2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Synthesis of Decoder Tables using Formal Verification Tools

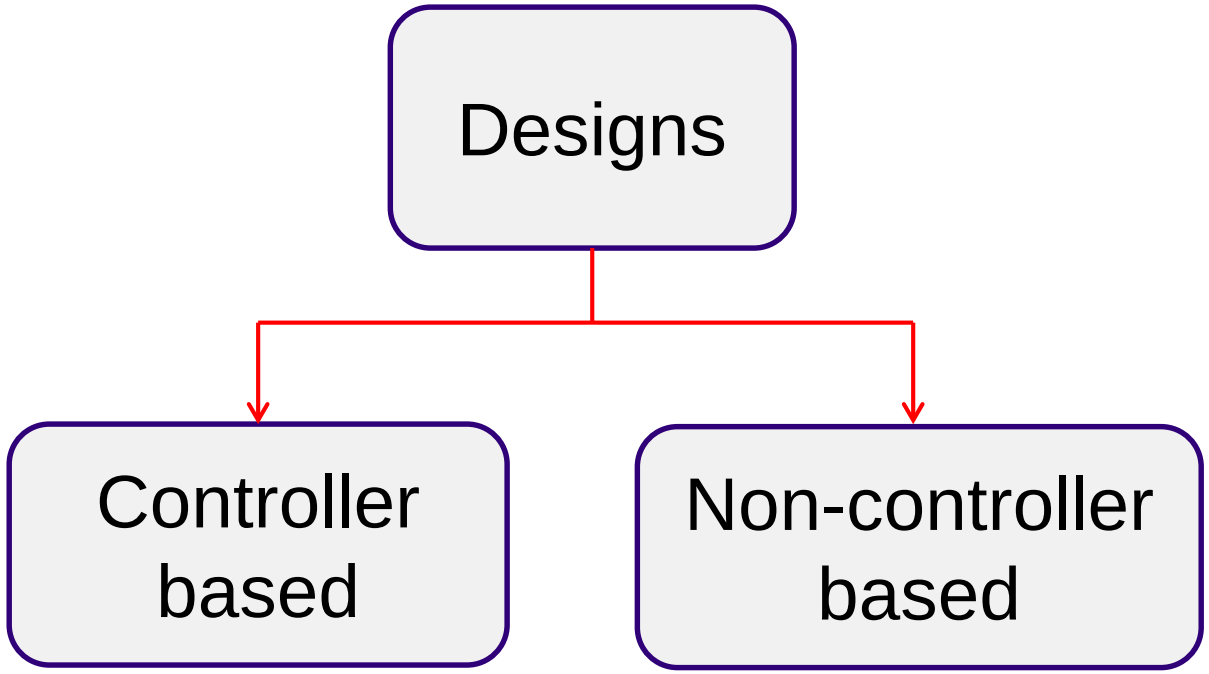
Keerthikumara Devarajegowda, Johannes Schreiner and Wolfgang Ecker

infineon

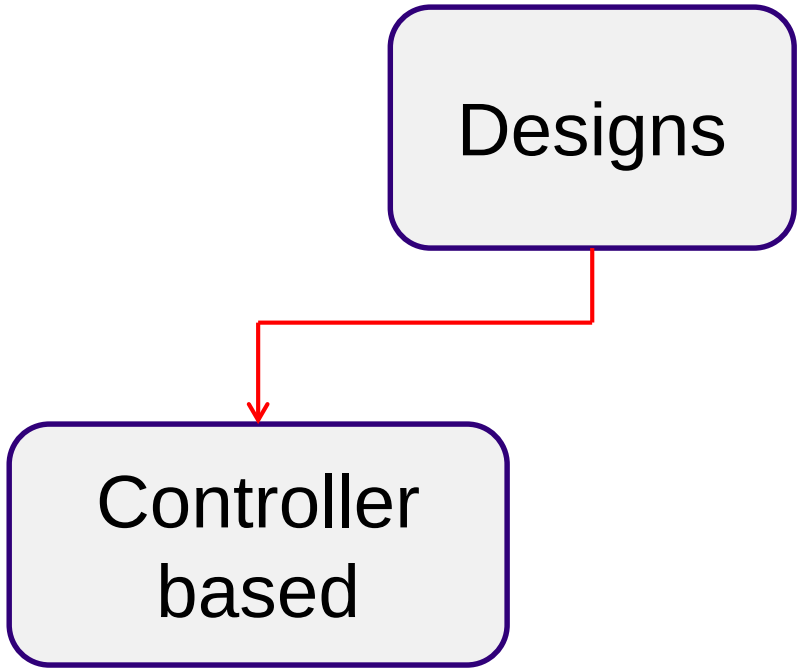
Technische Universität Kaiserslautern

Technische Universität München

Introduction

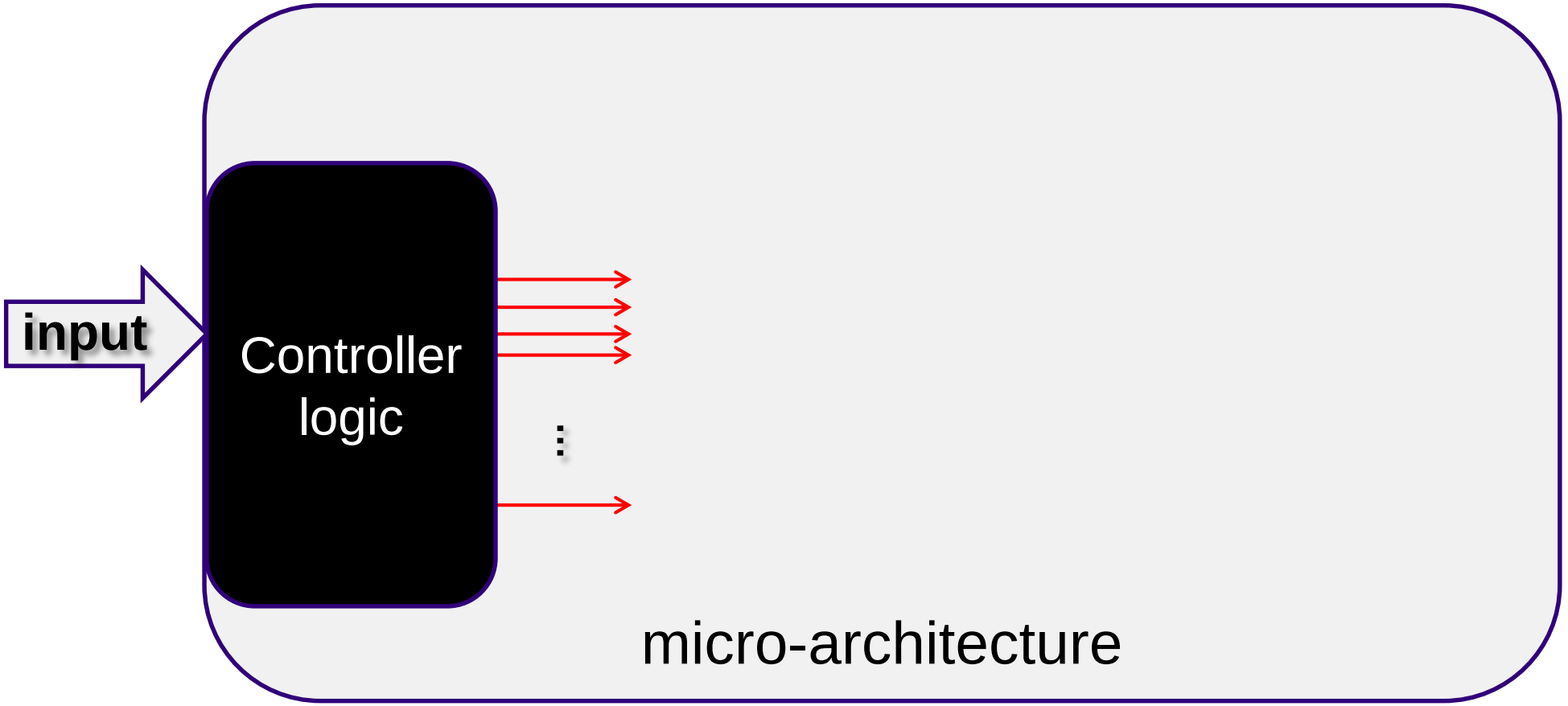


Introduction

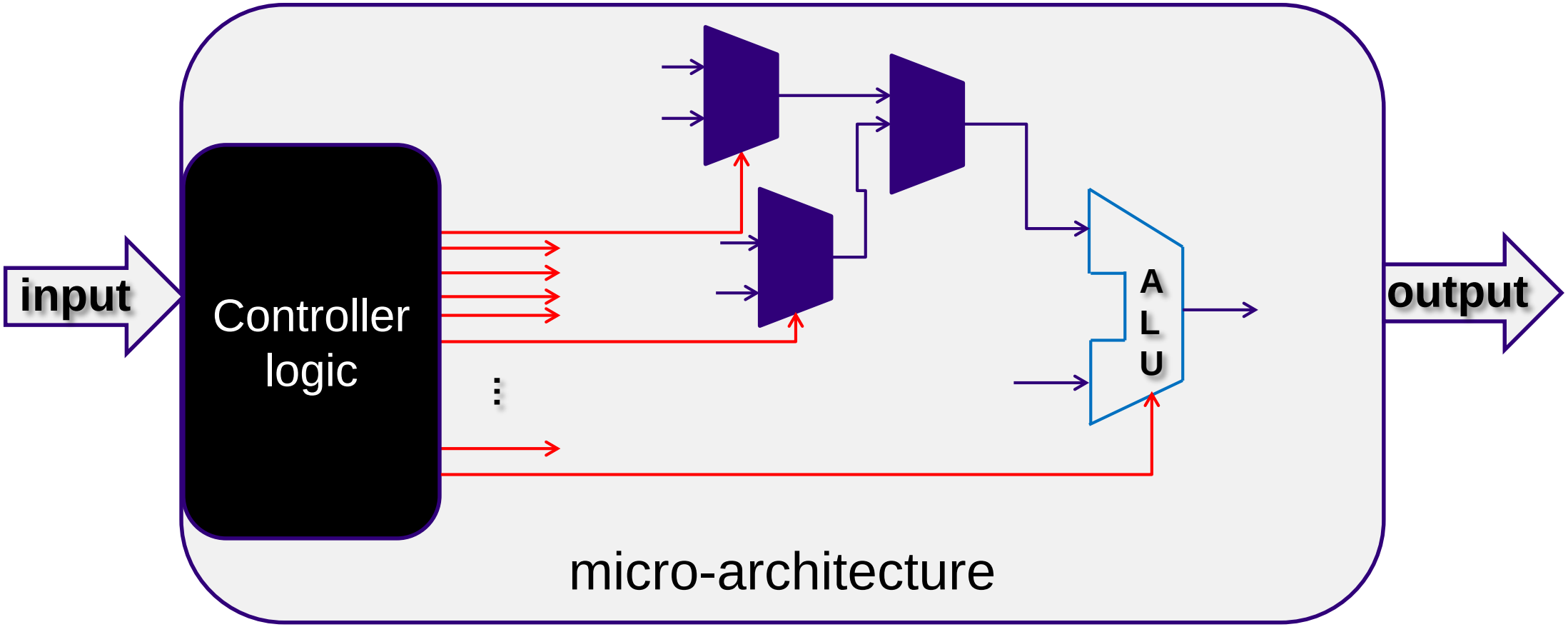


What we mean by **Controller based designs?**

Introduction

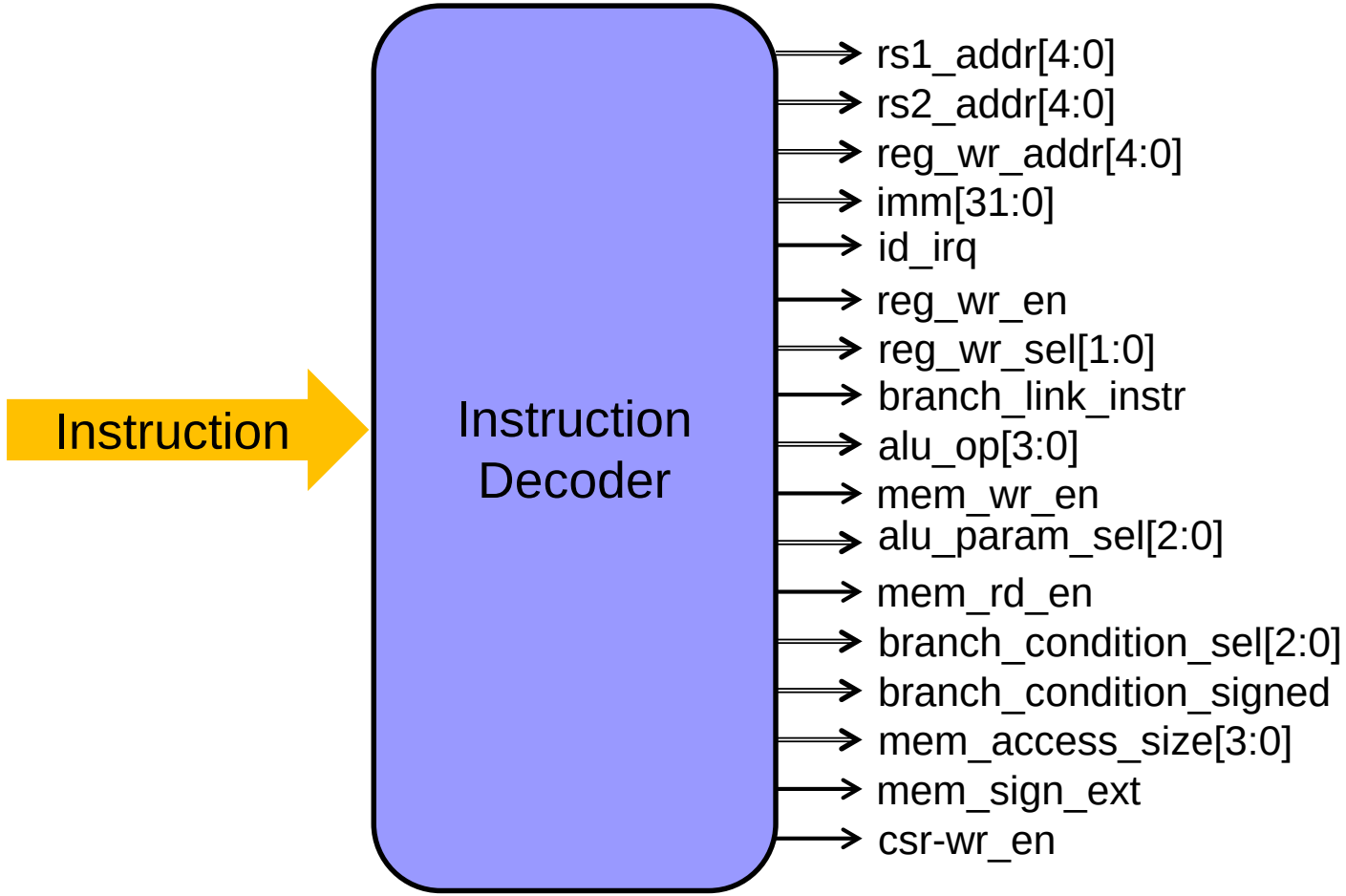


Introduction



Introduction

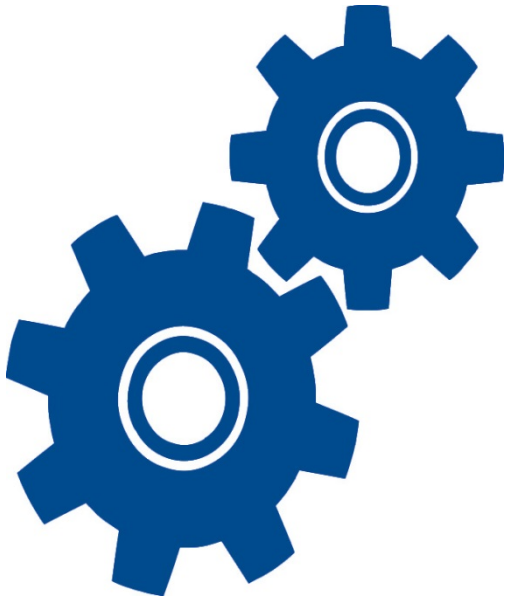
- Typical instruction decoder of a processor
- Decoding is generally implemented using a lookup-table



State-of-the-Art : Manual coding

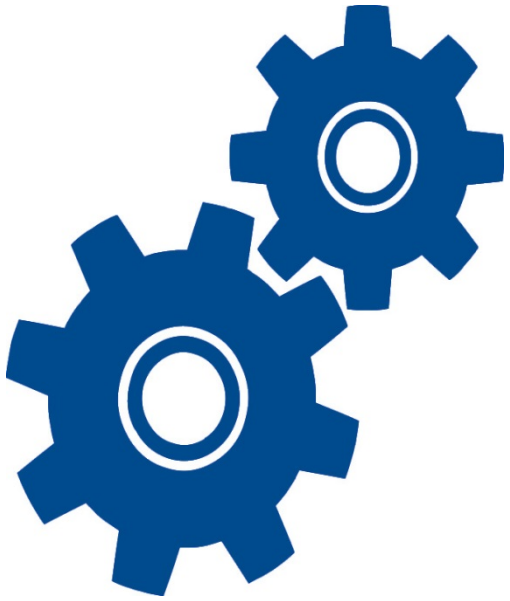
- Design engineer computes control signals
- Requires deep knowledge of the micro-architecture
- Not suitable for late specification changes/extensions
- Often repetitive

(NOT) State-of-the-Art : Simulation tech



- Simulation technique can be used to determine control signals
- Repetitive
- Time-tedious
- Not suitable for medium/large set of control signals

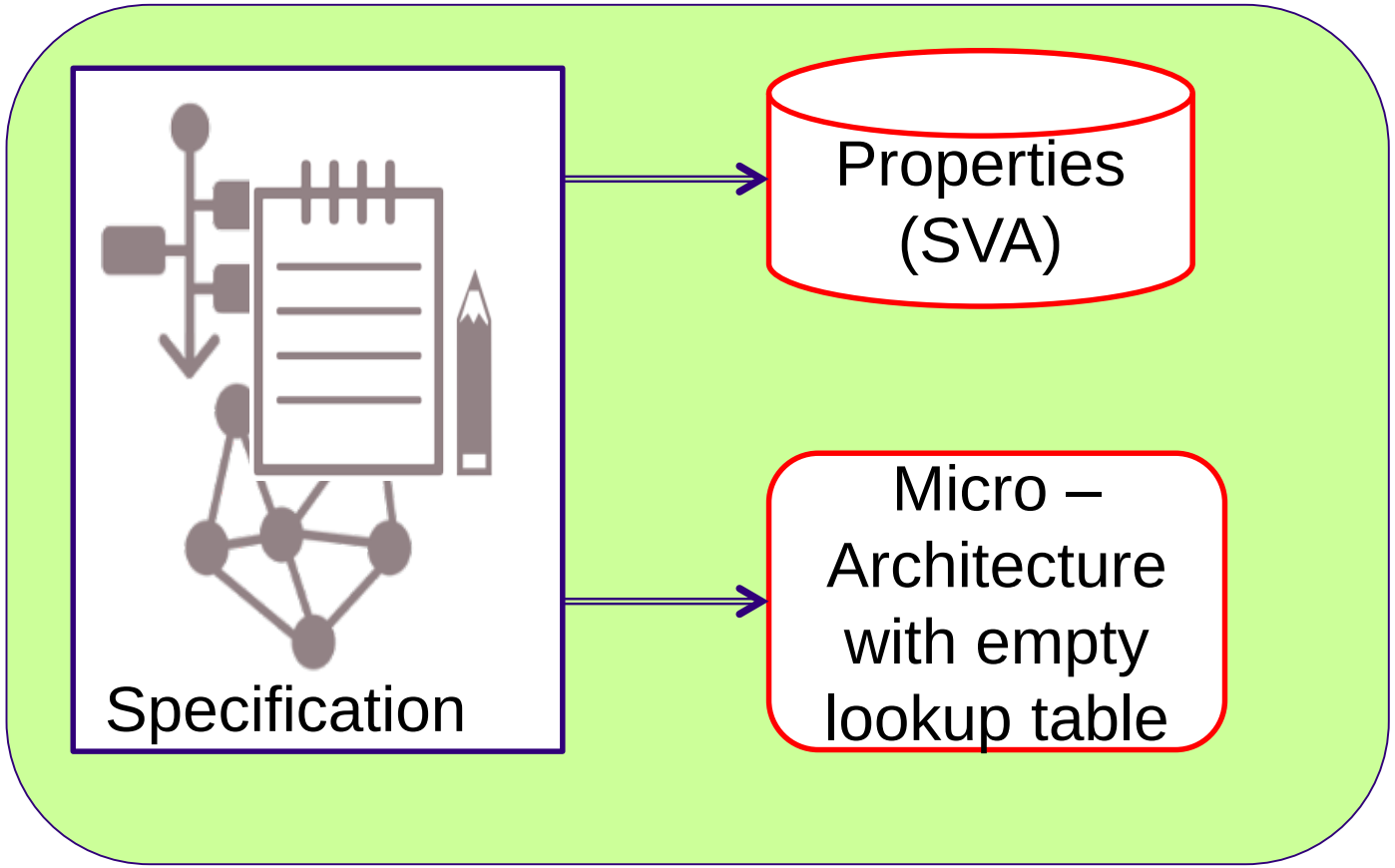
(NOT) State-of-the-Art : Simulation tech



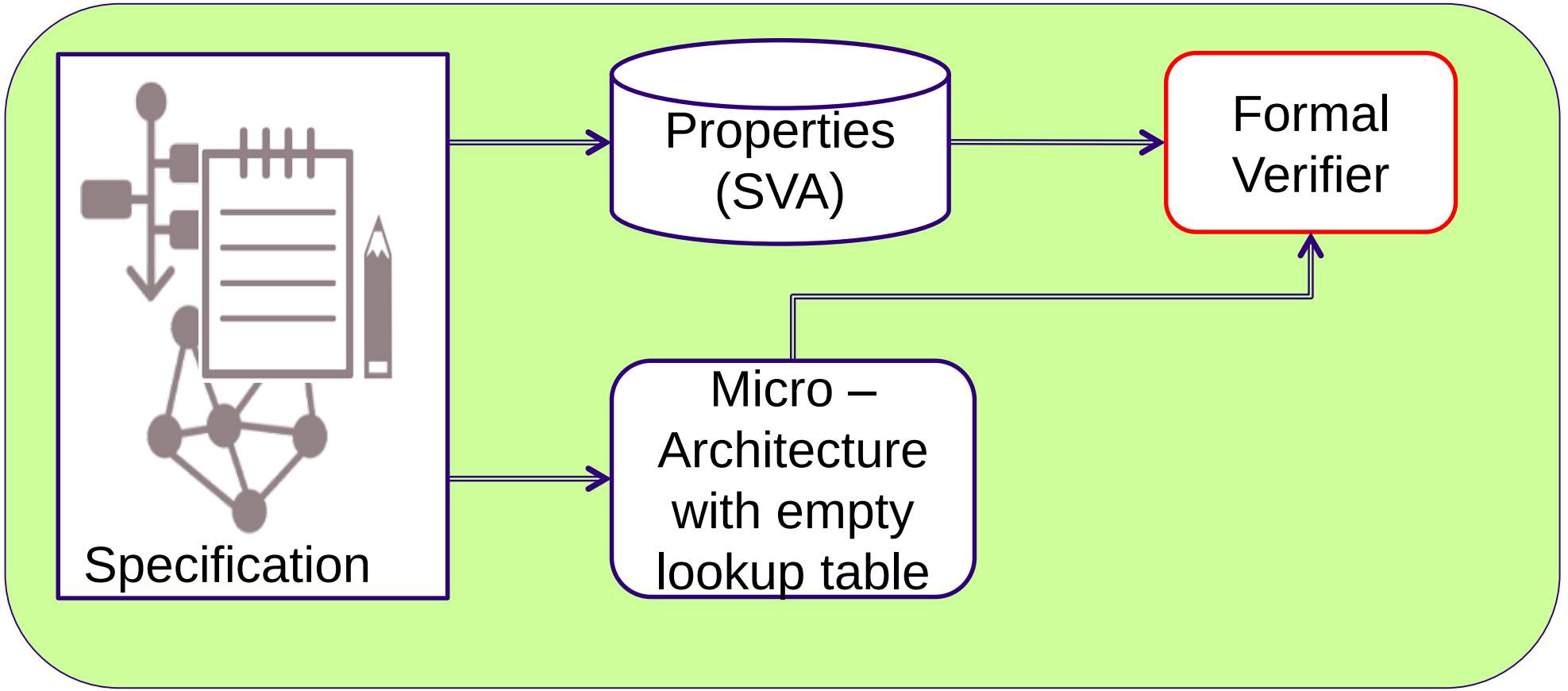
- Simulation technique can be used to determine control signals
- Repetitive
- Time-tedious
- Not suitable for medium/large set of control signals

❖**What about Formal verification tools?**

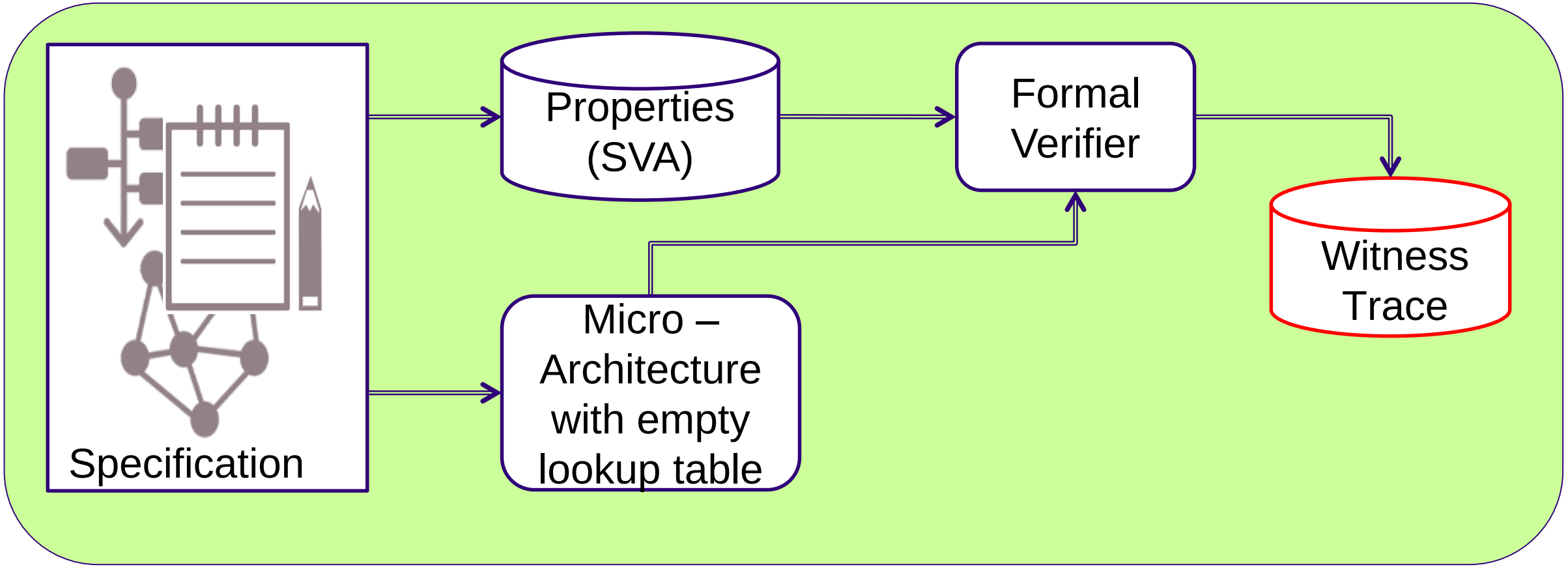
Our idea for synthesis of control signals



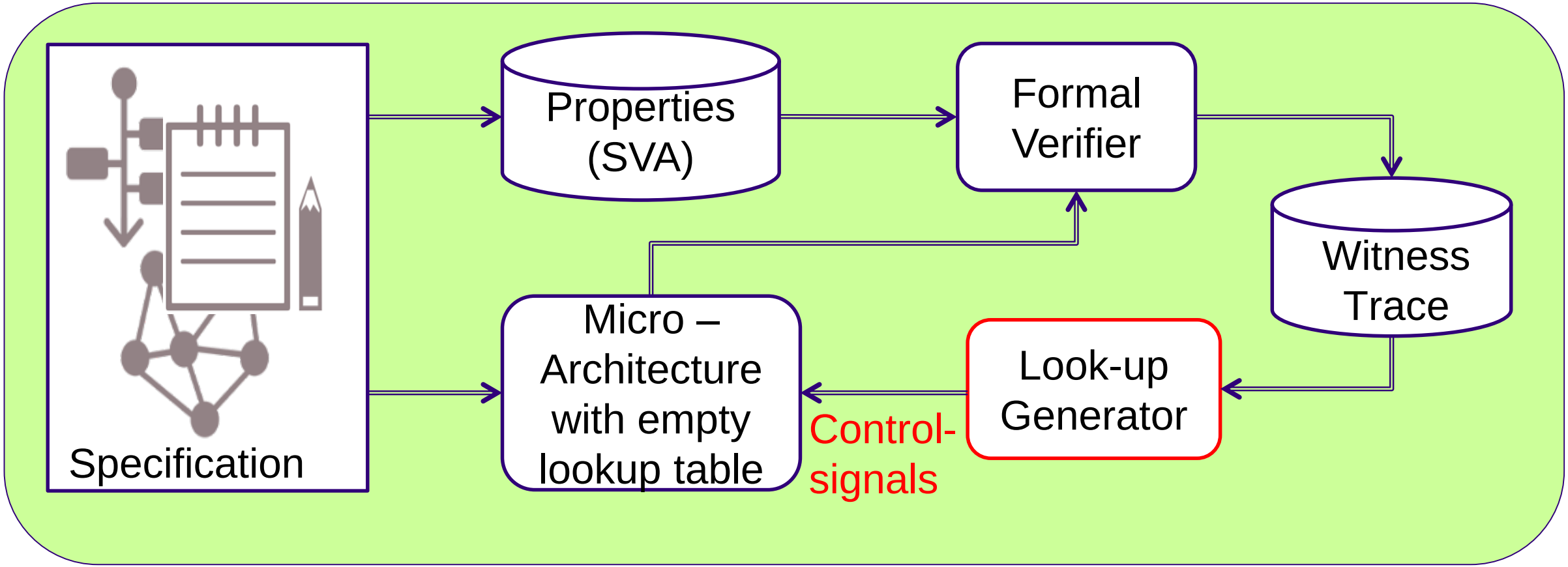
Our idea for synthesis of control signals



Our idea for synthesis of control signals



Our idea for synthesis of control signals



First Try…

- We simply used ***COVER*** directive to extract the decoder settings/control signals
- Did not work!
- The formal tool returned wrong decoder settings

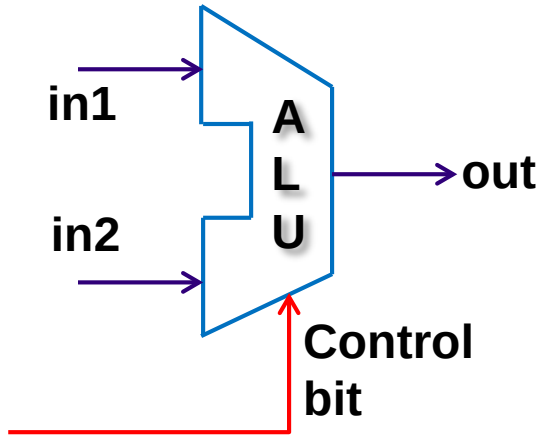
First Try…

- We simply used ***COVER*** directive to extract the decoder settings/control signals
- Did not work!
- The formal tool returned wrong decoder settings

Simple example:

0 + 0 = 0 ADD

0 – 0 = 0 SUB



Second Try…

- Next, we used ***ASSERT*** directive with negation
- ***i.e.,*** ***A -> B*** is transformed to ***A -> not B***
- Since decoder is empty, the tool shall give a ***CEX*** for ***A -> B***

Second Try…

- Next, we used ***ASSERT*** directive with negation
- ***i.e.,*** ***A -> B*** is transformed to ***A -> not B***
- Since decoder is empty, the tool shall give a ***CEX*** for ***A -> B***
- But, **De Morgan's law**
- Assume *B* = *b1* and *b2* and *b3*

  not B = *not b1* or *not b2* or *not b3*
- ***OR –*** operation allows alternative settings

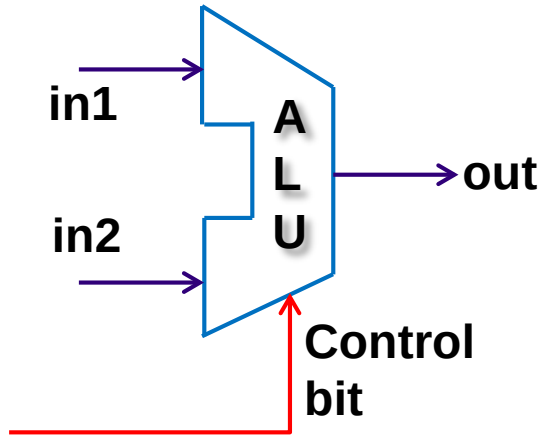
Solution!

- Finally, we extracted the correct control signals by excluding other operations
- ***A -> (B*** and ***not C)***

Solution!

- Finally, we extracted the correct control signals by excluding other operations
- ***A*** **->** (***B*** and ***not C)***
- Simple example: *(out* = *in1 + in2)* and

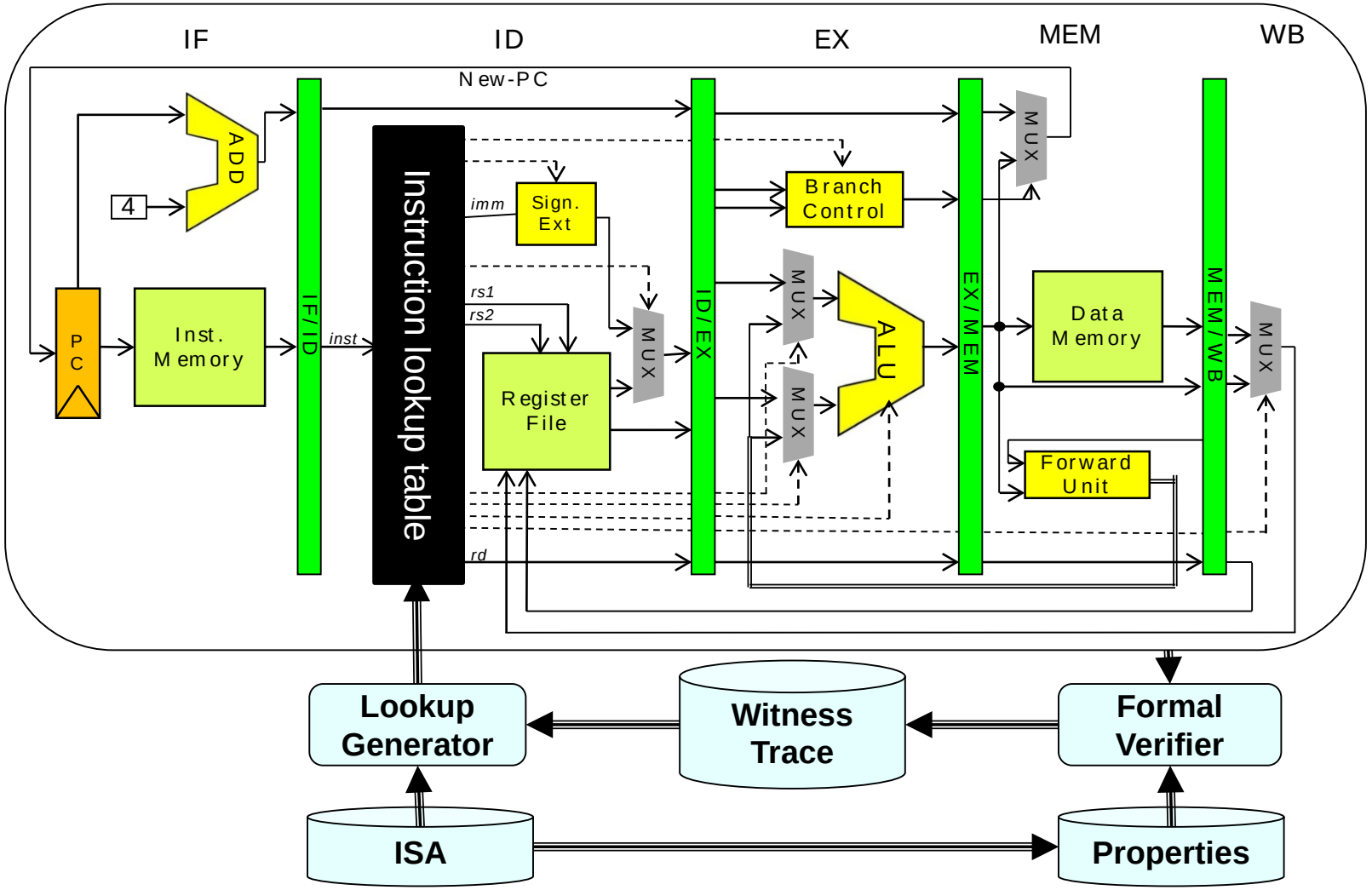
  (out != *in1 – in2)*



- The ***COVER*** directive shall give correct control values
- Since we generate the properties, generating exclusion is not a big effort

Case Study: 5-stage RISCV Core

- **5-stage RISCV core** with an **empty lookup table** in the instruction decoder
- Semi-automated approach
- Python based in-house automation framework



Case Study: 5-stage RISCV Core

```
1   property _ADD;
2   @(posedge clk) disable iff(reset)
3    assume_add_instr
4    |->
5    ##0
6       rs1_addr    == instr[19:15] &&
7       rs2_addr    == instr[24:20] &&
8       reg_wr_addr == instr[11:7]  &&
9    ##1
10      alu_in1 == $past(rs1_data)       &&
11      alu_in2 == $past(rs2_data)       &&
12      alu_result == alu_in_1 + alu_in2 &&
13      exclude_other_ops
14   ##1
15      !data_mem_wr_en
16   ##1
17       reg_wr_data == $past(alu_result,2) &&
18       reg_wr_en                          &&
19       reg_wr_addr == $past(instr[11:7],3);
20  endproperty
21  COV_add_instr: cover property(_ADD);
```

- SVA: Similar approach (as RTL) for Property generation
- Python based automation framework
- A property for every instruction
- Property captures the required signal behavior of the micro-architecture
- Later re-used for verification

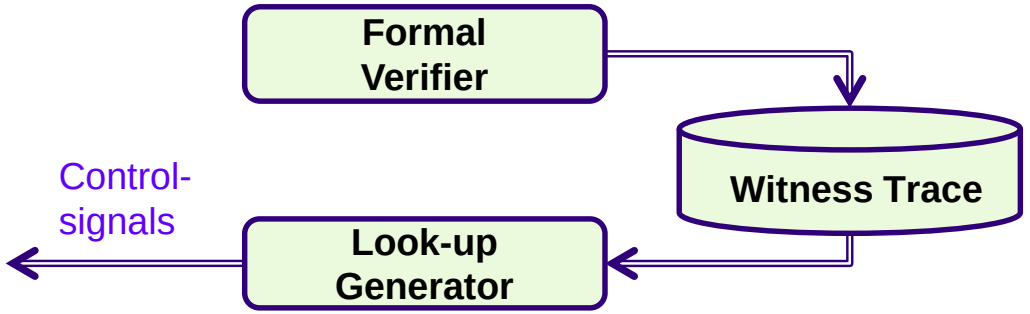
Case Study: 5-stage RISCV Core

```
1   property _ADD;
2   @(posedge clk) disable iff(reset)
3    assume_add_instr
4    |->
5    ##0
6       rs1_addr    == instr[19:15] &&
7       rs2_addr    == instr[24:20] &&
8       reg_wr_addr == instr[11:7]  &&
9    ##1
10      alu_in1 == $past(rs1_data)        &&
11      alu_in2 == $past(rs2_data)        &&
12      alu_result == alu_in_1 + alu_in2 &&
13      exclude_other_ops
14  ##1
15     !data_mem_wr_en
16  ##1
17      reg_wr_data == $past(alu_result,2) &&
18      reg_wr_en                          &&
19      reg_wr_addr == $past(instr[11:7],3);
20  endproperty
21  COV_add_instr: cover property(_ADD);
```

```
alu_result != (alu_in1 >> alu_in2[4:0])   &&
alu_result != (alu_in1 & alu_in2))        &&
alu_result != (alu_in1 << (alu_in2[4:0])))&&
alu_result != (alu_in1 - alu_in2))        &&
alu_result != (alu_in1 < alu_in2))        &&
alu_result != (alu_in1 | alu_in2))
```

Scripting to automate extraction

- We used formal tool Onespin 360 for our work
- But our approach shall work with all formal tools



- The tool allows to select specific signal values at specific time-points
- Script to parse the log file and extract control signal values
- Supplement RTL generator with required control signals

Conclusion

- ✓ Formal methods are powerful techniques and shall be used to support design development as well
- ✓ Significant reduction of boring and error prone manual efforts
- ✓ Developed SVAs are re-used for verification
- ✓ Quick turn-around time for late changes and/or extensions
- ✓ The approach is applicable for wide-variety of problems as soft and hard configuration of modules

2018 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Thank you!