Shift-Left: Complementing Simulation with VC Formal™

Srinivasan Venkataramanan,
L B Om Prakash, Balamurali N S, Mohammed Asad Rizvi

Infinera India

June 26-27, 2019
SNUG India – Bangalore

Agenda

Introduction

Design Structures

Deploying VC Formal

Results

Conclusion

Introduction

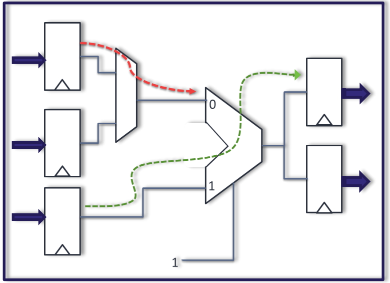
- Formal Verification:
  - Now mainstream
  - Augments Simulation based techniques very well
  - On select-tasks, provides shift-left in the timelines
- Infinera:
  - Vision: “Enable an Infinite Network that can provide unlimited services to everyone, everywhere, instantly.”
  - Designs some of the world’s most complex optical networking chips
  - Complex IPs, aggregated to sub-chip, full-chip, board, Software etc.
  - Design & Verification of IPs, sub-chip
- Team has been using VCS + UVM for many years
  - With custom Base Class Library underpinning standard UVM
  - Deployed VC Formal on a recent chip, with selected design structures as targets

Design Structures chosen for Formal

- Success of Formal – depends on sort of blocks/problems chosen
  - Unlike Simulation that's well-understood
- Following design structures chosen in our project:
  - Debug Structures in IPs
  - Mesh structures for re-ordering
  - PathDelays at sub-system
- Formal "apps" are a great starting point:
  - Quick and easy to start deploying
  - Does not require "Formal specialists"
  - Proven to deliver results across design styles
  - Leads to FPV (Formal Property Verification)
- Also using:
  - Coverage Unreachability (FCA)
  - FPV – Formal Property Verification

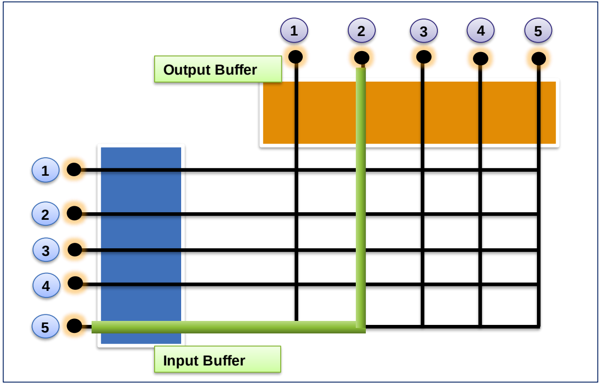
Debug Structures in IPs

- IPs get reused across generations of chips
- Recently added debug feature
  - Tracking health parameters
  - Intent - ease the process of debugging post-silicon
- Added during later stage of Design process
  - Each IP is verified rigorously using UVM + Coverage metrics
  - Closure of functional verification was at its final stage
  - Big impact to include DebugBus verification in UVM flow



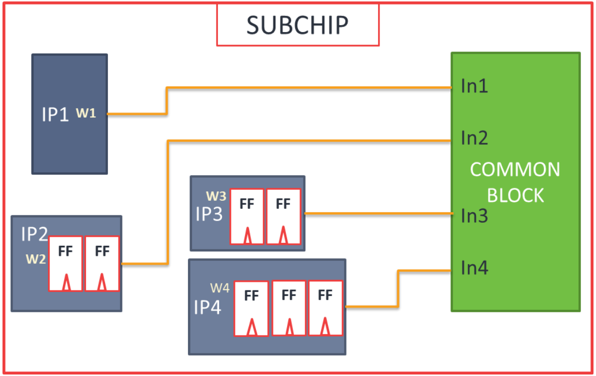
Mesh structures for lane re-ordering

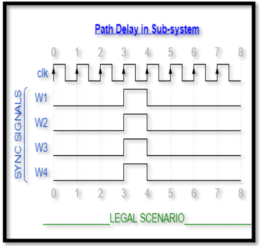
- Commonly referred as Crossbar (Xbar)
- Help in routing the data in switch fabrics



PathDelays at sub-system

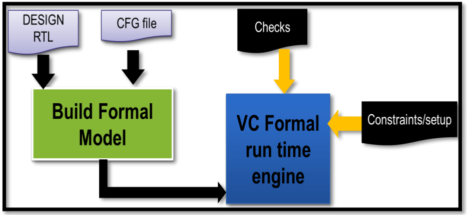
- All control signals should arrive at same time
- System performance reduction due to packet drops

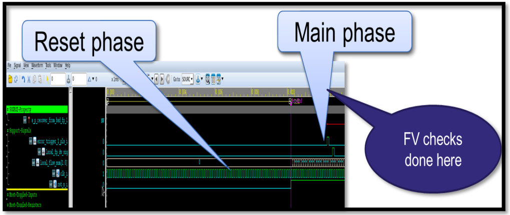




VC Formal Flow

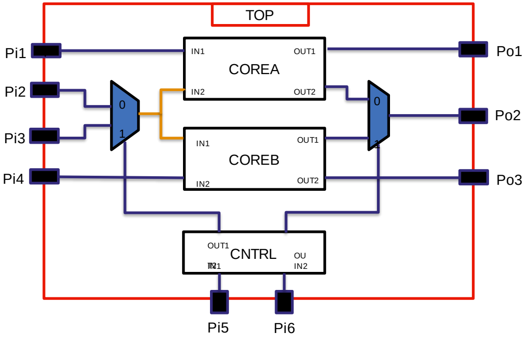
- Two phases :
  - Compile, analyze and build the model
  - Run checks on the built-model



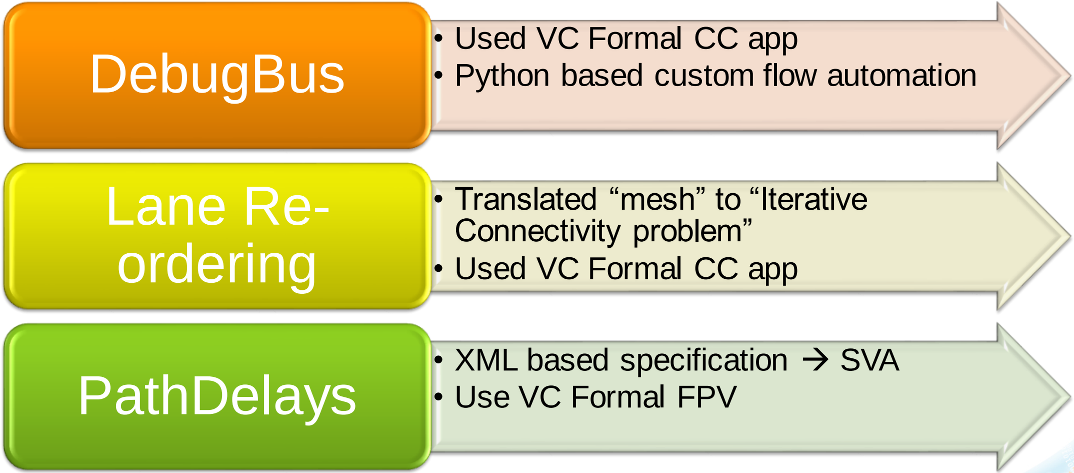


VC Formal - CC app

- Connectivity Checking (CC) app
  - Checks that logical & structural connection exists between source and destination



Deploying VC Formal on our Design Structures



DebugBus

- Used VC Formal CC app
- Python based custom flow automation

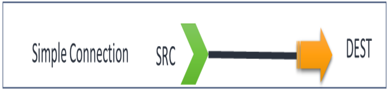
Lane Re-ordering

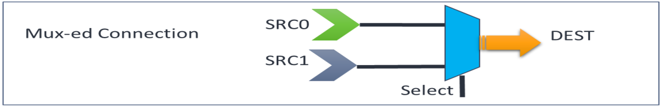
- Translated “mesh” to “Iterative Connectivity problem”
- Used VC Formal CC app

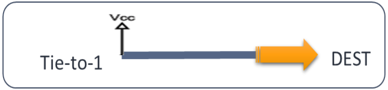
PathDelays

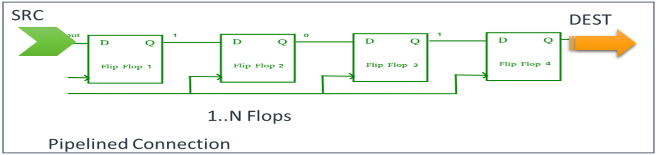
- XML based specification → SVA
- Use VC Formal FPV

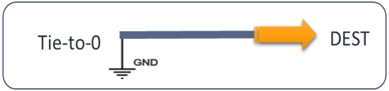
Types of Connections

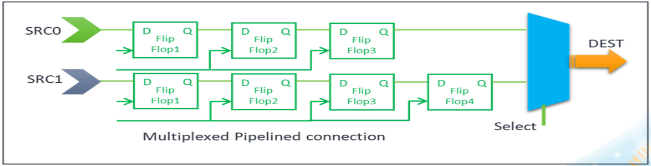






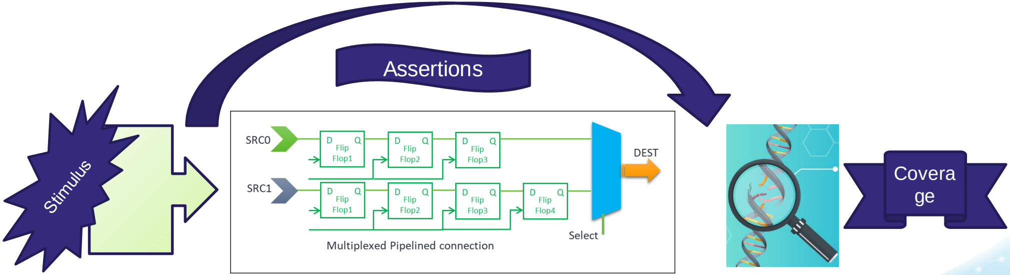






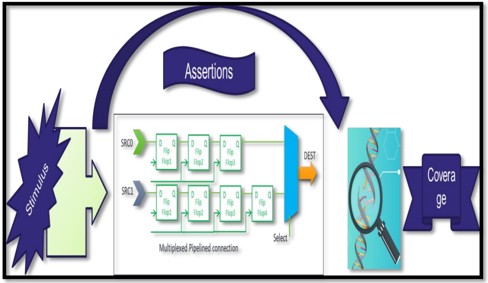
Verifying connectivity

- Connections - regular structures
- 3 key elements needed:
  - Stimulus (Src, Enable, Clk, Reset)
  - Checkers – *a_cc_chk_0 : assert property (en_0 == 0 |-> ##LAT dst_0 == $past(src_0, LAT));*
  - Coverage – ensure all combinations are verified



Verifying connectivity – simulation

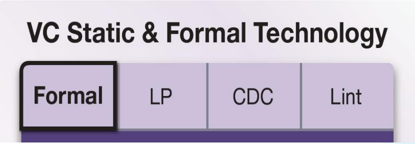
- Add assertions
- Devise quality patterns:
  - Walking one-s, zero-s etc.
  - Every bit toggle
  - Reverse of the default values
  - Negate all “un-selected paths” etc.
- Add FCOV to ensure all intended patterns are run



- Formal – specify what you need to “check”
  - Leave the rest to the engines

DON’T STRESS
DO YOUR BEST
FORGET THE REST

PICTUREQUOTES.com

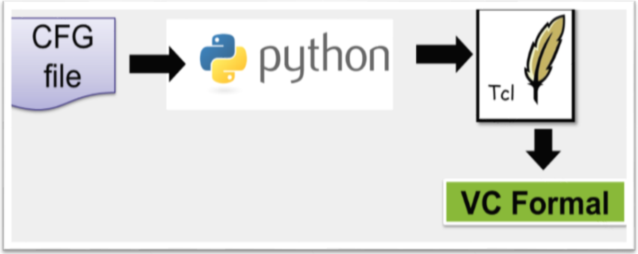


VC Static & Formal Technology

Custom flow around VC Formal

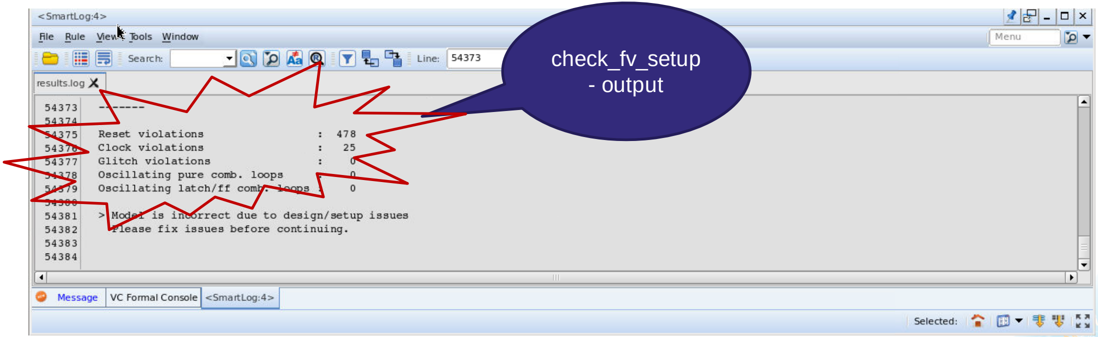
- Building a Formal Model requires basic design information
- We developed a CFG file to capture this
- Build Phase Requirements
  - Clocks
  - Resets
  - Derived clocks
  - Derived resets
  - Any black-boxes

```
[DebugBus]
Name: RX_IP
sink_sig: rx_ip.dbg_mux_out
sel_sig: rx_ip.sel_sig
clk_sig: rx_ip.clk
rst_sig: rx_ip.rst_n_i
rst_level: LOW
latency: 2
```



Importance of clocks and resets in FV

- Without proper FF identified, Formal runs will not be fruitful
- VC Formal has a mechanism to “check the FV setup” (check_fv_setup)
- Indicates potential missing clocks, resets etc.



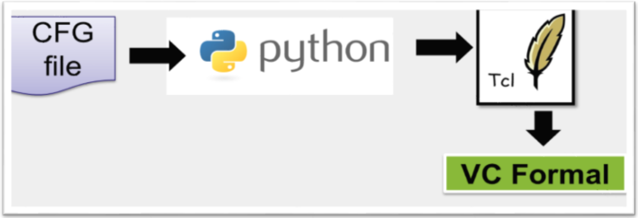
```
--------

Reset violations                  :  478
Clock violations                  :   25
Glitch violations                 :    0
Oscillating pure comb. loops      :    0
Oscillating latch/ff comb. loops  :    0

> Model is incorrect due to design/setup issues
  Please fix issues before continuing.
```

check_fv_setup - output

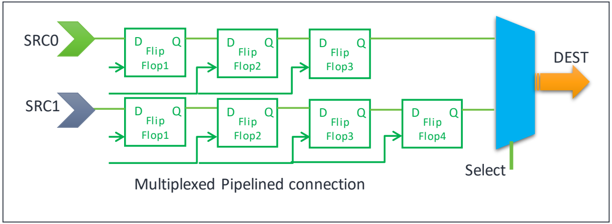
Connectivity specification - abstracted

- CFG file capturing DebugBus Requirement
- Connectivity information:
  - Source
  - Sink
  - Latency (Optional)
  - Enable value (Optional)

```
[MuxedConnections]
sel_sig: rx_ip.dbg_sel_sig

# select signal value
[0]
sink_slice: 31:2
src_sig: rx_ip.src_0

# select signal value
[1]
sink_slice: 0
src_sig: rx_ip.src_1_0
sink_slice: 1
src_sig: rx_ip.src_1_1
```



Specifying pipelined connection with Multiplexed path



Multiplexed Pipelined connection

Equiv. SVA (add_cc command in VC Formal)

```
[Muxed Pipelined Connections]
sel_sig: rx_ip.dbg_sel_sig
# select signal value
[0]
sink_slice: 31:1
src_sig: rx_ip.src_0
latency: 3
# select signal value
[1]
sink_slice: 0
src_sig: rx_ip.src_1_0
latency: 4
```

```
a_cc_chk_0 : assert property (##3 rx_ip.dbg_sel_sig == 0 |->
                         rx_ip.dbg_mux_out[31:1] == $past(rx_ip.src_0, 3));

a_cc_chk_1 : assert property (##4 rx_ip.dbg_sel_sig[0] == 1 |->
                         rx_ip.dbg_mux_out[0] == $past(rx_ip.src_1_0, 4));
```

Deploying VC Formal on our Design Structures

DebugBus

- Used VC Formal CC app
- Python based custom flow automation

Lane Re-ordering

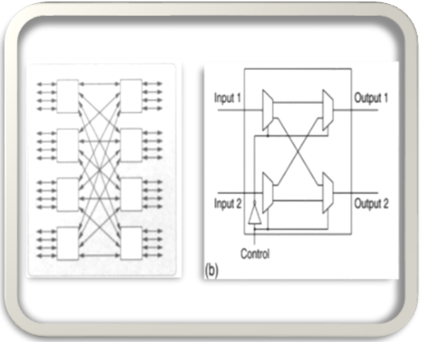
- Translated “mesh” to “Iterative Connectivity problem”
- Used VC Formal CC app

PathDelays

- XML based specification → SVA
- Use VC Formal FPV

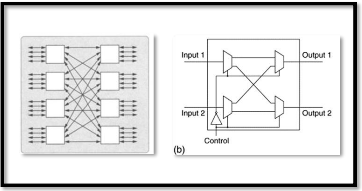
Lane re-ordering verification

- Classical crossbar
  - Routes traffic from input to output as per config
- Verifying each combination is critical for closure
- Each “lane config” consumes 180 minutes of simulation time
  - 16 lanes → 256 (16*16) various configurations to be tested
  - 256 * 180 == 46080 minutes → 768 hours → 32 days!
  - Also requires smart test + coverage model



Using VC Formal for Lane re-ordering

- Having verified one lane in simulation – we translated this problem into “iterative connectivity verification”
  - Do not write 256 “properties” and make mistakes!

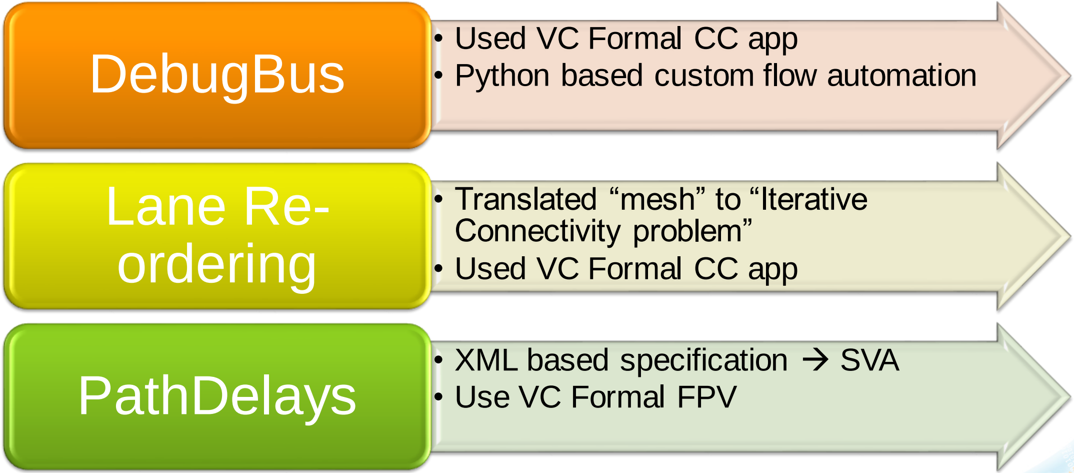


IterativeConnectivity
Name: Lane_reorder
loop_var: **itr_var**
loop_start: 0
loop_end: 15
xbar: true
sel_sig: ip_lane_<**itr_var**>_cfg
src_sig: ip_lane_<**itr_var**>_inp
sink_sig: ip_lane_<**itr_var**>_out

Equiv. SVA (add_cc command in VC Formal)

```
xbar_chk_sel_0_out_0: assert property (
     ##1 ip_lane_0_cfg[3:0] == 0 |->
          ip_lane_0_out[79:0]  == $past(ip_lane_0_inp[79:0] , 1));

 xbar_chk_sel_1_out_0: assert property (
    ##1 ip_lane_0_cfg[3:0] == 1 |->
          ip_lane_0_out[79:0]  == $past(ip_lane_1_inp[79:0] , 1));
```

Deploying VC Formal on our Design Structures



DebugBus

- Used VC Formal CC app
- Python based custom flow automation

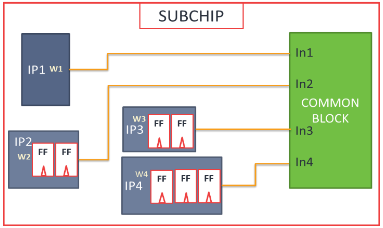
Lane Re-ordering

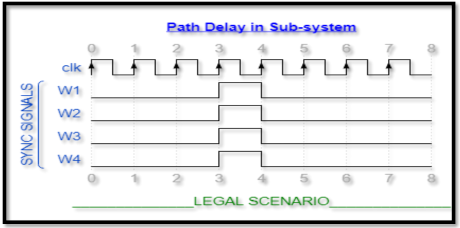
- Translated “mesh” to “Iterative Connectivity problem”
- Used VC Formal CC app

PathDelays

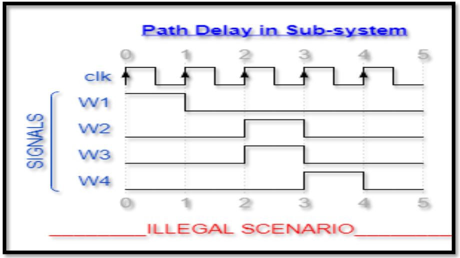
- XML based specification → SVA
- Use VC Formal FPV

Verifying PathDelays at subsystem



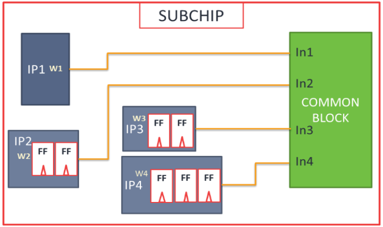


- Each IP verified independently
- Sub-chip sims are very long
- Not all scenarios from IP to sub-chip are ported/re-run



Using VC Formal for PathDelays

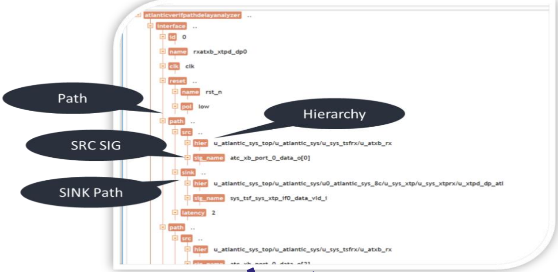
```
property p_const_path_del (src_sig, sink_sig, LATENCY);
  ##LATENCY $changed(src_sig) |=> ## (LATENCY - 1) $changed(sink_sig);
 endproperty : p_const_path_del

 a_data_del : assert property (p_const_path_del(
             .src_sig(ip1_inst.w2),
             .sink_sig(common_blk_inst.in2),
             .LATENCY(2)));

a_dv_del : assert property (p_const_path_del(
             .src_sig(ip1_inst.w3),
             .sink_sig(common_blk_inst.in3),
             .LATENCY(2)));
```



- Ask VC Formal to “disprove” the above assertions
- Measure one path latency in simulation
- Feed it to SVA model and run Formal
- System architects get involved in these requirements – hard for them to code SVA

Using XML for PathDelay specification

- XML is a popular format
- System architects familiar with XML
- We developed a flow to extract SVA + VC Formal setup from XML
- Several black-boxes setup to handle huge design



```
atlanticverifpathdelayanalyzer ..
  interface ..
    id 0
    name rxatxb_xtpd_dp0
    clk clk
    reset ..
      name rst_n
      pol low
    path ..
      src ..
        hier u_atlantic_sys_top/u_atlantic_sys/u_sys_tsfrx/u_atxb_rx
        sig_name atc_xb_port_0_data_o[0]
      sink ..
        hier u_atlantic_sys_top/u_atlantic_sys/u0_atlantic_sys_8c/u_sys_xtp/u_sys_xtprx/u_xtpd_dp_atl
        sig_name sys_tsf_sys_xtp_if0_data_vld_i
      latency 2
    path ..
      src ..
        hier u_atlantic_sys_top/u_atlantic_sys/u_sys_tsfrx/u_atxb_rx
        sig_name atc_xb_port_0_data_o[2]
```

```
property p_const_path_del (src_sig, sink_sig, LATENCY);
  ##LATENCY $changed(src_sig) |=> ## (LATENCY - 1)
$changed(sink_sig);
 endproperty : p_const_path_del

 a_data_del : assert property (p_const_path_del(
           .src_sig(ip1_inst.w2),
           .sink_sig(common_blk_inst.in2),
           .LATENCY(2)));

a_dv_del : assert property (p_const_path_del(
           .src_sig(ip1_inst.w3),
           .sink_sig(common_blk_inst.in3),
           .LATENCY(2)));
```

TCL
(VC Formal)

Results

Quality, Completeness & Shift-left

DebugBus verification

Using CC app

Name	Number Of Assertions	Constraints
IP1	16	13
IP2	29	10
IP3	539	7
IP4	131	9
IP5	2	5
IP6	1	30
IP7	36	4
IP8	358	12
IP9	21	42
IP10	36	27
IP11	48	563
IP12	51	316
IP13	52	37
IP14	35	5

Lane re-ordering verification results

Lane re-ordering typical simulation time

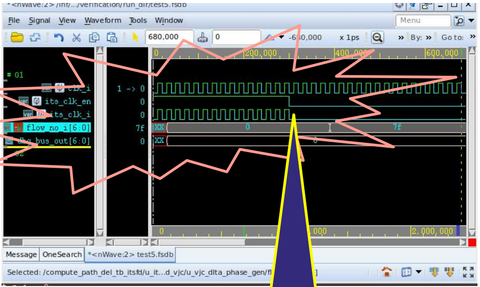
Number of lanes	Typical simulation time per-lane (minutes)	Total simulation time (Extrapolated in minutes)
256	180	46080

Lane reordering VC Formal build and run time

Number of assertions	Build time	Total VC Formal run time (in minutes)
256	5	10

Sample falsification – Debug with Verdi

- Given a set of connections, tool ran and produced few violations
- VC Formal generates FSDB for:
  - Debugging failures (Falsifications)
  - Witness generation
  - Reset trace



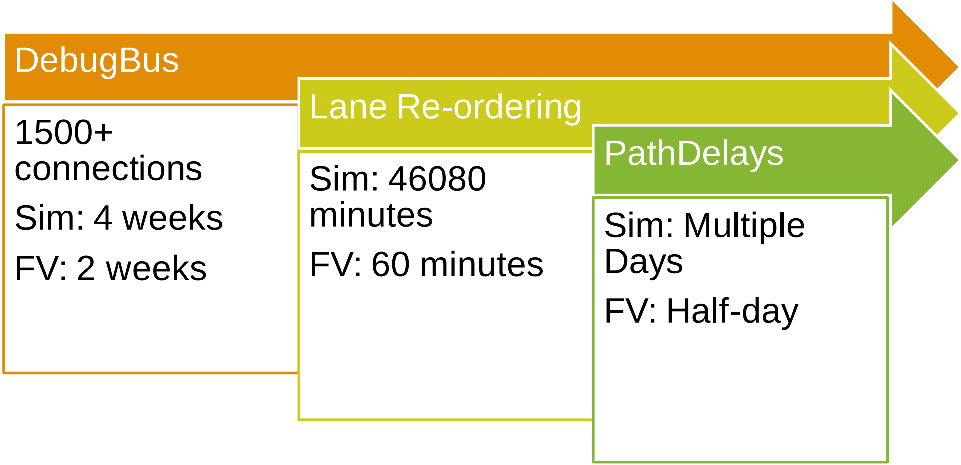
Selected: /compute_path_del_tb_itsfd/u_it...d_vjc/u_vjc_dlta_phase_gen/fl]

clk_en toggled during prove-phase

- Resolution
  - Gated clock, needs a stable clk_en
  - TCL, SystemVerilog etc.
  - set_constant/sim_force

Shift-left with VC Formal - results

Augments simulation



DebugBus

1500+ connections

Sim: 4 weeks

FV: 2 weeks

Lane Re-ordering

Sim: 46080 minutes

FV: 60 minutes

PathDelays

Sim: Multiple Days

FV: Half-day

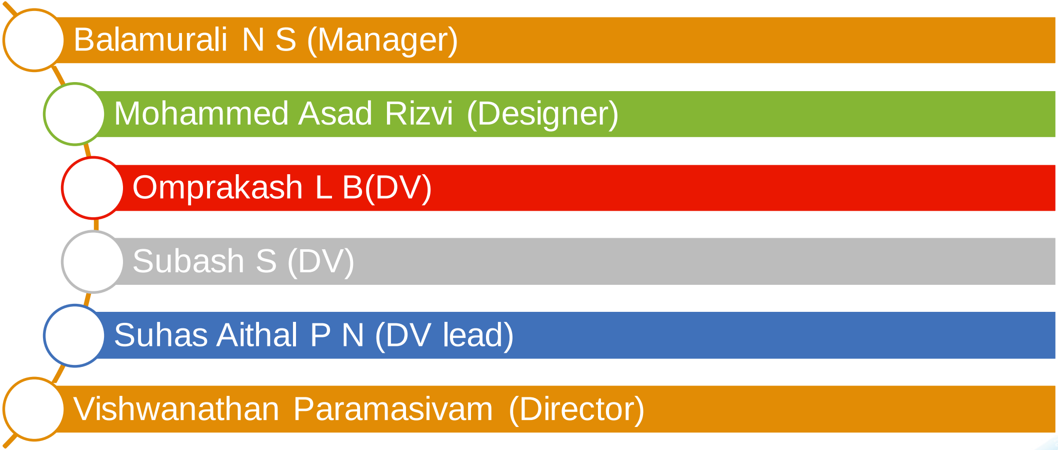
Conclusions & future work

It is not a destination, but a journey

- Team has deployed VC Formal on an ongoing project
- Used VC Formal with apps:
  - Connectivity
  - FPV
- Also using FCA (Formal Coverage-unreachability Analysis)
  - Saves RTL designer's review time
- FPV usage can increase on new designs
  - Fresh RTL, easier to add assertions
  - Can start even before simulation
- Formal is here to stay along with simulation

Acknowledgments

It is a teamwork!



- Balamurali N S (Manager)
- Mohammed Asad Rizvi (Designer)
- Omprakash L B(DV)
- Subash S (DV)
- Suhas Aithal P N (DV lead)
- Vishwanathan Paramasivam (Director)

Thank You