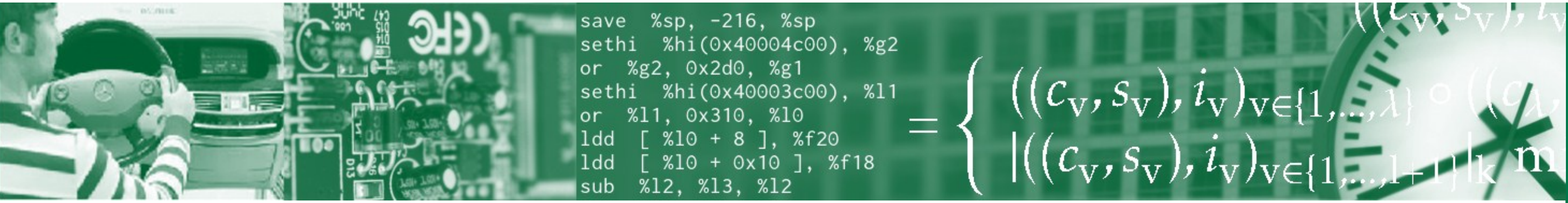


Trace-Based Context-Sensitive Timing Simulation Considering Execution Path Variations

Sebastian Ottlik, Jan Micha Borrmann,
Sadik Asbach, Alexander Viehl,
Wolfgang Rosenstiel, Oliver Bringmann

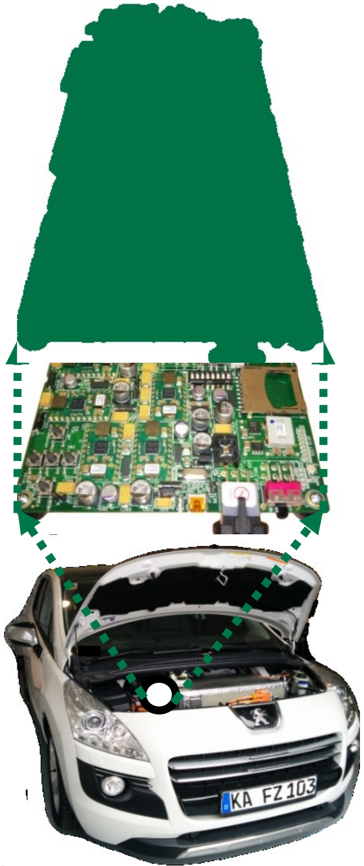
FZI Research Center for Information Technology
Karlsruhe, Germany
[ottlik|borrmann|asbach|viehl]@fzi.de

University of Tübingen
Tübingen, Germany
[rosenstiel|bringman]@informatik.uni-tuebingen.de

Motivation

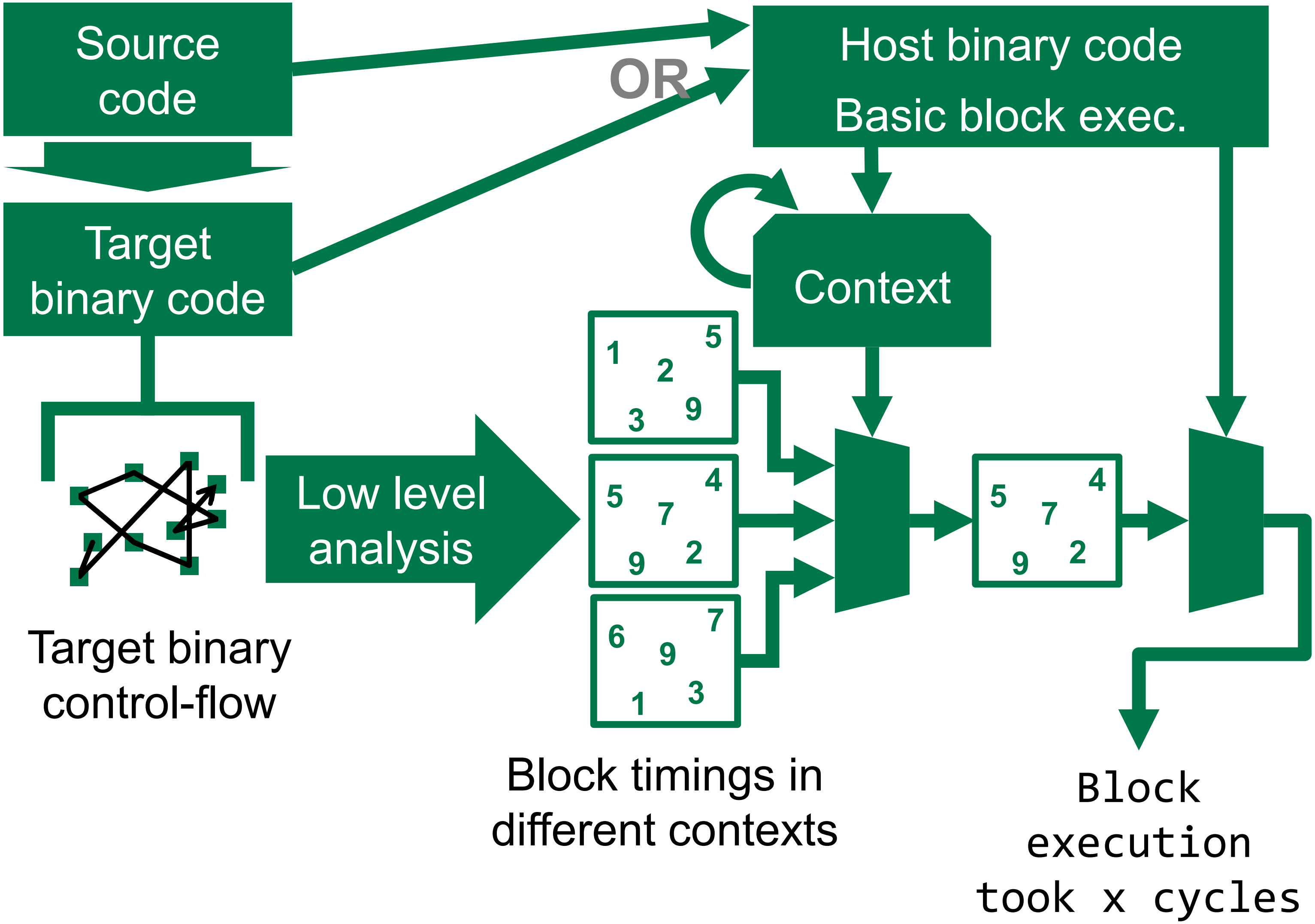
- Rising complexity of embedded functionality
  - Realized by complex, safety-critical SW
- Need SW performance analysis

- Advantages of performance simulation:
  - Good observability, repeatability, …
- Need accuracy/performance trade-off



Recent research demonstrates potential of context-sensitive simulation

Context-Sensitive Timing Simulation



Source code

Target binary code

OR

Host binary code

Basic block exec.

Context

Low level analysis

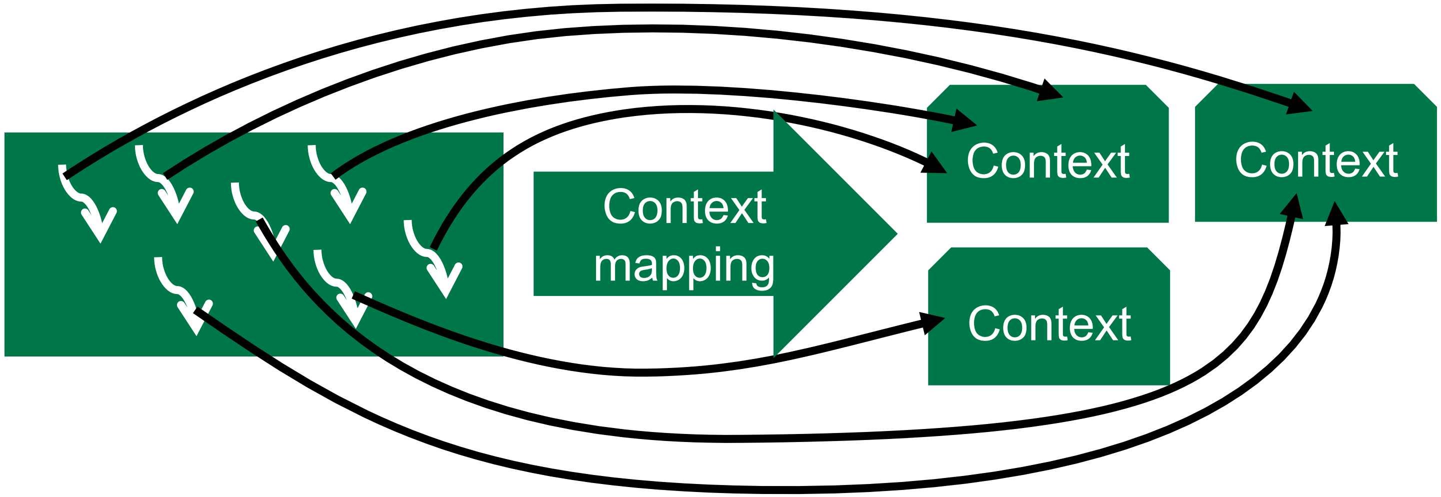
Target binary control-flow

Block timings in different contexts

Block execution took x cycles

What is a Context?

- Time to execute instruction block
  - Depends on uArch state
  - uArch state depends on preceding instructions
  - Use preceding control flow to approximate uArch State



- Necessary to make analysis feasible, but also useful to prevent excessive number of contexts

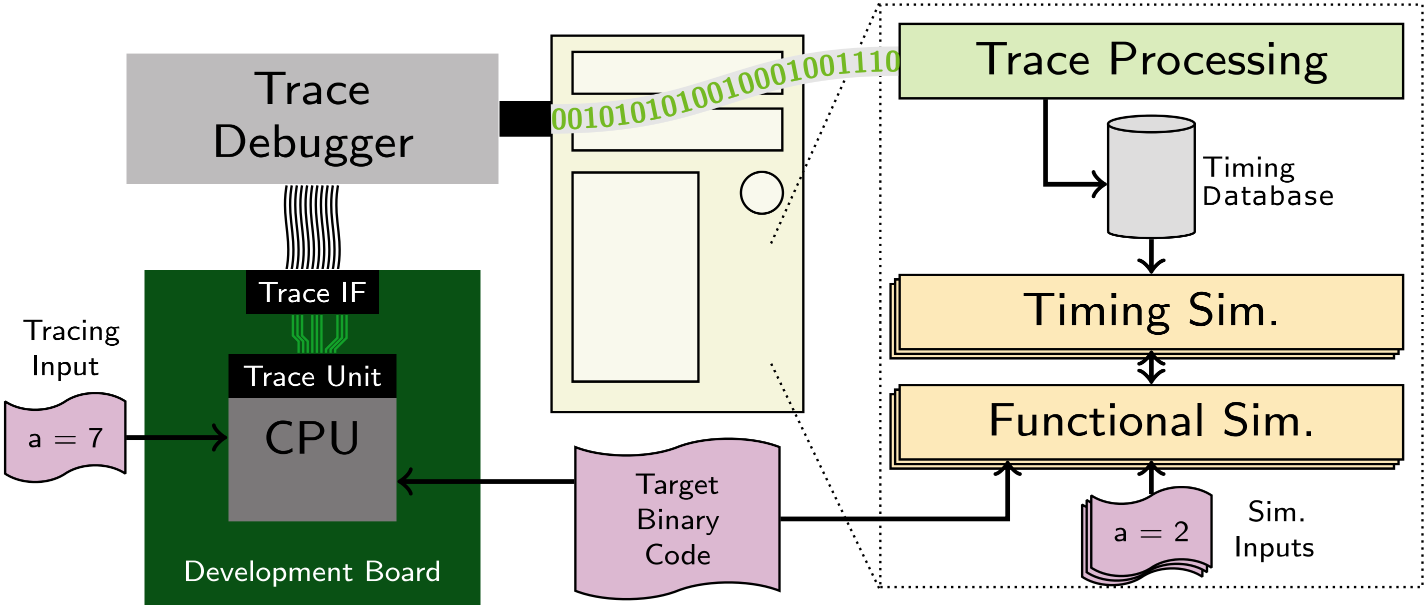
Related Work

Publication	Contexts	Analysis
Chakravarti et al., CODES 2013	1-block	Dynamic
Plyaskin et al., VLSI-SoC 2011	n-block	Dynamic
Stattelmann et al., DAC 2011	VIVU(n, k)	Static
Ottlik et al., CASES 2014	VIVU(n, k)	Static

Drawbacks of existing approaches:

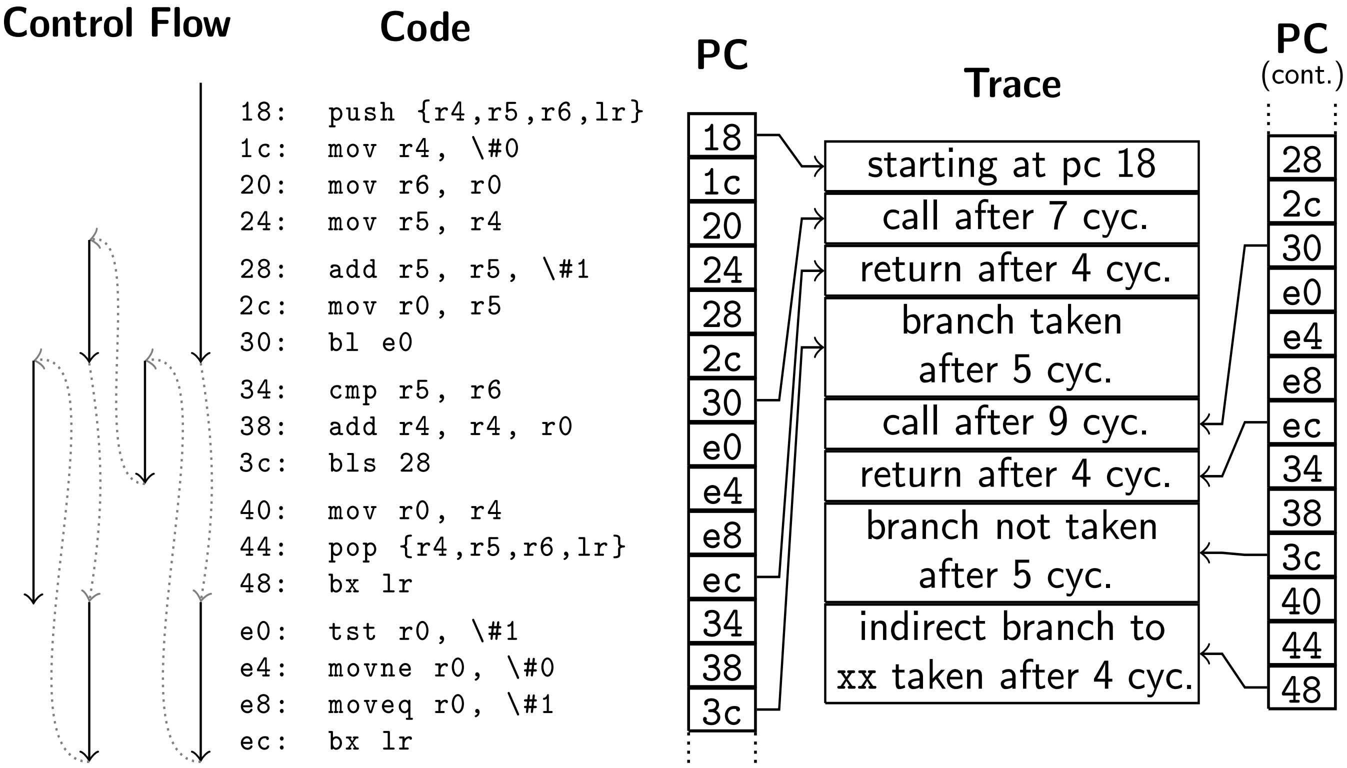
- Static analysis: Hard for superscalar processors
- Plyaskin et al.: No support for control flow variation between observation and simulation
- Chakravarti et al.: Limited use of contexts
- Need to know uArch details

Basic Idea of Trace-Based Approach



- Observing hardware executions
  - No need for uArch model, can consider workload
- Timings are stored in Timing Database (TDB)
  - Must consider differences in execution order between observation and simulation during TDB generation

Way-Point-Oriented HW Traces



```
18:   push {r4,r5,r6,lr}
1c:   mov r4, \#0
20:   mov r6, r0
24:   mov r5, r4

28:   add r5, r5, \#1
2c:   mov r0, r5
30:   bl e0

34:   cmp r5, r6
38:   add r4, r4, r0
3c:   bls 28

40:   mov r0, r4
44:   pop {r4,r5,r6,lr}
48:   bx lr

e0:   tst r0, \#1
e4:   movne r0, \#0
e8:   moveq r0, \#1
ec:   bx lr
```

PC: 18, 1c, 20, 24, 28, 2c, 30, e0, e4, e8, ec, 34, 38, 3c

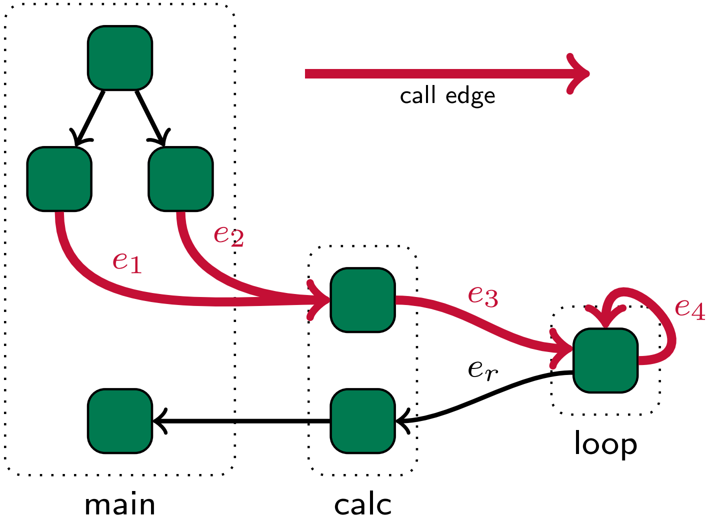
Trace:
- starting at pc 18
- call after 7 cyc.
- return after 4 cyc.
- branch taken after 5 cyc.
- call after 9 cyc.
- return after 4 cyc.
- branch not taken after 5 cyc.
- indirect branch to xx taken after 4 cyc.

PC (cont.): 28, 2c, 30, e0, e4, e8, ec, 34, 38, 3c, 40, 44, 48

- Output by a dedicated HW unit
- Supported by many vendors
  - ARM CoreSight, IEEE-ISTO Nexus 5001, …

Control-Flow Reconstruction

1. Process trace step-by-step
   - CFG created on-the-fly
   - May have to split blocks
   - Identify edge control flow type
2. Identify Loops
3. Extract Loops



Result: Interprocedural Control Flow Graph

- **Loops are represented as recursive routines**

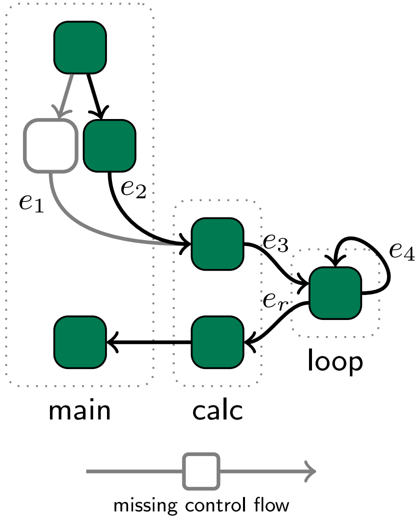
Context Extraction

- Walk through CFG, maintain current context
  - Need CFG first, second pass over trace required
- Write trace timings to TDB in current context
  - Single timing for multiple blocks
    - Interpolate individual timings using instruction count
  - Multiple timings for a single block/context
    - Take average value, store number of observations in TDB

Result: Context-sensitive timings of the observed execution

Timing Database Contents (so far)

Context	Timings e_4	Timings e_r
$(e_2, 0) \circ (e_3, 0)$	10	-
$(e_2, 0) \circ (e_3, 1)$	4	-
$(e_2, 0) \circ (e_3, 2)$	-	8

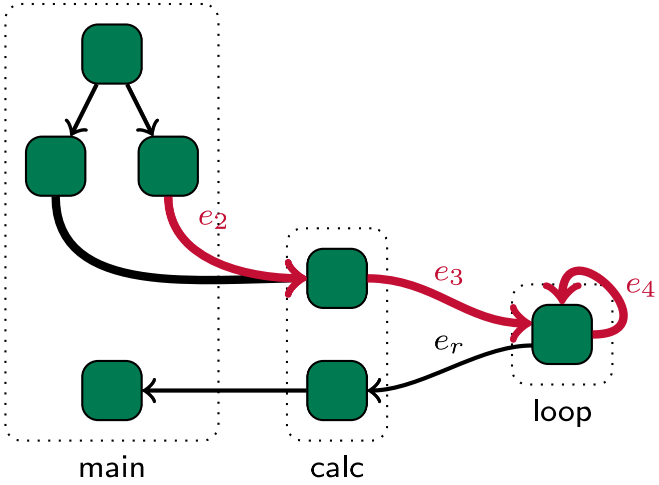


- Trace does not cover every block in every context
  - Block not executed at all, no contexts in other blocks
  - Block only executed in some context during tracing

Goal: Degrade simulation accuracy gracefully

VIVU(n,k) Contexts

- VIVU contexts abstract the call sequence leading to routine execution (incl. loops)
- Call string: sequence of call edges in CFG
  - For example: $e_2 \circ e_3 \circ e_4$
- VIVU: call strings $\longrightarrow$ contexts

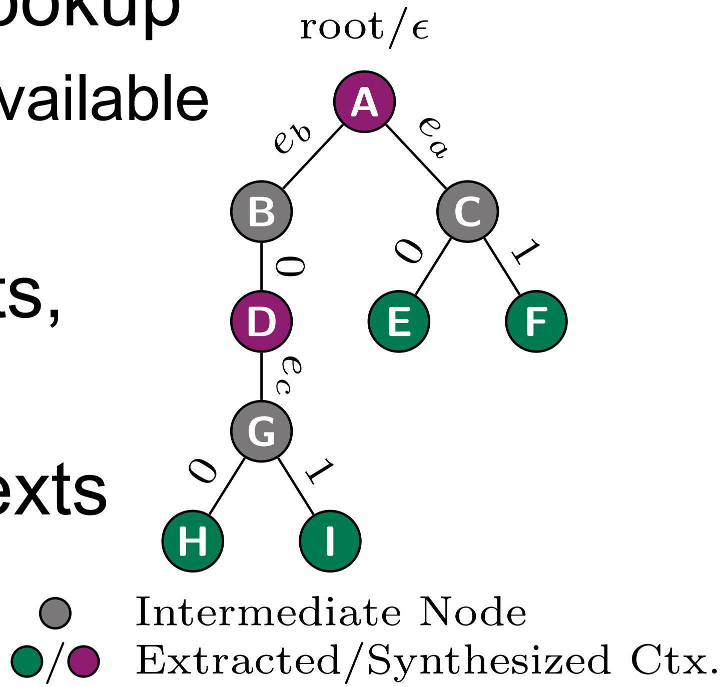


VIVU		Call String	
n	k	$e_2 \circ e_3 \circ e_4$	$e_1 \circ e_3 \circ e_4 \circ e_4$
>1	>2	$(e_2, 0) \circ (e_3, 1)$	$(e_1, 0) \circ (e_3, 2)$
>1	1	$(e_3, 1)$	$(e_3, 2)$
1	1	$(e_3, 1)$	$(e_3, 1)$

See: F. Martin et al., „Analysis of Loops“, Compiler Construction, Springer, 1998

Context Generalization

- Simulation uses tree-based lookup
  - Selects most specific context available
  - See: Ottlik et al., CASES 2014
- Idea: Synthesize new contexts, fill missing data
- Avg. timing from similar contexts
- For example: Fill node E
  - Prefer timing from F
  - If unavailable: Average of H and I
- Implemented in two walks over extended tree



Simulation Basics

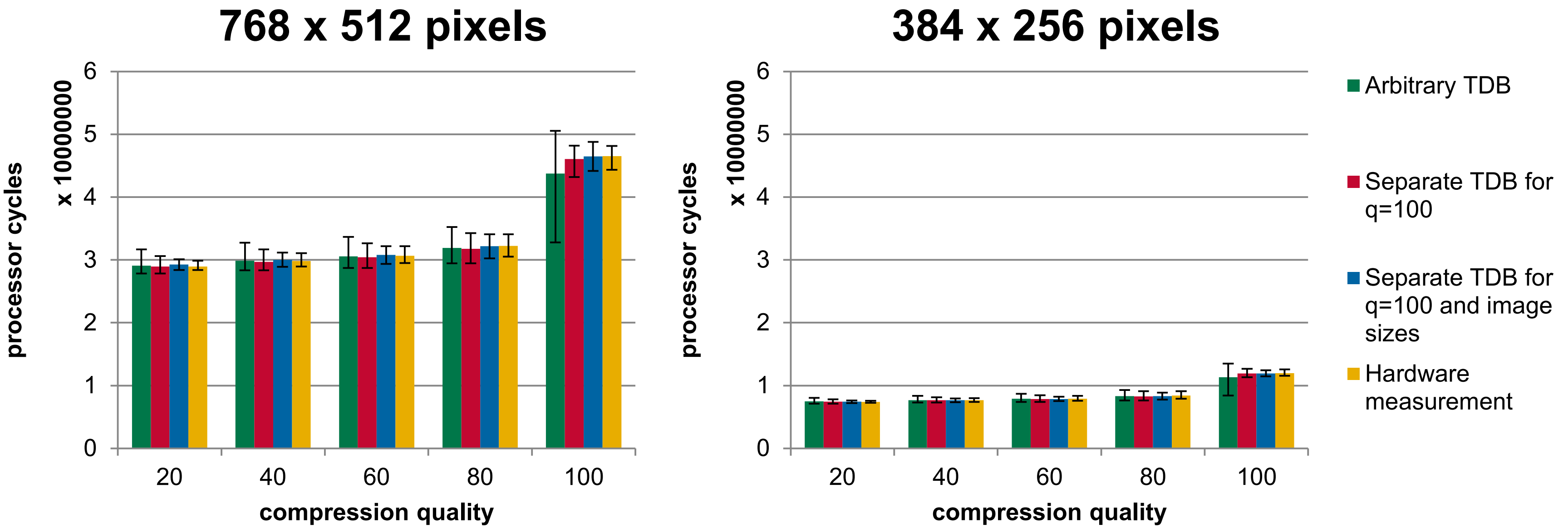
- For each simulated block
  - Advance time by cycle count in current context
  - Update current context if necesarry
- Handling unexpected control flow
  - Previous work: Context reset, not good for tracing
  - Solution: Apply heuristic to maintain context info

Experimental Setup

- Xilinx Zynq XC7Z020/Zedbo
- Software on Cortex-A9
  - OOO Pipeline w/ speculation
  - Caches: Split L1, shared L2
  - DDR3 RAM
- Custom tracing logic in FPG
  - Trace streaming via GigE

- Focus on simulation accuracy in presence of input variations between tracing and simulation

Case Study: Image Compression

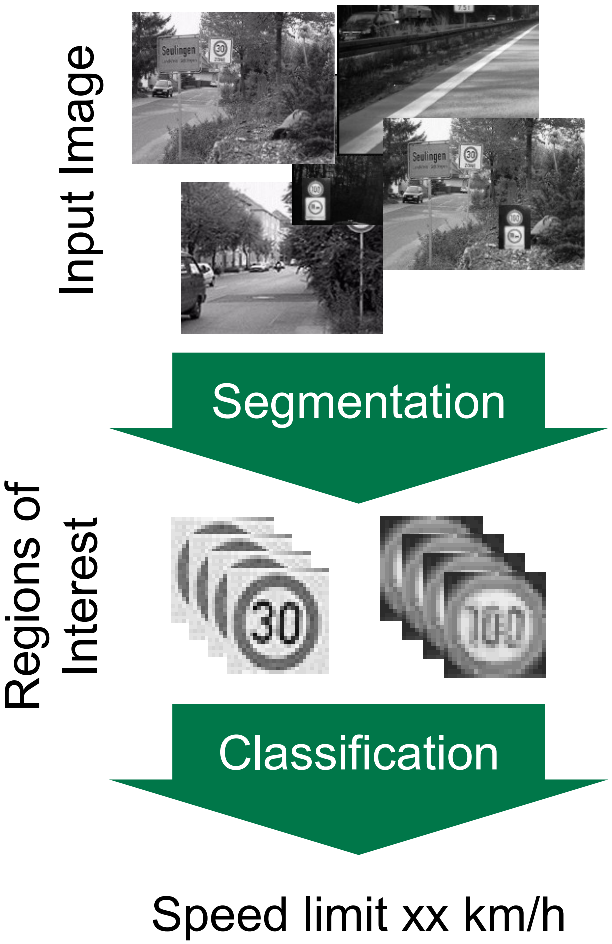


- libjpeg-turbo based image compression
  - Utilizes SIMD instructions of Cortex-A9
- Varied image size & contents, compression quality
  - Quality 100 is a special case, requires extra TDB (otherwise sim. error up to ~30%, with extra TDB ~17%)
  - Same size: additional improvements (max. error ~12%)

Case Study: Traffic Sign Recognition

- Video based recognition of circular traffic signs
  - Segmentation detects circles
  - Each circle input to SVM

- Varied inputs:
  - Image contents
  - Segmentation parameters
    - Min. gradient
    - Min. votes



Input Image

Segmentation

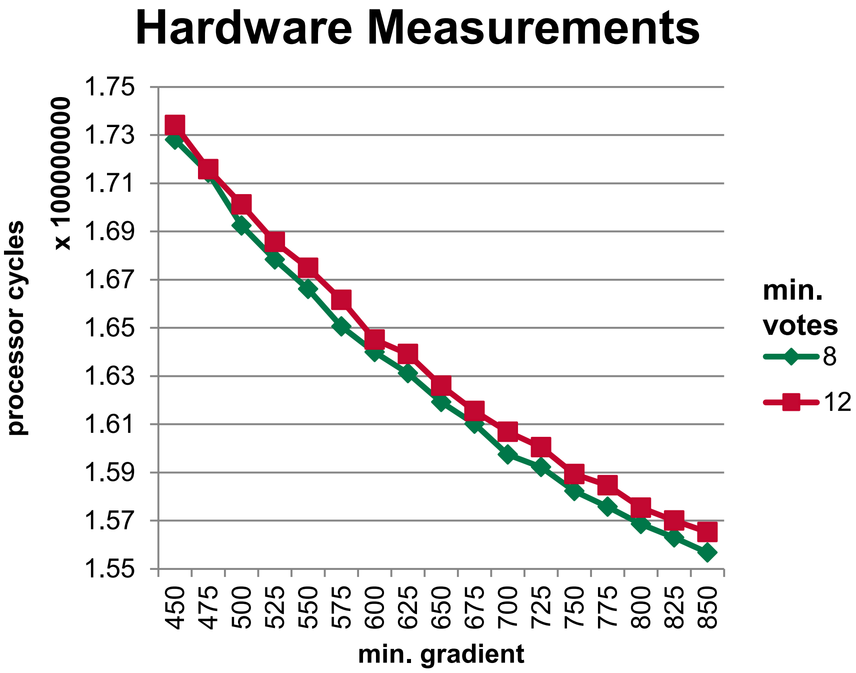
Regions of Interest

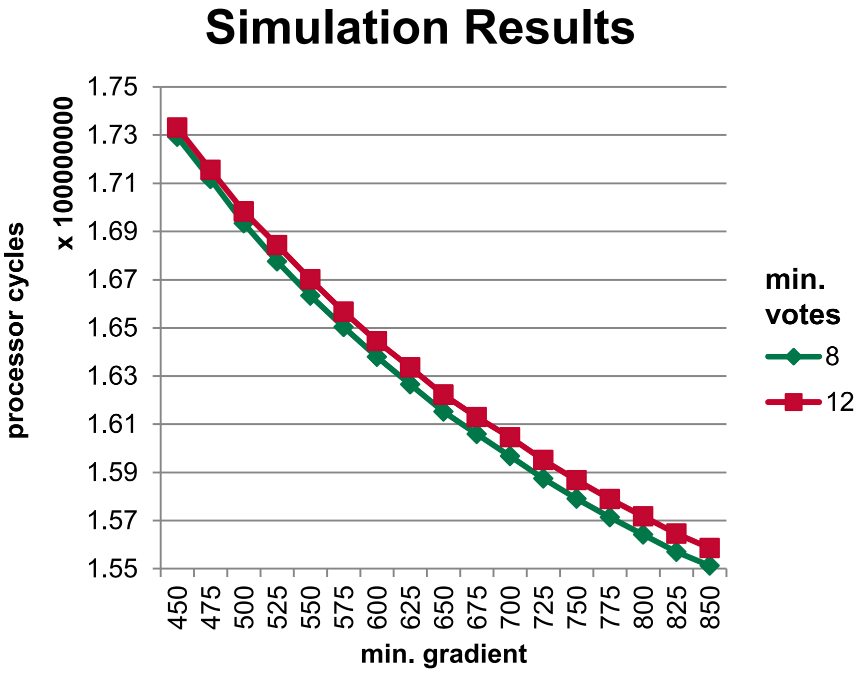
Classification

Speed limit xx km/h

Case Study: Traffic Sign Recognition

- High accuracy requires traffic sign during tracing
  - Max. and Avg. error less than 8% and 2%
  - Same image contents: Max. error nearly 2%
  - Application timing slightly idealized in simulation





Summary

- Tracing as data source for timing simulation
  - Different inputs in simulation

- Simulation highly accurate
  - Without cache model

- Supports complex embedded processors
- Supports complex embedded sofware
- No need for knowledge of processor details
  - Only requires limited knowledge of instruction set

Questions?

ottlik@fzi.de