Worst case energy consumption of programs at ISA and LLVM IR level

K. Georgiou$^1$, J. Morse$^1$, K. Eder$^1$,

$^1$University of Bristol

7th EACO Workshop

September 11, 2014

funded by
Overview

1. Mapping Technique
   - LLVM-IR/ ISA Mapping Example

2. WCEC 1st Approach

3. WCET
   - ISA Instruction Time Cost
   - Implicit Path Enumeration

4. WCEC 2nd Approach
   - Calculating WCEC bounds

5. Conclusion and Future Work

6. WCEC Energy Models
LLVM-IR/ISA Mapping Example

LoopBody:
\[
\text{%deref1 = load i32* %i2}
\text{store i32 %deref1, i32* %i1}
\text{br label %LoopTest2, dbg}
\]

LoopBody3:
\[
\text{%i3 = load i32* %numbers.bound}
\text{%deref6 = load [0 x i32]** %numbers}
\text{%deref7 = load i32* %i4}
\text{%btmp8 = sub i32 %deref7, 1}
\text{%subscript = getelementptr [0 x i32]** %deref6, i32 0, i32 %btmp8}
\text{%deref9 = load i32* %subscript}
\text{%i4 = load i32* %numbers.bound}
\text{%deref10 = load [0 x i32]** %numbers}
\text{%deref11 = load i32* %i4}
\text{%subscript12 = getelementptr [0 x i32]** %deref10, i32 0, i32 %deref1}
\text{%deref13 = load i32* %subscript12}
\text{%relcmp = icmp sgt i32 %deref9, %deref13}
\text{%cast = zext i1 %relcmp to i32}
\text{%zerocmp = icmp ne i32 %cast, 0}
\text{br i1 %zerocmp, label %iftrue, label %ifdone}
\]

ISA

.label10
0x000100da: 05 5c: ldw (ru6) r0, sp[0x5]
0x000100dc: 04 54: stw (ru6) r0, sp[0x4]
0x000100de: 20 73: bu (u6) 0x20 <.label5>

.label8
0x000100e0: 08 5c: ldw (ru6) r0, sp[0x8]
0x000100e2: 44 5c: ldw (ru6) r1, sp[0x4]
0x000100e4: 21 f8 ec 1f: ldaw (l3r) r2, r0[r1]
0x000100e8: 6a 9a: sub (2rus) r2, r2, 0x4
0x000100ea: 28 08: ldw (2rus) r2, r2[0x0]
0x000100ec: 01 48: ldw (3r) r0, r0[r1]
0x000100ee: 02 c0: lss (3r) r0, r0, r2
0x000100f0: 14 78: bl (ru6) r0, 0x14 <.label6>
0x000100f2: 00 73: bu (u6) 0x0 <.label7>
Approach 1 - Building Cost Functions

Original program

```c
int fact(int i) {
    if (i <= 0) {
        return 1;
    }
    return (i * fact(i - 1));
}
```

Extracted cost relations

- $C_{\text{fact}}(i) = C_a + C_b$ if $i \leq 0$
- $C_{\text{fact}}(i) = C_a + C_c(i)$ if $i > 0$
- $C_c(i) = C_d + C_{\text{fact}}(i - 1)$

- Substitute $C_a, C_b, C_d$ with actual energy required to execute low level instructions.
- Solve relations using off the shelf solvers to obtain closed form solution.
- Result: $C_{\text{fact}}(i) = 4563 + 7878i \, \text{pJ}$. 
Energy Measuring Set Up

- Arduino Uno Board to monitor current/power measurements
- I2C Interface Connection
- 5V supply
- INA219 Current/Power monitor
- GND
- Jumper connection to the supply voltage to the core
- I/O control pin
- Power supply
- Programming port
- XMO XTAG2 debug adapter
- 32-bit XCore XS1-SU1 64KB SRAM under energy monitoring
## Results

<table>
<thead>
<tr>
<th>Benchmark*</th>
<th>ARM (nJ)</th>
<th>XMOS (nJ)</th>
<th>Final error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>base64</td>
<td>$158 + 94 \cdot \left\lfloor \frac{P-1}{3} \right\rfloor$</td>
<td>$1270 + 734 \cdot \left\lfloor \frac{P-1}{3} \right\rfloor$</td>
<td>28.0 1.1</td>
</tr>
<tr>
<td>mac</td>
<td>$23P + 14$</td>
<td>$133P + 192$</td>
<td>-1.7 10.1</td>
</tr>
<tr>
<td>insertion sort</td>
<td>$25P^2 + 11P + 7.1$</td>
<td>$105P^2 + 30P + 75$</td>
<td>11.1 3.0</td>
</tr>
<tr>
<td>matrix multiply</td>
<td>$20P^3 + 13P^2 + 97P + 84$</td>
<td>$144P^3 + 200P^2 + 77P + 332$</td>
<td>-3.3 -3.4</td>
</tr>
<tr>
<td>jpegdct</td>
<td>$54 \text{ mJ}^{\dagger}$</td>
<td>$463 \text{ mJ}^{\dagger}$</td>
<td>8.5 2.6</td>
</tr>
</tbody>
</table>

**Mean relative error**

<table>
<thead>
<tr>
<th></th>
<th>ARM</th>
<th>XMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9.9</td>
<td>3.4</td>
</tr>
</tbody>
</table>
Worst Case Execution Time

How is this related to Energy Consumption?

- Is time and energy the same?
- Critical when optimizing between various resources
- Essential for time critical applications
- Essential for inferring Energy Consumption of concurrent programs
Bounding WCET

- Lower timing bound
- BCET
- Minimal observed execution time
- The actual WCET must be found or upper bounded
- Measured execution times
- Possible execution times
- Timing predictability

---

One ISA instruction needs 4 clock cycles to complete*

Cycle time

\[ T_{clk} = \frac{1}{F} \]

Instruction Time

\[ I_t = T_{clk} \times 4 \]

e.g. 400 MHz ⇒ \( I_t = 10\text{ns} \)

*up to four threads
• Division

• Communication Time is constant on the same core

• Communication Time btwn cores (Steve’s Network Modeling)

• Input output on ports time may vary
Implicit Path Enumeration (IPE)

- Very popular technique for WCET calculation

- It expresses the search of the WCET as an Integer Linear Programming problem where the execution time is to be maximized under some constraints on the execution counts of the basic blocks

- The worst case execution path is defined by the set of blocks with their respective execution counts but not the order in which they are executed
Constraints used to denote loop bounds and other path information that depend on the functionality of the program (Data Flow Analysis can minimize user input)

Minimum requirement from programmer to set the loop bounds

More functional constraints from the user can help to get tighter bounds
void jpegdct(short d[], short r[])
{
    long int t[12];
    int v=0;
    short i, j, k, m, n, p, ic, ik;
    for (ik=2; ik; ik--)
    {
        for (i = 8; i; i--, v+=8)
        {
            for (j = 3; j>=0; j--)
            {
                // some code
            }
            // some code
        }
        // some code
    }
}
1x1 <= x2  
(1)

x2 <= 2x1  
(2)

1x2 <= x3  
(3)

x3 <= 8x2  
(4)

1x3 <= x4  
(5)

x4 <= 4x3  
(6)
ILP Solving Example

\[ \text{max : } b_1 \times x_1 + b_2 \times x_2 + b_3 \times x_3 + b_4 \times x_4 + b_5 \times x_5 + b_6 \times x_6 + b_7 \times x_7 \]

\[
\begin{align*}
d_0 &= 1 \\
x_1 &= d_1 = d_0 \\
x_2 &= d_1 + d_9 = d_2 \\
x_3 &= d_2 + d_7 = d_3 \\
x_4 &= d_3 + d_5 = d_4 + d_5 \\
x_5 &= d_4 = d_6 + d_7 \\
x_6 &= d_6 = d_8 + d_9 \\
x_7 &= d_8 = d_{10} \\
d_{10} &= 1
\end{align*}
\]

- Solve this by `lp_solver`: standard linux pri-installed package
- Complexity: NP complete, although most of the cases it collapses to LP which can be solved in polynomial time
$f$-edges treated similar to $d$-edges

\[
x_1 = d_1 = f_1 \\
x_2 = f_1 = f_2 \\
d_2 = f_1 + f_2
\]
Worst Case Energy Consumption (WCEC)

Actual WCEC must be found or upper bounded.
Calculating WCEC bounds

- Using the same approach used for time to formulate and solve the problem by using ILP
- Replace the timing cost $B_i$ of the basic blocks on the CFG with their energy cost
- Solve the retrieved equation with its’ constraints by maximizing and minimizing it to get the upper and lower bounds of the energy consumption
- Currently retrieving CFGs and the ILP formulation and constraints automatically in both ISA and LLVM IR
## Experimental Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>HW VS ISA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mac</td>
<td>-2.5%</td>
</tr>
<tr>
<td>Matrix Mul</td>
<td>-1.1%</td>
</tr>
<tr>
<td>Levenshtein</td>
<td>-0.7%</td>
</tr>
<tr>
<td>Jpegdct</td>
<td>-2.3%</td>
</tr>
<tr>
<td>Base64</td>
<td>-2.2%</td>
</tr>
<tr>
<td>Radix4Division</td>
<td>45%</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

- WCET techniques can be used for WCEC
- If you need predictability make it predictable
  - predictable architecture
  - predictable software
- Compine this with abstract interpretation
- Try this with an arm Cortex M3
- Extend this to multi-threaded case
WCEC Models

- The program path length can scale with input data (and can be analysed by WCET)
- However, energy consumption of individual instructions can scale with input data too
- In Steve’s XCore energy model, \texttt{imul} can range from consuming energy at 131mW to 222mW, depending on the value of it’s operands
- Assuming the worst-case consumption for each instruction is safe, but not a particularly tight approximation.
int fir(int xn, int coeffs[], int state[], int ELEMS) {
    int ynl = 0, ynh = (1<<23);
    for (j = ELEMS-1; i != 0; j--) {
        state[j] = state[j-1];
        {ynh, ynl} =
            maccs(coeffs[j], state[j], ynh, ynl);
    }
}

sub r6, r5, 0x1
ldw r7, r2[r6]
stw r7, r2[r5]
ldw r5, r1[r5]
maccs r0, r3, r5, r7
add r5, r7, 0x0
bt r6, -0xF <label>
• Verify that power consumption scales with the frequency of instructions operating on the data path
• Data flow or taint analysis to identify that data path
• Integration into ENTRA project toolchain
• Evaluation
Thank you!

kyriakos.georgiou@bristol.ac.uk
jeremy.morse@bristol.ac.uk
kerstin.eder@bristol.ac.uk