Practical experience sharing of adopting Jasper on register/connection verification

Presented to: UIT DV Club

CNBG DV team
Richard An
richard_an@realtek.com
2025-12-16 Tue.

Realtek AI : Changing the Future

The Spirit of the Crab

Inspired by its communal nature fostering unity and teamwork, we chose the crab as our corporate logo and courageously embrace challenges in a world demanding innovation and change.

人信信仰

We value the Realtek culture of
SELF-CONFIDENCE AND TRUST IN PEOPLE

We believe that we can achieve our best,
and trust that our partners can also do the same.
Working and learning in Realtek,
we openly share knowledge and experience
with one another to inspire innovation
and pursue growth of the company,
as well as the individual.

REALTEK

Corporate Profile

- Founded 1987
- Capital US$167M
- Revenue US$3.5B (2024)
- Headquarters Hsinchu, Taiwan
- Employees 7,700+
- Products Communications Network ICs
  Connected Media ICs
  Computer Peripheral ICs
  Multimedia ICs
  Smart Interconnect ICs

Competitive Advantages

Technology

Solid

Service

Total Customer Satisfaction

Innovation

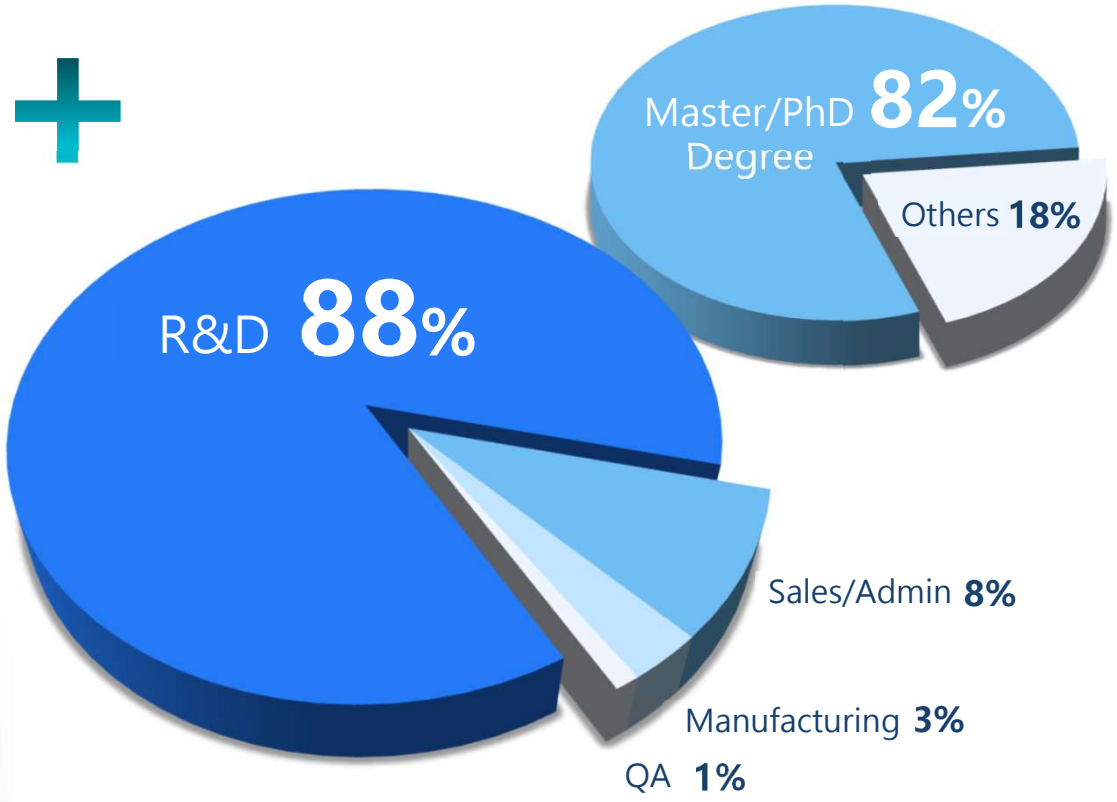
Creative

Realtek Worldwide Locations

Our Asset — People

7,700+

Total Employees (as of Sep, 2025)



Realtek Product Lines

Communications Network

- Automotive Ethernet Switch / PHY
- Switch / Gateway
- xDSL / xPON
- Wi-Fi / Router / BT/ IoT

Connected Media

- **Ethernet NIC / PHY**
- STB / OTT
- USB High Speed Bridge

Computer Peripheral

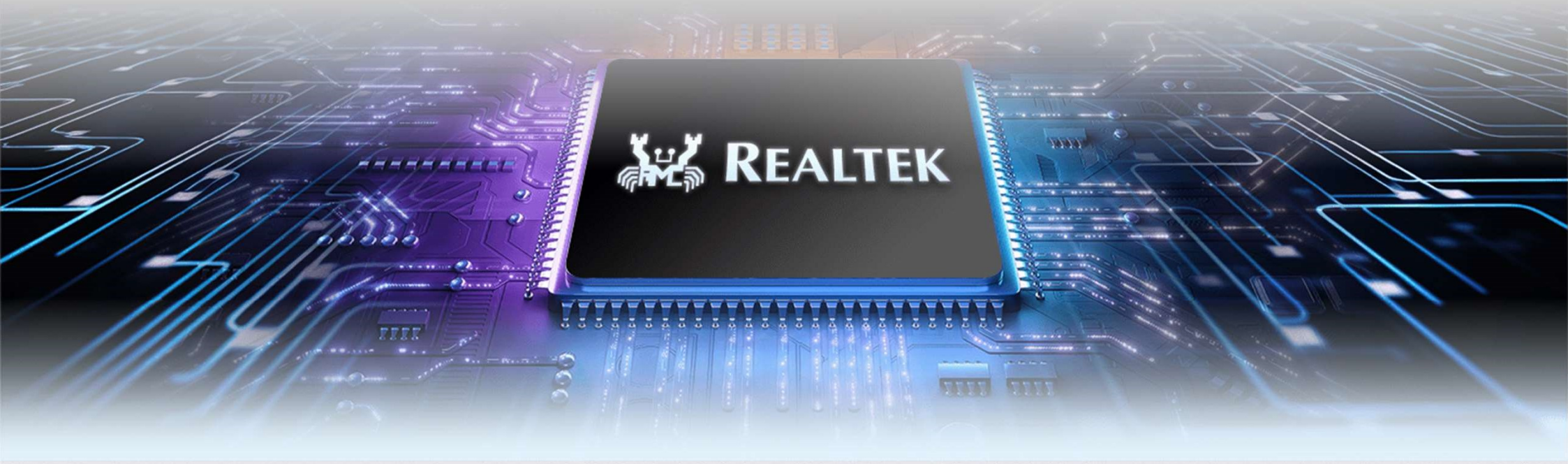
- **PC Audio Codec and DSP**
- IP / **PC Camera**
- Consumer Audio and DSP

Multimedia

- LCD TV
- **LCD Monitor**
- **Display Translator**
- Display Hub / Retimer

Smart Interconnect

- **Card Reader**
- USB 3.2 / USB4 Hub
- Type-C PD / Redriver
- Embedded Controller



Realtek Product Applications

Consumer

- Ethernet
- OTT
- Wi-Fi / BT
- IoT
- CE Codec
- IP Cam
- TV

Computer Peripheral

- Ethernet
- Wi-Fi / BT
- PC Codec
- CE Codec, Audio Amplifier
- Card Reader
- PC Cam
- Type-C, PD, SSD
- USB Hub
- Display Translator
- Display Hub / Retimer
- Monator Scaler
- Embedded Controller

Communications

- Ethernet
- Router
- PON
- Switch
- Wi-Fi / BT
- OTT

Healthcare

- Wi-Fi / BT
- IoT
- IP Cam
- Monitor Scaler

Automotive / Humanoid Robot

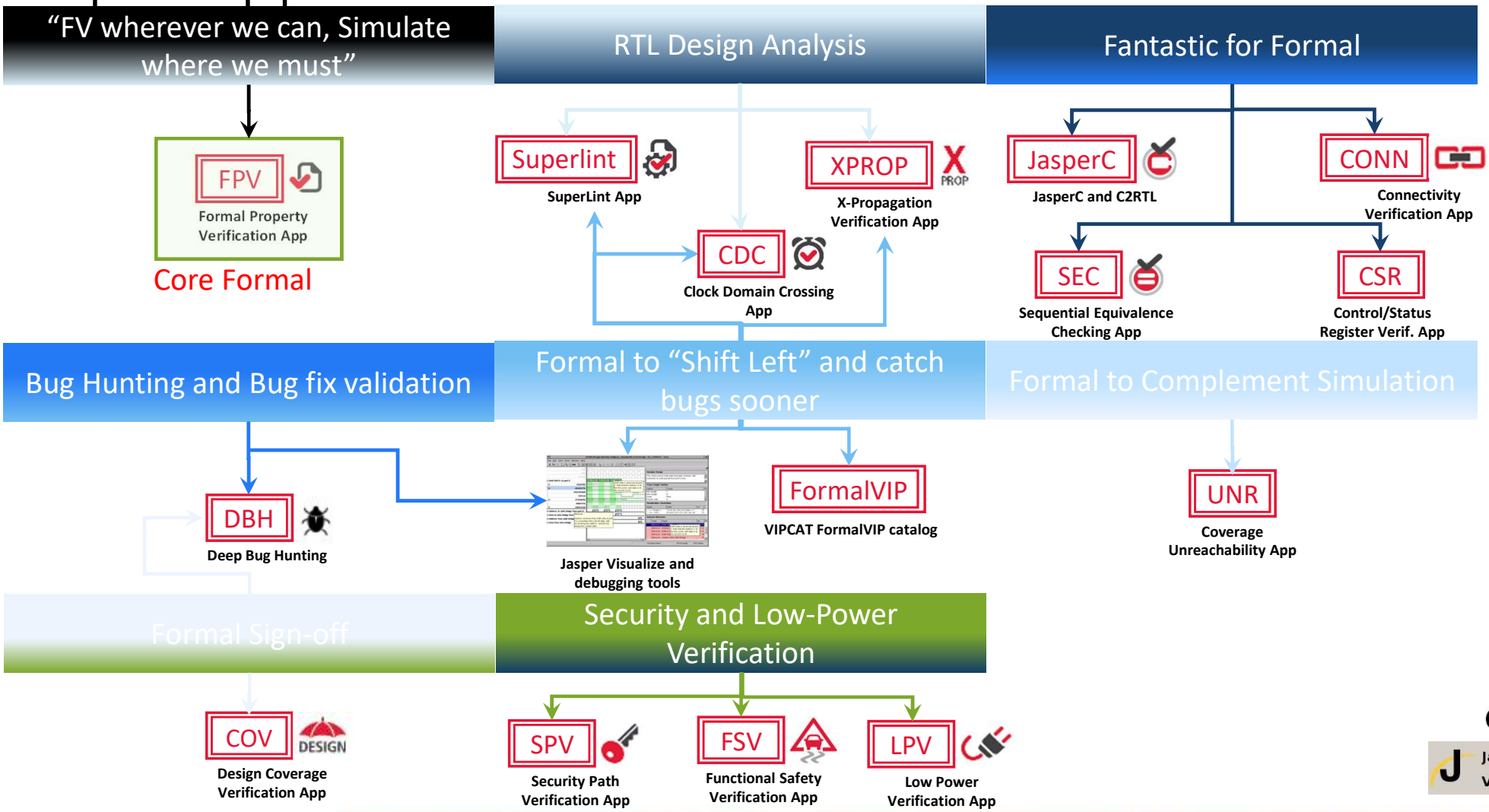
- Ethernet Switch
- Ethernet PHY
- Wi-Fi / BT
- Audio DSP & AMP
- Automotive SoC
- Display Scaler
- Display Hub / Retimer

Industrial Automation

- Ethernet
- Switch
- Wi-Fi / BT
- IoT

Jasper Apps : Formal Verification Solution Leader

Expanding your horizons with Formal Verification



cadence

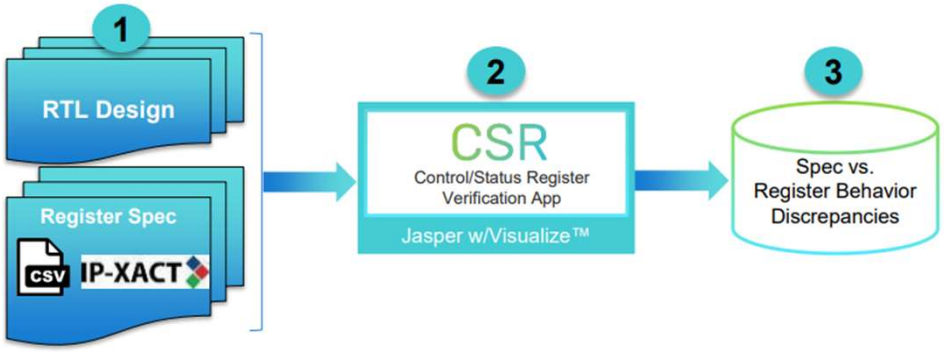
Jasper™ Apps Verification System

Jasper CSR App

Control and Status Register Verification

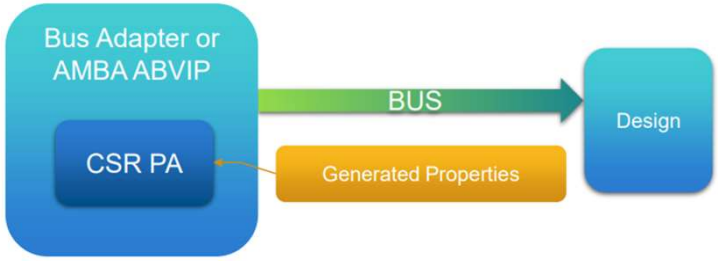
CSR
Control/Status
Register Verif. App

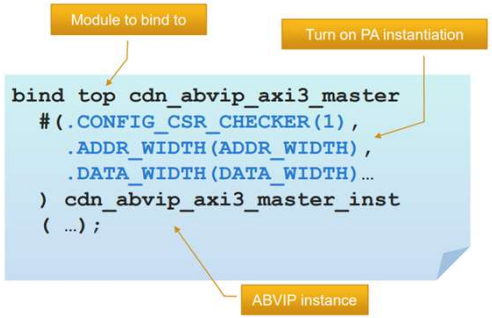
Flow Overview



- 1. Inputs: register spec and your RTL design
- 2. Run Jasper CSR App
  - App automatically derives & generates all properties
  - Automatically runs formal engines under-the-hood to prove all properties
- 3. Output: any counter-examples are discrepancies between the spec and RTL

Simple CSR Setup





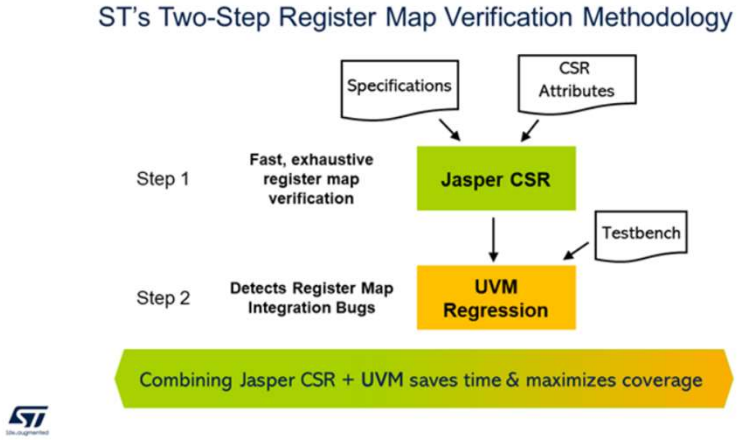
- CSR ***Exhaustively*** verifies that DUT values matches internal model value under **all** conditions

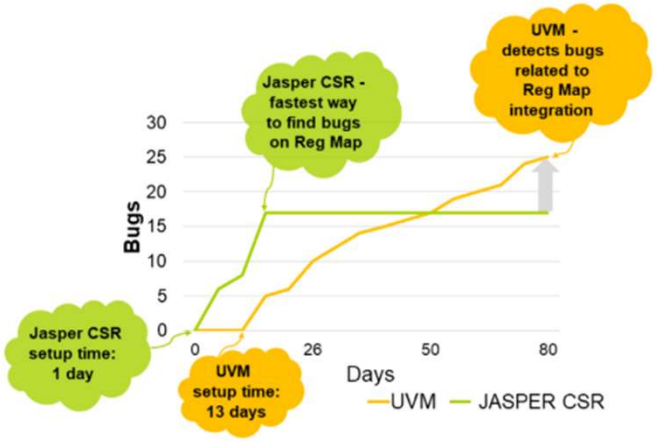
cadence
Jasper™ Apps
Verification System

Jasper CSR App

Customer's case in 2023 : combine UVM and Jasper CSR

- UVM finds bugs related to how the register map interacts with the other parts of the design — something that Jasper CSR does not cover.
- Jasper CSR found bugs that were missed by UVM. So, the combination of the two methodologies enables us to find all possible bugs



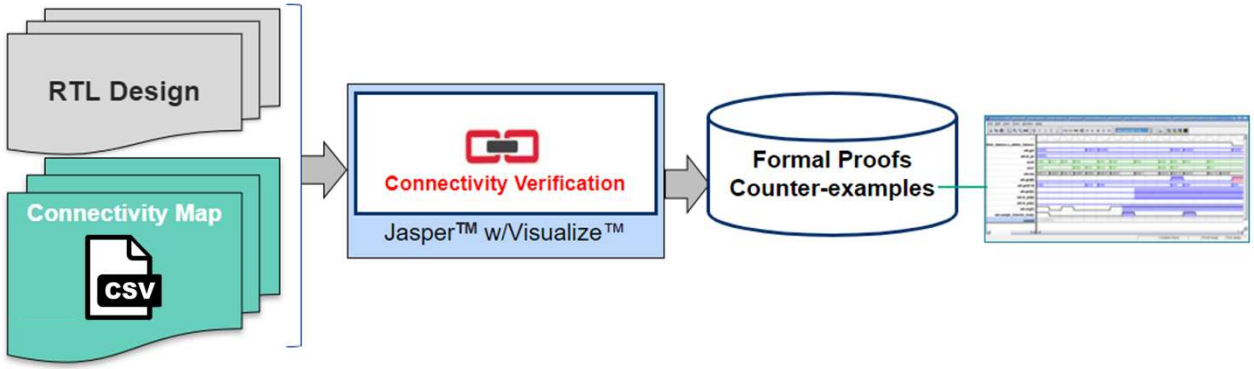


- **Is it possible to only use Jasper CSR (Formal) ?**
  - **Complexity concern**

Jasper CONN App

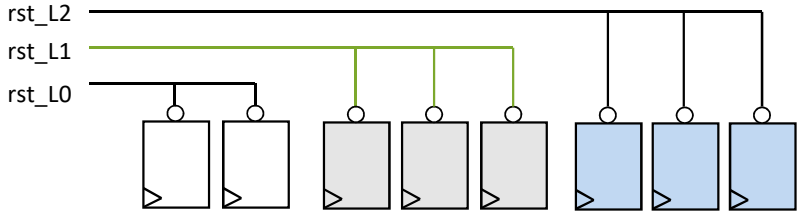
Connectivity Verification

- Connectivity verification is a significant piece of the **SoC** integration challenge
- High ROI
  - No testbench needed
  - CONN automatically generates all properties for the given spec
  - Easy to setup
  - Easy to debug (provide CEX)

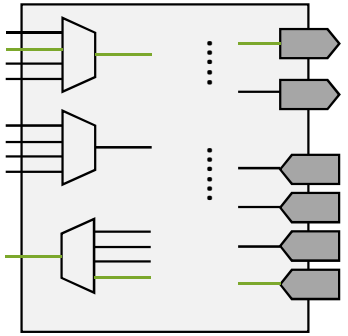


- Usage models
  - **Wire connections** between modules

  - **Reset** connections



  - **PinMux** verification



cadence

Jasper™ Apps Verification System

Practical Case 1

Jasper CSR + CONN

Case 1 : Jasper CSR + CONN

Result Summary

- Deploy Jasper CSR and CONN to fully handle verification of configure registers
- CSR : Register functionality behavior check

# of Mod/Field	Runtime	RTL bug	Spec issue	TB Refine
10/2664	42469 sec	6	8	1

- CONN : Connectivity check between **register** and **analog modules**

# of Conn	Runtime	RTL bug	Spec issue	TB Refine
414	3807 sec	1	11	9

- **Easily found RTL bug / Spec issue in both Apps**

Case 1 (CSR) : RTL Bug

Write 1 Pulse Reg is **WRONGLY** coded

CSR
Control/Status
Register Verif. App

- Spec

[14]	R/W1P	0x0	C_REG_A

- Write 1 Pulse Design

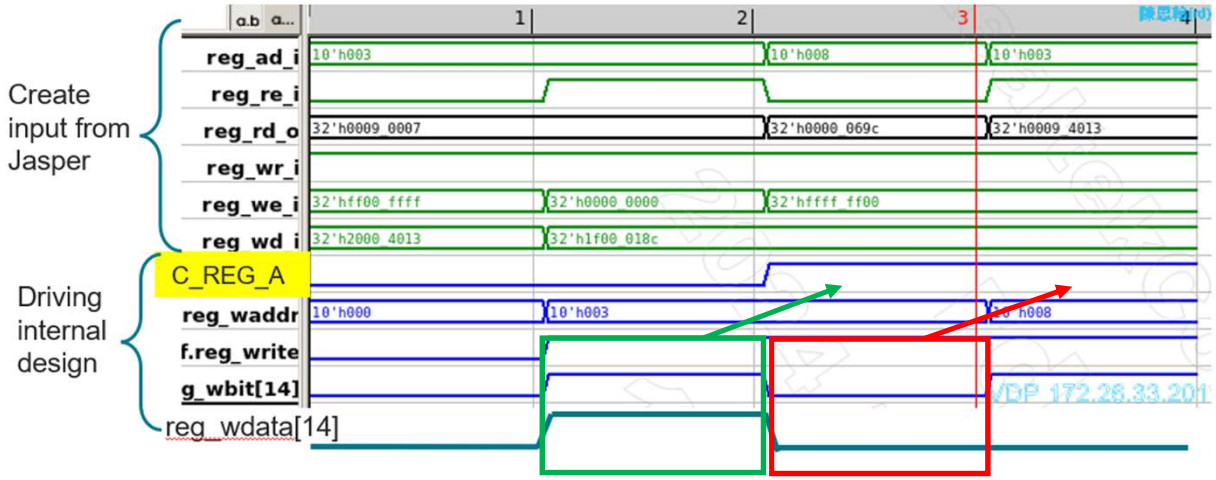
if reg_write
C_REG_A <= wbit[X] ? wdata[X] : 0

- RTL in Design

if reg_write
C_REG_A <= wbit[X] ? wdata[X] : C_REG_A

Solution : Change RTL to correct Write 1 Pulse

- Jasper CSR Trace



Case 1 (CSR) : Spec Issue

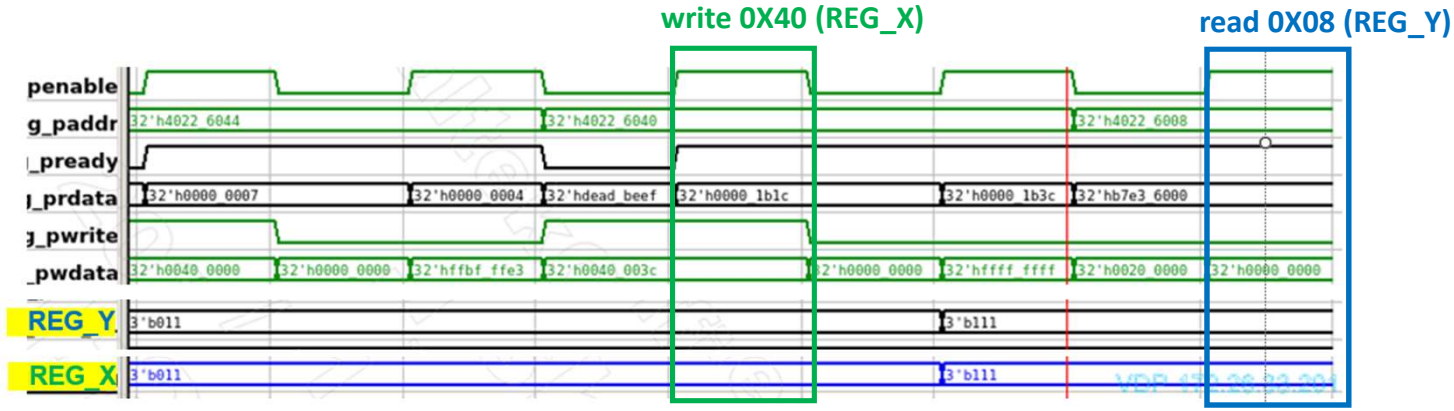
Direct Write misses in SPEC / mapping address wrong

CSR
Control/Status Register Verif. App

- Spec : While write REG_X, the same value is also written to REG_Y @ ~~0X18~~ 0X08

REG_X @ 0X40	[5:3]	R/W	0x3	REG_X	Map to 0x18[25:23]
REG_Y @ 0X08	[25:23]	R	0x3	REG_Y	

- Jasper CSR Trace



Solution : Add correct "Direct Write" to CSR Register Spec

FIELD, REG_X, 5, 3, RW, 0x3

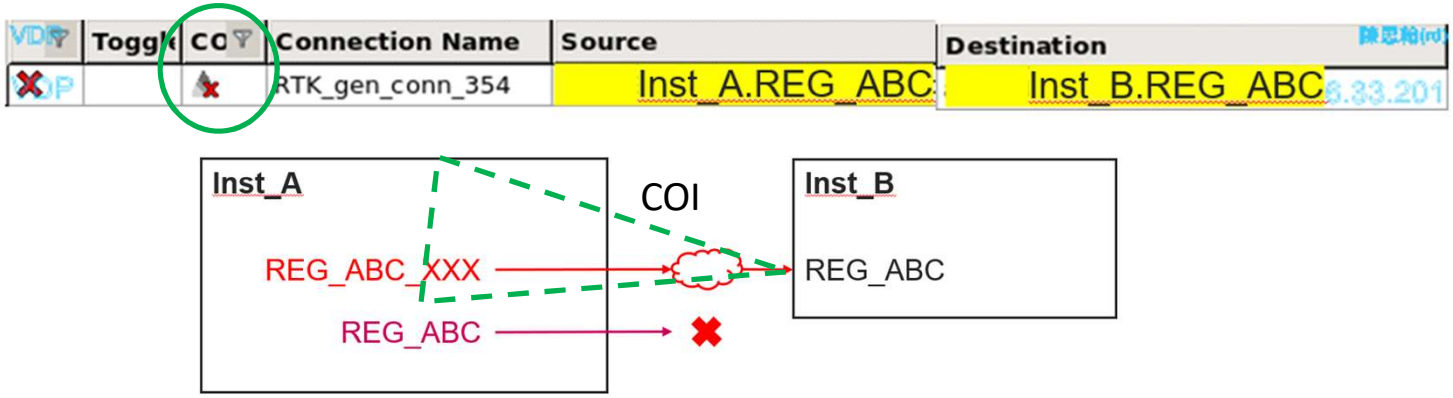
```
,FIELD, REG_Y 25, 23, RO, 0x3, """ """,
,ATTR, DIRECT_WRITE, 1'b1, Expression of REG_X with other signals
```

Case 1 (CONN) : RTL Bug

CONN
Connectivity Verification App

RTL Connection Wrong

- **COI Check Fail** : Source is NOT in the COI (Cone-of-Influence) of Destination
  - It's a **structure check** and **NEED NOT** to run any formal
  - **Very Quickly to get the result**



Solution : Change RTL to correct connection

Case 1 (CONN) : Spec Issue

CONN Connectivity Verification App

Direction Opposite

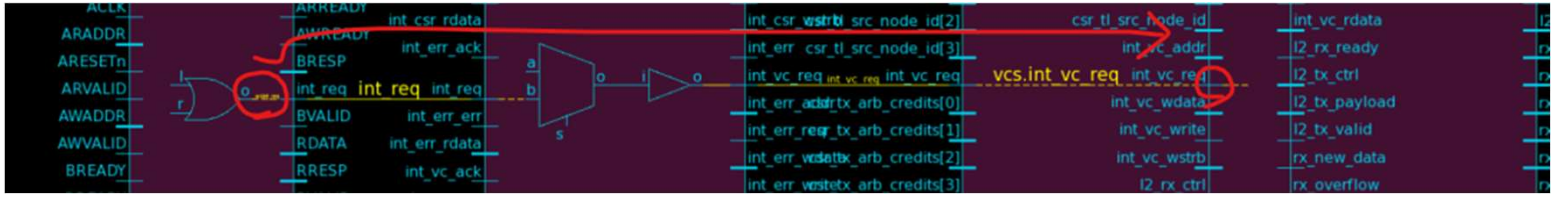
- **COI Check Fail** : Source is NOT in the COI (Cone-of-Influence) of Destination
- **Value Check PASS in this case (Assertion : src==dest)**

VDR	Toggle	COI	Connection Name	Source	Destination
✓		✗	RTK_gen_conn_310	Inst_D.sig_D	a_ Inst_E.sig_E

Solution : Swap Source and Dest in Connectivity Table

- Jasper CONN could extract the **schematic** of specific connection to debug

```
check_conn -connection connection_name -show_schematic
```



Jasper could *explore* design

CONN
Connectivity Verification App

- Jasper provides plentiful commands to explore design
- "***graph -shortest***" is a **general** command to find the shortest path of two points with direction

Spec direction :

```
[Connectivity] % graph -shortest –from Inst_D.sig_D –to Inst_E.sig_E
2024-12-16 16:35:33: INFO (IGPH001): Creating Graph (50% completed).
2024-12-16 16:35:33: INFO (IGPH001): Creating Graph (100% completed).
2024-12-16 16:35:33: WARNING (WGPH001): No path between Inst_D.sig_D and Inst_E.sig_E
```

Opposite direction :

```
[Connectivity] % graph -shortest path -from Inst_E.sig_E –to Inst.D.sig_D
2024-12-16 16:37:35: INFO (IGPH001): Creating Graph (80% completed).
2024-12-16 16:37:35: INFO (IGPH001): Creating Graph (90% completed).
2024-12-16 16:37:35: INFO (IGPH001): Creating Graph (100% completed).
{a
{a
{a
{a
{a
{a
```

(Trace the path and list the nets through the path)

REALTEK

Practical Case 2

Jasper CSR + FPV

Case 2 : Jasper FPV + CSR

FPV Formal Property Verification App

CSR Control/Status egister Verif. App

- Configure register modules are located in deep hierarchy different blocks
- From top AMBA interface, DUT need to pass through **multiple stage in-house bridge designs** to access specific register (figure is shown in next page)

- Practical Difficult : Directly set CSR top to cover AMBA interface causes **high complexity CSR properties**
- Properties cannot converge and keep "undetermined" for long runtime
- Very common in SOC-level design

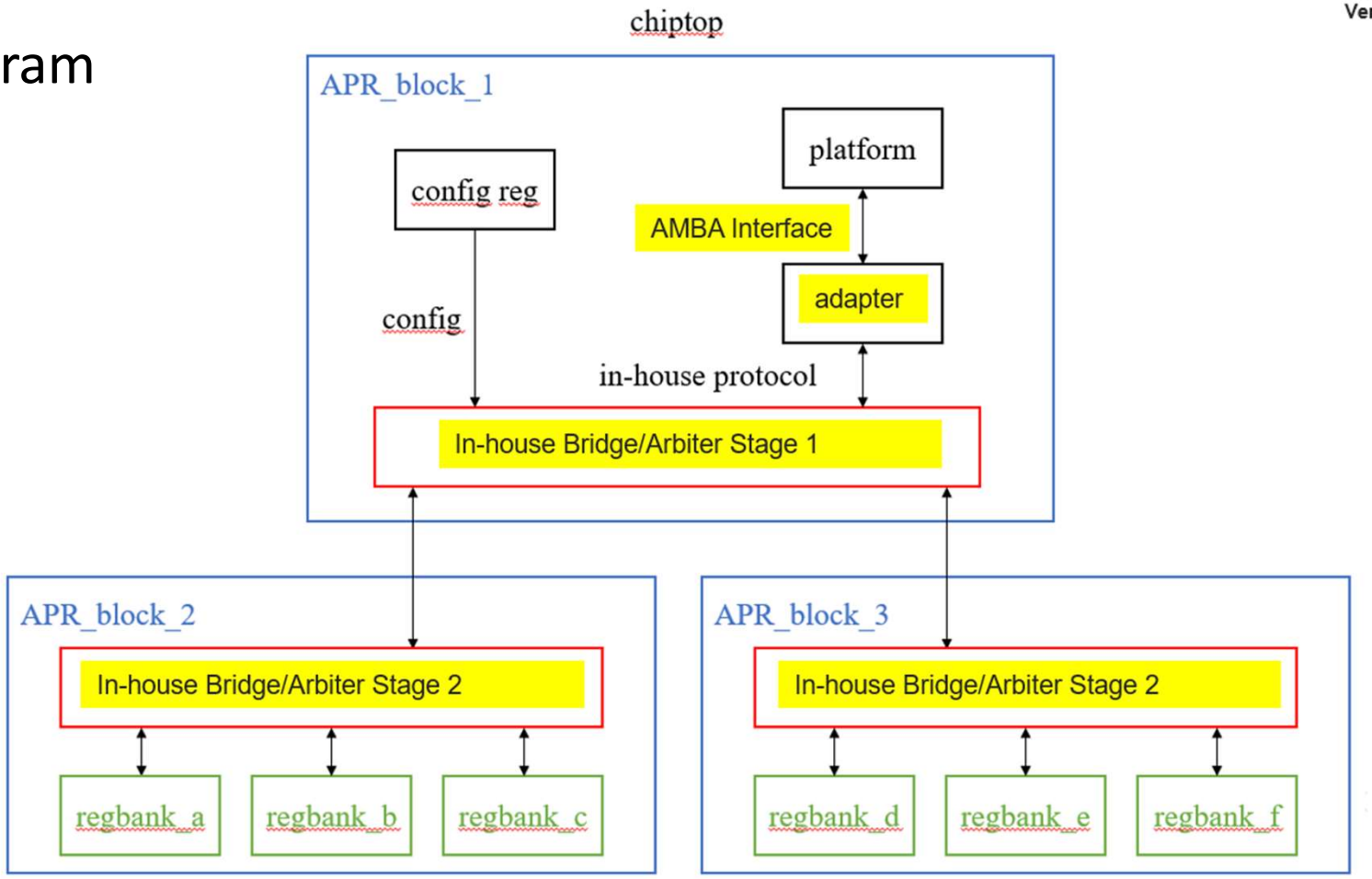
Design complexity	
Gate	2517317
Flop	550051
Counter	3926
FSM	884

Case 2 : Jasper FPV + CSR

FPV Formal Property Verification App

CSR Control/Status egister Verif. App

- Block Diagram



Case 2 : Jasper FPV + CSR

Reduce Complexity – Strategy

FPV Formal Property Verification App

CSR Control/Status egister Verif. App

- In practical, the decision to cut-off the design may depend on
  - Available License number
  - Desired runtime
  - The number/complexity of assertions
  - Script to automate the flow (generate property / parsing spec….)

- Experiment result : Run Jasper CSR on a **SINGLE** Regbank (334 regs and 438 fields)
  - The interface of CSR modules/APR block out of verification scope are tied to inactive

Verification Scope	Access interface	Strategy	Runtime		
			One field	One register	Entire regbank
Chiptop	Top AMBA	-		> 62 hours	
Apr block	Top AMBA	Blackbox irrelevant apr blocks	1.5 hours	> 3 hours	
Regbank	Top AMBA	Specify formal engine		10 minutes	> 14 hours
Apr block	Apr block	-		45 seconds	21.5 hours
Regbank	Regbank	-		4 seconds	108 minutes
Regbank	Regbank	Increase license from 1 to 3			**69 minutes**

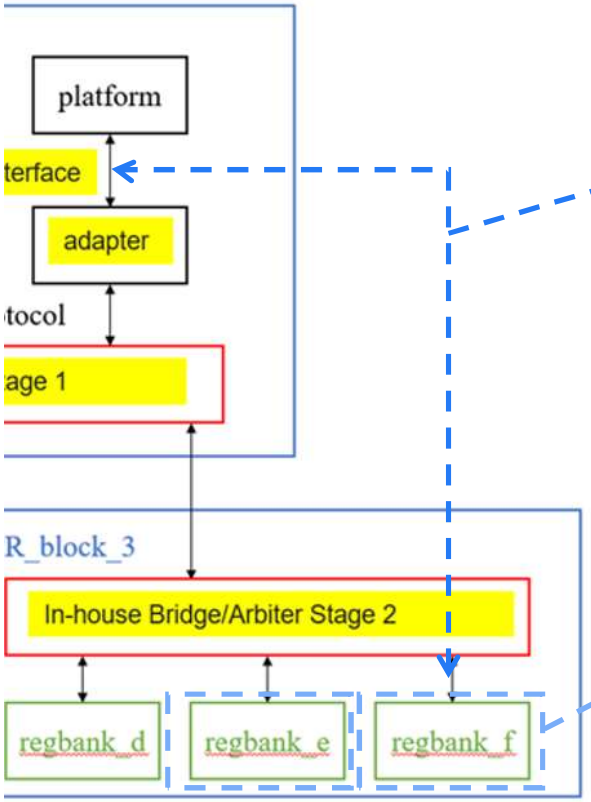
Case 2 : Jasper FPV + CSR

Reduce Complexity – Strategy

FPV Formal Property Verification App

CSR Control/Status egister Verif. App

- We **DIVIDE** the verification into **Regbank** and **Transaction** verification



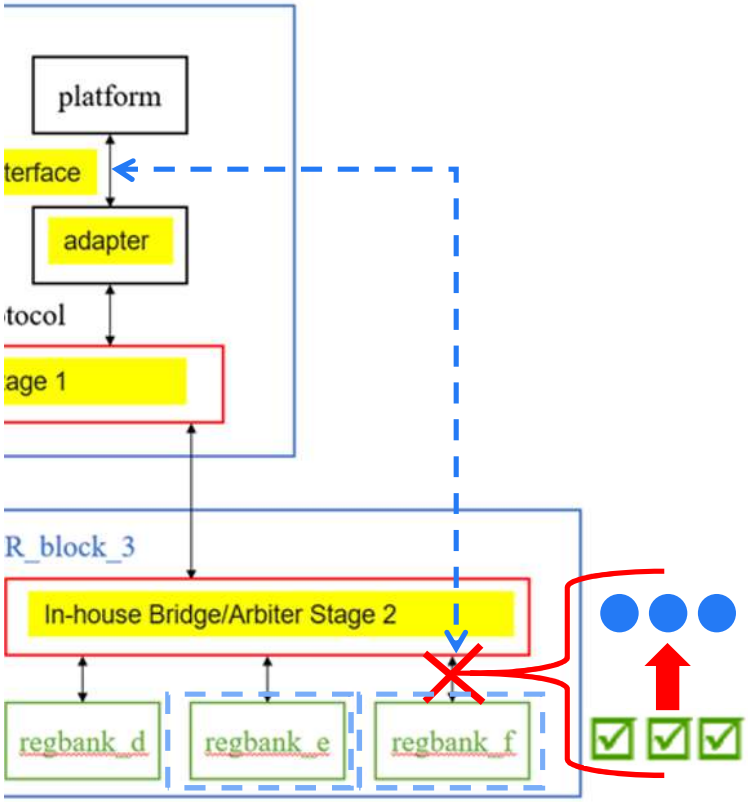
Transaction verification	
Target	Check the **transaction** on Top AMBA and regbank is aligned
Scope	Transaction path from Top AMBA to regbank interface
App	Jasper FPV

Regbank verification	
Target	Check **behavior** of each Regbank
Scope	Regbank only
App	Jasper CSR

Case 2 : Jasper FPV + CSR

Reduce Complexity – Strategy

FPV
Formal Property Verification App



Total idea of complexity reduction

- The behavior of reg R/W on RegBank is well verified by **Regbank verification**
- Then we **cut-off** read_data out and write **helper assertions** to model the behavior of it
- From the proven result of **help assertions**, we get **reliable constraints to abstract read_data out**
  - Jasper has “assert -> assumption” push button for this
- **With the abstraction** we successfully lower the complexity of **Transaction verification.** It handles the correctness of transaction itself
  - Regbank reg R/W behavior is already *guaranteed*

Case 2 : Jasper FPV + CSR

Reduce Complexity – Strategy

FPV
Formal Property Verification App

- Complexity of **Transaction verification** assertion after reduction

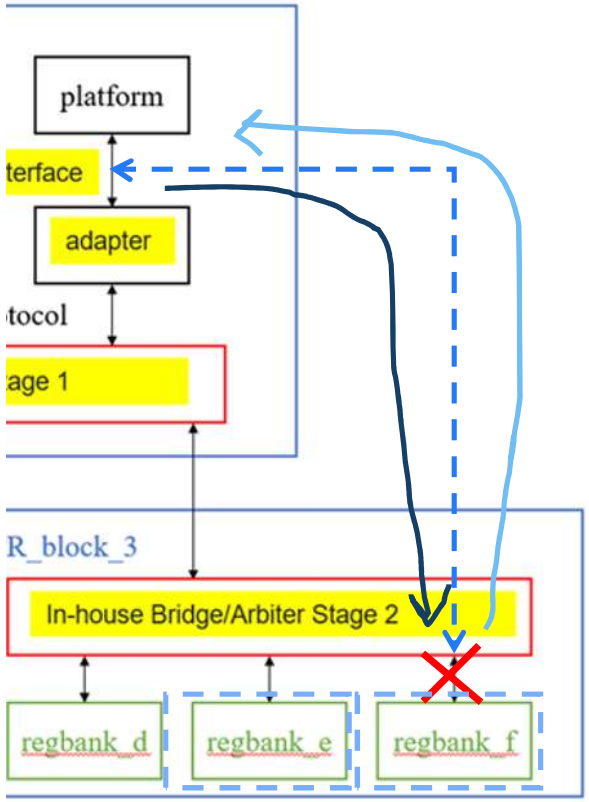
Instances	Gates		Flops	
	Self	Total	Self	Total
chip_top (chip_top)	-	2491121	-	551417
a[redacted]	-	-	-	-
a[redacted]	-	124583	-	69622
a[redacted]	-	1242795	-	267984
c[redacted]	-	1052	-	132
c[redacted]	-	9691	-	1294
d[redacted]	2	16	-	-
i[redacted]	-	-	-	-
i[redacted]	-	-	-	-
o[redacted]	-	-	-	-
p[redacted]	-	-	-	-
r[redacted]	-	-	-	-
s[redacted]	-	-	-	-
s[redacted]	-	1111938	-	212385

Instances	Gates		Flops	
	Self	Total	Self	Total
chip_top (chip_top)	-	182259	-	13020
a[redacted]	-	-	-	-
a[redacted]	-	62720	-	5282
a[redacted]	-	41307	-	5150
c[redacted]	-	352	-	44
c[redacted]	-	3030	-	435
d[redacted]	2	9	-	-
i[redacted]	-	-	-	-
i[redacted]	-	-	-	-
o[redacted]	-	-	-	-
p[redacted]	-	-	-	-
r[redacted]	-	-	-	-
s[redacted]	-	-	-	-
s[redacted]	-	73795	-	2109

Case 2 (FPV) : Transaction verification

3 types of properties

FPV
Formal Property Verification App



- For the transaction verification, we verifies the transaction from top AMBA to the Regbank interface by **Jasper FPV**
  - **Transaction Check :** Verify the Regbank interface will receive current transaction **if and only if** there is a corresponding AMBA transaction
  - **Write Data :** Verify the write data that receive on Regbank interface is identical with the write data on AMBA interface
  - **Read Data :** Verify the read data that receive on AMBA interface is equal to the read data on Regbank interface

Case 2 (FPV) : Transaction verification

Transaction Check

FPV
Formal Property Verification App

- **Transaction Check**

```
assert -name assert_regbank_RegBank_A._trans_check1 { \
$fell(RegBank hier. bank_access_en_bar                                                    .) |-> \
(       Top AMBA interface.paddr[31:16] == 16'h4a20) && \
(       Top AMBA interface.pwrite == ~RegBank_A. bank_write_bar                           .) && \
(       Top AMBA interface.paddr == RegBank_A. bank_address                               .)}

assert -name assert_regbank_RegBank_A._trans_check2 { \
(       Top AMBA interface.paddr[31:16] == 16'h4a20) && \
$rose(      Top AMBA interface.psel && i_checker.apb4_master.penable) && \
((      Top AMBA interface.pstrb == 4'b1111 &&      Top AMBA interface.pwrite) || ~     Top AMBA interface.pwrite) |-> ##[0:15] \
$fell(RegBank hier. bank_access_en_bar                                                    .) && \
(       Top AMBA interface.pwrite == ~RegBank_A. bank_write_bar                           .) && \
(       Top AMBA interface.paddr == RegBank_A. bank_address                               .)}
```

Case 2 (FPV) : Transaction verification

Write Data/Read Data

FPV
Formal Property Verification App

- **Write Data**

```
assert -name assert_regbank_RegBank_A._wdata_check { \
$fell(RegBank hier. bank_access_en_bar                                    ) && \
~RegBank_A. bank_write_bar                                                 . |-> \
RegBank_A. bank_write_data                                                 . == Top AMBA interface.pwdata}
```

- **Read Data**
  - Using **Non-deterministic constant** concept to cover all possible read_data
  - Assume RegBank's read_data is stable in read transaction (**Need to prove help assertion in advance**)

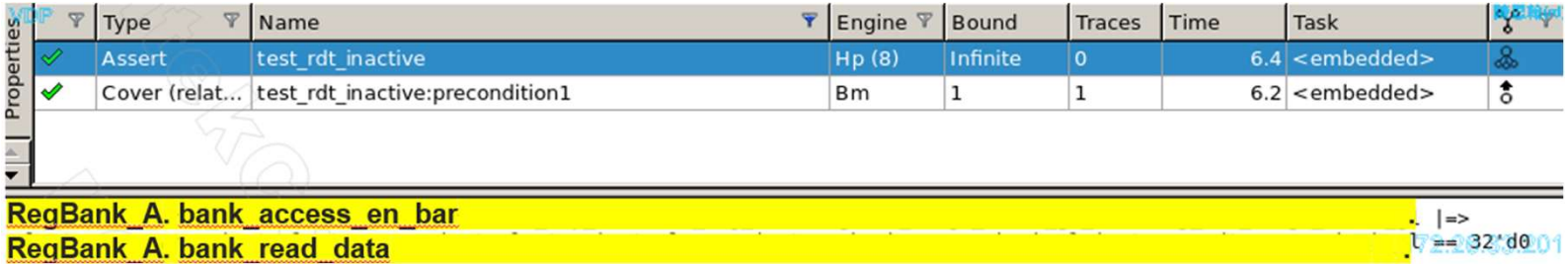
```
stopat -task <embedded> RegBank_A.bank_read_data

virtual_net Jasper_net                  -width 32
assume -name assume_free RegBank_A     _rdata_stable {$stable(Jasper_net                  )}

assume -name assume_range RegBank_A.data_stable { \
~RegBank hier. bank_access_en_bar                                          && \
~($fell(RegBank hier. bank_access_en_bar                                   )) |=> \
Jasper_net                  . == RegBank_A. bank_read_data                 }

assert -name assert_RegBank_A     _rdata_check { \
(      Top AMBA interface.paddr[31:16] == 16'h4A20) && \
$rose(      Top AMBA interface.pready) && \
~      Top AMBA interface.pwrite |-> \
      Top AMBA interface.prdata[i_checker.verify_bit] == Jasper_net                  .[i_checker.verify_bit]}
```

Case 2 (FPV) : Transaction verification

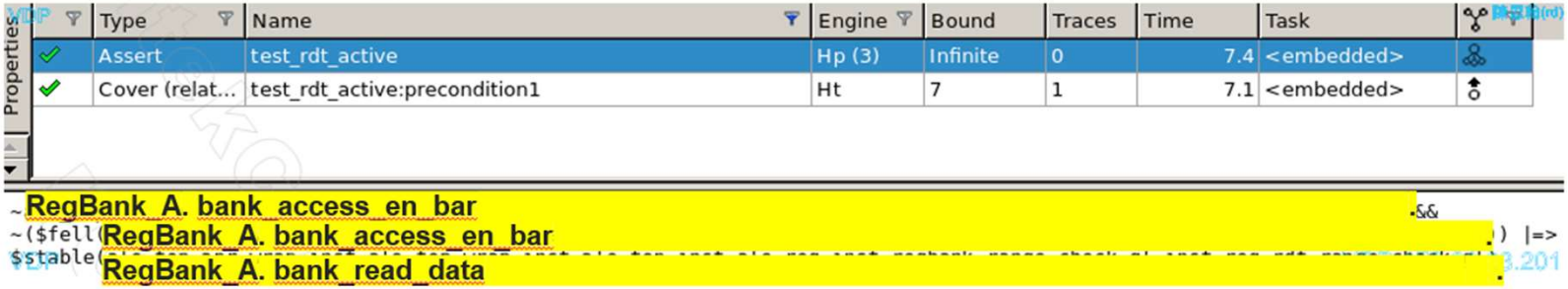
Helper assertions (assume-guarantee)

FPV
Formal Property Verification App

Regbank Inactive : bank_read_data always to be 0



Regbank Active : bank_read_data stable



Case 2 : Jasper FPV + CSR

Result Summary

FPV
Formal Property Verification App

- CSR : Regbank verification
  - Based on the specific format of the auto-generated reg spec, a script can be used to easily generate the CSV file
  - Even the Regbank RTL is auto-generated, it is still with some unexpected behaviors from spec ambiguity

# of Mod/Field	Runtime	RTL bug	Spec issue
100/14659	**30 HRs**	0	6

- FPV : Transaction verification

# of Instance	# of Assertion	# of helper Assertions	Runtime	Spec/Script issue
375	2648	750	**25 HRs**	1

- Expected use model is to fire a run on Friday, check result on next Monday :)

Thank you for listening