Timed Games for Computing WCET on Pipelined Processors with Caches

Franck Cassez

CNRS & Marie Curie Fellow
IRCCyN, Nantes, France

June 24th, 2011
ACSD'2011, Newcastle, UK
Outline of the talk

1. The Worst-Case Execution-Time Problem
2. Modular Computation of WCET
3. Experiments & Results
4. Conclusion & Future Work
Worst-Case Execution-Time

Program P

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

Hardware H
Worst-Case Execution-Time

Program P

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

Hardware H

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

Program P

Input data \( d \in D \) → \( \text{time}(H, P, d) \) → Hardware H

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

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator

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

Tools: RapiTime (based on pWCET) and Mtime

Exhaustive - Static Analysis & Integer Linear Programming

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

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

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

Partial - Tests/Simulation

- random
- probabilistic
- real board, simulator

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

Tools: RapiTime (based on pWCET) and Mtime

Exhaustive - Static Analysis & Integer Linear Programming

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

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

Tools: Bound-T, OTAWA, TuBound, Chronos, SWEET and aiT (AbsInt)
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

The Fibonacci Program

Listing 1. Binary Search Program

```
ENTRY
120:  stmdb sp!,{lr}
124:  sub sp,sp,#12
128:  mov r3,#300
132:  str r3,[sp,#4]
136:  ldr r0,[sp,#4]
140:  bl 0
144:  mov r3,r0
148:  mov r0,r3
152:  sub sp,sp,#32
156:  ldmia sp!,{lr}
160:  bx lr
END

ENTRY
0:     e24dd020  sub sp,sp,#32
4:     e58d0004  str r0,[sp,#4]
8:     e3a03001  mov r3,#1
12:    e58d3010  str r3,[sp,#16]
16:    e3a03000  mov r3,#0
20:    e58d3014  str r3,[sp,#20]
24:    e3a03002  mov r3,#2
28:    e58d30cc  str r3,[sp,#12]
32:    e0833001  add r3,sp,#1
36:    e58d3020  str r3,[sp,#2]
40:    ea00000a  b 50 <fib+0x50>
44:    e59d3010  ldr r3,[sp,#16]
48:    e58d3014  str r3,[sp,#20]
52:    e59d3018  ldr r3,[sp,#24]
56:    e59d2010  ldr r2,[sp,#16]
60:    e58d3014  str r3,[sp,#20]
64:    e0833002  add r3,sp,#2
68:    e59d3018  ldr r3,[sp,#24]
72:    e59d2010  ldr r2,[sp,#16]
76:    e58d3014  str r3,[sp,#20]
80:    e59d301a  ldr r3,[sp,#22]
84:    e59d301c  ldr r3,[sp,#24]
```
The Fibonacci Program

Listing 1. Binary Search 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: e59d300c  ldr r2, [sp, #12]
```

Listing 1. Binary Search Program

```assembly
ENTRY
   120 stmdb sp!,{lr}
   124 sub sp,sp,#12
   128 mov r3,#300
   132 str r3,[sp,#4]
   136 ldr r0,[sp,#4]
   140 bl 0
   144 mov r3,r0
   148 mov r0,r3
   152 add sp,sp,#12
   156 ldmia sp!,{lr}
   160 bx lr
END
```

ACSD'2011 (Newcastle, UK, June 2011)
The Fibonacci Program

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

Listing 1. Binary Search Program

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

Listing 1. Binary Search Program

ENTRY
120 stmdb sp!,{lr}
END
124 sub sp,sp,#12
128 mov r3,#300
132 str r3,[sp,#4]
136 ldr r0,[sp,#4]
140 bl ... mov r0,r3
112 add sp,sp,#32
116 bx lr
144 mov r3,r0
148 mov r0,r3
152 add sp,sp,#12
156 ldmia sp!,{lr}
160 bx lr

ACSD'2011 (Newcastle, UK, June 2011)
Target Architecture: ARM9xx family - ARM920T

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

Pipelining = split execution of instructions into simple stages

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

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline of the ARM920T

Pipelining = split execution of instructions into simple stages

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline of the ARM920T

Pipelining = split execution of instructions into simple stages

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

Cycle:
```
j | j+1 | j+2 | j+3 | j+4 | j+5 | j+6
```

Concurrent execution of stages: on average one cycle per instruction

...but sometimes pipeline stalls
Pipeline Stalls

- Data dependences between instructions

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

Next instruction is a target of a “branch” instruction

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

- **Data dependences between instructions**

  \[\text{ldr } r2, [sp, #4]\]
  \[\text{add } r1, r2, #2\]

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

  \[\text{ble } 32\]

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

- **Data dependences between instructions**

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

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

```assembly
ble 32
```

**Summary:**

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

Set 1

Set 2

Set $2^n$

Line size

CacheRead(X)!

CacheWrite(X)!

HIT

MISS

Read/Write cached data

Main Memory transfer cach update
Caches

Set 1

Set 2

Set $2^n$

Line size

CacheRead(X)!

CacheWrite(X)!

HIT

MISS

Read/Write cached data

Main Memory transfer cach update

ACSD’2011 (Newcastle, UK, June 2011) Timed Games for Computing WCET
Caches

Set 1
Set 2
Set $2^n$

Line size

CacheRead(X)!
CacheWrite(X)!

Read/Write cached data
HIT
MISS
Main Memory transfer cach update

Main Memory transfer cach update
Caches

Set 1
Set 2
Set $2^n$

Line size

Read/Write cached data

CacheRead(X)!
CacheWrite(X)!

HIT
MISS

Main Memory transfer cach update
What is really needed to compute WCET?

```
00000000 <fib>:
  0: e24dd020 sub sp, sp, #32
  4: e58d0004 str r0, [sp, #4]
  8: e3a03001 mov r3, #1
c: e58d3010 ...
10: e3a03000 mov r3, #0
14: e58d3014 str r3, [sp, #20]
18: e3a03002 mov r3, #2
20: ea00000a b 50 <fib+0x50>
24: e59d3010 ldr r3, [sp, #16]
28: 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: e59d301c ldr r3, [sp, #28]
40: e59d301c ldr r3, [sp, #28]
44: e1a00003 mov r0, r3
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
What is really needed to compute WCET?

Listing 1. Binary Search Program

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

32: add r2, r2, #2

12: stm sp!, {r2, lr}

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: e59d3014 ldr r3, [sp, #20]
40: e59d2010 ldr r2, [sp, #16]
44: e59d3014 ldr r3, [sp, #20]
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>

What is really needed to compute WCET?

Listing 1. Binary Search Program

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

not stall ∧ y ≥ 1
\[ 32: \text{add } r2, r2, \#2; \varnothing \]
Reset x

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

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

\[ \Sigma^* \rightarrow \mathbb{N} \]
What is really needed to compute WCET?

Listing 1. Binary Search Program

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

Listing 1. Binary Search Program

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

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

Program $P$

Semantics

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

Finite Automaton
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

Hardware H

HDL, ...

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

Timed Automaton

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

Real-Time Model-Checking

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

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

**WCET-equivalent Program**

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

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 $\text{Aut}(P)$: $16 \times 32 + \lvert \text{stack} \rvert \times 32 + 4$
- WCET depends on $L(P)$

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

WCET-equivalent Program

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

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

WCET-Equivalent Reduced Program

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

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

**WCET-equivalent Program**

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

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

Program slicing.
WCET-Equivalent Reduced Program

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

WCET-equivalent Program

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

Program slicing.
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 $\text{WCET}(H, P) = \text{WCET}(H, P')$

**WCET-equivalent Program**

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

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


Program slicing.

Sliced Fibonacci Program
{r0,r2,r3,stack_3020,stack_3028,stack_3052}

```assembly
0: e24dd020 sub sp, sp, #32
4: e58d0004 str r0, [sp, #4]
8: e3a03001 mov r3, #1
10: e3a03000 mov r3, #0
14: e58d3014 str r3, [sp, #20]
18: e3a03002 mov r3, #2
20: ea00000a b 50 <fib+0x50>
24: e2833001 add r3, r3, #1
24: e0823003 add r3, r2, r3
28: e58d3014 str r3, [sp, #20]
32: e59d3010 ldr r3, [sp, #16]
34: e58d3010 str r3, [sp, #16]
36: e58d3010 str r3, [sp, #16]
38: e58d3003 add r3, r2, r3
40: e58d3003 add r3, r3, #1
40: e58d3014 str r3, [sp, #24]
44: e58d3003 add r3, r3, #1
44: e58d3014 str r3, [sp, #24]
48: e2833001 add r3, r3, #1
50: e58d3003 add r3, r2, r3
52: e58d3003 add r3, r3, #1
54: e58d3004 ldr r3, [sp, #4]
56: e1520003 cmp r2, r3
58: daafffff0 ble 24 <fib+0x24>
60: e59d3010 ldr r3, [sp, #16]
62: e59d3010 ldr r3, [sp, #16]
64: e59d3010 ldr r3, [sp, #16]
68: e1a00003 mov r0, r3
70: e28dd020 add sp, sp, #32
72: 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: ebfffd0b bl 0 <fib>
90: e1a00003 mov r3, r0
94: e1a00003 mov r0, r3
98: e28dd00c add sp, sp, #12
9c: e49de004 pop {lr}
a0: e12fff1e bx lr
```

Listing 2. Binary Search Program

```assembly
ENTRY
120 stmdb sp!,{lr}
124 sub sp,sp,#12
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 mov r3,#1
12 str r3,[sp,#16]
16 mov r3,#0
20 str r3,[sp,#20]
24 mov r3,#2
28 str r3,[sp,#12]
32 b 50
80 ldr r2,[sp,#12]
84 ldr r3,[sp,#4]
88 cmps r2,r3
92 ble 24
96 ldr r3,[sp,#16]
100 ldr r3,[sp,#28]
104 ldr r3,[sp,#28]
108 mov r0,r3
112 add sp,sp,#32
116 bx lr
144 mov r3,r0
148 mov r0,r3
152 add sp,sp,#12
156 ldmia sp!,{lr}
160 bx lr
36 ldr r3,[sp,#16]
40 ldr r3,[sp,#24]
76 str r3,[sp,#12]
80 ldr r3,[sp,#16]
84 ldr r3,[sp,#28]
144 mov r3,r0
152 add sp,sp,#12
80 ldr r2,[sp,#12]
84 ldr r3,[sp,#4]
88 cmps r2,r3
92 ble 24
160 bx lr
ENTRY
END
```

ACSD’2011 (Newcastle, UK, June 2011)
Automatic Computation of CFG using Program Slicing

ENTRY

120  stmdb sp!,{lr}
124  sub sp,sp,#12
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  mov r3,#1
12  str r3,[sp,#16]
16  mov r3,#0
20  str r3,[sp,#20]
24  mov r3,#2
28  str r3,[sp,#12]
32  b 50

80  ldr r2,[sp,#12]
84  ldr r3,[sp,#4]
88  cmps r2,r3
92  ble 24

96  ldr r3,[sp,#16]
100  str r3,[sp,#28]
104  ldr r3,[sp,#28]
108  mov r0,r3
112  add sp,sp,#32
116  bx lr

144  mov r3,r0
148  mov r0,r3
152  add sp,sp,#12
156  ldmia sp!,{lr}
160  bx lr

EXIT_0x100807a00
Automatic Computation of CFG using Program Slicing
Handling Unknown Input Data

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

```
ENTRY
120  stmdb sp!,{lr}
END
124  sub sp,sp,#12
128  mov r3,#300
132  str r3,[sp,#4]
136  ldr r0,[sp,#4]
140  bl ... mov r0,r3
112  add sp,sp,#32
116  bx lr
144  mov r3,r0
148  mov r0,r3
152  add sp,sp,#12
156  ldmia sp!,{lr}
160  bx lr
```

```
88: cmps r2,r3
92: ble 24
96: ldr r3,[sp,#16]
36: ldr r3,[sp,#16]
100: str r3,[sp,#28]
40: str r3,[sp,#24]

88: cmp r2,r3
if ($r_2 = \bot \lor r_3 = \bot$) le := $\bot$
else le := ($r_2 = r_3$)

92: ble 0x24 (32)
le := $\bot$ or
le := false

96: ldr r3,[sp,#16]
36: ldr r3,[sp,#16]
100: str r3,[sp,#28]
40: str r3,[sp,#24]

Game: program vs outcomes of comparisons
```
Handling Unknown Input Data

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

Game: program vs outcomes of comparisons
**Hardware Formal Models**

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

**Hardware Specs?**
- Data sheets
- Incomplete or sketchy

Bad formal models $\rightarrow$ very bad WCET results

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

Formal Models

- Timed Automata: automata extended with dense-time clocks

Hardware Specs?

- Data sheets
- Incomplete or sketchy

Bad formal models $\Rightarrow$ very bad WCET results

How can we build better models?

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

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

Hardware Specs?
- Data sheets
- Incomplete or sketchy

Bad formal models $\Rightarrow$ very bad WCET results

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

- **fetch_completed!**
- **prog_completed?**
- **decode!**
- **copy(me, me+1)**
- **CacheReadStart[INSTR_CACHE]!**
- **CacheReadEnd[INSTR_CACHE]?!**
- **n_steps++**

- **fetch?**
- **execute!**
- **copy(me, me+1)**
- **execute_completed!**
- **memory_completed?**
- **writeback?**
- **memory_completed!**

- **t<=CYCLE**
- **t==CYCLE**
- **DONE**

- **ACSD'2011 (Newcastle, UK, June 2011) Timed Games for Computing WCET**
Pipeline Model – Memory Stage

**ACSD’2011 (Newcastle, UK, June 2011)**
Timed Games for Computing WCET
Instruction Cache & Main Memory

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

x <= CACHE_SPEED
MainMemEnd?
CacheReadStart[num]?
PMT = is_in(m)?0:insert(m)
PMT--
PMT > 0
MainMemStart!
ICcachemiss++

x <= 6
MainMemEndWB?
MainMemStartWB?
t = MAINMEMTRANS
t <= MAINMEMTRANS

x <= 6
MainMemEnd!
t = MAINMEMTRANS
t <= MAINMEMTRANS

nb > 0
MainMemStartWB!
x = 0

nb--
x = 6
MainMemEndWB!
x = 0

ACSD'2011 (Newcastle, UK, June 2011) Timed Games for Computing WCET
Data Cache

- $x \leq 1$
- $x \leq \text{CACHE\_SPEED}$
- $x = 1$
- $x > 1$

- $\text{index}(A) \neq \text{index}(\text{local}_m)$
- $\text{WriteHit!}$
- $!\text{write\_hit} \land \text{index}(A) = \text{index}(\text{local}_m)$
- $\text{WriteHit!}$
- $\text{is\_in}(m)$
- $\text{PMT} = \text{update}(m, 1), x = 0$
- $!\text{is\_in}(m)$
- $x = 0$
- $\text{PMT} = \text{insert}(m, 0), x = 0$
- $\text{Initialize?}$
- $\text{initCache()}$
- $x = 0$
- $\text{x\_SPEED} \land !\text{op\_write}$
- $\text{Cache\_WriteEnd[num]}!$
- $\text{op\_write} = 0, \text{local}_m = -1$
- $x = 0$
- $\text{x\_SPEED} \land \text{op\_write}$
- $\text{Cache\_WriteEnd[num]}!$
- $\text{op\_write} = 0, \text{local}_m = -1$

- $\text{Main\_MemEnd?}$
- $\text{PMT} --$
- $\text{PMT} > 0$

- $\text{Main\_MemStart!}$
- $\text{Data\_Cache\_MissR++}$
- $\text{index}(\text{local}_m) \neq \text{index}(A)$
- $\text{Hurry!}$

- $\text{main\_MemStart!}$
- $\text{Data\_Cache\_MissR++}$
- $\text{index}(\text{local}_m) \neq \text{index}(A)$
- $\text{Hurry!}$

- $\text{nb} < 4$
- $\text{Hurry!}$
- $\text{nb}++$

- $x > 1$
- $x = 1$
- $x \leq 1$

ACSD'2011 (Newcastle, UK, June 2011) Timed Games for Computing WCET
Tool Chain

Program P

Slicing

Reduced Aut(P') generates L(P) Finite Automaton

Hardware H

HDL, ...

Synchronization Aut(P') || Aut(H)

UPPAAL

WCET(H,P)

Slicing


Real-Time Model-Checking

<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-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/9110</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td>22/71</td>
</tr>
<tr>
<td>janne-complex-O0*</td>
<td>65</td>
<td>1.15s/38014</td>
<td>4264</td>
<td>4164</td>
<td>2.4%</td>
<td>78/173</td>
</tr>
<tr>
<td>janne-complex-O1*</td>
<td>65</td>
<td>0.48s/14600</td>
<td>1715</td>
<td>1680</td>
<td>2.0%</td>
<td>30/89</td>
</tr>
<tr>
<td>janne-complex-O2*</td>
<td>65</td>
<td>0.46s/13004</td>
<td>1557</td>
<td>1536</td>
<td>1.3%</td>
<td>32/78</td>
</tr>
<tr>
<td>fdct-O1</td>
<td>238</td>
<td>1.67s/60418</td>
<td>4245</td>
<td>4092</td>
<td>3.7%</td>
<td>100/363</td>
</tr>
<tr>
<td>fdct-O2</td>
<td>238</td>
<td>3.24s/55285</td>
<td>19231</td>
<td>18984</td>
<td>1.3%</td>
<td>166/3543</td>
</tr>
<tr>
<td><strong>Single-Path Programs† with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>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>43.7s/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/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>
### Experiments & Results

#### Benchmarks, Mälardalen Univ.

<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 × 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>fb-O0</td>
<td>74</td>
<td>1.74s/74181</td>
<td>8098</td>
<td>8064</td>
<td>0.42%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fb-O1</td>
<td>74</td>
<td>0.61s/22332</td>
<td>2597</td>
<td>2544</td>
<td>2.0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fb-O2</td>
<td>74</td>
<td>0.3s/9710</td>
<td>1209</td>
<td>1164</td>
<td>3.8%</td>
<td></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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>3.0%</td>
<td></td>
<td></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>0.8%</td>
<td></td>
<td></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></td>
<td></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></td>
<td></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>1068</td>
<td>1056</td>
<td>1.1%</td>
<td></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></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></td>
<td></td>
</tr>
</tbody>
</table>

ACSD’2011 (Newcastle, UK, June 2011)
## Experiments & Results

### Benchmarks, Mälardalen Univ.

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>$(C-M) \times 100$</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>States Explored</td>
<td></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-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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single-Path Programs‡ with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>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>526864</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>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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-O0</td>
<td>174</td>
<td>42.6s/1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75/151</td>
</tr>
<tr>
<td>bs-O1</td>
<td>174</td>
<td>28s/1214673</td>
<td>738</td>
<td>720</td>
<td>2.5%</td>
<td>28/82</td>
</tr>
<tr>
<td>bs-O2</td>
<td>174</td>
<td>15s/655870</td>
<td>628</td>
<td>600</td>
<td>4.6%</td>
<td>28/65</td>
</tr>
<tr>
<td>cnt-O0*</td>
<td>115</td>
<td>2.3s/76238</td>
<td>9028</td>
<td>8836</td>
<td>2.1%</td>
<td>99/235</td>
</tr>
<tr>
<td>cnt-O1*</td>
<td>115</td>
<td>1s/27279</td>
<td>4123</td>
<td>3996</td>
<td>3.1%</td>
<td>42/129</td>
</tr>
<tr>
<td>cnt-O2*</td>
<td>115</td>
<td>0.5s/11540</td>
<td>3065</td>
<td>2928</td>
<td>4.6%</td>
<td>39/263</td>
</tr>
<tr>
<td>insertsort-O0*</td>
<td>91</td>
<td>10m35s/24250737</td>
<td>3133</td>
<td>3108</td>
<td>0.8%</td>
<td>79/175</td>
</tr>
<tr>
<td>insertsort-O1*</td>
<td>91</td>
<td>7m2s/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>
## Experiments & Results

[Benchmarks, Mälardalen Univ.]

<table>
<thead>
<tr>
<th>Program</th>
<th>Ioc</th>
<th>UPPAAL Time</th>
<th>Computed WCET (C)</th>
<th>Measured WCET (M)</th>
<th>(C–M) ×100</th>
<th>Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>States Explored</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fib-00</td>
<td>74</td>
<td>1.74s/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>0.1%</td>
<td>158/314</td>
</tr>
<tr>
<td>matmult-02*</td>
<td>162</td>
<td>43.78s/1780548</td>
<td>[122046,148299]</td>
<td>116844</td>
<td>5.4%</td>
<td>75/288</td>
</tr>
<tr>
<td>jfdcint-00</td>
<td>374</td>
<td>2.79s/100784</td>
<td>[12699,12699]</td>
<td>12588</td>
<td>0.8%</td>
<td>199/792</td>
</tr>
<tr>
<td>jfdcint-01</td>
<td>374</td>
<td>1.02s/35518</td>
<td>[4897,4899]</td>
<td>4668</td>
<td>7.0%</td>
<td>25/325</td>
</tr>
<tr>
<td>jfdcint-02</td>
<td>374</td>
<td>5.38s/175661</td>
<td>[16746,16938]</td>
<td>16380</td>
<td>3.4%</td>
<td>56/2512</td>
</tr>
</tbody>
</table>

| **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|
## Experiments & Results [Benchmarks, Mälardalen Univ.]

<table>
<thead>
<tr>
<th>Program</th>
<th>loc</th>
<th>Time</th>
<th>States Explored</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>
<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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td><strong>Single-Path Programs</strong> with MUL/MLA/SMULL instructions (instructions durations depend on data)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-00</td>
<td>238</td>
<td>2.41s / 85007</td>
<td>11448</td>
<td>11448</td>
<td>3.0%</td>
<td>253 / 831</td>
<td></td>
</tr>
<tr>
<td>matmult-00*</td>
<td>162</td>
<td>5m9s / 10531230</td>
<td>[502850, 529250]</td>
<td>51584</td>
<td>528684</td>
<td>5.4%</td>
<td>159 / 792</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>2.0%</td>
<td>52 / 288</td>
</tr>
<tr>
<td>jfdcint-00</td>
<td>374</td>
<td>2.79s / 100784</td>
<td>12588</td>
<td>12588</td>
<td>0.8%</td>
<td>159 / 792</td>
<td></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>
<td></td>
</tr>
<tr>
<td>jfdcint-02</td>
<td>374</td>
<td>5.38s / 175661</td>
<td>[1646, 16938]</td>
<td>16380</td>
<td>3.4%</td>
<td>56 / 2512</td>
<td></td>
</tr>
<tr>
<td><strong>Multiple-Path Programs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bs-00</td>
<td>174</td>
<td>42.6s / 1421474</td>
<td>1068</td>
<td>1056</td>
<td>1.1%</td>
<td>75 / 151</td>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
</tbody>
</table>
## Experiments & Results

[Benchmarks, Mälardalen Univ.]

<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>$(C-M) \times 100 \over M$</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† with MUL/MLA/SMULL instructions (instructions durations depend on data)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fdct-O0</td>
<td>238</td>
<td>2.41s/85007</td>
<td>[11242,11800]</td>
<td>11448</td>
<td>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>5.4%</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/11455293</td>
<td>1533</td>
<td>1500</td>
<td>2.2%</td>
<td>40/115</td>
</tr>
<tr>
<td>insertsort-O2*</td>
<td>91</td>
<td>11.5s/387292</td>
<td>1371</td>
<td>1344</td>
<td>2.0%</td>
<td>43/108</td>
</tr>
<tr>
<td>ns-O0*</td>
<td>497</td>
<td>83.4s/3064315</td>
<td>30968</td>
<td>30732</td>
<td>0.8%</td>
<td>132/215</td>
</tr>
<tr>
<td>ns-O1*</td>
<td>497</td>
<td>11.3s/368719</td>
<td>11701</td>
<td>11568</td>
<td>1.1%</td>
<td>61/124</td>
</tr>
<tr>
<td>ns-O2*</td>
<td>497</td>
<td>29s/1030746</td>
<td>7343</td>
<td>7236</td>
<td>1.4%</td>
<td>566/863</td>
</tr>
</tbody>
</table>

ACSD'2011 (Newcastle, UK, 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](http://www.irccyn.fr/franck/wcet))
- Command line compiler from binary programs to UPPAAL
- Qt Interface
Ongoing and Future Work

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

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

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

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

New Features

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

New Architectures

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

Enhanced Analysis (model-checking)

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

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

- Command line compiler from binary programs to UPPAAL
- Qt Interface
Is Slicing Critical?

**METAMOC** [Master's Thesis 2009, WCET'2010]
- 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/](http://metamoc.dk/))
- Programs compiled with **O2 option**
- **Simple cache** formal models
### Timed Games for Computing WCET

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>O2</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>adpcm</td>
<td>OOM</td>
<td>19629472</td>
<td>OOM</td>
<td>19629472</td>
<td>OOM</td>
<td>19629472</td>
<td>OOM</td>
<td>19629472</td>
</tr>
<tr>
<td>bsort100</td>
<td>0.03:66</td>
<td>0.01:09</td>
<td>0.01:06</td>
<td>0.01:27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cht</td>
<td>0.02:46</td>
<td>0.02:06</td>
<td>0.02:62</td>
<td>0.04:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>compress</td>
<td>6:29:88</td>
<td>0.06:52</td>
<td>0.09:11</td>
<td>69486</td>
<td>0:04:56+0:22:65</td>
<td>0:27:21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>crc</td>
<td>1:56:36</td>
<td>0:01:09</td>
<td>0:01:08</td>
<td>599</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ecm</td>
<td>0.12:81</td>
<td>0.04:33</td>
<td>0.08:75</td>
<td>0:12:72</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expint</td>
<td>4:56:17</td>
<td>0.34:33</td>
<td>0:08:26</td>
<td>0:13:25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fac</td>
<td>0.01:43</td>
<td>0.01:05</td>
<td>0:01:02</td>
<td>0:01:35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fact</td>
<td>0:02:43</td>
<td>0:07:10</td>
<td>0:08:26</td>
<td>0:41:22:70</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fbscall</td>
<td>0.01:04</td>
<td>0.01:09</td>
<td>0.01:08</td>
<td>0:01:52</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fl</td>
<td>762494</td>
<td>131508</td>
<td>2:13:54</td>
<td>18763</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insertion</td>
<td>0.01:26</td>
<td>0.01:20</td>
<td>0:01:31</td>
<td>37816</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>janne_complex</td>
<td>0.01:17</td>
<td>0.01:17</td>
<td>0:01:33</td>
<td>1853</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jocint</td>
<td>54383</td>
<td>293957</td>
<td>133059</td>
<td>111510</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mattmult</td>
<td>52303</td>
<td>583449</td>
<td>2636715</td>
<td>712809</td>
<td>OOM</td>
<td>error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ns</td>
<td>0:01:79</td>
<td>0:02:20</td>
<td>0:03:43</td>
<td>0:07:09</td>
<td>10813</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prime</td>
<td>170806</td>
<td>6477669</td>
<td>470372</td>
<td>469060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>std</td>
<td>0:12:52</td>
<td>1:02:36</td>
<td>1:56:36</td>
<td>2:41:87</td>
<td>168174</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0
- **Value analysis fails:** 0

- **3 errors / 19 benchmarks**
- **0 errors / 7 1 benchmarks**
- **1 errors / 21 benchmarks**
- **2 errors / 21 benchmarks**
- **3 errors / 21 benchmarks**
- **2 errors / 21 benchmarks**
- **3 errors / 21 benchmarks**
- **4 errors / 21 benchmarks**
- **5 errors / 21 benchmarks**
- **6 errors / 21 benchmarks**
- **7 errors / 21 benchmarks**
- **8 errors / 21 benchmarks**
- **9 errors / 21 benchmarks**
Program Slicing, M. Weiser [Weiser, 1984]

I = set of instructions in P

Slicing

- slice criterion: subset I' ⊆ I and variables V(i) for each i ∈ I'

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

- Slice of P = sub-program P' of P satisfying (1) and (2)
  - given input d ∈ D,
  - run of P on d: ϱ = (i₀, v₀) (i₁, v₁) ... (iₖ, vₖ) ... (iₙ, vₙ)
  - run of P' on d: ϱ' = (i₀', v₀') (i₁', v₁') ... (iₖ', vₖ') ... (iₙ', vₙ')

- projection: for a pair (i, v), proj(i, v) = \{ ε if i ∉ I' \\
  (i, projᵥ(i)(v)) otherwise
- for sequences of pairs: proj*(ε) = ε and proj*(w,a) = proj*(w),proj(a)

1. on input d ∈ D, if P terminates then P' terminates
2. on input d ∈ D, proj*(ϱ) = proj*(ϱ')

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

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

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 $d$ $\varrho = (i_0, v_0) \ (i_1, v_1) \ \ldots \ (i_k, v_k) \ \ldots \ (i_n, v_n)$
    - run of $P'$ on $d$ $\varrho' = (i'_0, v'_0) \ (i'_1, v'_1) \ \ldots \ (i'_l, v'_l) \ \ldots \ (i'_m, v'_m)$
  - projection: for a pair $(i, v)$, $\text{proj}(i, v) = \begin{cases} \varepsilon & \text{if } i \notin I' \\ (i, \text{proj}_{V(i)}(v)) & \text{otherwise} \end{cases}$
  - 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 [Weiser, 1984]

\( I = \) set of instructions in \( P \)

**Slicing**

- **slice criterion:** subset \( I' \subseteq I \) and variables \( \mathcal{V}(i) \) for each \( i \in I' \)
- **Slice of** \( P = \) 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 \( (i, v) \), \( \text{proj}(i, v) = \begin{cases} 
\varepsilon & \text{if } i \not\in I' \\
(i, \text{proj}_{\mathcal{V}(i)}(v)) & \text{otherwise}
\end{cases} \)
  - 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: ldr r2, [r1, #4] (32, r1)
- each conditional branch instruction is in C
  36: 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: 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 \( C \)
  
  \[ 32: \text{ldr } r2, \lbrack r1, \#4 \rbrack \] (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: ldr r2, [r1, #4] (32, r1)

- each conditional branch instruction is in C
  36: 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: 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.
Measuring Execution-Time on the ARM920T

```c
#define timerToCPUClockRatio 12

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

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

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

00003214 <fib>:
00003214: e24dd020 sub sp, sp, #32
00003218: e58d0004 str r0, [sp, #4]
0000321c: e3a03001 mov r3, #1
...
00003288: e12fff1e bx lr
0000328c <printbin>:
0000328c: e52de004 bx lr

000033a8 <timerSetPrescaler>:
000033a8: e52de004 push {lr}
000033ab: e24dd014 sub sp, sp, #20
000033c0: e58d0004 str r0, [sp, #4]
000033c8: e59d3004 ldr r0, [sp, #4]
000033cc: e3a01010 mov r1, #16
000033d0: ebffffcb bl 000033d0 <timerGetRegisterAddress>
000033d4: e1a00003 mov r3, r0
000033d8: e58d300c str r3, [sp, #12]
000033de: e59d300c ldr r3, [sp, #12]
000033e4: e5933000 ldr r3, [r3]
000033e8: e58d3008 str r3, [sp, #8]
000033ec: e59d3008 ldr r3, [sp, #8]
000033f0: e1a00003 mov r0, r3
000033f4: e28dd014 add sp, sp, #20
000033f8: e49df004 pop (pc)

000033e4 <timerGetValue>:
000033e4: e52de004 push {lr}
000033e7: e24dd014 sub sp, sp, #20
000033ec: e58d0004 str r0, [sp, #4]
000033f0: e59d3004 ldr r0, [sp, #4]
000033f4: e3a01010 mov r1, #16
000033f8: ebffffcb bl 000033d4 <timerGetRegisterAddress>
000033fc: e1a03000 mov r3, r0
00003400: e58d300c str r0, [sp, #4]
00003404: e59d300c ldr r0, [sp, #4]
00003408: e5933000 ldr r0, [r0]
0000340c: e58d3008 str r0, [sp, #8]
00003410: e59d3008 ldr r0, [sp, #8]
00003414: e1a00003 mov r0, r3
00003418: e28dd014 add sp, sp, #20
0000341c: e49df004 pop (pc)

000033d0 <timerGetRegisterAddress>:
000033d0: e59d3000 ldr r3, [sp]
000033d4: e2833602 add r3, r3, #2097152 ; 0x2000000
000033d6: e2833a02 add r3, r3, #8192 ; 0x2000
000033de: ea0e0000 beq 000033d4 <timerGetRegisterAddress+0x34>
000033e2: e59d3000 ldr r3, [sp]
000033e8: e2833602 add r3, r3, #2097152 ; 0x2000000
000033ee: e2833a03 add r3, r3, #12288 ; 0x3000
000033f6: e1a00003 mov r0, r3
000033f8: e28dd008 add sp, sp, #8
000033fa: e12fff1e bx lr

000033f6 <timerInit>:
000033f6: e52de004 bx lr

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

ACSD’2011 (Newcastle, UK, June 2011) Timed Games for Computing WCET
Compiled Program

...  
0004d44: ebfff9a6  bl  00033e4  <timerGetValue>
0004d48: e1a03000  mov  r3,  r0
0004d4c: e58d3004  str  r3,  [sp,  #4]
0004d50: e3a0f4b   mov  r0,  #300
0004d54: ebfff92e  bl  0003214  <fib>
0004d58: e1a03000  mov  r3,  r0
0004d5c: e58d3000  str  r3,  [sp]
0004d60: e3a00001  mov  r0,  #1
0004d64: ebfff99e  bl  00033e4  <timerGetValue>
0004d68: e1a03000  mov  r3,  #3
...
Interval in Execute Stage

\[ t \leq DUR_{MAX\_INSTR} \]

\[ memory! \]

\[ copy(me,me+1) \]

\[ execute\_completed! \]

\[ t \geq DUR_{MIN\_INSTR} \]
\[ \land t \leq DUR_{MAX\_INSTR} \]
\[ \text{execute?} \]
\[ t=0, DUR_{MAX\_INSTR}=\text{max\_dur}(), \]
\[ DUR_{MIN\_INSTR}=\text{min\_dur}() \]

\[ t \leq DUR_{MAX\_INSTR} \]

\[ t \geq DUR_{MIN\_INSTR} \]
\[ \land t \leq DUR_{MAX\_INSTR} \]

\[ decode\_completed? \]

\[ execute\_completed! \]
### Timed Games for Computing WCET

#### Qt GUI

<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>e243300a</td>
<td>sub</td>
<td>r3, r3, #10</td>
<td>-</td>
<td>r3</td>
<td>r3</td>
<td>def(r3)=[r3]</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_000334c [13132]</td>
<td>e583d000</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003350 [13136]</td>
<td>e59d2004</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>-</td>
<td>-</td>
<td></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>r3</td>
<td>def(r3)=[]</td>
<td>-</td>
<td>-</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_000335c [13148]</td>
<td>e1520003</td>
<td>cmps</td>
<td>r2, r3</td>
<td>-</td>
<td>r2, r3</td>
<td>le</td>
<td>def(le)=[r2, r3]</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003360 [13152]</td>
<td>daaffff0</td>
<td>ble</td>
<td>0003328</td>
<td>&lt;complex+0x74&gt;</td>
<td>le</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003366 [13156]</td>
<td>e3a03001</td>
<td>mov</td>
<td>r3, #1</td>
<td>-</td>
<td>r3</td>
<td>r3</td>
<td>def(r3)=[r3]</td>
<td>-</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>r0</td>
<td>def(r0)=[r3]</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_000336c [13164]</td>
<td>e28dd008</td>
<td>add</td>
<td>sp, sp, #8</td>
<td>-</td>
<td>sp</td>
<td>sp</td>
<td>def(sp)=[sp]</td>
<td>-</td>
<td>-</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003374 [13172]</td>
<td>e52de004</td>
<td>stdmdb</td>
<td>spl, [lr]</td>
<td>(str lr, [sp, #4])</td>
<td>lr, sp</td>
<td>sp, stack</td>
<td>def(sp)=[sp] def(stack)=[lr, 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>-</td>
<td>-</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>r3</td>
<td>def(r3)=[r3]</td>
<td>-</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003384 [13188]</td>
<td>e3a03001</td>
<td>mov</td>
<td>r3, #1</td>
<td>-</td>
<td>r3, sp</td>
<td>stack</td>
<td>def(stack)=[r3, sp]</td>
<td>-</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_000338c [13196]</td>
<td>e3a03000</td>
<td>mov</td>
<td>r3, #0</td>
<td>-</td>
<td>r3, sp</td>
<td>stack</td>
<td>def(stack)=[r3, sp]</td>
<td>-</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003394 [13204]</td>
<td>e59d0004</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>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>0x_0003398 [13208]</td>
<td>e59d1008</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>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
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


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

[21] Xianfeng Li, Yun Liang, Tulika Mitra, and Abhik Roychoudhury.
Chronos: A Timing Analyzer for Embedded Software.
Special Issue on Experimental Software and Toolkit.
References V

Combining Abstract Interpretation with Model Checking for Timing Analysis of Multicore Software.  

WCET Project – Benchmarks.  

Why model checking can improve wcet analysis.  
In Rajeev Alur and Doron Peled, editors, CAV, volume 3114 of LNCS, pages 334–347.  

TuBound – A Conceptually New Tool for WCET Analysis.  

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.

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

Program slicing.

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