Validation of Timing Constraints on RTL

Reducing Risk and Effort on Gate-Level

Peter Limmer, Dirk Moeller,
Marcus Mueller, Clemens Roettgermann

NXP Semiconductors, Munich, Germany

accellera SYSTEMS INITIATIVE

NXP

2016 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Outline

- **Introduction**
  - Motivational example
- Specification-centered approach
  - Capturing the specification of timing constraints
- Verification of STA deliverables on RTL
  - Clocks, multi-cycle exceptions
- Conclusion & Summary

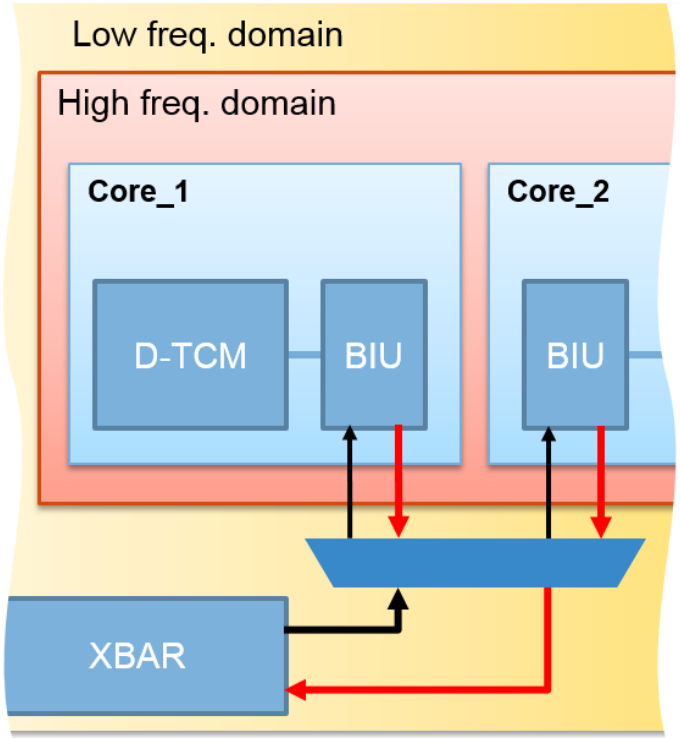
Introduction

- SoC designs for MCU devices w/ 2-6 Mio. cells (gate equiv.)
- Verification gap
  - Gate Level Simulation (GLS) – inherently late in design cycle, computationally expensive, hard to debug
  - GLS covers only 10% of RTL regression
- Shift of responsibilities – design and verification
  - earlier and on higher abstraction levels – the right time for everything
  - the appropriate approach for the right problem

Motivational example

- Processor core w/ tightly-coupled memory and bus interface
- Read data path signals
  - STA: timing violated → Multi-cycle exception applied
  - subsequently - GLS failed!
- BIU cycle adjust logic failed
  - RTL not verified for multi-cycle behavior

➔ Expensive late redesign cycle



- Lessons learned:
  - Multi-cycle spec. and review late in GL process is **too late**
  - Awareness low, info capture and exchange method weak

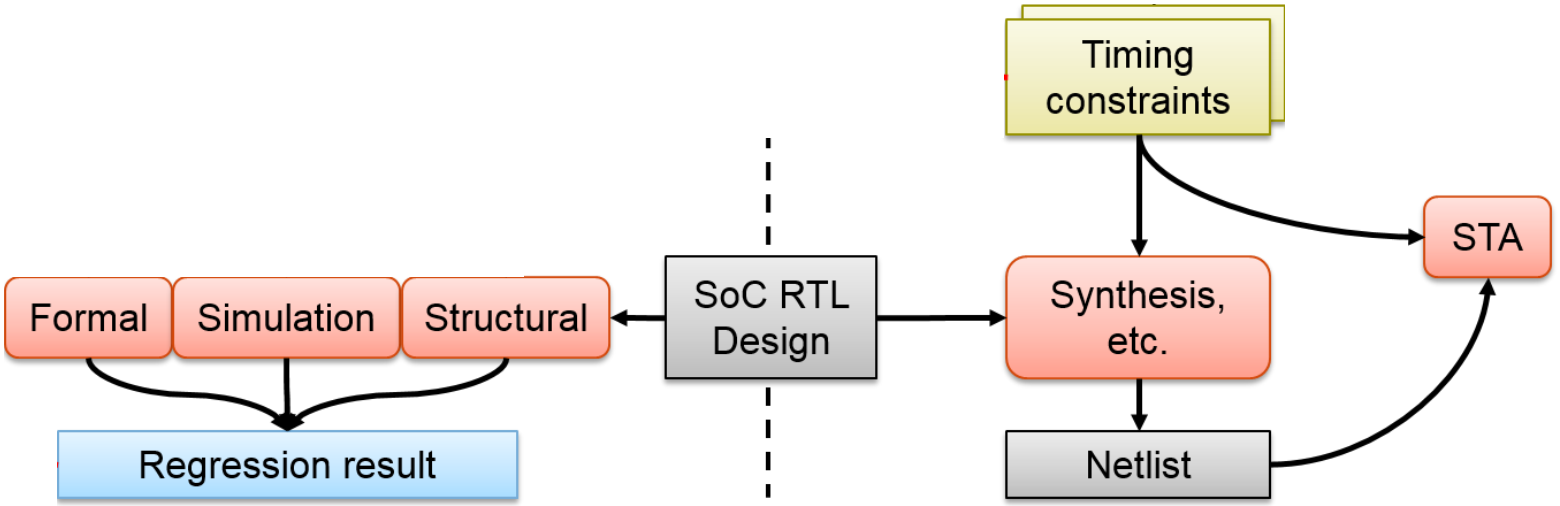
Outline

- Introduction
  - Motivational example
- **Specification-centered approach**
  - Capturing the specification of timing constraints
- Verification of STA deliverables on RTL
  - Clocks, multi-cycle exceptions
- Conclusion & Summary

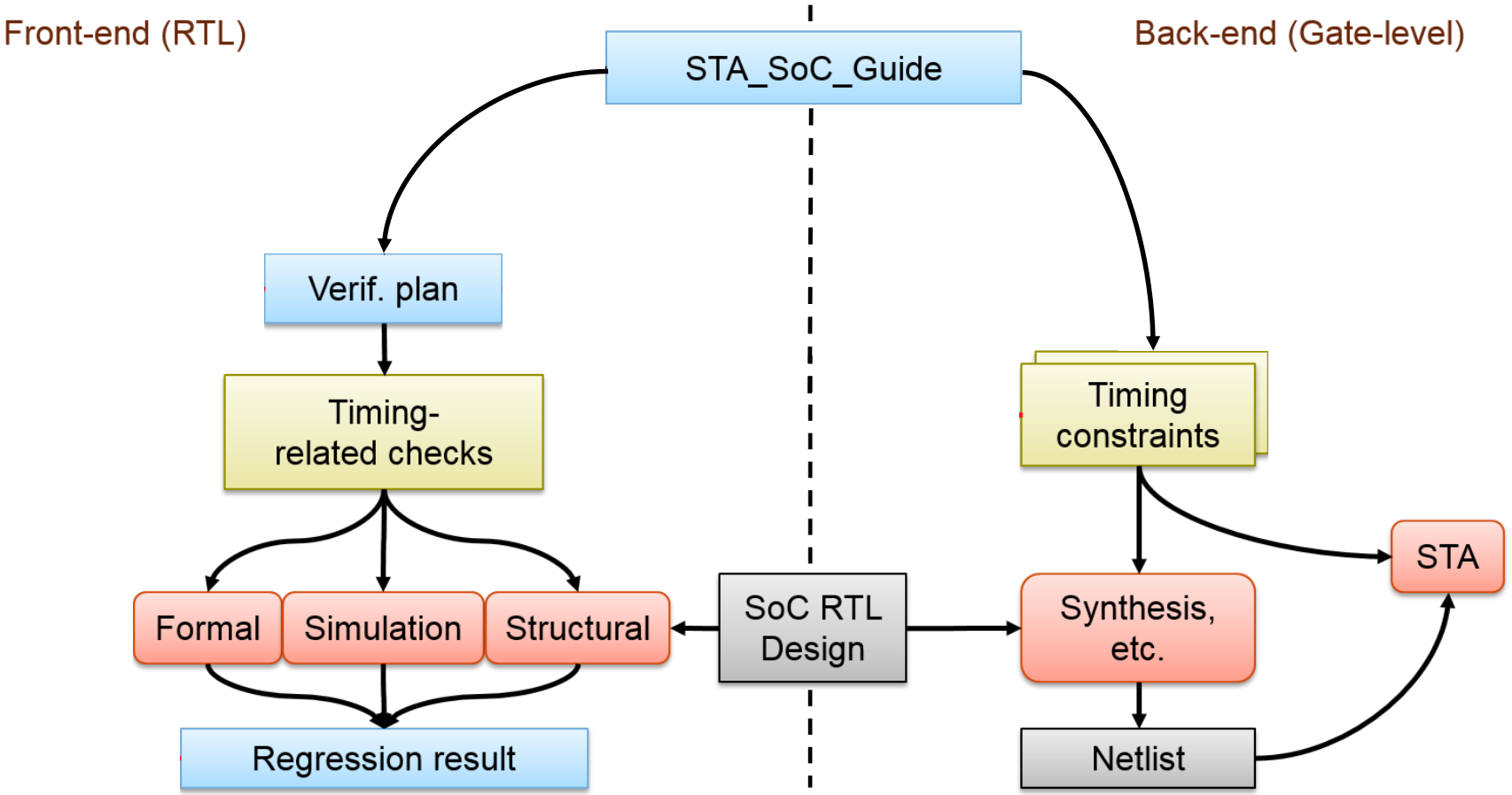
Specification-centered approach

Front-end (RTL)

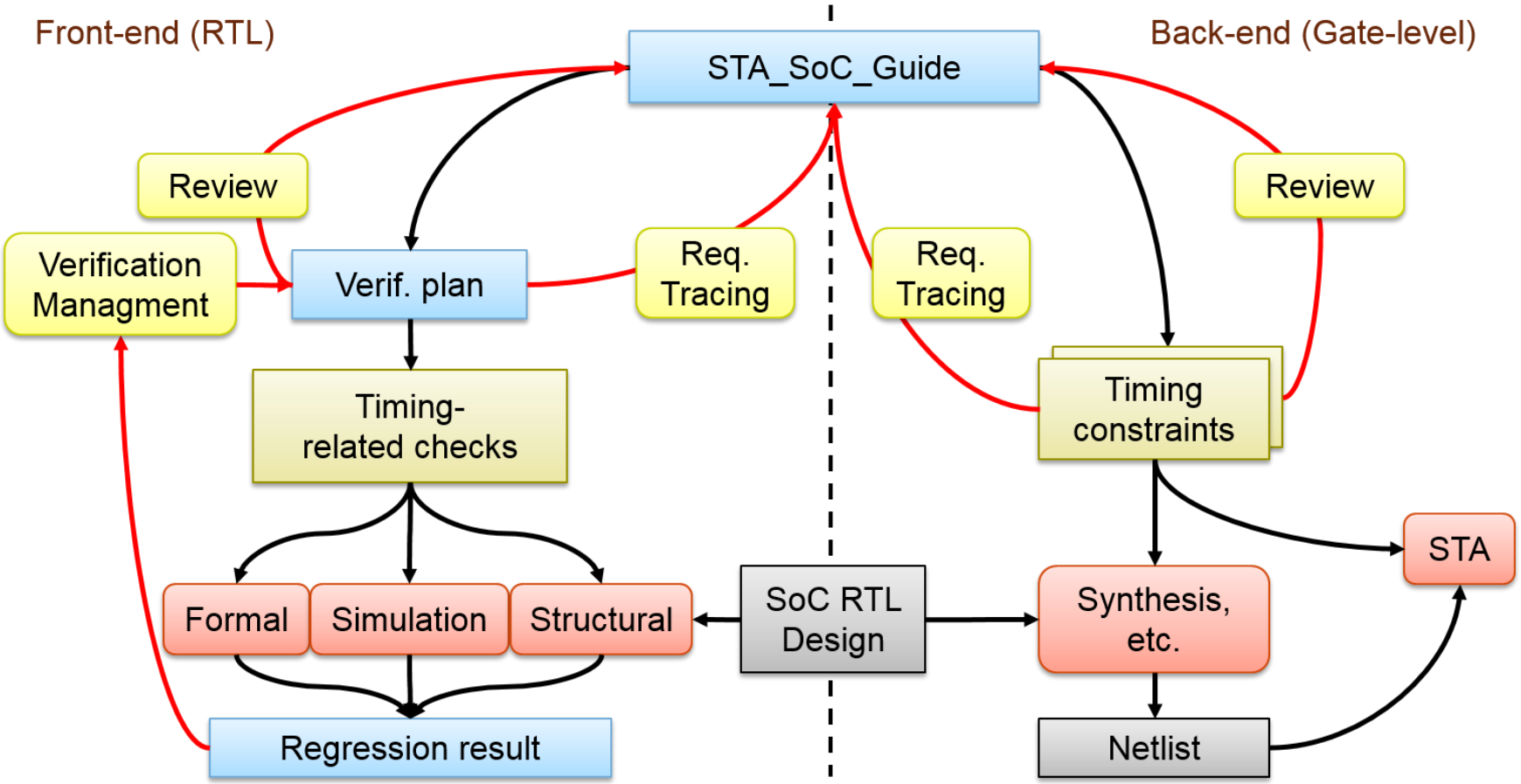
Back-end (Gate-level)



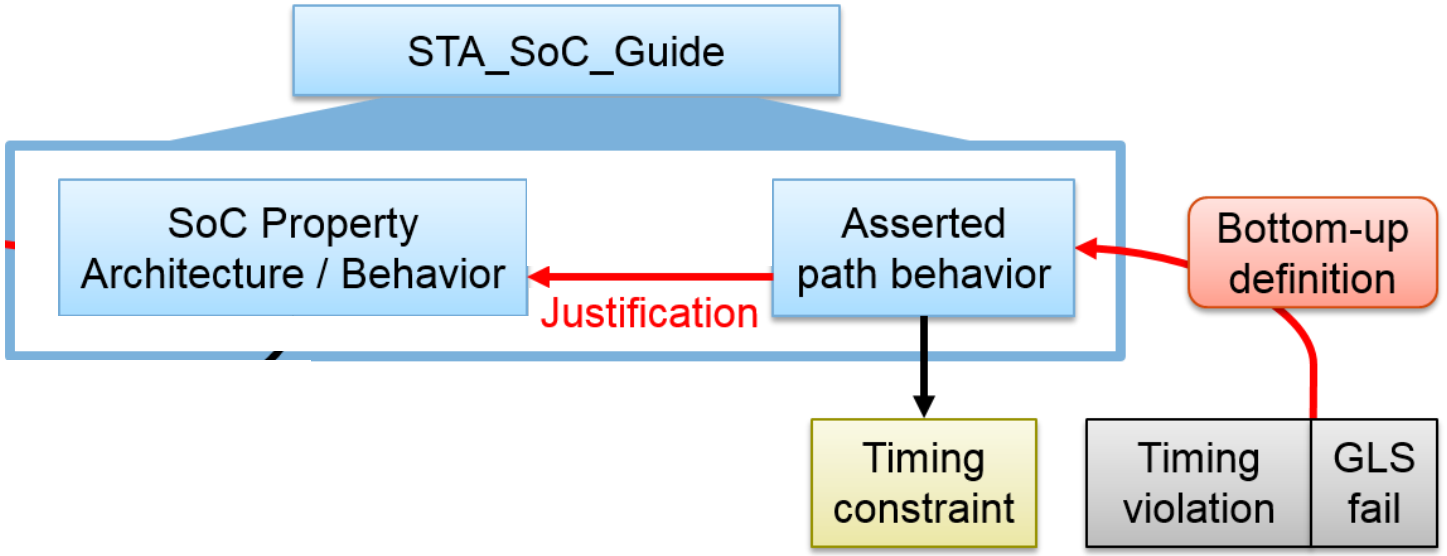
Specification-centered approach



Specification-centered approach



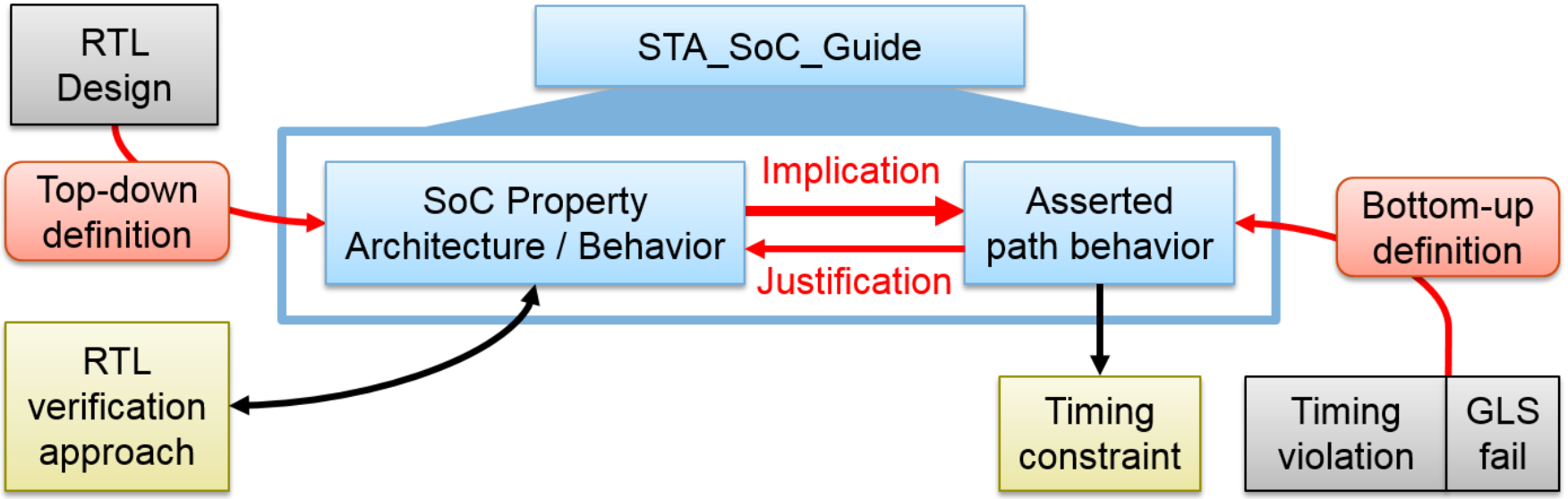
Capturing the specification



- Bottom-up - Constraining timing violations on GL
- Justification based on SoC design property on RTL + common Documentation

➔ Reactively resolving single issues

Capturing the specification



- Top-down - Deriving possible timing relaxations on RTL
- Implication from ***verified*** SoC properties/behavior

➔ a-priori definition of relaxation potential

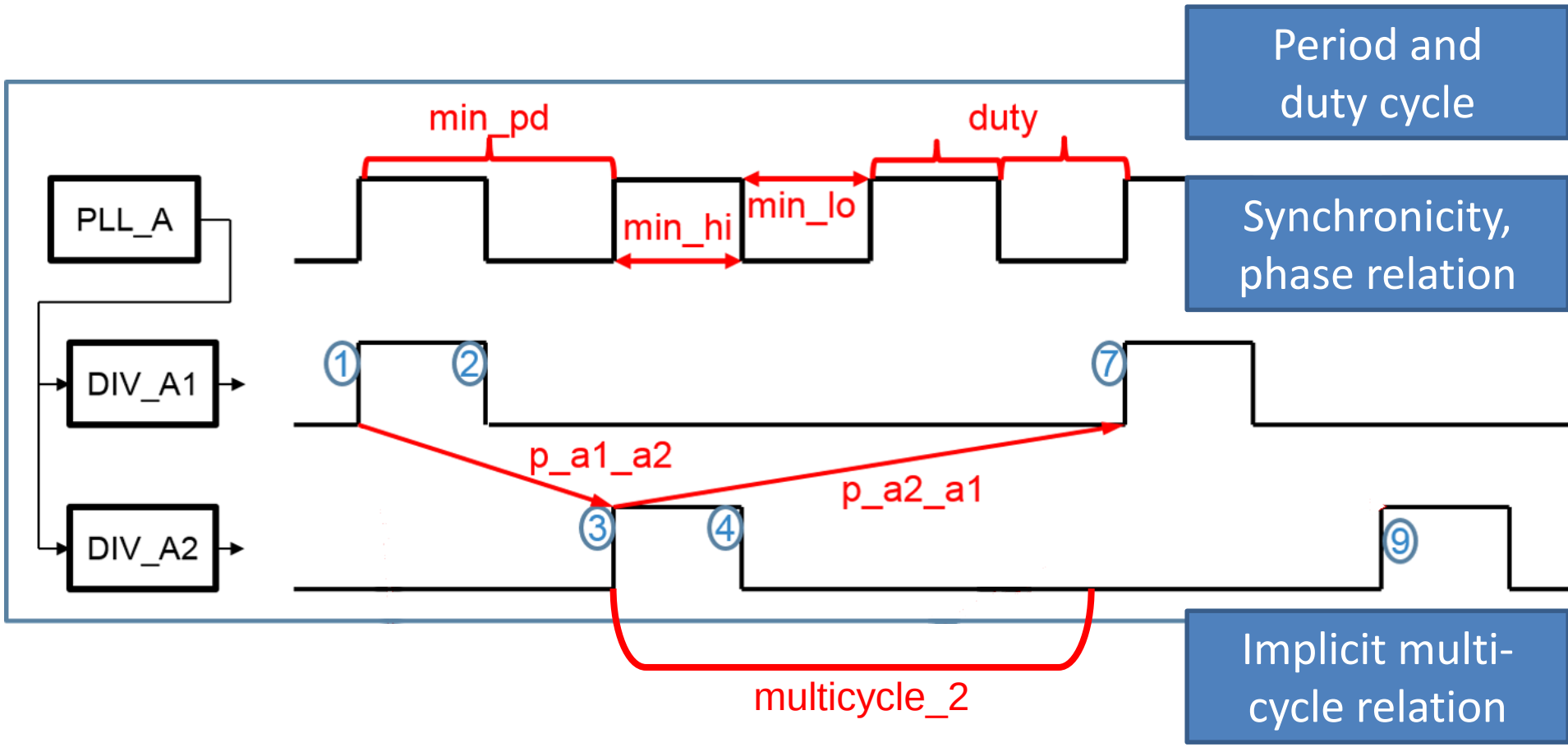
Verification intention

- NOT about verifying the timing on RTL …
- Verification of ***chain of causality*** on RTL
  - Verify underlying SoC property
  - Verify implication of a certain timing behavior
  - Verify the correct functionality under these conditions
- Allow formulation of a timing constraint/exception on GL based on verified data
  - Consistent clock definition
  - Correct multi-cycle path specification

Outline

- Introduction
  - Motivational example
- Specification-centered approach
  - Capturing the specification of timing constraints
- **Verification of STA deliverables on RTL**
  - Clocks, multi-cycle exceptions
- Conclusion & Summary

Clock definition



Clock verification

- **Intra-clock properties**
  - period, duty cycle
- **Inter-clock properties**
  - synchronicity, phase relation
- Implicit multi-cycles
  - between phase-shifted derived (synch.) clocks

→ Verify via configured clock checkers in RTL regression

➔ Consistent clock constraints provided to STA

Multi-cycle path specification

- Operation-mode dependent exceptions

- Example - **Device configuration phase**
  - System running on slow clk … to be switched to fast clk
  - Config. signals only change in this phase, stable otherwise
  - Significant relaxation possible - **multi-cycle *fast_clk/slow_clk***

→ apply multi-cycle exceptions, only when verification holds:

1. $M_{slow} \rightarrow f_{clk} \leq f_{slow}$
2. $\forall s \in DeviceConfigSignals: \quad s = const \quad \leftarrow \quad M_{fast}$ **or**
   $\exists s \in DeviceConfigSignals: \quad s \neq const \quad \rightarrow \quad M_{slow}$

➔ Verify with assertions as sim. monitors or prove formally

Multi-cycle path specification

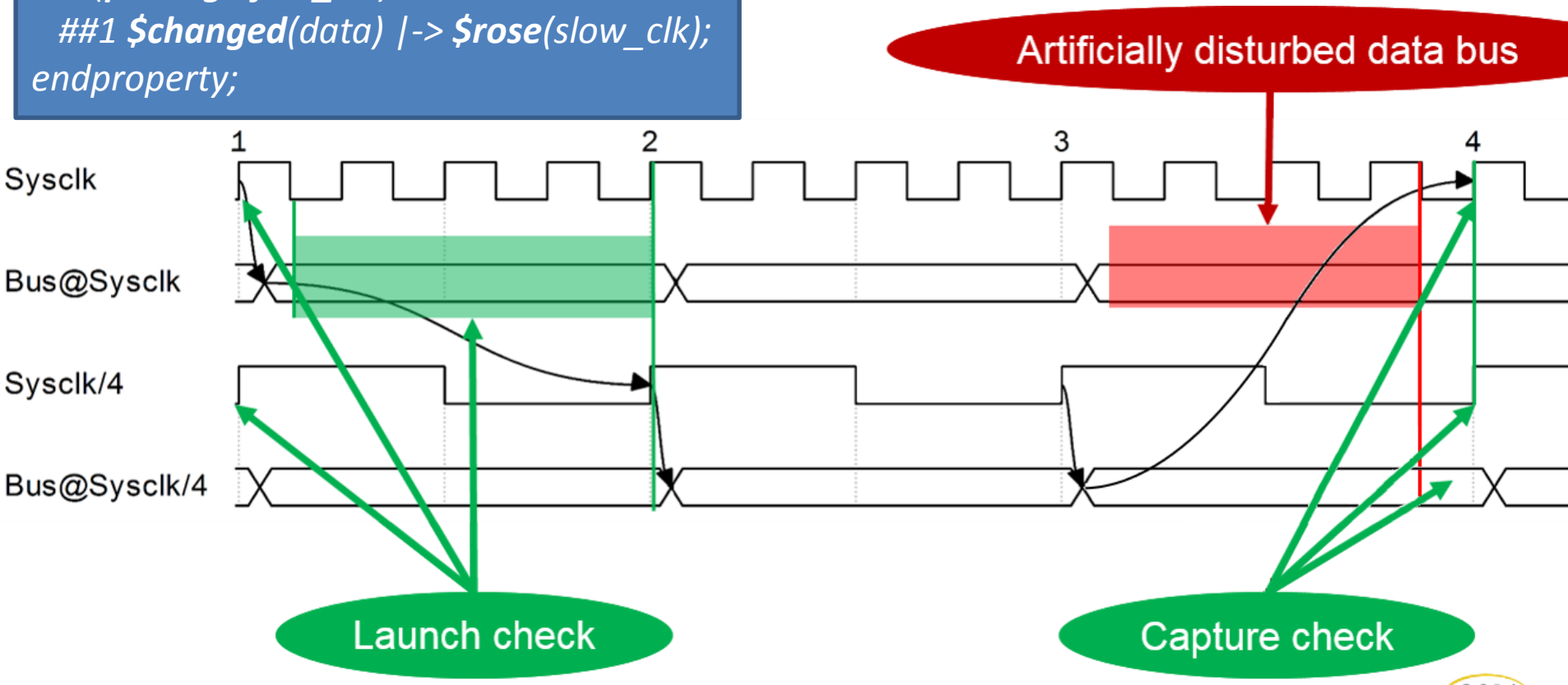
- Protocol-driven exceptions in component interaction

- Example - **Fast-to-Slow component communication**
  - Possible relaxation of data path: **multi-cycle *fast_clk/slow_clk***

→ apply multi-cycle exceptions, only when verification holds:
  - Clock checks – synchronicity, etc.
  - **Launch check** - *check that data only change with rising edge of slow clock -> stable through multi-cycle*
  - **Capture check** - *check that data captured with a fast clock, but only when aligned with the slow clock edges*

Verification of multi-cycle comm.

```
property p_data_toggle;      // SVA
  @(posedge fast_clk)
  ##1 $changed(data) |-> $rose(slow_clk);
endproperty;
```



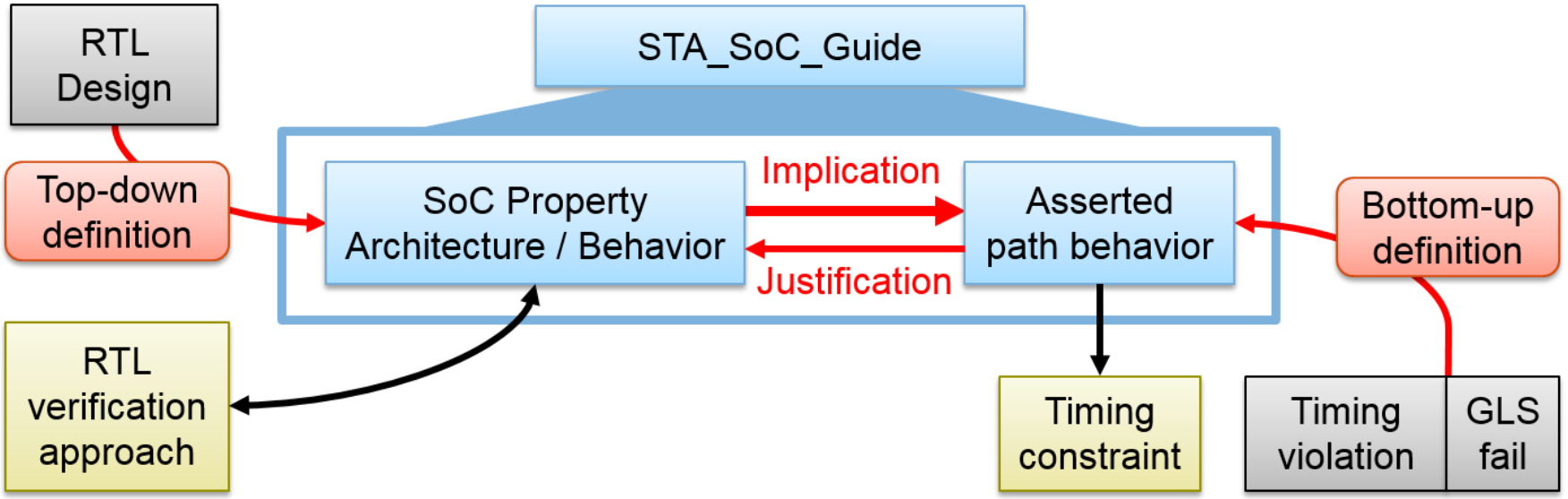
Outline

- Introduction
  - Motivational example
- Specification-centered approach
  - Capturing the specification of timing constraints
- Verification of STA deliverables on RTL
  - Clocks, multi-cycle exceptions
- **Conclusion & Summary**

Conclusion

- Established methodical work flow
  - Central specification to bridge team and process boundaries
  → raising awareness, increasing reliability
  - Top-down definition of SoC properties on RTL with influence on GL timing → dedicated verification
- **Prevention:**
  - Increase responsibilities of specification and RTL verification
  - Avoid failures, that are difficult to deal with, when detected only at GL
- Reducing the risk of costly debug/redesign cycles

Thank you for your attention …



Questions?