

# Formal Verification Sign-off Methodology for Pin Multiplexing Based on Jasper CONN

Xubao.Wang (xubao.wang@unisoc.com)

Chengzhe.Li (chengzhe.li@unisoc.com)

Xufeng.Zhang (xufeng.zhang@unisoc.com)

# Catalogue

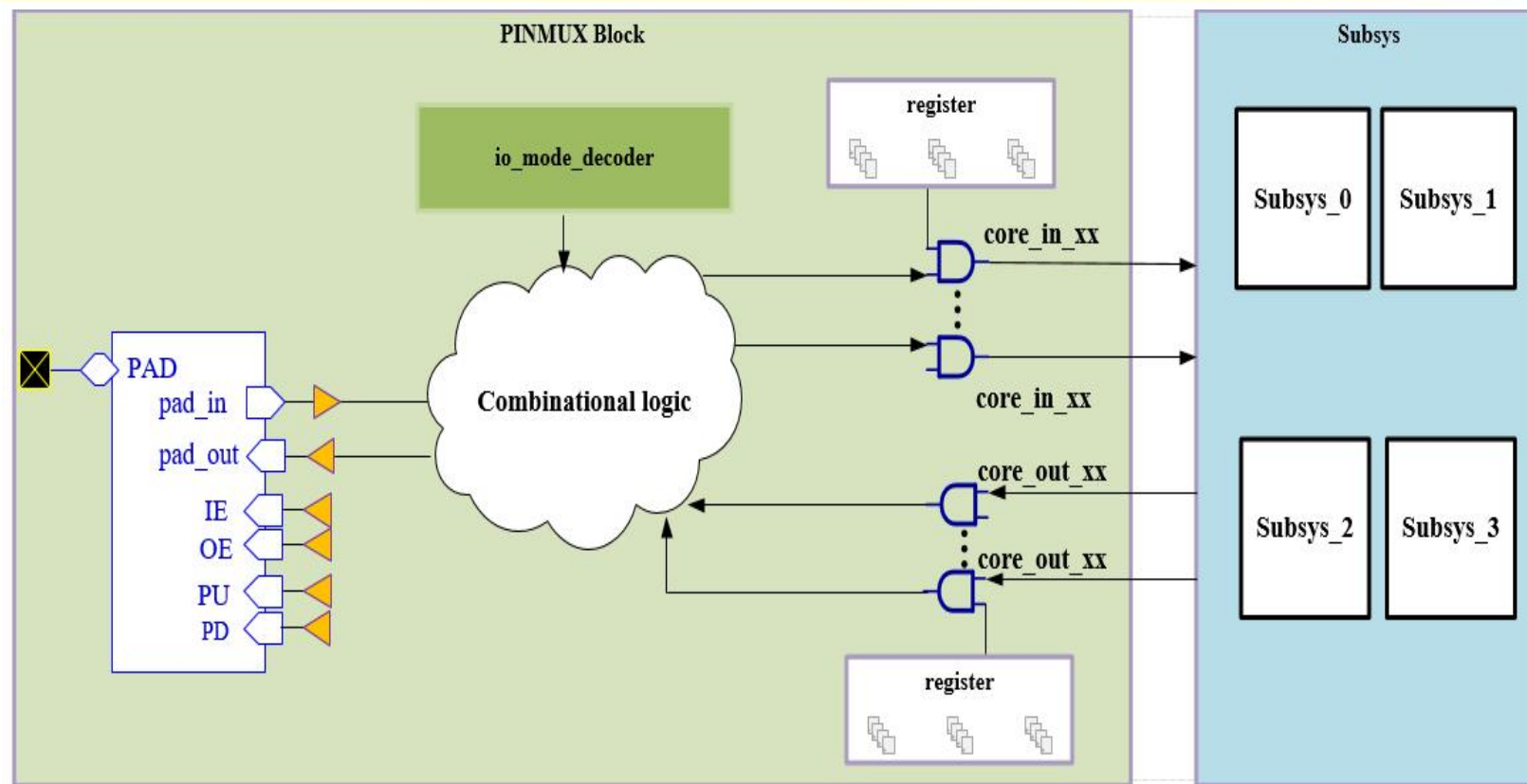
---

**01 Pinmux Simulation Verification**

**02 Pinmux Formal Verification**

**03 Comparison and Summary**

# Pinmux Design Architecture



## Design Feature:

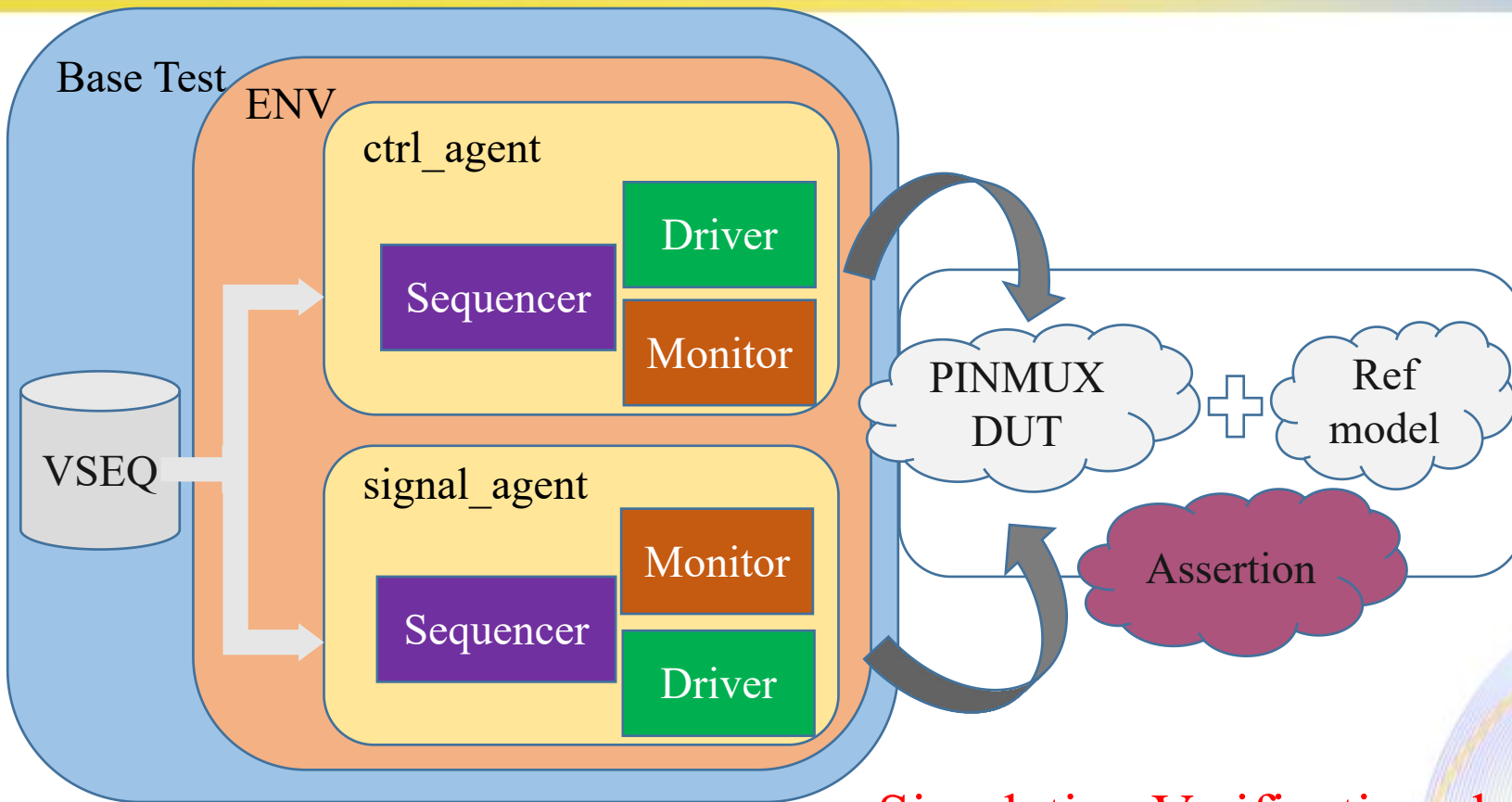
- ① PAD cell: instance pad model
- ② Mode decoder: various mode confirmed
- ③ Combinational logic: logic selection with basic logic gates
- ④ Registers: control of pull-up/pull-down, sleep mode, and ie/oe enable

## Verification plan:

- a) Block Level: Check the connection between PAD and core inside PINMUX
- b) Soc Level: the connections between PINMUX and subsystems within the SoC



# Simulation Verification Solution

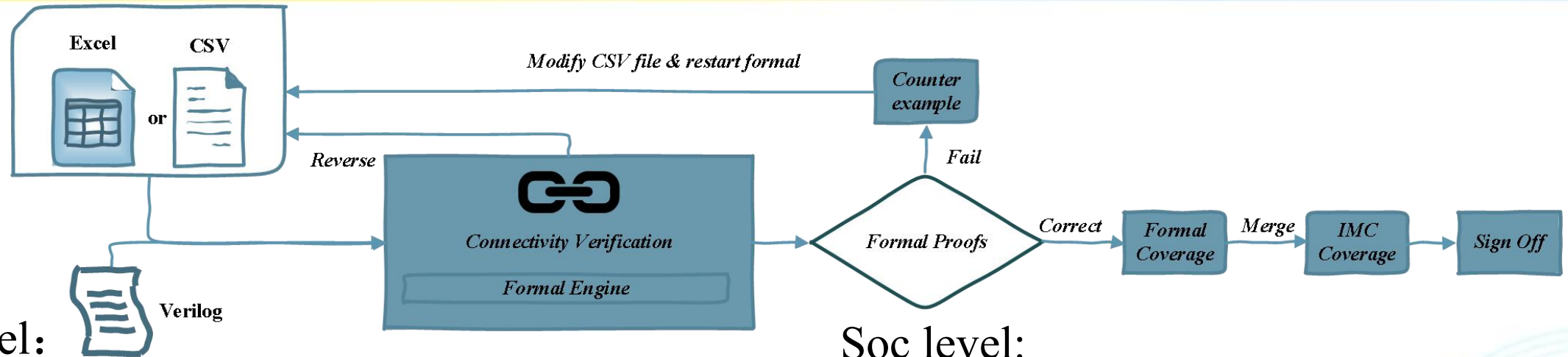


- I. **ctrl\_agent**: configure the PINMUX module to a specific operational mode
- II. **signal\_agent**: generate data transactions;
- III. **Assertion**: checks the reference signal and DUT output; (automatically generated by python)

## Simulation Verification challenges:

1. A single simulation cannot exhaustively cover all incentives, need a large number of random test cases;
2. Coverage collected is time-consuming and inefficient with random transactions (now simAI can help)

# Formal Verification Solution



Block level:

Soc level:

➤ Excel sheet or a comma-separated-value (CSV) file; The generation of csv files is fully automated by python script;

➤ wire reverse extraction from analyzed and compiled RTL, directly output an Excel file or csv for review

➤ Register-Transfer Level (RTL)

```
check_conn -reverse -src {pinmux} -dest {subsys_0} -complexity (straightforward | simple | conditional | complex) -save_as reverse.csv
```

1 NAME, SRC BLOCK, SRC SIGNAL, DEST BLOCK, DEST SIGNAL

2 CONNECTION, a\_0,chip\_top,GPIO\_xx, IO\_MUX\_CORE, core\_in\_xx

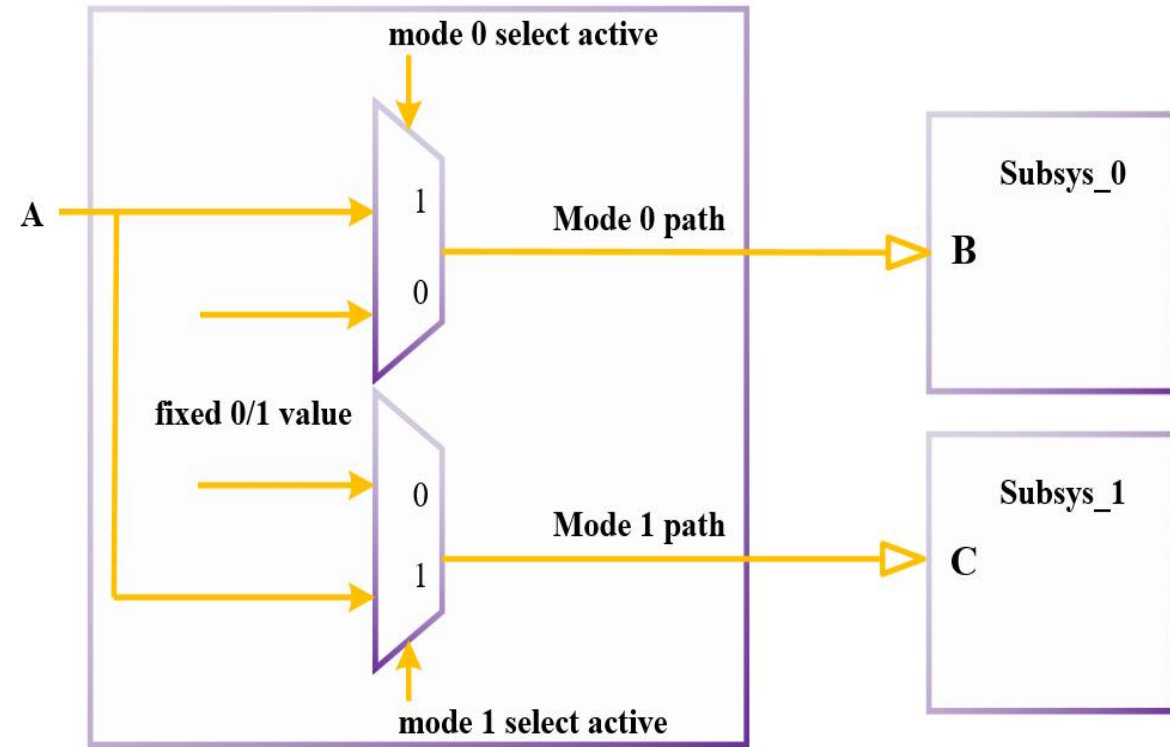
3 ,COND\_EXPR, ptest==1&&PAD\_xx==0



(ptest==1&&PAD\_XX==0) |-> (chip\_top.GPIO\_xx ==  
IO\_MUX\_CORE.core\_in\_xx)

# Pinmux Verification Completeness

a single PAD has input functions across multiple modes, the CSV for one input function must reflect checks for fixed values of signals under other modes



1 NAME, SRC BLOCK, SRC SIGNAL, DEST BLOCK, DEST SIGNAL

2 CONNECTION, b\_0,chip\_top, A, Subsys\_0, B

3 ,COND\_EXPR, mode 0 = 1 && mode 1=0

1 NAME, SRC BLOCK, SRC SIGNAL, DEST BLOCK, DEST SIGNAL

2 CONNECTION, b\_1,chip\_top,A, Subsys\_1, 0

3 ,COND\_EXPR, mode 0 = 1 && mode 1=0



# Formal Prove Result

File Edit View Design Application Window Help

Connectivity Veri... Coverage

File Design Setup Conn. Setup Formal Verification Search

M Search the Message Log

Connectivity Viewer

Worksheet Browser Connections Table

Toggle	COI	Connection Name	Source	Destination	Classification
✓		p_1			partial equal name
✓		p_2			partial equal name
✓		p_3			partial equal name
✓		p_4			partial equal name
✓		p_5			partial equal name
✓		p_6			partial equal name
✓		p_7			partial equal name
✓		p_8			none
✓		p_9			partial equal name
✓		p_10			partial equal name
✓		p_11			partial equal name

session\_0

```

2025-04-03 17:52:02: INFO (ICOV030): Completed measure for task {Connectivity}.
[Connectivity] %
[Connectivity] % # # # save coverage database
[Connectivity] % # check cov -save ./cov db/cov.${conn target}.db -force
[Connectivity] %
[Connectivity] %
[Connectivity] % set time end [tcl clock seconds]
1743673923
[Connectivity] % puts "[CDNS INFO] Total Time : [expr $time end - $time start] seconds"
[CDNS_INFO] Total Time : 435 seconds
[Connectivity] %

[Connectivity] %
    
```

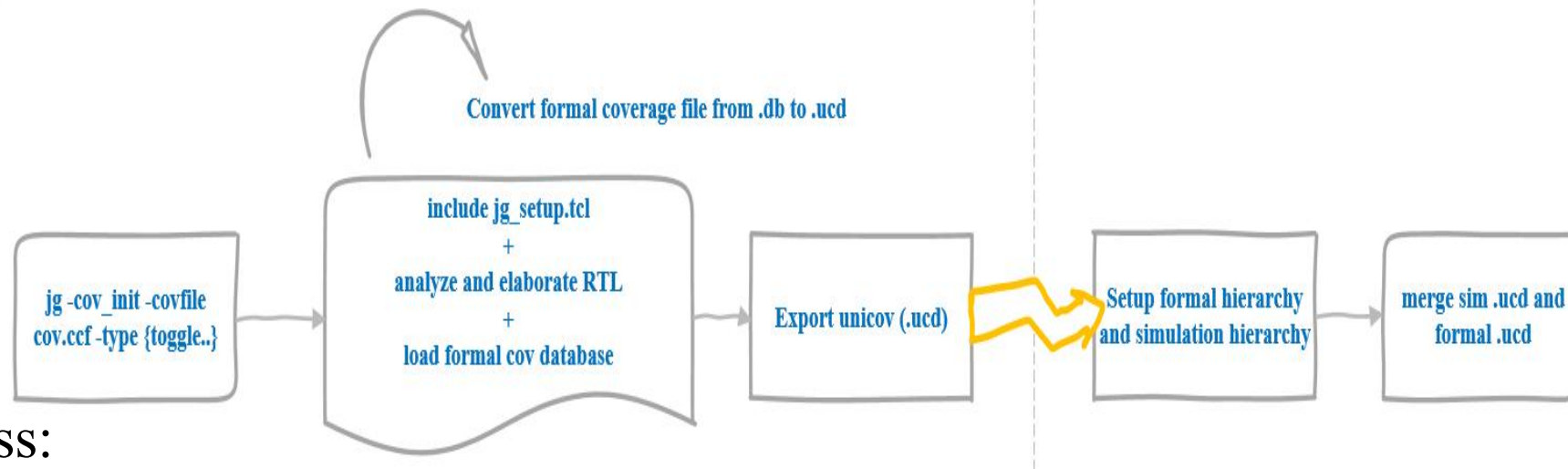
Console Lint Messages Warnings / Errors Proof Messages

Total: 37 Filtered: 37 Selected: 0 Validity: 36:0:0:1 Run: 1:0:0:36

No proofs running Console input ready

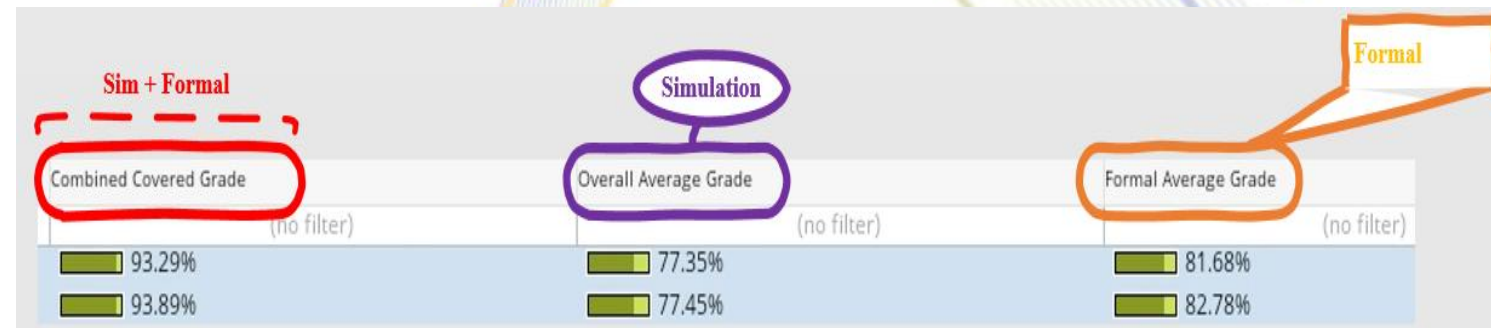
Toggle	COI	Connection Name	Source	Destination
✗		e_33		

# Coverage Merge Flow



Merge process:

1. First, convert formal's .db to .ucd;  
(database –export\_unicov)
2. Specify source (formal\_top) and target  
(sim\_top)
3. merge /sim\_cov/cov /formal/cov/





# Difficulties and Breakthrough Achieved

Compile need lots of time



Black Box  
Assistant  
(BBA)

## 1. BBA

automatically add modules that are not related to the source signal and DEST signal path to the black box list

```
set_balckbox_assistant_handle_bbox true
```

```
set_balckbox_assistant_require_control_paths_in_connections true
```

```
set_balckbox_assistant_require_clock_paths_in_connections true
```

```
blackbox_assistant -config -connectivity_map jg_${conn_target}.csv  
-export [get_proj_dir]/.bballist
```

# Difficulties and Breakthrough Achieved

BBA will cause the previously passed assertion to fail

```
#Jasper bba proc
```

```
include jg_proc.tcl
```

```
#Jasper setup
```

```
include jg_setup.tcl
```

```
#cov init
```

```
include jg_cov_init.tcl
```

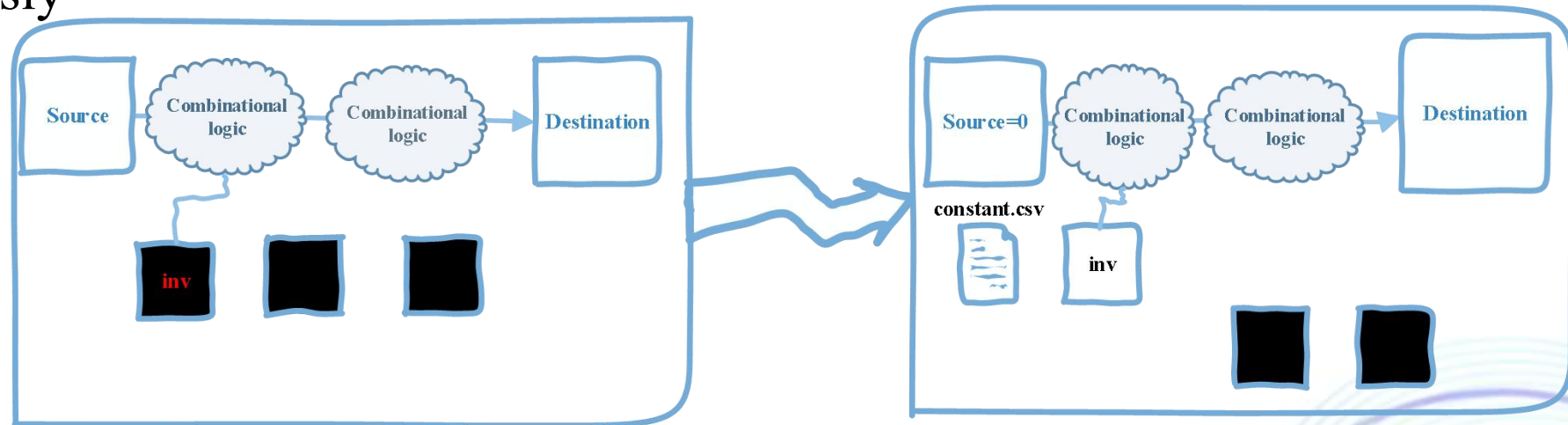
```
#analyze + elaborate design
```

```
include jg_read_design.tcl
```

```
#load conn CSV file
```

```
check_conn -load ${conn_target}.csv
```

```
enable_bba ${conn_target}.csv
elaborate ...
```



The solution involves setting the input source signal as a constant during the verification phase

```
1 NAME, SRC BLOCK, SRC SIGNAL, DEST BLOCK, DEST SIGNAL
```

```
2 CONNECTION, a_0,chip_top,GPIO_xx, IO_MUX_CORE, core_in_xx
```

```
3 ,COND_EXPR, ptest==1&&PAD_xx==0
```

```
1 NAME, SRC BLOCK, SRC SIGNAL, DEST BLOCK, DEST SIGNAL
```

```
2 CONNECTION, a_0,chip_top,0, IO_MUX_CORE, core_in_xx
```

```
3 ,COND_EXPR, ptest==1&&PAD_xx==0
```

`${conn_target}.csv`

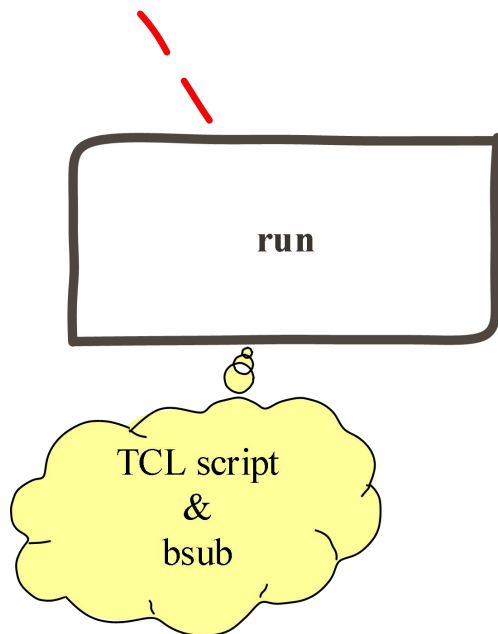
`${conn_target}.constant.csv`

# Difficulties and Breakthrough Achieved

## 2. Parallel verification

a template tcl and leverage the  
bsub command for execution;

Manually run sequentially in Linux terminals



```
clear -all
```

```
set time_start [tcl_clock seconds]
```

```
set conn target CONN TARGET
```

```
#Jasper setup
```

```
include jg_setup.tcl
```

```
#cov init
```

```
include jg_cov_init.tcl
```

```
#analyze + elaborate design
```

```
include jg_read_design.tcl
```

```
#load conn CSV file
```

```
check_conn -load ./csv/${conn_target}.csv
```

```
#clock and reset
```

```
clock -none
```

```
reset -none
```

```
#include conn constraints
```

```
include ./cons/cons.${conn_target}.tcl
```

```
#conn structural check
```

```
check_conn -validate
```

```
#conn functional check
```

```
check_conn -prove
```

```
#switch tab to Connectivity
```

```
check_cov -configure_gui -task  
Connectivity
```

```
#coverage measurement
```

```
check_cov -measure -task Connectivity
```

```
#save coverage database
```

```
check_cov -  
save ./cov_db/cov.${conn_target}.db -  
force
```

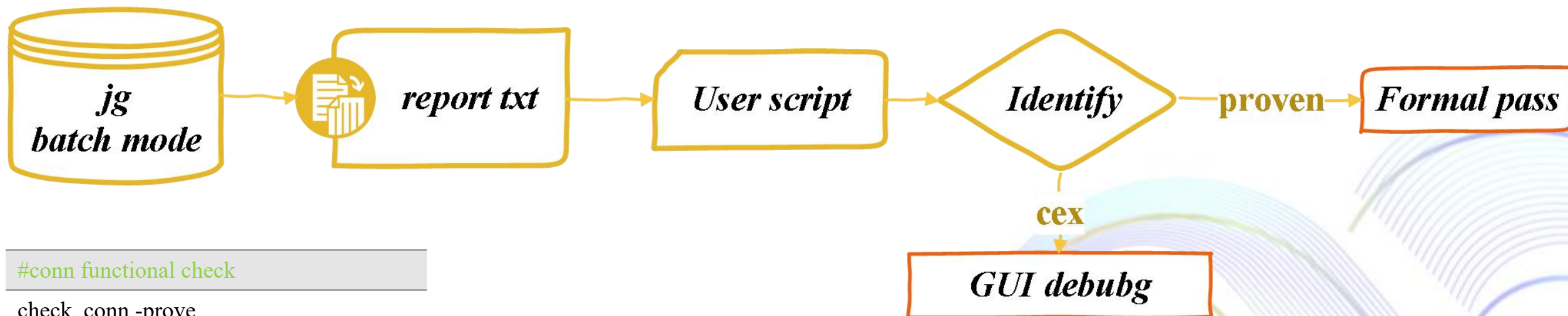
```
set time_end [tcl_clock seconds]
```



# Difficulties and Breakthrough Achieved

## 3. accelerate the convergence of results

identify the formal verification tool has  
fully proven the design in batch mode



#conn functional check

check\_conn -prove

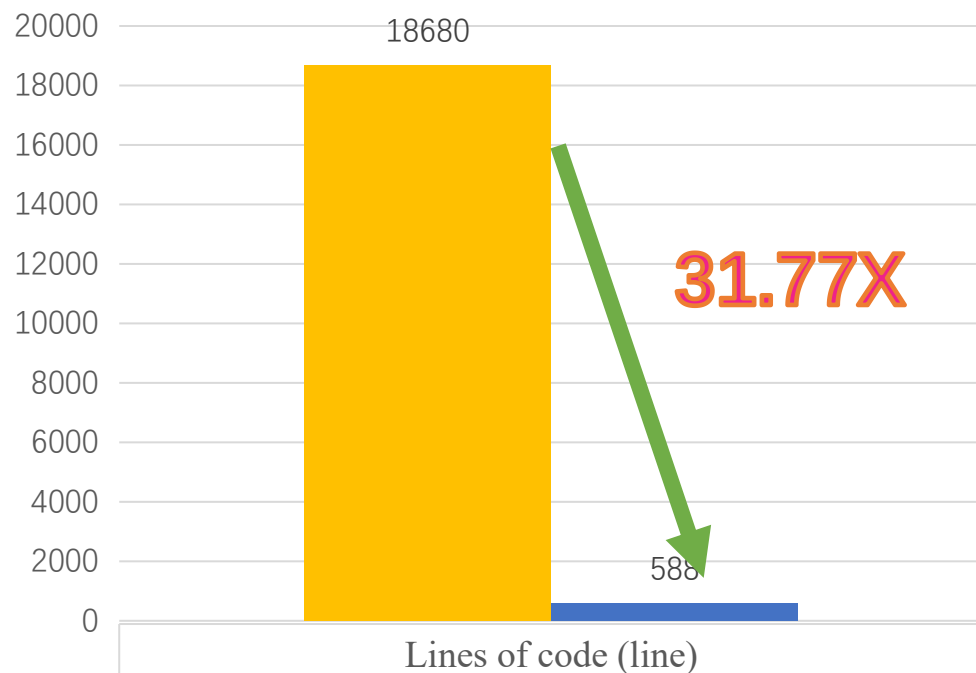
#report verification result

check\_conn -report -file xx.txt -force

CONNECTIVITY REPORT								
Connection Name	Source Block	Source	Destination Block	Destination	Status	Toggle	COI	Classification
o_0					proven	not-generated	Source is inside destination's COI	none
o_10					cex	not-generated	Source is inside destination's COI	none

# Formal & Simulation Comparison

## Efficiency improvement in formal verification

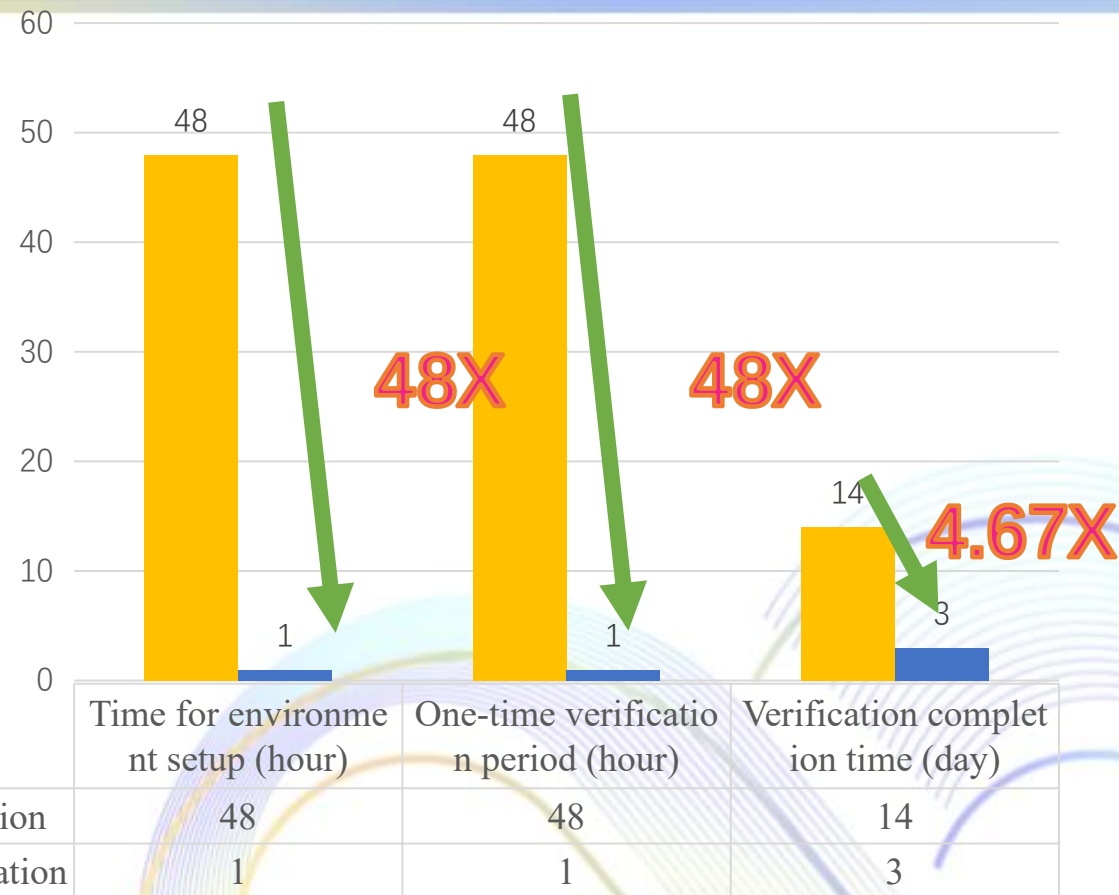


UVM verification  
Formal verification

18680

588

UVM verification  
Formal verification



Time for environment setup (hour)  
One-time verification period (hour)  
Verification completion time (day)

48

1

48

1

14

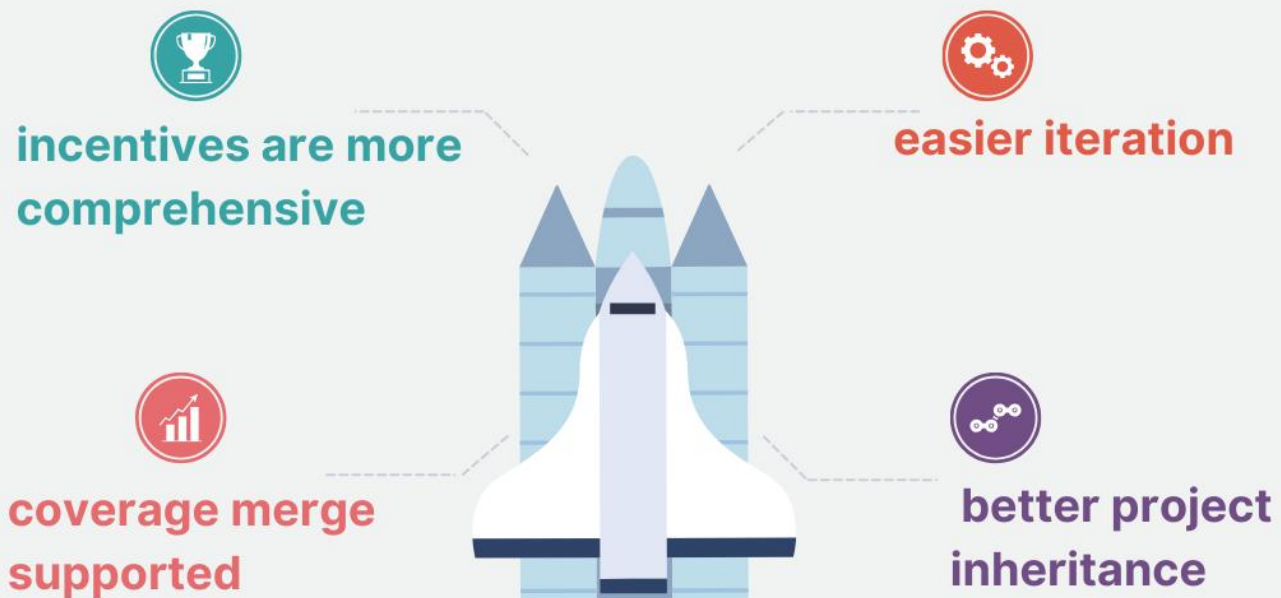
3

16th  
Apr 25  
SHANGHAI

- Formal streamlines the 10,000+ lines of code to just 588 lines;
- Time required for environment setup and a single formal verification within an hour;
- Formal verification complete the whole verification in merely three days.

# Summary

## Formal Verification



Faster



Stronger



The image is a composite of three distinct astronomical or space-themed elements. On the left, a large, brownish-orange planet with prominent rings is shown against a dark, starry background. In the center, a large, glowing sphere with a rainbow-colored, grid-like texture is positioned. On the right, a blue and white nebula or galaxy is visible, with a smaller planet and its rings in the foreground. The word "THANKS" is written in large, white, sans-serif capital letters across the center of the image, overlapping the rainbow sphere and the central part of the planet with rings.

# THANKS

This document is for informational purposes only and the data and information contained in it do not represent any form (express or implied) of warranty or commitment, and the views or conclusions expressed in this document do not constitute any advice. UNISOC reserves the right to make any changes to this document without prior notice. Under no circumstances shall UNISOC be liable for any direct or indirect damage or loss related to the document. The data and information contained in this document are the confidential information and property of UNISOC and/or its licensors. You may not use, copy or distribute any part of this document without prior consent of UNISOC.