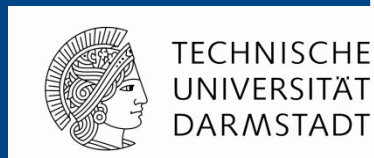
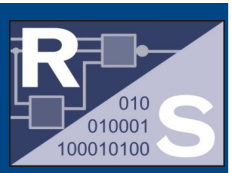
Combining Runtime Verification and Execution Time Estimation





Boris Dreyer
dreyer@rs.tu-darmstadt.de

Prof. Dr.-Ing. Christian Hochberger
Computer Systems Group
Department of Electrical Engineering and Information Technology
Technische Universität Darmstadt, Germany

This work was funded within the project CONIRAS by the German Federal Ministry for Education and Research with the funding ID 01IS13029. The responsibility for the content remains with the authors.

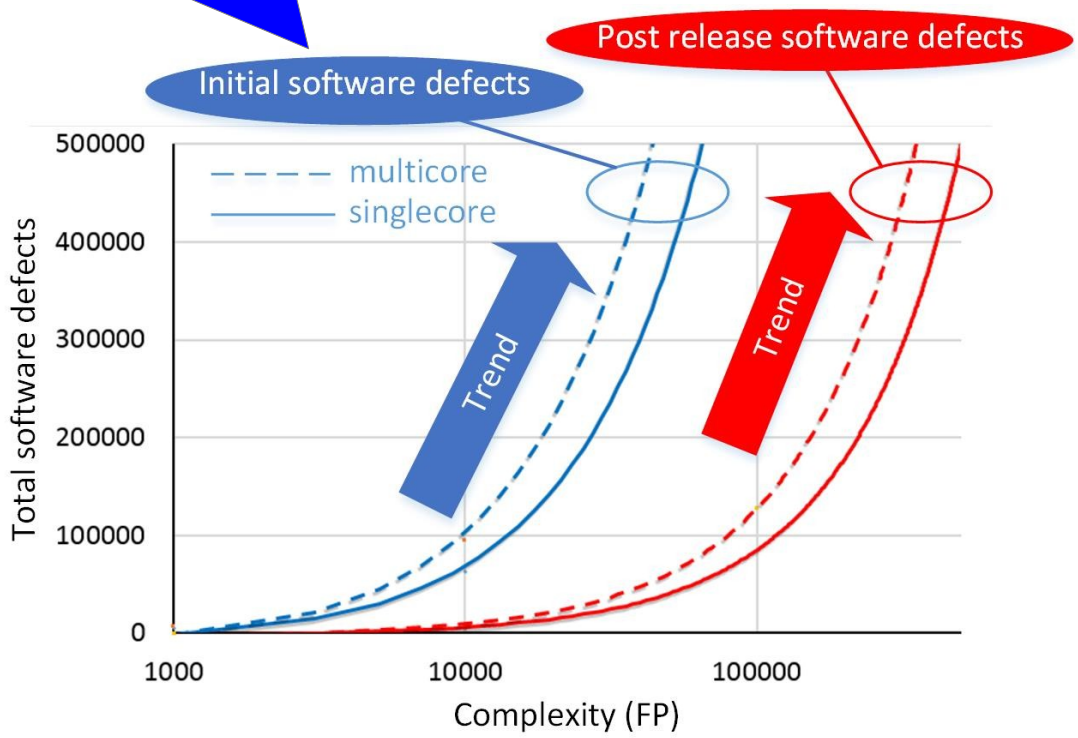
Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

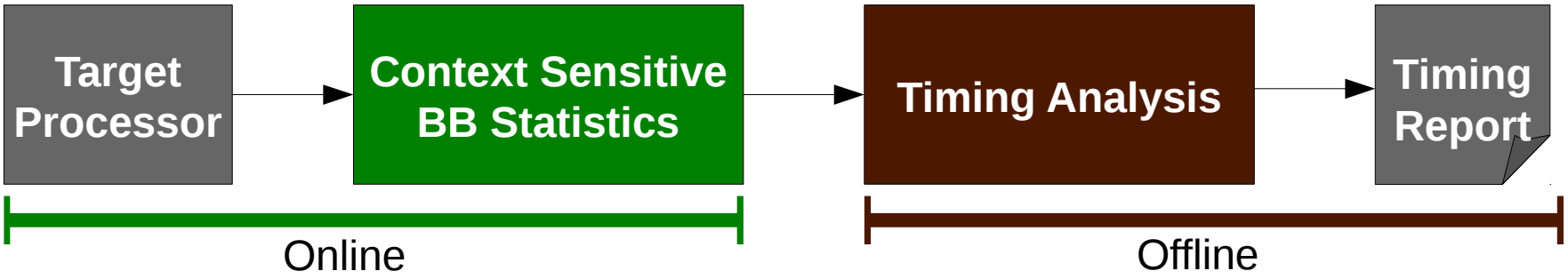
- **Motivation**
- **Measurement-based execution time analysis**
  - Context sensitive BB statistics
  - Architecture
- **Continuous timing validation**
  - Constraints monitors
  - Architecture
- **Conclusion**

Motivation – Timing Analysis

Solution: Measurement-based Execution Time Analysis



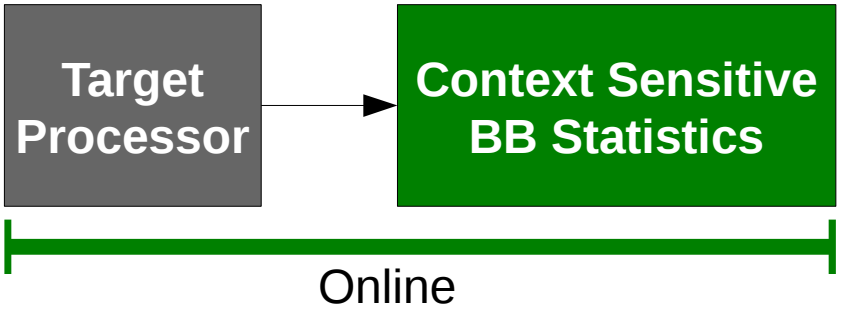
Measurement-based Execution Time Analysis



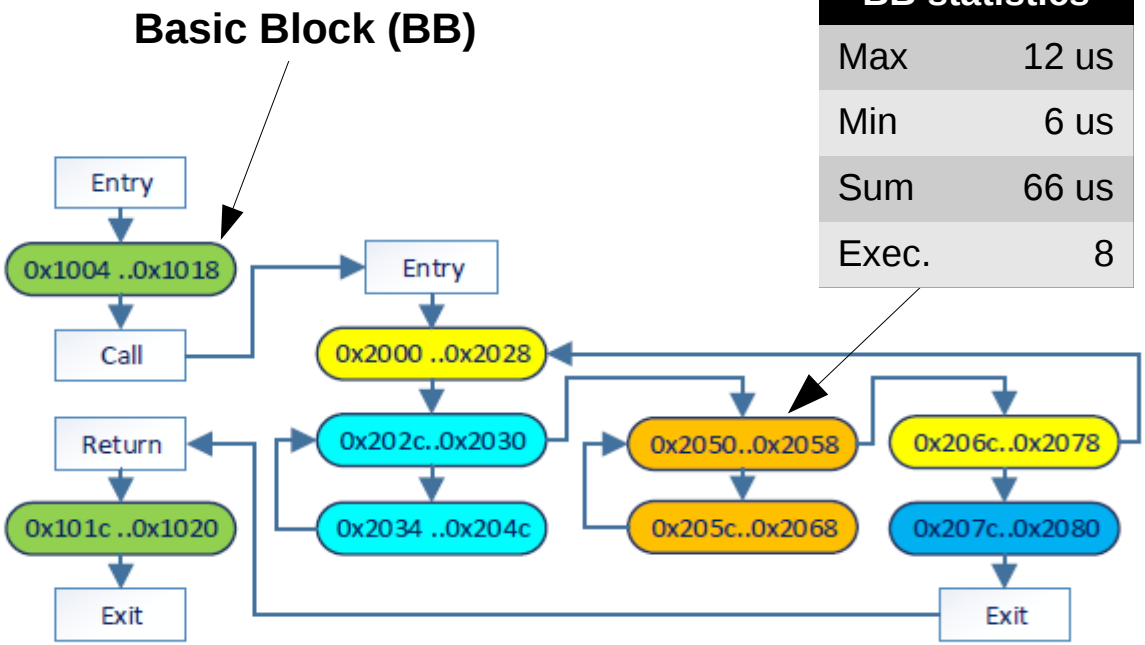
- **Online timing aggregation**
  - Storage depends on program size
  - Storage is independent of observation time
  - Arbitrary long observation time based on fine granular measurements

- **Offline timing analysis**
  - Precise timing analysis for developers
  - Timing constraints generation for *continuous timing validation*

BB Statistics

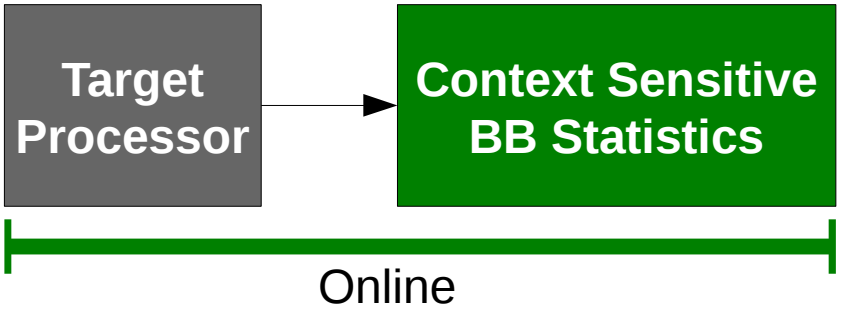



```
void f (…)
{
  g(…);
}
void g (…)
{
  do {                  // L1
    …;
    for (…;…;…)         // L2
      …;
    for (…;…;…)         // L3
      …;
  } while (…);
}
```



BB statistics	
Max	12 us
Min	6 us
Sum	66 us
Exec.	8

Context Sensitive BB Statistics

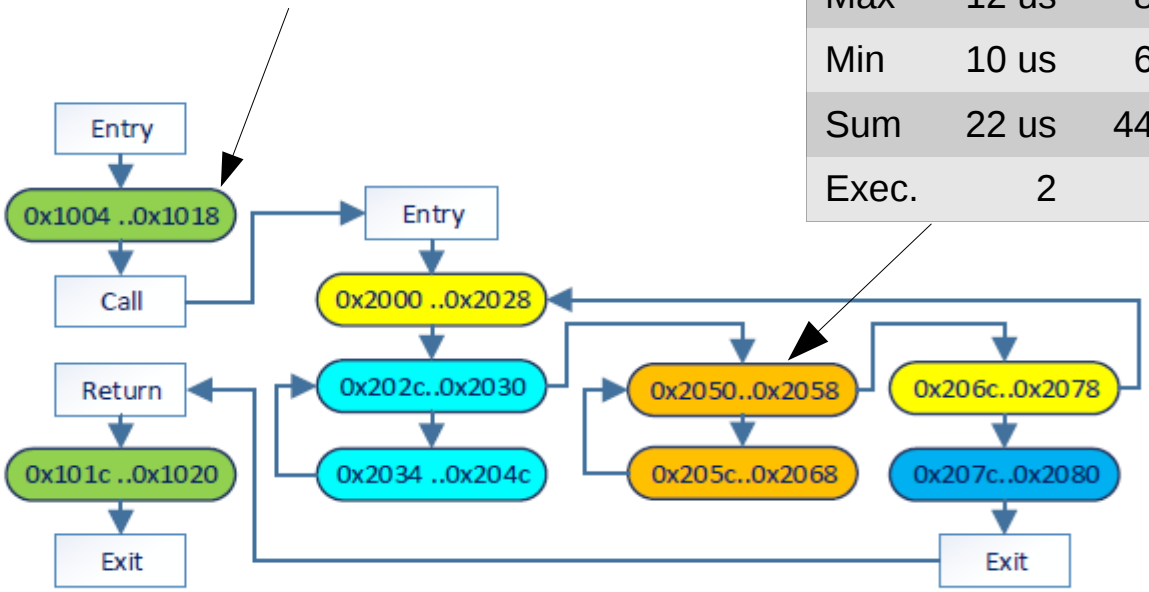


Context sensitive BB statistics

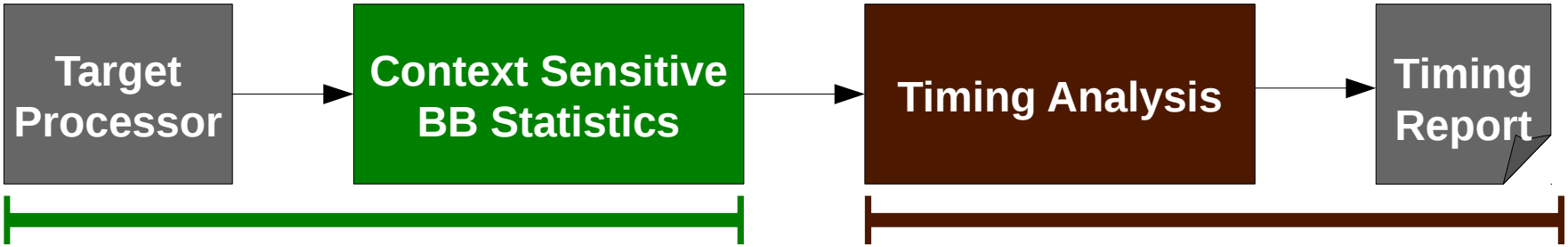
	Iterations First	Further
Max	12 us	8 us
Min	10 us	6 us
Sum	22 us	44 us
Exec.	2	6

Basic Block (BB)

```
void f (…)
{
  g(…);
}
void g (…)
{
  do {              // L1
    …;
    for (…;…;…)     // L2
      …;
    for (…;…;…)     // L3
      …;
  } while (…);
}
```

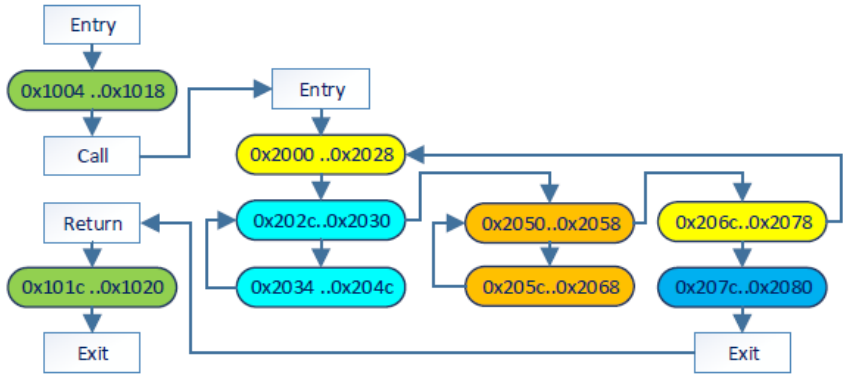


Timing Analysis



BB Address	Max First	Max Further
0x1004	8 us	0 us
0x2000	22 us	11 us
0x202c	16 us	8 us
0x2034	2 us	1 us
0x2050	12 us	8 us
0x205c	2 us	1 us
0x206c	8 us	4 us
0x207c	6 us	0 us
0x101c	6 us	0 us

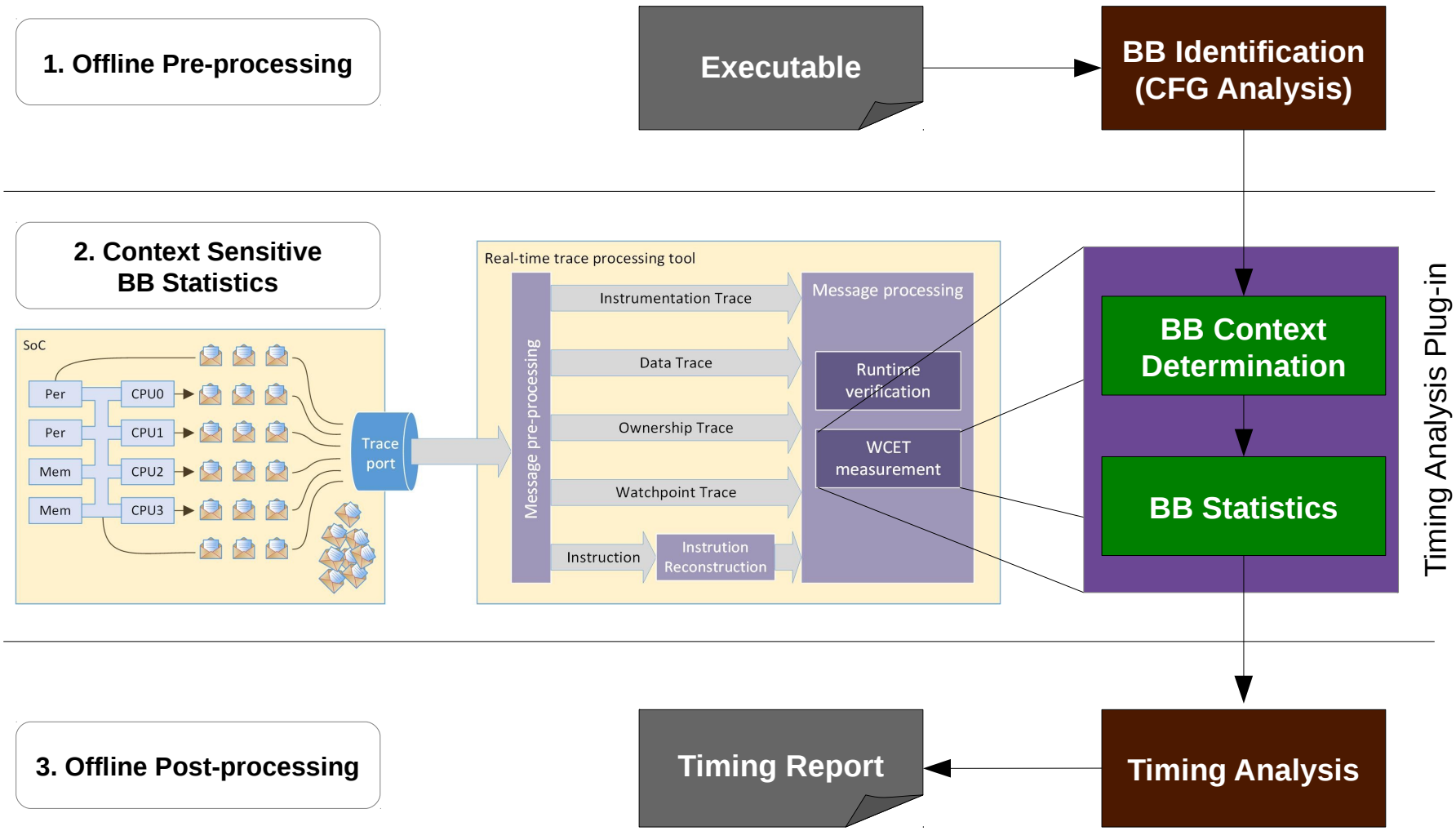
1. Annotate CFG



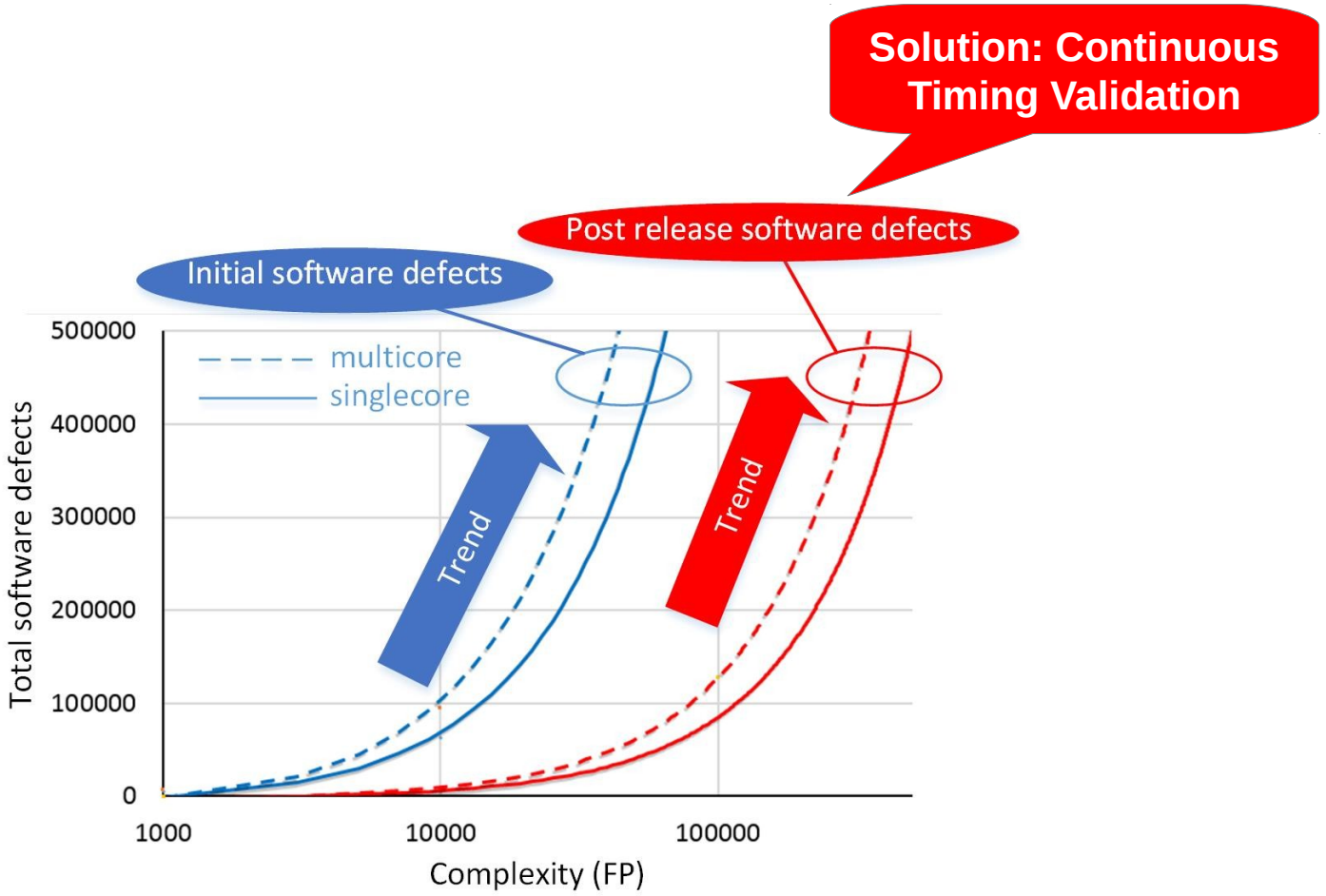
2. Find longest path (ILP based)

Overall execution time estimate
Our method: 191 us
Context insensitive: 208 us

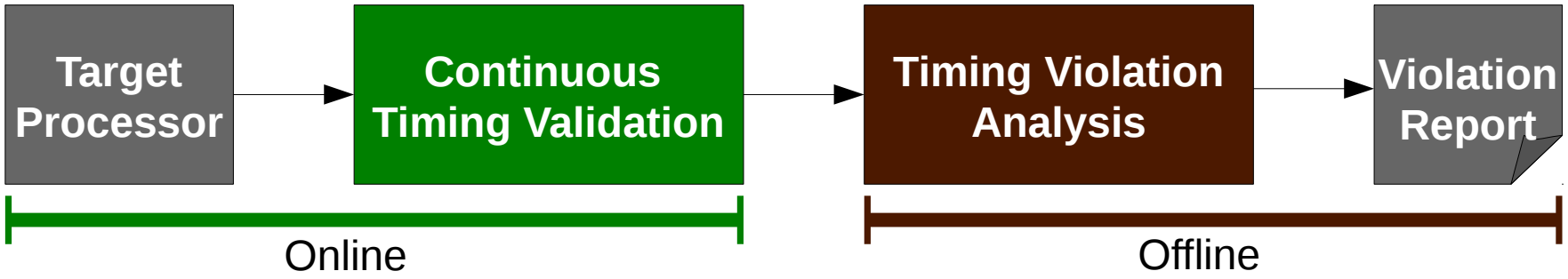
Timing Analysis Architecture



Motivation – Continuous Timing Validation

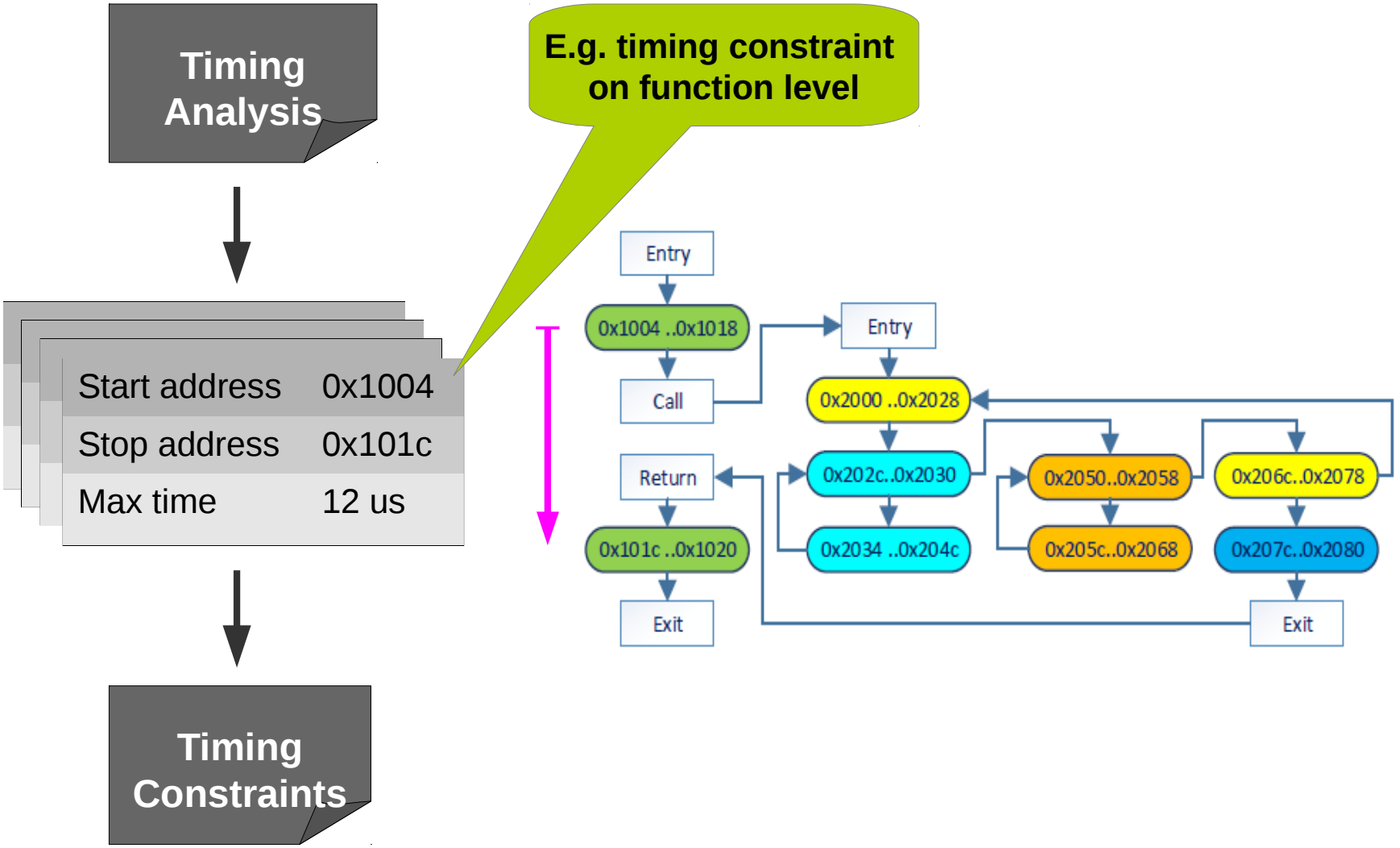


Continuous Timing Validation

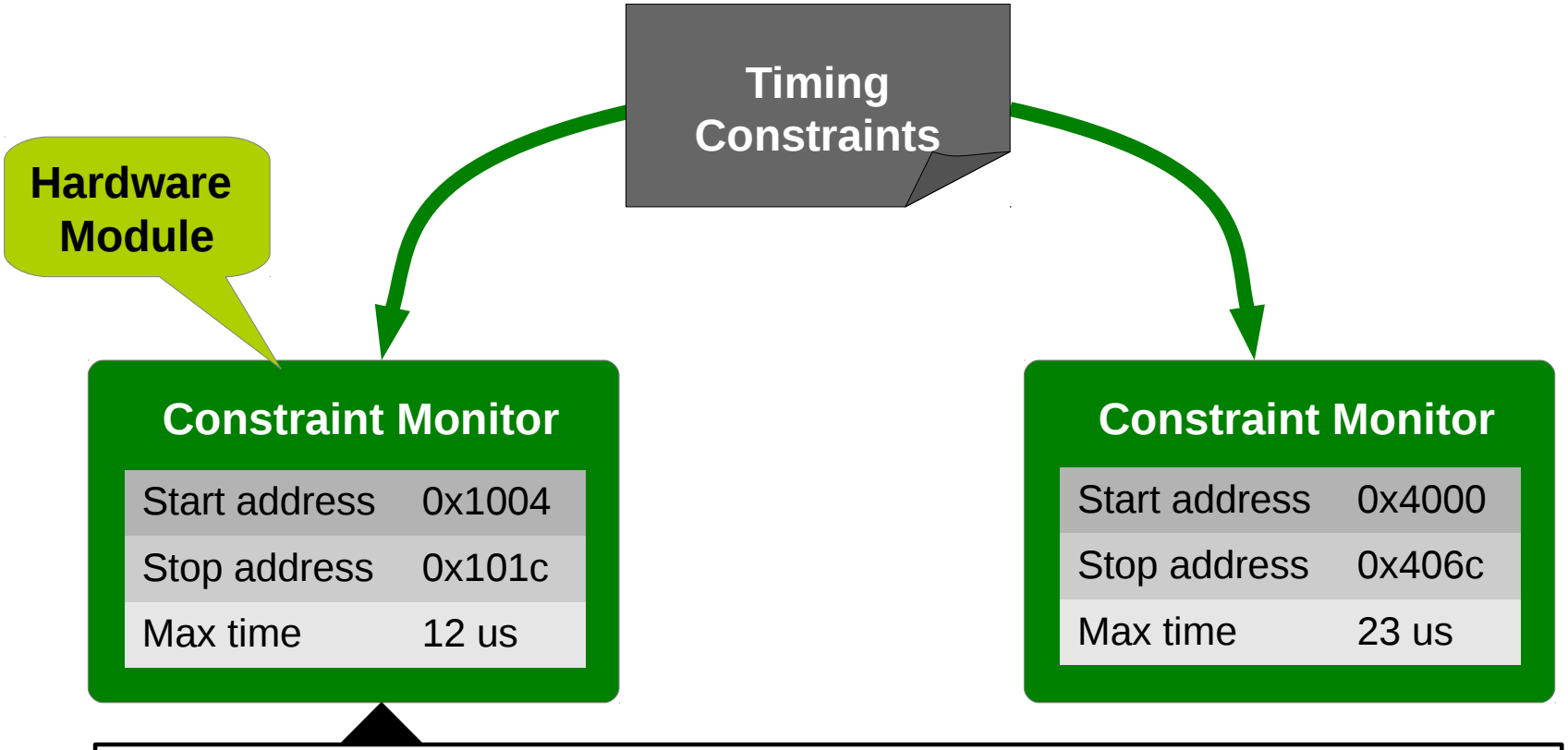


- **Online timing validation**
  - Storage depends on the timing violation error log size
  - Storage is independent of observation time
  - Arbitrary long timing validation
    - At arbitrary levels: BB level, function level, task level, ...
- **Offline violation analysis**
  - CFG annotation with recorded timing violations

Timing Constraints Extraction



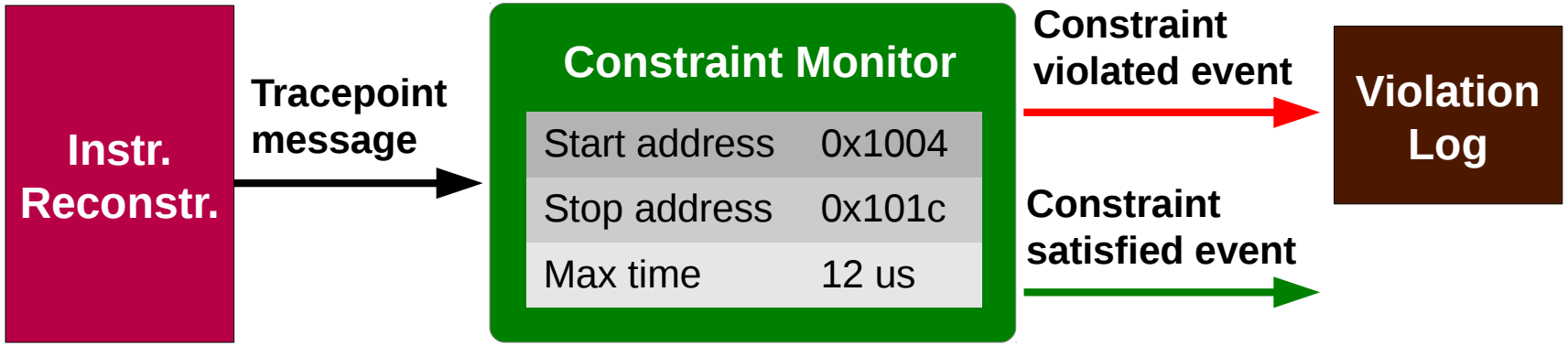
Timing Constraint Monitor



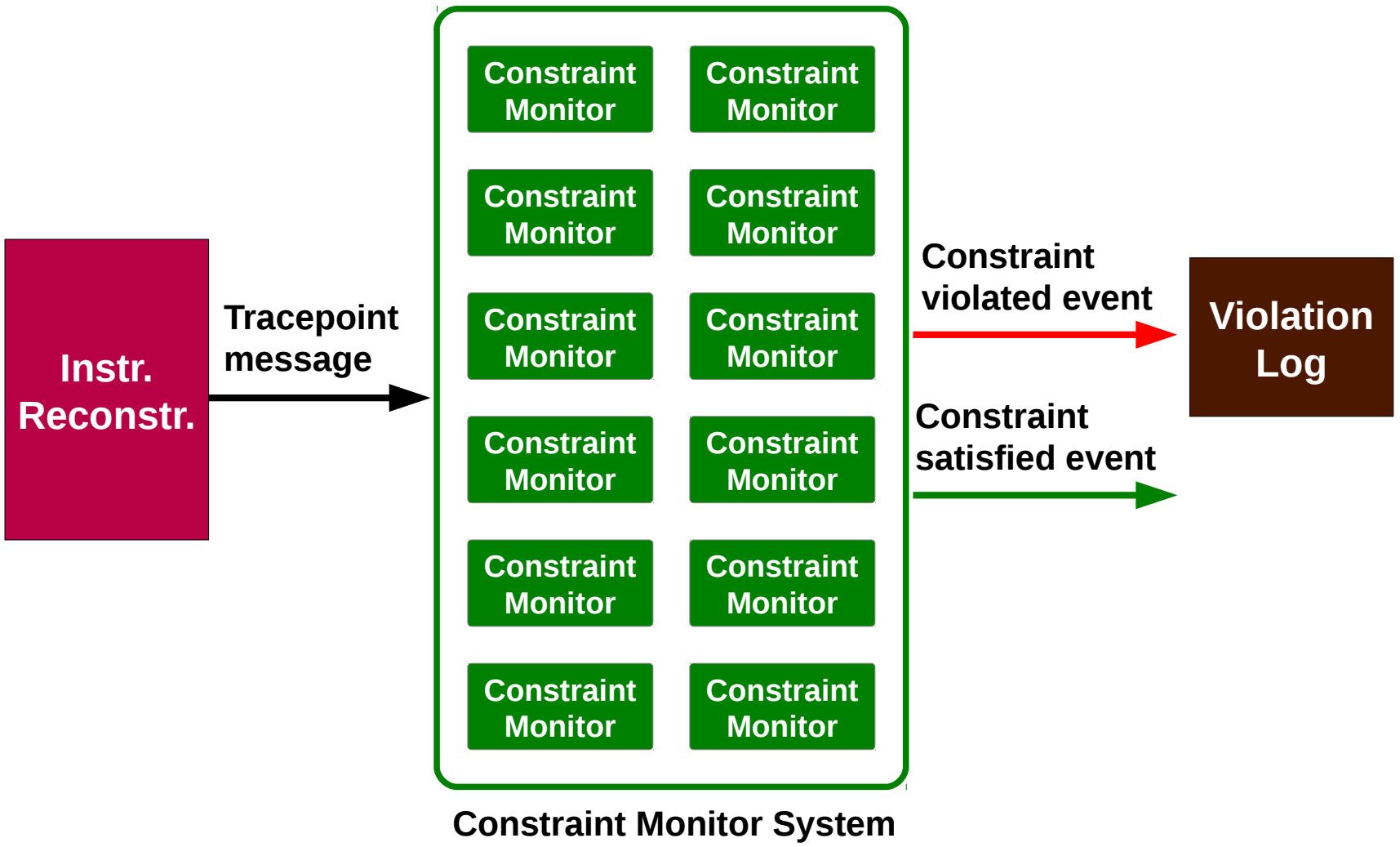
TeSSLa Specification

```
define block_start      := onExecuting 0x1004
define block_end        := onExecuting 0x101c
define timing_violation := on block_start if not (block_end within <=12μs)
```

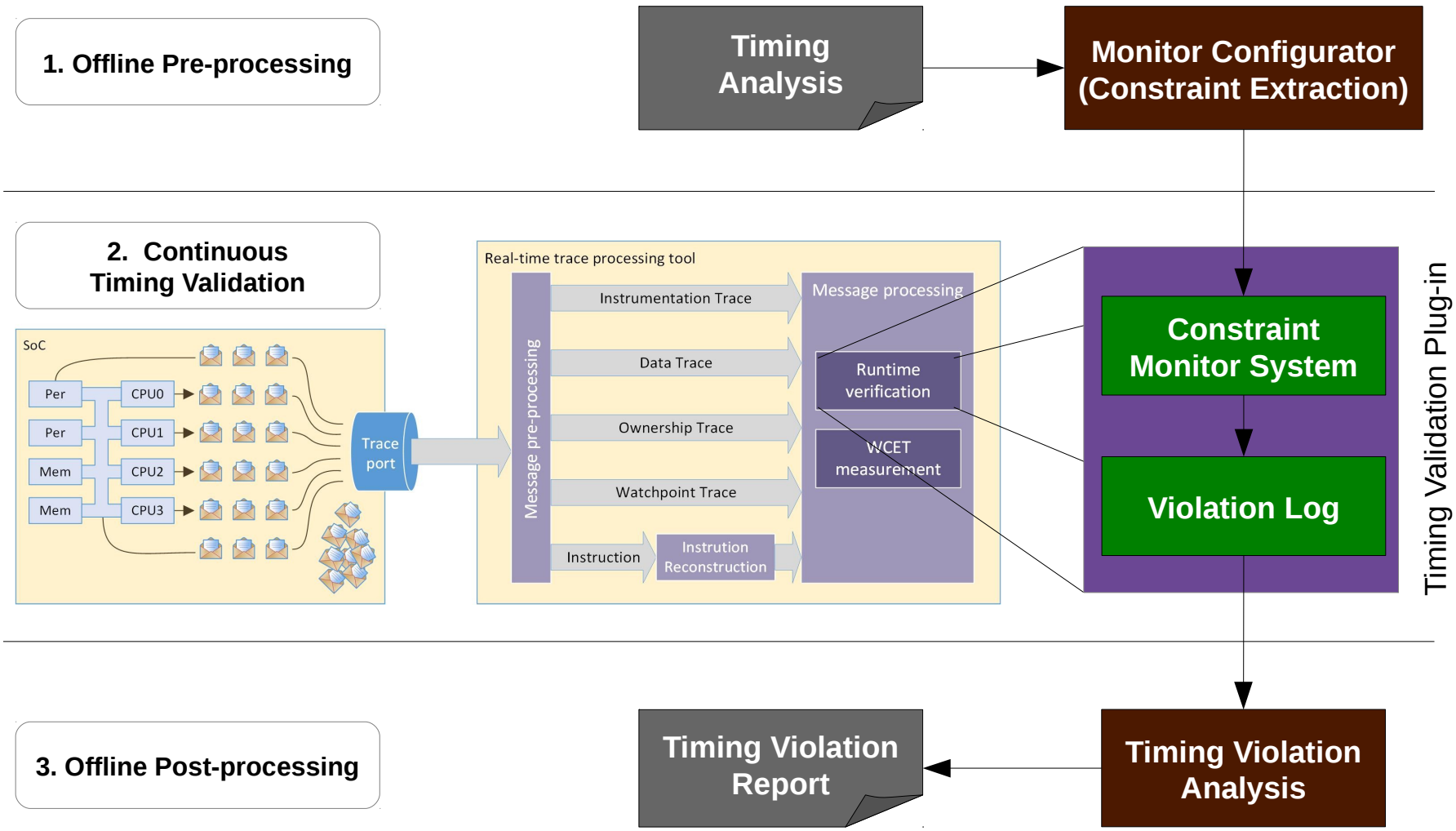
Timing Constraint Monitor



Timing Constraint Monitor System



Timing Validation Architecture



Conclusion

	Measurement-based Timing Analysis	Continuous Timing Validation
Task	Precise timing analysis	Timing constraints validation
Requirements	-	Initial timing analysis
Granularity	Basic Block level	Arbitrary
Application	Development	Field research Autonomous long-term observation

Thank you!

Acknowledgement

The author like to thank
Simon Wegener `wegener@absint.com`
for his valuable comments.