RISC-V Summit China
August 23-25, 2023

Addressing the Complexities of RISC-V Functional Verification

解决RISC-V功能验证中的复杂性

Tony Wang
王浩为

Valtrix - An Introduction 简介

1. Valtrix - EDA company headquartered in Bangalore, India;
2. Developing products/solutions for CPU/SoC verification
3. STING - First commercially available and most advanced design verification solution for RISC-V
4. Software-driven portable self-checking architecturally correct stimulus generator
5. Used by several RISC-V CPU/SoC companies such as Google, Seagate, Sifive and Tenstorrent for verifying functional correctness and architectural compliance of designs

Agenda Of The Talk 议程

01 Challenges in **RISC-V Design Verification**
RISC- V 设计验证中的挑战

02 Challenges imposed by **DUT Configuration**
DUT配置带来的挑战

03 Popular **Test Generation Methodologies**
现行有效的测试生成方法

04 **Functional Correctness** and **Compliance**
功能正确性和合规性

05 **Test Generation** over **Design Life-Cycle**
在设计周期不同阶段的测试生成方法

06 **Case Studies** for **Test Generation**
案例研究

07 **Conclusion**
结论

CPU/SoC Verification – Challenges
传统芯片验证中会遇到的挑战

1. How many tests to run to ensure comprehensive verification?
2. How to distribute the testing across simulation, emulation, FPGA and silicon for best throughput?
3. How to write test stimulus which is portable across all DUT environments?
4. How to debug failures from silicon quickly on simulations?
5. How to develop a test which runs on simulation and silicon alike?
6. How does test generation scale from top-level simulation test benches all the way to a complete SoC on silicon?

Unique Challenges in RISC-V Verification
RISC-V验证中独有的挑战

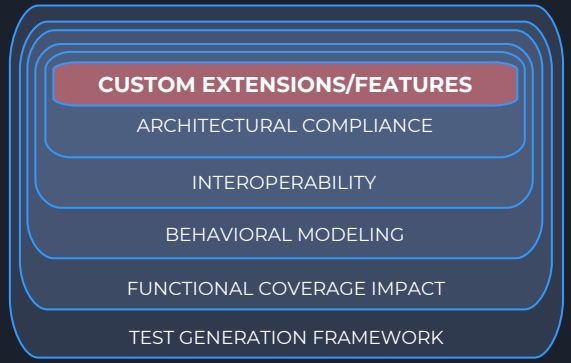
1. Plug and play extension architecture leads to huge number of designs which need to be verified
即插即用的扩展架构导致大量验证需求

- RV32I + M
- RV32IM+ M
- RV32IMC + M
- RV32IMAC + M
- RV32IMAFDC + M
- RV32IMAFDCV + M
- RV32IMAFDCV + M
- RV32IMAFDCV + MU
- RV32IMAFDCV + MSU
- RV32IMAFDCV + MSU + MMU
- RV32IMAFDCV + MSU + MMU + PMP
- and many many many more . . .

Unique Challenges in RISC-V Verification
RISC-V验证中独有的挑战

1. Plug and play extension architecture leads to huge number of designs which need to be verified

2. Custom extensions and features
   自定义指令和特性



Unique Challenges in RISC-V Verification
RISC-V验证中独有的挑战

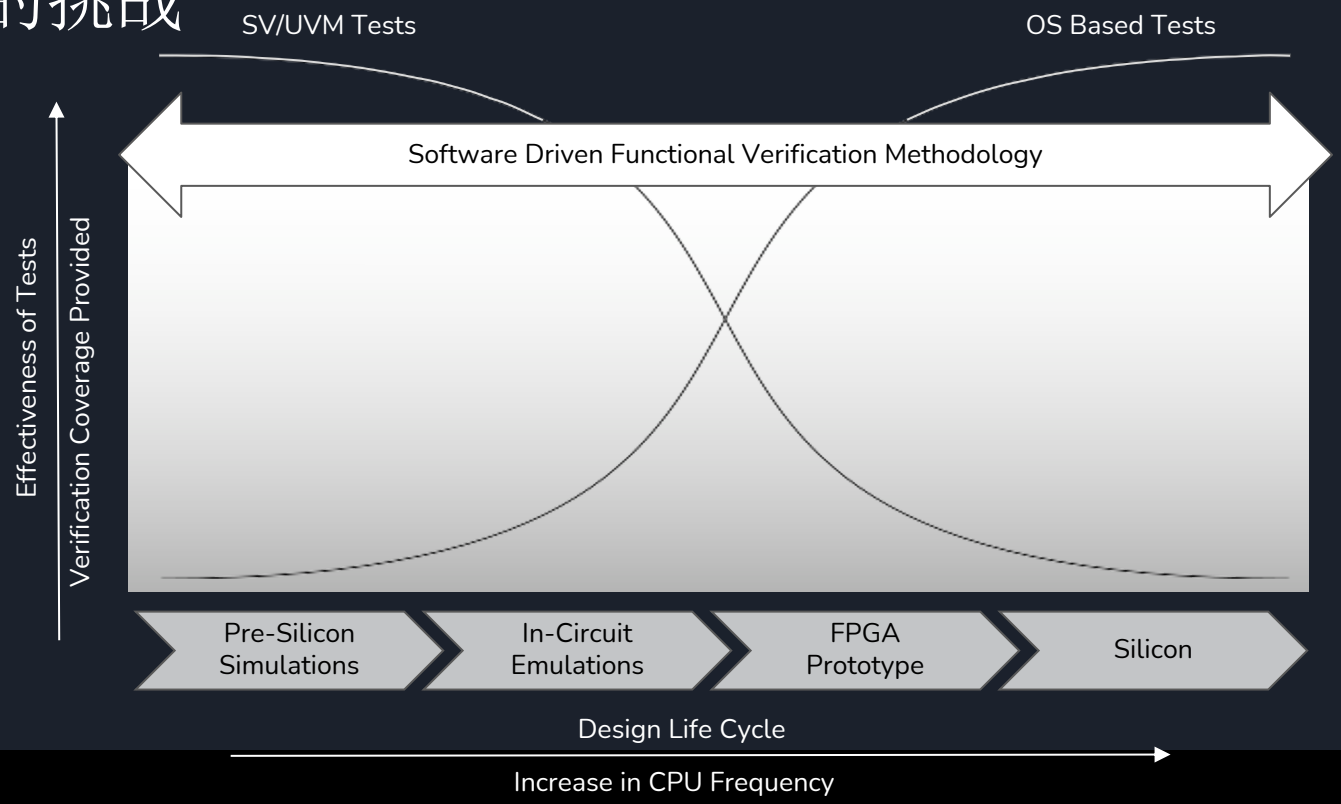
1. Plug and play extension architecture leads to huge number of designs which need to be verified
2. Custom extensions and features
3. Continuously evolving specifications
   不断进化的规格

Challenges imposed by DUT Configuration
DUT配置带来的挑战

1. Limited verification cycles in simulations
2. Emulation, FPGA and silicon based DUT platforms faster but suffer from lack of debug visibility
3. Debug features to reduce time taken to isolate failures
4. Different test methodologies (based on SV/UVM, OS based tools and so on) across different DUT environments increases redundancy and reduces reuse
5. Portability of stimulus
6. Test generator design to address challenges imposed by DUT

Challenges imposed by DUT Configuration
DUT配置带来的挑战

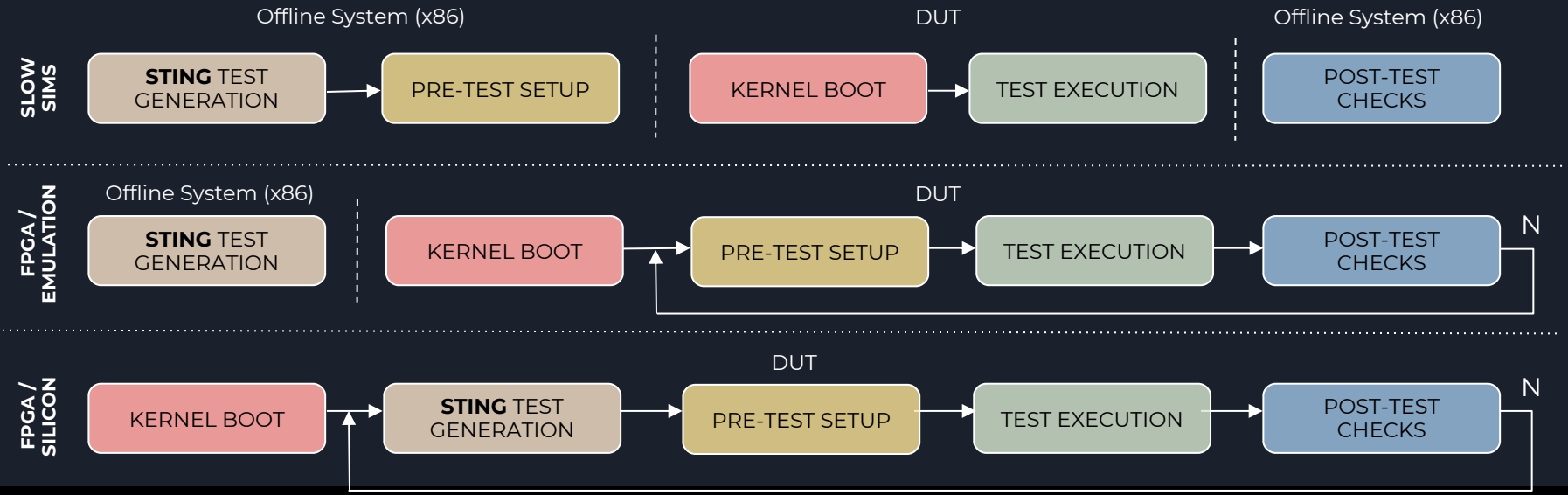
1. Allow consistent execution on all DUT environments
2. Reuse of test stimulus
3. Failures hit on silicon can be migrated to an earlier stage for faster debug
4. Early enabling of software driven stimulus increases the chances of hitting complex bugs early
5. Save on duplicate efforts spent on design verification and system validation



Challenges imposed by DUT Configuration
DUT配置带来的挑战

1. Modular test generator design to achieve maximum test throughput as per the DUT environment

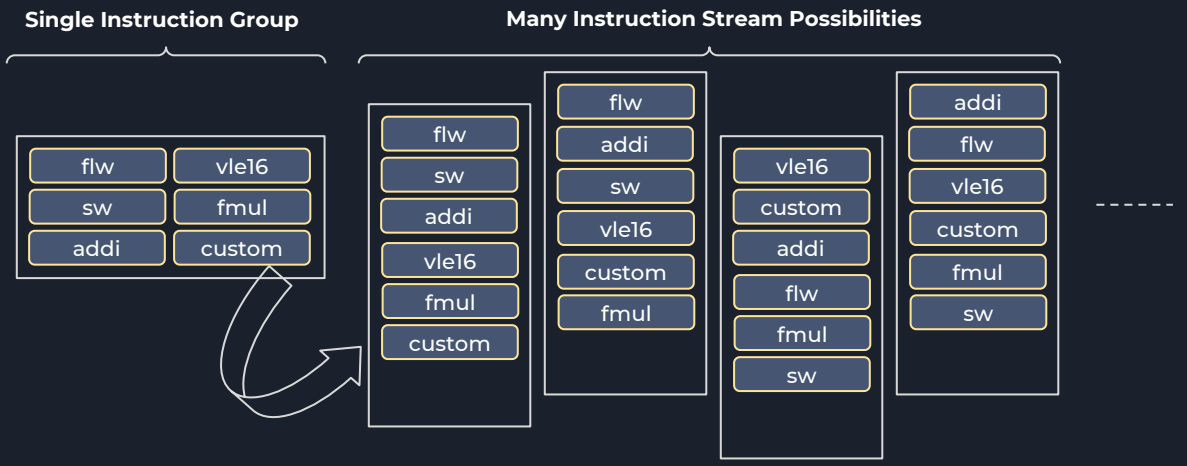
2. Self generating mode on FPGA/silicon enables long runtimes



Popular Test Generation Methodologies
测试生成方法

1. Constrained Random

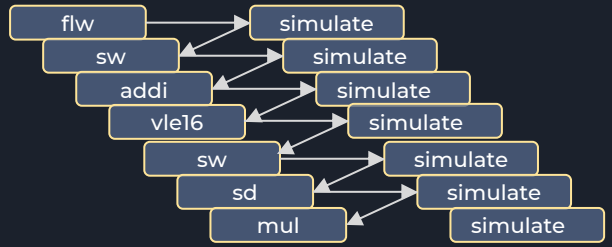
- Randomize within fixed constraints
- Advantageous in sweeping through many random combinations
- Not very good in targeting fixed use cases



Popular Test Generation Methodologies
测试生成方法

1. Constrained Random
2. Dynamic Random

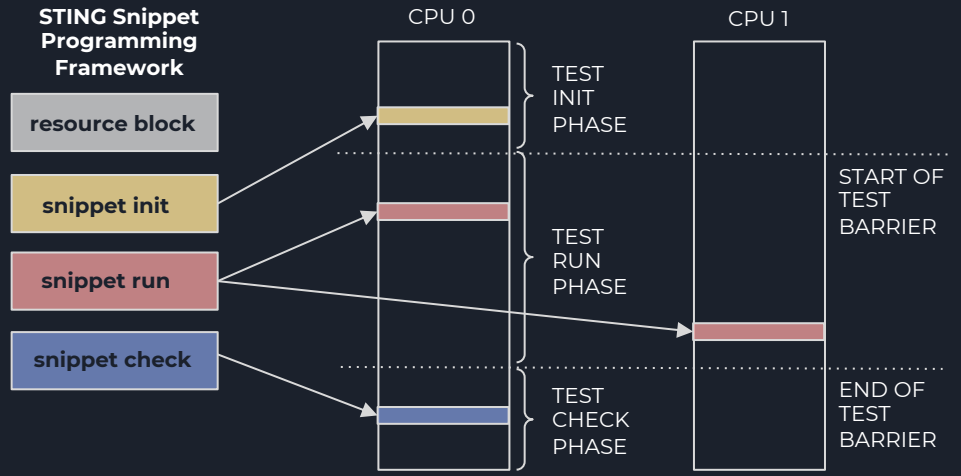
- Simulate as the test is getting generated
- Availability of complete state during test generation helps in targeting more complex scenarios
- Very slow because of simulation overhead and frequent re-evaluations of choices made during test generation



Popular Test Generation Methodologies
测试生成方法

1. Constrained Random
2. Dynamic Random
3. Directed Testing

- Highly directed scenarios can be covered
- Advantageous in covering architectural compliance tests
- High on manual effort and does not scale very well across multiple projects/designs if not written properly

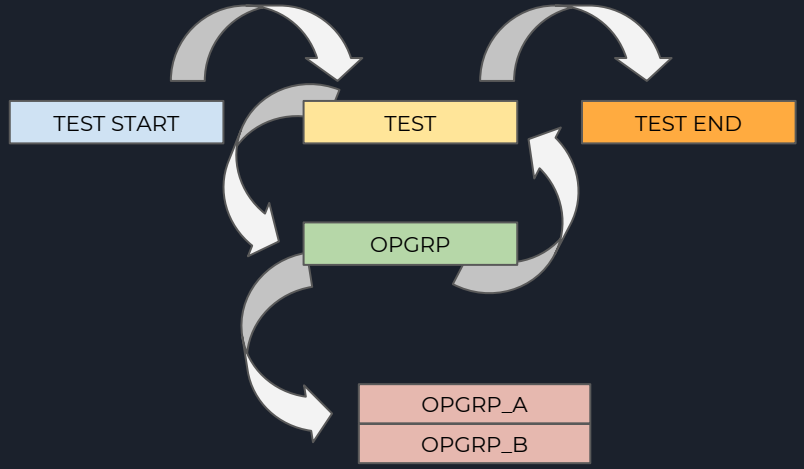


Popular Test Generation Methodologies
测试生成方法

1. Constrained Random
2. Dynamic Random
3. Directed Testing

- Stimulus graph based test generation gives greater control on the shape of the test
- Easily cover scenarios which are otherwise very difficult to achieve using constrained random or are not random enough for directed test cases

04. Graph Based



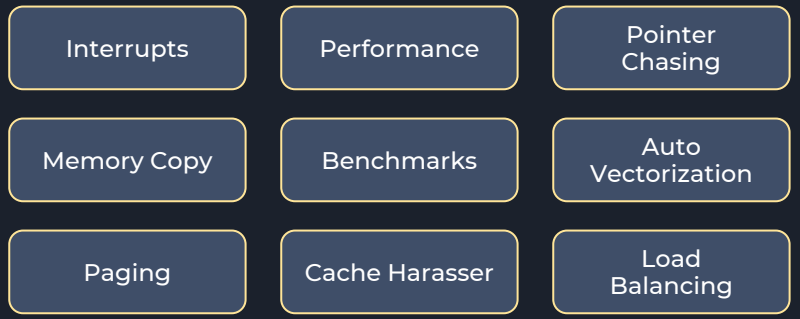
Popular Test Generation Methodologies
测试生成方法

1. Constrained Random
2. Dynamic Random
3. Directed Testing

- Coverage for OS use cases/real world scenarios
- Running them early on simulations help hitting the complex bugs early

04. Graph Based

05. Use Case Testing/Real World Scenarios



Verifying Functional Correctness and Compliance
验证功能正确性和合规性

1. **Lockstep Checking**
   a. Run golden model along with RTL simulation
   b. Check model result vs RTL result
   c. **ADVANTAGES** : Easy debug, extensive checking of memory and registers
   d. **DISADVANTAGES** : Simulation only, Hard to implement for MP

2. **Model Reference Checking**
   a. Model used to generate golden data
   b. Checking at end of test
   c. Portable programs which can be run on any DUT environment
   d. **ADVANTAGE** : No infrastructure work on TB, Fast, Portable (Sim, Emu..)
   e. **DISADVANTAGE** : Slightly higher debug effort, granularity of check is lower

3. **Directed Checks**
   a. Covers compliance and corner case scenarios, designer concerns.
   b. Checks will be part of test
   c. **ADVANTAGE** : Coverage of a given scenario is guaranteed
   d. **DISADVANTAGE** : Difficult to cover all scenarios, Labour intensive

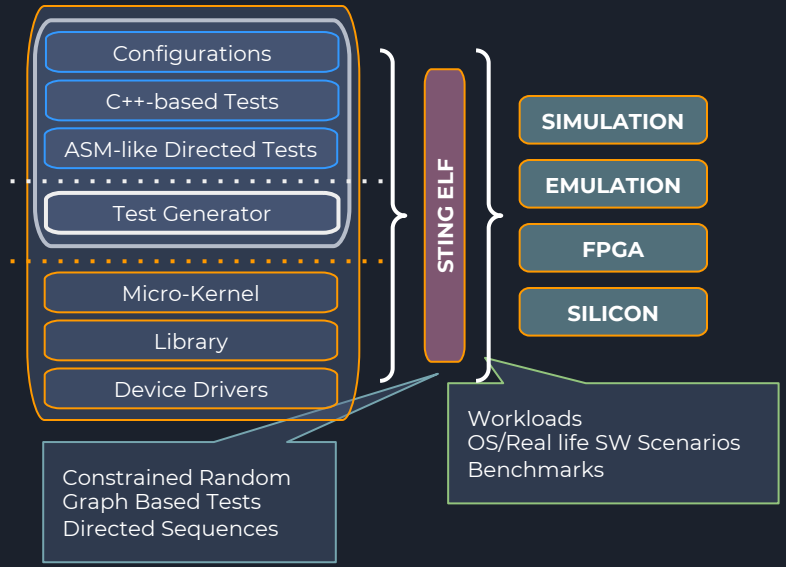
4. **Multi-Pass Checking**
   a. Effective for timing bugs
   b. Consistency checks across multiple passes
   c. **ADVANTAGE** : Very Fast; Instruction simulation is not required.
   d. **DISADVANTAGE** : Not good at finding functional bugs

Test Generation over Design Life Cycle

1. Verify **basic functionality** of each instruction
    a. **Fixed data pattern** tests
    b. **Random data pattern** tests
    c. Check whether **reads** and **writes** can **access memories**
2. Test **interaction between different instructions**
    a. **Pipeline** dependencies
    b. **Control** dependencies
    c. **Address** dependencies
3. Test **interaction with different events**
    a. Exceptions
    b. External interrupts
4. More **complex use cases**
    a. **Interaction with MMU** - page size switching, address bits change, change of memory type
    b. **Algorithmic tests** - Memory Copy, pointer chasing
    c. **Microarchitectural tests** - Cache fill/evictions, self modifying code, cross modifying code
5. Add **traffic from other peripherals** in a SoC test bench
    a. Test **concurrent traffic** from multiple sources

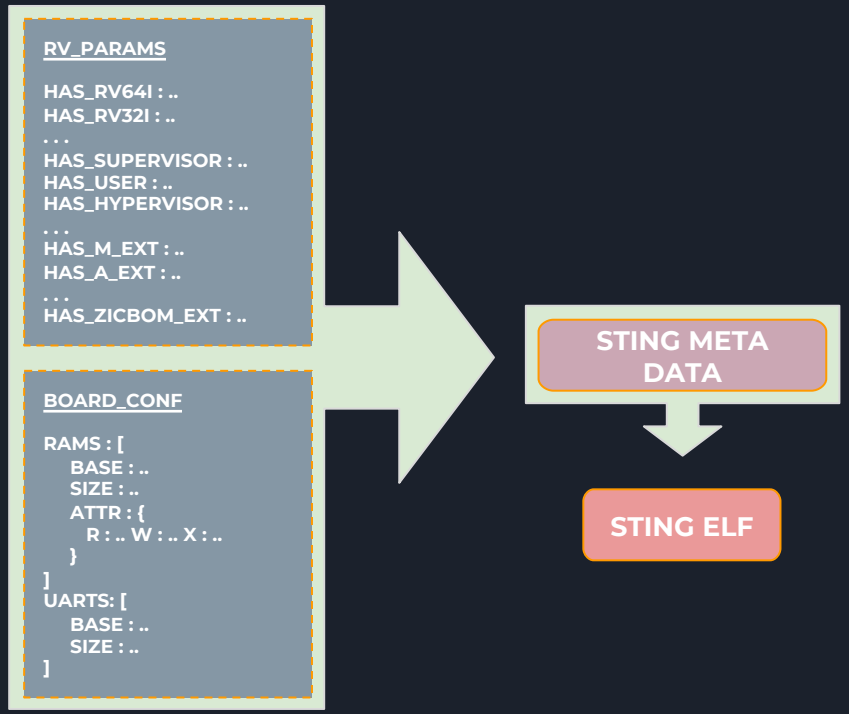
STING Design Verification Tool

1. Bare metal tool using a software driven methodology for RISC-V design verification
2. Self-checking architecturally correct stimulus portable across simulation, emulation, FPGA and silicon
3. Highly scalable and quick test generation; Compatible with any system configuration/memory map; IoT/embedded to server class; MP-ready
4. Complete support for 32-bit and 64-bit RISC-V base integer extensions along with all standard extensions; Supports RVA23 profile
5. Comprehensive coverage of privilege specification areas such as MMU, PMP, Hypervisor



Case Study - Support for any RISC-V configuration
对所有RISC-V配置的支持

1. JSON-like rich configuration mechanism for input specification
2. Every RISC-V extension/feature can be controlled using configuration overrides
3. The configuration overrides determine many things such as the valid directed tests and instructions to select in the test, toolchain options to compile and generate the ELF file and so on
4. Placement of text/data sections randomized based on physical memory attributes specified in the memory map
5. Any RISC-V implementation can be tested without worrying about tool compatibility issues

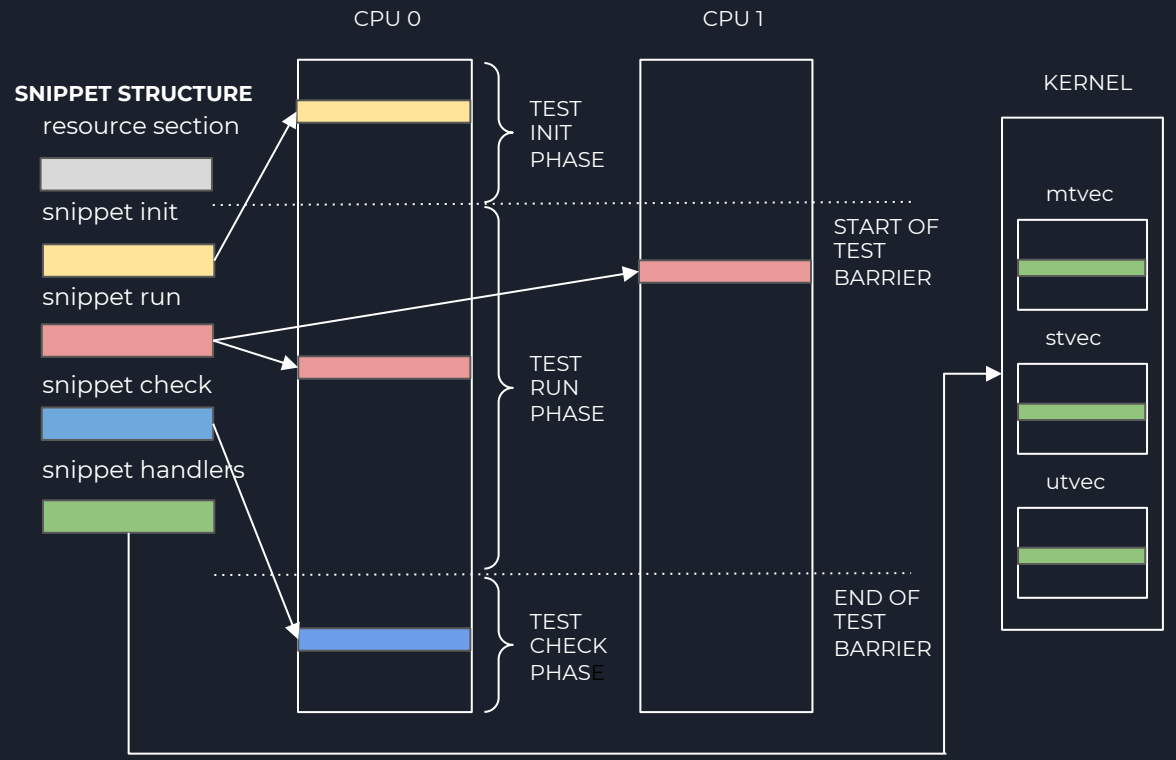


Case Study - Directed Test Development
定向测试开发

1. STING Snippets - a programming framework for development of directed stimulus

2. Language constructs to make development of complex test cases seamless and easy

3. Integrates with random instruction streams to produce interesting cross products every time its rendered into the test stream

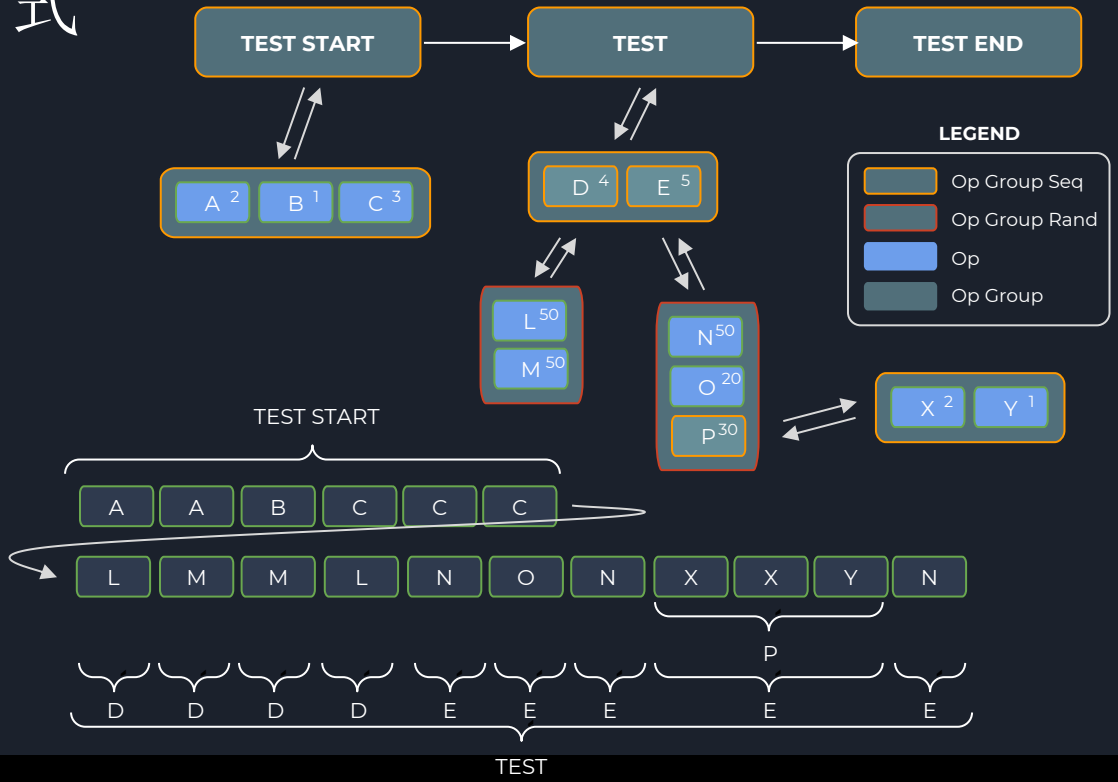


Case Study : Graph Based Test Generation
基于图形的测试生成方式

1. Graph based instruction stream generation gives extremely high control on the shape of the test

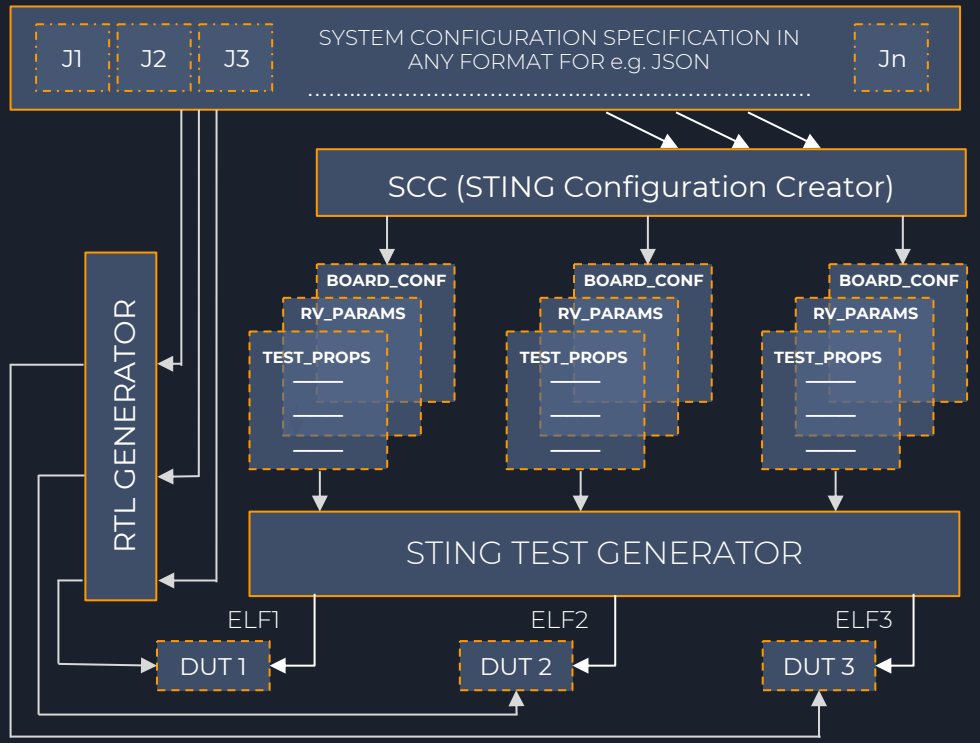
2. Scheduling of directed sequences/drivers inside the test very useful for use case verification

3. Have been used to mix elements of constrained random and directed testing methodologies in the same test resulting in extremely high coverage



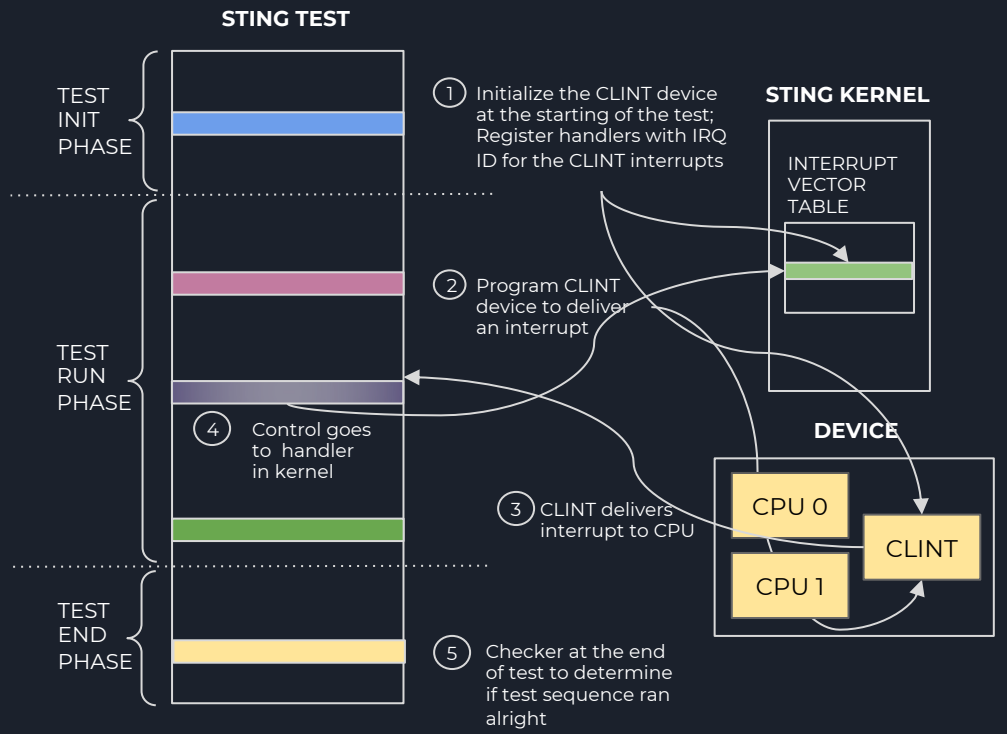
Case Study - Configurable Test Generator

1. Configurable stimulus for configurable cores; Very useful for CPU vendors
2. Reusability of valid stimulus across multiple generation of cores
3. Reduces manual effort and improves verification efficiency
4. Automatically generate STING configurations for any address map and SoC device configuration



Case Study : Enabling External Interrupts

1. STING snippet framework for development of device drivers for INTC
2. Graph based stimulus generation mechanism to render the driver instantiations in different points of interest in the test
3. Customizable kernel allows registration of interrupt handlers for each IRQ ID
4. Snippet based checkers at the end of the test determine if the interrupts were received; In case the intent was not met, a failure is flagged



Conclusion

1. RISC-V provides great flexibility to CPU designers but increases the complexity of verification

2. Design of test generators very crucial to address the unique verification needs of RISC-V

3. Combining different test generation methodologies and execution mechanisms achieves best possible verification throughput

4. Scale test stimulus and add more complex blocks as the design matures

5. Consistent execution on all DUT environments important for portability of stimulus

THANK YOU

Addressing the Complexities of RISC-V Functional Verification
解决RISC-V功能验证中的复杂性

Tony Wang
王浩为
twang@valtrix.in

For more information please visit www.valtrix.in / **write to us at** contact@valtrix.in