Computing WCET using Program Slicing and Real-Time Model-Checking

Franck Cassez*
[joint work with Jean-Luc Béchennec]
* Work supported by a Marie Curie Fellowship, FP7 of the EU

CNRS, IRCCyN
Nantes, France

November 15th, 2011
CISS, Aalborg
Outline of the talk

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
Worst-Case Execution-Time

Input data $d \in D$ → time($H, P, d$)

Hardware $H$

Program $P$

Listing 1. Binary Search Program

```
00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

Computing WCET using Slicing and Model-Checking

Franck Cassez (CISS, Aalborg, November 2011)
Worst-Case Execution-Time

Program P

Input data \(d \in D\) → time(\(H, P, d\))

Hardware H

\[ WCET(H, P) = \max_{d \in D} \text{time}(H, P, d) \]
Worst-Case Execution-Time

Program P

Hardware H

\[
\text{WCET}(H, P) \leq \text{WCET-UB}(H, P) \leq (1 + \varepsilon) \times \text{WCET}(H, P)
\]
Related Work & Existing Methods

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator

- easy to implement
- not exhaustive
- not safe: gives a lower bound

Tools: RapiTime (based on pWCET) and Mtime

Exhaustive - Static Analysis & Integer Linear Programming

1. Compute a control flow graph of $P$
2. Determine loop upper bounds
3. Build a weighted CFG
4. Solve an integer linear program

- harder to implement
- safe: gives an upper bound
- manual annotations
- algorithm is monolithic

Tools: Bound-T, OTAWA, TuBound, Chronos, SWEET and aiT (AbsInt)
Related Work & Existing Methods

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator
  - easy to implement
  - not exhaustive
  - not safe: gives a lower bound

Tools: RapiTime (based on pWCET) and Mtime

Exhaustive - Static Analysis & Integer Linear Programming

1. Compute a control flow graph of P
2. Determine loop upper bounds
3. Build a weighted CFG
4. Solve an integer linear program
  - harder to implement
  - safe: gives an upper bound
  - manual annotations
  - algorithm is monolithic

Tools: Bound-T, OTAWA, TuBound, Chronos, SWEET and aiT (AbsInt)
**METAMOC** [Master's Thesis 2009, WCET'2010]

**METAMOC = Modular Execution Time Analysis using Model Checking**
- Andreas E. Dalsgaard, Mads C. Olesen and Martin Toft, Aalborg Univ. Denmark
- Timed automata to model hardware
- Annotate programs with loop bounds
- Value analysis phase
- Loop unfolding
- Compute WCET using **UPPAAL**

**Results** (from [http://metamoc.dk/](http://metamoc.dk/))
- Mälardalen University benchmarks compiled with **O2** option
- Simple cache formal models
<table>
<thead>
<tr>
<th>ATML</th>
<th>ARM9 O2</th>
<th>ARM9 O2</th>
<th>ARM9 O2</th>
<th>ARM9 O2</th>
<th>ARM9 O2</th>
<th>ARM9 O2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization</td>
<td>O2</td>
<td>miss_writeback</td>
<td>miss_writeback</td>
<td>miss_writeback</td>
<td>Concrete, 128 lines/set, LRU</td>
<td>Concrete, LRU</td>
</tr>
<tr>
<td>Data-cache</td>
<td>-</td>
<td>No</td>
<td>No</td>
<td>Concrete, 128 lines/set</td>
<td>No</td>
<td>Concrete, 128 lines/set</td>
</tr>
<tr>
<td>Instr.-cache</td>
<td>-</td>
<td>No</td>
<td>No</td>
<td>Concrete, 128 lines/set</td>
<td>LRU</td>
<td>Concrete, 128 lines/set</td>
</tr>
<tr>
<td>Value-analysis</td>
<td>No</td>
<td></td>
<td>No</td>
<td>Concrete, 128 lines/set</td>
<td>LRU</td>
<td>Concrete, 128 lines/set</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs</td>
<td>146</td>
<td>3666</td>
<td>964</td>
<td>OOM</td>
<td>0:01:27</td>
<td>00</td>
</tr>
<tr>
<td>lsorru</td>
<td>0:03:66</td>
<td>0:01:09</td>
<td>0:01:06</td>
<td>0:01:27</td>
<td>OOM</td>
<td>640</td>
</tr>
<tr>
<td>cht</td>
<td>29572</td>
<td>188466</td>
<td>52481</td>
<td>OOM</td>
<td>0:00:01</td>
<td>01:37</td>
</tr>
<tr>
<td>compress</td>
<td>6:29:88</td>
<td>0:06:52</td>
<td>0:09:11</td>
<td>69486</td>
<td>OOM</td>
<td>0000</td>
</tr>
<tr>
<td>jrc</td>
<td>170959</td>
<td>204391</td>
<td>328310</td>
<td>Concrete, 128 lines/set, LRU</td>
<td>0:04:56+0:22:65</td>
<td>310:22</td>
</tr>
<tr>
<td>edn</td>
<td>345499</td>
<td>2093937</td>
<td>687463</td>
<td>Concrete, LRU</td>
<td>0:39:83</td>
<td>0000</td>
</tr>
<tr>
<td>expint</td>
<td>7583931</td>
<td>652215</td>
<td>31485</td>
<td>Concrete, LRU</td>
<td>420:71</td>
<td>0000</td>
</tr>
<tr>
<td>rac</td>
<td>905</td>
<td>11553</td>
<td>1206</td>
<td>Concrete, LRU</td>
<td>0:12:72</td>
<td>0000</td>
</tr>
<tr>
<td>fact</td>
<td>4806</td>
<td>344234</td>
<td>1908692</td>
<td>Concrete, LRU</td>
<td>0:17:30</td>
<td>0000</td>
</tr>
<tr>
<td>ficall</td>
<td>435</td>
<td>13434</td>
<td>632</td>
<td>Concrete, LRU</td>
<td>0:01:52</td>
<td>0000</td>
</tr>
<tr>
<td>fr</td>
<td>782494</td>
<td>131508</td>
<td>21354</td>
<td>Concrete, LRU</td>
<td>0:04:53</td>
<td>0000</td>
</tr>
<tr>
<td>insertd</td>
<td>2872</td>
<td>78078</td>
<td>43858</td>
<td>Concrete, LRU</td>
<td>0:02:44</td>
<td>0000</td>
</tr>
<tr>
<td>janne_complex</td>
<td>0:01:17</td>
<td>0:01:17</td>
<td>0:01:33</td>
<td>Concrete, LRU</td>
<td>0:01:92</td>
<td>0000</td>
</tr>
<tr>
<td>jdctnt</td>
<td>54383</td>
<td>294957</td>
<td>133059</td>
<td>Concrete, LRU</td>
<td>111:10</td>
<td>0000</td>
</tr>
<tr>
<td>matmul</td>
<td>525335</td>
<td>5630494</td>
<td>2637615</td>
<td>Concrete, LRU</td>
<td>712:09</td>
<td>0000</td>
</tr>
<tr>
<td>ns</td>
<td>0:01:79</td>
<td>0:02:20</td>
<td>0:03:43</td>
<td>0:07:09</td>
<td>OOM</td>
<td>0000</td>
</tr>
<tr>
<td>nschmer</td>
<td>3:04:11</td>
<td>10:2957</td>
<td>622904</td>
<td>Concrete, LRU</td>
<td>25:59:57</td>
<td>0000</td>
</tr>
<tr>
<td>prime</td>
<td>170806</td>
<td>6477669</td>
<td>470372</td>
<td>Concrete, LRU</td>
<td>3:37:70</td>
<td>0000</td>
</tr>
<tr>
<td>od</td>
<td>0:12:52</td>
<td>1:02:36</td>
<td>1:56:36</td>
<td>2:41:87</td>
<td>155:14</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>0:11:81</td>
<td>0:07:76</td>
<td>0:14:64</td>
<td>0:23:07</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td></td>
<td>3 errors / 19 benchmarks</td>
<td>0 errors / 21 benchmarks</td>
<td>1 errors / 21 benchmarks</td>
<td>2 errors / 21 benchmarks</td>
<td>3 errors / 21 benchmarks</td>
<td>4 errors / 21 benchmarks</td>
</tr>
<tr>
<td></td>
<td>Model checking fails: 3</td>
<td>Model checking fails: 1</td>
<td>Model checking fails: 1</td>
<td>Model checking fails: 2</td>
<td>Model checking fails: 3</td>
<td>Model checking fails: 4</td>
</tr>
<tr>
<td></td>
<td>Value analysis fails: 0</td>
<td>Value analysis fails: 0</td>
<td>Value analysis fails: 0</td>
<td>Value analysis fails: 0</td>
<td>Value analysis fails: 0</td>
<td>Value analysis fails: 0</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
### Table 6 Measured and Simulated Execution Times

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Benchmarks</th>
<th>aiT V1</th>
<th>aiT V2</th>
<th>aiT V3</th>
<th>Chronos V9</th>
<th>Chronos V10</th>
<th>Chronos V11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>adpcm</td>
<td>buffer</td>
<td>buffer</td>
<td>buffer</td>
<td>160891</td>
<td>183526</td>
<td>126258</td>
</tr>
<tr>
<td>2</td>
<td>cnt</td>
<td>19622</td>
<td>16853</td>
<td>7235</td>
<td>4792</td>
<td>5586</td>
<td>3515</td>
</tr>
<tr>
<td>3</td>
<td>compress</td>
<td>27308</td>
<td>19970</td>
<td>6824</td>
<td>5859</td>
<td>7504</td>
<td>4744</td>
</tr>
<tr>
<td>4</td>
<td>cover</td>
<td>10080</td>
<td>6778</td>
<td>4299</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>crc</td>
<td>buffer</td>
<td>buffer</td>
<td>buffer</td>
<td>22688</td>
<td>26861</td>
<td>18098</td>
</tr>
<tr>
<td>6</td>
<td>duff</td>
<td>6919</td>
<td>4610</td>
<td>1028</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>7</td>
<td>edn</td>
<td>838686</td>
<td>299734</td>
<td>buffer</td>
<td>87444</td>
<td>108973</td>
<td>62995</td>
</tr>
<tr>
<td>8</td>
<td>insertsorts</td>
<td>4720</td>
<td>3990</td>
<td>1770</td>
<td>897</td>
<td>1364</td>
<td>949</td>
</tr>
<tr>
<td>9</td>
<td>janne_complex</td>
<td>1294</td>
<td>827</td>
<td>359</td>
<td>185</td>
<td>454</td>
<td>356</td>
</tr>
<tr>
<td>10</td>
<td>matmunt</td>
<td>936602</td>
<td>438435</td>
<td>buffer</td>
<td>186899</td>
<td>185937</td>
<td>90834</td>
</tr>
<tr>
<td>11</td>
<td>ndes</td>
<td>401294</td>
<td>190530</td>
<td>buffer</td>
<td>65600</td>
<td>86639</td>
<td>53625</td>
</tr>
<tr>
<td>12</td>
<td>ns</td>
<td>73738</td>
<td>36097</td>
<td>buffer</td>
<td>65777</td>
<td>7568</td>
<td>4784</td>
</tr>
<tr>
<td>13</td>
<td>nsichneu</td>
<td>28328</td>
<td>18825</td>
<td>8052</td>
<td>6305</td>
<td>42966</td>
<td>40931</td>
</tr>
<tr>
<td>14</td>
<td>recursion</td>
<td>8318</td>
<td>7143</td>
<td>5096</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>statemate</td>
<td>2486</td>
<td>3810</td>
<td>1260</td>
<td>1120</td>
<td>6207</td>
<td>5898</td>
</tr>
</tbody>
</table>

N/A = not applicable.
buffer = Because of the buffer limitation, it is not possible to measure the WCETs.
Table 9 Overview of Functional and Service Quality of WCET Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Average Tightness</th>
<th>Benchmarks not Handled by the Tool</th>
<th>Benchmarks Analyzed</th>
<th>Benchmarks under Test</th>
<th>Average Service Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>aiT</td>
<td>7-8%</td>
<td>0</td>
<td>17</td>
<td>17</td>
<td>100%</td>
</tr>
<tr>
<td>Bound-T</td>
<td>N/A</td>
<td>4</td>
<td>13</td>
<td>17</td>
<td>76.5%</td>
</tr>
<tr>
<td>SWEET</td>
<td>N/A</td>
<td>2</td>
<td>15</td>
<td>17</td>
<td>88.2%</td>
</tr>
<tr>
<td>Chronos</td>
<td>81-89%</td>
<td>4</td>
<td>13</td>
<td>17</td>
<td>76.5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tool</th>
<th>Programs Analyzed Without Annotation</th>
<th>Programs Tasks under Test</th>
<th>Average Automation Rate</th>
<th>Complexity of Processor Supported*</th>
</tr>
</thead>
<tbody>
<tr>
<td>aiT</td>
<td>45</td>
<td>84</td>
<td>54%</td>
<td>Simple, Medium, Very Complex</td>
</tr>
<tr>
<td>Bound-T</td>
<td>13</td>
<td>51</td>
<td>26%</td>
<td>Simple, Medium</td>
</tr>
<tr>
<td>SWEET</td>
<td>15</td>
<td>17</td>
<td>88%</td>
<td>Medium</td>
</tr>
<tr>
<td>Chronos</td>
<td>12</td>
<td>51</td>
<td>24%</td>
<td>Configurable Simulated Processor</td>
</tr>
</tbody>
</table>

N/A = No measured WCET was available and no WCET tightness was available at this time.
* = The classification of the processors type is based on the challenge statement.

[WCET-06] Lili Tan
The Worst Case Execution Time Tool Challenge 2006: The External Test
Leveraging Applications of Formal Methods, pp 241-248, 2006
Our Contribution

Assumptions on binary program $P$:
- $\exists K \in \mathbb{N}, \ P \text{ terminates in less than } K \text{ steps for all input data}$
- references to stack via specialized register sp only
- references to memory cells do not depend on input data

Results
- Fully automatic computation of WCET
  computation of CFG (and stack size) and of WCET-equivalent program
- Modular method
  1. Program model
  2. Hardware model
  3. Analysis (computation of WCET)
- Comparison of computed WCET and actual WCET
  WCET Benchmarks from Mälardalen University
  measurements on real platform ARM920T
  WCET computed with UPPAAL

Our Contribution

Assumptions on binary program $P$:

- $\exists K \in \mathbb{N}, P$ terminates in less than $K$ steps for all input data
- references to stack via specialized register $sp$ only
- references to memory cells do not depend on input data

Results

- Fully automatic computation of WCET
  computation of CFG (and stack size) and of WCET-equivalent program
- Modular method
  1. Program model
  2. Hardware model
  3. Analysis (computation of WCET)
- Comparison of computed WCET and actual WCET

WCET Benchmarks from Mälardalen University
measurements on real platform ARM920T
WCET computed with UPPAAL

[Bec11] J.-L. Béchennec and F. Cassez,
Computation of WCET using Program Slicing and Real-Time Model-Checking,
Research report, IRCCyN/CNRS, Updated November 2011,
arXiv:1105.1633v3 [cs.SE].
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
The Fibonacci Program

```
00000000 <fib>:
0: e24dd020  sub  sp, sp, #32
4: e58d0004  str  r0, [sp, #4]
8: e3a03001  mov  r3, #1
c: e58d3010  str  r3, [sp, #16]
10: e3a03000  mov  r3, #0
14: e58d3014  str  r3, [sp, #20]
18: e3a03002  mov  r3, #2
1c: e58d300c  str  r3, [sp, #12]
20: ea00000a  50 <fib+0x50>
24: e59d3010  ldr  r3, [sp, #16]
28: e58d3018  str  r3, [sp, #20]
30: e59d3014  ldr  r3, [sp, #20]
34: e0823003  add  r3, r2, r3
38: e58d3010  str  r3, [sp, #16]
3c: e59d2010  ldr  r2, [sp, #16]
40: e59d3014  ldr  r3, [sp, #20]
44: e59d300c  ldr  r3, [sp, #12]
48: e2833001  add  r3, r3, #1
4c: e58d300c  str  r3, [sp, #12]
50: e59d2010  ldr  r2, [sp, #16]
54: e59d3004  ldr  r3, [sp, #4]
58: e1520003  cmp  r2, r3
5c: daffffff0  ble  24 <fib+0x24>
60: e59d3010  ldr  r3, [sp, #16]
64: e58d301c  str  r3, [sp, #28]
68: e59d301c  ldr  r3, [sp, #28]
6c: e1a00003  mov  r0, r3
70: e28dd020  add  sp, sp, #32
74: e12fff1e  bx  lr

00000078 <main>:
78: e52de004  push  {lr}
7c: e24dd00c  sub  sp, sp, #12
80: e3a03f4b  mov  r3, #300
84: e58d3004  str  r3, [sp, #4]
88: e59d0004  ldr  r0, [sp, #4]
8c: ebf0000f  bl  0 <fib>
90: e1a03000  mov  r3, r0
94: e1a00003  mov  r0, r3
98: e28dd00c  add  sp, sp, #12
9c: e49de004  pop  {lr}
a0: e12fff1e  bx  lr
```

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
The Fibonacci Program

Listing 1. Binary Search Program

ENTRY
120  STMDB SP!,{LR}
END
124  SUB SP,SP,#12
128  MOV R3,#300
132  STR R3,[SP,#4]
136  LDR R0,[SP,#4]
140  BL ... MOV R0,R3
112  ADD SP,SP,#32
116  BX LR
144  MOV R3,R0
148  MOV R0,R3
152  ADD SP,SP,#12
156  LDMIA SP!,{LR}
160  BX LR

Franck Cassez (CISS, Aalborg, November 2011)

Computing WCET using Slicing and Model-Checking
The Fibonacci Program

Listing 1. Binary Search Program

00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr

00000078 <main>:
78: e52de004 push {lr}
7c: e24dd00c sub sp, sp, #12
80: e3a03f4b mov r3, #300
84: e58d3004 str r3, [sp, #4]
88: e59d0004 ldr r0, [sp, #4]
8c: ebffffdb bl 0 <fib>
90: e1a03000 mov r3, r0
94: e1a00003 mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
**Target Architecture: ARM9xx family - ARM920T**

- **RISC processor**, 16 registers, memory load/store and multiple ldr/str
- Data and Instruction **Caches**
Pipeline of the ARM920T

Pipelining = split execution of instructions into simple stages

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline of the ARM920T

Pipelining = split execution of instructions into simple stages

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline of the ARM920T

Pipelining = split execution of instructions into simple stages

add r2,r2,#1
sub r3,r3,#2
ldr r1,[sp,#4]

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline Stalls

Data dependences between instructions

Next instruction is a target of a “branch” instruction

Summary:
- on ARM9xxx: no branch prediction
- because of stalls, optimal flow (1 instruction/cycle) can be slowed down
Pipeline Stalls

- **Data dependences between instructions**

```assembly
ldr r2,[sp,#4]
add r1,r2,#2
```

- **Next instruction is a target of a “branch” instruction**

```assembly
ble 32
```

**Summary:**
- on ARM9xxx: no branch prediction
- because of stalls, optimal flow (1 instruction/cycle) can be slowed down
Pipeline Stalls

- **Data dependences** between instructions

```assembly
ldr r2, [sp, #4]
add r1, r2, #2
```

- **Next instruction is a target of a “branch” instruction**

```assembly
ble 32
```

**Summary:**
- on ARM9xxx: no **branch prediction**
- because of **stalls**, optimal flow (1 instruction/cycle) can be slowed down
Caches

- Set 1
- Set 2
- Set $2^n$

Line size

- CacheRead(X)!
- CacheWrite(X)!
- HIT
- MISS
- Read/Write cached data
- Main Memory transfer cache update
Caches

The diagram illustrates the operation of a cache system with multiple sets. Each set contains a number of cache lines, denoted as $2^n$. The line size is represented horizontally. There are two primary operations:

- **CacheRead($X$)**: When a data item $X$ is requested, the cache checks if it is already present in any of the sets.
- **CacheWrite($X$)**: When writing data to the cache, the cache updates the relevant set.

The diagram shows a scenario where a request for $X$ hits the cache in Set 2 (HIT) and another request for $X$ misses the cache in Set $2^n$ (MISS). The cache update process involves transferring data from main memory to the cache.

- **HIT**: This occurs when the requested data is already in the cache, allowing for quick access.
- **MISS**: This occurs when the requested data is not in the cache, necessitating a transfer from main memory.

The diagram highlights the efficiency gained from caching, reducing the need for frequent main memory transfers.
Caches

- CacheRead(X)!
- CacheWrite(X)!

Set 1
Set 2
Set $2^n$

Line size

HIT
MISS

Read/Write cached data

Main Memory transfer cache update
Caches

- CacheRead(X)!
- CacheWrite(X)!

Set 1
Set 2
Set $2^n$

Line size

HIT
MISS

Read/Write cached data
Main Memory transfer cache update

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
What is really needed to compute Execution Time?

Listing 1. Binary Search Program

```
00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
What is really needed to compute Execution Time?

```
00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr

Listing 1. Binary Search Program
```

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
What is really needed to compute Execution Time?

```
00000000 <fib>:
  0: e24dd020 sub sp, sp, #32
  4: e58d0004 str r0, [sp, #4]
  8: e3a03001 mov r3, #1
   c: e58d3010 str r3, [sp, #16]
  10: e3a03000 mov r3, #0
  14: e58d3014 str r3, [sp, #24]
  18: e3a03002 mov r3, #1
  20: ea00000a b 50 <fib+0x50>
  24: e59d3010 ldr r3, [sp, #16]
  28: e58d3018 str r3, [sp, #20]
  30: e59d2010 ldr r2, [sp, #16]
  32: add r2, r2, #2
  34: e0823003 add r3, r2, r3
  38: e58d3004 str r3, [sp, #4]
  40: e59d0004 ldr r0, [sp, #4]
  44: e59d3014 ldr r3, [sp, #20]
  48: e2833001 add r3, r3, #1
  50: e58d300c ldr r3, [sp, #12]
  52: e59d3001 ldr r2, [sp, #12]
  54: e59d300c ldr r3, [sp, #4]
  58: e1520004 cmp r2, r3
  5c: dafffff0 ble 24 <fib+0x24>
  60: e59d3010 ldr r3, [sp, #16]
  64: e58d301c str r3, [sp, #28]
  68: e59d301c ldr r3, [sp, #28]
  6c: e1a00003 mov r0, r3
  70: e28dd00c add sp, sp, #12
  74: e12fff1e bx lr

00000078 <main>:
  78: e52de004 push {lr}
  7c: e24dd00c sub sp, sp, #12
  80: e3a03f4b mov r3, #300
  84: e58d3004 str r3, [sp, #4]
  88: e59d0004 ldr r0, [sp, #4]
  8c: ebffffdb bl 0 <fib>
  90: e1a03000 mov r3, r0
  94: e1a00003 mov r0, r3
  98: e28dd00c add sp, sp, #12
  9c: e49de004 pop {lr}
  a0: e12fff1e bx lr
```

32: add r2, r2, #2

\[\mathcal{L}(P) \subseteq \Sigma^*\]

\[\sum^* \rightarrow \mathbb{N}\]
What is really needed to compute Execution Time?

```asm
00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

Listing 1. Binary Search Program

```
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

```
not stall ∧ y ≥ 1
[32: add r2, r2, #2; Ø]
```

Reset x

```
CacheWrite!
[12: stm sp!, {r2, lr};{100, 104}]
```

Reset y

```
Σ* → N
```

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
What is really needed to compute Execution Time?

Listing 1. Binary Search Program

```
00000000 <fib>:
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
c: e58d3010 ... mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking  15 / 39
Modular Computation of WCET

Program $P$

Semantics

$\text{Aut}(P)$ generates $\mathcal{L}(P) \subseteq \Sigma^*$

Finite Automaton

Hardware $\text{Aut}(H)$ accepts $\ast$

Timed Automaton

$\text{WCET}(H,P)$

Real-Time Model-Checking
Modular Computation of WCET

Program $P$

Semantics

$\text{Aut}(P)$ generates $L(P) \subseteq \Sigma^*$

Finite Automaton

Hardware $H$

HDL, ...

$\text{Aut}(H)$ accepts $\Sigma^*$

Timed Automaton
Modular Computation of WCET

Program $P$

Semantics

$\text{Aut}(P)$ generates $\mathcal{L}(P) \subseteq \Sigma^*$

Finite Automaton

Hardware $H$

HDL, ...

$\text{Aut}(H)$ accepts $\Sigma^*$

Timed Automaton

Synchronization $\text{Aut}(P) \parallel \text{Aut}(H)$

Real-Time Model-Checking

$\text{WCET}(H,P)$
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced $\text{Aut}(P)$
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
**WCET-Equivalent Reduced Program**

- Two runs of $P$ can generate the same word in $\mathcal{L}(P)$
  - e.g., Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $\text{Aut}(P)$: 16 32-bit registers, stack, status bits
  - size of a state of $\text{Aut}(P)$: $16 \times 32 + |\text{stack}| \times 32 + 4$
- WCET depends on $\mathcal{L}(P)$

$$\text{if } \mathcal{L}(P') = \mathcal{L}(P) \text{ then } \text{WCET}(H,P) = \text{WCET}(H,P')$$

**WCET-equivalent Program**

$P'$ and $P$ are WCET-equivalent iff $\mathcal{L}(P') = \mathcal{L}(P)$.

**Compute a reduced WCET-equivalent $P'$ using Program Slicing**


*Program slicing.*


---
WCET-Equivalent Reduced Program

- Two runs of $P$ can generate the same word in $L(P)$
  - e.g., Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $\text{Aut}(P)$: 16 32-bit registers, stack, status bits
  - size of a state of $\text{Aut}(P)$: $16 \times 32 + |\text{stack}| \times 32 + 4$
- WCET depends on $L(P)$

$$\text{if } L(P') = L(P) \text{ then } \text{WCET}(H, P) = \text{WCET}(H, P')$$

---

**WCET-equivalent Program**

$P'$ and $P$ are WCET-equivalent iff $L(P') = L(P)$.

**Compute a reduced WCET-equivalent $P'$ using Program Slicing**

---

Program slicing.
WCET-Equivalent Reduced Program

- Two runs of $P$ can generate the same word in $L(P)$
  e.g., Fibonacci with initial values $u_0 = 0, u_1 = 1$ and $u_0 = 2, u_1 = 3$
- state of $Aut(P)$: 16 32-bit registers, stack, status bits
  size of a state of $Aut(P)$: $16 \times 32 + |\text{stack}| \times 32 + 4$
- WCET depends on $L(P)$

$$\text{if } L(P') = L(P) \text{ then } WCET(H, P) = WCET(H, P')$$

WCET-equivalent Program
$P'$ and $P$ are WCET-equivalent iff $L(P') = L(P)$.

Compute a reduced WCET-equivalent $P'$ using Program Slicing

Program slicing.
WCET-Equivalent Reduced Program

- Two runs of \( P \) can generate the same word in \( L(P) \)
  - e.g., Fibonacci with initial values \( u_0 = 0, u_1 = 1 \) and \( u_0 = 2, u_1 = 3 \)
- state of \( \text{Aut}(P) \): 16 32-bit registers, stack, status bits
  - size of a state of \( \text{Aut}(P) \): \( 16 \times 32 + |\text{stack}| \times 32 + 4 \)
- WCET depends on \( L(P) \)
  
  \[
  \text{if } L(P') = L(P) \text{ then } \text{WCET}(H, P) = \text{WCET}(H, P')
  \]

WCET-equivalent Program

\( P' \) and \( P \) are WCET-equivalent iff \( L(P') = L(P) \).

---

Program slicing.
WCET-Equivalent Reduced Program

- Two runs of P can generate the same word in \( \mathcal{L}(P) \)
e.g., Fibonacci with initial values \( u_0 = 0, u_1 = 1 \) and \( u_0 = 2, u_1 = 3 \)
- state of \( \text{Aut}(P) \): 16 32-bit registers, stack, status bits
  size of a state of \( \text{Aut}(P) \): \( 16 \times 32 + |\text{stack}| \times 32 + 4 \)
- WCET depends on \( \mathcal{L}(P) \)

\[
\text{if } \mathcal{L}(P') = \mathcal{L}(P) \text{ then } \text{WCET}(H, P) = \text{WCET}(H, P')
\]

WCET-equivalent Program

P’ and P are WCET-equivalent iff \( \mathcal{L}(P') = \mathcal{L}(P) \).

Program slicing.
Program Slicing [Weiser, 1984]

I = set of instructions in P

Slicing

- slice criterion: subset $I' \subseteq I$ and variables $\nu(i)$ for each $i \in I'$

4: str r0, [sp,#4] and variable sp

- Slice of $P = \text{sub-program } P'$ of $P$ satisfying (1) and (2)

  - given input $d \in D$,
    
    run of $P$ on $d$    $\varrho = (i_0, v_0) (i_1, v_1) \cdots (i_k, v_k) \cdots (i_n, v_n)$
    
    run of $P'$ on $d$    $\varrho' = (i'_0, v'_0) (i'_1, v'_1) \cdots (i'_k, v'_k) \cdots (i'_m, v'_m)$

  - projection: for a pair $(i, v)$, $\text{proj}(i, v) = \begin{cases} 
  \varepsilon & \text{if } i \not\in I' \\
  (i, \text{proj}_\nu(i)(v)) & \text{otherwise}
\end{cases}$

  - for sequences of pairs: $\text{proj}^*(\varepsilon) = \varepsilon$ and $\text{proj}^*(w:a) = \text{proj}^*(w) \cdot \text{proj}(a)$

1 on input $d \in D$, if $P$ terminates then $P'$ terminates

2 on input $d \in D$, $\text{proj}^*(\varrho) = \text{proj}^*(\varrho')$

- a sub-program $P'$ can be effectively computed (no optimal one)

  compute data dependences and control dependences
Program Slicing [Weiser, 1984]

I = set of instructions in P

Slicing

- slice criterion: subset $I' \subseteq I$ and variables $V(i)$ for each $i \in I'$
- Slice of $P = \text{sub-program } P'$ of $P$ satisfying (1) and (2)
  - given input $d \in D$,
    
    run of $P$ on $d$ \hspace{1cm} \varrho = (i_0, v_0) (i_1, v_1) \cdots (i_k, v_k) \cdots (i_n, v_n)$
    
    run of $P'$ on $d$ \hspace{1cm} \varrho' = (i'_0, v'_0) (i'_1, v'_1) \cdots (i'_l, v'_l) \cdots (i'_m, v'_m)$
  
  - projection: for a pair $(i, v)$, $\text{proj}(i, v) = \begin{cases} 
  \varepsilon & \text{if } i \not\in I' \\
  (i, \text{proj}_{V(i)}(v)) & \text{otherwise}
  \end{cases}$
  - for sequences of pairs: $\text{proj}^*(\varepsilon) = \varepsilon$ and $\text{proj}^*(w.a) = \text{proj}^*(w).\text{proj}(a)$

1. on input $d \in D$, if $P$ terminates then $P'$ terminates
2. on input $d \in D$, $\text{proj}^*(\varrho) = \text{proj}^*(\varrho')$

• a sub-program $P'$ can be effectively computed (no optimal one)
  compute data dependences and control dependences
Program Slicing [Weiser, 1984]

I = set of instructions in P

Slicing

- **slice criterion**: subset $I' \subseteq I$ and variables $V(i)$ for each $i \in I'$
- $4:\text{str } r0, [sp, \#4]$ and variable $sp$
- **Slice of $P$** = sub-program $P'$ of $P$ satisfying (1) and (2)
  - given input $d \in D$,
    - run of $P$ on $d$ $\varrho = (i_0, v_0) \ (i_1, v_1) \ \cdots \ (i_k, v_k) \ \cdots \ (i_n, v_n)$
    - run of $P'$ on $d$ $\varrho' = (i'_0, v'_0) \ (i'_1, v'_1) \ \cdots \ (i'_l, v'_l) \ \cdots \ (i'_m, v'_m)$
  - **projection**: for a pair $(i, v)$, $\text{proj}(i, v) = \begin{cases} \varepsilon & \text{if } i \not\in I' \\ (i, \text{proj}_{V(i)}(v)) & \text{otherwise} \end{cases}$
  - for sequences of pairs: $\text{proj}^*(\varepsilon) = \varepsilon$ and $\text{proj}^*(w.a) = \text{proj}^*(w) \cdot \text{proj}(a)$

1. on input $d \in D$, if $P$ terminates then $P'$ terminates
2. on input $d \in D$, $\text{proj}^*(\varrho) = \text{proj}^*(\varrho')$

- a sub-program $P'$ can be **effectively** computed (no optimal one)
- compute data dependences and control dependences
Computing a WCET-equivalent Slice

Assume \( \text{CFG of } P \) is known

Slice criterion \( C \)

- Each memory transfer instruction is in the criterion \( C \)
  
  \[
  32: \text{ldr } r2, [r1, \#4] \quad (32, r1)
  \]

- Each conditional instruction is in \( C \)
  
  \[
  36: \text{beq } 34, 96: \text{subeq } r3, r3, \#1
  \]

What's in the Slice?

1. Initially slice \( S = C \) (memory transfers and conditional instructions)
2. Add to \( S \) instructions and variables that define the values of vars in \( S \)
   
   e.g., \( 28: \text{add } r1, r3, \#1 \)
3. Add to \( S \) instructions and variables that influence the control flow
   
   e.g., “hidden” loop counters, variables used in comparisons
4. Repeat from (2) until fixpoint is reached

Key Result [Weiser 1984]

A slice can be automatically computed.
Computing a WCET-equivalent Slice

Assume CFG of P is known

Slice criterion $C$
- each memory transfer instruction is in the criterion $C$
  32: ldr r2, [r1, #4] (32, r1)
- each conditional instruction is in $C$
  36: beq 34, 96: subeq r3, r3, #1

What’s in the Slice?
1. initially slice $S = C$ (memory transfers and conditional instructions)
2. add to $S$ instructions and variables that define the values of vars in $S$
   e.g., 28: add r1, r3, #1
3. add to $S$ instructions and variables that influence the control flow
   e.g., “hidden” loop counters, variables used in comparisons
4. repeat from (2) until fixpoint is reached

Key Result [Weiser 1984]
A slice can be automatically computed.
Computing a WCET-equivalent Slice

Assume CFG of $P$ is known

Slice criterion $C$

- each memory transfer instruction is in the criterion $C$
  
  32: ldr r2, [r1, #4] (32, r1)

- each conditional instruction is in $C$
  
  36: beq 34, 96: subeq r3, r3, #1

What’s in the Slice?

1. initially slice $S = C$ (memory transfers and conditional instructions)
2. add to $S$ instructions and variables that define the values of vars in $S$
   
   e.g., 28: add r1, r3, #1
3. add to $S$ instructions and variables that influence the control flow
   
   e.g., “hidden” loop counters, variables used in comparisons
4. repeat from (2) until fixpoint is reached

Key Result [Weiser 1984]

A slice can be automatically computed.
Sliced Fibonacci Program
\{r0,r2,r3,stack_3020,stack_3028,stack_3052\}

\begin{verbatim}
00000000 <fib>:
  0: e24dd020 sub sp, sp, #32
  4: e58d0004 str r0, [sp, #4]
  8: e3a03001 mov r3, #1
 c: e58d3010 str r3, [sp, #16]
10: e3a03000 mov r3, #0
14: e58d3014 str r3, [sp, #20]
18: e3a03002 mov r3, #2
 20: ea00000a b 50 <fib+0x50>
 24: e59d3010 ldr r3, [sp, #16]
 28: e58d3014 ldr r3, [sp, #20]
 34: e0823003 add r3, r2, r3
 38: e58d3010 str r3, [sp, #16]
 44: e59d3004 ldr r0, [sp, #4]
 48: e2833001 add r3, r3, #1
 50: e58d300c cmp r2, r3
 54: e52de004 push {lr}
 58: e852de000 sub sp, sp, #32
 60: e24dd00c sub sp, sp, #12
 64: e3a0307b mov r3, #300
 68: e58d3004 str r3, [sp, #4]
 70: e1a00003 mov r0, r3
 72: e28d020 add sp, sp, #32
 74: e12fffe bx lr

00000000 <main>:
  78: e52de004 push {lr}
  7c: e24dd00c sub sp, sp, #12
  80: e3a03f4b mov r3, #300
  84: e58d3004 str r3, [sp, #4]
  88: e59d0004 ldr r0, [sp, #4]
  8c: ebffffdb bl 0 <fib>
  90: e1a00000 mov r3, r0
  94: e1a00003 mov r0, r3
  98: e28d020 add sp, sp, #12
  9c: e49de004 pop {lr}
a0: e12fffe bx lr
\end{verbatim}
Automatic Computation of CFG using Program Slicing
Handling Unknown Input Data

Input data are unknown: extended domain: $\mathcal{D}_\bot = \mathcal{D} \cup \{\bot\}$

Game: program vs outcomes of comparisons
Handling Unknown Input Data

Input data are unknown: extended domain: \( D_\perp = D \cup \{ \perp \} \)

Game: program vs outcomes of comparisons
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
Hardware Formal Models

Formal Models

- Timed Automata: automata extended with dense-time clocks

Hardware Specs?

- Data sheets
- Incomplete or sketchy

Bad formal models $\Rightarrow$ very bad WCET results

How can we build better models?

- Find a specialist in computer architecture
- Design programs to stress particular features of the hardware
- Compare actual execution-times with computed execution-times
- Refine formal model
Hardware Formal Models

Formal Models
- **Timed Automata**: automata extended with dense-time clocks

Hardware Specs?
- Data sheets
- Incomplete or sketchy

Bad formal models $\implies$ very bad WCET results

How can we build better models?
- Find a specialist in computer architecture
- Design programs to stress particular features of the hardware
- Compare actual execution-times with computed execution-times
- Refine formal model
Hardware Formal Models

Formal Models

- **Timed Automata**: automata extended with dense-time clocks

Hardware Specs?

- Data sheets
- Incomplete or sketchy

Bad formal models $\implies$ very bad WCET results

How can we build better models?

- Find a specialist in computer architecture
- Design programs to stress particular features of the hardware
- Compare actual execution-times with computed execution-times
- Refine formal model
Pipeline Model

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking 26 / 39
Instruction Cache & Main Memory

```plaintext
x <= CACHE_SPEED
CacheReadStart[num]?
PMT = is_in(m)?0:insert(m)
initialize?
initCache()
x == CACHE_SPEED
CacheReadEnd[num]!
PMT == 0
Hurry!
x = 0
MainMemEnd?
PMT--
PMT > 0
MainMemStart!
ICcachemiss++
x <= 6
x == 6
MainMemEndWB!
nb--

MainMemStart?
t == MAINMEMTRANS
MainMemEnd!
t <= MAINMEMTRANS
MainMemEndWB?
MainMemStartWB?
t == MAINMEMTRANS
MainMemEnd!
t = 0
MainMemStart?
```

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
Data Cache

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
Tool Chain

Program P

Slicing

Reduced $\text{Aut}(P')$ generates $L(P)$
Finite Automaton

Hardware H

HDL, ...

$\text{Aut}(H)$
Timed Automaton

Synchronization
$\text{Aut}(P')\parallel \text{Aut}(H)$

UPPAAL

$\text{WCET}(H,P)$

Slicing

A fast algorithm for finding dominators in a flowgraph.

Real-Time Model-Checking

UPPAAL in a Nutshell.
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work
Measuring Execution-Time on the ARM920T

```c
#define timerToCPUClockRatio 12

main ()
{
    int result;
    unsigned int start;
    unsigned int stop;

    start = timerGetValue(1);
    result = fib(300);
    stop = timerGetValue(1);
    printf("fib(300): %d, time=%lu\n", result,
            (stop-start)*timerToCPUClockRatio);
    while (1);
}
```

- **Embedded** hardware timer: 1/12th of processor clock frequency
- measurement error is ±24 cycles
- a program executing in ≥ 1200 cycles may be accurately measured less than 1% of measurement error
Compiled Program

00003214 <fib>:
0003214: e24dd020 sub sp, sp, #32
0003218: e58d0004 str r0, [sp, #4]
000321c: e3a03001 mov r3, #1
0004d64: ebfff99e bl 00033e4 <timerGetValue>
0004d68: e1a03000 mov r3, #3

Listing 3. Binary Search Program

000033e4 <timerGetValue>:
00033e4: e52de004 push {lr}
00033e8: e24dd014 sub sp, sp, #20
00033ec: e58d0004 str r0, [sp, #4]
00033f0: e59d0004 ldr r0, [sp, #4]
00033f8: ebfff9cb bl 000332c <timerGetRegisterAddress>
00033fc: e1a03000 mov r3, #3
0003400: e58d300c str r3, [sp, #12]
0003404: e59d300c ldr r3, [sp, #12]
0003408: e5933000 ldr r3, [r3]
000340c: e58d3008 str r3, [sp, #8]
0003410: e59d3008 ldr r3, [sp, #8]
0003414: e1a00003 mov r0, r3
0003418: e28dd014 add sp, sp, #20
000341c: e49df004 pop (pc)

0000332c <timerGetRegisterAddress>:
0000332c: e24dd008 sub sp, sp, #8
00003330: e58d0004 str r0, [sp, #4]
00003334: e58d1000 str r1, [sp]
00003338: e59d3004 ldr r3, [sp, #4]
0000333c: e3530001 cmp r3, #1
00003340: 1a000003 bne 0003354 <timerGetRegisterAddress+0x28>
00003344: e59d3000 ldr r3, [sp]
00003348: e2833602 add r3, r3, #2097152 ; 0x2000000
0000334c: e2833a02 add r3, r3, #8192 ; 0x2000
00003350: ea000002 b 0003360 <timerGetRegisterAddress+0x34>
00003354: e59d3000 ldr r3, [sp]
00003358: e2833602 add r3, r3, #2097152 ; 0x2000000
0000335c: e2833a03 add r3, r3, #12288 ; 0x3000
00003360: e1a00003 mov r0, r3
00003364: e28dd008 add sp, sp, #8
00003368: e12fff1e bx lr

0000336c <timerInit>:
0000336c: e52de004 bx lr

000033a8 <timerSetPrescaler>:
000033a8: e52de004 bx lr
Compiled Program

...  
0004d44: ebfff9a6  bl 00033e4 <timerGetValue>
0004d48: e1a03000  mov r3, r0
0004d4c: e58d3004  str r3, [sp, #4]
0004d50: e3a00f4b  mov r0, #300
0004d54: ebfff92e  bl 0003214  <fib>
0004d58: e1a03000  mov r3, r0
0004d5c: e58d3000  str r3, [sp]
0004d60: e3a00001  mov r0, #1
0004d64: ebfff99e  bl 00033e4 <timerGetValue>
0004d68: e1a03000  mov r3, #3
...
Interval in Execute Stage

execute?
\[ t=0, \text{DUR\_MAX\_INSTR= } \text{max\_dur()}, \]
\[ \text{DUR\_MIN\_INSTR= } \text{min\_dur()} \]

\[ t\leq \text{DUR\_MAX\_INSTR} \]

memory!
copy(me,me+1)

execute_completed!

\[ t\geq \text{DUR\_MIN\_INSTR} \]
\[ \text{&& } t\leq \text{DUR\_MAX\_INSTR} \]

de decode_completed?

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
## Experiments & Results [Benchmarks, Mälardalen Univ.]

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>(C–M) × 100</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>States Explored</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-00</td>
<td>74</td>
<td>1.7s/7181</td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td>47/131</td>
</tr>
<tr>
<td>fib-01</td>
<td>74</td>
<td>0.6s/2232</td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fib-02</td>
<td>74</td>
<td>0.3s/9710</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>janne-complex-00*</td>
<td>65</td>
<td>1.1s/38014</td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-01*</td>
<td>65</td>
<td>0.4s/14600</td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>janne-complex-02*</td>
<td>65</td>
<td>0.4s/13004</td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-01</td>
<td>238</td>
<td>1.7s/60418</td>
<td>2425</td>
<td>2409</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-02</td>
<td>238</td>
<td>3.2s/55285</td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td><strong>Single-Path Programs‡ with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-00</td>
<td>238</td>
<td>2.4s/85007</td>
<td>[1242,11800]</td>
<td>11448</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-00*</td>
<td>162</td>
<td>5m9s/1053230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-01*</td>
<td>162</td>
<td>43.7s/178054</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-00</td>
<td>374</td>
<td>2.7s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>0.8%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-01</td>
<td>374</td>
<td>1.0s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-02</td>
<td>374</td>
<td>5.3s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>3.4%</td>
<td>56/2512</td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-00</td>
<td>174</td>
<td>42.6s/1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-01</td>
<td>174</td>
<td>28s/1214673</td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-02</td>
<td>174</td>
<td>15s/655870</td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-00*</td>
<td>115</td>
<td>2.3s/76238</td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-01*</td>
<td>115</td>
<td>1s/27279</td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-02*</td>
<td>115</td>
<td>0.5s/11540</td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-00*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-01*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-02*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-00*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-01*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-02*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011) | Computing WCET using Slicing and Model-Checking

36 / 39
<table>
<thead>
<tr>
<th>Program</th>
<th>loc</th>
<th>UPPAAL Time</th>
<th>States Explored</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>((C-M)/M \times 100)</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-00</td>
<td>74</td>
<td>1.74s/74181</td>
<td></td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td>47/131</td>
</tr>
<tr>
<td>fib-01</td>
<td>74</td>
<td>0.61s/22332</td>
<td></td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fib-02</td>
<td>74</td>
<td>0.3s/9710</td>
<td></td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>jane-complex-00*</td>
<td>65</td>
<td>1.15s/38014</td>
<td></td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>jane-complex-01*</td>
<td>65</td>
<td>0.48s/14600</td>
<td></td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>jane-complex-02*</td>
<td>65</td>
<td>0.46s/13004</td>
<td></td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-01</td>
<td>238</td>
<td>1.67s/60418</td>
<td></td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-02</td>
<td>238</td>
<td>3.24s/55285</td>
<td></td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td><strong>Single-Path Programs†</strong> with MUL/MLA/SMULL instructions (instructions durations depend on data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-00</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>11448</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-00*</td>
<td>162</td>
<td>5m9s/10531230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>528684</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-02*</td>
<td>162</td>
<td>43.78s/178054</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>140664</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-00</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>12588</td>
<td>0.8%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-01</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>4668</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-02</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>16380</td>
<td>3.4%</td>
<td>56/2512</td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-00</td>
<td>174</td>
<td>42.6s/1421474</td>
<td></td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-01</td>
<td>174</td>
<td>28s/1214673</td>
<td></td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-02</td>
<td>174</td>
<td>15s/655870</td>
<td></td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-00*</td>
<td>115</td>
<td>2.3s/76238</td>
<td></td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-01*</td>
<td>115</td>
<td>1s/27279</td>
<td></td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-02*</td>
<td>115</td>
<td>0.5s/11540</td>
<td></td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-00*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td></td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-01*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td></td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-02*</td>
<td>91</td>
<td>11.5s/387292</td>
<td></td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-00*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td></td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-01*</td>
<td>497</td>
<td>11.3s/368719</td>
<td></td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-02*</td>
<td>497</td>
<td>29s/1030746</td>
<td></td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
### Experiments & Results

[Benchmarks, Mälardalen Univ.]

<table>
<thead>
<tr>
<th>Program</th>
<th>loc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>$(C-M) \times 100$</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>States Explored</td>
<td>Time</td>
<td>States Explored</td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-O0</td>
<td>74</td>
<td>1.74s/74181</td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td>47/131</td>
</tr>
<tr>
<td>fib-O1</td>
<td>74</td>
<td>0.61s/22332</td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fib-O2</td>
<td>74</td>
<td>0.3s/9710</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>janne-complex-O0*</td>
<td>65</td>
<td>1.15s/38014</td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-O1*</td>
<td>65</td>
<td>0.48s/14600</td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>janne-complex-O2*</td>
<td>65</td>
<td>0.46s/13004</td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-O1</td>
<td>238</td>
<td>1.67s/60418</td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-O2</td>
<td>238</td>
<td>3.24s/55285</td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td><strong>Single-Path Programs† with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>511584</td>
<td>528684</td>
<td>0.1%</td>
</tr>
<tr>
<td>matmult-O0*</td>
<td>162</td>
<td>5m9s/10531230</td>
<td>[502850,529250]</td>
<td>116444</td>
<td>140664</td>
<td>5.4%</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>162</td>
<td>43.78s/1780548</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>140664</td>
<td>5.4%</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[126999,12699]</td>
<td>12588</td>
<td>159/792</td>
<td>0.8%</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>25/325</td>
<td>7.0%</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>56/2512</td>
<td>3.4%</td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>42.6s/1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>28s/1214673</td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>15s/655870</td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>2.3s/76238</td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1s/27279</td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>0.5s/11540</td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011)  Computing WCET using Slicing and Model-Checking
# Experiments & Results

[Benchmarks, Mälardalen Univ.]

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>(\frac{(C-M)}{M} \times 100)</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>fb-00</td>
<td>74</td>
<td>1.74s/74181</td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td>47/131</td>
</tr>
<tr>
<td>fb-01</td>
<td>74</td>
<td>0.61s/22332</td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fb-02</td>
<td>74</td>
<td>0.3s/9710</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>janne-complex-00*</td>
<td>65</td>
<td>1.15s/38014</td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-01*</td>
<td>65</td>
<td>0.48s/14600</td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>janne-complex-02*</td>
<td>65</td>
<td>0.46s/13004</td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-01</td>
<td>238</td>
<td>1.67s/60418</td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-02</td>
<td>238</td>
<td>3.24s/55285</td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-Path Programs with MUL/MLA/SMULL instructions (instructions durations depend on data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-00</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-00*</td>
<td>162</td>
<td>5ms9s/10531230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>528684</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-02*</td>
<td>162</td>
<td>43.78s/1780548</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>140664</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-00</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>0.8%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-01</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-02</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>3.4%</td>
<td>56/2512</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple-Path Programs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-00</td>
<td>174</td>
<td>42.6s/1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-01</td>
<td>174</td>
<td>28s/1214673</td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-02</td>
<td>174</td>
<td>15s/655870</td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-00*</td>
<td>115</td>
<td>2.3s/76238</td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-01*</td>
<td>115</td>
<td>1s/27279</td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-02*</td>
<td>115</td>
<td>0.5s/11540</td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-00*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-01*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-02*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-00*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-01*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-02*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
### Experiments & Results

**[Benchmarks, Mälardalen Univ.]**

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>((C - M) / M \times 100)</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-00</td>
<td>74</td>
<td>1.74s/74181</td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td>47/131</td>
</tr>
<tr>
<td>fib-01</td>
<td>74</td>
<td>0.61s/22332</td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fib-02</td>
<td>74</td>
<td>0.3s/9710</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>janne-complex-00*</td>
<td>65</td>
<td>1.15s/38014</td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-01*</td>
<td>65</td>
<td>0.48s/14600</td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>janne-complex-02*</td>
<td>65</td>
<td>0.46s/13004</td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-O1</td>
<td>238</td>
<td>1.67s/60418</td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-O2</td>
<td>238</td>
<td>3.24s/55285</td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong> with MUL/MLA/SMULL instructions (instructions durations depend on data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-O0*</td>
<td>162</td>
<td>5m9s/10531230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>162</td>
<td>43.78s/1780548</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>0.8%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16/46,16938]</td>
<td>16380</td>
<td>3.4%</td>
<td>56/2512</td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>42.6s/1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>28s/1214673</td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-02</td>
<td>174</td>
<td>15s/655870</td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>2.3s/76238</td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1s/27279</td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>0.5s/11540</td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
<tr>
<td>Program</td>
<td>loc</td>
<td>UPPAAL Time</td>
<td>States Explored</td>
<td>Computed WCET (C)</td>
<td>Measured WCET (M)</td>
<td>( \frac{(C-M)}{M} \times 100 )</td>
</tr>
<tr>
<td>------------------</td>
<td>-----</td>
<td>-------------</td>
<td>----------------</td>
<td>------------------</td>
<td>------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><strong>Experiments &amp; Results</strong> [Benchmarks, Mälardalen Univ.]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Program</strong></td>
<td></td>
<td><strong>UPPAAL Time</strong></td>
<td><strong>States Explored</strong></td>
<td><strong>Computed WCET (C)</strong></td>
<td><strong>Measured WCET (M)</strong></td>
<td><strong>( \frac{(C-M)}{M} \times 100 )</strong></td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-O0</td>
<td>74</td>
<td>1.74s/74181</td>
<td></td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
</tr>
<tr>
<td>fib-O1</td>
<td>74</td>
<td>0.61s/22332</td>
<td></td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
</tr>
<tr>
<td>fib-O2</td>
<td>74</td>
<td>0.3s/9710</td>
<td></td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
</tr>
<tr>
<td>janne-complex-O0</td>
<td>65</td>
<td>1.15s/38014</td>
<td></td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
</tr>
<tr>
<td>janne-complex-O1</td>
<td>65</td>
<td>0.48s/14600</td>
<td></td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
</tr>
<tr>
<td>janne-complex-O2</td>
<td>65</td>
<td>0.46s/13004</td>
<td></td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
</tr>
<tr>
<td>fdct-O1</td>
<td>238</td>
<td>1.67s/60418</td>
<td></td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
</tr>
<tr>
<td>fdct-O2</td>
<td>238</td>
<td>3.24s/55285</td>
<td></td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
</tr>
<tr>
<td><strong>Single-Path Programs‡ with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>11558</td>
<td>0.1%</td>
</tr>
<tr>
<td>matmult-O0</td>
<td>162</td>
<td>5m9s/10531230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>528684</td>
<td>0.6%</td>
</tr>
<tr>
<td>matmult-O2</td>
<td>162</td>
<td>43.78s/1780548</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>140664</td>
<td>5.4%</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>12792</td>
<td>0.8%</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>4768</td>
<td>2.0%</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>16720</td>
<td>2.4%</td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>42.6s/1421474</td>
<td></td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>28s/1214673</td>
<td></td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>15s/655870</td>
<td></td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>2.3s/76238</td>
<td></td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1s/27279</td>
<td></td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>0.5s/11540</td>
<td></td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td></td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td></td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td></td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td></td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td></td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td></td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
</tr>
</tbody>
</table>

Franck Cassez (CISS, Aalborg, November 2011) Computing WCET using Slicing and Model-Checking
Outline

1. The Worst-Case Execution-Time Problem
2. Program & Architecture
3. Modular Computation of WCET
4. Computing a Reduced Aut(P)
5. Hardware Formal Models
6. Implementation
7. Experiments & Results
8. Conclusion & Future Work

Franck Cassez (CISS, Aalborg, November 2011)
Summary

Fully automatic computation of WCET

- Computation of CFG of binary programs + reduced program
  Program slicing
- Formal models of hardware (pipeline and caches)
  Identification of hardware features
- Computation of WCET as a reachability property
  Real-time model-checking with UPPAAL

Experiments to evaluate tightness of results

- Method to measure execution-times on ARM920T
- Evaluation on benchmarks from Mälardalen University, Sweden
- Over-approximation is less than 5%

Advantages of our method

- Modular
- Fully automatic
- Can easily accommodate new features
Summary

Fully automatic computation of WCET

- Computation of CFG of binary programs + reduced program
  Program slicing
- Formal models of hardware (pipeline and caches)
  Identification of hardware features
- Computation of WCET as a reachability property
  Real-time model-checking with UPPAAL

Experiments to evaluate tightness of results

- method to measure execution-times on ARM920T
- evaluation on benchmarks from Mälardalen University, Sweden
- over-approximation is less than 5%

Advantages of our method

- Modular
- Fully automatic
- Can easily accommodate new features
Summary

Fully automatic computation of WCET

- Computation of CFG of binary programs + reduced program
  Program slicing
- Formal models of hardware (pipeline and caches)
  Identification of hardware features
- Computation of WCET as a reachability property
  Real-time model-checking with UPPAAL

Experiments to evaluate tightness of results

- method to measure execution-times on ARM920T
- evaluation on benchmarks from Mälardalen University, Sweden
- over-approximation is less than 5%

Advantages of our method

- Modular
- Fully automatic
- Can easily accommodate new features
Ongoing and Future Work

New Features

- Processor speed changes
- For OS programs, model for interruptions’ arrivals

New Architectures

- models of PowerPC (multi-core)
- refine cache models

Enhanced Analysis (model-checking)

- Compute a witness trace that gives the WCET
- Refinement CEGAR
- Design a customized real-time model-checker taking advantage of particular features of the WCET problem
- reduce the reduced program
  - reduce number of paths to explore

Tool Release forthcoming (http://www.irccyn.fr/franck/wcet)

- Command line compiler from binary programs to UPPAAL
- Qt Interface
Ongoing and Future Work

New Features
- Processor speed changes
- For OS programs, model for interruptions’ arrivals

New Architectures
- models of PowerPC (multi-core)
- refine cache models

Enhanced Analysis (model-checking)
- Compute a witness trace that gives the WCET
- Refinement CEGAR
- Design a customized real-time model-checker
  - taking advantage of particular features of the WCET problem
- reduce the reduced program
  - reduce number of paths to explore

Tool Release forthcoming (http://www.irccyn.fr/franck/wcet)
- Command line compiler from binary programs to UPPAAL
- Qt Interface
Ongoing and Future Work

New Features
- Processor speed changes
- For OS programs, model for interruptions’ arrivals

New Architectures
- models of PowerPC (multi-core)
- refine cache models

Enhanced Analysis (model-checking)
- Compute a witness trace that gives the WCET
- Refinement CEGAR
- Design a customized real-time model-checker taking advantage of particular features of the WCET problem
- reduce the reduced program reduce number of paths to explore

Tool Release forthcoming (http://www.irccyn.fr/franck/wcet)
- Command line compiler from binary programs to UPPAAL
- Qt Interface
Ongoing and Future Work

New Features
- Processor speed changes
- For OS programs, model for interruptions’ arrivals

New Architectures
- models of PowerPC (multi-core)
- refine cache models

Enhanced Analysis (model-checking)
- Compute a witness trace that gives the WCET
- Refinement CEGAR
- Design a customized real-time model-checker
  taking advantage of particular features of the WCET problem
- reduce the reduced program
  reduce number of paths to explore

Tool Release forthcoming (http://www.irccyn.fr/franck/wcet)
- Command line compiler from binary programs to UPPAAL
- Qt Interface
References I


References II

**Uppaal 4.0.**

**pWCET a Toolset for automatic Worst-Case Execution Time Analysis of Real-Time Embedded Programs.**

**Timed Games for Computing WCET for Pipelined Processors with Caches.**

**Counterexample-guided abstraction refinement for symbolic model checking.**

**Web site.**

**A Simple, Fast Dominance Algorithm.**
References III

Metamoc: Modular execution time analysis using model checking.  

Modular execution time analysis using model checking.  
Master’s thesis, Department of Computer Science, Aalborg University, Denmark, 2009.

Worst-case execution-time analysis for embedded real-time systems.  

Analyzing the worst-case execution time by abstract interpretation of executable code.  

Finding dominators in practice.  

[17] Jan Gustafsson, Adam Betts, Andreas Ermedahl, and Björn Lisper.  
The Mälardalen WCET benchmarks – past, present and future.  
pages 137-147, Brussels, Belgium, July 2010. OCG.

*WCET 2008 - report from the tool challenge 2008.*


*Comparison of Implicit Path Enumeration and Model Checking Based WCET Analysis.*


*UPPAAL in a Nutshell.*


*A fast algorithm for finding dominators in a flowgraph.*

[21] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury.

*Chronos: A Timing Analyzer for Embedded Software.*
Special Issue on Experimental Software and Toolkit.
References V


References VI

Using Model Checking to Derive Loop Bounds of General Loops within ANSI-C Applications for Measurement Based WCET Analysis.  

[27] Tidorum Ltd.  
Bound-T time and stack analyser.  
http://www.tidorum.fi/bound-t/.

Program slicing.  

The Worst-Case Execution-Time Problem - Overview of Methods and Survey of Tools.  