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

Franck Cassez
[joint work with Jean-Luc Béchennec]

CNRS & Marie Curie Fellow
IRCCyN, Nantes, France

June 3rd, 2011
NICTA, Sydney
Outline of the talk

1. Introduction
2. Program & Architecture
3. Computation of WCET
4. Hardware Model
5. Implementation
6. Experiments & Results
7. Conclusion & Future Work
Outline

1 Introduction

2 Program & Architecture

3 Computation of WCET

4 Hardware Model

5 Implementation

6 Experiments & Results

7 Conclusion & Future Work
Worst-Case Execution-Time

- Program $P$ runs on hardware $H$
  - $H$: processor, main memory, caches, ...
- Time = number of processor cycles
- Input data of $P$ range over finite domain $D$
- Given $d \in D$, $\text{time}(H, P, d)$ is the execution-time of $P$ for $d$ on $H$
- Worst-Case Execution-Time (WCET)

\[
\text{WCET}(H, P) = \max_{d \in D} \text{time}(H, P, d)
\]

Why do we need tight WCET (upper) bounds?

- Real-time systems = set of programs (tasks) with deadlines
- Processor is shared: scheduling
- Requires upper bounds on programs durations

\[
\text{WCET}(H, P) \leq \text{WCET-UB}(H, P) \leq (1 + \varepsilon) \times \text{WCET}(H, P)
\]
Worst-Case Execution-Time

- Program $P$ runs on hardware $H$
  $H$: processor, main memory, caches, ...
- Time = number of processor cycles
- Input data of $P$ range over finite domain $D$
- Given $d \in D$, $\text{time}(H, P, d)$ is the execution-time of $P$ for $d$ on $H$
- Worst-Case Execution-Time (WCET)

\[
\text{WCET}(H, P) = \max_{d \in D} \text{time}(H, P, d)
\]

Why do we need tight WCET (upper) bounds?

- Real-time systems = set of programs (tasks) with deadlines
- Processor is shared: scheduling
- Requires upper bounds on programs durations

\[
\text{WCET}(H, P) \leq \text{WCET-UB}(H, P) \leq (1 + \varepsilon) \times \text{WCET}(H, P)
\]
Main Difficulties for Computing WCET

- **Binary program**
  unstructured, control flow graph unknown at compile time

- **Take into account all possible input data**
  Finite domain but still huge number of input data …

- **Complex architecture**: pipelining, caching
  even for in-order pipeline, timing is complex

- **Formal specification** of the hardware is partial or unavailable

Sources of over-approximations

- Program model contains infeasible paths
- Hardware model too abstract
  model of cache is “always miss”
  model of pipeline mistakenly “stalls”
  timing of instruction’s duration is over-approximated
Main Difficulties for Computing WCET

- **Binary program**
  - unstructured, control flow graph unknown at compile time
- **Take into account all possible input data**
  - Finite domain but still huge number of input data...
- **Complex architecture:** pipelining, caching
  - even for in-order pipeline, timing is complex
- **Formal specification** of the hardware is partial or unavailable

Use abstract models of programs and hardware

Sources of over-approximations

- Program model contains infeasible paths
- hardware model too abstract
  - model of cache is “always miss”
  - model of pipeline mistakenly “stalls”
  - timing of instruction’s duration is over-approximated
Main Difficulties for Computing WCET

- **Binary program**
  unstructured, control flow graph unknown at compile time
- **Take into account all possible input data**
  Finite domain but still huge number of input data ...
- **Complex architecture**: pipelining, caching
  even for in-order pipeline, timing is complex
- **Formal specification of the hardware is partial or unavailable**

Use abstract models of programs and hardware

Sources of over-approximations

- **Program model contains infeasible paths**
- **hardware model too abstract**
  model of cache is “always miss”
  model of pipeline mistakenly “stalls”
  timing of instruction’s duration is over-approximated
Related Work & Existing Methods

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator

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

Tools: RapiTime [29] (based on pWCET [9]) and Mtime [30]

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 [31], OTAWA [7], TuBound [28], Chronos [24], SWEET [16] and most advanced is aiT [4, 17] (AbsInt)
Related Work & Existing Methods

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator

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

Tools: RapiTime [29] (based on pWCET [9]) and Mtime [30]

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 [31], OTAWA [7], TuBound [28], Chronos [24], SWEET [16] and most advanced is aiT [4, 17] (AbsInt)
Table 5

<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>dpcm</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>matmult</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.

---

**WCET Challenge 2006 - Results**

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>dpcm</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>matmult</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 7 \( WCET \) Tightness

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Benchmarks</th>
<th>aiT 1</th>
<th>aiT 2</th>
<th>aiT 3</th>
<th>Chronos 1</th>
<th>Chronos 2</th>
<th>Chronos 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>N / A</td>
<td>6.5%</td>
<td>89.48%</td>
<td>151.35%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2c</td>
<td>3.2%</td>
<td>1.19%</td>
<td>1.95%</td>
<td>2.17%</td>
<td>15.25%</td>
<td>53.66%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>compress</td>
<td>37.58%</td>
<td>1.55%</td>
<td>38.64%</td>
<td>0.24%</td>
<td>289.33%</td>
<td>500.48%</td>
</tr>
<tr>
<td>4c</td>
<td>cover</td>
<td>3.69%</td>
<td>0.03%</td>
<td>16.45%</td>
<td>N / A</td>
<td>N / A</td>
<td>N / A</td>
</tr>
<tr>
<td>5</td>
<td>crc</td>
<td>N / A</td>
<td>N / A</td>
<td>N / A</td>
<td>110.62%</td>
<td>130.26%</td>
<td>194.37%</td>
</tr>
<tr>
<td>6d</td>
<td>uff</td>
<td>4.05%</td>
<td>0.04%</td>
<td>31.81%</td>
<td>N / A</td>
<td>N / A</td>
<td>N / A</td>
</tr>
<tr>
<td>7e</td>
<td>dnn</td>
<td>10.54%</td>
<td>2.72%</td>
<td>N / A</td>
<td>2.24%</td>
<td>4.26%</td>
<td>41.33%</td>
</tr>
<tr>
<td>8</td>
<td>insertsorts</td>
<td>3.18%</td>
<td>0.05%</td>
<td>3.84%</td>
<td>0.45%</td>
<td>13.56%</td>
<td>31.19%</td>
</tr>
<tr>
<td>9j</td>
<td>anne</td>
<td>2.7%</td>
<td>0.24%</td>
<td>6.69%</td>
<td>2.16%</td>
<td>76.21%</td>
<td>121.63%</td>
</tr>
<tr>
<td>10</td>
<td>matmult</td>
<td>2.15%</td>
<td>N / A</td>
<td>N / A</td>
<td>0.0%</td>
<td>3.05%</td>
<td>31.59%</td>
</tr>
<tr>
<td>11</td>
<td>ndes</td>
<td>12.97%</td>
<td>2.06%</td>
<td>N / A</td>
<td>1.61%</td>
<td>24.18%</td>
<td>60.22%</td>
</tr>
<tr>
<td>12</td>
<td>ns</td>
<td>2.68%</td>
<td>5.39%</td>
<td>N / A</td>
<td>24.66%</td>
<td>32.02%</td>
<td>81.35%</td>
</tr>
<tr>
<td>13</td>
<td>nsichneu</td>
<td>5.34%</td>
<td>0.01%</td>
<td>3.42%</td>
<td>115.84%</td>
<td>127.87%</td>
<td>138.27%</td>
</tr>
<tr>
<td>14</td>
<td>recursion</td>
<td>21.13%</td>
<td>4.31%</td>
<td>8.46%</td>
<td>N / A</td>
<td>N / A</td>
<td>N / A</td>
</tr>
<tr>
<td>15</td>
<td>statemate</td>
<td>5.39%</td>
<td>0.05%</td>
<td>2.7%</td>
<td>79.2%</td>
<td>160.75%</td>
<td>132.02%</td>
</tr>
</tbody>
</table>

N/A = not applicable.

### Table 8 Usability Assessment: Taking into account both the Mälardalen and the PapaBench Benchmark Programs

<table>
<thead>
<tr>
<th>Tool</th>
<th>Average Tightness</th>
<th>Warning of Accomplishment of Acceptability</th>
<th>Intended Tasks</th>
<th>Time Analysis</th>
<th>Annotation Errors</th>
<th>Annotation Hint, Links, User manual</th>
<th>Time Run messages, Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>aiT</td>
<td>7-8%</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Acceptable</td>
<td>Hints, Links, User manual</td>
<td>No case was founded</td>
</tr>
<tr>
<td>Bound-T</td>
<td>N/A</td>
<td>Generally acceptable</td>
<td>Acceptable</td>
<td>Generally</td>
<td>Mostly acceptable</td>
<td>User manual</td>
<td>No case founded</td>
</tr>
<tr>
<td>SWEET</td>
<td>N/A</td>
<td>Generally acceptble</td>
<td>Acceptable</td>
<td>Generally</td>
<td>Mostly acceptable</td>
<td>User manual</td>
<td>No case founded</td>
</tr>
<tr>
<td>Chronos</td>
<td>81-89%</td>
<td>Generally acceptable</td>
<td>Acceptable</td>
<td>Good</td>
<td>Good</td>
<td>User manual</td>
<td>Good</td>
</tr>
</tbody>
</table>

N/A = Chronos GUI could not be tested in the challenge.

### 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

Our Contribution

Assumptions on binary program $P$:

- termination of $P$ does not depend on input data
- $P$ always terminates

Results

- Fully automatic computation of WCET
  - computation of CFG (and stack size)
  - computation 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 $P$:
- termination of $P$ does not depend on input data
- $P$ always terminates

Results

- **Fully automatic computation of WCET**
  - computation of CFG (and stack size)
  - computation 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


Outline

1 Introduction
2 Program & Architecture
3 Computation of WCET
4 Hardware Model
5 Implementation
6 Experiments & Results
7 Conclusion & Future Work
The Fibonacci Program

```
ENTRY
120  stmdb sp!,{lr}
124  sub sp,sp,#32
128  mov r3,#300
132  str r3,[sp,#4]
136  ldr r0,[sp,#4]
140  bl 0
0  sub sp,sp,#32
4  str r0,[sp,#4]
8  ldr r2,[sp,#12]
12  str r3,[sp,#16]
16  mov r3,#0
20  str r3,[sp,#20]
24  mov r3,#2
28  str r3,[sp,#12]

Franck Cassez (NICTA, Sydney, June 2011) Computing WCET using Slicing and Model-Checking
```

```
The Fibonacci Program

```assembly
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  b      50 <fib+0x50>
24: e59d3010  ldr   r3, [sp, #16]
28: e58d3018  str   r3, [sp, #24]
2c: e59d2010  ldr   r2, [sp, #16]
30: e59d3014  ldr   r3, [sp, #20]
34: e0823003  add   r3, r2, r3
38: e58d3010  str   r3, [sp, #16]
3c: e59d3018  ldr   r3, [sp, #24]
40: e58d3014  str   r3, [sp, #20]
44: e59d300c  ldr   r3, [sp, #12]
48: e2833001  add   r3, r3, #1
4c: e58d300c  str   r3, [sp, #12]
50: e59d200c  ldr   r2, [sp, #12]
54: e59d3004  ldr   r3, [sp, #4]
58: e1520003  cmp   r2, r3
5c: dafffff0  ble   24 <fib+0x24>
```

Listing 1. Binary Search Program

Franck Cassez (NICTA, Sydney, June 2011)
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 0 <fib>
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

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: ebfffffd      bl      0 <fib>
90: e1a03000      mov     r3, r0
94: e1a00003      mov     r0, r3
98: e28dd00c      add     sp, sp, #12
9c: e49de004      pop     {lr}
a0: e12ffff1e      bx      lr
```

Franck Cassez (NICTA, Sydney, June 2011)  Computing WCET using Slicing and Model-Checking  11 / 47
Target Architecture: ARM920T
- RISC processor, 16 registers, memory load/store and multiple ldr/str
- Data and Instruction Caches
Pipeline of the ARM920T

- **Split** execution of instructions into simple stages
- ARM9xxx, 5 stages:
  1. F: Fetch instruction from memory
  2. D: Decode instruction
  3. E: Execute instruction
  4. M: Memory transfers
  5. W: Writeback registers' values
- concurrent execution of stages

### Code Examples
- `add r2, r2, #1`
- `sub r3, r3, #2`
- `ldr r1, [sp, #4]`

On average one cycle per instruction
Pipeline of the ARM920T

- **Split execution of instructions into simple stages**
- **ARM9xxx, 5 stages:**
  1. F: Fetch instruction from memory
  2. D: Decode instruction
  3. E: Execute instruction
  4. M: Memory transfers
  5. W: Writeback registers' values

- **concurrent execution of stages**

On average one cycle per instruction
Pipeline Stalls

- **Data dependences between instructions**

  - Instruction 1: `ldr r2, [sp, #4]`
  - Instruction 2: `add r1, r2, #2`

  ![Instruction Timeline]

  - `F D E M W`: Instruction 1
  - `F D` (stall): Instruction 2
  - Cycle: `jl j+2 j+3 j+4 j+5 j+6`

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

Summary:

- on ARM9xxx: no branch prediction
- because of stalls, optimal flow can be slowed down
Pipeline Stalls

- **Data dependences between instructions**

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

Next instruction is a target of a “branch” instruction

```
ble 32
```

Summary:
- on ARM9xxx: no branch prediction
- because of stalls, optimal flow can be slowed down
Pipeline Stalls

- **Data dependences between instructions**

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

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

`ble 32`

**Summary:**
- on ARM9xxx: no branch prediction
- because of stalls, optimal flow can be slowed down
Caches

**Cache Read**

1. read data at address \( a \) is requested
2. if \( a \in \text{Set}(a) \) return cached \( \text{val}(a) \) \hspace{1cm} \text{HIT}
3. otherwise fetch \( a \) from main memory, add to \( \text{Set}(a) \) and return \( \text{val}(a) \) \hspace{1cm} \text{MISS}

**Case 3:** if cache is full, first select a line \( \in \text{Set}(a) \) to clear

**Selection strategy = Replacement Policy**

**Cache Write**

1. write to data at address \( a \) is requested
2. if \( a \in \text{Set}(a) \) update cached \( \text{val}(a) \) \hspace{1cm} \text{HIT}
3. otherwise fetch \( a \) from main memory, add to \( \text{Set}(a) \) and update \( \text{val}(a) \) \hspace{1cm} \text{MISS}

Write back or write through

---

**ARM920T:** 16Kbytes
64-way associative cache
8 sets, 64 lines/set
line size 8 (4-byte) words
FIFO replacement
### Caches

**Cache Read**

1. read data at address \( a \) is requested
2. if \( a \in \text{Set}(a) \) return cached \( \text{val}(a) \) \( \text{HIT} \)
3. otherwise fetch \( a \) from main memory, add to \( \text{Set}(a) \) and return \( \text{val}(a) \) \( \text{MISS} \)

**Case 3:** if cache is full, first select a line \( \in \text{Set}(a) \) to clear

**Selection strategy = Replacement Policy**

**Cache Write**

1. write to data at address \( a \) is requested
2. if \( a \in \text{Set}(a) \) update cached \( \text{val}(a) \) \( \text{HIT} \)
3. otherwise fetch \( a \) from main memory, add to \( \text{Set}(a) \) and update \( \text{val}(a) \) \( \text{MISS} \)

Write back or write through

---

**ARM920T:** 16Kbytes
64-way associative cache
8 sets, 64 lines/set
line size 8 (4-byte) words
FIFO replacement

Franck Cassez (NICTA, Sydney, June 2011)
**Caches**

**Cache Read**

1. read data at address $a$ is requested
2. if $a \in \text{Set}(a)$ return cached $\text{val}(a)$ \text{HIT}
3. otherwise fetch $a$ from main memory, add to $\text{Set}(a)$ and return $\text{val}(a)$ \text{MISS}

**Case 3:** if cache is full, first select a line $\in \text{Set}(a)$ to clear

**Selection strategy = Replacement Policy**

**Cache Write**

1. write to data at address $a$ is requested
2. if $a \in \text{Set}(a)$ update cached $\text{val}(a)$ \text{HIT}
3. otherwise fetch $a$ from main memory, add to $\text{Set}(a)$ and update $\text{val}(a)$ \text{MISS}

**Write back or write through**
Caches

**Cache Read**
1. read data at address $a$ is requested
2. if $a \in \text{Set}(a)$ return cached $\text{val}(a)$ \[ \text{HIT} \]
3. otherwise fetch $a$ from main memory, add to $\text{Set}(a)$ and return $\text{val}(a)$ \[ \text{MISS} \]

**Case 3:** if cache is full, first select a line $\in \text{Set}(a)$ to clear

**Selection strategy = Replacement Policy**

**Cache Write**
1. write to data at address $a$ is requested
2. if $a \in \text{Set}(a)$ update cached $\text{val}(a)$ \[ \text{HIT} \]
3. otherwise fetch $a$ from main memory, add to $\text{Set}(a)$ and update $\text{val}(a)$ \[ \text{MISS} \]

**Write back or write through**

ARM920T: 16Kbytes
64-way associative cache
8 sets, 64 lines/set
line size 8 (4-byte) words
FIFO replacement
Outline

1. Introduction
2. Program & Architecture
3. Computation of WCET
4. Hardware Model
5. Implementation
6. Experiments & Results
7. Conclusion & Future Work
What is really needed to compute WCET?

Execution-time depends on
- instruction type and location, data dependences, branching
  
  32: \texttt{add r2,r2,#2}  
  36: \texttt{mov r1,r2}  
  64: \texttt{cmp r4,#0}  

- addresses referenced by an instruction
  
  4: \texttt{ldr r2,[sp,#4]}  
  12: \texttt{stm sp!,\{r2,lr\}}

Programs as language generators
- Program \(P\): finite set of instructions and memory addresses
- Alphabet \(\Sigma\): pairs (instruction; memory addresses)
  
  \((32:\texttt{add r2,r2,#2}; \emptyset)\)  
  \((12:\texttt{stm sp!,\{r2,lr\}};\{100,104\})\) if val(sp) = 100

- \(P\) generates a language \(L(P)\)

Hardware as language acceptors
- Hardware \(H\) (pipeline, caches) is an automaton with timing constraints
- State of \(H\): current state of pipeline, caches
- \(H\) can be viewed as a function \(\text{time}_H : \Sigma^* \rightarrow \mathbb{N}\)
- \(\text{time}_H(s_0,w)\): value associated with \(w\) when starting from \(H\)-state \(s_0\)

\[
\text{WCET}(H,P) = \max \{ \text{time}_H(s_0,w) \mid w \in L(P) \} 
\]
What is really needed to compute WCET ?

Execution-time depends on
- instruction type and location, data dependences, branching
  32: add r2, r2, #2  
  36: mov r1, r2  
  64: cmp r4, #0
- addresses referenced by an instruction
  4: ldr r2, [sp, #4]  
  12: stm sp!, {r2, lr}

Programs as language generators
- Program P: finite set of instructions and memory addresses
- Alphabet Σ: pairs (instruction; memory addresses)
  (32: add r2, r2, #2; ∅)
  (12: stm sp!, {r2, lr}; {100, 104}) if val(sp) = 100
- P generates a language $L(P)$

Hardware as language acceptors
- Hardware H (pipeline, caches) is an automaton with timing constraints
- State of H: current state of pipeline, caches
- H can be viewed as a function $time_H : \Sigma^* \rightarrow \mathbb{N}$
- $time_H(s_0, w)$: value associated with w when starting from H-state $s_0$

$WCET(H, P) = \max \{ time_H(s_0, w) \mid w \in L(P) \}$
What is really needed to compute WCET?

Execution-time depends on
- instruction type and location, data dependences, branching
  32: add r2, r2, #2  36: mov r1, r2  64: cmp r4, #0
- addresses referenced by an instruction
  4: ldr r2, [sp, #4]  12: stm sp!, {r2, lr}

Programs as language generators
- Program P: finite set of instructions and memory addresses
- Alphabet Σ: pairs (instruction; memory addresses)
  (32: add r2, r2, #2; 0)
  (12: stm sp!, {r2, lr}; {100, 104}) if val(sp) = 100
- P generates a language $L(P)$

Hardware as language acceptors
- Hardware H (pipeline, caches) is an automaton with timing constraints
- State of H: current state of pipeline, caches
- H can be viewed as a function $\text{time}_H : \Sigma^* \rightarrow \mathbb{N}$
- $\text{time}_H(s_0, w)$: value associated with w when starting from H-state $s_0$

$\text{WCET}(H, P) = \max \{ \text{time}_H(s_0, w) \mid w \in L(P) \}$
What is really needed to compute WCET?

Execution-time depends on:
- instruction type and location, data dependences, branching
  - 32: add r2, r2, #2
  - 36: mov r1, r2
  - 64: cmp r4, #0
- addresses referenced by an instruction
  - 4: ldr r2, [sp, #4]
  - 12: stm sp!, {r2, lr}

Programs as language generators:
- Program P: finite set of instructions and memory addresses
- Alphabet \( \Sigma \): pairs (instruction; memory addresses)
  - \((32: \text{add } r2, r2, #2; \emptyset)\)
  - \((12: \text{stm } sp!, \{r2, lr\}; \{100, 104\})\) if \(\text{val}(sp) = 100\)
- P generates a language \(\mathcal{L}(P)\)

Hardware as language acceptors:
- Hardware \(H\) (pipeline, caches) is an automaton with timing constraints
- State of \(H\): current state of pipeline, caches
- \(H\) can be viewed as a function \(\text{time}_H : \Sigma^* \rightarrow \mathbb{N}\)
- \(\text{time}_H(s_0, w)\): value associated with \(w\) when starting from \(H\)-state \(s_0\)

\[
\text{WCET}(H, P) = \max \{ \text{time}_H(s_0, w) \mid w \in \mathcal{L}(P) \}
\]
Modular Computation of WCET

Program $P$

Semantics

$\text{Aut}(P)$

generates $L(P) \subseteq \Sigma^*$

Finite Automaton

Hardware $H$

$\text{Aut}(H)$

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

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 $L \subseteq \Sigma^*$

Timed Automaton
Modular Computation of WCET

Program $P$

Semantics

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

Finite Automaton

Hardware $H$

HDL, ...

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

Timed Automaton

Synchronization

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

Real-Time

Model-Checking

$\text{WCET}(H,P)$
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 $P'$
**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** $\mathcal{L}(P)$

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

---

**Compute a reduced WCET-equivalent $P'$**

**WCET-equivalent Program**

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

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

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

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

WCET-equivalent Program

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

Compute a reduced WCET-equivalent $P'$
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 $\mathcal{L}(P)$

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

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

\text{Compute a reduced WCET-equivalent } P'$
Program Slicing, M. Weiser [32]

$I = \text{set of instructions in } P$

Slicing

- **slice criterion**: subset $I' \subseteq I$ and variables $V(i)$ 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 $(i, v)$, $\text{proj}(i, v) = \begin{cases} \varepsilon & \text{if } i \not\in I' \\ (i, \text{proj}_{V(i)}(v)) & \text{otherwise} \end{cases}$
  - 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, M. Weiser \[32\]

$I = \text{set of instructions in } \mathbb{P}$

Slicing

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

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

- **Slice of $\mathbb{P} = \text{sub-program } \mathbb{P}'$ of $\mathbb{P}$ satisfying (1) and (2)**
  
  ▶ given input $d \in \mathbb{D}$,
  
  \begin{align*}
  \text{run of } \mathbb{P} \text{ on } d & = (i_0, v_0) (i_1, v_1) \cdots (i_k, v_k) \cdots (i_n, v_n) \\
  \text{run of } \mathbb{P}' \text{ on } d & = (i'_0, v'_0) (i'_1, v'_1) \cdots (i'_l, v'_l) \cdots (i'_m, v'_m)
  \end{align*}

  ▶ 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: \[\text{proj}^*(\varepsilon) = \varepsilon \text{ and } \text{proj}^*(w.a) = \text{proj}^*(w).\text{proj}(a)\]

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

- a sub-program $\mathbb{P}'$ can be **effectively computed** (no optimal one)

compute data dependences and control dependences
Program Slicing, M. Weiser [32]

I = set of instructions in P

Slicing

- slice criterion: subset $I' \subseteq I$ and variables $V(i)$ 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 $d$ \( \rho = (i_0, v_0) \ (i_1, v_1) \ \cdots \ (i_k, v_k) \ \cdots \ (i_n, v_n) \)
    - run of $P'$ on $d$ \( \rho' = (i'_0, v'_0) \ (i'_1, v'_1) \ \cdots \ (i'_l, v'_l) \ \cdots \ (i'_m, v'_m) \)

  - 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 $d \in D$, $\text{proj}^*(\rho) = \text{proj}^*(\rho')$

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

Assume CFG of $P$ is known

Slice criterion $C$

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

- each \textit{conditional} branch instruction is in $C$
  
  \[ 36:\text{beq} \ 34 \]

What’s in the Slice?

1. initially slice $S = C$ (memory transfers and conditional branching)
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 $\text{CFG}$ 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 branch instruction is in $C$
  \[ 36:\text{beq} \ 34 \]

What’s in the Slice?

1. initially slice $S = C$ (memory transfers and conditional branching)
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 $C$
  
  $32: \text{ldr } r2, [r1, #4] \ (32, r1)$

- each conditional branch instruction is in $C$
  
  $36: \text{beq } 34$

What’s in the Slice?

1. initially slice $S = C$ (memory transfers and conditional branching)
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.
Sliced Fibonacci Program
\{r0, r2, r3, stack_3020, stack_3028, stack_3052\}

Listing 2. Binary Search Program

Franck Cassez (NICTA, Sydney, June 2011) Computing WCET using Slicing and Model-Checking
Handling Unknown Input Data

- Input data are unknown
- Extended domain: $D_\perp = D \cup \{\perp\}
- Extended semantics
  - Semantics of $\text{add } r_0, r_1, #1$
    $$\text{val}(r_0) := \begin{cases} 
    \perp & \text{if } \text{val}(r_1) = \perp \\
    \text{val}(r_1) + 1 & \text{otherwise}
    \end{cases}$$
  - Semantics of $\text{cmp } r_0, r_1$ that sets predicate $eq$
    $$\text{val}(eq) := \begin{cases} 
    \perp & \text{if } \text{val}(r_0) = \perp \text{ or } \text{val}(r_1) = \perp \\
    \text{val}(r_0) = \text{val}(r_1) & \text{otherwise}
    \end{cases}$$
- $\text{Aut}(P)$ is a non-deterministic automaton
  - $4: \text{ble } 32$ if $\text{val(le)} = \perp$ successors are 8 and 32
Handling Unknown Input Data

- Input data are unknown
- Extended domain: $\mathcal{D}_\bot = \mathcal{D} \cup \{\bot\}$
- Extended semantics
  - Semantics of $\text{add } r_0, r_1, \#1$:
    \[
    \text{val}(r_0) := \begin{cases} 
    \bot & \text{if } \text{val}(r_1) = \bot \\
    \text{val}(r_1) + 1 & \text{otherwise}
    \end{cases}
    \]
  - Semantics of $\text{cmp } r_0, r_1$ that sets predicate $\text{eq}$
    \[
    \text{val}(\text{eq}) := \begin{cases} 
    \bot & \text{if } \text{val}(r_0) = \bot \text{ or } \text{val}(r_1) = \bot \\
    \text{val}(r_0) = \text{val}(r_1) & \text{otherwise}
    \end{cases}
    \]
- $\text{Aut}(P)$ is a non-deterministic automaton
  - $4: \text{ble } 32$ if $\text{val}(\text{le}) = \bot$ successors are 8 and 32
Handling Unknown Input Data

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

  semantics of $\text{add } r_0,r_1,#1$:
  
  $$
  \text{val}(r_0) := \begin{cases} 
  \bot & \text{if } \text{val}(r_1) = \bot \\
  \text{val}(r_1) + 1 & \text{otherwise}
  \end{cases}
  $$

  semantics of $\text{cmp } r_0,r_1$ that sets predicate $\text{eq}$
  
  $$
  \text{val}(\text{eq}) := \begin{cases} 
  \bot & \text{if } \text{val}(r_0) = \bot \text{ or } \text{val}(r_1) = \bot \\
  \text{val}(r_0) = \text{val}(r_1) & \text{otherwise}
  \end{cases}
  $$

- $\text{Aut}(P)$ is a non-deterministic automaton
  
  4: $\text{ble } 32$ if $\text{val}(\text{le}) = \bot$ successors are 8 and 32
Outline

1. Introduction
2. Program & Architecture
3. Computation of WCET
4. Hardware Model
5. Implementation
6. Experiments & Results
7. 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 $\implies$ very bad WCET results

How can we build better models?

- Find a specialist in computer architecture: Jean-Luc
- 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 $\longrightarrow$ very bad WCET results

How can we build better models?
- Find a specialist in computer architecture: Jean-Luc
- 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 $\Rightarrow$ very bad WCET results

How can we build better models?

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

CacheReadEnd[DATA_CACHE]?
CacheWriteEnd[DATA_CACHE]?

is_ldx()
CacheReadStart[DATA_CACHE]!
CD=dataAdr[me],
num_word[me]--

CacheWriteStart[DATA_CACHE]!
CD=dataAdr[me],
num_word[me]--

Todo[me-1] && is_mem_transaction()
memory?
t=0

t<=CYCLE
num_word[me]>0
dataAdr[me]=dataAdr[me]+BLK_SIZE
num_word[me]==0
CacheWriteEnd[DATA_CACHE]?
CacheReadEnd[DATA_CACHE]?

!is_ldx()
CacheReadStart[DATA_CACHE]!
CD=dataAdr[me],
num_word[me]--

is_ldx()
CacheWriteStart[DATA_CACHE]!
CD=dataAdr[me],
num_word[me]--

memory_completed!
copy(me,me+1)

execute_completed?
writeback!

t<=CYCLE

t==CYCLE
memory_completed!
copy(me,me+1)

t==CYCLE

Franck Cassez (NICTA, Sydney, June 2011) Computing WCET using Slicing and Model-Checking 27 / 47
Instruction Cache & Main Memory

Franck Cassez (NICTA, Sydney, June 2011)  Computing WCET using Slicing and Model-Checking
Data Cache

\[ x = \text{CACHE\_SPEED} \land \neg \text{op\_write} \]
\[ \text{CacheReadEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ x = \text{CACHE\_SPEED} \land \text{op\_write} \]
\[ \text{CacheReadEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ x = 1 \]
\[ \text{WriteHit!} \]
\[ \neg \text{write\_hit} \land \text{index}(A)=\text{index}(\text{local\_m}) \]
\[ \text{WriteHit!} \]
\[ \text{is\_in}(m) \]
\[ \text{PMT}=\text{update}(m,1), x=0 \]
\[ \neg \text{is\_in}(m) \]
\[ x=0, \text{Data\_cachemissW}++ \]
\[ \text{CacheWriteStart}[\text{num}]? \]
\[ \text{op\_write}=1, \text{local\_m}=m \]
\[ x = 0 \]
\[ \text{Hurry!} \]
\[ k++ \]
\[ x = 1 \]
\[ \text{WriteHit!} \]
\[ \neg \text{write\_hit} \land \text{index}(A)=\text{index}(\text{local\_m}) \]
\[ \text{Hurry!} \]
\[ l++ \]
\[ x > 1 \]
\[ \text{CacheWriteEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{CacheReadStart}[\text{num}]? \]
\[ \text{local\_m}=m \]
\[ \text{MainMemEnd}? \]
\[ \text{PMT}-- \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ k++ \]
\[ x = 0 \]
\[ \text{PMT}=0 \]
\[ \text{Hurry!} \]
\[ x = 0 \]
\[ \text{PMT}==0 \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ l++ \]
\[ x > 1 \]
\[ \text{CacheWriteEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{CacheReadEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{MainMemEnd}? \]
\[ \text{PMT}-- \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ k++ \]
\[ x = 0 \]
\[ \text{PMT}=0 \]
\[ \text{Hurry!} \]
\[ x = 0 \]
\[ \text{PMT}==0 \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ l++ \]
\[ x > 1 \]
\[ \text{CacheWriteEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{CacheReadEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{MainMemEnd}? \]
\[ \text{PMT}-- \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ k++ \]
\[ x = 0 \]
\[ \text{PMT}=0 \]
\[ \text{Hurry!} \]
\[ x = 0 \]
\[ \text{PMT}==0 \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ l++ \]
\[ x > 1 \]
\[ \text{CacheWriteEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{CacheReadEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
\[ \text{MainMemEnd}? \]
\[ \text{PMT}-- \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ k++ \]
\[ x = 0 \]
\[ \text{PMT}=0 \]
\[ \text{Hurry!} \]
\[ x = 0 \]
\[ \text{PMT}==0 \]
\[ \text{PMT}>0 \]
\[ \text{MainMemStart!} \]
\[ \text{Data\_cachemissR}++ \]
\[ \text{index}(\text{local\_m})=\text{index}(A) \]
\[ \text{Hurry!} \]
\[ l++ \]
\[ x > 1 \]
\[ \text{CacheWriteEnd}[\text{num}]! \]
\[ \text{op\_write}=0, \text{local\_m}=-1 \]
Tool Chain

Program $P$

Slicing

Reduced $\text{Aut}(P')$

generates $\mathcal{L}(P)$

Finite Automaton

Hardware $H$

Synchronization

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

UPPAAL

WCET($H,P$)

Real-Time Model-Checking


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

Franck Cassez (NICTA, Sydney, June 2011)
Compiled Program

...  
0004d44: ebfff9a6  bl  00033e4  <timerGetValue>
0004d48: e1a03000  mov  r3, r0
0004d4c: e58d3004  str  r3, [sp, #4]
0004d50: e3a00001  mov  r3, #1
0004d58: e1a03000  mov  r3, r0
0004d5c: e58d3000  str  r3, [sp]
0004d60: e3a00001  mov  r0, #1
0004d64: ebfff92e  bl  0003214  <fib>
0004d68: e1a03000  mov  r3, #3
0004d6c: ebfff99e  bl  00033e4  <timerGetValue>
0004d70: e1a03000  mov  r3, #3
...
## Results

<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-O0</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-O1</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-O2</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>janne-complex-O0*</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>janne-complex-O1*</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>janne-complex-O2*</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-O1</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-O2</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-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td></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>528684</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>140664</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></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></td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td></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-O0</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-O1</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-O2</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-O0*</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-O1*</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-O2*</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-O0*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>[3133,3108]</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></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></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></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></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></td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (NICTA, Sydney, June 2011)  Computing WCET using Slicing and Model-Checking
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>UPPAAL Time States Explored</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>((C-M)) \text{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-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</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>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>[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-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/1455529</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>
## Results

<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 × 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>1.74s/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-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>166/353</td>
</tr>
</tbody>
</table>

| **Single-Path Programs‡ with MUL/MLA/SMULL instructions (instructions durations depend on data)** |     |              |                   |                   |                |       |
| fdct-00            | 238 | 2.41s/85007  | [11242,11800]     | 11448             | 3.0%           | 253/831 |
| matmult-00*        | 162 | 5m9s/10531230| [502850,529250]   | 511584            | 528684         | 0.1%   | 158/314|
| matmult-02*        | 162 | 43.78s/1780548| [122046,148299]   | 116844            | 140664         | 5.4%   | 75/288 |
| jfdcint-00         | 374 | 2.79s/100784 | [12699,12699]     | 12588             | 12588          | 0.8%   | 159/792|
| jfdcint-01         | 374 | 1.02s/35518  | [4897,4899]       | 4668              | 4668           | 7.0%   | 25/325 |
| jfdcint-02         | 374 | 5.38s/175661 | [16746,16938]     | 16380             | 16380          | 3.4%   | 56/2512|

| **Multiple-Path Programs** |     |              |                   |                   |                |       |
| bs-00               | 174 | 42.6s/1421474| 1068             | 1056              | 1.1%           | 75/151 |
| bs-01               | 174 | 28s/1214673  | 738               | 720               | 2.5%           | 28/82  |
| bs-02               | 174 | 15s/655870   | 628               | 600               | 4.6%           | 28/65  |
| cnt-00*             | 115 | 2.3s/76238   | 9028              | 8836              | 2.1%           | 99/235 |
| cnt-01*             | 115 | 1s/27279     | 4123              | 3996              | 3.1%           | 42/129 |
| cnt-02*             | 115 | 0.5s/11540   | 3065              | 2928              | 4.6%           | 39/263 |
| insertsort-00*      | 91  | 10m35s/24250737| 3133            | 3108              | 0.8%           | 79/175 |
| insertsort-01*      | 91  | 7m2s/11455293| 1533              | 1500              | 2.2%           | 40/115 |
| insertsort-02*      | 91  | 11.5s/387292 | 1371              | 1344              | 2.0%           | 43/108 |
| ns-00*              | 497 | 83.4s/3064315| 30968             | 30732             | 0.8%           | 132/215|
| ns-01*              | 497 | 11.3s/368719 | 11701             | 11568             | 1.1%           | 61/124 |
| ns-02*              | 497 | 29s/1030746  | 7343              | 7236              | 1.4%           | 566/863|
## Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>WCET (C)</th>
<th>Measured WCET (M)</th>
<th>(C-M) M x 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>
</tr>
<tr>
<td>fib-O0</td>
<td>74</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>2597</td>
<td>2544</td>
<td>2.0%</td>
<td>18/72</td>
</tr>
<tr>
<td>fib-O2</td>
<td>74</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>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-O1*</td>
<td>65</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>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-O1</td>
<td>238</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>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>
</tr>
<tr>
<td>matmult-O0*</td>
<td>162</td>
<td>511584</td>
<td>528684</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>162</td>
<td>116844</td>
<td>140664</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>12588</td>
<td>[12699,12699]</td>
<td>0.8%</td>
<td>197/792</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>4668</td>
<td>[4897,4899]</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>16380</td>
<td>[16746,16938]</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>
</tr>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>1056</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>720</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>600</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</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>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</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>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>1500</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</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>30732</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11568</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>7236</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

Franck Cassez (NICTA, Sydney, June 2011)

Computing WCET using Slicing and Model-Checking
<table>
<thead>
<tr>
<th>Program</th>
<th>loc</th>
<th>UPPAAL Time States Explored</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><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-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><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>2.41s/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>5m9s/10531230</td>
<td>[502850,529250]</td>
<td>511584</td>
<td>528684</td>
<td>0.1%</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>5.4%</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>[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-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>
## Results

<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 × 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-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</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></td>
<td>[11242,11800]</td>
<td>11448</td>
<td>3.0%</td>
</tr>
<tr>
<td>matmult-O0*</td>
<td>162</td>
<td>5m9s/10531230</td>
<td></td>
<td>[502850,529250]</td>
<td>511584</td>
<td>0.1%</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>162</td>
<td>43.78s/1780548</td>
<td></td>
<td>[122046,148299]</td>
<td>116844</td>
<td>5.4%</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>2.79s/100784</td>
<td></td>
<td>[12699,12699]</td>
<td>12588</td>
<td>0.8%</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>1.02s/35518</td>
<td></td>
<td>[4897,4899]</td>
<td>4668</td>
<td>7.0%</td>
</tr>
<tr>
<td>jfdcint-O2</td>
<td>374</td>
<td>5.38s/175661</td>
<td></td>
<td>[16746,16938]</td>
<td>16380</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></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>
Is Slicing Critical?

**METAMOC** [15, 14]
- Mälardalen University Sweden and Aalborg Univ. Denmark
- Timed automata (hardware model)
- Annotate with loop bounds
- Value analysis phase
- Loop unfolding
- Compute WCET using UPPAAL

**Results** (from http://metamoc.dk/)
- Programs compiled with **O2** option
- Simple cache formal models
<table>
<thead>
<tr>
<th>Optimization</th>
<th>Data-cache</th>
<th>Instr.-cache</th>
<th>Value-analysis</th>
<th>Core</th>
<th>Cache</th>
<th>Cache</th>
<th>Cycle counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2</td>
<td>-</td>
<td>-</td>
<td>No</td>
<td>19589</td>
<td>1636</td>
<td>7676</td>
<td>1:28:47 (OOM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model checking fails: 3</th>
<th>Manual modification: 1</th>
<th>Value analysis fails: 0</th>
</tr>
</thead>
</table>

Franck Cassez (NICTA, Sydney, June 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](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](http://www.irccyn.fr/franck/wcet))
- Command line compiler from binary programs to UPPAAL
- Qt Interface
## Qt GUI

### ARM Program Analyser

<table>
<thead>
<tr>
<th>PC Address</th>
<th>Hex code</th>
<th>Mnemonic</th>
<th>Arguments</th>
<th>Comment</th>
<th>Referenced Variables</th>
<th>Defined Variables</th>
<th>Def Map</th>
<th>Triggered Conditions</th>
<th>Read from Registers</th>
<th>Written to Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x_0003348 [13128]</td>
<td>e233300a</td>
<td>sub</td>
<td>r3,r3,#10</td>
<td>-</td>
<td>r3</td>
<td>r3</td>
<td>def(r3)=[r3]</td>
<td>r3</td>
<td>r3</td>
<td>r3</td>
</tr>
<tr>
<td>0x_000334c [13132]</td>
<td>e8d3000</td>
<td>str</td>
<td>r3,[sp,#0]</td>
<td>-</td>
<td>r3,sp</td>
<td>stack</td>
<td>def(stack)=[r3,sp]</td>
<td>r3,sp</td>
<td>r3</td>
<td>r3,sp</td>
</tr>
<tr>
<td>0x_0003350 [13136]</td>
<td>e9d2004</td>
<td>ldr</td>
<td>r2,[sp,#4]</td>
<td>-</td>
<td>sp,stack</td>
<td>r2</td>
<td>def(r2)=[sp,stack]</td>
<td>sp</td>
<td>r2</td>
<td>r2</td>
</tr>
<tr>
<td>0x_0003354 [13140]</td>
<td>e3a03f4a</td>
<td>mov</td>
<td>r3,#296</td>
<td>0x128</td>
<td>r3</td>
<td>defr(3)=[]</td>
<td>r3</td>
<td>r3</td>
<td>r3</td>
<td></td>
</tr>
<tr>
<td>0x_0003358 [13144]</td>
<td>e2833003</td>
<td>add</td>
<td>r3,r3,#3</td>
<td>-</td>
<td>r3</td>
<td>r3</td>
<td>def(r3)=[r3]</td>
<td>r3</td>
<td>r3</td>
<td></td>
</tr>
<tr>
<td>0x_000335c [13148]</td>
<td>e150003</td>
<td>cmpl</td>
<td>r2,r3</td>
<td>-</td>
<td>r2,r3</td>
<td>le</td>
<td>def(le)=[r2,r3]</td>
<td>le</td>
<td>r2,r3</td>
<td></td>
</tr>
<tr>
<td>0x_0003360 [13152]</td>
<td>daf0000</td>
<td>ble</td>
<td>0003328</td>
<td>&lt;complex+0x74&gt; le</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_0003364 [13156]</td>
<td>e3a03001</td>
<td>mov</td>
<td>r3,#1</td>
<td>-</td>
<td>r3</td>
<td>def(r3)=[]</td>
<td>r3</td>
<td>r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_0003368 [13160]</td>
<td>e1a00003</td>
<td>mov</td>
<td>r0,r3</td>
<td>-</td>
<td>r3</td>
<td>def(r0)=[]</td>
<td>r3</td>
<td>r0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_000336c [13164]</td>
<td>e20dd008</td>
<td>add</td>
<td>sp,sp,#8</td>
<td>-</td>
<td>sp</td>
<td>sp</td>
<td>def(sp)=[sp]</td>
<td>sp</td>
<td>sp</td>
<td></td>
</tr>
<tr>
<td>0x_0003370 [13168]</td>
<td>e12ff1e</td>
<td>bx</td>
<td>lr</td>
<td>-</td>
<td>lr</td>
<td>pc</td>
<td>def(pc)=[lr]</td>
<td>lr</td>
<td>lr,sp</td>
<td></td>
</tr>
<tr>
<td>0x_0003374 [13172]</td>
<td>e52de004</td>
<td>stmdb</td>
<td>sp,[lr]</td>
<td>(str lr,[sp,#-4])</td>
<td>lr,sp</td>
<td>sp,stack</td>
<td>def(sp)=[sp]</td>
<td>lr,sp</td>
<td>sp</td>
<td></td>
</tr>
<tr>
<td>0x_0003378 [13176]</td>
<td>e24dd014</td>
<td>sub</td>
<td>sp,sp,#20</td>
<td>-</td>
<td>sp</td>
<td>sp</td>
<td>def(sp)=[sp]</td>
<td>sp</td>
<td>sp</td>
<td></td>
</tr>
<tr>
<td>0x_000337c [13180]</td>
<td>e3a03001</td>
<td>mov</td>
<td>r3,#1</td>
<td>-</td>
<td>r3</td>
<td>def(r3)=[]</td>
<td>r3</td>
<td>r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_0003380 [13184]</td>
<td>e58d3004</td>
<td>str</td>
<td>r3,[sp,#4]</td>
<td>-</td>
<td>r3,sp</td>
<td>stack</td>
<td>def(stack)=[r3,sp]</td>
<td>r3,sp</td>
<td>r3,sp</td>
<td></td>
</tr>
<tr>
<td>0x_0003384 [13188]</td>
<td>e3a03001</td>
<td>mov</td>
<td>r3,#1</td>
<td>-</td>
<td>r3</td>
<td>def(r3)=[]</td>
<td>r3</td>
<td>r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_0003388 [13192]</td>
<td>e58d3008</td>
<td>str</td>
<td>r3,[sp,#8]</td>
<td>-</td>
<td>r3,sp</td>
<td>stack</td>
<td>def(stack)=[r3,sp]</td>
<td>r3,sp</td>
<td>r3,sp</td>
<td></td>
</tr>
<tr>
<td>0x_000338c [13196]</td>
<td>e3a03000</td>
<td>mov</td>
<td>r3,#0</td>
<td>-</td>
<td>r3</td>
<td>def(r3)=[]</td>
<td>r3</td>
<td>r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x_0003390 [13200]</td>
<td>e58d300c</td>
<td>str</td>
<td>r3,[sp,#12]</td>
<td>-</td>
<td>r3,sp</td>
<td>stack</td>
<td>def(stack)=[r3,sp]</td>
<td>r3,sp</td>
<td>r3,sp</td>
<td></td>
</tr>
<tr>
<td>0x_0003394 [13204]</td>
<td>e9d0004</td>
<td>ldr</td>
<td>r0,[sp,#4]</td>
<td>-</td>
<td>sp,stack</td>
<td>r0</td>
<td>def(r0)=[sp,stack]</td>
<td>sp</td>
<td>r0</td>
<td></td>
</tr>
<tr>
<td>0x_0003398 [13208]</td>
<td>e9d1008</td>
<td>ldr</td>
<td>r1,[sp,#8]</td>
<td>-</td>
<td>sp,stack</td>
<td>r1</td>
<td>def(r1)=[sp,stack]</td>
<td>sp</td>
<td>r1</td>
<td></td>
</tr>
</tbody>
</table>
Interval in Execute Stage

\[ t \leq DUR\_MAX\_INST \]
\[ \\text{memory!} \]
\[ \\text{copy}(me, me+1) \]
\[ t \geq DUR\_MIN\_INST \]
\[ \text{&& } t \leq DUR\_MAX\_INST \]
\[ \\text{execute?} \]
\[ t=0, DUR\_MAX\_INST = \text{max\_dur}(), \]
\[ DUR\_MIN\_INST = \text{min\_dur}() \]

Franck Cassez (NICTA, Sydney, June 2011)
References I


   aiT Worst-Case Execution Time Analyzers.
   http://www.absint.com/ait/.

   A theory of timed automata.

   Application Note 93 – Benchmarking with ARMulator.
   http://infocenter.arm.com/help/topic/com.arm.doc.dai0093a/DAI0093A_ 
   benchmarking_appsnote.pdf.

   Otawa: An open toolbox for adaptive wcet analysis.
   In Sang Lyul Min, Robert G. Pettit IV, Peter P. Puschner, and Theo Ungerer, editors,
   Software Technologies for Embedded and Ubiquitous Systems (SEUS) - 8th IFIP WG 10.2
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.
forthcoming.

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.  

[19] 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.
References IV


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

UPPAAL in a Nutshell. 

A fast algorithm for finding dominators in a flowgraph. 

[24] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury. 
Chronos: A Timing Analyzer for Embedded Software. 
Special Issue on Experimental Software and Toolkit.
Combining Abstract Interpretation with Model Checking for Timing Analysis of Multicore Software.  

WCET Project – Benchmarks.  

[27] Alexander Metzner.  
Why model checking can improve wcet analysis.  

TuBound – A Conceptually New Tool for WCET Analysis.  

[29] Rapita Systems Ltd.  
Rapita Systems for timing analysis of real-time embedded systems.  
http://www.rapitasystems.com/.
References VI

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

[31] 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.