2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION TAIWAN

Verifying Configurable AndesCore™ Processors by Using PSS

Luther Lee (Speaker), Andy Lo, Brett Yeh

RD-VLSI of Andes Technology

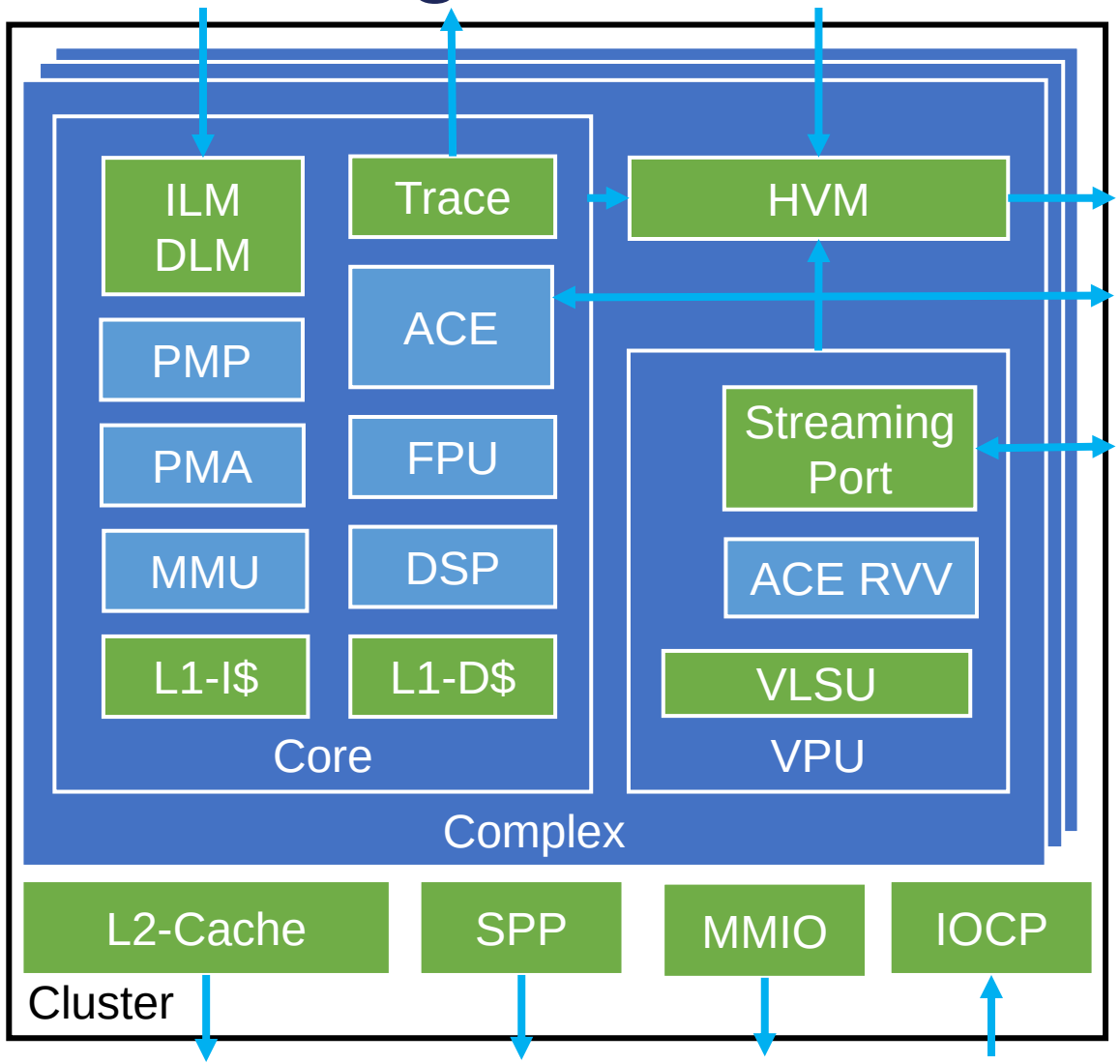
accellera SYSTEMS INITIATIVE

Agenda

- Verification Challenge
  - More Configurations
  - Non-standarized Language
- Andes Technology PSSGen
- Case Study
  - RAW Data Hazard
  - Basic Block (Vector)
- Conclusion and Future Work

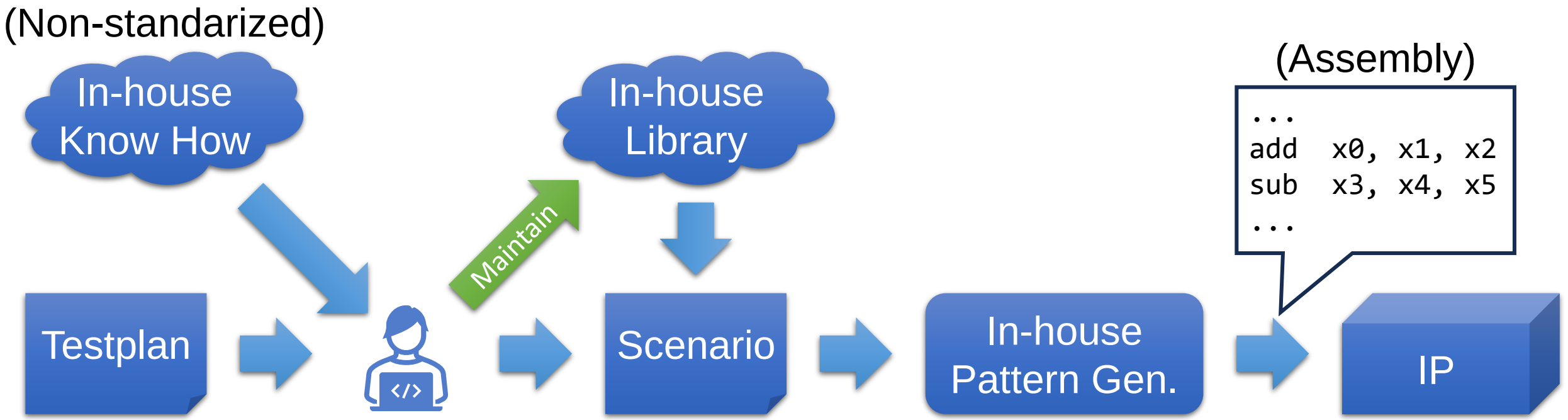
Verification Challenge: More Configurations

- AndesCore™ AX45MPV
  - Up to 8 cores per cluster
  - L1 I/D Cache: 8KB~64KB
  - L2 I/D Cache: 128KB~8MB
  - RVV
    - VLEN: 128~1024 bits
    - DLEN: 128~1024 bits
    - FLEN: 32/64 bits
    - ELEN: 32/64 bits
  - etc.
- Maintain combination by using PSS-type library



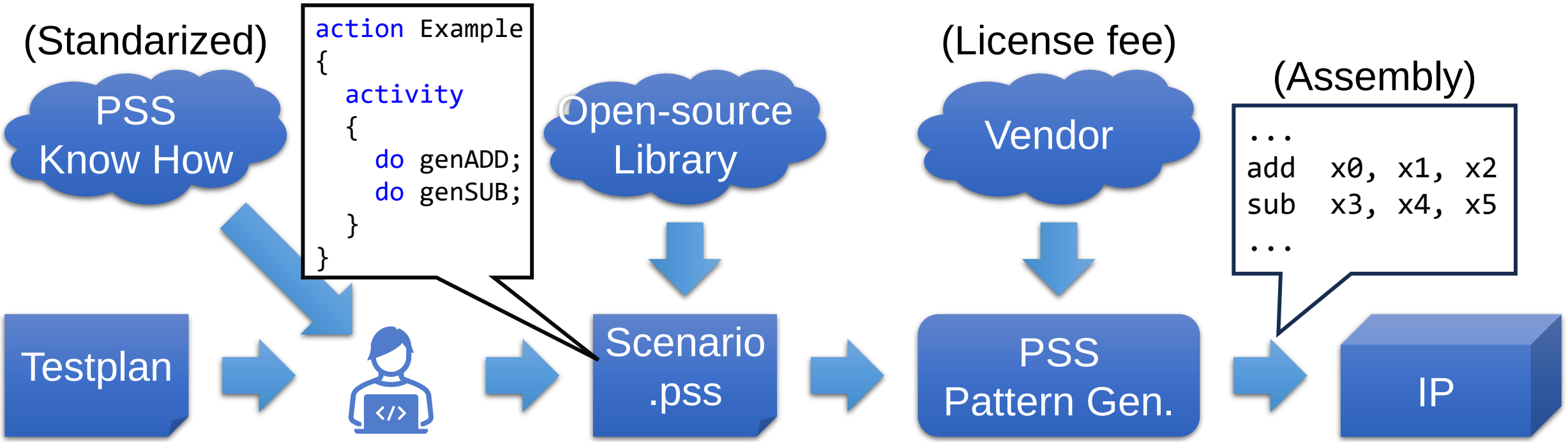
Verification Challenge: Non-standarized Language

- Verification using in-house environment:
  - Scenario written by non-standarized in-house language
  - Need to maintain in-house library



Verification Challenge: Non-standarized Language

- Verification using standarized PSS language:
  - Scenario written by PSS and can be shareable
  - Pattern generator may be provided by vendor

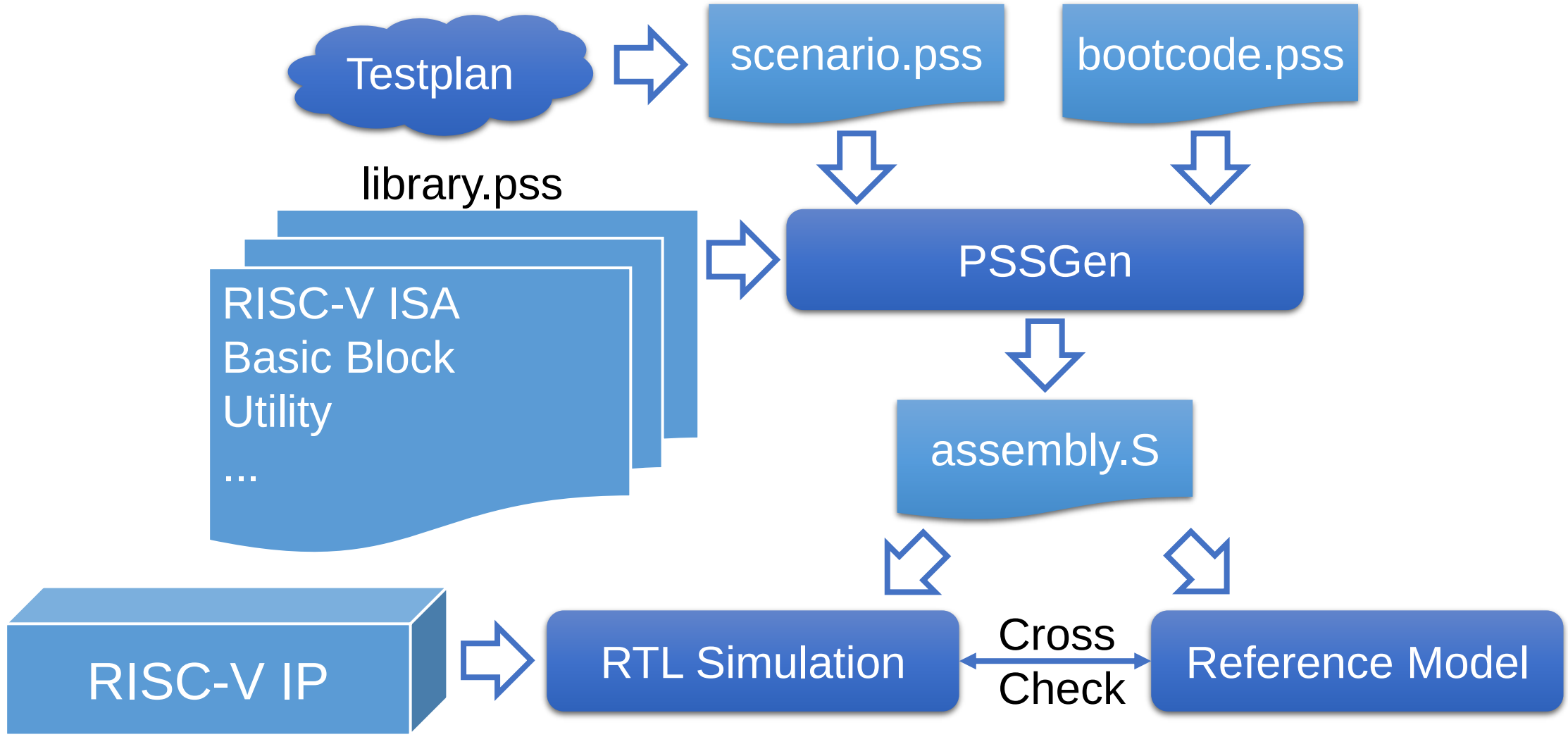


Andes Technology PSSGen

- Target on generate pattern for verify RISC-V CPU
- ✅ Open-source on GitHub/andestech/pss-gen (MIT license)
- ✅ Implemented based on PSS LRM v2.0
- ✅ Our PSSGen is FREE !
- ✅ Have a tutorial website called PSS Cookbook
  - Summarise syntax from LRM
  - Friendly to new learner

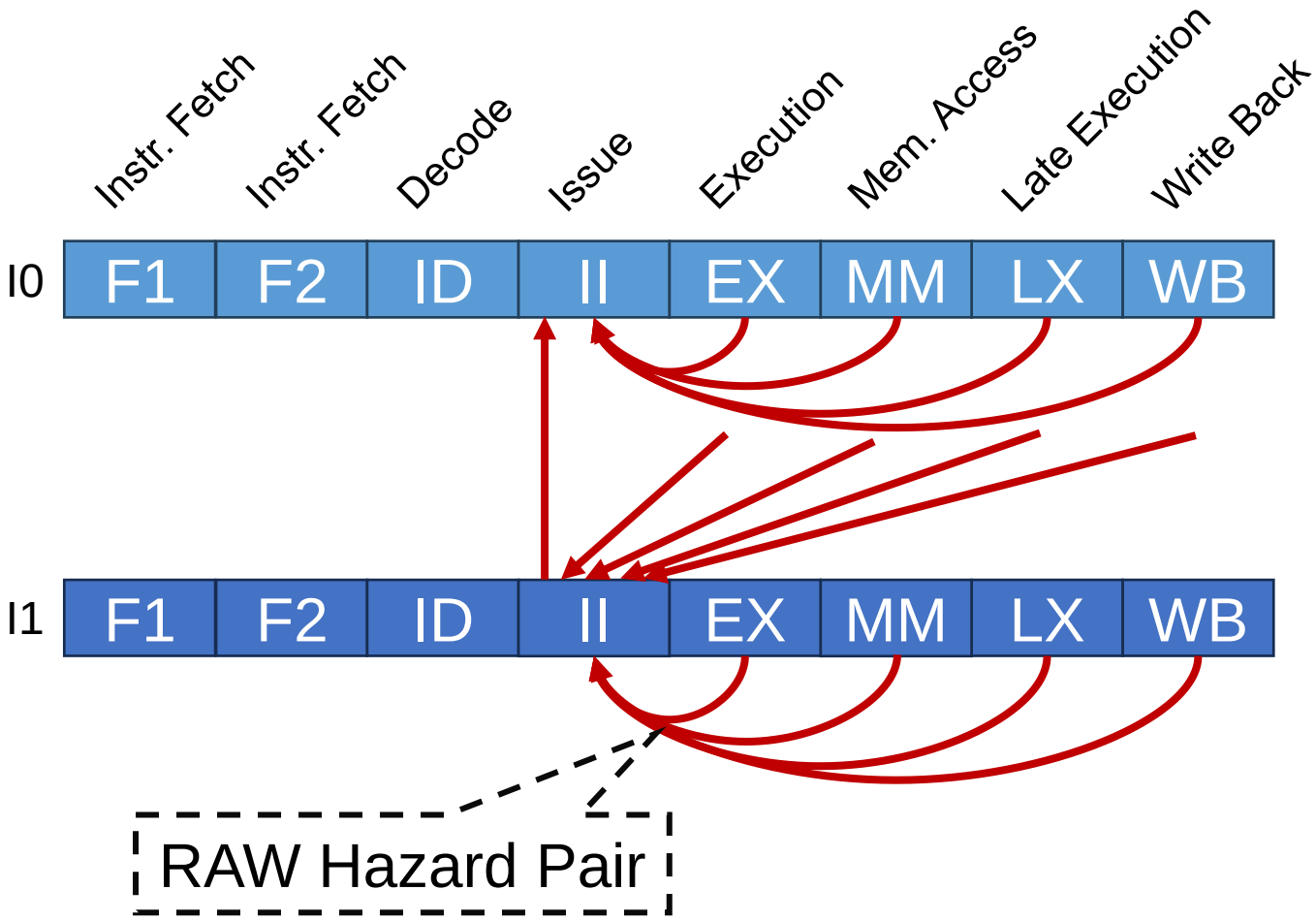
- 🎉 Please give it a try !

How we use Andes Technology PSSGen



Case Study: RAW Data Hazard

- Data Hazard
  - 8-Stage
  - Dual-Issue
  - 20+ Instruction Types
- Total Testcases:
  - stage X issue X types
  - more than 200+ pairs



Case Study: RAW Data Hazard

- Testplan: Verify Read-After-Write hazard between pipeline
  - Test all ALU instructions
  - Test predecessor.RD == successor.RS
  - Insert independent instructions between them
- Scenario (PSS):


```
action RAWHazard_Wrapper {
    activity {
        foreach (x : BasicBlock::getAllALUInstr()) {
            foreach (y : BasicBlock::getAllALUInstr()) {
                do genRAWHazardPair with { first_instr == x; second_instr == y; };
            }
        }
    }
}
```

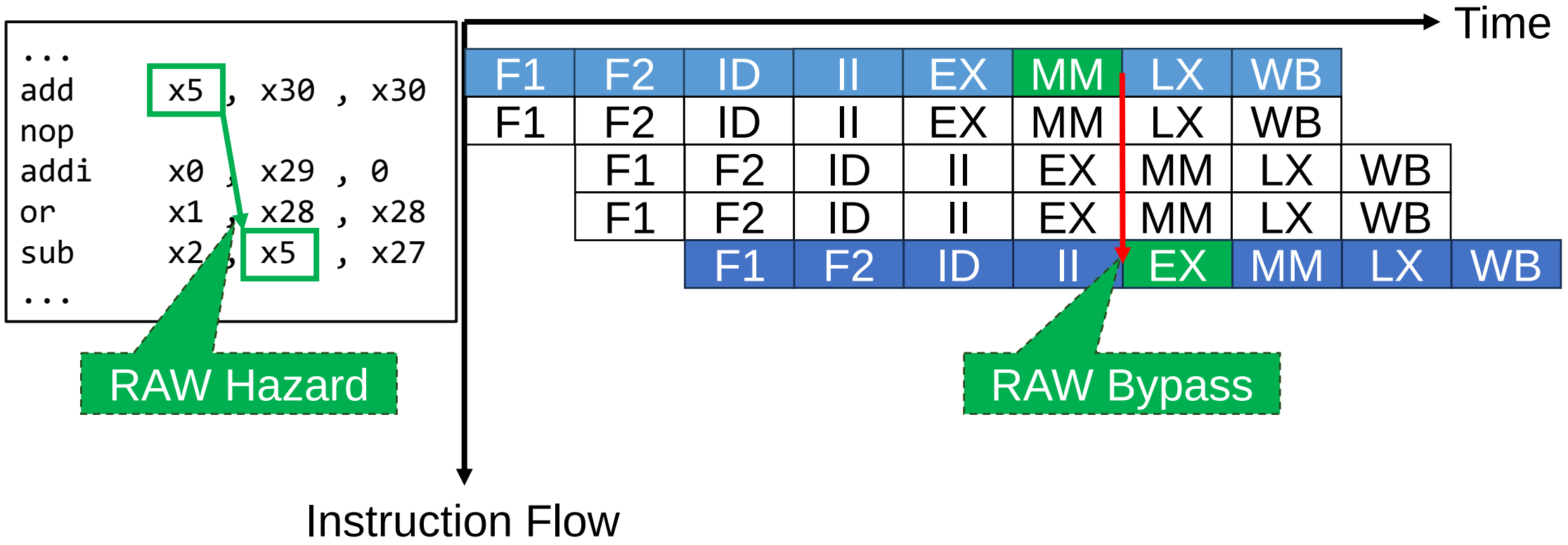
Case Study: RAW Data Hazard

- Implement **genRAWHazardPair** action:

```
action genRAWHazardPair {
    rand string first_instr, second_instr;
    rand int dummy_instr in [0..10];
    rand bit [5] r_hazard in [0..31];
    activity {
        do ISA::genInstr with { instr == first_instr ; rd == r_hazard; };
        repeat (dummy_instr) {
            do ISA::genInstr with { rd != r_hazard; rs != r_hazard; };
        }
        do ISA::genInstr with { instr == second_instr; rs == r_hazard; };
    }
}
```

Case Study: RAW Data Hazard

- PSSGen generates assembly code:



Case Study: Basic Block (Vector)

- Vector Instructions
  - LMUL: MF8~M8
  - SEW: 8~64
  - 600+ Vector Instructions
- Total Combinations
  - LMUL X SEW X Instructions
  - 16K+ combinations
- Library inferences legal cases

```
action VectorLoadStore_Wrapper {
    list<lmul_t> legal_lmul = Util::getLegalAllLmul();
    list<sew_t>  legal_sew  = Util::getLegalAllSew() ;

    exec pre_solve {
        BasicBlock::addInstr(RVV::getInstr("all"));
    }

    activity {
        foreach (i: legal_lmul) {
            foreach (j: legal_sew) {
                do BasicBlock::genAllLegalInstr with {
                    lmul == i; sew == j;
                };
            }
        }
    }
}
```

Case Study: Basic Block (Vector)

- PSSGen generates assembly code:

```
...
vsetvli     x5 , x0 , e8 , mf8, ta, ma
vadd.vv     v8 , v16, v24
vsetvli     x6 , x0 , e8 , mf8, ta, ma
vsub.vv     v9 , v17, v25
vsetvli     x7 , x0 , e8 , mf8, ta, ma
vand.vv     v10, v18, v26
vsetvli     x8 , x0 , e8 , mf8, ta, ma
vxor.vv     v11, v19, v27
...
```

LMUL=mf8

SEW=e8

```
action VectorLoadStore_Wrapper {
    list<lmul_t> legal_lmul = Util::get
    list<sew_t>  legal_sew  = Util::get

    exec pre_solve {
        BasicBlock::addInstr(RVV::getIn
    }

    activity {
        foreach (i: legal_lmul) {
            foreach (j: legal_sew) {
                do BasicBlock::genAllLe
                    lmul == i; sew == j
            }
        }
    }
}
```

What's PSS better than In-house Language?

- Reduce learning curve for test pattern creation & maintenance
- 3x number of test patterns produced
- Easier to read

In-house

```
sub basic_call {
...
    for (my $i = 0; $i < $func_no; $i++) {
        my $func_ref = function_begin();
        $block_size = (int rand(($average_block_size  << 1) + 1)) & 0xfffffffc;
        basic_block_selection($block_size, $sel, $mode);
        $total_block_size -= $block_size;
        if ($i > 0) {function_call($func_ref_list[$i - 1]);}
        $block_size = (int rand(($average_block_size << 1) + 1)) & 0xfffffffc;
        basic_block_selection($block_size, $sel, $mode);
        $total_block_size -= $block_size;
        function_end();
        push(@func_ref_list, $func_ref);
    }
    for (my $i = 0; $i < $func_call_no; $i++) {
        $block_size = (int rand(($average_block_size << 1) + 1)) & 0xfffffffc;
        basic_block_selection($block_size, $sel, $mode);
        $total_block_size -= $block_size;
        function_call($func_ref_list[rand($func_no)]);
...
```

PSS

```
action run_rvv_instruction_only {
    set<string> inst_set;
    exec pre_solve {
        inst_set = get_inst_by_set( {"all_inst"} );
        disable_inst_by_set( inst_set );
        inst_set = get_inst_by_set( {"rvv_inst"} );
        add_inst_by_set( inst_set, 1 );
    }
    activity {
        do inst_util::basic_block with {instr_cnt==128;};
    }
}
```

Future Work

- Share PSS instructions library
- Enhance PSSGen
  - Support more features
  - Support new LRM
- Enhance PSS Cookbook
  - Add more tutorials
  - Add more tips & examples

- Welcome again, everyone! Try and contribute to PSSGen!

Q & A

GitHub: Andes Technology PSSGen

Email (Luther Lee): luthe590@andestech.com