Performance Analysis of Embedded System Software

Tai-Yi Huang

Embedded & Operating System Lab
National Tsing Hua University
Introduction

- The importance of Worst-Case Execution Time (WCET) in embedded systems
- Simple architecture
  - instruction execution time is fixed
- Cached architecture
  - two possible execution times – cache & miss
- Advanced architecture
  - an instruction has many execution times
Outline

- Machine Model
- DMA I/O & Execution Times
- Integer Linear Programming (ILP) Method
- Iterative ILP Method
- Experimental Results
- Conclusions
DMA I/O

- DMA controller (DMAC) operates in either
  - burst mode
  - cycle-stealing mode
- Burst mode: DMAC will not release the bus until all data transfers are done
- **Cycle-stealing**: DMAC transfers data by stealing bus cycles from CPU
### Pipeline & DMA I/O

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>E</td>
<td>B</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EX</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>WB</td>
<td></td>
<td></td>
<td>B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The schedule indicates a sequence of operations across the pipeline stages with specific stages marked as "E" or "B".
Cycle-Stealing DMA I/O
Bounding the Delay

As the following figure, we can use these two formulas to calculate the delay time

\[ m = \left\lfloor \frac{T_k - BMT}{DT} \right\rfloor \quad \Rightarrow \quad d = m \times DT + 2 \times BMT - T_k \]
Outline

- Machine Model
- DMA I/O & Execution Times
- Integer Linear Programming (ILP) Method
- Iterative ILP Method
- Experimental Results
- Conclusions
Basic Block

**Basic block:** only have straight-line sequence of instructions

```java
int i=4, j=6;
if (j<=6)
    for (int k=0; k<5; k++)
        statements1;
else
    statements2;
    statements3;
```
Integer Linear Programming

- **W**: the execution time
- **X**: the execution count
- **e**: the edge weight

\[
\begin{align*}
X_1 &= 1 \\
X_1 &= e_1 + e_2 \\
X_2 &= e_1 = e_3 \\
X_3 &= e_2 = e_4 \\
X_4 &= e_3 + e_4 \\
\{e_1, e_2\} &= \{1,0\} \text{ or } \{0,1\}
\end{align*}
\]

\[
WCET = \sum_{i=1}^{4} (W_i \times X_i)
\]

\(ILP\)
Let $x_i$ be the number of times instruction $i$ is executed.

L1: if (A == 1)
L2: B=0;
L3: if (B!=0)
L4: B+=6;

integer linear constraints
X1=1
X2+X5=X1
X3=X2+X5
X4+X6=X3
(X2=1 && X4=0) || (X2=0)

Goal: $\max \left\{ \sum_{i=1}^{K} w_i x_i \right\} = \max (w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4)$

WCET $= \max (w_1 + w_2 + w_3, w_1 + w_3 + w_4)$
Iterative Integer Linear Programming

- Designed for **advanced architecture** where an instruction has many execution times
- Example: **block A** may have several possible execution times
Iterative Integer Linear Programming

Let us discuss cases which often used in a program: *if-then-else* and *for-loop*.
Iterative ILP - if-then-else

\[
X_{1,1}^{hit} = e_1 + e_2 + e_3 + e_6 \\
X_{1,1}^{miss} = e_3 + e_4 + e_7 + e_8 \\
e_1 + e_2 + e_3 + e_4 + e_5 + e_6 + e_7 + e_8 = l
\]
Iterative ILP - *for-loop*

\[
\begin{align*}
X_{1,1}^{\text{hit}} + X_{1,1}^{\text{miss}} &= e_{1,1}^{\text{hit}} + e_{1,2}^{\text{hit}} + e_{1,3}^{\text{hit}} + e_{1,4}^{\text{hit}} \\
&e_{1,1}^{\text{hit}} + e_{1,2}^{\text{hit}} = X_{2,1}^{\text{hit}} + X_{2,1}^{\text{miss}} \\
e_{1,3}^{\text{miss}} + e_{1,4}^{\text{miss}} &= X_{2,2}^{\text{hit}} + X_{2,2}^{\text{miss}} \\
&\vdots
\end{align*}
\]
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>$W_s/A_s$</th>
<th>$W_s^a/A_s$</th>
<th>$P_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sels</td>
<td>1.05</td>
<td>1.84</td>
<td>43%</td>
</tr>
<tr>
<td>Bubs</td>
<td>1.07</td>
<td>1.84</td>
<td>42%</td>
</tr>
<tr>
<td>Mtxm</td>
<td>1.03</td>
<td>1.64</td>
<td>37%</td>
</tr>
<tr>
<td>Mtx2</td>
<td>1.02</td>
<td>1.71</td>
<td>40%</td>
</tr>
</tbody>
</table>

### 4 line instruction cache

$W_s$: program executes with DMA I/O concurrently

$W_s^a$: program executes with DMA I/O without concurrently

$A_s$: program executes alone without DMA I/O

$P_s$: \[ P_s = \frac{W_s^a - W_s}{W_s^a} \times 100\% \]
Results on Different Cache Lines

- Sels
- Bubs
- Mtxm
- Mtx2

4 line
8 line
16 line
32 line
Conclusions

- Iterative ILP is a powerful method for analyzing embedded software performance.
- Iterative ILP can be applied to a variety of advanced architectures.
- Our research encourages inclusion of I/O instructions in hard-real-time embedded systems.