Branch Target Buffers: WCET Analysis Framework & Timing Predictability

Daniel Grund[1] Jan Reineke[1] Gernot Gebhard[2]

[1]Saarland University, Saarbrücken, Germany

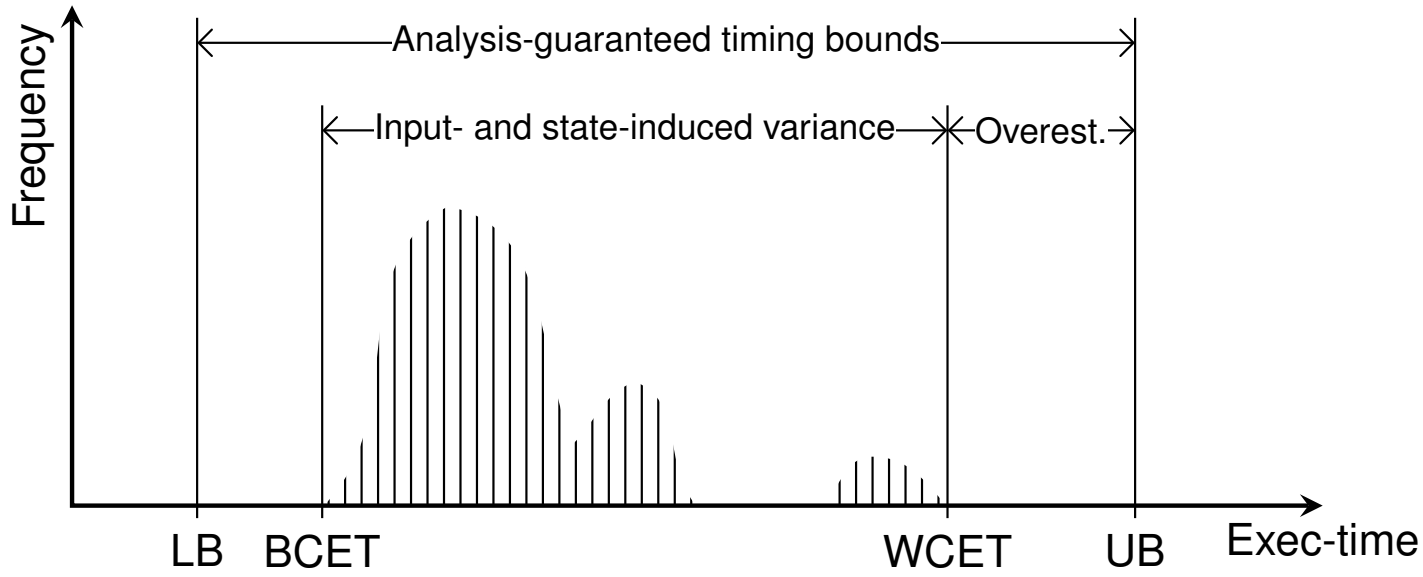
[2]AbsInt GmbH, Saarbrücken, Germany

RTCSA, Aug 2009

Outline

Outline

Notions in Timing Analysis



- Execution time depends on
  - program input
  - initial hardware state
- Bounds required for schedulability analysis

Branch Prediction vs. Branch Target Prediction

Branch Prediction

Predict whether conditional branches will be taken or not.

- static: prediction encoded in instruction
- dynamic: prediction depends on execution history

Branch Target Prediction

Predict target of (un-)conditional branches

- using a Branch Target Buffer (BTB)
- even before instruction decoding
- scan fetch buffer
- query BTB with address of instruction
- possibly redirect fetching

Disambiguation: 3 Types of BTBs

- Many different names for different kinds of BTBs
- Names not always used consistently. Disambiguation:

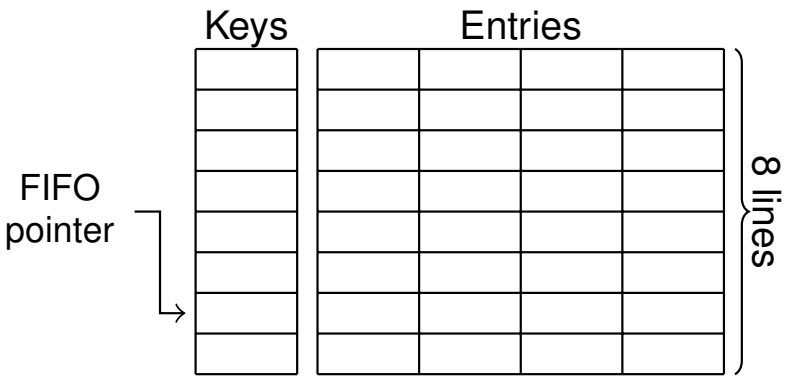
```
20: b 40
    :
40: i1
44: i2
48: i3
52: i4
```

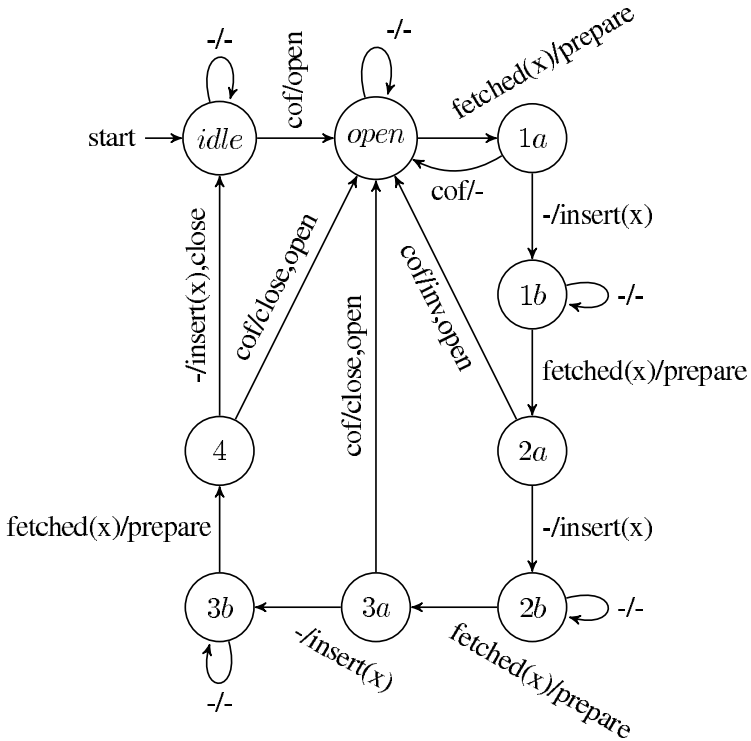
address-BTB	`20 ↦ 40`
source-BTIC	`20 ↦ [i1, i2, i3, i4]`
target-BTIC	`40 ↦ [i1, i2, i3, i4]`

⇒ Common ground: Associative structure that maps *keys* to *entries*

The BTB of the MPC56x: Facts & Figures

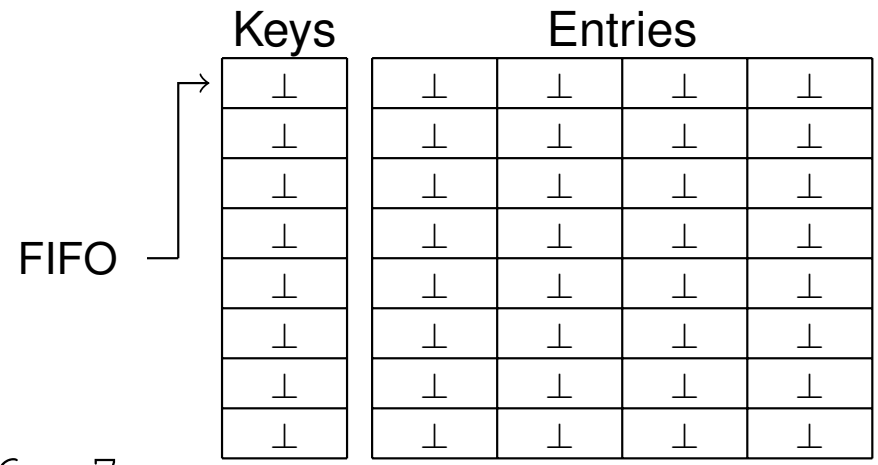
- The MPC56x employs a tgt-BTIC with:
  - 8 lines
  - 4 entries per line
  - FIFO replacement
  - intricate line filling





The BTB of the MPC56x: Example

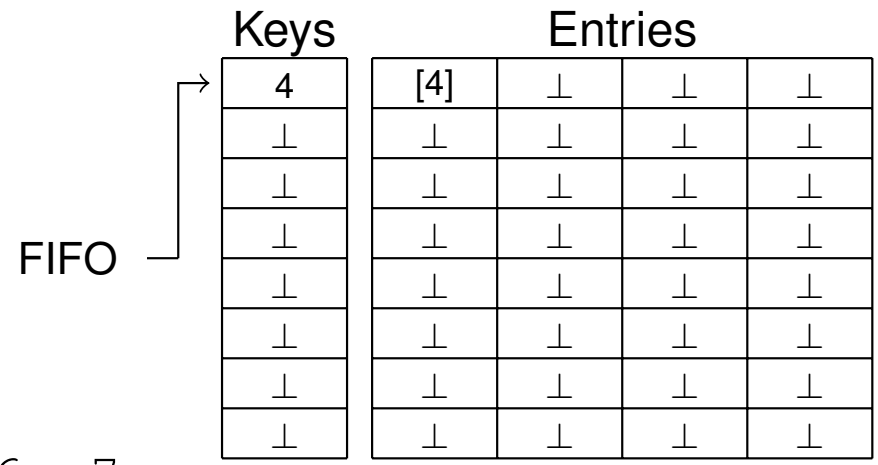

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```



The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

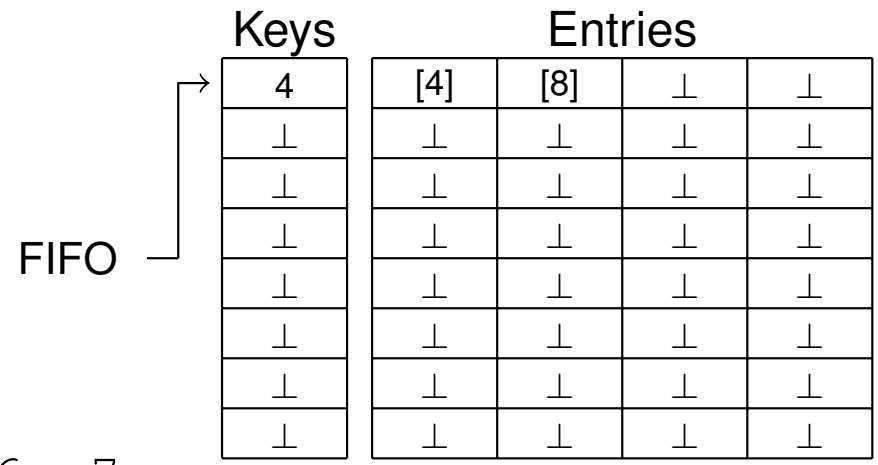
```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40     88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64    108:
```



The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

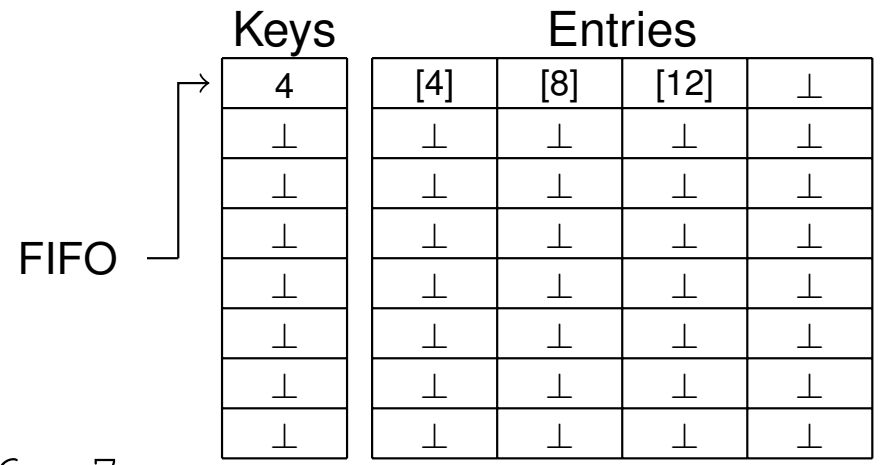
```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```



The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

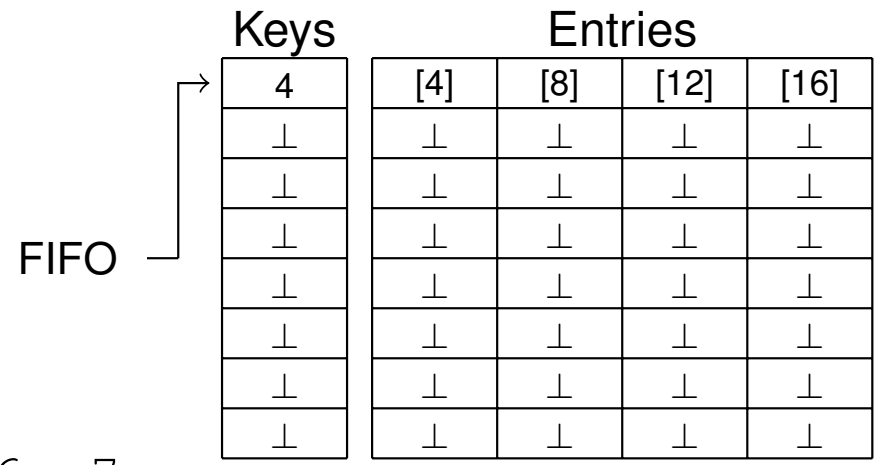
```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```



The BTB of the MPC56x: Example

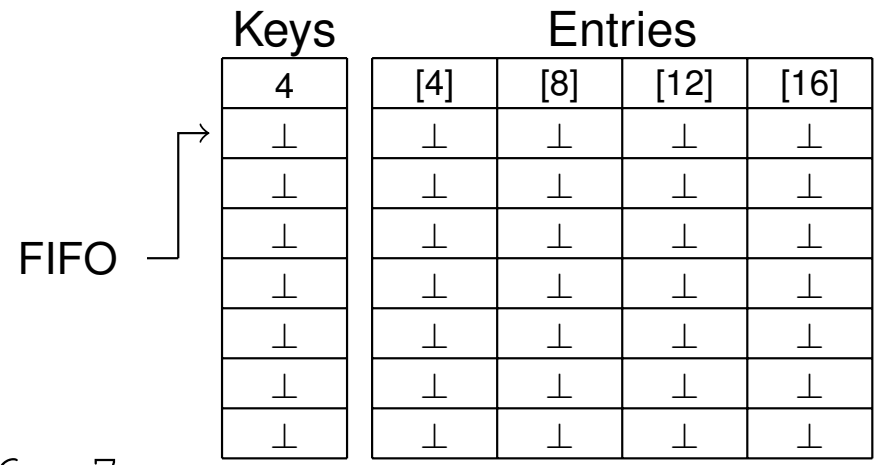
SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```



The BTB of the MPC56x: Example


```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```



The BTB of the MPC56x: Example

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```

FIFO

Keys	Entries			
4	[4]	[8]	[12]	[16]
40	[40]	[44]	[48]	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥

The BTB of the MPC56x: Example

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64      108:
```

FIFO

Keys	Entries			
4	[4]	[8]	[12]	[16]
40	[40]	[44]	[48]	⊥
64	[64]	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥

The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40     88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64    108:
```

FIFO

Keys	Entries			
4	[4]	[8]	[12]	[16]
40	[40]	[44]	[48]	⊥
⊥	[64]	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥

The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40     88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64    108:
```

FIFO →

Keys	Entries			
4	[4]	[8]	[12]	[16]
40	[40]	[44]	[48]	⊥
⊥	[64]	⊥	⊥	⊥
80	[80]	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥

The BTB of the MPC56x: Example

SAARLAND UNIVERSITY
COMPUTER SCIENCE

```
 0:   b 4      64:
 4:            68:   b 80
 8:
12:            80:
16:            84:
20:   b 40      88:
               92:
40:            96:   divw r3,r6,r7
44:           100:   cmp cr0,r3,r4
48:           104:   bc 88
52:   b 64    108:
```

FIFO →

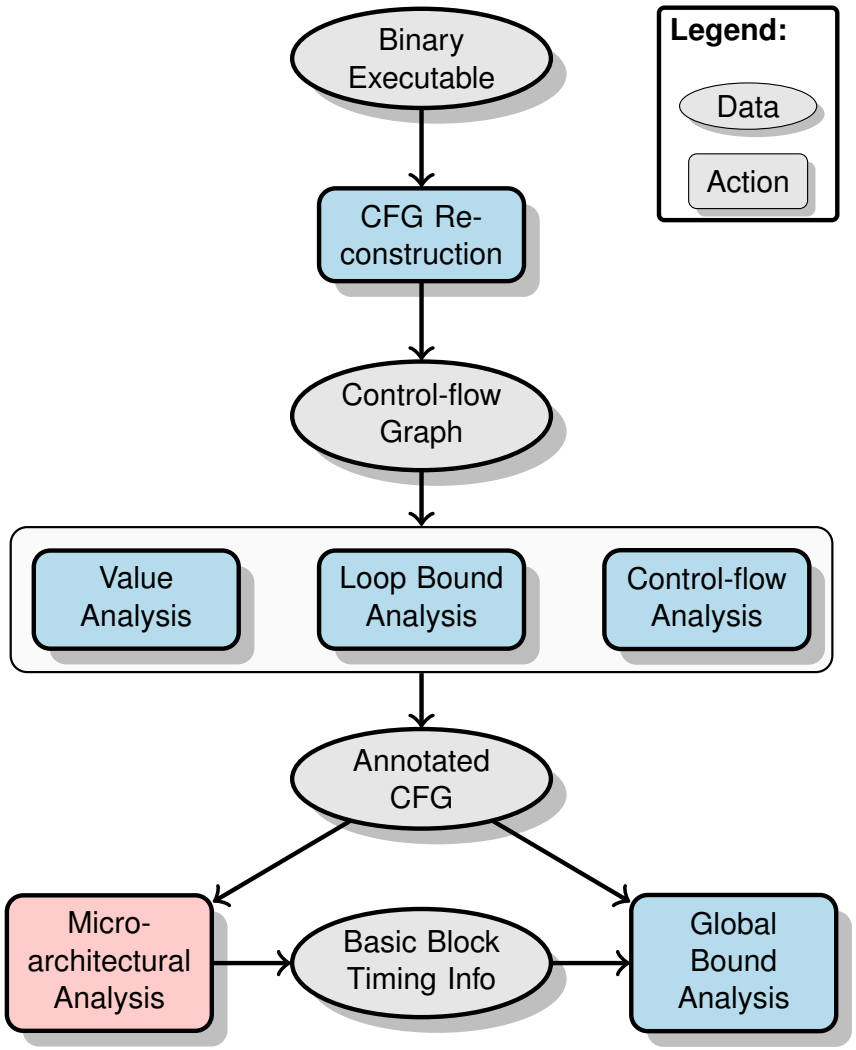
Keys	Entries			
4	[4]	[8]	[12]	[16]
40	[40]	[44]	[48]	⊥
⊥	[64]	⊥	⊥	⊥
80	[80]	[84]	[88]	[92]
88	[88]	[92]	⊥	⊥
108	[108]	[112]	[116]	[120]
⊥	⊥	⊥	⊥	⊥
⊥	⊥	⊥	⊥	⊥

Problem Statement

- Define a BTB analysis that
  - can classify BTB accesses
  - is generic and covers most BTB variants
- Built upon results of other analyses:
  - value analysis (addresses, values in registers)
  - other microarchitectural analyses
- Evaluate precision and applicability

Outline

General WCET Analysis Framework



- E.g. implemented by *aiT* of AbsInt
- Several static analyses
- μ-arch. analysis is most expensive
- μ-arch. analysis can be seen as a single abstract interpretation

- Contribution:
  BTB Analysis Framework
  Abstract domain templates for BTBs

BTB Analysis Framework: Idea

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Decompose the analysis into

- Main analysis:
  - drives the whole analysis
  - maintains BTB-unspecific analysis data
- Two parameter analyses:
  - further decomposition of the analysis
  - have fixed interfaces (towards main analysis)
  - can be instantiated for a specific BTB
- Formalization as abstract interpretation in paper

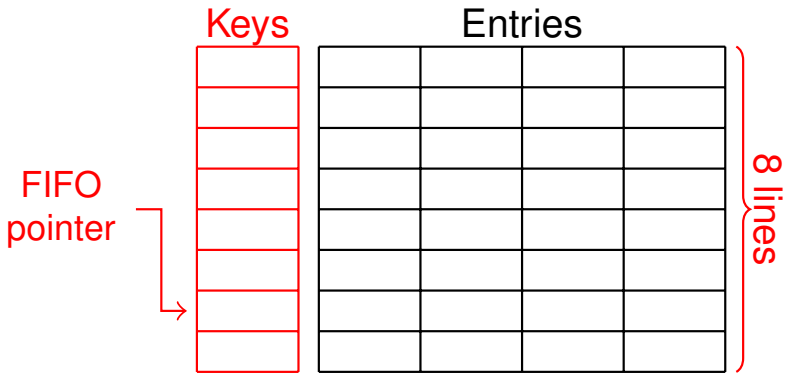
Main Analysis

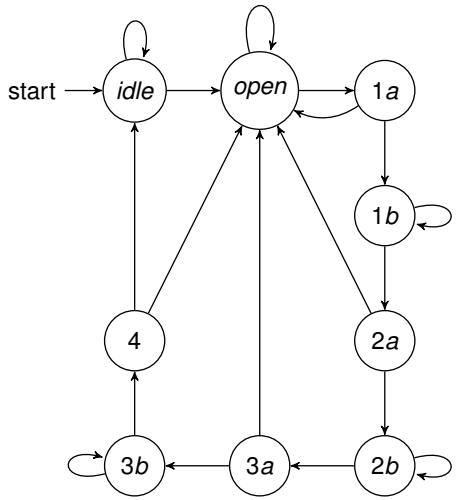
- Drives the analysis: currFetch $\neq$ succ(lastFetch) $\Rightarrow$ change of flow
- Triggers updates of parameter analyses
- Inhibited status
- Set of potential last branches

- See paper for details

Key Analysis

- For which keys does the BTB contain information?





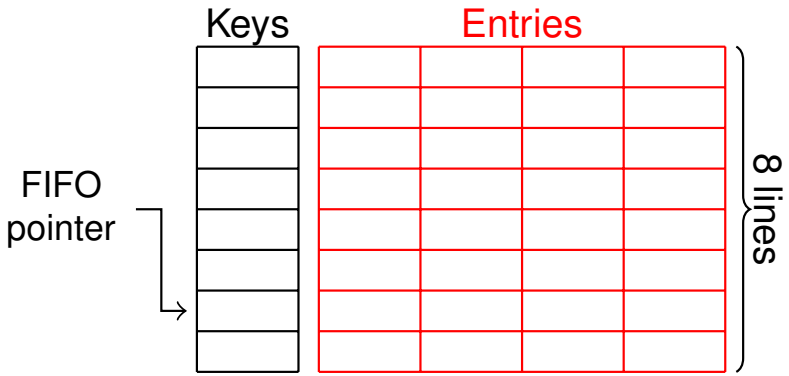
- Parameters: Replacement policy, #sets, #lines/set

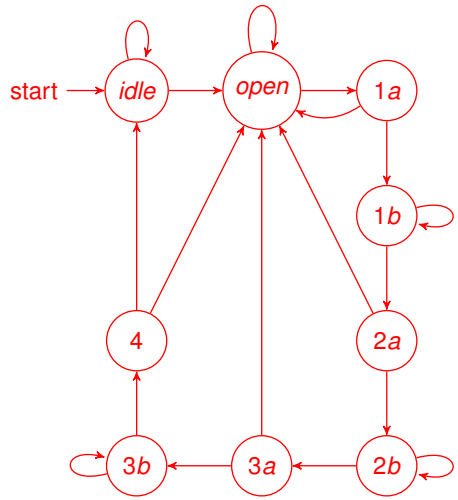
$\Rightarrow$ Similar to cache analysis

- Challenge in MPC56x: FIFO replacement

Contents Analysis

- What information is contained for a key?





- Parameters: line size, line filling process

$\Rightarrow$ One has to model the automaton

- Challenge in MPC56x: Intricate line filling and invalidation

Instantiation of Key Analysis for MPC56x

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Need FIFO cache analysis
- Back then, no viable FIFO analysis existed
- In principle, cache analyses could handle only LRU

Solution:

- Use *relative competitiveness* to obtain may-analysis for FIFO
- Define a FIFO must-analysis that can benefit from the may-analysis

Static Cache Analysis

May-analysis

- Overapproximation of cache contents
- Used to soundly predict cache misses

Must-analysis

- Underapproximation of cache contents
- Used to soundly predict cache hits

FIFO May-Analysis

- "How many misses would FIFO have if LRU has m_{LRU} misses?"

Definition: Relative Competitiveness

Policy P is (f, c) miss-competitive relative to policy Q if

$$m_P(p, s) \leq f \cdot m_Q(q, s) + c$$

for all access sequences s and compatible cache states p, q.

- E.g. LRU$(2k-1)$ is $(1, 0)$ miss-competitive vs. FIFO(k)
- I.e. $m_{\text{LRU}(2k-1)}(p, s) \leq m_{\text{FIFO}(k)}(q, s)$

$\Rightarrow$ LRU$(2k-1)$ may-analysis can be used for FIFO(k) may analysis

A Complementing FIFO Must-Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Our proposed must-analysis can predict
  - “trivial hits”
  - “hits after misses”

Trivial hit

Last accessed element is always cached

Hits after miss

- After a miss on element e it takes k misses to evict e
- Maintain upper bound b on number of misses since insertion of e
- If $b < k$, then e must still be cached

Instantiation of Contents Analysis for MPC56x

- Too many details for a talk
- Only one example: *Occupancy of open lines* (OOL) and closed lines (OCL)
- Problem at control-flow join points:
  - Unclear whether reached via branch or fall-through
  - BTB line maybe gets filled, maybe not
- Solution:
  - Maintain approximations for OOL and OCL
  - Upon *definite* line closing, transfer information from OOL to OCL
  - Upon *potential* line closing, take join of both informations

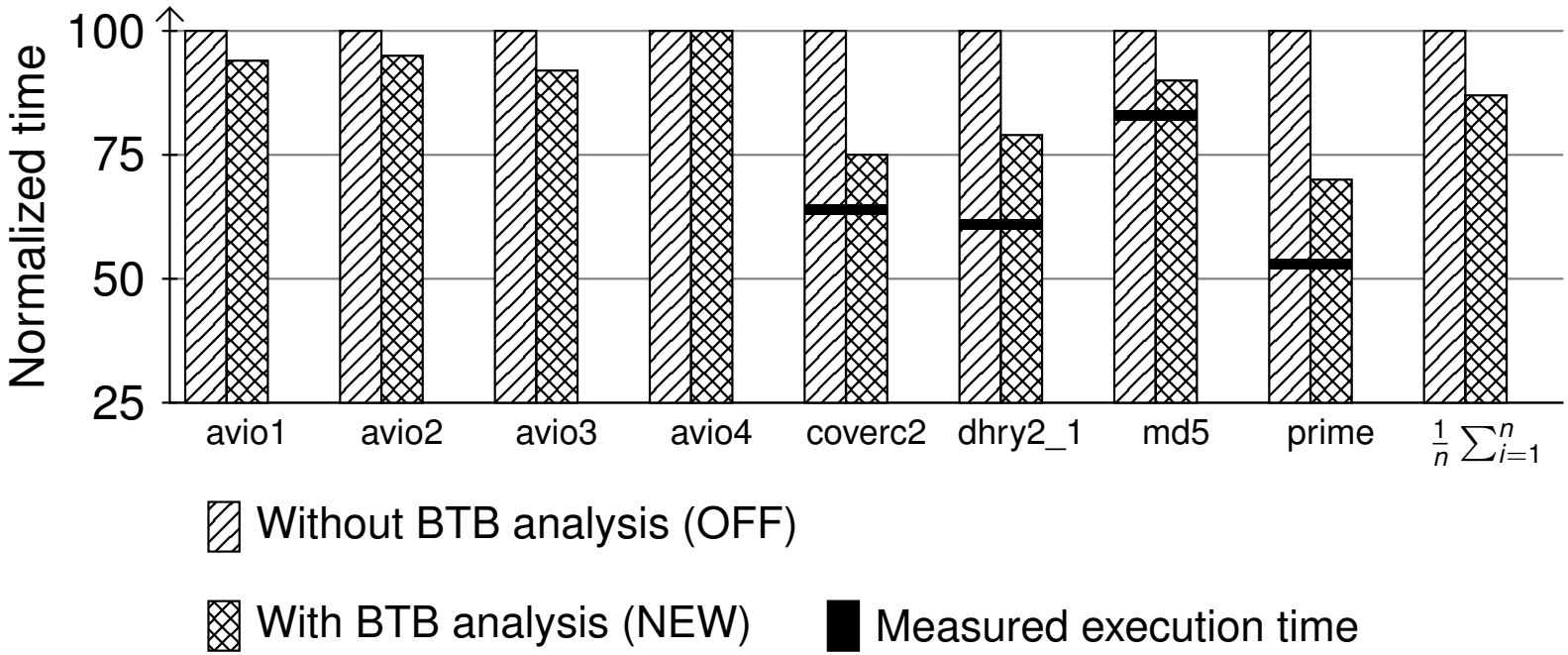
Evaluation Setup

- Model validated against traces of hardware events
- No observed behavior was excluded by analysis

- BTB analysis integrated in industry-strength timing analyzer *aiT*
- Configured MPC565 for rather low memory latency of 4 cycles

- 4 hard real-time avionics benchmarks (NDA)
- 4 other programs

Evaluation Results



- Improvement of NEW over OFF is 13% on average
- Measuring execution time possible only for non-avionics
- Overestimation reduced from 54% to 20%

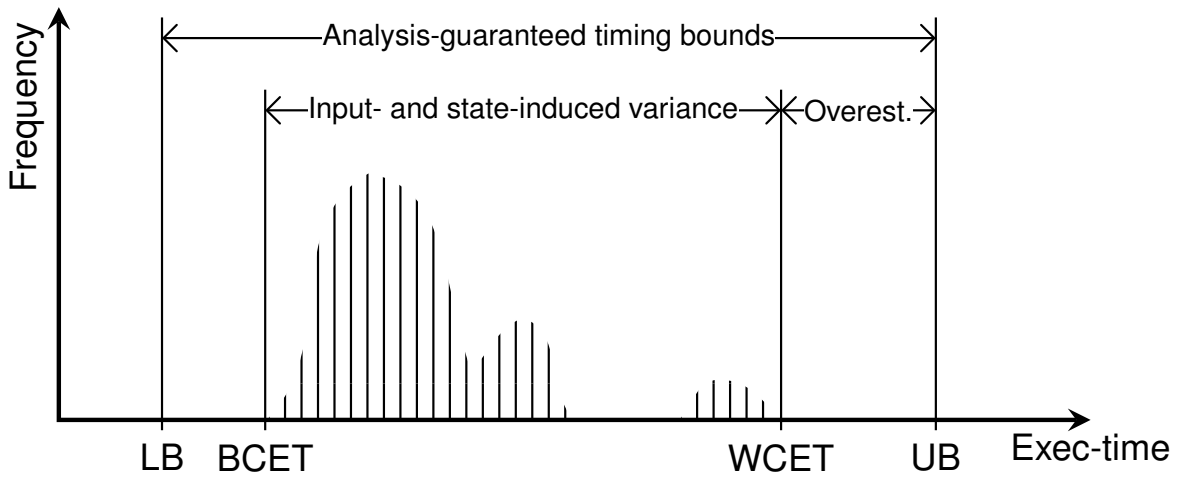
Outline

Predictability Considerations

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Predictability

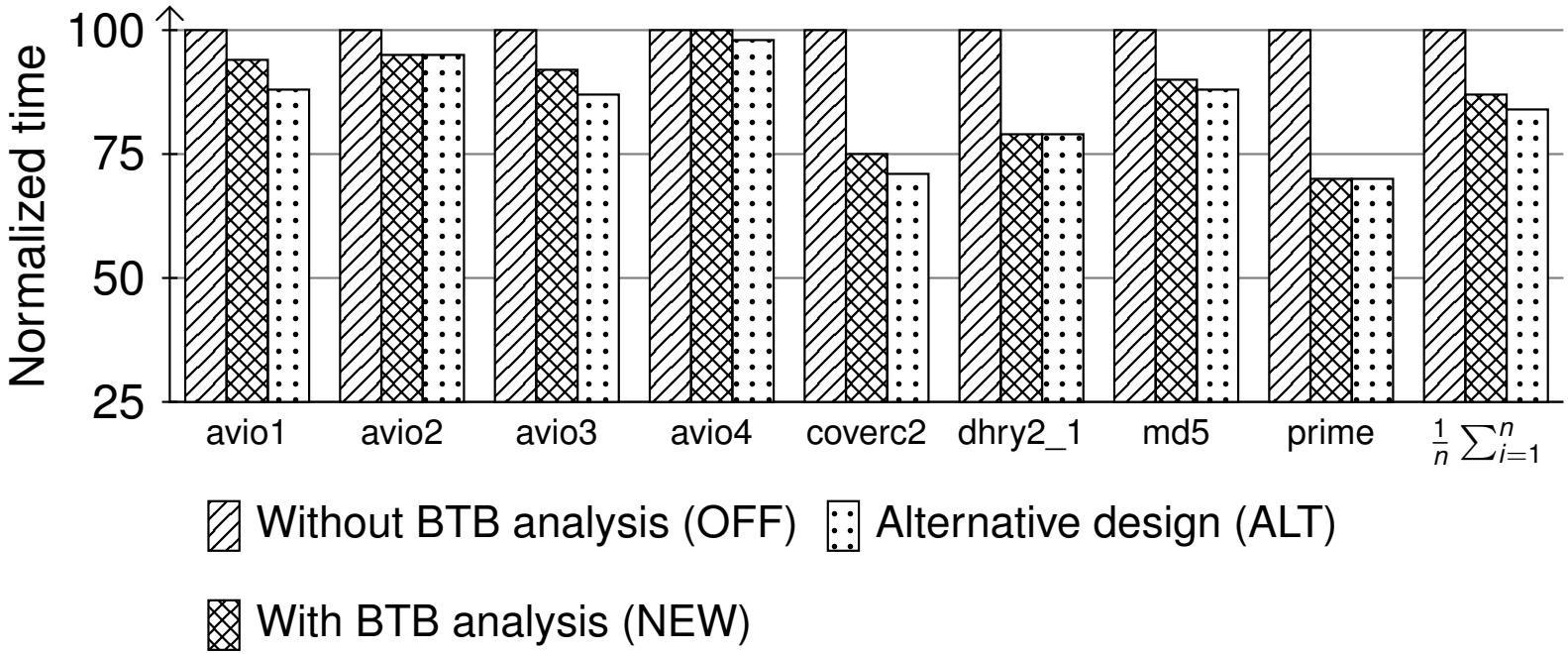
"How well does a system lend itself to static analysis?"
Obtainable WCET bounds, overestimation, analysis time



- Improve timing predictability by alternative BTB design
- Instantiate framework accordingly to quantify results

Predictability: LRU instead of FIFO

- Static analysis of LRU is easier than analysis of FIFO
- Assume LRU-managed BTB



- WCET bounds improve by 3% on average
- Reduced memory consumption during analysis

Predictability: Empty vs. Any Starting State

- Assume availability of BTB invalidation instruction
- Invalidate BTB at program start
- Analysis only needs to consider the *empty* starting state, not *any*

	Any			Empty		
Benchmark	Hit	Miss	Uncl	Hit	Miss	Uncl
coverc2	48.8	46.5	4.7	50.9	48.9	0.2
md5	17.1	19.3	63.6	44.3	55.7	0.0
prime	6.5	0.0	93.5	65.9	27.5	6.6
dhry2_1	44.3	22.0	33.7	57.6	42.4	0.0
avio1	25.9	3.1	71.0	36.1	22.5	41.4
avio2	21.0	73.9	5.1	21.0	77.2	1.8
avio3	22.7	1.1	76.2	29.4	27.9	42.7
avio4	0.0	0.0	100.0	4.5	42.4	53.1
Average	23.3	20.8	**55.9**	38.7	43.1	**18.2**

- WCET bounds improve by 5% on average

Outline

Summary

BTB analysis framework:

- Decomposition into key- and contents-analysis
- Relative competitiveness for FIFO may-analysis
- Complementing must-analysis

Predictability:

- For real-time systems use LRU
- Offer hardware-state-controlling instructions
- Hardware performance should be monotone in (buffer) sizes

Summary

BTB analysis framework:

- Decomposition into key- and contents-analysis
- Relative competitiveness for FIFO may-analysis
- Complementing must-analysis

Predictability:

- For real-time systems use LRU
- Offer hardware-state-controlling instructions
- Hardware performance should be monotone in (buffer) sizes

Thanks for listening. Questions?

Further Reading


R. Wilhelm et al.
The worst-case execution time problem—
overview of methods and survey of tools
Transactions on Embedded Computing Systems, 7(3), 2008


J. Reineke and D. Grund
Relative competitive analysis of cache replacement policies
LCTES 2008


J. Reineke, D. Grund, C. Berg, and R. Wilhlem
Timing predictability of cache replacement policies
Real-Time Systems, 37(2):99-122, 2007


AbsInt GmbH
aiT worst-case execution time analyzer
`http://www.absint.com/ait/`