2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SAN JOSE, CA, USA
MARCH 4-7, 2024

Automated Formal Verification of a Highly-Configurable Register Generator

Shuhang Zhang, Bryan Olmos, Basavaraj Naik

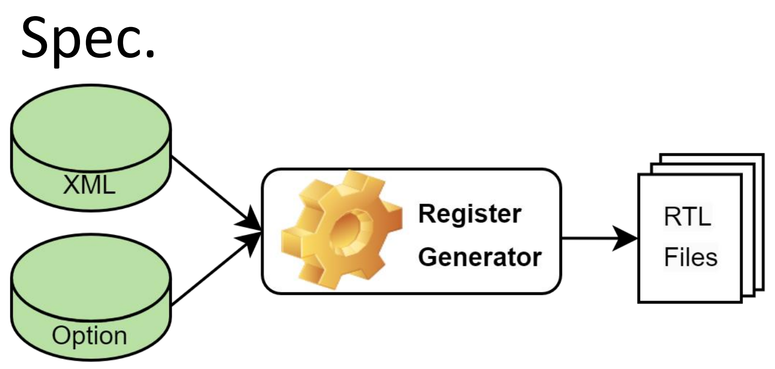
Infineon Technologies AG, Neubiberg, Germany

Outline

- Motivation
- Register Generator
- Verification Challenge
- Automated Formal Verification
- Results
- Conclusion

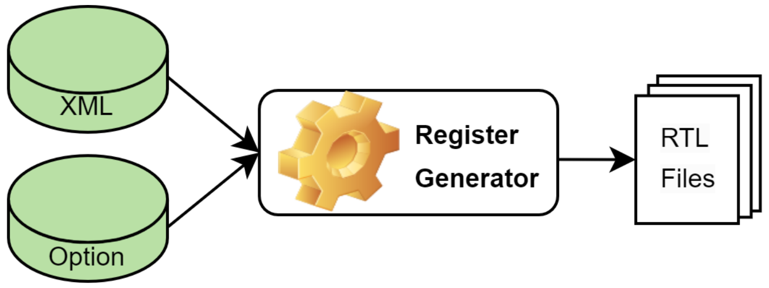
Motivation

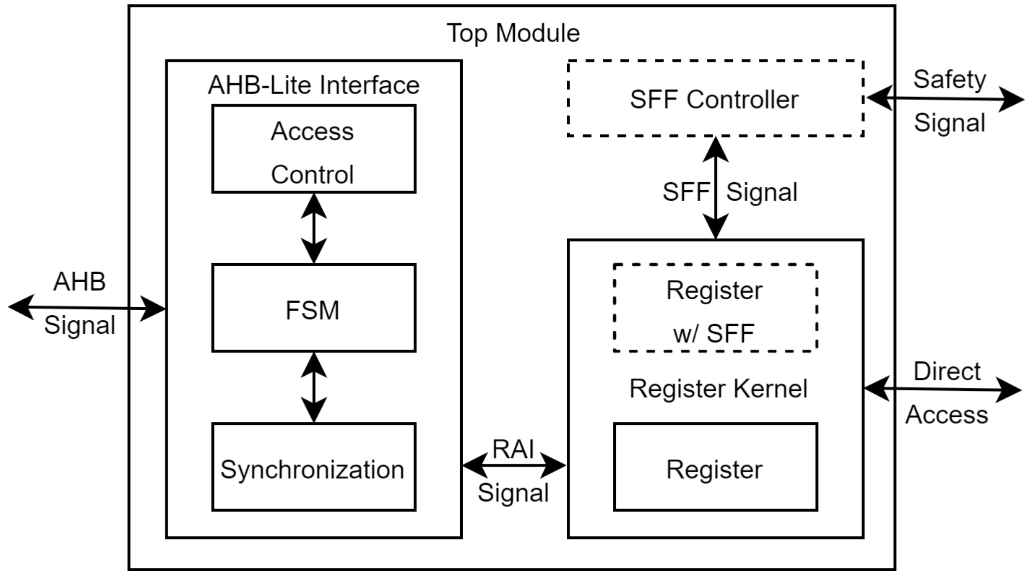
- Number of registers increases
- Ensure chips work correctly
- Low complexity
- Manual implementation error-prone



Implement numerous registers in seconds

Register Generator





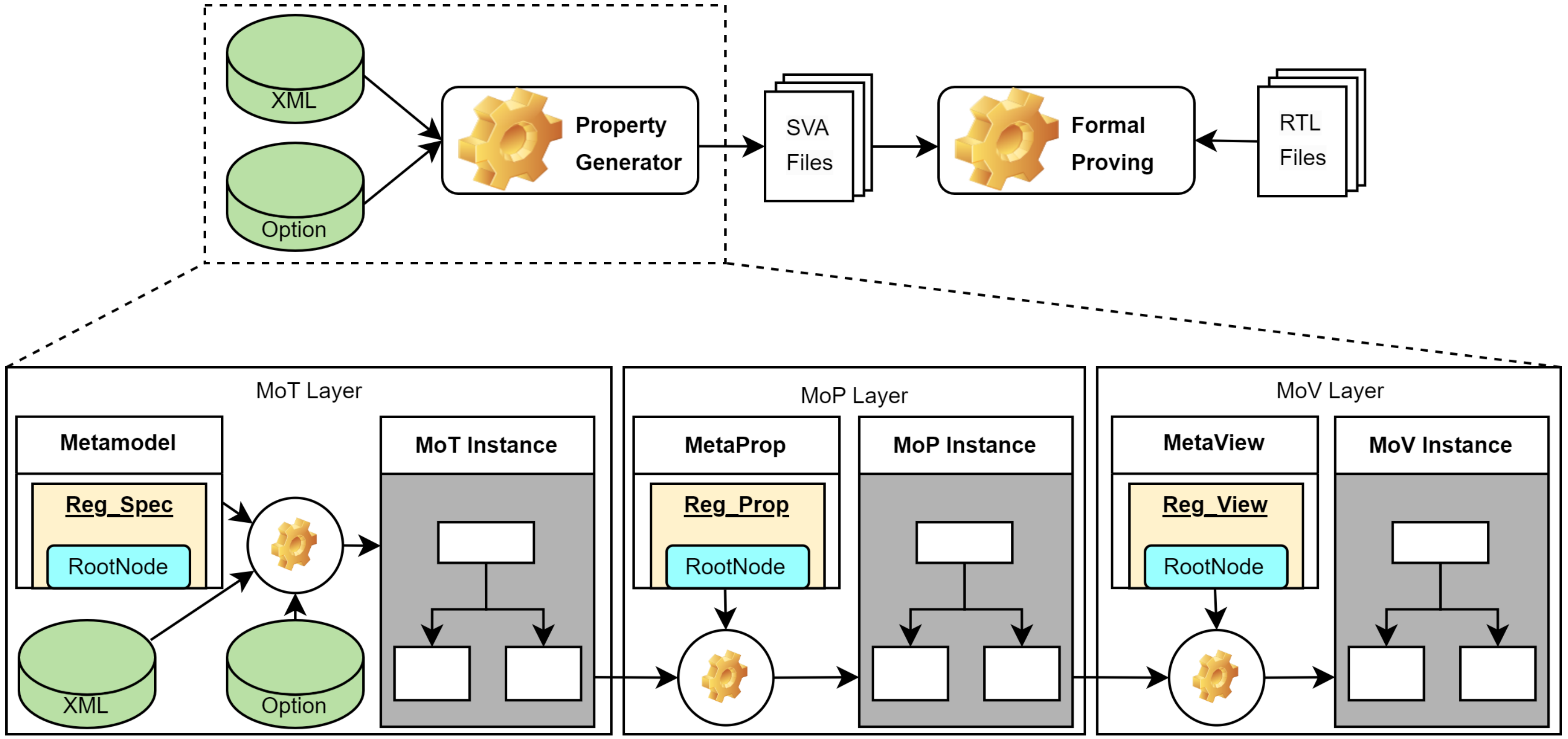
- Bus interface and register kernel
- Optional Safety IP
- 41 options to support various features
  - regAsync (async. Clock for register)
  - regBusClock (register still Bus Clock)
  - regUnrollAHB (unroll AHB signals)
  - …

Verification Challenge

- Configurability
  - 41 different options
- Diverse access policies
  - External access via the bus interface
  - Internal access via other blocks

- **20PD for one design with UVM**
- **Incomplete coverage**
- **Formal fits great**
- **Automated property generation**

Automated Formal Verification



Generation Option Analysis

- 41 generation options -> **Impossible** to verify all combinations

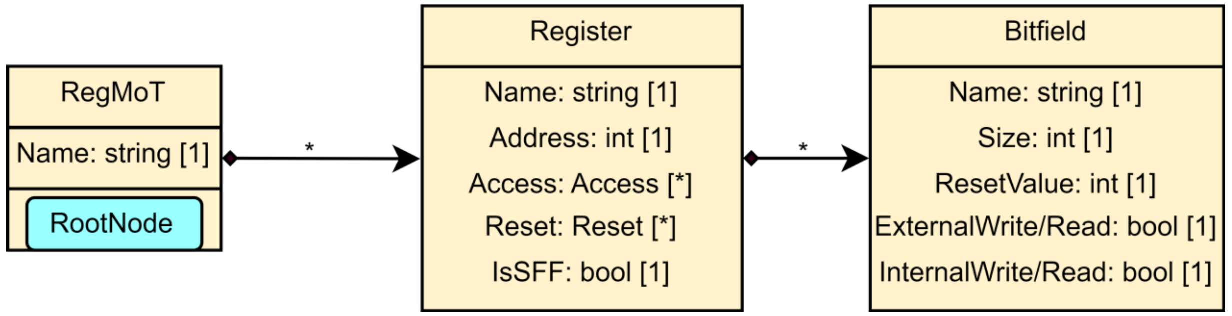
- Group generation options
  - Dependent option
    - regAsync
    - regBusClock

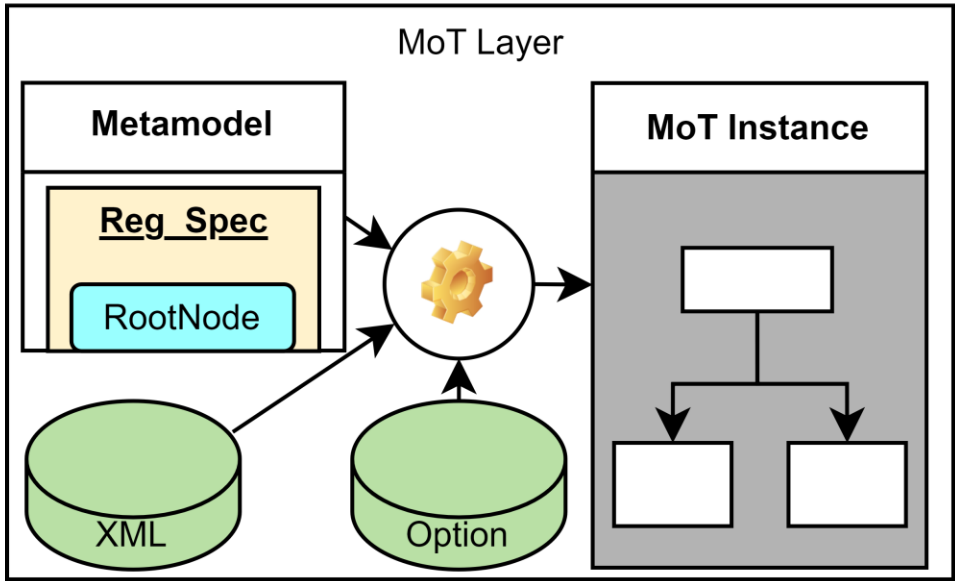
  - Independent option
    - regUnrollAHB

Specification Extraction

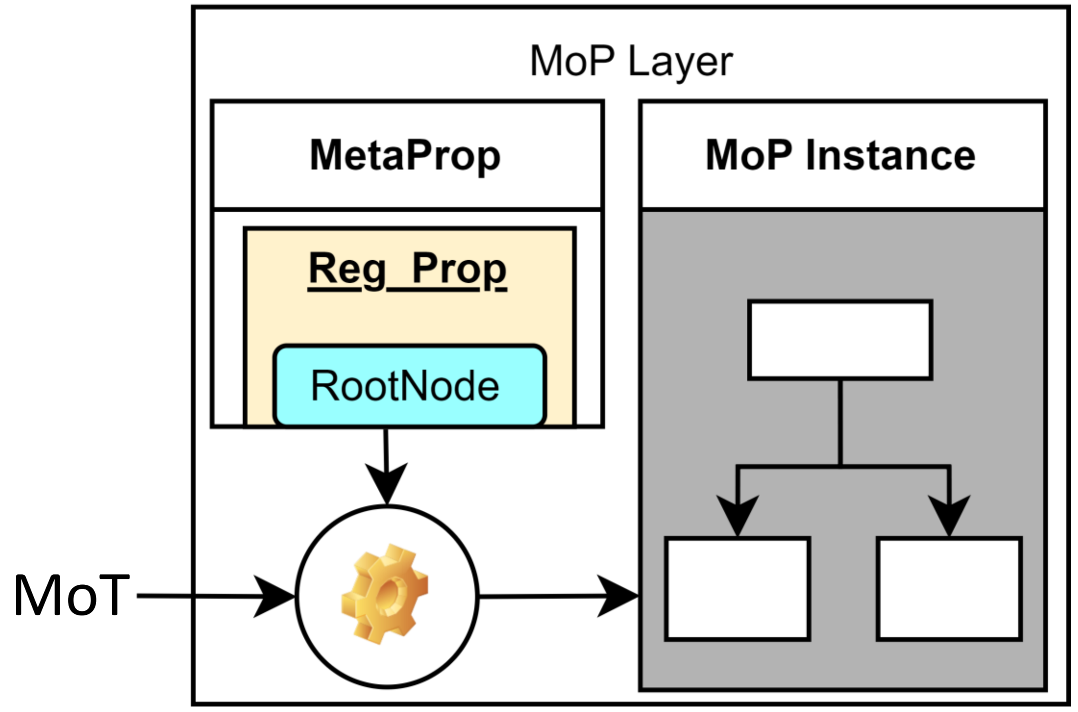
- MetaModel creation
  - Define which spec. are extracted
  - Both register and bitfield level

- Model of Things (MoT)
  - Internal spec.





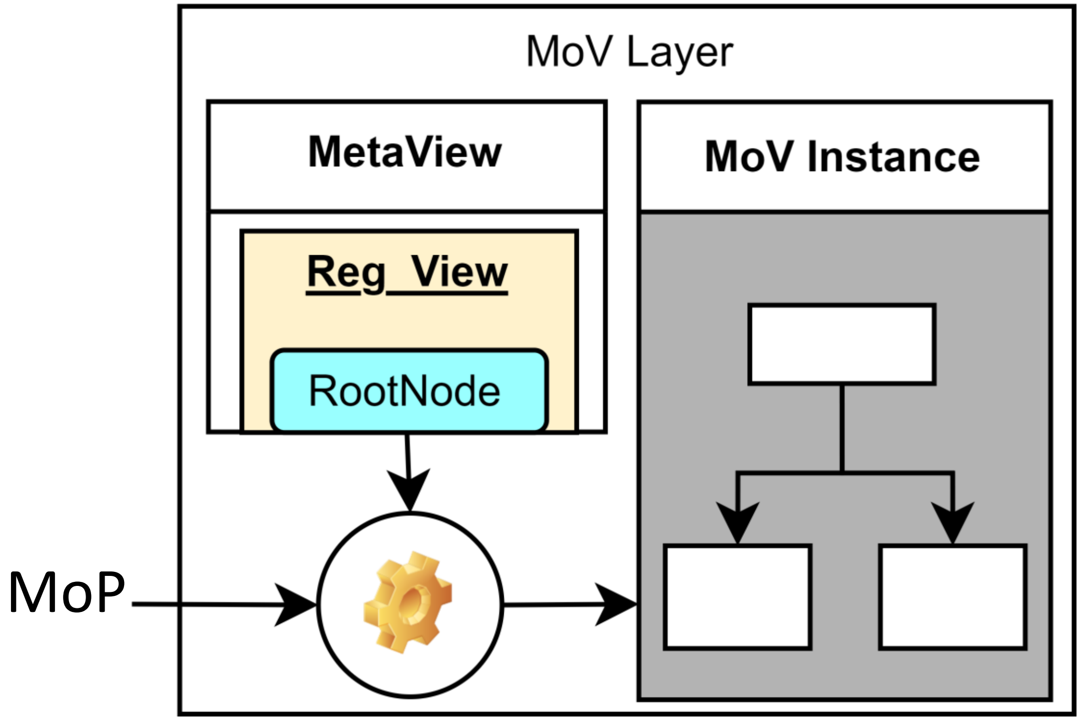
Property Definition



Property class defined using Python

- Bus protocol
- External Read/Write
- Access violations
- Internal Read/Write
- Dummy write
- Integrated Safety IP
- …

Property Generation



External Write:

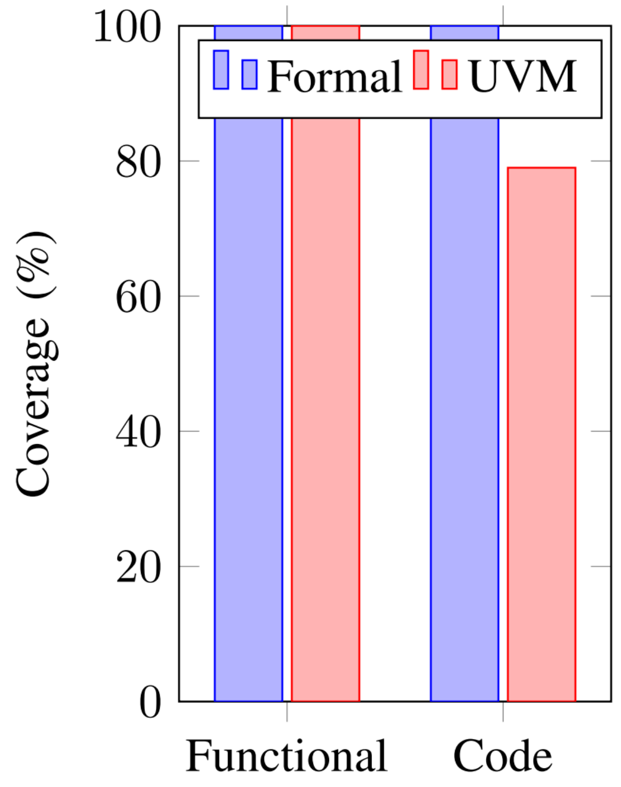
```
property reg_external_write;
@(posedge hclk_i)
disable iff(!hreset_n_i)
... && sx_ahb_lite_slave.haddr == 32'h0000_0002 &&
sx_ahb_lite_slave.hwrite == 1'b1 &&
debug_on_i == 1'b1;
|-> ##1
sx_ahb_lite_slave.hwdata == register_kernel_inst.reg_bf_value;
endproperty
chk_reg_external_write_assert: assert property(reg_external_write);
```

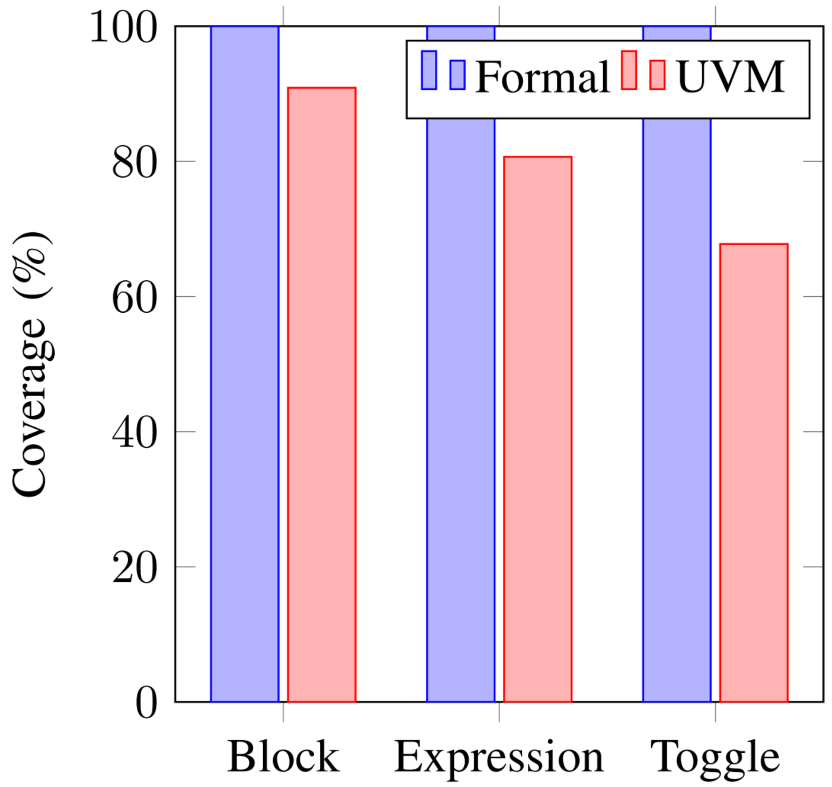
External Write Error:

```
property reg_external_write_error;
@(posedge hclk_i)
disable iff(!hreset_n_i)
... && sx_ahb_lite_slave.haddr == 32'h0000_0002 &&
sx_ahb_lite_slave.hwrite == 1'b1 &&
debug_on_i = 1'b0;
|-> ##1
sx_ahb_lite_slave.hreadyout_o == 1'b1 &&
sx_ahb_lite_slave.hresp_o == 1'b0;
endproperty
chk_reg_external_write_error_assert: assert property(reg_external_write_error);
```

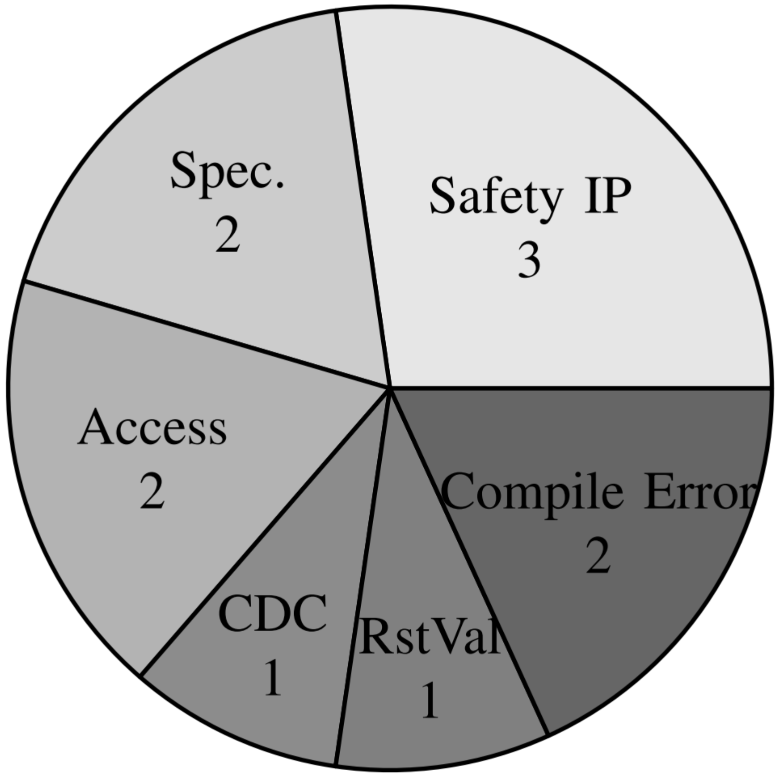
Results: Effort and Coverage

- Flow applied on a productive project
- Effort reduction
  - ~3PD Formal
    - One-time 80PD
  - ~20PD UVM
- Better coverage





Results: Detected Bugs



- Safety IP
  - Connectivity issues
- Spec.
  - Documented wrongly
- Access
  - Protection issues
- CDC
  - CDC check failure
- RstVal
  - Wrong reset value
- Compiling Error

Conclusion

- Great challenges brought by our in-house register generator
- Traditional UVM approach time-consuming
- Automated formal verification is more efficient

Questions?

Thank you!