Taming formal with intelligent automation?

Tobias Ludwig, CEO
LUBIS EDA

LUBIS EDA

HTML und JavaScript

zum Nachschlagen

Das 2-in-1 Kombi-Buch

1. Ausführliche Schritt-für-Schritt-Anleitungen
2. Referenz: Wichtiges von A-Z

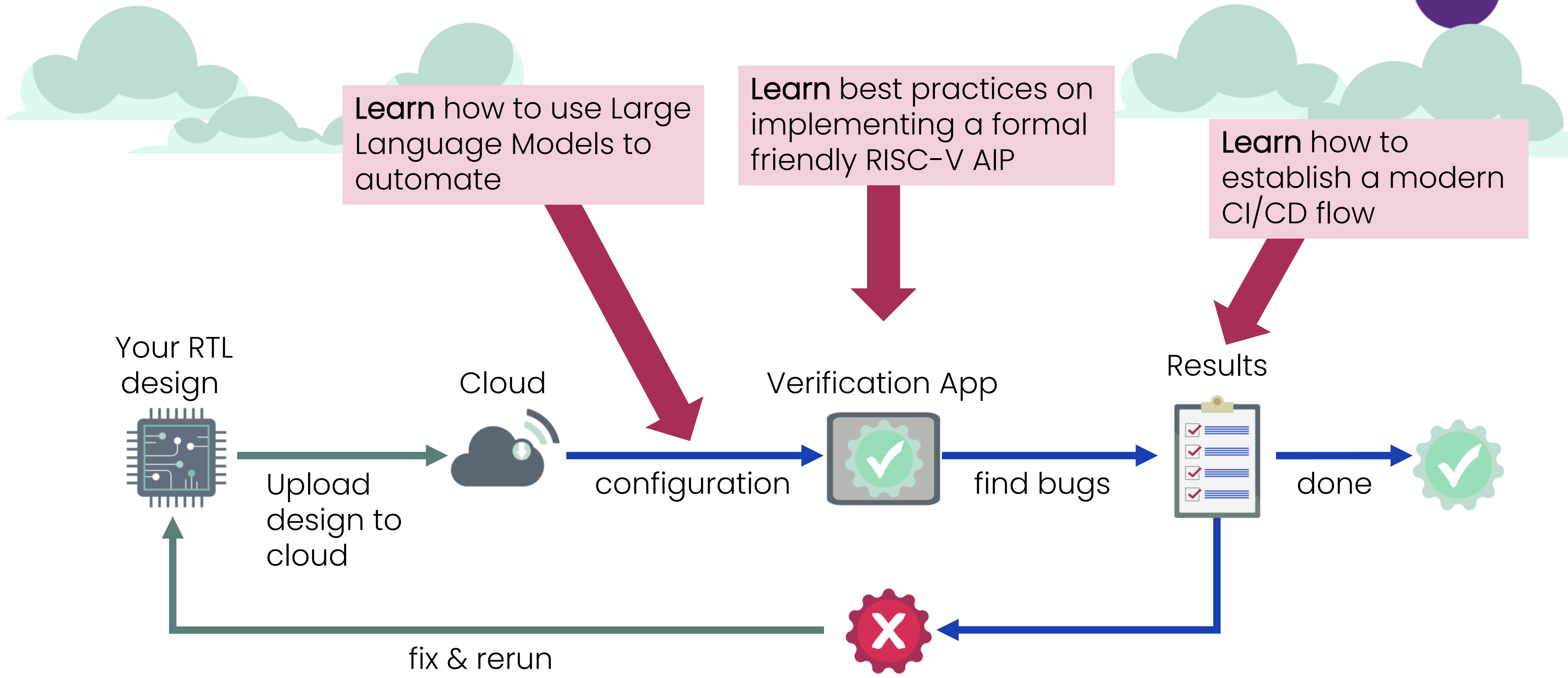
<html>

SCRIPTLANGUAGE

JavaScript

SYBEX

Bug detection machine

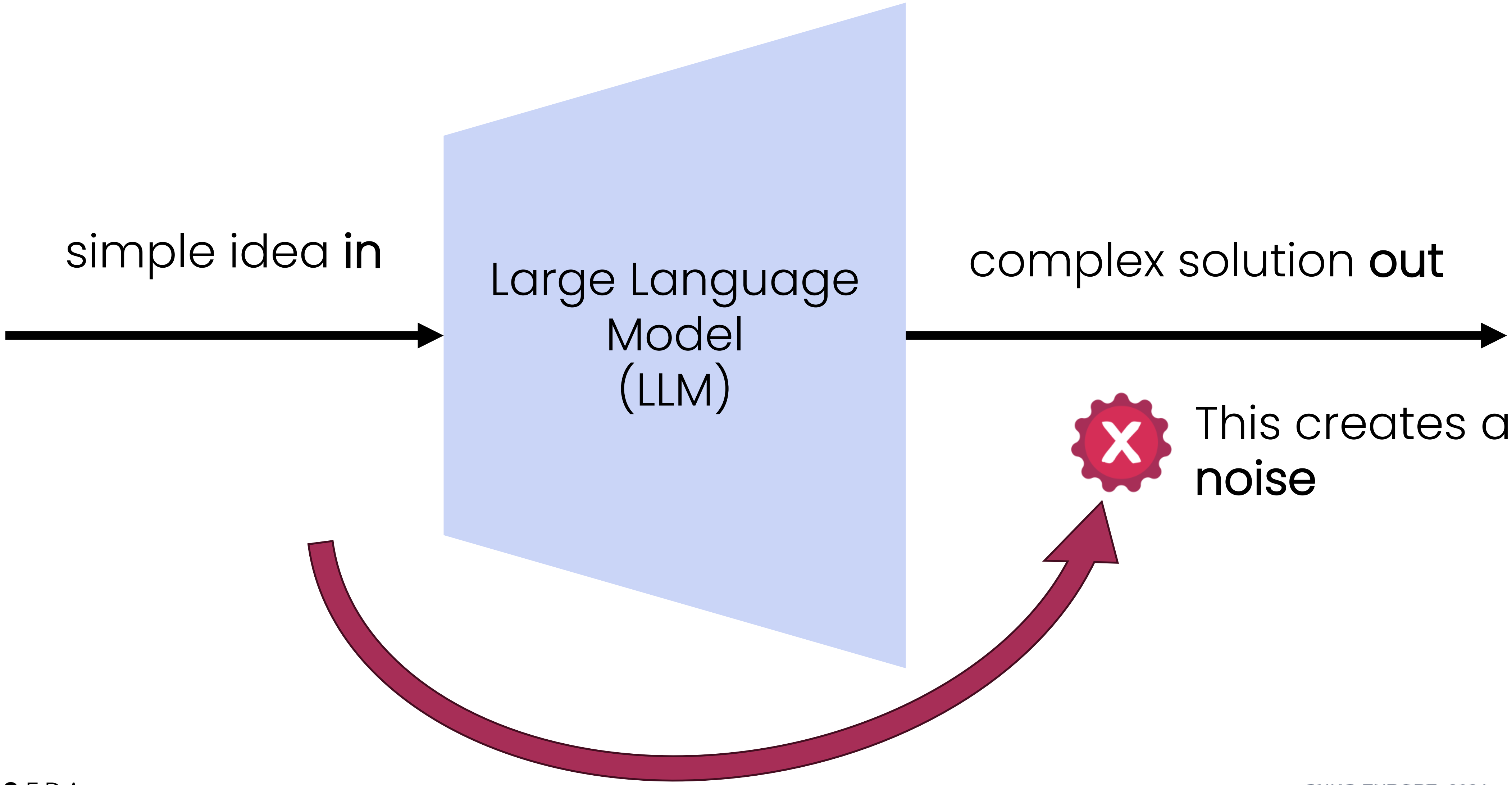

Learn how to use Large Language Models to automate
Learn best practices on implementing a formal friendly RISC-V AIP
Learn how to establish a modern CI/CD flow
Your RTL design
Cloud
Verification App
Results
Upload design to cloud
configuration
find bugs
done
fix & rerun

🟩 You 🟦 Automatic

Bug detection machine

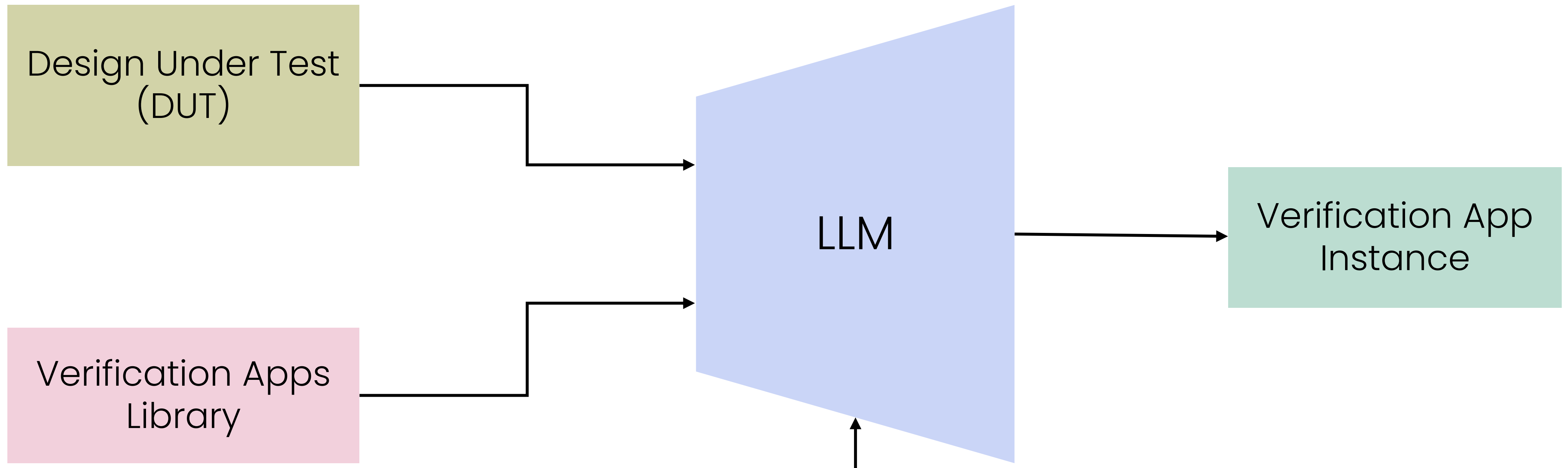
1) How to deal with common blocks
2) Free your schedule if you’re an expert
3) Why the cloud makes formal feasible

The jack of all trades!



This creates a lot of **noise**

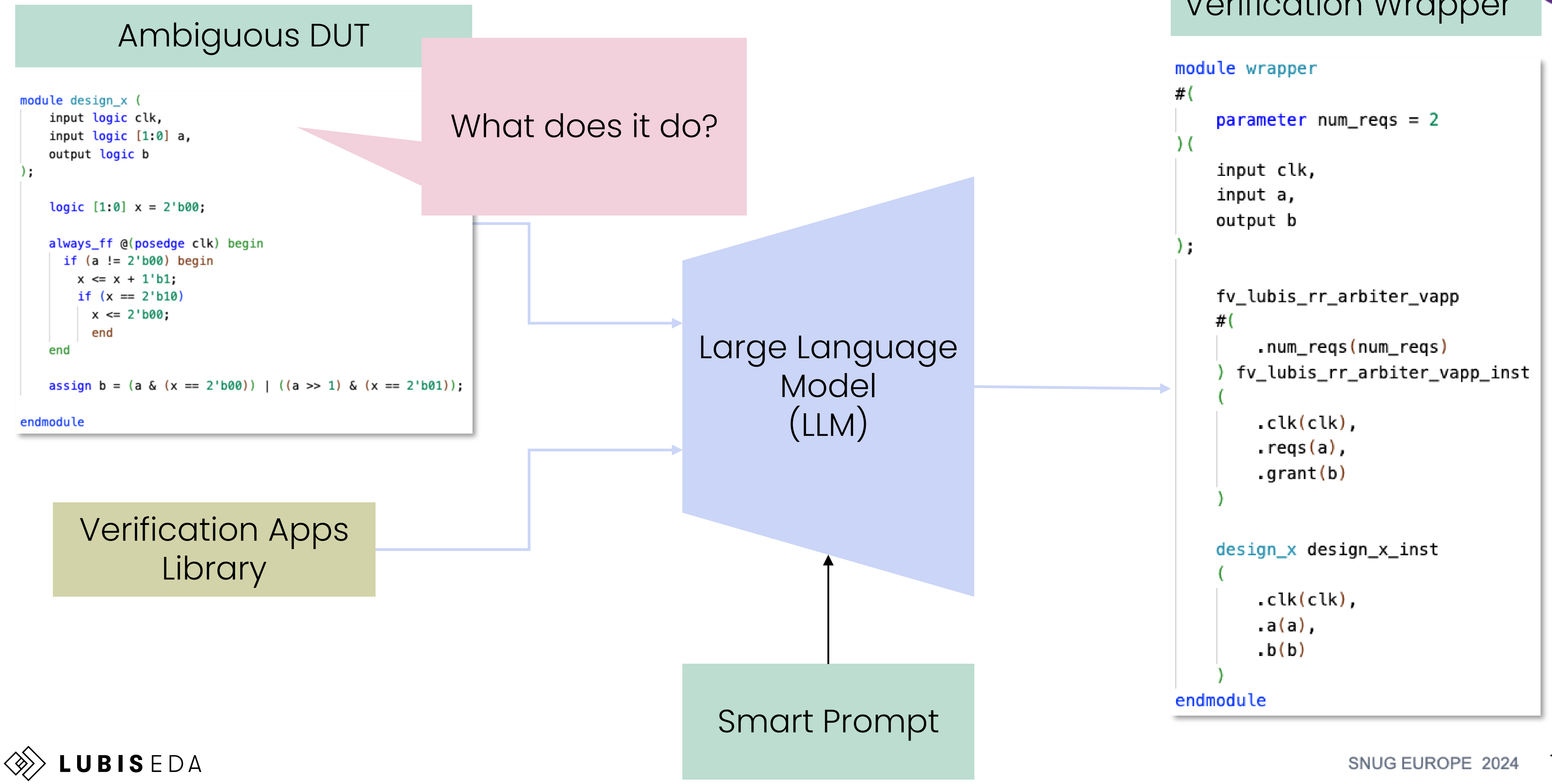
Smart prompting



Dear LLVM:

- Figure out what the design does
- Select the verification apps from the library
- Connect everything together
- Create a loadscript

Example



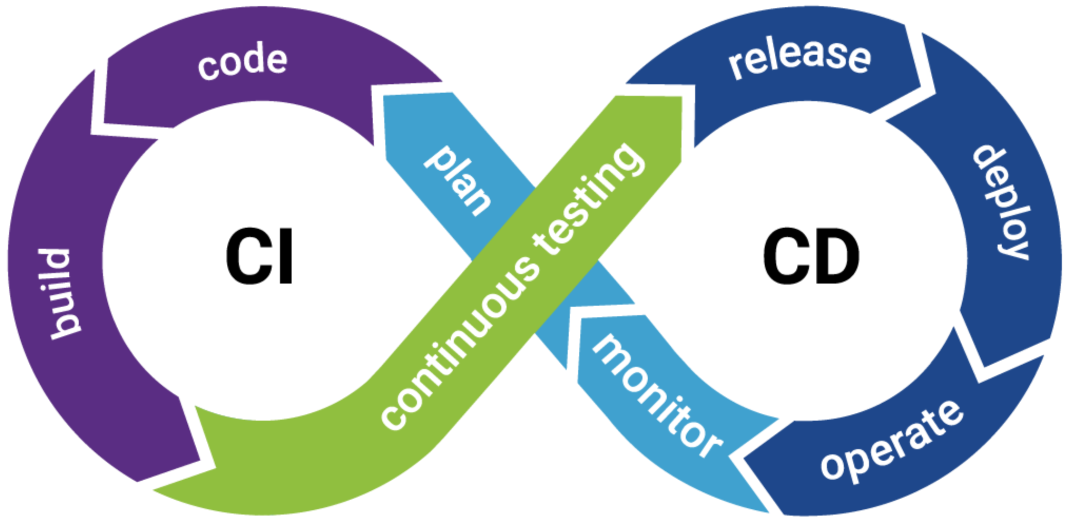
```
module design_x (
    input logic clk,
    input logic [1:0] a,
    output logic b
);

    logic [1:0] x = 2'b00;

    always_ff @(posedge clk) begin
      if (a != 2'b00) begin
        x <= x + 1'b1;
        if (x == 2'b10)
          x <= 2'b00;
        end
    end

    assign b = (a & (x == 2'b00)) | ((a >> 1) & (x == 2'b01));

endmodule
```

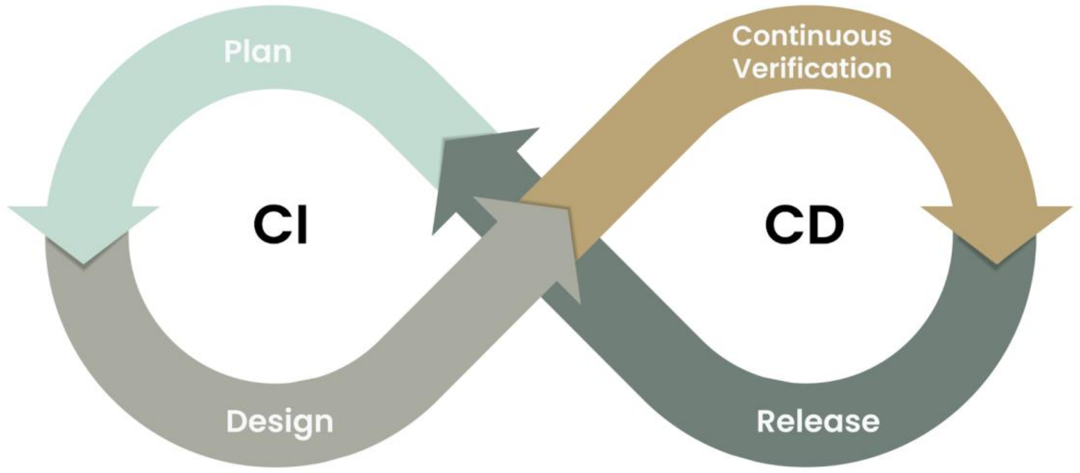
```
module wrapper
#(
    parameter num_reqs = 2
)(
    input clk,
    input a,
    output b
);

    fv_lubis_rr_arbiter_vapp
    #(
        .num_reqs(num_reqs)
    ) fv_lubis_rr_arbiter_vapp_inst
    (
        .clk(clk),
        .reqs(a),
        .grant(b)
    )

    design_x design_x_inst
    (
        .clk(clk),
        .a(a),
        .b(b)
    )
endmodule
```

What's the buzz about CI/CD

And how does that fit into hardware world?

Software CI/CD	Hardware CI/CD





We need two items:

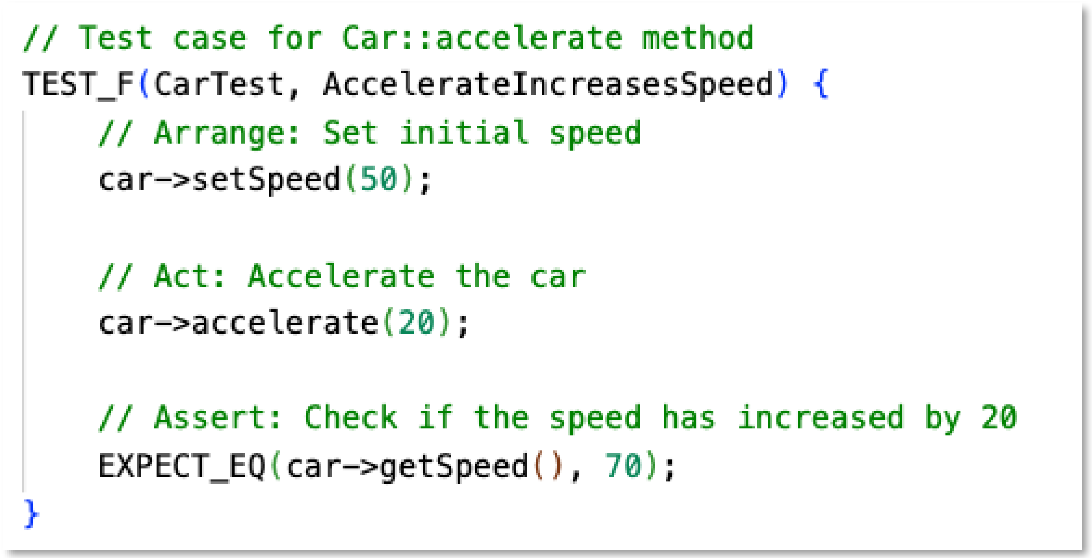
- Unit tests
- Pipelines

We need two items:

- Formal Properties
- Pipelines

Unit Tests vs Formal Properties

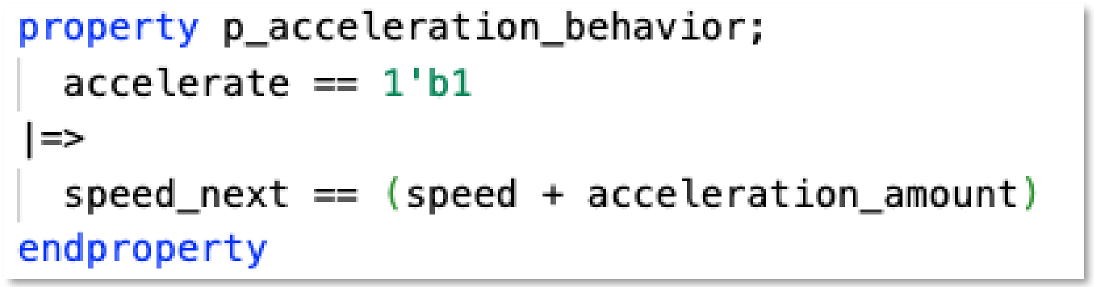
Software Unit Test



```
// Test case for Car::accelerate method
TEST_F(CarTest, AccelerateIncreasesSpeed) {
    // Arrange: Set initial speed
    car->setSpeed(50);

    // Act: Accelerate the car
    car->accelerate(20);

    // Assert: Check if the speed has increased by 20
    EXPECT_EQ(car->getSpeed(), 70);
}
```

- Checks specific situation
- A bug in accelerating by 40 cannot be caught

Hardware Formal Property



```
property p_acceleration_behavior;
  accelerate == 1'b1
|=>
  speed_next == (speed + acceleration_amount)
endproperty
```

- Checks a behavior
- Any bug in acceleration can be caught

Formal is different than Unit Testing!

How does a good and a bad formal property look like?

JAL: Jump and Link instruction

```
property p_jal_instruction;
    jal_instruction_executed
|->
    dut_register_file == expected_register_file
    dut_pc == expected_pc
    dut_csr == expected_csr
endproperty
```

✗ Very high complexity

Formal is different than Unit Testing!

How does a good and a bad formal property look like?

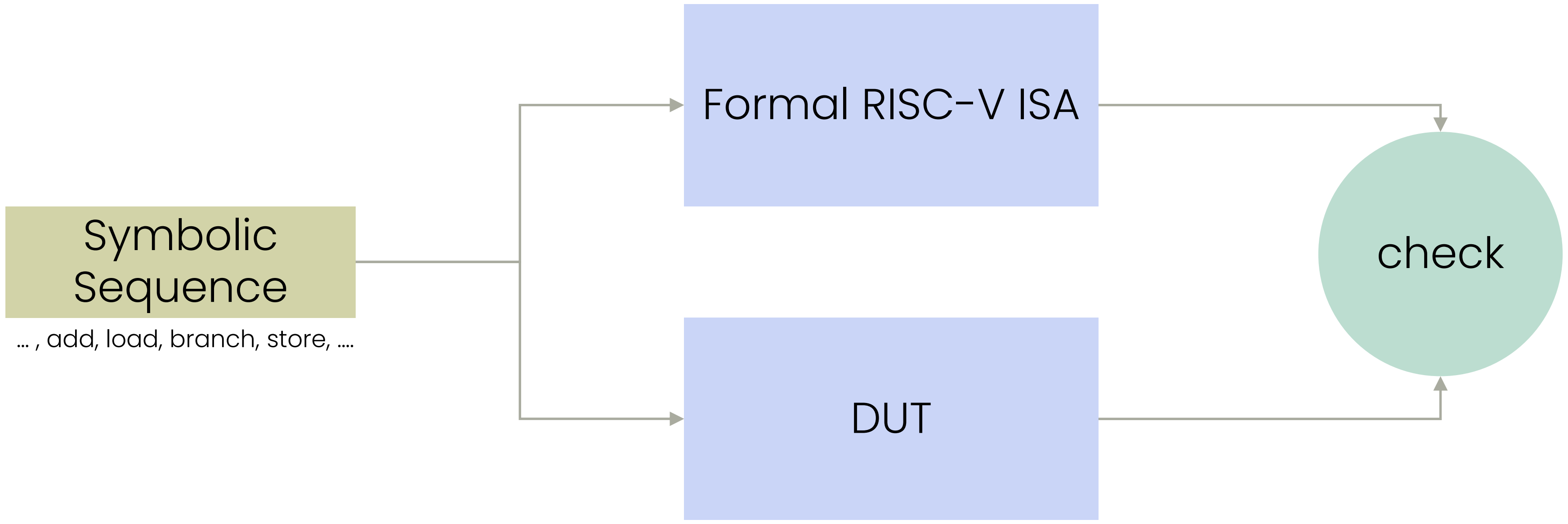
JAL: Jump and Link instruction

```
property p_jal_instruction_regfile;
    jal_instruction_executed
|->
    dut_register_file == expected_register_file
endproperty
```

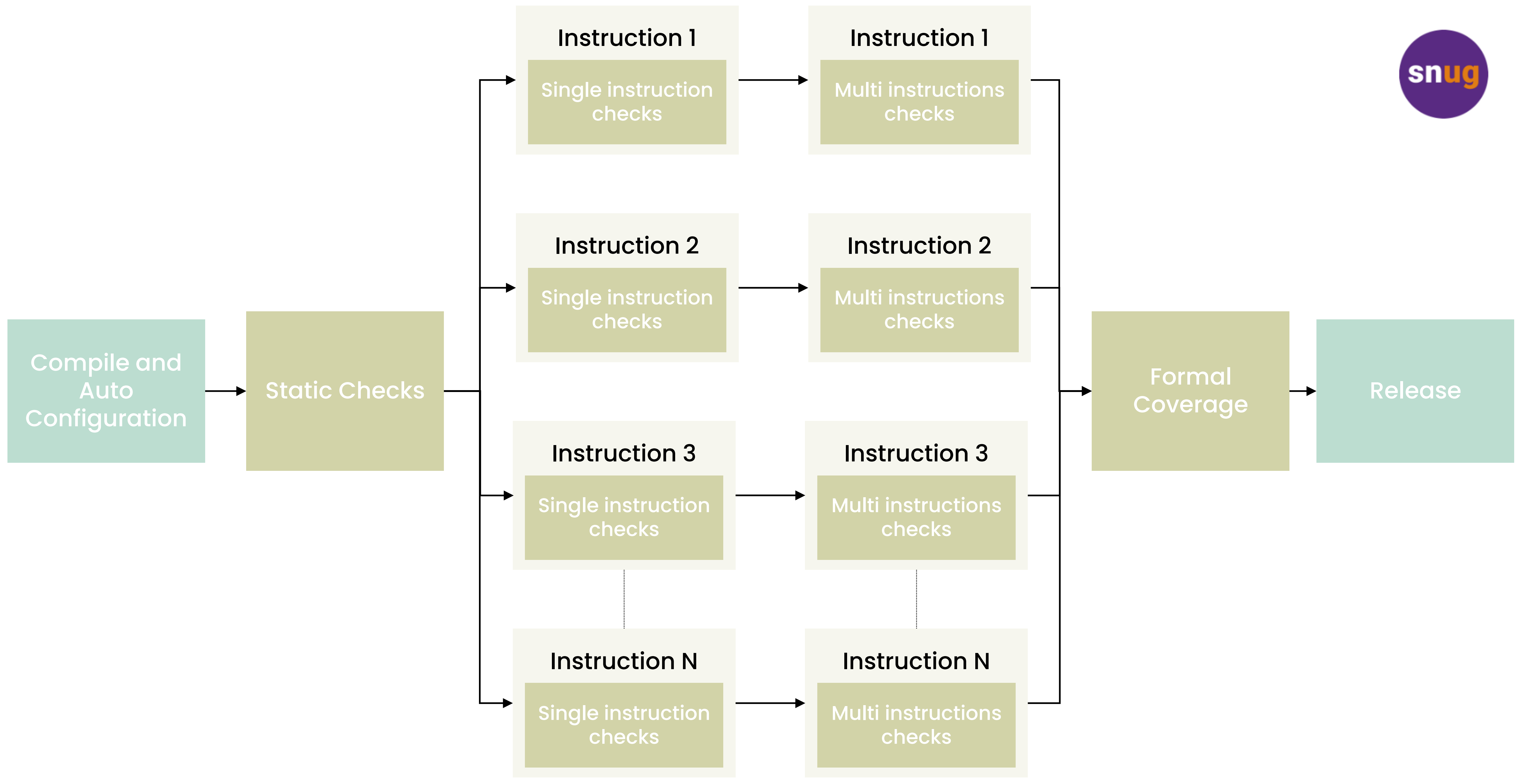
```
property p_jal_instruction_csr;
    jal_instruction_executed
|->
    dut_csr == expected_csr
endproperty
```

```
property p_jal_instruction_pc;
    jal_instruction_executed
|->
    dut_pc == expected_pc
endproperty
```

Piecing it all together



checkout ARMs paper on *formal ISA* for more info


Compile and Auto Configuration
Static Checks
Instruction 1
Single instruction checks
Instruction 1
Multi instructions checks
Instruction 2
Single instruction checks
Instruction 2
Multi instructions checks
Instruction 3
Single instruction checks
Instruction 3
Multi instructions checks
Instruction N
Single instruction checks
Instruction N
Multi instructions checks
Formal Coverage
Release

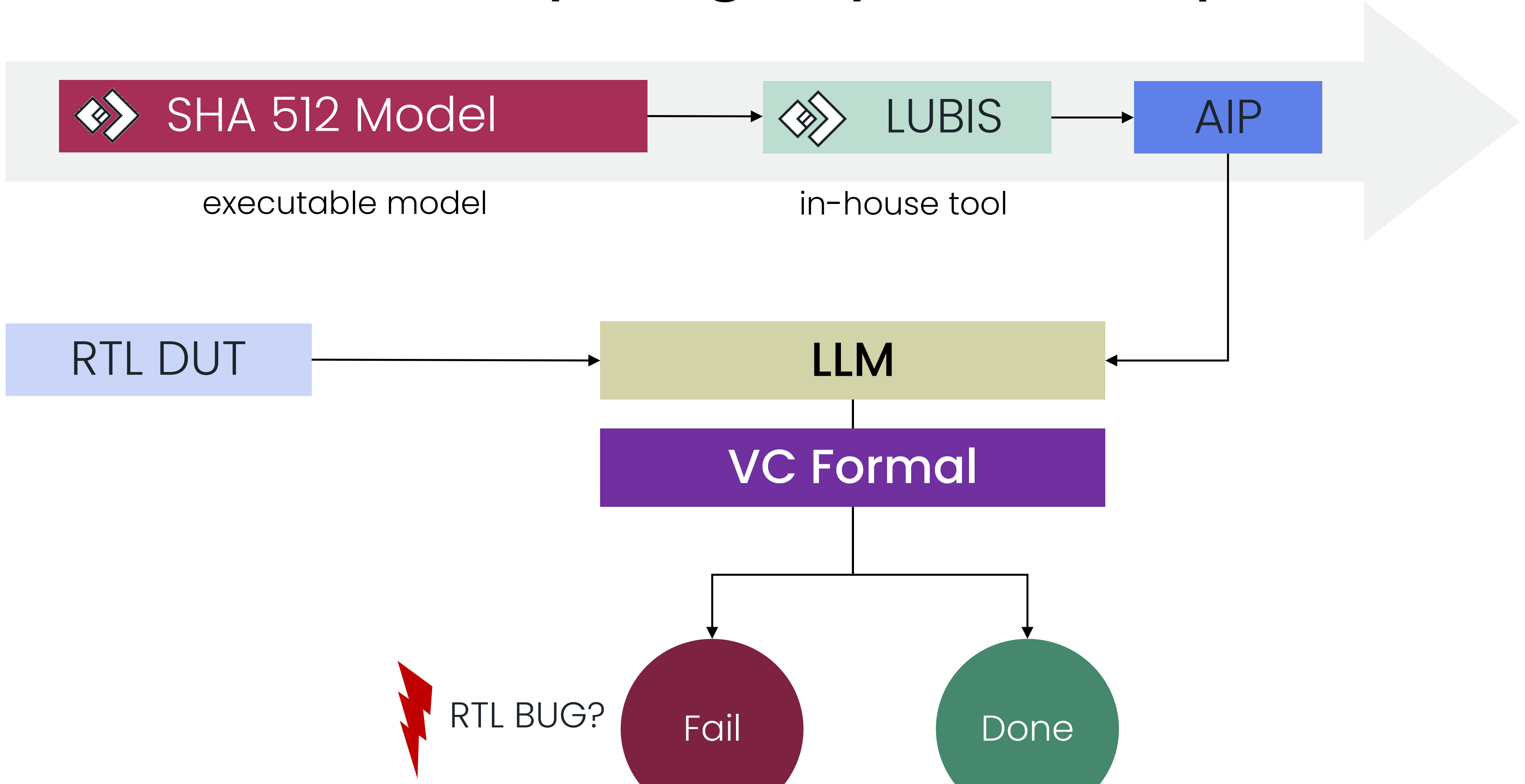
Results/Benchmarks

How much time does it take to setup and find the first bug?

Setup Time	< 1 hour on most cores
Time to first bug	< 5 min after setup
Average time per instruction	Any 3 instructions 1hr to 12hr

LUBIS EDA

Make sure you get your AIP quick!



The generated assertion

```
//SHA_ROUNDS: Compute a new digest
for (i=0; i<NUM_ROUNDS; ++i) {
    insert_state("SHA_Rounds");

    k = K[i];
    if (i < 16){
        w = W[i];
    } else {
        tmp_w = delta1(W[14]) + W[9] + delta0(W[1]) + W[0];
        for (j=0; j<15; ++j) {
            W[j] = W[(j+1)];
        };
        W[15] = tmp_w;
        w = tmp_w;
    };

    t1 = T1(e, f, g, h, k, w);
    t2 = T2(a, b, c);
    h = g;
    g = f;
    f = e;
    e = (d + t1);
    d = c;
    c = b;
    b = a;
    a = (t1 + t2);
};
//Done: Provide the new digest to the output interfae
insert_state("DONE");
```

```
SHA_Rounds_to_SHA_Rounds_1_a: assert property (disable iff(!rst) SHA_Rounds_to_SHA_Rounds_1_p);
property SHA_Rounds_to_SHA_Rounds_1_p;
  SHA_Rounds &&
  (i >= 'sd16) &&
  (('sd1 + i) < 'sd80)
|->
  ##1 (SHA_Input_notify == 0) and
  ##1 (out_notify == 0) and
  ##1
  SHA_Rounds &&
  H_0 == $past(H_0, 1) &&
  H_1 == $past(H_1, 1) &&
  [...]
  H_7 == $past(H_7, 1) &&
  W_0 == $past(W_1, 1) &&
  W_10 == $past(W_11, 1) &&
  W_11 == $past(W_12, 1) &&
  W_12 == $past(W_13, 1) &&
  W_13 == $past(W_14, 1) &&
  W_14 == $past(W_15, 1) &&
  W_15 == 64'((((delta1($past(W_14, 1)) + $past(W_9, 1)) + delta0($past(W_1, 1))) + $past(W_0, 1))) &&
  W_1 == $past(W_2, 1) &&
  W_2 == $past(W_3, 1) &&
  [...]
  W_8 == $past(W_9, 1) &&
  W_9 == $past(W_10, 1) &&
  a == 64'((T1($past(e, 1), $past(f, 1), $past(g, 1), $past(h, 1), (($past(i, 1) == 'sd16) ? 64'd1647287
  b == $past(a, 1) &&
  c == $past(b, 1) &&
  d == $past(c, 1) &&
  e == 64'(($past(d, 1) + T1($past(e, 1), $past(f, 1), $past(g, 1), $past(h, 1), (($past(i, 1) == 'sd16)
  f == $past(e, 1) &&
  g == $past(f, 1) &&
  h == $past(g, 1) &&
  i == ('sd1 + $past(i, 1));
endproperty
```

ABOUT US

We are helping our customers find **simulation-resistant** and **corner-case bugs** in high-risk silicon design or IP blocks

LUBIS on cloud enables you to:

1	Reach your silicon design verification **goals faster**
2	Uncover hard to **find functional bugs** in your design
3	Stay **within your budget** and tape-out schedule

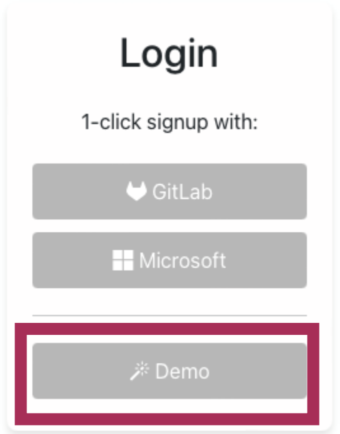
Team 20+

Kaiserslautern, Germany

We love formal

Demo

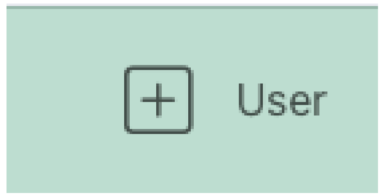
riscifier.lubis-eda.com


Login

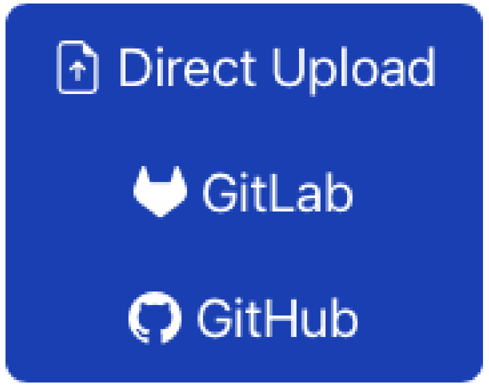
1-click signup with:

Setup

< 5 min



Creates a new project

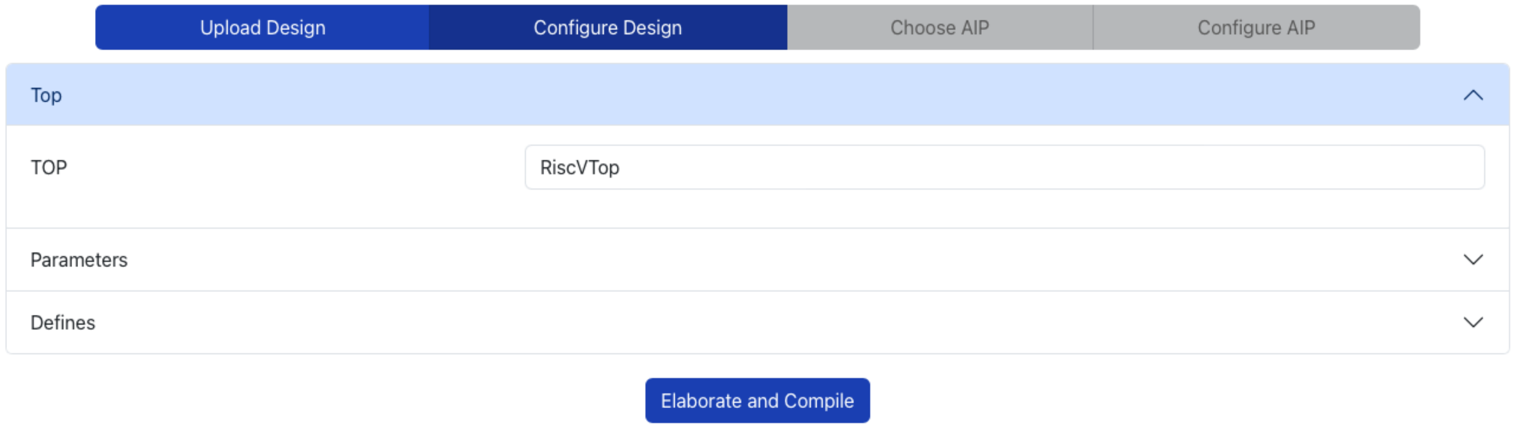


Integrate right into your infrastructure

Hit when done

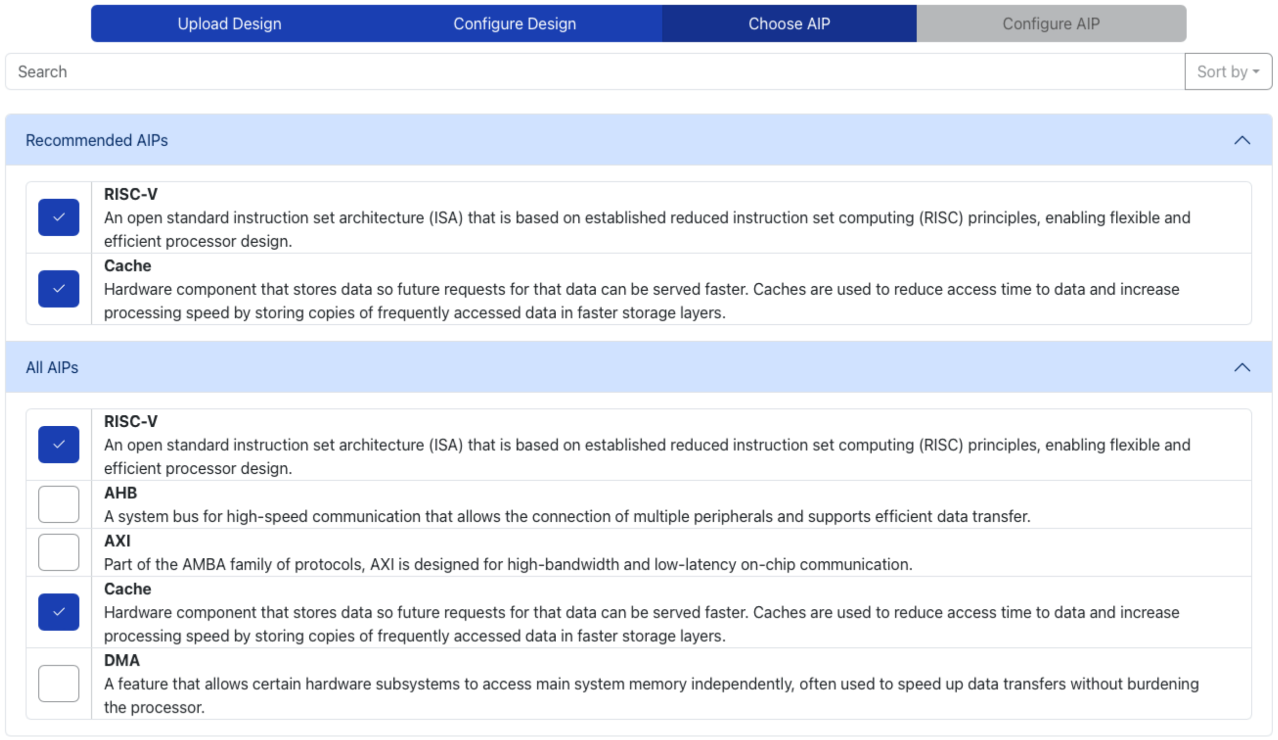
Click after each step

Configure



TOP: RiscVTop

AIP selection



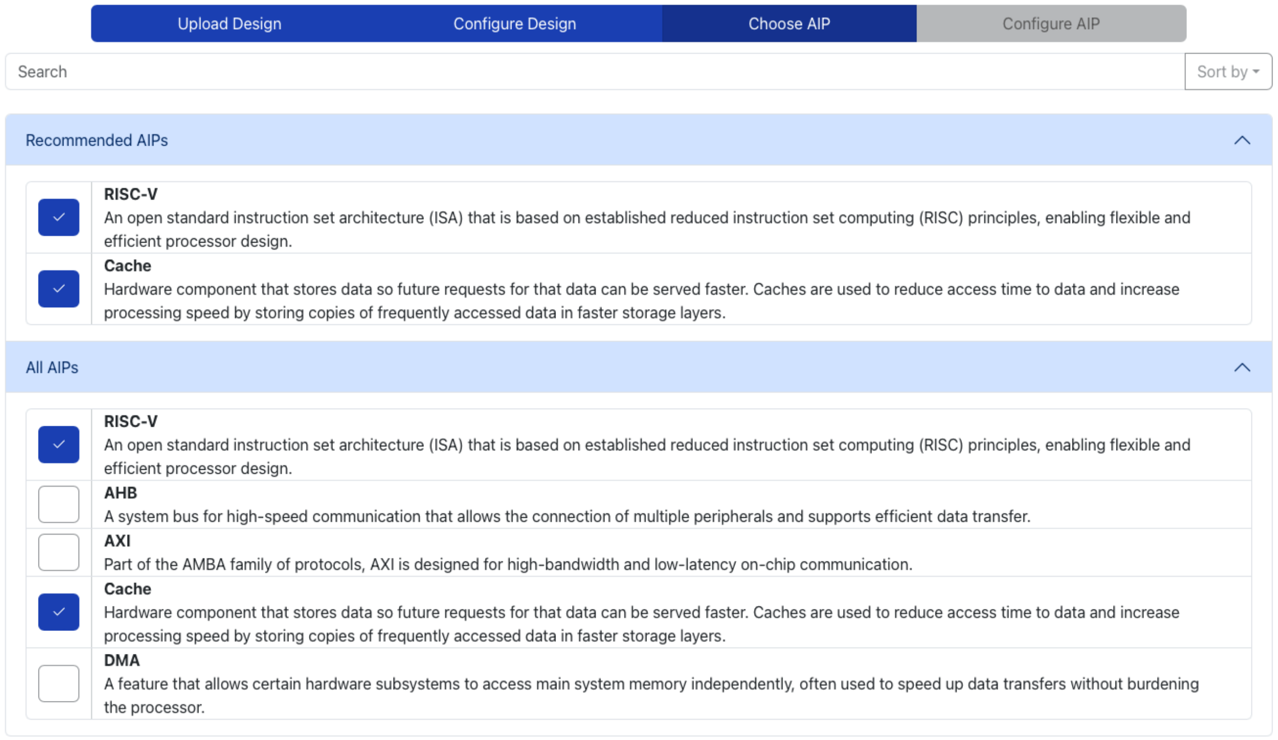
Recommended AIPs

✓	**RISC-V** An open standard instruction set architecture (ISA) that is based on established reduced instruction set computing (RISC) principles, enabling flexible and efficient processor design.
✓	**Cache** Hardware component that stores data so future requests for that data can be served faster. Caches are used to reduce access time to data and increase processing speed by storing copies of frequently accessed data in faster storage layers.

All AIPs

✓	**RISC-V** An open standard instruction set architecture (ISA) that is based on established reduced instruction set computing (RISC) principles, enabling flexible and efficient processor design.
	AHB A system bus for high-speed communication that allows the connection of multiple peripherals and supports efficient data transfer.
	AXI Part of the AMBA family of protocols, AXI is designed for high-bandwidth and low-latency on-chip communication.
✓	**Cache** Hardware component that stores data so future requests for that data can be served faster. Caches are used to reduce access time to data and increase processing speed by storing copies of frequently accessed data in faster storage layers.
	DMA A feature that allows certain hardware subsystems to access main system memory independently, often used to speed up data transfers without burdening the processor.

AIP selection



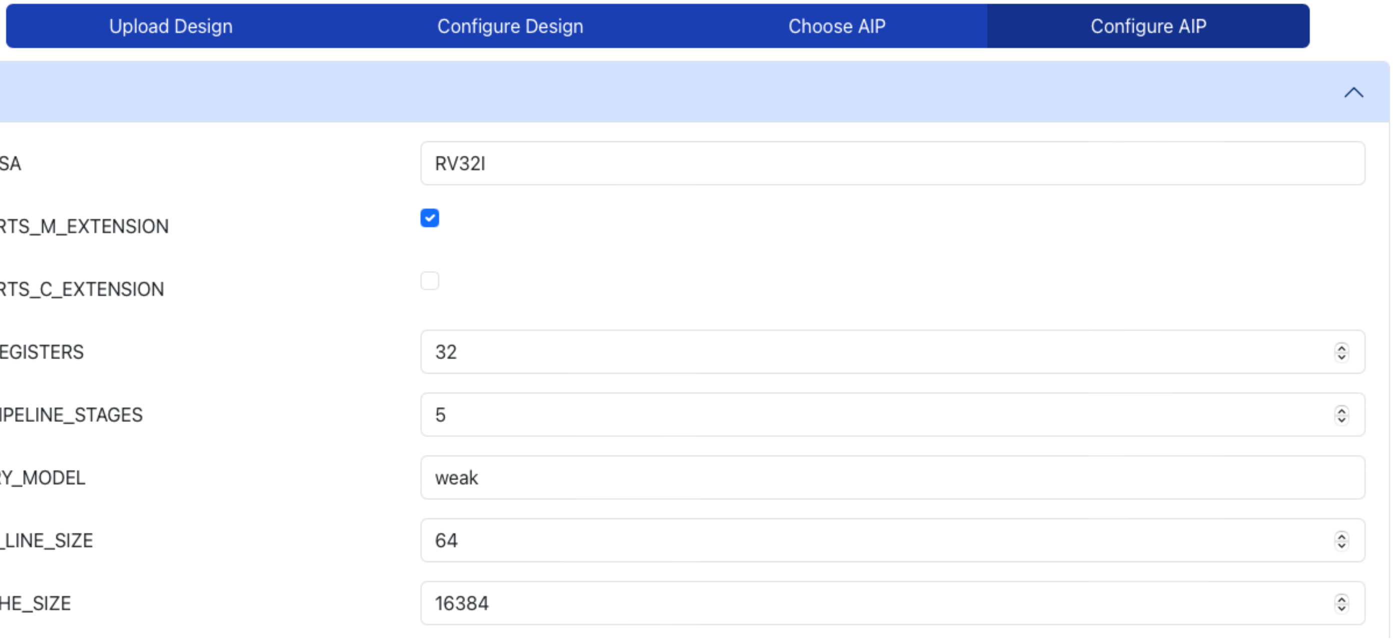
Recommended AIPs

- [✓] **RISC-V**
  An open standard instruction set architecture (ISA) that is based on established reduced instruction set computing (RISC) principles, enabling flexible and efficient processor design.
- [✓] **Cache**
  Hardware component that stores data so future requests for that data can be served faster. Caches are used to reduce access time to data and increase processing speed by storing copies of frequently accessed data in faster storage layers.

All AIPs

- [✓] **RISC-V**
  An open standard instruction set architecture (ISA) that is based on established reduced instruction set computing (RISC) principles, enabling flexible and efficient processor design.
- [ ] **AHB**
  A system bus for high-speed communication that allows the connection of multiple peripherals and supports efficient data transfer.
- [ ] **AXI**
  Part of the AMBA family of protocols, AXI is designed for high-bandwidth and low-latency on-chip communication.
- [✓] **Cache**
  Hardware component that stores data so future requests for that data can be served faster. Caches are used to reduce access time to data and increase processing speed by storing copies of frequently accessed data in faster storage layers.
- [ ] **DMA**
  A feature that allows certain hardware subsystems to access main system memory independently, often used to speed up data transfers without burdening the processor.

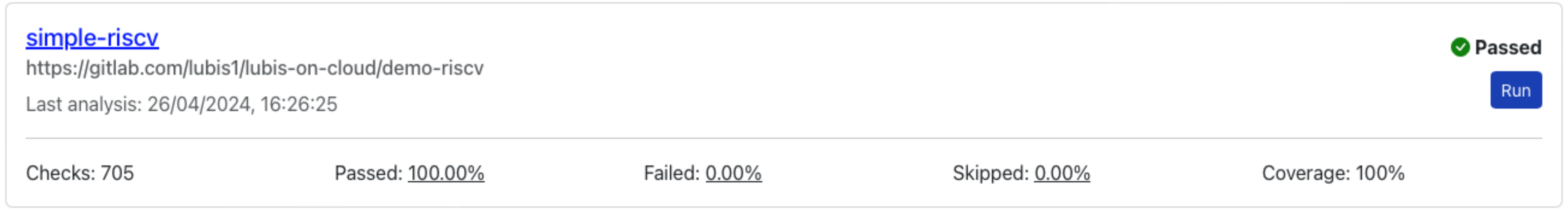
AIP config



CORE_ISA	RV32I
SUPPORTS_M_EXTENSION	✓
SUPPORTS_C_EXTENSION	☐
NUM_REGISTERS	32
NUM_PIPELINE_STAGES	5
MEMORY_MODEL	weak
CACHE_LINE_SIZE	64
L1_CACHE_SIZE	16384

Post setup

Project overview

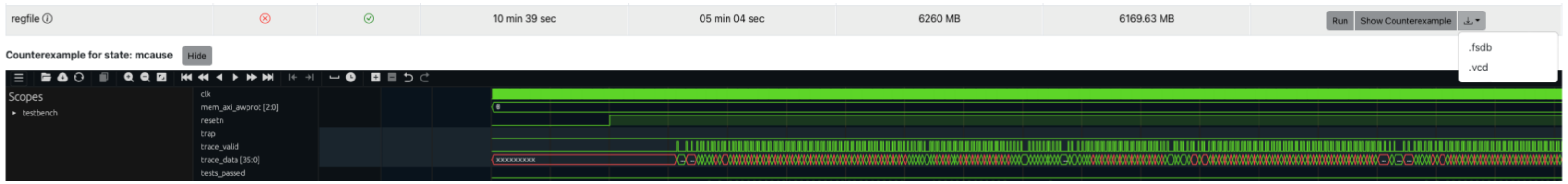


[simple-riscv](#)
https://gitlab.com/lubis1/lubis-on-cloud/demo-riscv
Last analysis: 26/04/2024, 16:26:25

✓ Passed
Run

Checks: 705 Passed: 100.00% Failed: 0.00% Skipped: 0.00% Coverage: 100%

Debugging



regfile ⓘ	⊗	✓	10 min 39 sec	05 min 04 sec	6260 MB	6169.63 MB	Run Show Counterexample

Counterexample for state: mcause

Bug detection machine

1) Everyone: Verify your common blocks!
2) You’re an expert? Don’t do repeating tasks.
3) Use a cloud to make formal scale

Questions?

Follow us to get notified on news about: services, formal training, software or the crypto hashing algorithms

Tobias Ludwig

CEO

Email: Tobias.ludwig@lubis-eda.com

Web: www.lubis-eda.com

LinkedIn