WCET Analysis in Shared Resources Real-Time Systems with TDMA Buses

Claire Maiza

in collaboration with:
Hamza Rihani, Matthieu Moy and Sebastian Altmeyer
published in RTNS 2015

OVSTR, 11/06/2018
Precise static WCET analysis: combining path and TDMA analyses

Main idea

- TDMA causes no direct interference
- Task execution time does not depend on what is executed on other cores
- New analysis that combines:
  - a precise analysis on when memory accesses happen (path analysis)
  - a precise analysis on tdma slot (bus analysis)
- Use of SMT encoding
Outline

1. Context: TDMA bus
2. Motivation on an example
3. Our Contribution
4. Performance Evaluation
5. Conclusion and Discussion
Shared resources: contention, access delays

- Time Division Multiple Access (TDMA) arbitration policy
  - Time triggered policy
  - Assign fixed time slots to each core
Core A viewpoint:

- Time
- Core A
- Core B
- Core C

Period: \( \pi \)
Slot length: \( \sigma \)

- \( \text{req}_1 \) requests
- \( \text{resp}_1 \) responses
- \( \text{acc} \)

- Execution time of a granted request: \( T \)
- Waiting time to grant requests: \( \in [0, \pi - (\sigma - \text{acc})] \)

**Pessimistic approach:**

\[
\text{Worst-Case}(T) = \pi - (\sigma - \text{acc})
\]
Time Division Multiple Access

Core A viewpoint:

- **Period** $\pi$
- **Slot length** $\sigma$

**Requests and Responses**:

- $req_1$
- $resp_1$
- $req_2$
- $resp_2$

**Accumulation and Execution Times**:

- $T_{acc}$: execution time of a granted request
- $T_{waiting}$: waiting time to grant requests

**Pessimistic Approach**:

$$Worst-Case(T) = \pi - (\sigma - acc)$$

**Offset relative to the TDMA period**:

$$Offset(req) = time_{instant(req)} \mod \pi$$

Claire Maiza

WCET TDMA bus

11/06/2018 5 / 16
Time Division Multiple Access

Core A viewpoint:

- req₁
- resp
- acc

- slot length $\sigma$
- period $\pi$

$p$-period

$\sigma$

$\pi$

$\text{req}_1$

$\text{resp}$

$\text{acc}$

$\text{T}_{\text{execution time of a granted request}}$

$\text{T}_{\text{waiting time to grant requests}}$

$\in [0, \pi - (\sigma - \text{acc})]$

Pessimistic approach:

$\text{Worst-Case}(T) = \pi - (\sigma - \text{acc})$
Core A viewpoint:

- req1: Request 1
- resp: Response
- acc: Acknowledgment
- Time Division Multiple Access (TDMA)
- Core A, Core B, Core C
- Period $\pi$
- Slot length $\sigma$
- Execution time of a granted request $T$
- Waiting time to grant requests
- Offset relative to the TDMA period: $\text{Offset}(\text{req}) = \text{time instant}(\text{req}) \mod \pi$

Pessimistic approach:

$\text{Worst-Case}(T) = \pi - (\sigma - \text{acc})$
Core A viewpoint:

- acc execution time of a granted request
- $T$ waiting time to grant requests $\in [0, \pi - (\sigma - acc)]$

**Pessimistic approach:** $Worst-Case(T) = \pi - (\sigma - acc)$
Core A viewpoint:

- $acc$: execution time of a granted request
- $T$: waiting time to grant requests $\in [0, \pi - (\sigma - acc)]$

**Pessimistic approach:** $Worst-Case(T) = \pi - (\sigma - acc)$

- Offset relative to the TDMA period:
  $$Offset(req) = time\_instant(req) \mod \pi$$
Motivation: WCET Analysis of TDMA

Goal: Estimate the WCET

BB 1
/* 3 cycles */
if (cond)

BB 2
/* 10 cycles*/

BB 3
if (!cond)

BB 4
/* bus access */

BB 5
/* 5 cycles */
return()
Motivation: WCET Analysis of TDMA

Goal: Estimate the WCET
  ▶ Option 1: Worst-case everywhere.

Claire Maiza

WCET TDMA bus

11/06/2018
Motivation: WCET Analysis of TDMA

Goal: Estimate the WCET

- Option 1: Worst-case everywhere.
- Option 2: Capture all possible offsets
Motivation: WCET Analysis of TDMA

Goal: Estimate the WCET

- Option 1: Worst-case everywhere.
- Option 2: Capture all possible offsets
- Option 3: Feasible Path Analysis
Contribution

TDMA Bus Analysis
Kelter et al., RTS’14
Chattopadhyay et al., SCOPES’10

Feasible Path Analysis with SMT
Henry et al. LCTES’14
Contribution:

Compute WCET by encoding the semantics and shared resource accesses into an optimization problem (SMT)
WCET Analysis by SMT Encoding

- Bounded Model Checking
  - Encode the semantics into a Satisfiability Modulo Theory problem

SMT query

\[ \text{assert} (\bigwedge \text{expr}) \]

Is there a trace with correct semantics such that the execution time is greater than $X$?

SMT-solver response:

- SAT:
- UNSAT:

Goal:

Find the smallest $X$, such that Execution Time $> X$ is UNSAT
WCET Analysis by SMT Encoding

- Bounded Model Checking
  - Encode the semantics into a Satisfiability Modulo Theory problem

SMT query = “Is there a trace with correct semantics?”

(assert(∧ expr)

- SMT-solver response:
  - SAT: There is a feasible execution path
  - UNSAT: There is no feasible execution path
WCET Analysis by SMT Encoding

- Bounded Model Checking
  - Encode the semantics into a Satisfiability Modulo Theory problem
- Add execution times on the paths

\[
\text{SMT query} = \text{“Is there a trace with correct semantics?”}
\]
\[
\text{assert(} \land \text{expr)}
\]

- SMT-solver response:
  - SAT:
  - UNSAT:
WCET Analysis by SMT Encoding

- Bounded Model Checking
  - Encode the semantics into a Satisfiability Modulo Theory problem
- Add execution times on the paths

SMT query = “Is there a trace with correct semantics

\[ \text{assert}(\land \text{expr}) \]

...such that the execution time is greater than \( X \)”

- SMT-solver response:
  - SAT: There is a feasible path with an execution time > \( X \)
  - UNSAT: We found a WCET upper bound
Bounded Model Checking

- Encode the semantics into a Satisfiability Modulo Theory problem
- Add execution times on the paths

\[
\text{SMT query} = \text{“Is there a trace with correct semantics such that the execution time is greater than } X?\text{”}
\]

SMT-solver response:
- SAT: There is a feasible path with an execution time \( > X \)
- UNSAT: We found a WCET upper bound

**Goal:**

Find the smallest \( X \), such that Execution Time \( > X \) is UNSAT
**Example: Semantics and Timing Encoding**

- $b_i$ "true" if $B_i$ executed
- $t_{i,j}$ "true" if transition $B_i \rightarrow B_j$ taken

\[
\begin{align*}
B_1 & : y = \text{read\_value()} \\
& \text{if } (y < 0) \\
& \text{pred} = (y < 0) \\
& t_{1,2} = b_1 \land \text{pred} \\
B_2 & : /* 10 cycles*/ \\
B_3 & : \text{if } (y \geq 0) \\
B_4 & : /* \text{bus access */} \\
B_5 & : \\
\end{align*}
\]
Example: Semantics and Timing Encoding

\[
\begin{align*}
\text{pred} &= (y < 0) \\
t_{1,2} &= b_1 \land \text{pred} \\
e_{1,2} &= \text{start} + \text{wcet}(B_1) \\
t_1 &= b_1 \land \text{pred} \\
e_1 &= \text{start} + \text{wcet}(B_1) \\
t_2 &= 10 \\
e_2 &= e_{1,2} + 10 \\
b_3 &= t_{1,3} \lor t_{2,3} \\
t_3 &= b_{1,3} \lor t_{2,3} \\
e_3 &= \text{start} + \text{wcet}(B_3) \\
\text{execution time} &= \begin{cases} e_{3,5} & \text{if } t_{3,5} \text{ then } e_{3,5} \text{ else } e_{4,5} \\ & \text{else} \end{cases}
\end{align*}
\]

- \(b_i\) "true" if \(B_i\) executed
- \(t_{i,j}\) "true" if transition \(B_i \rightarrow B_j\) taken
- \(e_{i,j}\) execution time at transition \(B_i \rightarrow B_j\)
Example: Semantics and Timing Encoding

```
start = 0

B_1
y = read_value()
if (y < 0)
  e_{1,2} = start + wcet(B_1)
else
  B_2
  /* 10 cycles*/
  e_{2,3} = e_{1,2} + 10
  B_3
  if (y ≥ 0)
    B_4
    /* bus access */
    e_{3,5}
  else
    e_{4,5}

 execution time = if t_{3,5} then e_{3,5} else e_{4,5}
```

- \( e_{i,j} \) execution time at transition \( B_i \rightarrow B_j \)
Example: Semantics and Timing Encoding

\[ \text{start} = 0 \]

\( B_1 \)

\[ y = \text{read\_value}() \]

\[ \text{if } (y < 0) \]

\( B_2 \)

\[ \text{/* 10 cycles*/} \]

\[ e_{2,3} = e_{1,2} + 10 \]

\( B_3 \)

\[ \text{if } (y \geq 0) \]

\( B_4 \)

\[ \text{/* bus access */} \]

\[ e_{4,5} = e_{3,4} + \text{tdma\_cost}(e_{3,4}) \]

\( B_5 \)

\[ e_{1,2} = \text{start} + \text{wcet}(B_1) \]

\[ e_{3,5} \]

execution time = if \( t_{3,5} \) then \( e_{3,5} \) else \( e_{4,5} \)

- \( e_{i,j} \) execution time at transition \( B_i \rightarrow B_j \)
- \( \text{tdma\_cost()} \) execution time of a bus access
Algorithm \( tdma\_cost \): returns the execution time of a bus access

Require: \( e_{entry} \)

1: \( \text{offset}_{entry} \leftarrow e_{entry} \mod \pi \)
2: if \( \text{offset}_{entry} \in [0, \sigma - acc] \) then /* offset is inside the slot */\( ① \)
3: \( \text{return} \ acc \)
4: else /* offset is outside the slot */\( ② \)
5: \( \text{return} \ (\pi - \text{offset}_{entry}) + acc \)
6: end if
Proof-of-Concept Implementation

C code \(\rightarrow\) LLVM compiler \(\rightarrow\) LLVM bitcode

\[\rightarrow\]

Timing model \(\rightarrow\) PAGAI

\[\rightarrow\]

SMT-solving \(\rightarrow\) SMT clauses

\[\rightarrow\]

WCET
Proof-of-Concept Implementation

C code → LLVM compiler → LLVM bitcode:
- Unrolled loops

Timing model:
- 1 instruction = 1 cycle
- Each load and store requests a bus access
- Timing Composable

SMT-solving → SMT clauses → PAGAI → WCET

Claire Maiza
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>#LLVM instr.</th>
<th>#bus access</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs</td>
<td>Binary search</td>
<td>231</td>
<td>12</td>
</tr>
<tr>
<td>insertsort</td>
<td>Insertion sort on a reversed array</td>
<td>493</td>
<td>65</td>
</tr>
<tr>
<td>jfdctint</td>
<td>Discrete Cosine Transformation</td>
<td>2334</td>
<td>448</td>
</tr>
<tr>
<td>fdct</td>
<td>Fast Discrete Cosine Transform</td>
<td>2502</td>
<td>385</td>
</tr>
<tr>
<td>compressdata</td>
<td>Data compression program adopted from SPEC95</td>
<td>674</td>
<td>131</td>
</tr>
<tr>
<td>fly-by-wire</td>
<td>UAV fly-by-wire software</td>
<td>2815</td>
<td>515</td>
</tr>
</tbody>
</table>

**Table:** Benchmark description
Estimated WCET of the benchmarks with different configurations of the TDMA bus
## Evaluation

<table>
<thead>
<tr>
<th>Name</th>
<th>$&lt; 40, 20, 10 &gt;$</th>
<th>$&lt; 400, 100, 40 &gt;$</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs</td>
<td>0.45s</td>
<td>0.98s</td>
</tr>
<tr>
<td>insertsort</td>
<td>1.37s</td>
<td>6.56s</td>
</tr>
<tr>
<td>jfdctint</td>
<td>44.10s</td>
<td>48.54s</td>
</tr>
<tr>
<td>fdct</td>
<td>41.36s</td>
<td>34.57s</td>
</tr>
<tr>
<td>compressdata</td>
<td>4.66s</td>
<td>3.23s</td>
</tr>
<tr>
<td>fly-by-wire</td>
<td>28.78s</td>
<td><strong>149.01s</strong></td>
</tr>
</tbody>
</table>

### Analysis time

Claire Maiza
Summary

- SMT encodings for TDMA access
- Validation with small but relevant benchmarks

Why does it work?

- Feasible Path Analysis combined with the WCET computation
- Best-case of gain: All requests are within the TDMA slots
- Worst-case of gain: All requests have worst-case delay

Discussion

- Very good approach for TDMA
- No TT: analysis depends on other tasks = this kind of approach does not scale anymore
- Missing step: implementation on binary level
References


J. Henry, M. Asavoae, D. Monniaux, and C. Maïza. How to compute worst-case execution time by optimization modulo theory and a clever encoding of program semantics. LCTES ’14, NY, USA.