An Efficient Random Instruction Sequence Generation for Verification of Domain Specific Architecture Processor

Wonjae Lee, Youngsub Ko, Yelim Han, Doowon Lee, Dahyun Yoo, Heewon Ahn, Joongbaik Kim

Samsung Electronics Co., Ltd.

2026 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES
SANTA CLARA, CA, USA
MARCH 2 – 5, 2026

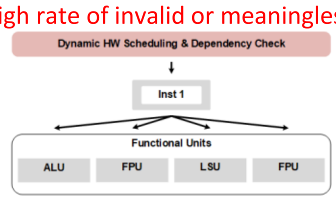
Introduction & Motivation

The Rise of DSAs & VLIW Implementation

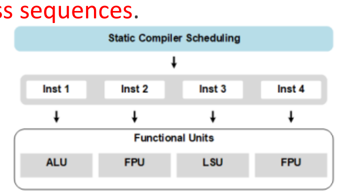
- Growing demand for parallel computation in embedded systems (e.g., AI, multimedia) drives **Domain Specific Architectures (DSAs)**.
- **VLIW (Very Long Instruction Word)** is an effective architecture for high-performance and power-efficient DSAs.

Verification Challenges in VLIW-based DSAs

- VLIW bundles multiple operations into a single long instruction, creating complex dependency and resource constraints.
- A limitation of conventional Random Instruction Sequence (RIS) verification : Pure randomization is insufficient to handle these constraints, resulting in a high rate of invalid or meaningless sequences.



Overview of general processor



Overview of VLIW processor

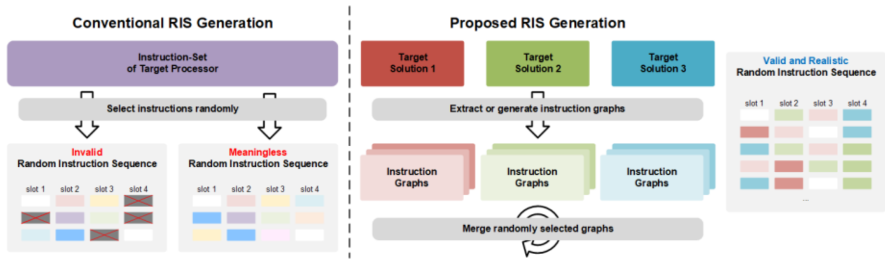
Proposed Method: Graph-Based RIS

Core Concept: Graph-Based Generation

- Instead of selecting individual instructions, we propose **selecting and merging pre-defined Instruction Graphs (sub-sequences)**.

Why Instruction Graph is Important?

- **Guaranteed Validity**: Graphs are pre-verified, ensuring connectivity and validity, increasing the probability of generating valid test sequences.
- **Domain Realism**: Graphs are extracted from target applications, reflecting realistic usage scenarios.



The concept of the proposed method

Implementation Details of Proposed Method

Overall Generation Flow

- **Selection**: Randomly select instruction graphs.
- **Allocation**: Schedule instructions into VLIW slots considering constraints.
- **Generation**: Emit final assembly and C-wrapper test code.

Input & Constraints Information

- **Instruction Graph**: Nodes (Instructions) + Connection (Dependencies)
- **HW Constraints**: Slot availability, Latency, and Register types

```
"Inst1" : {
   "instType"  : "vmv.s.x.8",
   "numInput"  : 1,
   "numOutput" : 1
},
"Inst2" : {
   "instType"  : "vnclip.wv.8",
   "numInput"  : 2,
   "numOutput" : 1
},
"Inst3" : {
   "instType"  : "vmacc.vv.8",
   "numInput"  : 3,
   "numOutput" : 1
}
```

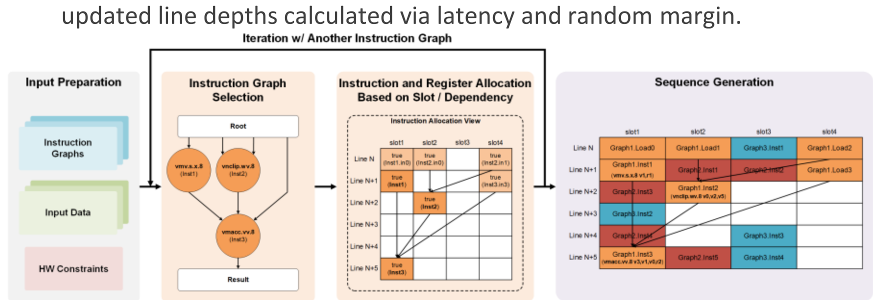
```
"Inst1" : [
   ["Root", 0]
],
"Inst2" : [
   ["Root", 1], ["Root", 2]
],
"Inst3" : [
   ["Inst1", 0], ["Inst2", 0], ["Root", 3]
],
"Result" : [
   ["Inst3", 0]
]
```

Instruction	Avail. Slot	Latency	...	Input Reg.	Output Reg.
vmv.s.x.8	1110	1	...	R	V
vnclip.wv.8	1100	1	...	V:V	V
vmacc.vv.8	1010	4	...	V:V:R	V
vwadd.vv.8	1010	2	...	V:V	V:V
...					

Example of instruction graph and HW constraint information

Instruction & Register Allocation Algorithm

- **InstReadyQueue**: A queue of instructions whose predecessors are allocated, sorted by line depth (earliest possible execution cycle).
- **Conflict Handling**: If resources (slot/reg) are unavailable, the instruction is deferred to the next cycle (line depth + 1).
- **Randomness**: Available resources are randomized for test diversity.
- **Successor Update**: Successor instructions are added to the queue with updated line depths calculated via latency and random margin.



Steps of the proposed RIS generation

Results

Verification Flow

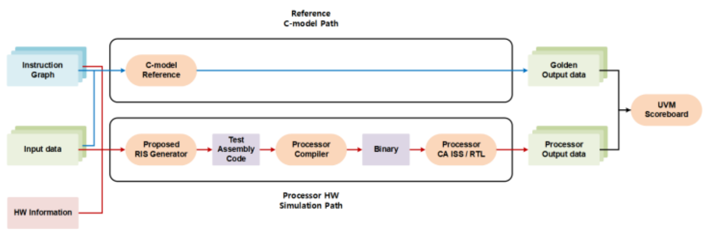
- Run a reference C-Model and ISS/RTL simulations to compare outputs.
- Environment: UVM testbench + Xcelium simulation + IMC analysis

Robust Validity

- Simple RIS validity drops to 7.4% at 40 bundle lines.
- Proposed RIS maintains close to 100% validity regardless of bundle lines.

High Efficiency & Coverage

- Achieved comparable or higher coverage (Toggle: 86.21% → 91.49%).
- Used ~50% fewer test vectors.



Verification flow with the proposed RIS generation

Conclusion

Overcoming Verification Complexity

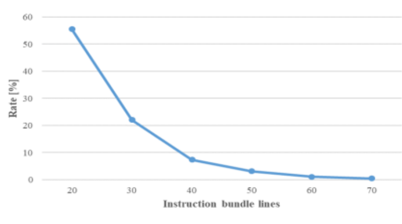
- Solves the VLIW validity challenge by integrating hardware constraints into a graph-based generation flow.

Scenario-Based Realism

- Enhances verification quality by generating complex test cases that reflect real-world workloads.

Optimal Solution for DSAs

- Balances the precision of directed testing with the diversity of random generation, offering a controllable and efficient verification solution.



Valid test rate of the simple RIS method

			Simple RIS	Proposed RIS
Number of test vectors			2,334	1,177
Coverage	Block	Total bins	361,384	
		Covered bins	334,421	336,421
		Score	92.54%	93.09%
	Expression	Total bins	2,105,767	
		Covered bins	1,880,120	1,869,113
		Score	89.28%	88.76%
	Toggle	Total bins	2,542,774	
		Covered bins	2,192,105	2,326,438
		Score	86.21%	91.49%

Comparison of code coverage

Contact Information

Email: Wonjae Lee (wonja2.lee@samsung.com)

Samsung Electronics Co., Ltd., Hwaseong-si, Korea