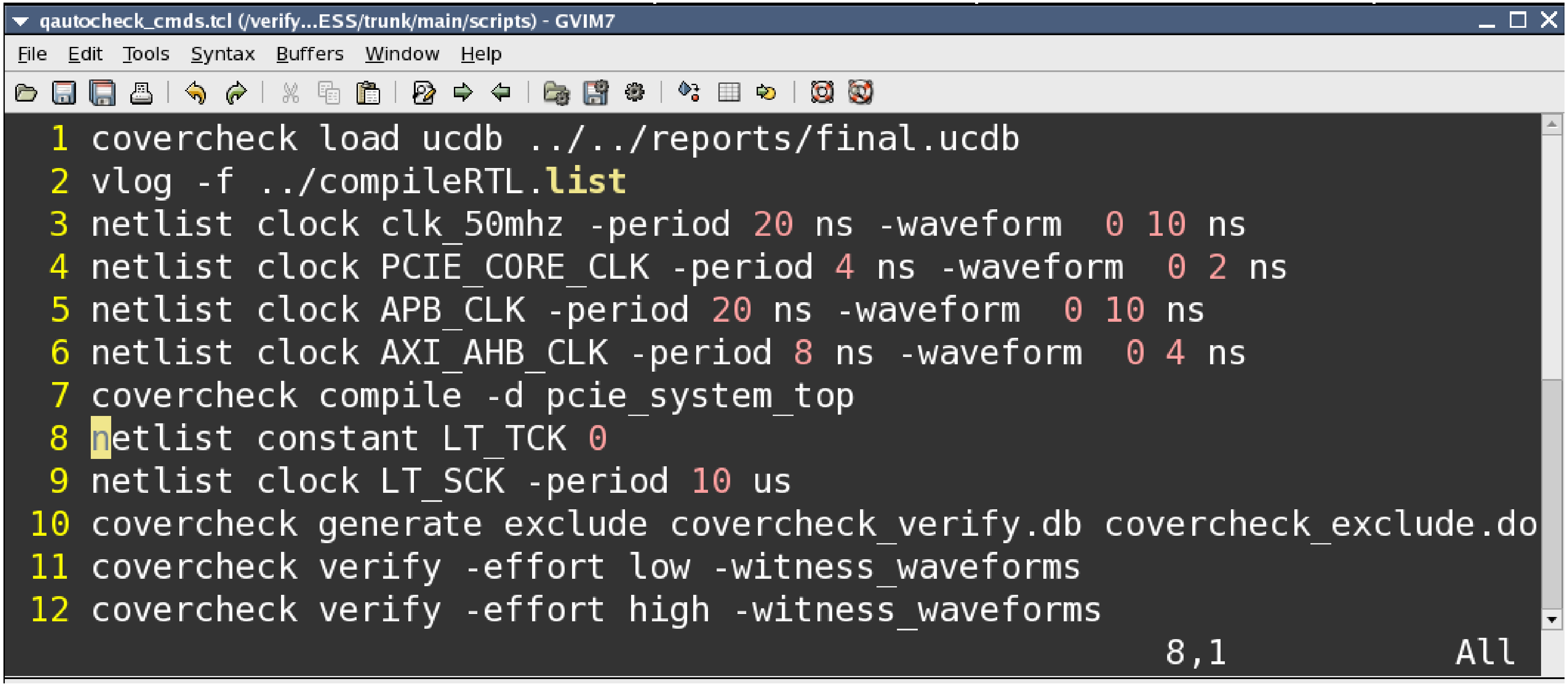
- We used PCIe block level environments for this exercise

- CoverCheck was chosen and used at the block level

Section Agenda

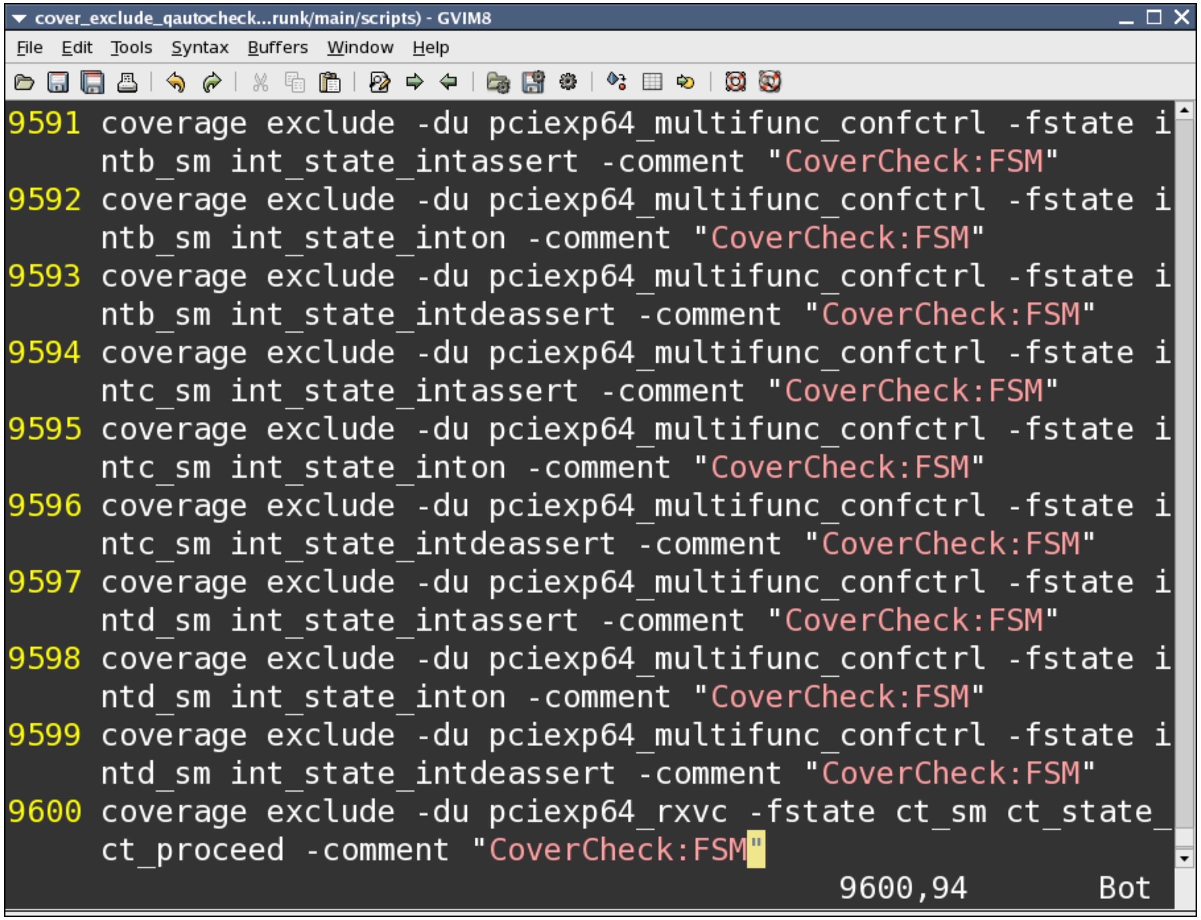
- Questa Covercheck
- PCIe evaluation bench approach
- Results
- Benefits
- References

CoverCheck was run on the final merged coverage database of PCIe block


qautocheck_cmds.tcl (/verify...ESS/trunk/main/scripts) - GVIM7
File
Edit
Tools
Syntax
Buffers
Window
Help
8,1
All

```
covercheck load ucdb ../../reports/final.ucdb
vlog -f ../compileRTL.list
netlist clock clk_50mhz -period 20 ns -waveform  0 10 ns
netlist clock PCIE_CORE_CLK -period 4 ns -waveform  0 2 ns
netlist clock APB_CLK -period 20 ns -waveform  0 10 ns
netlist clock AXI_AHB_CLK -period 8 ns -waveform  0 4 ns
covercheck compile -d pcie_system_top
netlist constant LT_TCK 0
netlist clock LT_SCK -period 10 us
covercheck generate exclude covercheck_verify.db covercheck_exclude.do
covercheck verify -effort low -witness_waveforms
covercheck verify -effort high -witness_waveforms
```

CoverCheck generated the exclusion list after running formal analysis with UCDB and RTL


cover_exclude_qautocheck...runk/main/scripts) - GVIM8
File Edit Tools Syntax Buffers Window Help
9600,94 Bot

```
coverage exclude -du pciexp64_multifunc_confctrl -fstate intb_sm int_state_intassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intb_sm int_state_inton -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intb_sm int_state_intdeassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intc_sm int_state_intassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intc_sm int_state_inton -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intc_sm int_state_intdeassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intd_sm int_state_intassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intd_sm int_state_inton -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_multifunc_confctrl -fstate intd_sm int_state_intdeassert -comment "CoverCheck:FSM"
coverage exclude -du pciexp64_rxvc -fstate ct_sm ct_state_ct_proceed -comment "CoverCheck:FSM"
```

- Generated exclusion list was then reviewed by Design Team for Sign-off.
- Saved approx. 3 weeks which involves (reviewing the coverage database for each uncovered item.)

Section Agenda

- Questa Covercheck
- PCIe evaluation bench approach
- Results
- Benefits
- References

Benefits

The CoverCheck tool saved time in coverage exclusion analysis

- **It only took 3 hours to run**
- **But it saved ~3 weeks of analysis/debug and design team interaction effort!**

Section Agenda

- Questa Covercheck
- PCIe evaluation bench approach
- Results
- Benefits
- References

References

- PCIe block internal specification
- http://www.mentor.com/products/fv/questa-formal/
- Questa CoverCheck User Guide, v10.3a

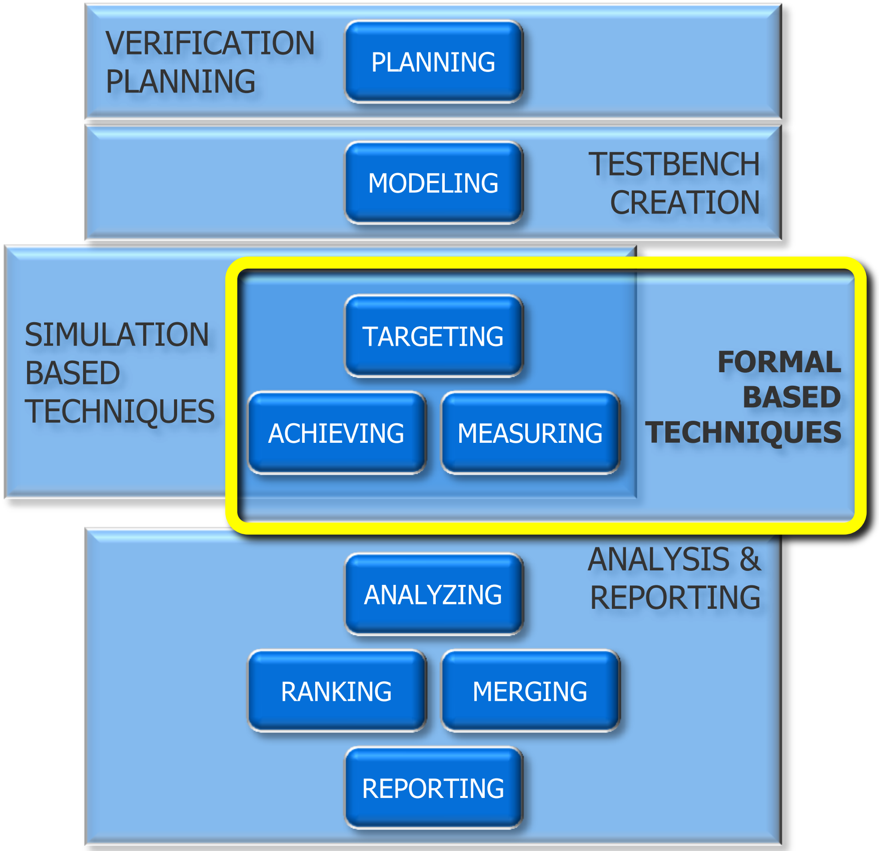
THANK YOU

Agenda

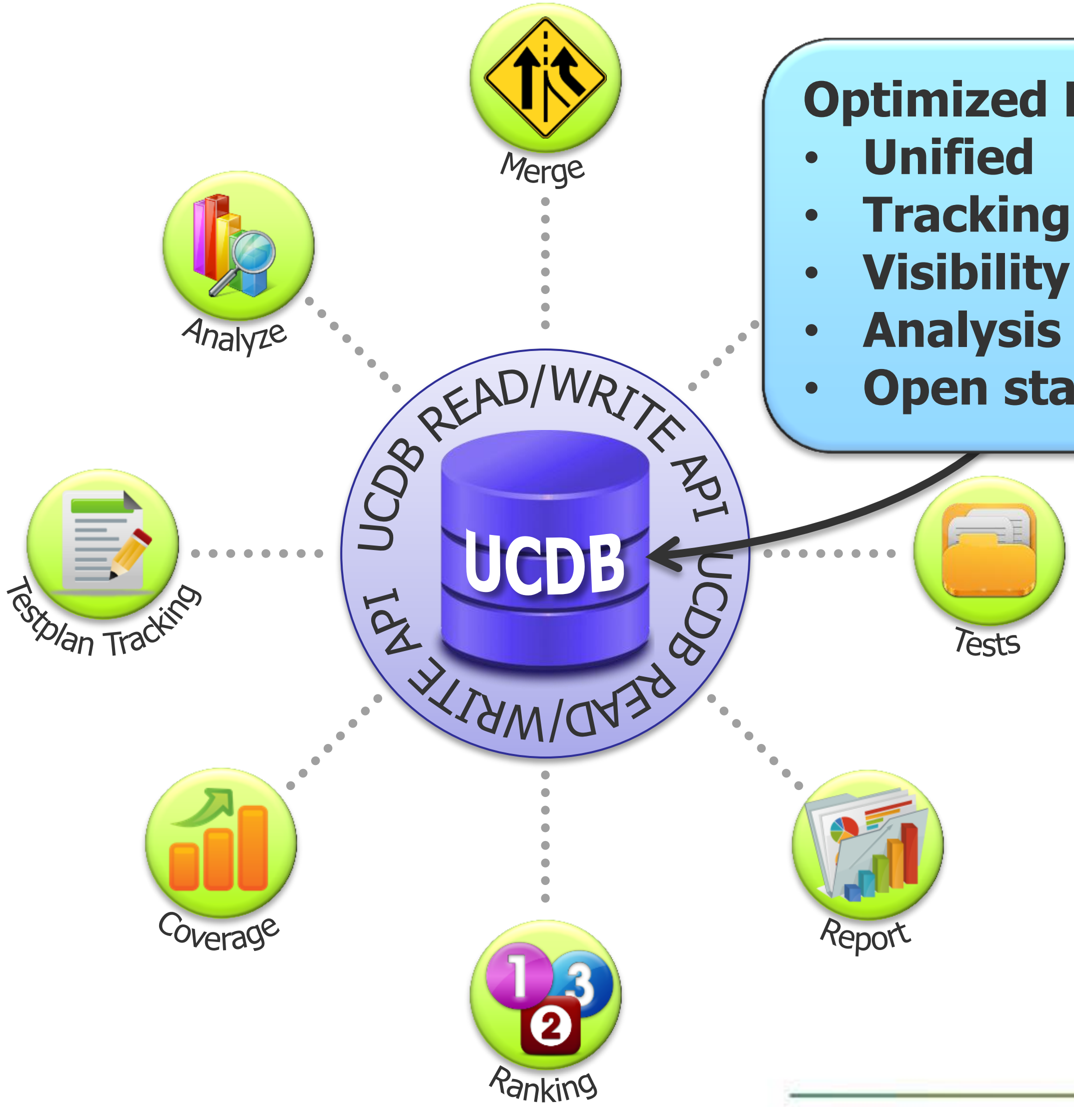
- Introduction
- Coverage Backgrounder
- Targeting Unreachable Coverage with Formal
- Reaching Coverage Closure Faster
- Conclusion

Coverage Closure Process

- Verification Planning
  - Requirements Mapping
  - Coverage Planning
- Testbench Creation
  - Coverage Modeling
  - Stimulus Modeling
  - Verification IP
- Achieving Coverage
  - Regression Management
  - Simulation-Based Techniques
  - **Formal-Based Techniques**
- Analysis & Reporting
  - Analyzing
  - Ranking & Merging
  - Reporting



Coverage Data Management is the Key to Reaching Overall Coverage Closure Faster



Optimized Database

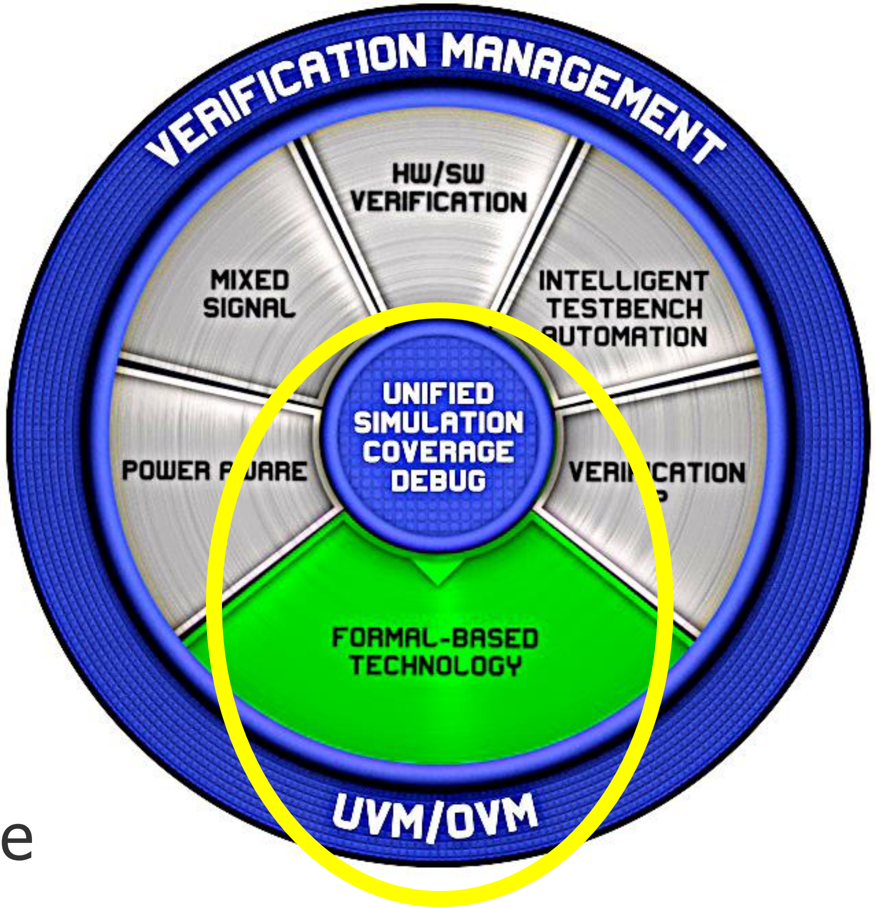
- **Unified**
- **Tracking**
- **Visibility**
- **Analysis**
- **Open standard**

Questa Verification Management

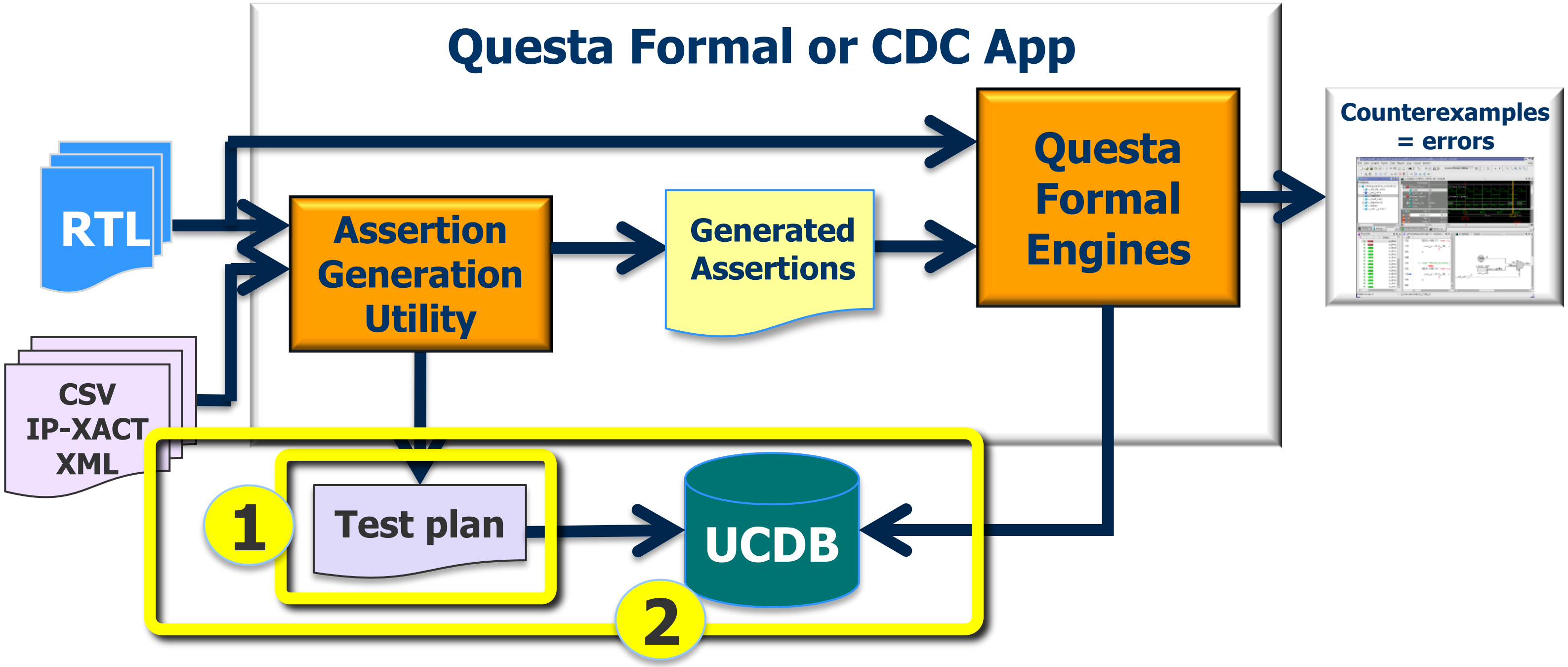
The intersection of Process, Tools and Data

- Built around a high performance Unified Coverage Database
- Electronic Coverage Closure with Testplan Tracking
- Improve Regression time-to-debug with Results Analysis
- *"Are we getting closer to done?"* Trend Analysis
- Improve Regression Productivity with Run Management

➢ Improve Code Coverage Closure with Questa CoverCheck



Automated Test Plan Flow



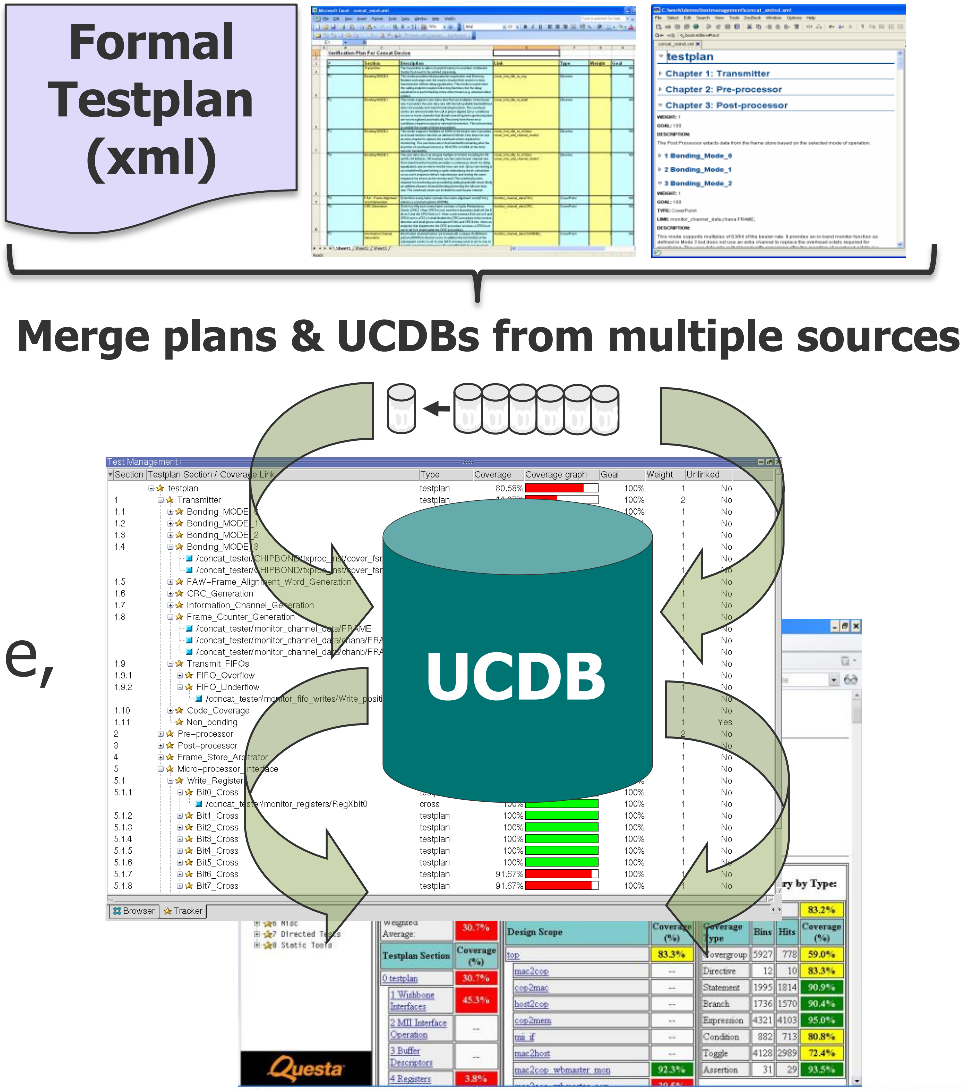
Create a "test plan" from your spec

1. An XML file generated from your CSV/IP-XACT/XML containing test plan entries for all checks & coverage
2. Can be converted to a UCDB and viewed/merged into the Questa Verification Management environment

Formal Integration with Project Testplan

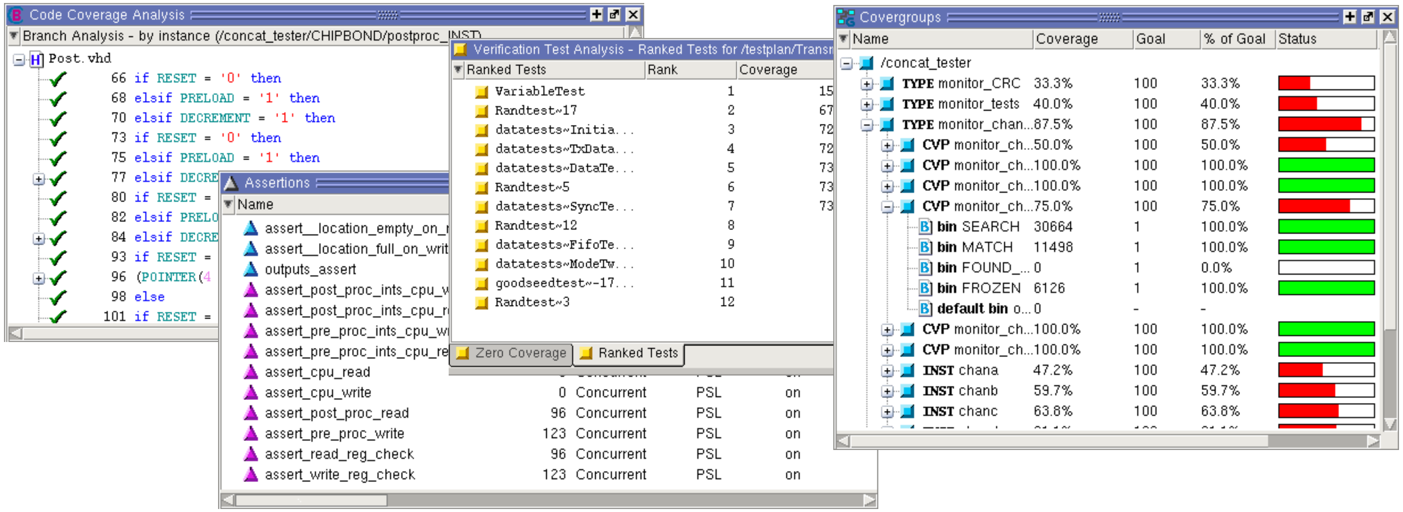
- ✓ Testplan flow provides [multi-tech] management, tracking, & analysis

- ✓ Formal data includes coverage, proofs, and property checks



Manage and Comprehend Volume of Results with Powerful Analysis and Reporting Capabilities

Analyze



Branch Analysis - by instance (/concat_tester/CHIPBOND/postproc_INST)

```
Post.vhd
✓   66 if RESET = '0' then
✓   68 elsif PRELOAD = '1' then
✓   70 elsif DECREMENT = '1' then
✓   73 if RESET = '0' then
✓   75 elsif PRELOAD = '1' then
✓   77 elsif DECRE
✓   80 if RESET =
✓   82 elsif PRELO
✓   84 elsif DECRE
✓   93 if RESET =
✓   96 (POINTER(4
✓   98 else
✓  101 if RESET =
```

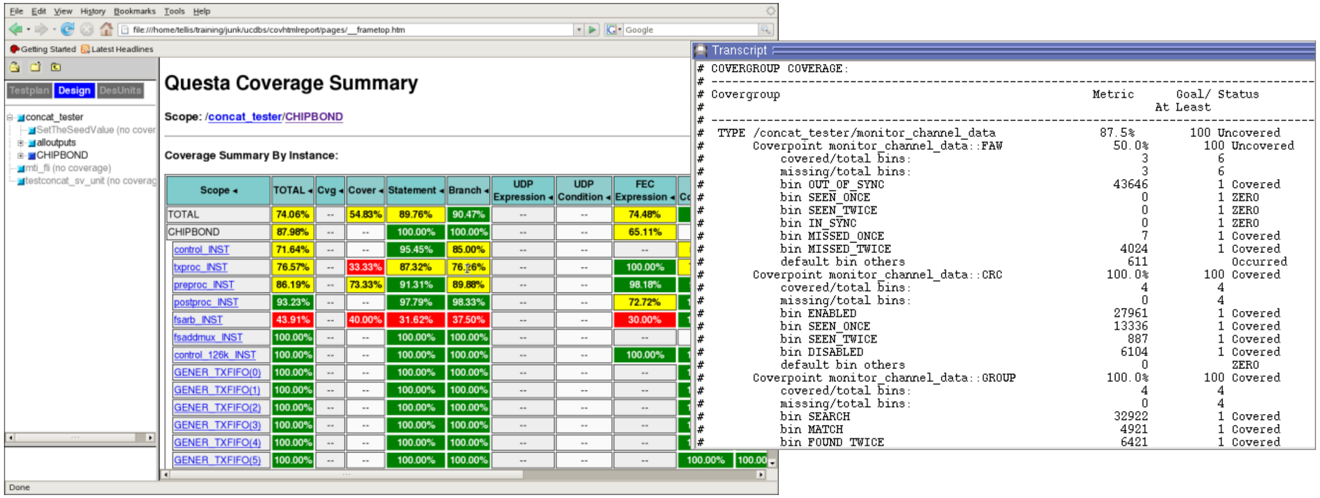
```
Name
assert__location_empty_on_r
assert__location_full_on_writ
outputs_assert
assert_post_proc_ints_cpu_w
assert_post_proc_ints_cpu_r
assert_pre_proc_ints_cpu_wi
assert_pre_proc_ints_cpu_re
assert_cpu_read
assert_cpu_write                0  Concurrent  PSL  on
assert_post_proc_read          96  Concurrent  PSL  on
assert_pre_proc_write         123  Concurrent  PSL  on
assert_read_reg_check          96  Concurrent  PSL  on
assert_write_reg_check        123  Concurrent  PSL  on
```

Verification Test Analysis - Ranked Tests for /testplan/Transm

Ranked Tests	Rank	Coverage
VariableTest	1	15
Randtest~17	2	67
datatests~Initia...	3	72
datatests~TxData...	4	72
datatests~DataTe...	5	73
Randtest~5	6	73
datatests~SyncTe...	7	73
Randtest~12	8	
datatests~FifoTe...	9	
datatests~ModeTw...	10	
goodseedtest~-17...	11	
Randtest~3	12	

Name	Coverage	Goal	% of Goal	Status
/concat_tester				
TYPE monitor_CRC	33.3%	100	33.3%	🟥
TYPE monitor_tests	40.0%	100	40.0%	🟥
TYPE monitor_chan...	87.5%	100	87.5%	🟥
CVP monitor_ch...	50.0%	100	50.0%	🟥
CVP monitor_ch...	100.0%	100	100.0%	🟩
CVP monitor_ch...	100.0%	100	100.0%	🟩
CVP monitor_ch...	75.0%	100	75.0%	🟥
bin SEARCH	30664	1	100.0%	🟩
bin MATCH	11498	1	100.0%	🟩
bin FOUND_...	0	1	0.0%	□
bin FROZEN	6126	1	100.0%	🟩
default bin o...	0	-	-	
CVP monitor_ch...	100.0%	100	100.0%	🟩
CVP monitor_ch...	100.0%	100	100.0%	🟩
INST chana	47.2%	100	47.2%	🟥
INST chanb	59.7%	100	59.7%	🟥
INST chanc	63.8%	100	63.8%	🟥

Report



File Edit View History Bookmarks Tools Help
file:///home/tellis/training/junk/ucdbs/covhtmlreport/pages/__frametop.htm
Google
Getting Started
Latest Headlines
Testplan
Design
DesUnits

```
concat_tester
  SetTheSeedValue (no cover
  alloutputs
  CHIPBOND
mti_fli (no coverage)
testconcat_sv_unit (no coverag
```

Questa Coverage Summary

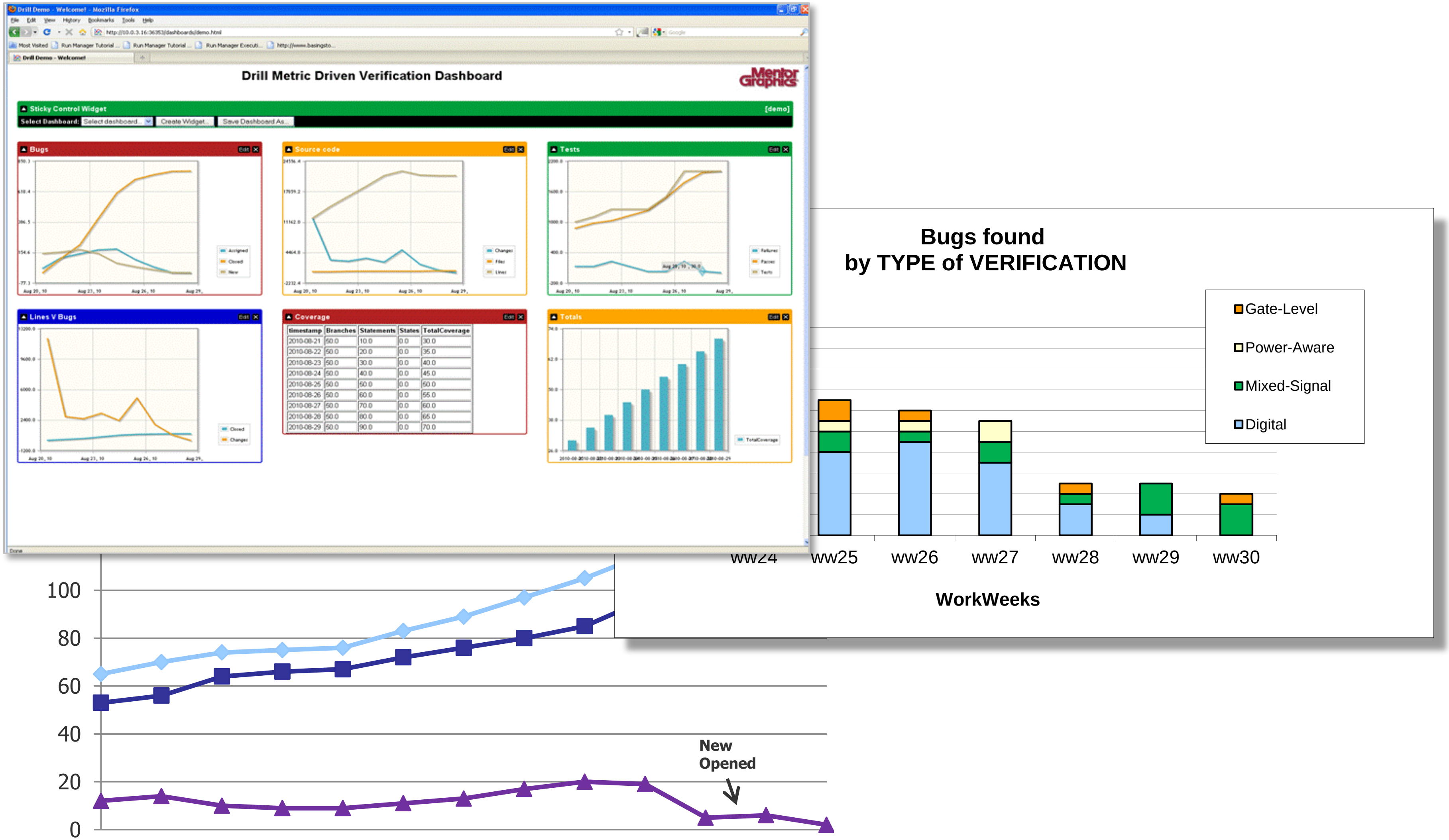
Scope: /concat_tester/CHIPBOND

Coverage Summary By Instance:

Scope	TOTAL	Cvg	Cover	Statement	Branch	UDP Expression	UDP Condition	FEC Expression	Co	
TOTAL	74.06%	--	54.83%	89.76%	90.47%	--	--	74.48%		
CHIPBOND	87.98%	--	--	100.00%	100.00%	--	--	65.11%		
control_INST	71.64%	--	--	95.45%	85.00%	--	--	--		
txproc_INST	76.57%	--	33.33%	87.32%	76.26%	--	--	100.00%		
preproc_INST	86.19%	--	73.33%	91.31%	89.88%	--	--	98.18%		
postproc_INST	93.23%	--	--	97.79%	98.33%	--	--	72.72%		
fsarb_INST	43.91%	--	40.00%	31.62%	37.50%	--	--	30.00%		
fsaddmux_INST	100.00%	--	--	100.00%	100.00%	--	--	--		
control_126k_INST	100.00%	--	--	100.00%	100.00%	--	--	100.00%		
GENER_TXFIFO(0)	100.00%	--	--	100.00%	100.00%	--	--	--		
GENER_TXFIFO(1)	100.00%	--	--	100.00%	100.00%	--	--	--		
GENER_TXFIFO(2)	100.00%	--	--	100.00%	100.00%	--	--	--		
GENER_TXFIFO(3)	100.00%	--	--	100.00%	100.00%	--	--	--		
GENER_TXFIFO(4)	100.00%	--	--	100.00%	100.00%	--	--	--		
GENER_TXFIFO(5)	100.00%	--	--	100.00%	100.00%	--	--	--	100.00%	100.00

```
# COVERGROUP COVERAGE:
# ----------------------------------------------------------------------------------
# Covergroup                                           Metric       Goal/ Status
#                                                                At Least
# ----------------------------------------------------------------------------------
#  TYPE /concat_tester/monitor_channel_data            87.5%        100 Uncovered
#      Coverpoint monitor_channel_data::FAW              50.0%        100 Uncovered
#          covered/total bins:                              3          6
#          missing/total bins:                              3          6
#          bin OUT_OF_SYNC                              43646          1 Covered
#          bin SEEN_ONCE                                    0          1 ZERO
#          bin SEEN_TWICE                                   0          1 ZERO
#          bin IN_SYNC                                      0          1 ZERO
#          bin MISSED_ONCE                                  7          1 Covered
#          bin MISSED_TWICE                              4024          1 Covered
#          default bin others                             611            Occurred
#      Coverpoint monitor_channel_data::CRC             100.0%        100 Covered
#          covered/total bins:                              4          4
#          missing/total bins:                              0          4
#          bin ENABLED                                  27961          1 Covered
#          bin SEEN_ONCE                                13336          1 Covered
#          bin SEEN_TWICE                                 887          1 Covered
#          bin DISABLED                                  6104          1 Covered
#          default bin others                               0            ZERO
#      Coverpoint monitor_channel_data::GROUP           100.0%        100 Covered
#          covered/total bins:                              4          4
#          missing/total bins:                              0          4
#          bin SEARCH                                   32922          1 Covered
#          bin MATCH                                     4921          1 Covered
#          bin FOUND_TWICE                               6421          1 Covered
```

Putting it All Together: Tracking Process and Coverage Metrics



Drill Metric Driven Verification Dashboard

Coverage

timestamp	Branches	Statements	States	TotalCoverage
2010-08-21	50.0	10.0	0.0	30.0
2010-08-22	50.0	20.0	0.0	35.0
2010-08-23	50.0	30.0	0.0	40.0
2010-08-24	50.0	40.0	0.0	45.0
2010-08-25	50.0	50.0	0.0	50.0
2010-08-26	50.0	60.0	0.0	55.0
2010-08-27	50.0	70.0	0.0	60.0
2010-08-28	50.0	80.0	0.0	65.0
2010-08-29	50.0	90.0	0.0	70.0

Bugs found
by TYPE of VERIFICATION

Users' Productivity Gains from Focusing on Coverage Closure & Verification Management

Industry	Sub process	Productivity	Before Questa VM	With Questa VM	Benefits
IP Developer	Nightly regression test time	Throughput	28 hours	→ 2.5 hours	= 9 X faster
	Results and Coverage Analysis	Turn-around	2 hours	→ 20 minutes	= 6 X faster
	Regression file cleanup	Capacity	15 minutes	→ 30 seconds	= 30 X faster
Automotive	Nightly regression test maximum	Throughput	40 tests	→ 320 tests	= 8 X more tests
	Nightly regression test setup time	Turn-around	30 minutes	→ 2 minutes	= 15 X less time
	Nightly regression addition time	Turn-around	60 minutes	→ 5 minutes	= 12 X less time
	Nightly regression Script Files	Turn-around	10 files	→ 1 file	= 10 X easier
	Nightly regression Results Analysis	Turn-around	>1 hour	→ <1 minute	= 60 X faster
Microprocessor	Test Merge Time	Turn-around	7 days	→ 7 hours	= 24 X faster
	Data Storage	Capacity	1 GB	→ 10 MB	= 100X reduction
Wireless	Results Analysis Queries	Turn-around	1 hour	→ 15 minutes	= 4 X faster
Semiconductor	Merge of all coverage from all tests	Turn-around	55 hours	→ 70 minutes	= 47 X faster

Agenda

- Introduction
- Coverage Backgrounder
- Targeting Unreachable Coverage with Formal
- Reaching Coverage Closure Faster
- Conclusion

Conclusion

- "Coverage" in all its forms is an effective way to measure progress, allocate resources, and reach sign-off faster

- The volume of coverage data is exceeding what manual inspection or basic scripting methodologies can handle

- Automated, exhaustive coverage analysis solutions have enabled engineers at companies like MicroSemi (and Microsoft, Micron, Rockwell Automation, Thales, and more) to save 1,000s of hours of compute and R&D time

Resources

- Appendix
  - Coverage closure case studies shared at DVCon USA last March
  - Rockwell Automation, Micron, Microsoft, Thales

- Conference Papers
  - DVCon 2015: "*Coverage Data Exchange Is No Robbery, or Is It?*", MGC
  - ARM Techcon 2014: "*Advanced Verification Management and Coverage Closure Techniques*", Nguyen Le, Microsoft

- Whitepapers
  - "An Automated Code Coverage Closure Solution", http://goo.gl/aZwUpK
  - "Verification Management Eases Those Re-spin Worries", http://goo.gl/J0oNAV

- On Demand Webinars & Courses
  - New School Coverage Closure: http://goo.gl/2JTy7o
  - Verification Academy: CoverCheck – Accelerating Coverage Closure https://verificationacademy.com/sessions/CoverCheck-Accelerating-Coverage-Closure

- Speaker contact info
  - Joe Hupcey III: Joe_Hupcey@mentor.com
  - Nuni Srikanth (a/k/a Shree): Nuni_Srikanth@mentor.com
  - Bhushan Safi: Bhushan_Safi@mentor.com

Mentor Graphics®

www.mentor.com

Appendix

Code Coverage case study at Rockwell Automation

Case Study: Rockwell Automation Analysis of Missed Coverage

- The uncovered condition:

Never hit the following conditions:

- D=1 and F=1
- D=0 and E=1

```
always @ (*) begin
   case (state_ff)
      STATE_x: begin
         if(A)  begin
            if ((B) || (C && !D))  begin
               …
            end else begin
→              if ((!E && !D) || (!F && D)) begin
                  …
               end
            end
         end
      end
   endcase
end
```

Case Study: Rockwell Automation Analysis of Missed Coverage

- The properties:

```
assert property (@(posedge clk) disable iff (!resetn)
    !((state_ff == STATE_X) && (A) && !((B) || (C && !D)) && (D==1) && (F==1)));

assert property (@(posedge clk) disable iff (!resetn)
    !((state_ff == STATE_X) && (A) && !((B) || (C && !D)) && (D==0) && (E==1)));
```

- Formal Results:
  - Assertions fired
  - The given counterexample was illegal protocol

Rockwell Automation Case Study Conclusion

- Use Formal early and often
  - Top level and block level connectivity verification
  - Top level address map verification
  - Complex control logic

- Add Formal analysis for missing coverage
  - Holes always show up late in design cycle

Appendix

Questa CoverCheck Success at Micron

Verification Closure

Bala Chandrasekaran

ASIC Verification Engineer

Micron

DVCon 2015

Mentor Graphics

About Micron SoC Design and Verification

- Micron SoC designs
  - Multi million gate NAND-controller IP blocks designed and verified
- Verification flow
  - Constrained-random, coverage driven approach using UVM
  - Testing at IP block and SoC level
  - Vplan - Requirements tracking
  - Coverage metrics
    - Functional coverage with SV cover groups
    - Assertion coverage with SVA covers
    - Code coverage
- Statement, Branch, Expression, Condition, FSM
- Sign-off requirements
  - All test requirements tracked through to completion
  - 100% functional and code coverage

Micron Case Study: Questa AutoCheck Results

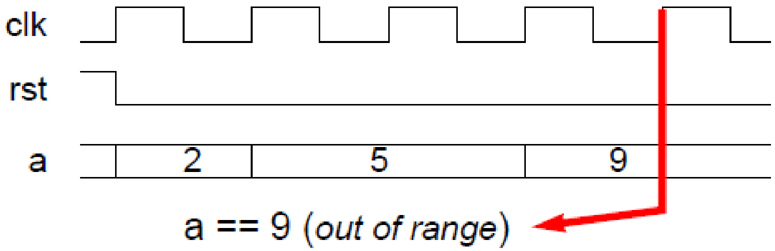
Check	Evaluations	Found	Waived
BLOCK_UNREACHABLE	1353	1	0
FSM_STUCK_BIT	101	1	0
FSM_UNREACHABLE_TRANS	220	2	0
INDEX_ILLEGAL	150	1	0
LOGIC_UNUSED	3038	118	0
X_ASSIGN_REACHABLE	2	1	0
X_UNRESOLVED	54	54	0
AC Total	**4918**	**178**	**0**

Micron Case Study: Violations

- INDEX_ILLEGAL (17 Found)

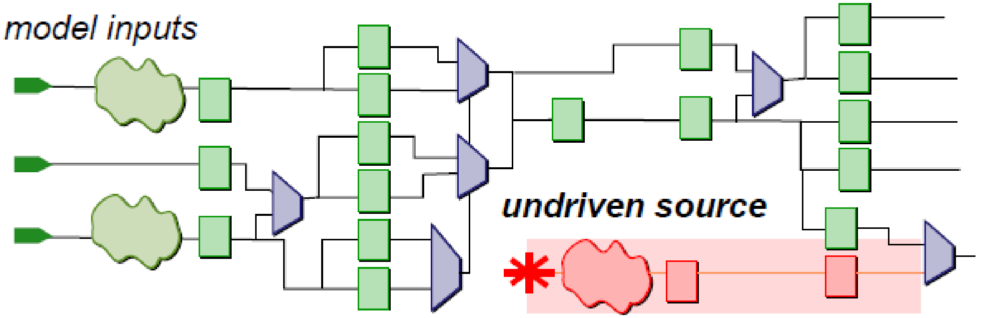
Illegal Array Index: Array index value is illegal.

```
reg [1:8] v;
always @(posedge clk)
   if (rst) v[1] <= b;
   else v[a] <= b;
```



- LOGIC_UNDRIVEN (863 Found)

Logic not Driven from Inputs: Design contains logic that has no driver.



Micron Case Study: Cautions

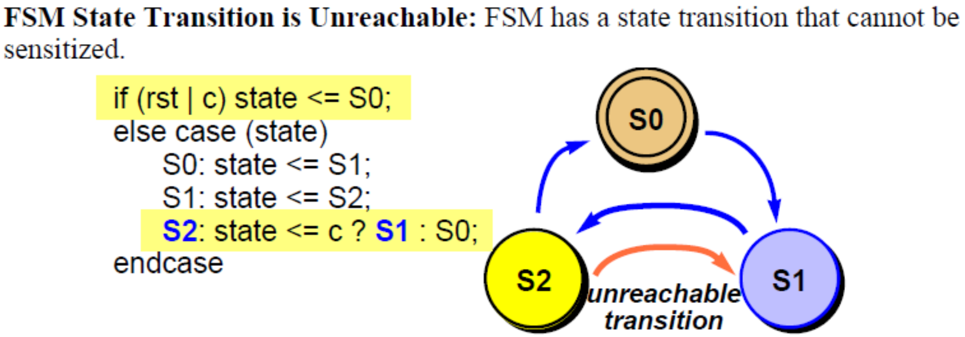
- ASSIGN_IMPLICIT_CONSTANT (65 Found)
  - RHS of an assignment statement includes a non-constant expression, but the statement only assigns a constant value when sensitized.

```
int a, b, var;
. . .
if (a == 0)
      var <= a;
else
      var <= b;
```

- BLOCK_UNREACHABLE (4 Found)
  - Block of code cannot be reached.

```
reg[2:0] a, b, var;
. . .
if (a == 9)
      var <= 0;
else
      var <= b;
```

- FSM_UNREACHABLE_TRANS (1 Found)



FSM State Transition is Unreachable: FSM has a state transition that cannot be sensitized.

```
if (rst | c) state <= S0;
else case (state)
      S0: state <= S1;
      S1: state <= S2;
      S2: state <= c ? S1 : S0;
endcase
```

Micron Case Study: Questa CoverCheck Results

Coverage Type	Active	Unreachable	Reachable	Inconclusives
Branch	731	78	636	17
Condition	135	15	113	7
Expression	483	153	315	15
FSM	34	6	28	0
States	3	1	2	0
Transitions	31	5	26	0
Statement	875	95	768	12
Toggle	0	0	0	0
Coverbin	0	0	0	0
Total	**2258**	**347**	**1860**	**51**

Micron Case Study: What we found

- We got AutoCheck and CoverCheck up & running in 30 min

- We found 347 unreachable items (our prior analysis missed)!

- These were found without constraints
  - If we add a reset/initialization state and constraints we could potentially find even more

- How does this impact schedule?
  - Assuming it takes 15 min to review each item
    - 347 exclusions * 15 minutes = 5,205 minutes (**86.75 h**)

> 2 Man Weeks Saved!

AutoCheck and CoverCheck analyses are now the required Plan of Record.

Appendix

2015 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION UNITED STATES

Questa CoverCheck Success at Microsoft

Verification Closure

Nguyen Le, Microsoft

DVCon 2015

Author Information

- Nguyen Le
  - Principal Design Verification Engineer
  - Microsoft Corp.
  - Email: ngle@microsoft.com

About Microsoft SoC Design and Verification

- IEB SoC designs
  - Multi million gate internal IP blocks designed and verified

- Verification flow
  - Constrained-random, coverage driven approach using UVM
  - Testing at IP block and SoC level
  - Testplan requirements tracking
  - Coverage metrics
    - Functional coverage with SV covergroups
    - Assertion coverage with SVA covers
    - Code coverage

      Statement, Branch, Expression, Condition, FSM

- Sign-off requirements
  - All test requirements tracked through to completion
  - 100% functional, assertion and code coverage

CoverCheck Case Study Results

- Exclusions improved code coverage by 10 – 15% in most blocks
  - Coverage number improved from 87% to 97%

- In auto-generated code for register blocks the improvement was 20%
  - There are simulation hooks that are unreachable

Benefits of Formal Code Exclusion

- Improved code coverage metrics
  - Metrics are automatically tuned to the relevant modes of operation for reused IP blocks

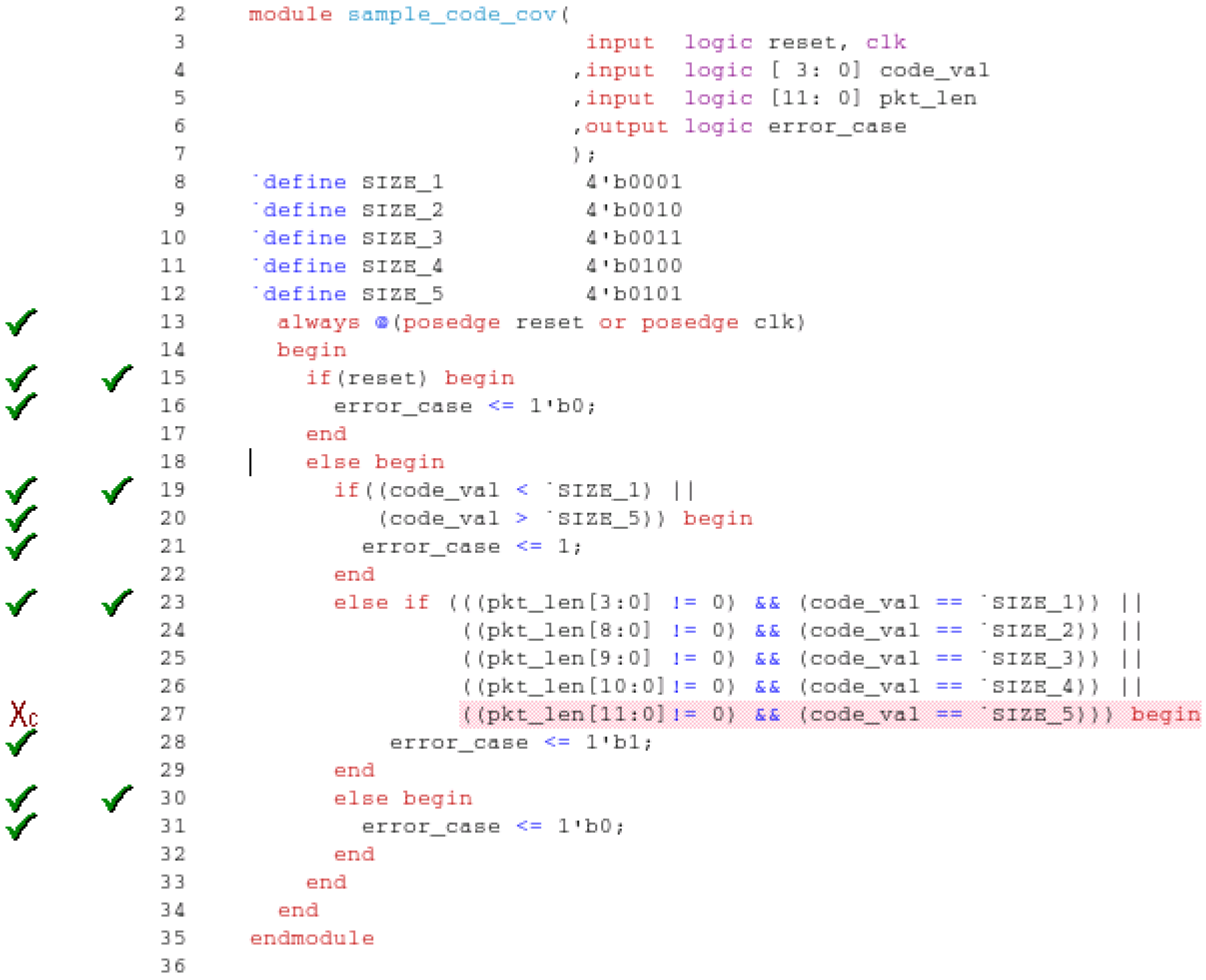
- Improved design quality
  - Exclusions are formally proven reducing the risk of ignoring important goals

- Case study ROI
  - Time to manually write exclusions vs. auto-generate

    (1 Design Engineer + 1 Verification Engineer) x 10 min/exclusion

    = 4 man months saved

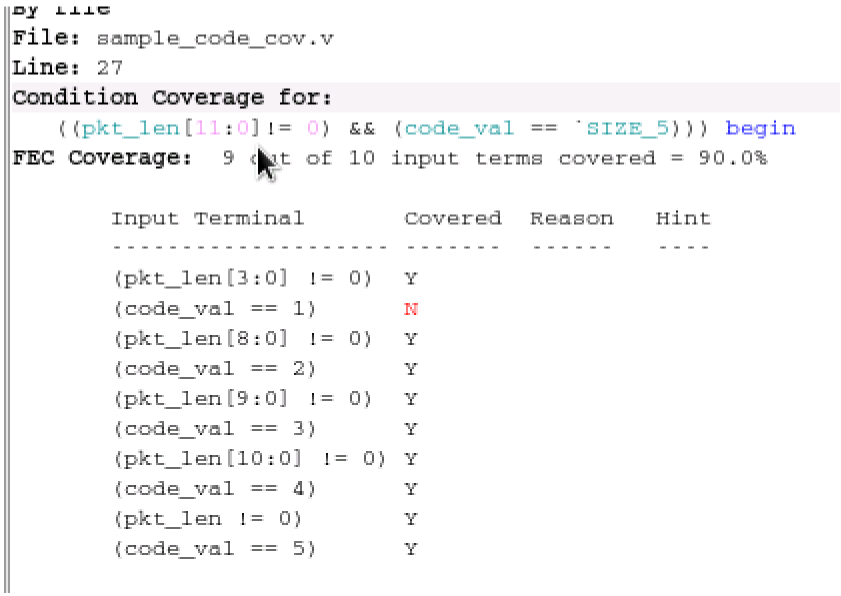
Example: Two Days to Manually Exclude



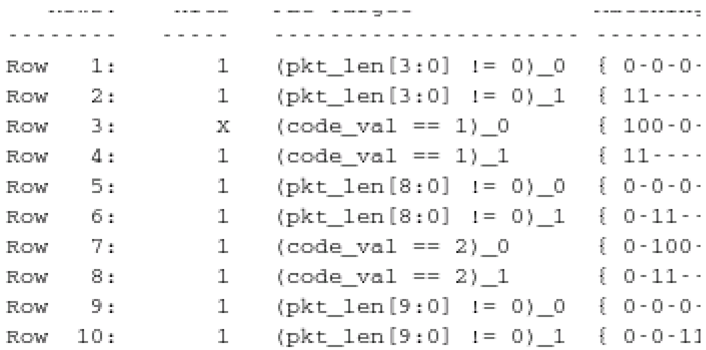
```
      module sample_code_cov(
                              input  logic reset, clk
                             ,input  logic [ 3: 0] code_val
                             ,input  logic [11: 0] pkt_len
                             ,output logic error_case
                             );
      `define SIZE_1          4'b0001
      `define SIZE_2          4'b0010
      `define SIZE_3          4'b0011
      `define SIZE_4          4'b0100
      `define SIZE_5          4'b0101
✓       always @(posedge reset or posedge clk)
        begin
✓  ✓      if(reset) begin
✓           error_case <= 1'b0;
          end
          else begin
✓  ✓        if((code_val < `SIZE_1) ||
✓              (code_val > `SIZE_5)) begin
✓             error_case <= 1;
            end
✓  ✓        else if (((pkt_len[3:0] != 0) && (code_val == `SIZE_1)) ||
                     ((pkt_len[8:0] != 0) && (code_val == `SIZE_2)) ||
                     ((pkt_len[9:0] != 0) && (code_val == `SIZE_3)) ||
                     ((pkt_len[10:0]!= 0) && (code_val == `SIZE_4)) ||
Xc                   ((pkt_len[11:0]!= 0) && (code_val == `SIZE_5))) begin
✓               error_case <= 1'b1;
            end
✓  ✓        else begin
✓             error_case <= 1'b0;
            end
          end
        end
      endmodule
```

Example: Detail Coverage

- After extracting this snippet of code and run 64k cases (exhaustive), we are convinced of the exclusions from CoverCheck



```
By file
File: sample_code_cov.v
Line: 27
Condition Coverage for:
    ((pkt_len[11:0]!= 0) && (code_val == `SIZE_5))) begin
FEC Coverage:  9 out of 10 input terms covered = 90.0%

        Input Terminal       Covered  Reason   Hint
        -------------------- -------  ------   ----
        (pkt_len[3:0]  != 0)  Y
        (code_val  == 1)      N
        (pkt_len[8:0]  != 0)  Y
        (code_val  == 2)      Y
        (pkt_len[9:0]  != 0)  Y
        (code_val  == 3)      Y
        (pkt_len[10:0]  != 0) Y
        (code_val  == 4)      Y
        (pkt_len  != 0)       Y
        (code_val  == 5)      Y
```



```
--------   -----   ---------------------- --------
Row   1:       1   (pkt_len[3:0]  != 0)_0  { 0-0-0-
Row   2:       1   (pkt_len[3:0]  != 0)_1  { 11----
Row   3:       X   (code_val  == 1)_0      { 100-0-
Row   4:       1   (code_val  == 1)_1      { 11----
Row   5:       1   (pkt_len[8:0]  != 0)_0  { 0-0-0-
Row   6:       1   (pkt_len[8:0]  != 0)_1  { 0-11--
Row   7:       1   (code_val  == 2)_0      { 0-100-
Row   8:       1   (code_val  == 2)_1      { 0-11--
Row   9:       1   (pkt_len[9:0]  != 0)_0  { 0-0-0-
Row  10:       1   (pkt_len[9:0]  != 0)_1  { 0-0-11
```

- Unreachable code_val == 1 never false

Microsoft Case Study Conclusions

- Verification of complex SoC projects is a always more difficult to manage than expected

- Time saved by automatic code coverage closure is easily an order of magnitude

- Wish list
  - There are still complex FECs that the tool would give up
    - We are hoping for more complex expression to be handled
  - Or possible RTL recoding suggestion that can help the tool

Appendix

Questa CoverCheck Success at Thales

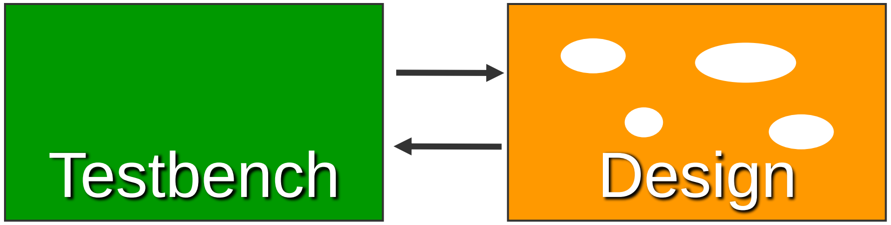
Questa CoverCheck Testimonial

Christian Bara

The coverage challenge

- Coverage driven verification well adopted by industry now
  - To measure that every lines of design code have been exercised
  - But difficult to reach 100 % coverage
    - Insufficient or incorrect input stimulus during simulation
    - Unreachable coverage items

      because of bugs , particular statements or configurations
  - Need to identify manually unreachable parts
    - Could be a painful task
  - Add extra tests to cover not reached items



This leads to a lot of effort to reach the coverage closure

Mentor solution : Questa Covercheck

- A formal tool
  - Automating the debug process of coverage closure
    - Inputs

      rtl design & ucdb simulation results (not mandatory) to focus analysis only on code items not reached by simulation
    - Outputs

      reports of proved unreachable code

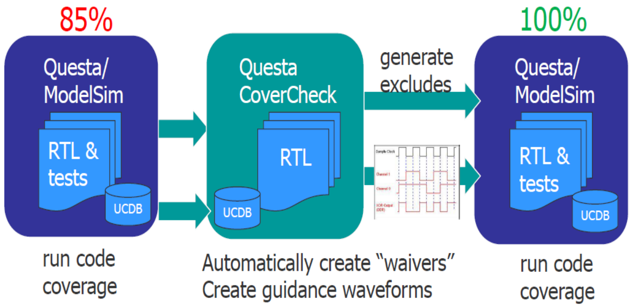
      bugs or conditions making the code unreachable

      exclusions files

      for truly unreachable code

      guidance waveforms

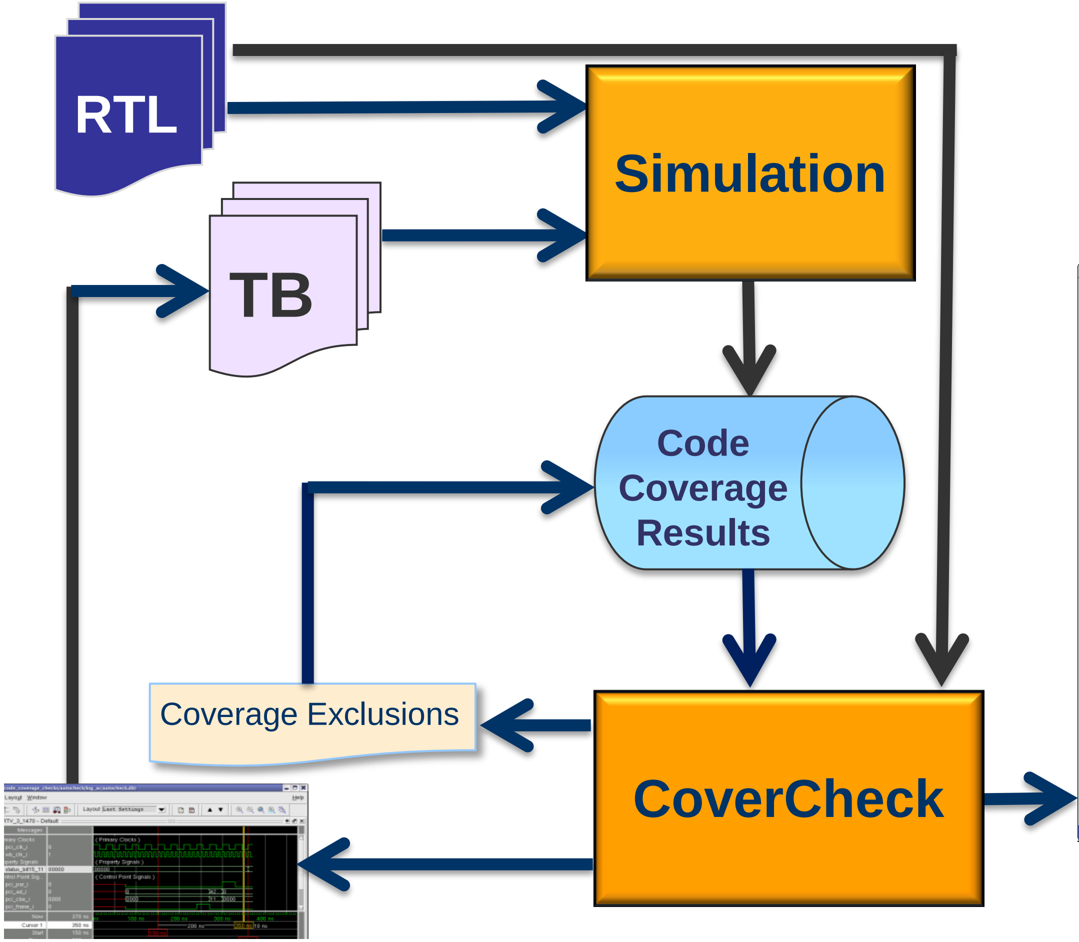
      for code that could be reac

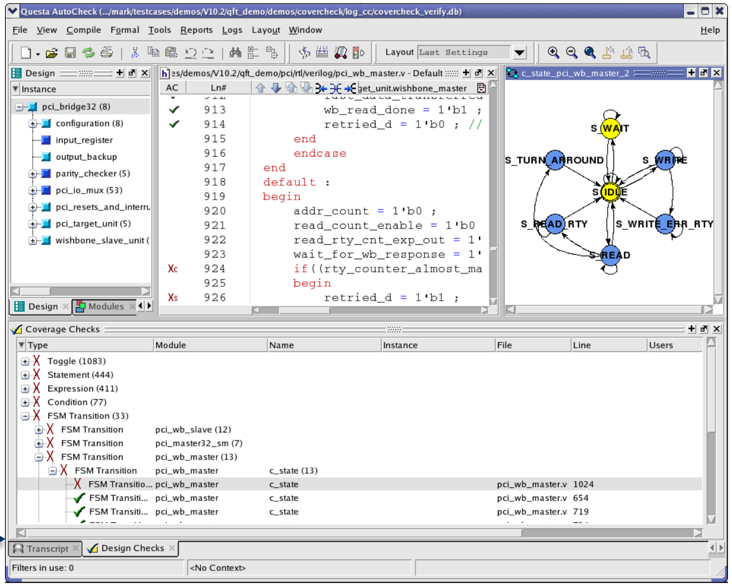


Points on causes of unreachables and help cover not yet reached code

Overview of Covercheck flow

— Use formal static analysis to improve code coverage results





```
            wb_read_done = 1'b1 ;
            retried_d = 1'b0 ; //
        end
        endcase
    end
    default :
    begin
        addr_count = 1'b0 ;
        read_count_enable = 1'b0
        read_rty_cnt_exp_out = 1'
        wait_for_wb_response = 1'
        if((rty_counter_almost_ma
        begin
            retried_d = 1'b1 ;
```

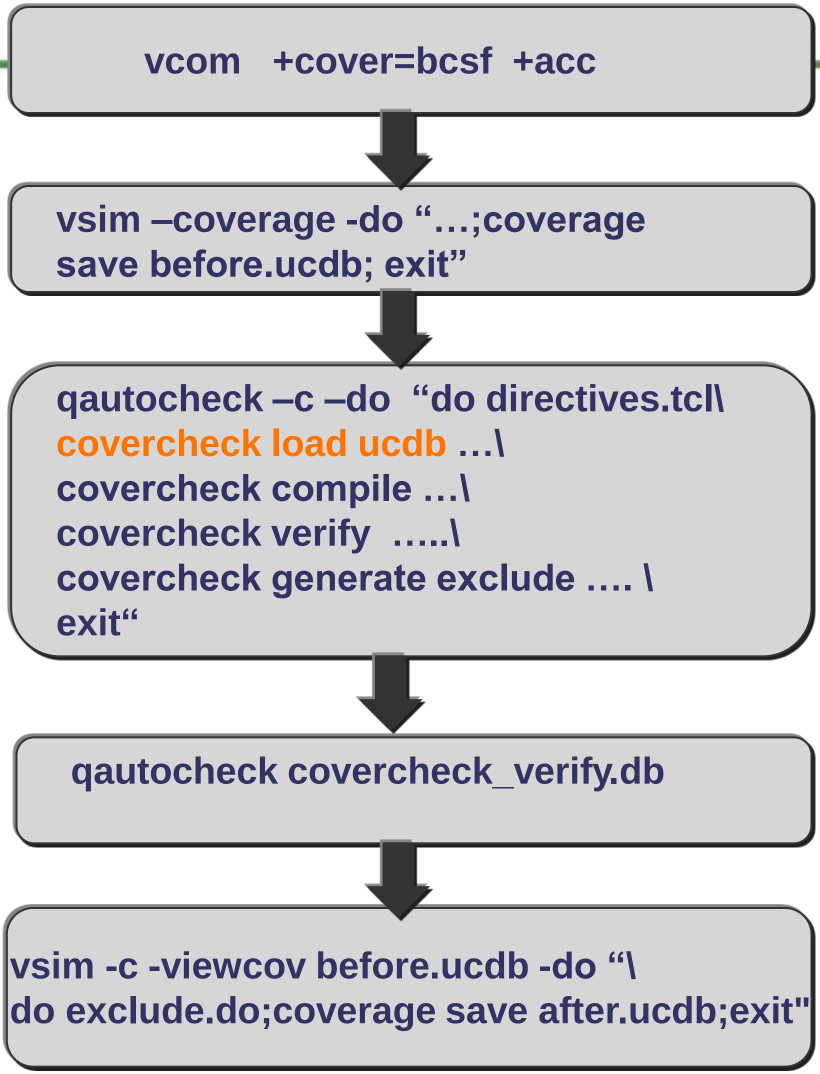
Coverage Checks

Type	Module	Name	Instance	File	Line	Users
Toggle (1083)						
Statement (444)						
Expression (411)						
Condition (77)						
FSM Transition (33)						
FSM Transition	pci_wb_slave (12)					
FSM Transition	pci_master32_sm (7)					
FSM Transition	pci_wb_master (13)					
FSM Transition	pci_wb_master	c_state (13)				
FSM Transitio...	pci_wb_master	c_state		pci_wb_master.v	1024	
FSM Transiti...	pci_wb_master	c_state		pci_wb_master.v	654	
FSM Transiti...	pci_wb_master	c_state		pci_wb_master.v	719	

Filters in use: 0
<No Context>

Can be used without simulation results

Covercheck implementation flow



```
vcom   +cover=bcsf  +acc
```

Phase 1 : compile the design

```
vsim –coverage -do “…;coverage
save before.ucdb; exit”
```

Phase 2 : simulate the design

```
qautocheck –c –do  “do directives.tcl\
covercheck load ucdb …\
covercheck compile …\
covercheck verify  …..\
covercheck generate exclude …. \
exit“
```

Phase 3 : run covercheck verification

```
qautocheck covercheck_verify.db
```

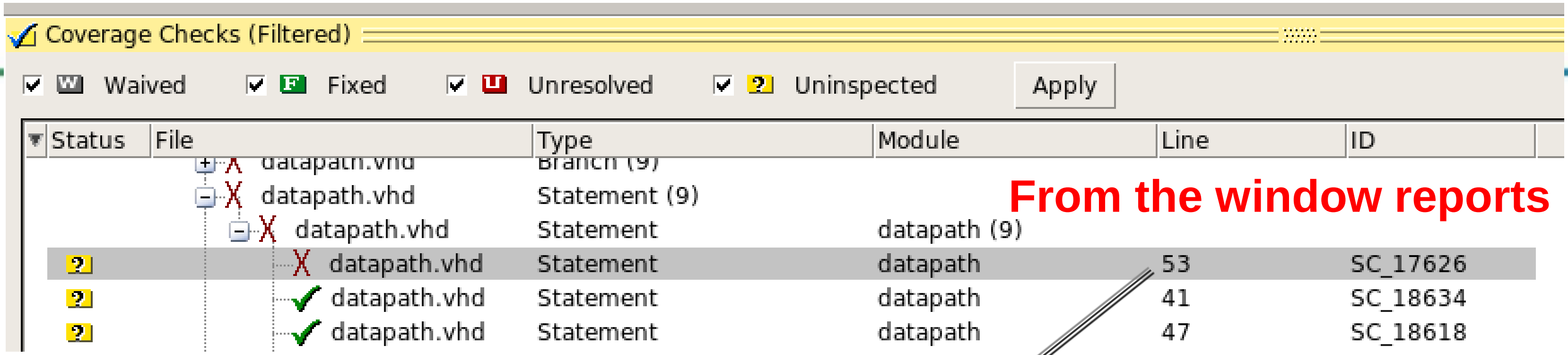
Phase 4 : analyse & debug the results

```
vsim -c -viewcov before.ucdb -do “\
do exclude.do;coverage save after.ucdb;exit"
```

Phase 5 : generate new coverage results

Every steps is tcl scriptable

Find root cause of unreachable code

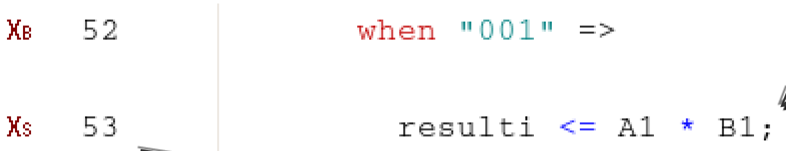


Coverage Checks (Filtered)

☑ W Waived ☑ F Fixed ☑ U Unresolved ☑ ? Uninspected [Apply]

Status	File	Type	Module	Line	ID
	☒ datapath.vhd	Branch (9)			
	☒ datapath.vhd	Statement (9)			
	☒ datapath.vhd	Statement	datapath (9)		
?	☒ datapath.vhd	Statement	datapath	53	SC_17626
?	✓ datapath.vhd	Statement	datapath	41	SC_18634
?	✓ datapath.vhd	Statement	datapath	47	SC_18618

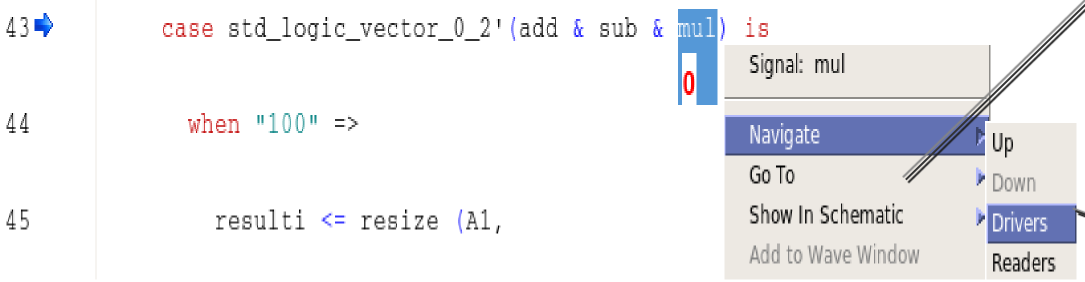
From the window reports



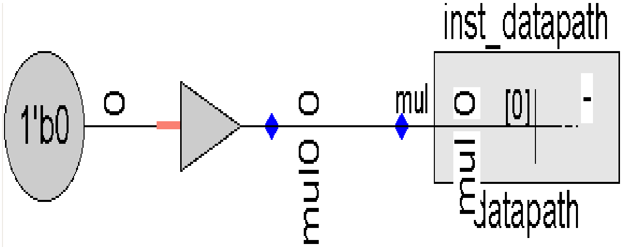
```
XB   52        when "001" =>
XS   53          resulti <= A1 * B1;
```

Show source

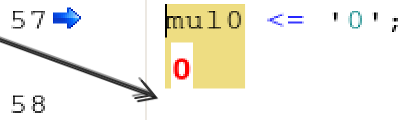
Understand Why



```
43      case std_logic_vector_0_2'(add & sub & mul) is
                                                 0
44        when "100" =>
45          resulti <= resize (A1,
```



Show root cause

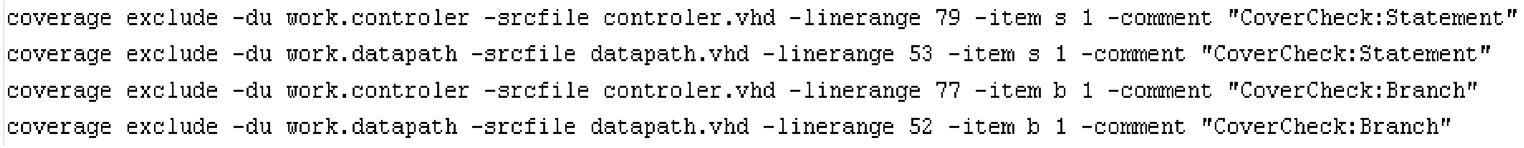


```
57   mul0 <= '0';
     0
58
```

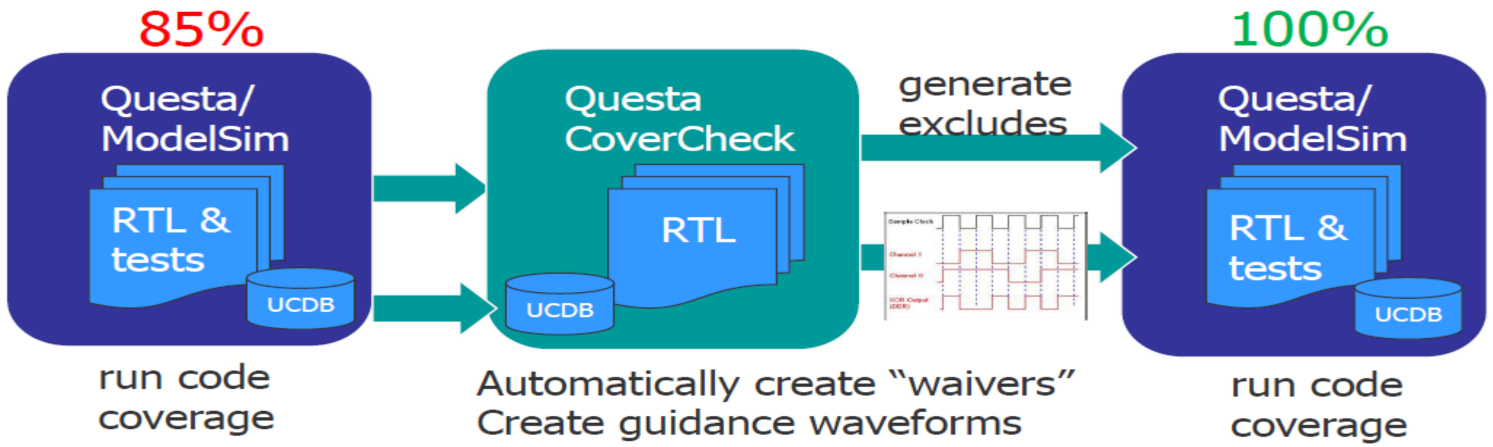
Easy to pinpoint the root cause of an unreachable code and analyse if it's a bug that need to be fixed or not

The generated exclude file

- vsim –c –viewcov before.ucdb –do “do exclude.do coverage save after.do”

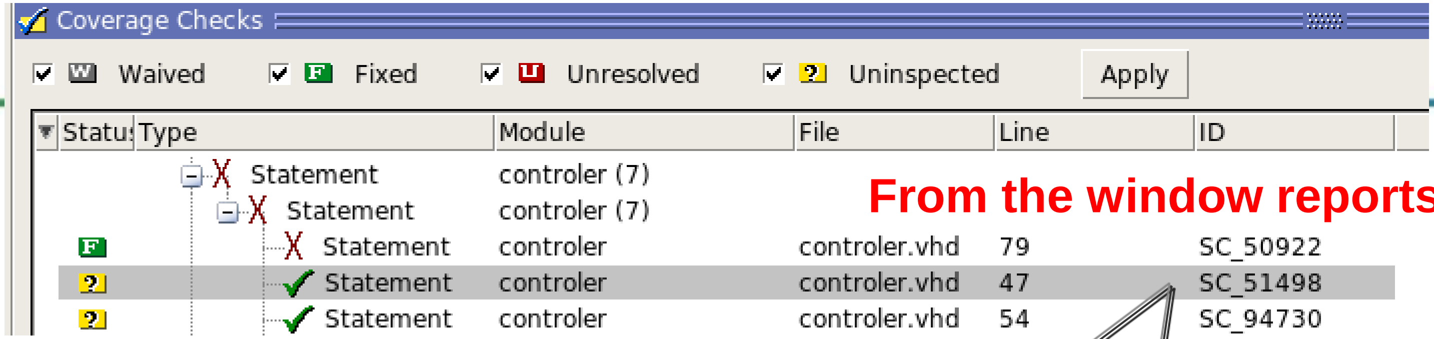


```
coverage exclude -du work.controler -srcfile controler.vhd -linerange 79 -item s 1 -comment "CoverCheck:Statement"
coverage exclude -du work.datapath -srcfile datapath.vhd -linerange 53 -item s 1 -comment "CoverCheck:Statement"
coverage exclude -du work.controler -srcfile controler.vhd -linerange 77 -item b 1 -comment "CoverCheck:Branch"
coverage exclude -du work.datapath -srcfile datapath.vhd -linerange 52 -item b 1 -comment "CoverCheck:Branch"
```



less effort to improve code coverage results to target 100%

Show guidance waveforms : waivers



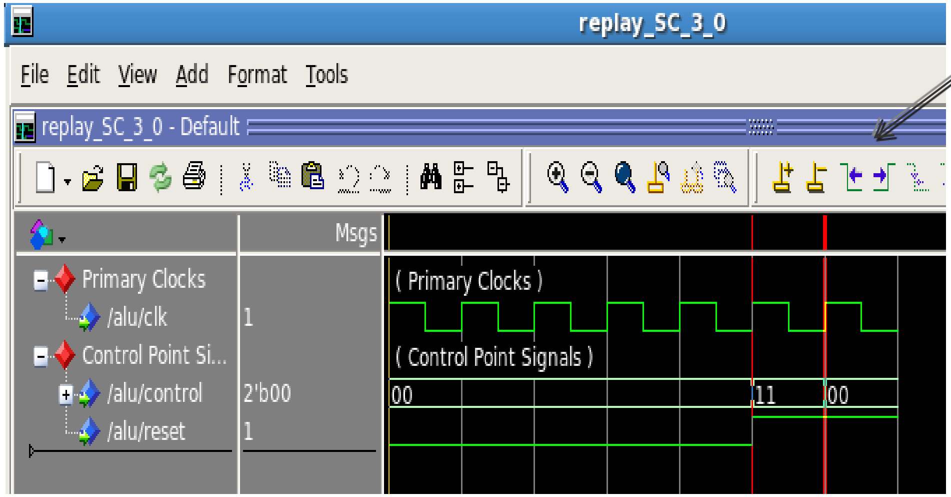
Coverage Checks

☑ W Waived ☑ F Fixed ☑ U Unresolved ☑ ? Uninspected Apply

Statu:	Type	Module	File	Line	ID
	⊟ ✗ Statement	controler (7)			
	⊟ ✗ Statement	controler (7)			
F	✗ Statement	controler	controler.vhd	79	SC_50922
?	✓ Statement	controler	controler.vhd	47	SC_51498
?	✓ Statement	controler	controler.vhd	54	SC_94730

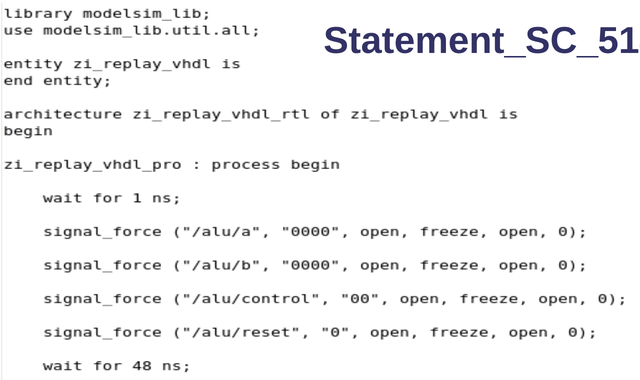
From the window reports

Show how to reach statement



Create a testbench

Statement_SC_51498.vhd



```
library modelsim_lib;
use modelsim_lib.util.all;

entity zi_replay_vhdl is
end entity;

architecture zi_replay_vhdl_rtl of zi_replay_vhdl is
begin

zi_replay_vhdl_pro : process begin

    wait for 1 ns;

    signal_force ("/alu/a", "0000", open, freeze, open, 0);

    signal_force ("/alu/b", "0000", open, freeze, open, 0);

    signal_force ("/alu/control", "00", open, freeze, open, 0);

    signal_force ("/alu/reset", "0", open, freeze, open, 0);

    wait for 48 ns;
```

Help to write a directed test or adjust constraints for a constrained random testbench

The evaluation (1)

- Machine
  - 32 cores @ 2.7 GHZ , 128 GB RAM , Linux RedHat 5U7 64 bits
- Questa CoverCheck 10.2b
- Design characteristics
  - 60 klines of vhdl code
  - implemented within a XILINX KINTEX7 device (xc7k325)
    - Slices 30k , DSP 178 , Bram 374

A real design

The Results

- Without UCDB file (branch condition statement verification)

	verify effort low 30 min	verify effort high 7 hours
Actives	52918	52918
Unreachables	**3310**	**3323**
Reachables	**23798**	**27809**
Inconclusives	25810	21786

- Using an UCDB file
  - Runtime 20 min with a verification effort to low

Passing low to high effort increases drastically the runtime verification (x 14) for a small decrease in inconclusives (15%)

Conclusion

- Easy to use tool
  - User guide , Tutorial , good support
  - Intuitive debug user interface
- Questa Covercheck brings benefits
  - Reducing time trying to hit truly unreachables code
  - Helping find stimulus to improve code coverage
  - Facilitating process review for justifying unreached code items
    - Particularly for code that does not matter
  - Eliminating manual errors for creating exclusions files
  - Easing maintenance of exclusions files as the design volves
- Next Steps
  - Run with higher effort levels to reduce inconclusives
  - Run on more designs