Complete Formal Verification of a Family of Automotive DSPs

Rafal Baranowski, Marco Trunzer

Robert Bosch GmbH, Reutlingen, Germany

Challenge

- Verification of an in-house family of Digital Signal Processors (DSP) with safety requirements
  - three-stage SIMD pipeline with in-order execution
- DSP family members vary in:
  - instruction set (ASIP-like features)
  - number and width of data paths and registers
  - safety features, peripheral functionality, etc.
- Family is open
  - future variants are yet to be specified

Goals

- Good maintainability of the verification framework
  - little effort to verify new DSPs
- Early verification start
- 100% functional coverage at affordable cost

Agenda

- Introduction
- Completeness Concepts
- Verification Process
- Semi-Formal Specification
- Implementation & Results
- Conclusion

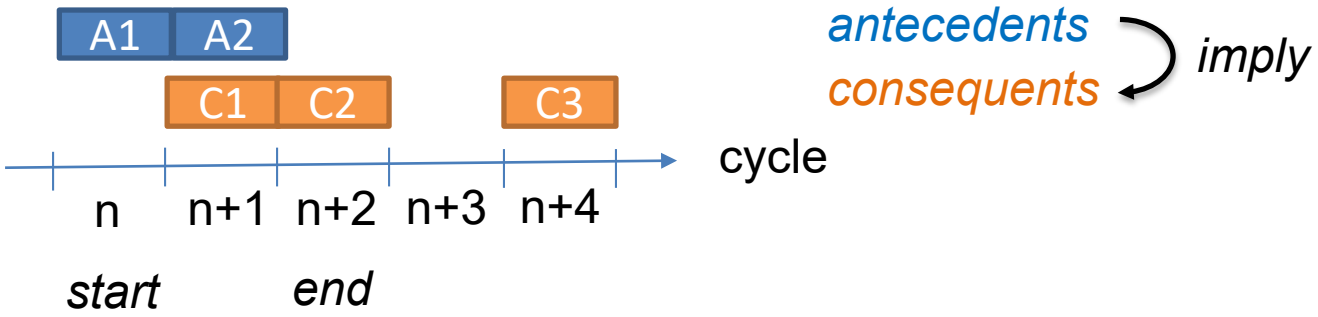
Complete Formal Specification

- Formal specification is **complete** when any two implementations adhering to it are equivalent
  [Bormann & Busch 2005, Claessen 2007]

- Complete formal specification is derived:
  - manually from informal specifications
    [Bormann et al. 2007, OneSpin GapFree Methodology]
  - automatically from intermediate formal models
    [Kühne et al. 2010, Loitz et al. 2010]

Operational Property

- Basic component of a complete formal specification
- Defined as **implication** over time
  - e.g.

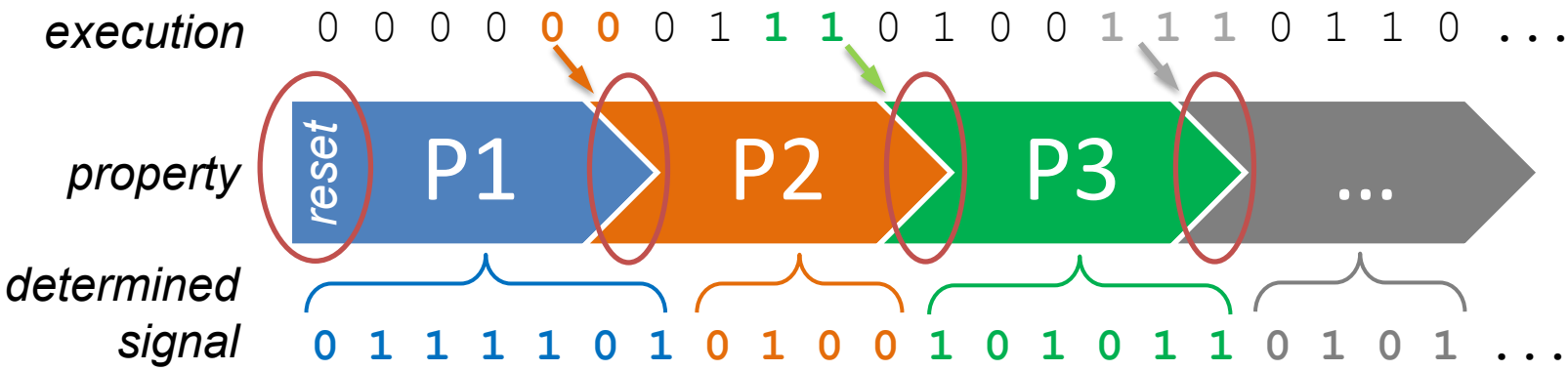


- Has a defined **start** and **end** cycle for sequencing
  - end cycle = start cycle of the next operational property
  - sequencing defined with a “property graph”

Completeness Requirements

- Every execution trace must seamlessly match to a sequence of properties, such that:
  - around every start cycle, the antecedent of exactly one property is satisfied
  - in every clock cycle, the state of all relevant signals is unambiguously determined (specified) by the consequents of consecutive properties

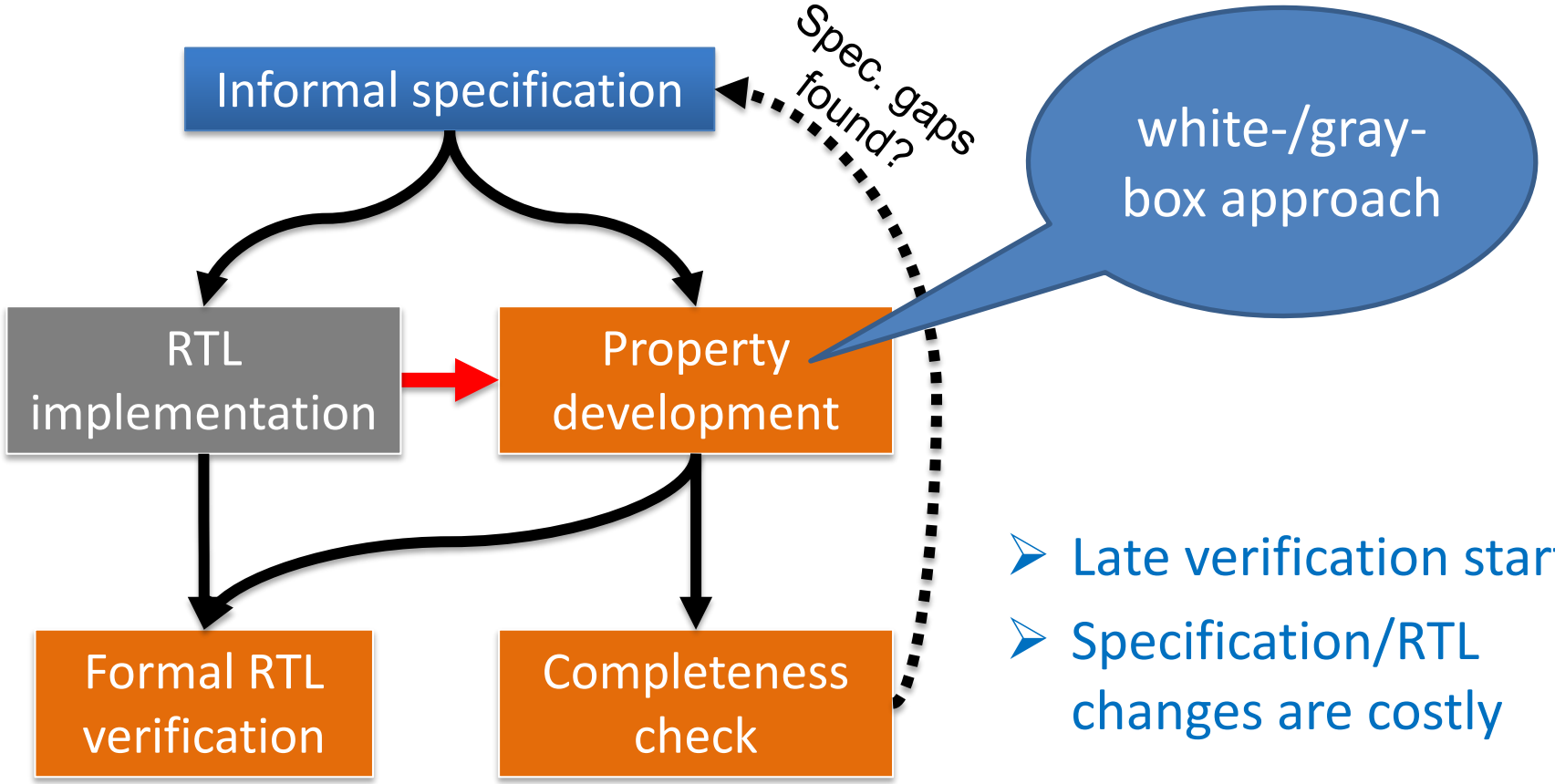
[Bormann et al. 2007]



Agenda

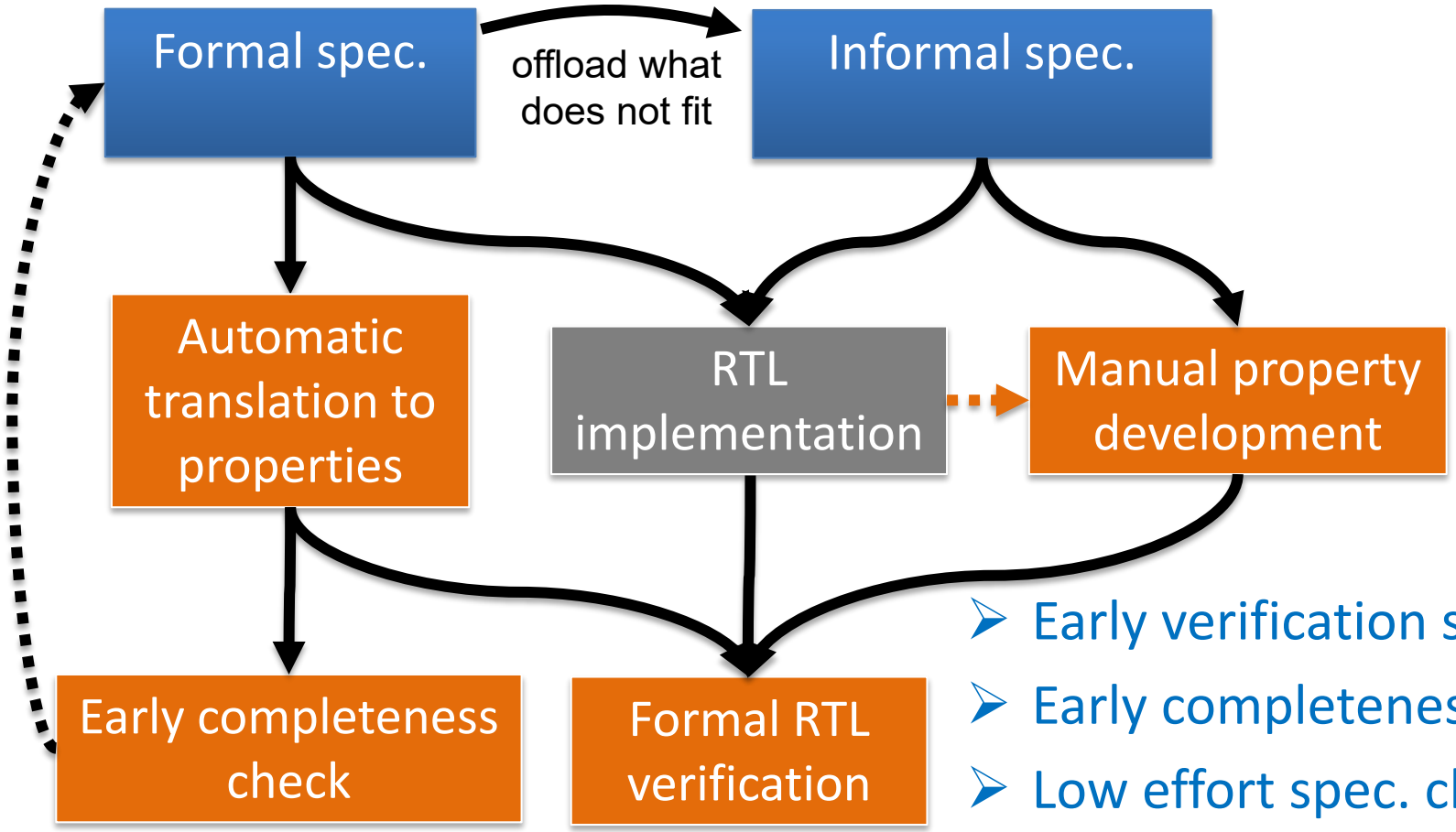
- Introduction
- Completeness Concepts
- Verification Process
- Semi-Formal Specification
- Implementation & Results
- Conclusion

Traditional Formal Verification Process



- Late verification start
- Specification/RTL changes are costly

Verification Process with Semi-Formal Specification

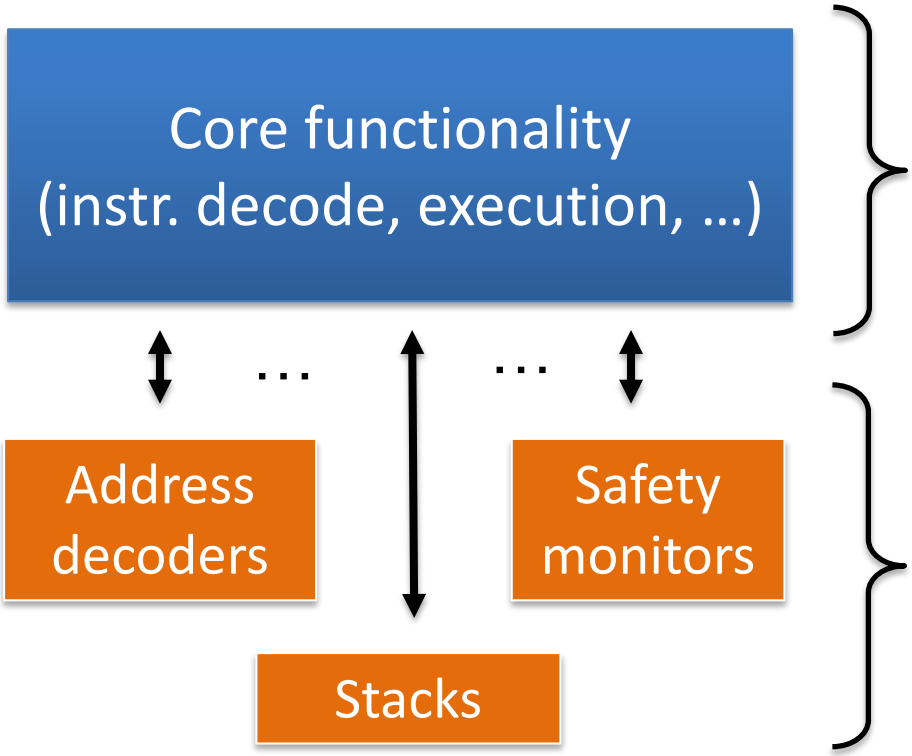


- Early verification start
- Early completeness check
- Low effort spec. changes

Agenda

- Introduction
- Completeness Concepts
- Verification Process
- Semi-Formal Specification
- Implementation & Results
- Conclusion

Semi-Formal Specification



covered by
formal (tabular) specification

covered by
informal specification

Tabular (Formal) Specification

- Built upon instruction set description
- Defines **triggers** and **commitments** of instructions
- Each row is a cycle: start cycle = n, end cycle = n+1
- Example:

Mnemonic	Instruction [n]	PC [n+1]	ACC [n+2]
NOP	0000000	PC[n]+1	stable

Tabular (Formal) Specification (2)

- Features:
  - bit patterns
  - expressions in SystemVerilog
  - overlay rows for conditional execution, interrupts, etc

stack functionality in informal spec.

Mnemonic	Instruction [n]	PC [n+1]	ACC [n+2]	STACK_OP [n]	STACK_VAL [n]
NOP	0000000	PC[n]+1	stable	nop	
ADD D ACC	01aaddd	PC[n]+1	ACC(a)[n+1] + D(d)[n+1]	nop	
JSR addr	10bbbbb	PC[n]+1	stable	nop	
↳ *cycle 2*	ccccccc	cb	stable	push	PC[n]

Informal Specification

- Functionality specified **informally** if:
  - complex and used by many instructions
  - would bloat tabular representation
  - e.g. stacks, queues, interrupt controllers, etc.
- Off-loading to informal specification allows to:
  - improve clarity and reduce redundancy of tabular representation
  - improve verification efficiency

Agenda

- Introduction
- Completeness Concepts
- Verification Process
- Semi-Formal Specification
- Implementation and Results
- Conclusion

Automatic Property Generation

- Straightforward translation:
  - input: tabular specification (standard spreadsheet)
  - output: properties and constraints (SystemVerilog/TiDAL)
- Implementation: Java Emitter Templates (JET)
- Followed rules:
  - keep the translator simple (3400 LOC)
  - minimum design knowledge in the translator
  - one property per multi-cycle instruction (simplifies completeness check)

Property Example

Mnemonic	Instruction [n]	INT_REQ [n]	PC [n+1]	ACC [n+2]
ADD D ACC	01aaddd	0	PC[n]+1	ACC(a)[n+1] + D(d)[n+1]
interrupt	*overlay row*	1	INT_NO[n]*2	

```
property prop_ADD_D_ACC;
  reg [p_instr_width-1:0] opcode;                          // variable for the opcode
    t ##0 Instruction ==? 7'b01XXXX and                 // TRIGGERS
    ...
    t ##0 set_freeze(opcode, Instruction) and             // store opcode
implies                                                 // COMMITMENTS
    t ##1 PC == ($past(INT_REQ) ?   $past(INT_NO)*2
                                  : $past(PC) + 1'b1) and
    t ##2 ACC[getField_a(opcode)] ==   $past(ACC[getField_a(opcode)])
                                     + $past(  D[getField_d(opcode)]) and
    t ##2 (getField_a(opcode) == 'd0) || $stable(ACC[0]) and // ACC[0] modified or stable
    t ##2 (getField_a(opcode) == 'd1) || $stable(ACC[1]) and // ACC[1] modified or stable
    ...
    t ##1 right_hook;
endproperty
```

Verification Summary

- Specification includes:
  - tabular: about 260 rows & 60 columns
  - informal: address generation, I/O and interrupt control, stacks, queues, MAC
- Implementation effort:
  - 7 person months in total (incl. debug of 5 DSPs)
  - easy management
- Achieved 100% functional coverage for tabular spec.
  - found about 50 spec. issues (mostly gaps) and 10 corner case bugs
  - typical compute time: 5 h
- Just a few hour's work to apply to a new DSP!
  - with added / removed / modified instructions
  - with changed data path width, amount of data paths and registers, etc.

Agenda

- Introduction
- Completeness Concepts
- Verification Process
- Semi-Formal Specification
- Implementation and Results
- Conclusion

Conclusion

- Good trade-off between spec. precision and readability
  - 100% functional coverage for tabular spec.
  - spec. development requires little/no formal background
- Improved maintainability and reusability
  - late specification/implementation changes not critical
- Weaker dependency on RTL
  - early verification start
  - early completeness check

Questions

Literature

- Bormann, J., Beyer, S., Maggiore, A., Siegel, M., Skalberg, S., Blackmore, T., and Bruno, F. 2007. **Complete formal verification of TriCore2 and other processors**. In Design and Verification Conference (DVCon'07).
- Bormann, J. and Busch, H. 2005. **Method for determining the quality of a set of properties**. European Patent Application, publication number EP1764715.
- Claessen, K. 2007. **A coverage analysis for safety property lists**. In IEEE Proc. International Conference on Formal Methods in Computer-Aided Design (FMCAD'07), pp. 139-145.
- Kühne, U., Beyer, S., Bormann, J., and Barstow, J. 2010. **Automated formal verification of processors based on architectural models**. In IEEE Proc. Formal Methods in Computer-Aided Design (FMCAD'10), pp. 129-136.
- Loitz, S., Wedler, M., Stoffel, D., Brehm, C., Kunz, W., and Wehn, N. 2010. **Complete verification of weakly programmable IPs against their operational ISA model**. In IEEE Proc. Forum on Specification & Design Languages (FDL'10).