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

Franck Cassez
NICTA and UNSW, Australia

Jean-Luc Béchennec
CNRS/IRCCyN, France

Toulouse, July 2013
The WCET Problem

Program $P$

```
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: daffffff  ble  24 <fib+0x24>
```

Hardware $H$

- ARM9TDMI Core
- Data Cache
- Instruction Cache
- Coprocessor 15
- MMU
- Write Buffer
- AMBA bus interface

© NICTA 2013
The WCET Problem

Program $P$

```
10: e3a03000  mov  r3, #0
14: e58d0314  str  r3, [sp, #20]
18: e3a03002  mov  r3, #2
1c: e58d030c  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: e59d3014  ldr  r3, [sp, #24]
3c: e59d3014  ldr  r3, [sp, #20]
40: e59d3018  ldr  r3, [sp, #16]
44: e59d3018  ldr  r3, [sp, #12]
48: e2833001  add  r3, r3, #1
4c: e58d030c  str  r3, [sp, #12]
50: e59d200c  ldr  r2, [sp, #12]
54: e59d3004  ldr  r3, [sp, #4]
58: e1520003  cmp  r2, r3
5c: daffffff  bne  24  <fib+0x24>
```

Hardware $H$

$d \in D$ $\rightarrow$ time($H$, $P$, $d$)
The WCET Problem

\[ d \in D \rightarrow \text{Program } P \rightarrow \text{Hardware } H \rightarrow \text{time}(H, P, d) \]
The WCET Problem

Program $P$

Hardware $H$

$\forall d \in D \Rightarrow \text{time}(H, P, d)$

$$\text{WCET}(H, P) = \max_{d \in D} \text{time}(H, P, d)$$
The WCET Problem

\[ d \in \mathcal{D} \rightarrow\]

\[ \begin{align*}
\text{Program } P \\
\text{Hardware } H \\
\text{time}(H, P, d)
\end{align*} \rightarrow\]

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

\[ WCET(H, P) \leq WCET-UB(H, P) \]
The WCET Problem

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

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

• Tests & Simulation
  – Random, probabilistic
  – Real board or simulator
  – Non exhaustive
  – Tools: RapiTime (pWCET, mTime)

• Abstract Interpretation & Integer Linear Programming
  – Compute a CFG
  – Determine loop bounds
  – Build a weighted graph
  – Solve an ILP
  – Tools: Bound-T, OTAWA, TuBound, aiT (Absint)
Related Work & Existing Tools

• **Tests & Simulation**
  – Random, probabilistic
  – Real board or simulator
  – Non exhaustive
  – Tools: RapiTime (pWCET, mTime)

• **Abstract Interpretation & Integer Linear Programming**
  – Compute a CFG
  – Determine loop bounds
  – Build a weighted graph
  – Solve an ILP
  – Tools: Bound-T, OTAWA, TuBound, aiT (Absint)


### Evaluation of WCET Techniques and Tools

- **WCET 2006 challenge**

#### 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>adpcm</td>
<td>558342</td>
<td>137588</td>
<td>430274</td>
<td>265588</td>
<td>347742</td>
<td>317354</td>
</tr>
<tr>
<td>2</td>
<td>cnt</td>
<td>20250</td>
<td>17053</td>
<td>7376</td>
<td>4896</td>
<td>6438</td>
<td>5401</td>
</tr>
<tr>
<td>3</td>
<td>compress</td>
<td>37570</td>
<td>20280</td>
<td>9461</td>
<td>5873</td>
<td>29215</td>
<td>28487</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>109735</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>86393</td>
<td>53625</td>
</tr>
<tr>
<td>12</td>
<td>ns</td>
<td>73738</td>
<td>36097</td>
<td>buffer</td>
<td>6577</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.

---

[1] Lili Tan

The Worst Case Execution Time Tool Challenge 2006: The External Test
Leveraging Applications of Formal Methods, pp 241–248, 2006
Evaluation of WCET Techniques and Tools (2)

- **WCET 2006 challenge**

<table>
<thead>
<tr>
<th>Table 9 Overview of Functional and Service Quality of WCET Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>aiT</td>
</tr>
<tr>
<td>Bound-T</td>
</tr>
<tr>
<td>SWEET</td>
</tr>
<tr>
<td>Chronos</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.

[1] Lili Tan

The Worst Case Execution Time Tool Challenge 2006: The External Test
Leveraging Applications of Formal Methods, pp 241–248, 2006
What about Model-Checking Based Techniques?

Bad News
Wilhelm, R.

Why AI+ILP is good for WCET, but MC is not, nor ILP alone.
VMCAI, 2004
What about Model-Checking Based Techniques?

Bad News
Wilhelm, R.
Why AI+ILP is good for WCET, but MC is not, nor ILP alone.
VMCAI, 2004

Good News
Metzner, A.
Why model checking can improve WCET analysis.
CAV, 2004
METAMOC: Dalsgaard, A.E., Olesen, M.C., Toft, M.
Modular execution time analysis using model checking.
METAMOC (Aalborg University, DK)
- Modular Execution Time Analysis using Model Checking
- Timed Automata to model Hardware
- Loop bounds, value analysis to compute CFG
- Loop unfolding
- UPPAAL to compute longest path (WCET)

Our Methodology
- Automatic computation of WCET
- No loop bounds computation
- UPPAAL to compute longest path (WCET)
- Timed games for computing WCET for pipelined processors with caches. In *ACSD'2011*.
- Timed analysis of binary programs with UPPAAL, In *ACSD'2013*
## METAMOC

<table>
<thead>
<tr>
<th>Optimization</th>
<th>ATML</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-cache</td>
<td>O2</td>
<td>O2</td>
<td>O2</td>
<td>O2</td>
<td>O2</td>
<td>O2</td>
<td>O2</td>
</tr>
<tr>
<td>Instr-cache</td>
<td>No</td>
<td>miss_writeback</td>
<td>Concrete, 128 lines/set, LRU</td>
<td>Concrete, 128 lines/set, LRU</td>
<td>Concrete, 64 lines/set, LRU</td>
<td>Concrete, LRU</td>
<td>Yes</td>
</tr>
<tr>
<td>Value-analysis</td>
<td>No</td>
<td>miss</td>
<td>Concrete, LRU</td>
<td>No</td>
<td>Concrete, LRU</td>
<td>No</td>
<td>Concrete, LRU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>ATML</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm</td>
<td>1:44:19</td>
<td>20:12.95</td>
<td>1:46.67</td>
<td>OOM/error</td>
<td>4:44.83</td>
<td>OOM/error</td>
<td>4:44.83</td>
</tr>
<tr>
<td>bs</td>
<td>0:03:66</td>
<td>0:01:09</td>
<td>0:01:06</td>
<td>964</td>
<td>0:01:27</td>
<td>640</td>
<td>0:01:27</td>
</tr>
<tr>
<td>bsit/bsitoo</td>
<td>0:29:54</td>
<td>7997421</td>
<td>0:57.66</td>
<td>3425888</td>
<td>408745</td>
<td>OOM</td>
<td>1:50.13</td>
</tr>
<tr>
<td>cnt</td>
<td>0:02:49</td>
<td>29572</td>
<td>188456</td>
<td>52481</td>
<td>0:04:00</td>
<td>6137</td>
<td>0:04:00</td>
</tr>
<tr>
<td>compress</td>
<td>6:29:88</td>
<td>568922</td>
<td>158097</td>
<td>69460</td>
<td>0:04:56+0.2265</td>
<td>0:27:21</td>
<td>310722</td>
</tr>
<tr>
<td>crc</td>
<td>0:36:71</td>
<td>170959</td>
<td>2043591</td>
<td>328310</td>
<td>964</td>
<td>0:39:83</td>
<td>433344</td>
</tr>
<tr>
<td>dm</td>
<td>0:29:42</td>
<td>345399</td>
<td>2923937</td>
<td>687463</td>
<td>2:06:10</td>
<td>42071</td>
<td>305777</td>
</tr>
<tr>
<td>expit</td>
<td>4:56:17</td>
<td>7563931</td>
<td>652215</td>
<td>31483</td>
<td>0:12:72</td>
<td>689</td>
<td>0:12:72</td>
</tr>
<tr>
<td>fss</td>
<td>0:01:43</td>
<td>968</td>
<td>11353</td>
<td>1250</td>
<td>0:01:35</td>
<td>158827</td>
<td>0:01:35</td>
</tr>
<tr>
<td>fuce</td>
<td>0:02:43</td>
<td>4866</td>
<td>344234</td>
<td>198602</td>
<td>4122.70</td>
<td>599</td>
<td>4122.70</td>
</tr>
<tr>
<td>fuce</td>
<td>0:01:04</td>
<td>438</td>
<td>13434</td>
<td>532</td>
<td>0:01:52</td>
<td>18763</td>
<td>0:01:52</td>
</tr>
<tr>
<td>frr</td>
<td>0:25:90</td>
<td>792494</td>
<td>131508</td>
<td>21364</td>
<td>0:04:53</td>
<td>37814</td>
<td>0:04:53</td>
</tr>
<tr>
<td>insersion</td>
<td>0:01:26</td>
<td>2672</td>
<td>75078</td>
<td>43688</td>
<td>0:02:44</td>
<td>1883</td>
<td>0:02:44</td>
</tr>
<tr>
<td>janne_complex</td>
<td>2:11:27</td>
<td>54383</td>
<td>294957</td>
<td>133059</td>
<td>0:01:92</td>
<td>111510</td>
<td>0:01:92</td>
</tr>
<tr>
<td>jocent</td>
<td>0:05:80</td>
<td>525383</td>
<td>5856449</td>
<td>2636719</td>
<td>712869</td>
<td>1:36:94</td>
<td>1:36:94</td>
</tr>
<tr>
<td>maxmul</td>
<td>0:17:02</td>
<td>22:16</td>
<td>729579</td>
<td>29060</td>
<td>4:25:09</td>
<td>10873</td>
<td>4:25:09</td>
</tr>
<tr>
<td>notes</td>
<td>0:01:79</td>
<td>13116</td>
<td>279579</td>
<td>29060</td>
<td>0:07:09</td>
<td>10873</td>
<td>0:07:09</td>
</tr>
<tr>
<td>nscmem</td>
<td>0:01:79</td>
<td>13116</td>
<td>279579</td>
<td>29060</td>
<td>0:07:09</td>
<td>10873</td>
<td>0:07:09</td>
</tr>
<tr>
<td>prime</td>
<td>0:12:52</td>
<td>1020957</td>
<td>622904</td>
<td>3:37:70</td>
<td>469860</td>
<td>2:41:87</td>
<td>469860</td>
</tr>
<tr>
<td>ud</td>
<td>0:11:81</td>
<td>58217</td>
<td>1160049</td>
<td>175359</td>
<td>23:07</td>
<td>158819</td>
<td>23:07</td>
</tr>
</tbody>
</table>

Model checking fails: 3
Model checking fails: 4
Value analysis fails: 0
Value analysis fails: 2
Manual modification: 1
Manual modification: 0
Manual modification: 3
Manual modification: 2
Manual modification: 1
### Evaluation of WCET Techniques and Tools (3)

#### METAMOC

<table>
<thead>
<tr>
<th>Optimization</th>
<th>ATML</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
<th>ARM9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-cache</td>
<td>O2</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
</tr>
<tr>
<td>Instr-cache</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value-analysis</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>O2</th>
<th>O2</th>
<th>O2</th>
<th>O2</th>
<th>O2</th>
<th>O2</th>
<th>O2</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc</td>
<td>OOM</td>
<td>1929472</td>
<td>1.46</td>
<td>1.46</td>
<td>4.44</td>
<td>OOM</td>
<td>4.44</td>
</tr>
<tr>
<td>ts</td>
<td>0.03</td>
<td>145</td>
<td>3685</td>
<td>0.01</td>
<td>0.01</td>
<td>0.85</td>
<td>0.85</td>
</tr>
<tr>
<td>bssort</td>
<td>0.29</td>
<td>7997421</td>
<td>3425888</td>
<td>0.06</td>
<td>0.06</td>
<td>1.50</td>
<td>1.50</td>
</tr>
<tr>
<td>cnt</td>
<td>0.02</td>
<td>29572</td>
<td>188456</td>
<td>0.02</td>
<td>0.02</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>compress</td>
<td>6.29</td>
<td>58692</td>
<td>158697</td>
<td>0.09</td>
<td>0.09</td>
<td>27.21</td>
<td>27.21</td>
</tr>
<tr>
<td>crc</td>
<td>0.37</td>
<td>17959</td>
<td>204391</td>
<td>0.27</td>
<td>0.27</td>
<td>39.83</td>
<td>39.83</td>
</tr>
<tr>
<td>tcam</td>
<td>0.29</td>
<td>345499</td>
<td>203937</td>
<td>0.25</td>
<td>0.25</td>
<td>40.72</td>
<td>40.72</td>
</tr>
<tr>
<td>export</td>
<td>4.56</td>
<td>7583931</td>
<td>552215</td>
<td>0.09</td>
<td>0.09</td>
<td>31483</td>
<td>31483</td>
</tr>
<tr>
<td>fast</td>
<td>0.01</td>
<td>996</td>
<td>11533</td>
<td>0.01</td>
<td>0.01</td>
<td>6893</td>
<td>6893</td>
</tr>
<tr>
<td>fcol</td>
<td>0.02</td>
<td>4806</td>
<td>344424</td>
<td>0.08</td>
<td>0.08</td>
<td>412270</td>
<td>412270</td>
</tr>
<tr>
<td>frcache</td>
<td>0.01</td>
<td>438</td>
<td>13434</td>
<td>0.01</td>
<td>0.01</td>
<td>599</td>
<td>599</td>
</tr>
<tr>
<td>fr</td>
<td>0.25</td>
<td>792494</td>
<td>131508</td>
<td>0.02</td>
<td>0.02</td>
<td>18763</td>
<td>18763</td>
</tr>
<tr>
<td>insersion</td>
<td>0.01</td>
<td>2872</td>
<td>79078</td>
<td>0.01</td>
<td>0.01</td>
<td>183</td>
<td>183</td>
</tr>
<tr>
<td>janne_complex</td>
<td>OOM</td>
<td>577877</td>
<td>1883</td>
<td>0.01</td>
<td>0.01</td>
<td>111510</td>
<td>111510</td>
</tr>
<tr>
<td>jslot</td>
<td>0.05</td>
<td>54383</td>
<td>294957</td>
<td>0.06</td>
<td>0.06</td>
<td>1084016</td>
<td>1084016</td>
</tr>
<tr>
<td>maxmult</td>
<td>0.17</td>
<td>525383</td>
<td>5836449</td>
<td>0.40</td>
<td>0.40</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>notes</td>
<td>0.64</td>
<td>OOM</td>
<td>2575424</td>
<td>0.03</td>
<td>0.03</td>
<td>29060</td>
<td>29060</td>
</tr>
<tr>
<td>nscnmeu</td>
<td>0.01</td>
<td>13116</td>
<td>279579</td>
<td>8.27</td>
<td>8.27</td>
<td>37.70</td>
<td>37.70</td>
</tr>
<tr>
<td>prime</td>
<td>0.12</td>
<td>170806</td>
<td>647769</td>
<td>1.56</td>
<td>1.56</td>
<td>4187</td>
<td>4187</td>
</tr>
<tr>
<td>udi</td>
<td>0.11</td>
<td>58217</td>
<td>1160049</td>
<td>0.14</td>
<td>0.14</td>
<td>2307</td>
<td>2307</td>
</tr>
</tbody>
</table>

- 3 errors / 19 benchmarks
- 1 errors / 21 benchmarks
- 2 errors / 21 benchmarks
- 3 errors / 21 benchmarks

- Model checking fails: 3
- Model checking fails: 2
- Model checking fails: 1
- Model checking fails: 0

- Value analysis fails: 5
- Value analysis fails: 4
- Value analysis fails: 3
- Value analysis fails: 2
- Value analysis fails: 1

- Manual modification: 1
- Manual modification: 2
- Manual modification: 3
- Manual modification: 4
- Manual modification: 5

---

© NICTA 2013
Our Contribution

• Fully **automatic** computation of WCET
• **Modular Technique**
  – Build CFG
  – Build model of hardware
  – Model-check product of CFG and Hardware
• **Formal Hardware** model for ARM920T
• **Comparison** of computed WCETs and measured WCETs
• **Features:**
  – Can accommodate different processor speeds
  – Easy to change hardware
  – Distribution of ExecTime (statistical model-checking)
Overview

- What type of programs and architectures?
- Modular computation of WCET
- Reduced WCET-equivalent program
- CFG reconstruction
- Hardware formal models for ARM920T
- Experiments
- Conclusion
Binary Programs and Architecture
### Binary Program: Fibonacci

#### Listing 1.

```assembly
Listing 1. Binary Search Program
```

<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>&lt;main&gt;:</td>
<td>00000000</td>
</tr>
<tr>
<td>0:</td>
<td>e24dd020</td>
<td>&lt;fib&gt;:</td>
</tr>
<tr>
<td>4:</td>
<td>e58d0004</td>
<td>mov r3, $1</td>
</tr>
<tr>
<td>8:</td>
<td>e3a03001</td>
<td>mov r3, [sp, #16]</td>
</tr>
<tr>
<td>c:</td>
<td>e58d3010</td>
<td>str r3, [sp, #20]</td>
</tr>
<tr>
<td>10:</td>
<td>e3a03000</td>
<td>mov r3, $0</td>
</tr>
<tr>
<td>14:</td>
<td>e58d3014</td>
<td>str r3, [sp, #12]</td>
</tr>
<tr>
<td>18:</td>
<td>e3a03002</td>
<td>mov r3, $2</td>
</tr>
<tr>
<td>1c:</td>
<td>e58d300c</td>
<td>str r3, [sp, #4]</td>
</tr>
<tr>
<td>20:</td>
<td>ea00000a</td>
<td>b 50 &lt;fib+0x50&gt;</td>
</tr>
<tr>
<td>24:</td>
<td>e59d3010</td>
<td>ldr r3, [sp, #16]</td>
</tr>
<tr>
<td>28:</td>
<td>e58d3018</td>
<td>str r3, [sp, #24]</td>
</tr>
<tr>
<td>2c:</td>
<td>e59d2010</td>
<td>ldr r2, [sp, #16]</td>
</tr>
<tr>
<td>30:</td>
<td>e59d3014</td>
<td>ldr r3, [sp, #20]</td>
</tr>
<tr>
<td>34:</td>
<td>e0823003</td>
<td>add r3, r2, r3</td>
</tr>
<tr>
<td>38:</td>
<td>e58d3010</td>
<td>str r3, [sp, #16]</td>
</tr>
<tr>
<td>3c:</td>
<td>e59d3018</td>
<td>ldr r3, [sp, #24]</td>
</tr>
<tr>
<td>40:</td>
<td>e58d3014</td>
<td>str r3, [sp, #20]</td>
</tr>
<tr>
<td>44:</td>
<td>e59d300c</td>
<td>ldr r3, [sp, #12]</td>
</tr>
<tr>
<td>48:</td>
<td>e2833001</td>
<td>add r3, r3, #1</td>
</tr>
<tr>
<td>4c:</td>
<td>e58d300c</td>
<td>str r3, [sp, #12]</td>
</tr>
<tr>
<td>50:</td>
<td>e59d200c</td>
<td>ldr r2, [sp, #16]</td>
</tr>
<tr>
<td>54:</td>
<td>e59d3004</td>
<td>ldr r3, [sp, #4]</td>
</tr>
<tr>
<td>58:</td>
<td>e1520003</td>
<td>cmp r2, r3</td>
</tr>
<tr>
<td>5c:</td>
<td>daaffff0</td>
<td>ble 24 &lt;fib+0x24&gt;</td>
</tr>
<tr>
<td>60:</td>
<td>e59d3010</td>
<td>ldr r3, [sp, #16]</td>
</tr>
<tr>
<td>64:</td>
<td>e58d301c</td>
<td>str r3, [sp, #28]</td>
</tr>
<tr>
<td>68:</td>
<td>e59d301c</td>
<td>ldr r3, [sp, #28]</td>
</tr>
<tr>
<td>6c:</td>
<td>e1a00003</td>
<td>mov r0, r3</td>
</tr>
<tr>
<td>70:</td>
<td>e28dd020</td>
<td>add sp, sp, $32</td>
</tr>
<tr>
<td>74:</td>
<td>e12fff1e</td>
<td>bx lr</td>
</tr>
<tr>
<td>00000078</td>
<td>&lt;main&gt;:</td>
<td></td>
</tr>
<tr>
<td>78:</td>
<td>e52de004</td>
<td>push {lr}</td>
</tr>
<tr>
<td>7c:</td>
<td>e24dd00c</td>
<td>sub sp, sp, $12</td>
</tr>
<tr>
<td>80:</td>
<td>e3a03f4b</td>
<td>mov r3, $300</td>
</tr>
<tr>
<td>84:</td>
<td>e58d3004</td>
<td>str r3, [sp, #4]</td>
</tr>
<tr>
<td>88:</td>
<td>e59d0004</td>
<td>ldr r0, [sp, #4]</td>
</tr>
<tr>
<td>8c:</td>
<td>ebfffd0b</td>
<td>bl 0 &lt;fib&gt;</td>
</tr>
<tr>
<td>90:</td>
<td>e1a03000</td>
<td>mov r3, r0</td>
</tr>
<tr>
<td>94:</td>
<td>e1a00003</td>
<td>mov r0, r3</td>
</tr>
<tr>
<td>98:</td>
<td>e28dd00c</td>
<td>add sp, sp, $12</td>
</tr>
<tr>
<td>9c:</td>
<td>e49de004</td>
<td>pop {lr}</td>
</tr>
<tr>
<td>a0:</td>
<td>e12fff1e</td>
<td>bx lr</td>
</tr>
</tbody>
</table>

© NICTA 2013
Binary Program: Fibonacci

Listing 1. Binary Search Program

```
ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY

ENTRY
Example: 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  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
```

```
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: ebfffd5b  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
```

Hardware: ARM920T

- **RISC** processor, 16 registers
- memory load/store and **multiple ldr/str**
- Data and Instruction **Caches**
### Pipelining and Pipeline Stalls

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>D</td>
<td>E</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>E</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>D</td>
<td>E</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>j</td>
</tr>
<tr>
<td>j+1</td>
</tr>
<tr>
<td>j+2</td>
</tr>
<tr>
<td>j+3</td>
</tr>
<tr>
<td>j+4</td>
</tr>
<tr>
<td>j+5</td>
</tr>
<tr>
<td>j+6</td>
</tr>
</tbody>
</table>

- **add r2,r2,#1**
- **sub r3,r3,#2**
- **ldr r1,[sp,#4]**
Pipelining and Pipeline Stalls

Stalls may occur!

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

cycle

Stalls may occur!
Pipeline Stalls

- Data dependences

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

```
F    D    E    M    W
F    D    stall    E    M    W
```
• **Data dependences**

\[
\begin{align*}
\text{ldr } r2, [sp, #4] & \quad \text{F} \quad \text{D} \quad \text{E} \quad \text{M} \quad \text{W} \\
\text{add } r1, r2, #2 & \quad \text{F} \quad \text{D} \quad \text{stall} \quad \text{E} \quad \text{M} \quad \text{W}
\end{align*}
\]

• **Dynamic computation of address of next instruction**

\[
\begin{align*}
\text{ble 32} & \quad \text{F} \quad \text{D} \quad \text{E} \quad \text{M} \quad \text{W} \\
& \quad \text{stall} \quad \text{stall} \quad \text{F} \quad \text{D} \quad \text{E} \quad \text{M} \quad \text{W}
\end{align*}
\]
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

© NICTA 2013
64-way associative cache
8 sets, 64 lines/set
line size 8 (4-byte) words
FIFO replacement

```
CacheRead(X)!

CacheWrite(X)!

Memory Location X

Miss/FIFO replacement
```
Modular Computation of WCET
What do we Need to Compute the WCET?

10: e3a03000 mov r3, #0
14: e58d3014 str r3, [sp, #20]
16: 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: daffffff ble 24 <fib+0x24>

© NICTA 2013
What do we Need to Compute the WCET?

| 10: e3a03000 | mov | r3, #0 |
| 14: e58d3014 | str | r3, [sp, #10] |
| 18: e3a03002 | mov | r3, #2 |
| 1c: e58d300c | str | r3, [sp, #12] |
| 20: ea0000a | 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: e0830003 | add | r3, r2, r3 |
| 38: e58d3010 | str | r3, [sp, #16] |
| 3c: e58d3014 | ldr | r3, [sp, #24] |
| 40: e59d3014 | str | r3, [sp, #20] |
| 44: e59d300c | ldr | r3, [sp, #12] |
| 48: e2830001 | 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> |
What do we Need to Compute the WCET?

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

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Register(s)</th>
<th>Note(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>mov r3, #0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>str r3, [sp, #20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>mov r3, #2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>str r3, [sp, #12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>ldr r3, [sp, #16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>str r3, [sp, #24]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2c</td>
<td>ldr r2, [sp, #16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>ldr r3, [sp, #20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>add r3, r2, r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>str r3, [sp, #16]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3c</td>
<td>ldr r3, [sp, #24]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>str r3, [sp, #20]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>ldr r3, [sp, #12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>add r3, r3, #1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4c</td>
<td>str r3, [sp, #12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>ldr r2, [sp, #12]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>ldr r3, [sp, #4]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>cmp r2, r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5c</td>
<td>ble 24 &lt;fib+0x24&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© NICTA 2013
What do we Need to Compute the WCET?

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

\( S_i \xrightarrow{32: \text{add } r2, r2, \#2} S_j \)

\( S_i \xrightarrow{12: \text{stm } sp!, \{r2, lr\}} \{100, 104\} \)

\( S_n \xrightarrow{\text{not stall} \land y \geq 1} \)

\[ [32: \text{add } r2, r2, \#2; \emptyset] \]

Reset \( x \)

CacheWrite!

\[ [12: \text{stm } sp!, \{r2, lr\}; \{100, 104\}] \]

Reset \( y \)

\[ \Sigma^* \rightarrow \mathbb{N} \]
Input data are unknown: extended domain: $\mathcal{D}_\bot = \mathcal{D} \cup \{ \bot \}$
Modular Computation of WCET

Program $P$

Semantics

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

Finite Automaton

Hardware $H$

\[ \text{HDL, …} \]

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

Timed Automaton
Modular Computation of WCET

Program $P$

Semantics

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

Finite Automaton

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

Hardware $H$

HDL, …

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

Timed Automaton

Real-Time Model-Checking

$\text{UPPAAL}$

$\text{WCET}(H, P)$
Reduced WCET-Equivalent Program
Reduced Automaton

- **Two runs of** $P$ **can generate the same word in** $\mathcal{L}(P)$
  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** $Aut(P)$: $16 \times 32 + |stack| \times 32 + 4$
• Two runs of \( P \) can generate the same word in \( \mathcal{L}(P) \)

  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 \( \text{WCET}(H, P) = \text{WCET}(H, P') \)
Reduced Automaton

- Two runs of $P$ can generate the same word in $\mathcal{L}(P)$
  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
000000000 <fib>:
  0: e24dd020 sub sp, sp, #32
  4: e58dd004 str r0, [sp, #4]
  8: e3a03001 mov r3, #1
 10: e3a03000 str r3, [sp, #16]
 14: e58dd014 str r3, [sp, #20]
 18: e3a03002 mov r3, #2
 1c: e58dd00c str r3, [sp, #12]
 20: ea00000a b 50 <fib+0x50>
 24: e59d3010 ldr r3, [sp, #16]
 28: e58dd018 str r3, [sp, #24]
 2c: e59d3010 ldr r2, [sp, #16]
 30: e59d3014 ldr r3, [sp, #20]
 34: e0833003 add r3, r2, r3
 38: e58dd010 str r3, [sp, #16]
 3c: e59d3018 ldr r3, [sp, #24]
 40: e58dd014 str r3, [sp, #20]
 44: e59d300c ldr r3, [sp, #12]
 48: e2833001 add r3, r3, #1
 4c: e58dd00c 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>
 60: e59d3010 ldr r3, [sp, #16]
 64: e58dd01c ldr r3, [sp, #28]
 68: e59d301c ldr r3, [sp, #28]
 6c: ea000000 mov r0, r3
 70: e28dd020 add sp, sp, #32
 74: e2fff0e bx lr

000000078 <main>:
  78: e52de004 push {lr}
  7c: e24dd00c sub sp, sp, #12
  80: e3a03f4b mov r3, #300
  84: e58dd004 str r3, [sp, #4]
  88: e59d0004 ldr r0, [sp, #4]
  8c: ebff00f0 bl 0 <fib>
  90: ea003000 mov r3, r0
  94: ea000000 mov r0, r3
  98: e28dd00c add sp, sp, #12
  9c: e49de004 pop {lr}
 a0: e2fff0e bx lr

© NICTA 2013
WCET-Equivalent Fibonacci

000000000 <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 ldr r3, [sp, #24]
  2c: e59d2010 ldr r2, [sp, #16]
  30: e59d3014 ldr r3, [sp, #20]
  34: e0823003 add r3, r2, r3
  38: e59d3010 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: daaffff0 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

000000078 <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: ebfffdcb 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

ENTRY

80  ldr r2,[sp,#12]
84  ldr r3,[sp,#4]
88  cmps r2,r3
92  ble 24
96  ldr r0,[sp,#16]
100 str r3,[sp,#28]
104 str r3,[sp,#24]
108 mov r0,r3
112 add sp,sp,#32
116 bx lr
120 stmdb sp!,{lr}
124 sub sp,sp,#12
128 mov r3,#300
132 str r3,[sp,#4]
136 ldr r0,[sp,#16]
140 bl 0
144 mov r3,r0
148 mov r0,r3
152 add sp,sp,#12
156 ldmia sp!,{lr}
160 bx lr
164 str r3,[sp,#28]
168 str r3,[sp,#24]
172 add r3,r2,r3
176 str r3,[sp,#12]
180 str r3,[sp,#20]
184 str r3,[sp,#16]
188 cmps r2,r3
192 ble 24
196 ldr r1,[sp,#16]
200 str r3,[sp,#24]
204 str r3,[sp,#20]
208 str r3,[sp,#12]
212 bx lr
216 end

END
WCET-Equivalent Fibonacci

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: e58d0310  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: e283001  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>
  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

000000078 <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: ebffff0b  bl  0 <lib>
  90: e1a03000  mov  r3, r0
  94: e1a00003  mov  r0, r3
  98: e28dd00c  add  sp, sp, #12
  9c: e49de004  pop  (lr)
 a0: e12fff1e  bx  lr
```

---

© NICTA 2013
WCET-Equivalent Fibonacci

Size of Aut(P): CFG + r0,r2,r3,stack_3020,stack_3028,stack_3052
CFG Reconstruction
Slicing ... Can also be Used to Build CFG
Slicing ... Can also be Used to Build CFG
Slicing ... Can also be Used to Build CFG
Formal Hardware Models
Fetch and Decode Stages

- **prog_completed?**
  - fetch_completed!
  - !stall()
  - decode!
  - copy(me, me+1)
  - CacheReadStart[INSTR_CACHE]!
  - CacheReadEnd[INSTR_CACHE]?
  - t=0, DUR_INSTR=dur()
  - num_word[me]--
  - CD=dataAdr[me],
  - CacheWriteStart[DATA_CACHE]!
  - CacheWriteEnd[DATA_CACHE]?
  - CacheReadEnd[DATA_CACHE]?
  - num_word[me]>0
  - num_word[me]==0
  - t<=CYCLE
  - done

- **decode?**
  - fetch_completed?
  - !stall()
  - decode!
  - copy(me, me+1)
  - CacheReadStart[DATA_CACHE]!
  - CacheReadEnd[DATA_CACHE]?
  - !stall()
  - execute!
  - copy(me, me+1)
  - Memory completed!
  - t=0
  - num_word[me]--
  - CD=dataAdr[me],
  - CacheWriteStart[DATA_CACHE]!
  - CacheWriteEnd[DATA_CACHE]?
  - num_word[me]>0
  - num_word[me]==0
  - t<=CYCLE
  - done
Execute Stage with Data Dependent Timing

- \( t = 0, \text{DUR\_MAX\_INSTR} = \text{max\_dur}(), \text{DUR\_MIN\_INSTR} = \text{min\_dur}() \)
- \( t \leq \text{DUR\_MAX\_INSTR} \)
- \( t > \text{DUR\_MIN\_INSTR} \) \&\& \( t \leq \text{DUR\_MAX\_INSTR} \)
- \( \text{decode\_completed}? \)
- \( \text{execute\_completed}! \)
- \( \text{copy}(\text{me}, \text{me}+1) \)
Memory Stage

- `CacheReadEnd[DATA_CACHE]`?
- `CacheWriteEnd[DATA_CACHE]`?
- `is_ldx()`
- `CacheReadStart[DATA_CACHE]`!
  - `CD=dataAdr[me]`
  - `num_word[me]--`
- `!is_ldx()`
- `CacheWriteStart[DATA_CACHE]`!
  - `CD=dataAdr[me]`
  - `num_word[me]--`
- `Todo[me-1]` && `is_mem_transaction()`
  - `memory?`
  - `t=0`
- `!Todo[me-1]` || `is_mem_transaction()`
  - `memory?`
  - `t=0`
- `execute_completed?`
- `num_word[me]>0`
- `td=dataAdr[me]=dataAdr[me]+BLK_SIZE`
- `num_word[me]==0`
- `t==CYCLE`
- `num_word[me]==0`
- `writeback!`
  - `copy(me,me+1)`
- `execute_completed?`
- `memory_completed!`
Instruction Cache, Main Memory and Write Buffer

- \( x = \text{CACHE\_SPEED} \)
  - \( x = \text{initCache}() \)
  - \( x = 0 \)
  - \( x = \text{CACHE\_SPEED} \)

- \( x = 0 \)
  - \( \text{CacheReadStart}[\text{num}]? \)
    - \( \text{PMT}=0 \)
    - \( \text{PMT} = \text{is\_in(m)?0:insert(m)} \)
    - \( \text{MainMemStart!} \)
    - \( \text{ICcachemiss}++ \)

- \( \text{MainMemEnd?} \)
  - \( \text{PMT} = \text{is\_in(m)?0:insert(m)} \)
  - \( \text{PMT} = \text{insert(m)} \)
  - \( \text{MainMemStart}! \)

- \( \text{MainMemEnd}\)
Data Cache

- \( x \leq 1 \)
- \( x = 1 \)
- \( x \leq 1 \)
- \( \text{write}_\text{hit} \land \text{index}(A) = \text{index}(\text{local}_m) \)
- \( \text{WriteHit!} \)
- \( \text{is}_\text{in}(m) \)
- \( \text{PMT}=\text{update}(m,1), x=0 \)
- \( \text{CacheWriteStart}[\text{num}]? \)
  \( \text{op}\_\text{write}=1, \text{local}_m=m \)
- \( \text{CacheReadStart}[\text{num}]? \)
  \( \text{local}_m=m \)
- \( \text{MainMemEnd}? \)
  \( \text{PMT}-- \)
  \( \text{PMT}>0 \)
  \( \text{MainMemStart}? \)
  \( \text{Datacache}\text{missR}++ \)
  \( \text{index}(\text{local}_m) = \text{index}(A) \)
  \( \text{Hurry!} \)
  \( k++ \)
- \( \text{PMT}=\text{insert}(m,0), x=0 \)
- \( \text{CacheReadEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{CacheWriteEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{initialize}? \)
  \( \text{initCache}() \)
- \( x = 0 \)
- \( x = 0, \text{Datacache}\text{missW}++ \)
- \( x = 0\)
- \( \text{PMT}-- \)
- \( \text{PMT}>0 \)
- \( \text{MainMemStart}? \)
- \( \text{Datacache}\text{missR}++ \)
- \( \text{index}(\text{local}_m) = \text{index}(A) \)
- \( \text{Hurry!} \)
- \( l++ \)
- \( \text{write}_\text{hit} \land \land \text{index}(A) = \text{index}(\text{local}_m) \)
- \( \text{WriteHit!} \)
- \( \text{is}_\text{in}(m) \)
  \( \text{PMT}=\text{update}(m,1), x=0 \)
- \( \text{CacheWriteStart}[\text{num}]? \)
  \( \text{op}\_\text{write}=1, \text{local}_m=m \)
- \( \text{CacheReadStart}[\text{num}]? \)
  \( \text{local}_m=m \)
- \( \text{MainMemEnd}? \)
  \( \text{PMT}-- \)
  \( \text{PMT}>0 \)
  \( \text{MainMemStart}? \)
  \( \text{Datacache}\text{missR}++ \)
  \( \text{index}(\text{local}_m) = \text{index}(A) \)
  \( \text{Hurry!} \)
  \( l++ \)
- \( \text{PMT}=\text{insert}(m,0), x=0 \)
- \( \text{CacheReadEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{CacheWriteEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{initialize}? \)
  \( \text{initCache}() \)
- \( x = 0 \)
- \( x = 0, \text{Datacache}\text{missW}++ \)
- \( x \leq 1 \)
- \( x > 1 \)
- \( x \leq 1 \)
- \( x > 1 \)
- \( x \leq 1 \)
- \( x > 1 \)
- \( \text{nb} < 4 \)
- \( \text{nb}++ \)
- \( x > 1 \)
- \( x \leq 1 \)
- \( \text{write}_\text{hit} \land \text{index}(A) = \text{index}(\text{local}_m) \)
- \( \text{WriteHit!} \)
- \( \text{is}_\text{in}(m) \)
  \( \text{PMT}=\text{update}(m,1), x=0 \)
- \( \text{CacheWriteStart}[\text{num}]? \)
  \( \text{op}\_\text{write}=1, \text{local}_m=m \)
- \( \text{CacheReadStart}[\text{num}]? \)
  \( \text{local}_m=m \)
- \( \text{MainMemEnd}? \)
  \( \text{PMT}-- \)
  \( \text{PMT}>0 \)
  \( \text{MainMemStart}? \)
  \( \text{Datacache}\text{missR}++ \)
  \( \text{index}(\text{local}_m) = \text{index}(A) \)
  \( \text{Hurry!} \)
  \( l++ \)
- \( \text{PMT}=\text{insert}(m,0), x=0 \)
- \( \text{CacheReadEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{CacheWriteEnd}[\text{num}]! \)
  \( \text{op}\_\text{write}=0, \text{local}_m=-1 \)
- \( \text{initialize}? \)
  \( \text{initCache}() \)
- \( x = 0 \)
- \( x = 0, \text{Datacache}\text{missW}++ \)
- \( x \leq 1 \)
- \( x > 1 \)
- \( x \leq 1 \)
- \( x > 1 \)
- \( \text{nb} < 4 \)
- \( \text{nb}++ \)
- \( x \leq 1 \)
- \( x > 1 \)
Experiments
Measuring Execution Times

```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
## Experimental Results

<table>
<thead>
<tr>
<th>Program(^\ddagger)</th>
<th>loc(^\dagger)</th>
<th>Time/States Explored</th>
<th>UPPAAL</th>
<th>Computed BCET/WCET (C)</th>
<th>Measured BCET/WCET (M)</th>
<th>Error (%)(^\ddagger)</th>
<th>Slice(^\S)</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>2s/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>0.6s/22333</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/9711</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>1.7s/38038</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>0.5s/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>0.5s/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>21s/60534</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(^\ddagger) with MUL/MLA/SMULL instructions (duration of instruction depends on data)</strong></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>124s/85008</td>
<td></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>217s/10531262</td>
<td></td>
<td>502849/529250</td>
<td>511584/528684</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-O1(^*)</td>
<td>162</td>
<td>25s/1112527</td>
<td></td>
<td>129967/156367</td>
<td>127356/153000</td>
<td>2.2%</td>
<td>71/172</td>
</tr>
<tr>
<td>matmult-O2(^*)</td>
<td>162</td>
<td>121s/6780931</td>
<td></td>
<td>122045/148299</td>
<td>116844/140664</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>92s/100861</td>
<td></td>
<td>12726/12918</td>
<td>12588</td>
<td>2.6%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>12s/35419</td>
<td></td>
<td>4880/5072</td>
<td>4668</td>
<td>8.6%</td>
<td>25/325</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>
<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>30s/1421274</td>
<td></td>
<td>478/1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>23s/1214673</td>
<td></td>
<td>321/738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>12s/655870</td>
<td></td>
<td>273/628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0(^*)</td>
<td>115</td>
<td>4s/77002</td>
<td></td>
<td>9025/9027</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1(^*)</td>
<td>115</td>
<td>1.4s/27146</td>
<td></td>
<td>4123/4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2(^*)</td>
<td>115</td>
<td>9s/11490</td>
<td></td>
<td>3067/3067</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0(^*)</td>
<td>91</td>
<td>598.98s/24250738</td>
<td></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>353.80s/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>11.68s/387292</td>
<td></td>
<td>1326</td>
<td>1320</td>
<td>0.4%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0(^*)</td>
<td>497</td>
<td>60s/3064316</td>
<td></td>
<td>940/30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1(^*)</td>
<td>497</td>
<td>8s/368720</td>
<td></td>
<td>605/1701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2(^*)</td>
<td>497</td>
<td>55s/1030746</td>
<td></td>
<td>441/7280</td>
<td>7236</td>
<td>0.6%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

\(^\ddagger\) file-Ox indicates that file was compiled using gcc -Ox
\(^\dagger\) lines of code in the C source file
\(^\ddagger\) \(\frac{(C-M)}{M} \times 100\) computed using the upper bound for C and M
\(^\S\) Instructions in Slice/Instructions in Program
\(^*\) Program selected for the WCET Challenge 2006

\(\text{UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB}\)
## Experimental Results

### Program

<table>
<thead>
<tr>
<th>Program†</th>
<th>loc†</th>
<th>UPPAAL Time/States Explored¶</th>
<th>Computed BCET/WCET (C)</th>
<th>Measured BCET/WCET (M)</th>
<th>Error (%)‡</th>
<th>Slice§</th>
</tr>
</thead>
<tbody>
<tr>
<td>fib-O0</td>
<td>74</td>
<td>2s/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>0.6s/22333</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/9711</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>1.7s/38038</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>0.5s/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>0.5s/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>21s/60534</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>
</tbody>
</table>

### Single-Path Programs† with MUL/MLA/SMULL instructions (duration of instruction depends on data)

<table>
<thead>
<tr>
<th>Program†</th>
<th>loc†</th>
<th>UPPAAL Time/States Explored¶</th>
<th>Computed BCET/WCET (C)</th>
<th>Measured BCET/WCET (M)</th>
<th>Error (%)‡</th>
<th>Slice§</th>
</tr>
</thead>
<tbody>
<tr>
<td>matmult-O0*</td>
<td>238</td>
<td>124s/85008</td>
<td>11242/11800</td>
<td>11448</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-O1*</td>
<td>238</td>
<td>217s/1053126</td>
<td>502849/529250</td>
<td>511584/528684</td>
<td>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>238</td>
<td>255s/1112527</td>
<td>129967/156367</td>
<td>127356/153000</td>
<td>2.2%</td>
<td>71/172</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>92s/100861</td>
<td>12726/12918</td>
<td>12588</td>
<td>2.6%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>12s/35419</td>
<td>4880/5072</td>
<td>4668</td>
<td>8.6%</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>
</tbody>
</table>

### Multiple-Path Programs

<table>
<thead>
<tr>
<th>Program†</th>
<th>loc†</th>
<th>UPPAAL Time/States Explored¶</th>
<th>Computed BCET/WCET (C)</th>
<th>Measured BCET/WCET (M)</th>
<th>Error (%)‡</th>
<th>Slice§</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>30s/1421274</td>
<td>478/1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>23s/1214673</td>
<td>321/738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>12s/655870</td>
<td>273/628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>4s/77002</td>
<td>9025/9027</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1.4s/27146</td>
<td>4123/4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>9s/11490</td>
<td>3067/3067</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>598.98s/24250738</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>353.80s/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>11.68s/387292</td>
<td>1326</td>
<td>1320</td>
<td>0.4%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>60s/3064316</td>
<td>940/30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>8s/368720</td>
<td>605/11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>55s/1030746</td>
<td>441/7280</td>
<td>7236</td>
<td>0.6%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

† file-Ox indicates that file was compiled using gcc -Ox
‡ lines of code in the C source file
¶ (C - M) / M × 100 computed using the upper bound for C and M
§ Instructions in Slice/Instructions in Program
• Program selected for the WCET Challenge 2006
UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB
## Experimental Results

<table>
<thead>
<tr>
<th>Program†</th>
<th>loc‡</th>
<th>Time/States Explored¶</th>
<th>Computed BCET/WCET (C)</th>
<th>Measured BCET/WCET (M)</th>
<th>Error (%‡)</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>2s/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>0.6s/22333</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/9711</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>1.7s/38038</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>0.5s/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>0.5s/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>21s/60534</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 (duration of instruction depends on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>matmult-O0*</td>
<td>238</td>
<td>12s/10531262</td>
<td>502849/529250</td>
<td>511584/528684</td>
<td>3.0%</td>
<td>253/831</td>
</tr>
<tr>
<td>matmult-O1*</td>
<td>162</td>
<td>25s/1112527</td>
<td>129967/156367</td>
<td>127356/153000</td>
<td>2.2%</td>
<td>71/172</td>
</tr>
<tr>
<td>matmult-O2*</td>
<td>162</td>
<td>12s/6789311</td>
<td>122045/148299</td>
<td>116844/140664</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-O0</td>
<td>374</td>
<td>92s/100861</td>
<td>12726/12918</td>
<td>129588</td>
<td>2.6%</td>
<td>159/792</td>
</tr>
<tr>
<td>jfdcint-O1</td>
<td>374</td>
<td>12s/35419</td>
<td>4880/5072</td>
<td>4668</td>
<td>8.6%</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>30s/1421274</td>
<td>478/1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>23s/1214673</td>
<td>321/738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>12s/655870</td>
<td>273/628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>4s/77002</td>
<td>9025/9027</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1.4s/27146</td>
<td>4123/4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>9s/11490</td>
<td>3067/3067</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>598.98s/24250738</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>353.80s/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>11.68s/387292</td>
<td>1326</td>
<td>1320</td>
<td>0.4%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>60s/3064316</td>
<td>940/30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>8s/368720</td>
<td>605/11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>55s/1030746</td>
<td>441/7820</td>
<td>7236</td>
<td>0.6%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

† file-Ox indicates that file was compiled using gcc -Ox
‡ lines of code in the C source file
¶ \( \frac{(C-M)}{M} \times 100 \) computed using the upper bound for \( C \) and \( M \)
§ Instructions in Slice/Instructions in Program
* Program selected for the WCET Challenge 2006
© NICTA 2013

UPPAAL 4.1.11/Intel Pentium 5/3.1Ghz/16GB
UPPAAL Statistical Model Checking: Sorting Algorithm (insertsort-O2)

Almost Sorted List

No Assumption
Conclusion

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 Malardalen, Sweden
- over-approximation is less than 5%

Advantages of our method
- Modular
- Fully automatic
- Can easily accommodate new features
Ongoing Work

• Support for new architectures
  – Multi-core
  – Assembly languages
• Improved Analysis
  – Model checking of binary programs
  – Abstraction refinement: Interpolant automata for programs
• Computation of most unfavorable initial cache state
  – Interpolant