Synthesis of Decoder Tables Using Formal Verification Tools

Keerthikumara Devarajegowda[1,2] and Johannes Schreiner[1,3] and Wolfgang Ecker[1,3]
Infineon Technologies AG[1] / Technische Universität Kaiserslautern[2] / Technische Universität München[3]

1. Abstract

- Classical hardware design tasks involve constructing a micro-architecture and a control unit that makes the hardware unit functional.
- Traditionally, these control units are built by analyzing the micro-architecture implementation and the intended function.
- We introduce a novel approach for automatic synthesis of control signals for a given micro-architecture.
- Our approach uses formal methods to automatically derive a working control unit and/or decoder for a certain micro-architecture.
- For synthesizing control signals of the decoder unit, we formulate a set of SystemVerilog-Assertions (SVA).
- The developed SVAs are applied to the formal tool to extract control signals and the same SVAs are later re-used for design verification with minor refinement.

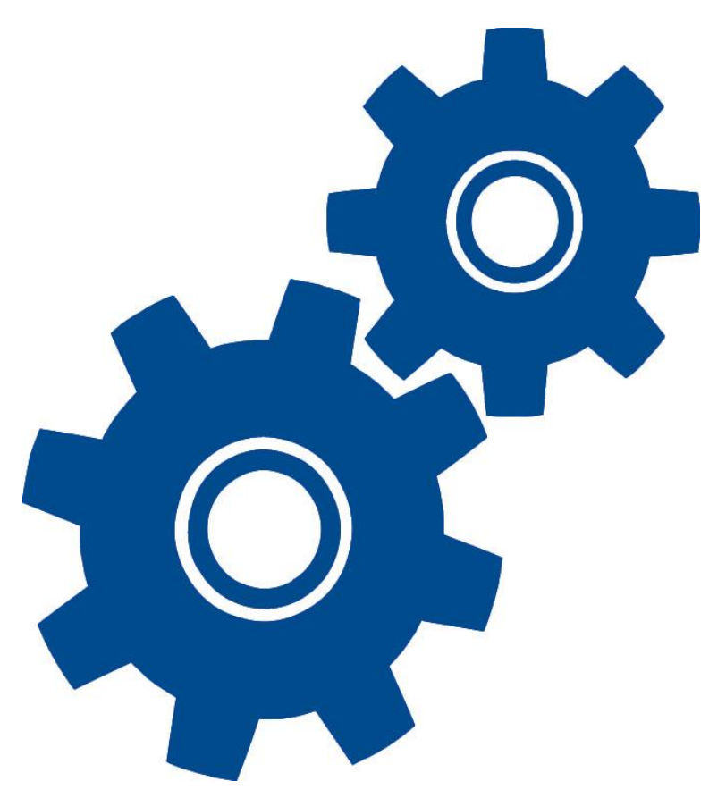
2. State of the Art

How are decoders / control-signals built today?

➢ ***Manual Coding***

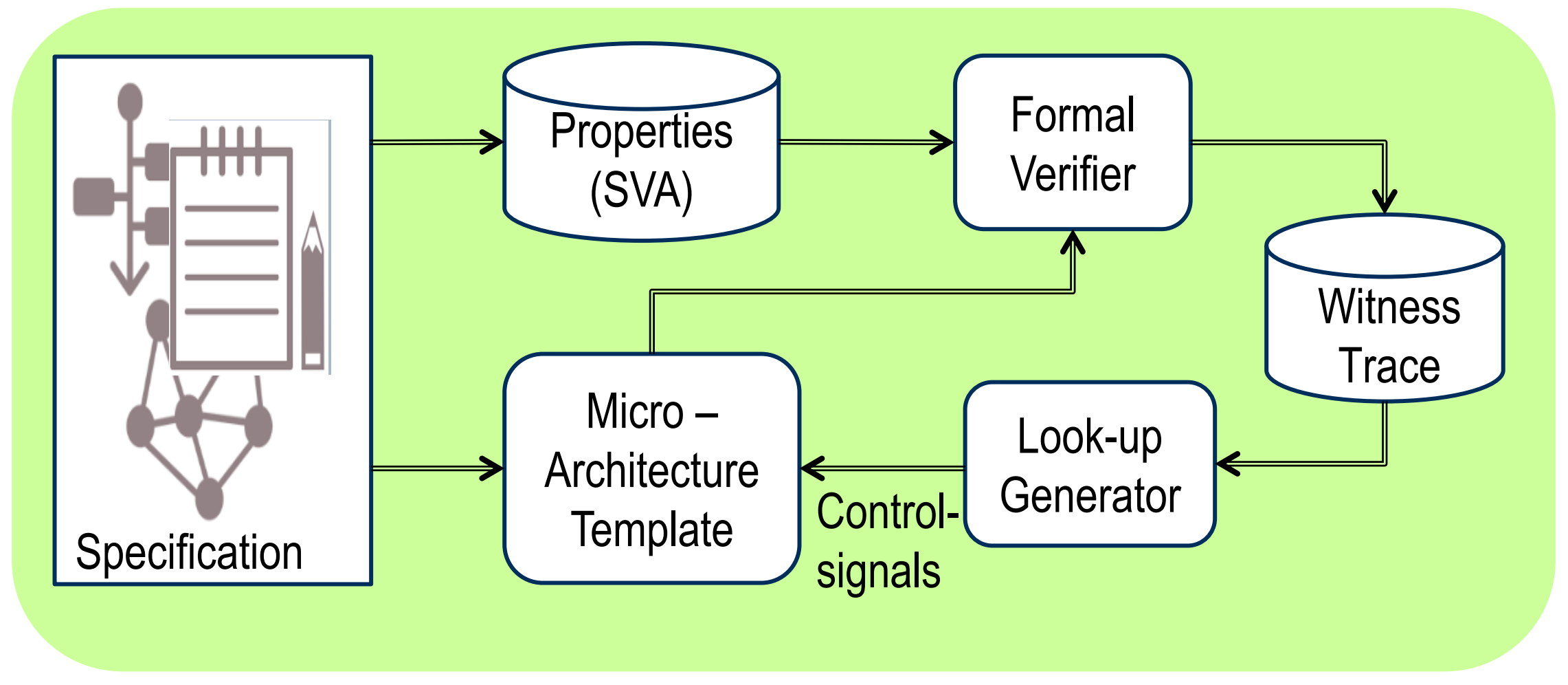
- ❑ Design engineer computes control signals
- Repetitive
- Requires deep knowledge of the micro-architecture
- Not suitable for late specification changes/extensions

➢ ***Using Simulation Technique***



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

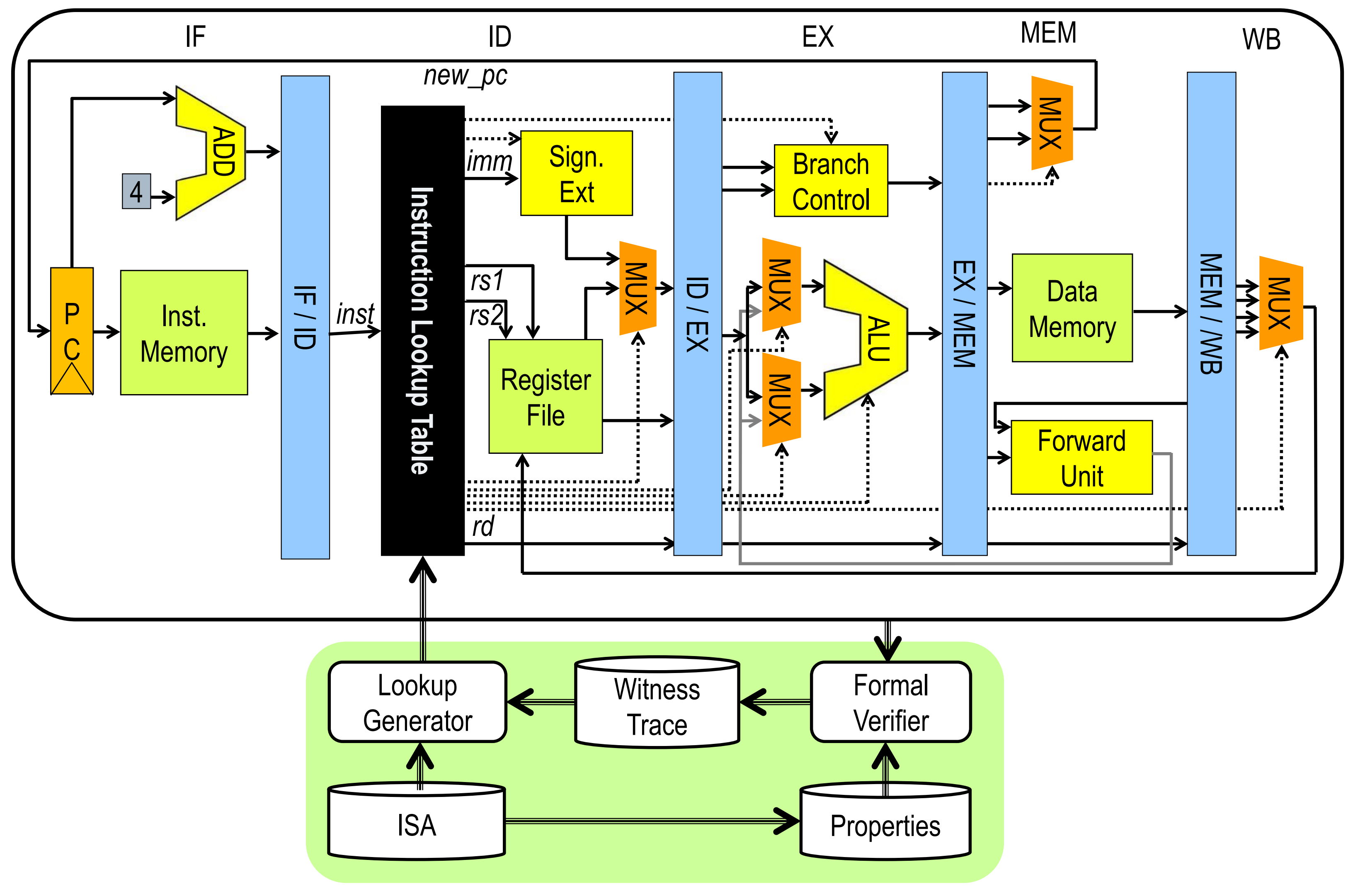
3. Synthesis of decoder unit using Formal Tools



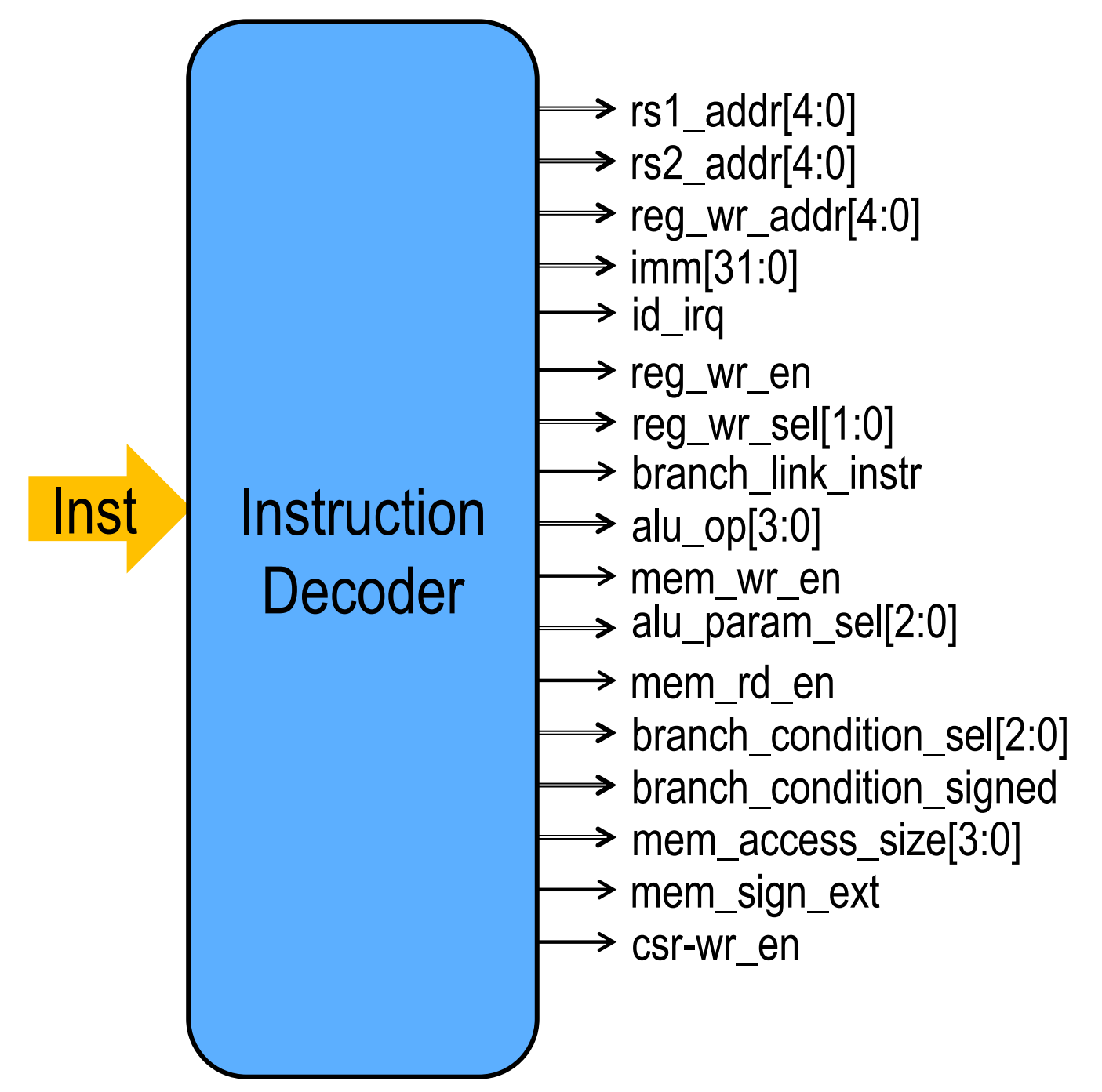
General approach for synthesizing control signals

1. The micro-architecture with an empty look-up-table(decoding unit) and SVAs are read-in to the formal tool.
2. SVAs capture the intended behavior of the design for specific operation (shall be coded to exclude operations other than the intended operation).
3. Witness traces from the formal tool are investigated to extract required control signal values at certain time-points.
4. Extracted control-signals are supplemented to the micro-architecture to complete the look-up-table.

4. Case Study : Control Signals of an instruction decoder



Automatic synthesis of control signals for a 5-stage Processor Unit



Typical control signals of an instruction decoder.

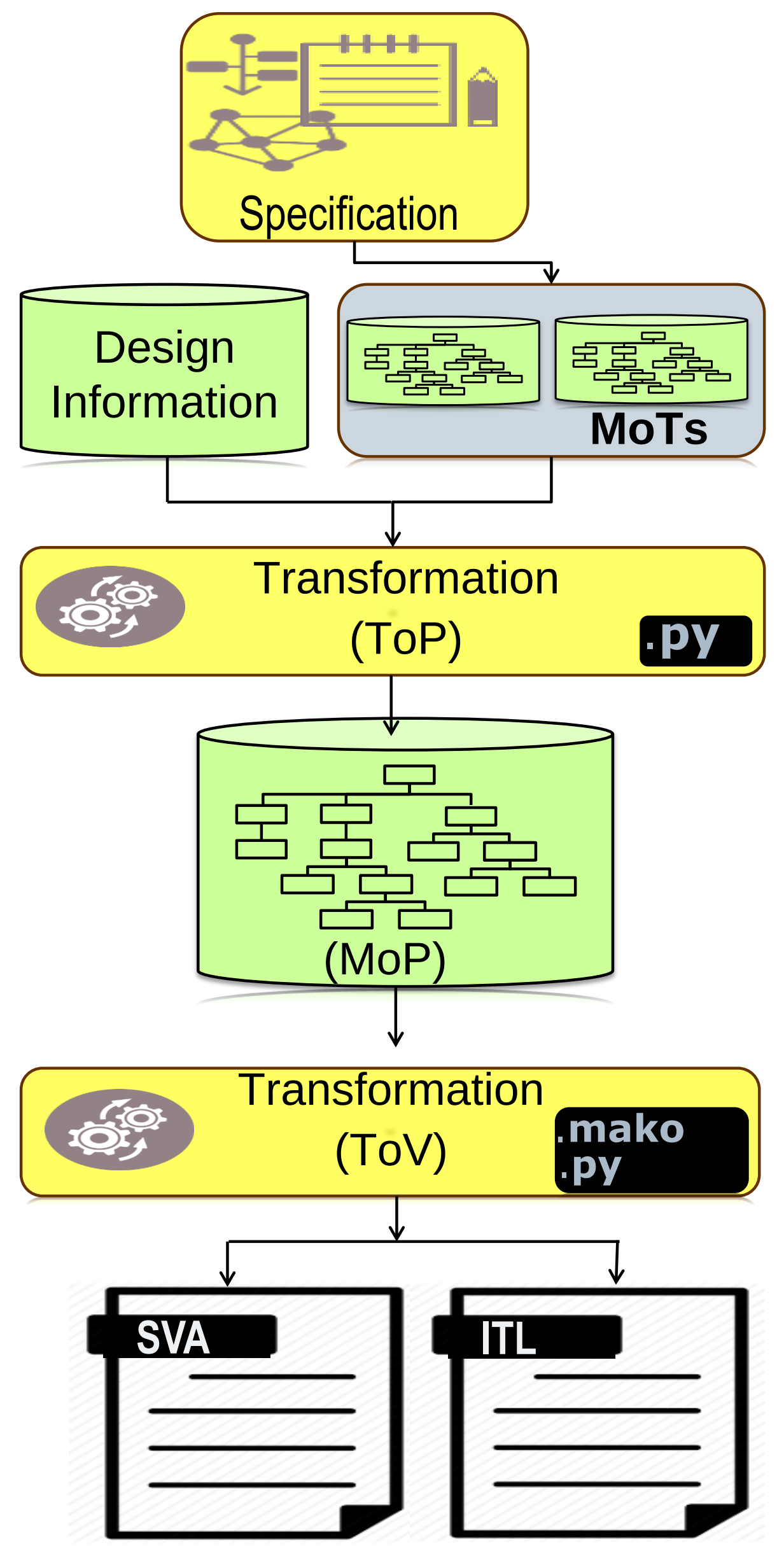
(Displays the complexity involved in deducing large set of control signals)

1. A 5-stage and a 3-stage pipelined processor architectures supporting the RISC-V RV32I Base Integer Instruction Set are used as the test vehicles.
2. The design and SVAs are automatically made using an in-house generation framework. We set-up the look-up-table for decoding instructions.
3. A property is generated for each instruction, capturing required signal behavior over a period of 3-to-5 clock cycles.
4. Pipelined processor architecture implementations with an empty look-up-table and SVAs are applied to the formal tool.
5. Control signals are automatically extracted from the witness traces (covers) provided by the tool.
6. Extracted control signals are supplied to the look-up-table to complete the micro-architecture.

Instruction	reg_wr_en	reg_wr_sel	branch_link_inst	alu_op	mem_wr_en	alu_param_sel	mem_rd_en	branch_condition_sel	branch_cond_sign	mem_acc_size	mem_sign-ext	csr_wr
AND	1	00	0	0110	0	000	0	000	0	0000	0	0
OR	1	00	0	0111	0	000	0	011	0	0000	0	0
SLL	1	00	0	0010	0	000	0	011	0	0000	0	0
SUB	1	00	0	0001	0	000	0	010	0	0000	0	0
ADD	1	00	0	0000	0	000	0	000	0	0000	0	0
SRL	1	00	0	0100	0	000	0	011	0	0000	0	0
SLTU	1	00	0	1010	0	000	0	011	0	0000	0	0
XOR	1	00	0	1000	0	000	0	000	0	0000	0	0
SRA	1	00	0	0101	0	000	0	000	0	0000	0	0
SLT	1	00	0	1001	0	000	0	011	0	0000	0	0

Generated control signals for R-Type instructions of RISC-V RV 32I instruction set

5. Property generation



Property generation flow

1. Property generation framework is based on Model-Driven-Architecture principle from OMG
2. MoTs (Model-of-Things) are formalized specification
3. ToP (Templates-of-Property) extracts information from MoT and DUV to define the property trace
4. MoP (Model-of-Property) is platform independent means of defining property traces
5. ToV (Templates-of-View) finally transform MoPs into properties in multiple languages

6. Conclusion

- ✓ **Significant reduction of 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**
- ✓ **A new direction of application for formal verification tools**

Contact : *Wolfgang.Ecker@infineon.com*
Keerthikumara.Devarajegowda@infineon.com
Johannes.Schreiner@infineon.com