Towards Automated Generation of Time-Predictable Code

Daniel Prokesch, Benedikt Huber, Peter Puschner
{daniel,benedikt,peter}@vmars.tuwien.ac.at

Institute of Computer Engineering
Vienna University of Technology, Austria

July 8th, 2014
WCET 2014, Madrid

1 Part of this work was supported by the FP7-ICT Project 288008 Time-predictable Multi-Core Architecture for Embedded Systems (T-CREST), http://www.t-crest.org/
Motivation

Uncertainty in program execution time

Hardware
Duration of operations dependent on internal state

Software
Different inputs lead to paths of varying execution time
The Single-Path Approach

A proposed solution

- Singleton execution path
- Execution time stable w.r.t. varying input data
Benefits

…of single-path / low execution time jitter

► Reduce complexity of path analysis
► Increase predictability
► Control loops: performance and stability
► Security: Avoid timing-based side-channel attacks
The Single-Path Approach

Basic Idea

Eliminate input-data dependent control flow.

**Predicated execution**

(guard) instruction;

**If-conversion**

```plaintext
if (cond) {
    x = a + 1;
} else {
    x = b - 2;
}
```

( cond ) x = a + 1;
( !cond ) x = b - 2;
Treatment of loops:

```plaintext
while (cond) {
    // max N times
    { stmts;
    } finished := false;
    for i = 1 .. N {
        if (!cond) {
            finished := true;
            if (!finished) {
                stmts;
            }
        }
    }
}
```
Single-Path Transformation Rules

Set of rules to transform from high-level language to single-path sequence of predicated statements [Puschner et al., 2012]

- Clear conceptual understanding
- BUT:

  Not sufficient for implementation in a compiler backend!

This work

Single-Path Transformation from a low-level perspective as graph transformation on control-flow graphs
Control-flow Graph

Assumptions

- Nodes with at most two successors
- Branch condition
- Loops: assume reducible control flow graph
Control-flow Graph

Loop header tree

Control flow graph

Loop header tree
The Single-Path Graph Transformation

- **Admissible** paths in CFG: local loop bounds
- Predicated Nodes
- Semantic actions on predicates
  - Node may set predicate to (negated) branch condition
  - Edge may set/clear predicates
  - Edge may copy value of predicate

Control flow graph + local loop bounds $\xrightarrow{SPGT}$ Single-path graph + predicated execution
The Single-Path Graph Transformation

Example

Control flow graph

Single-path CFG
Acyclic regions

The RK algorithm

[Park and Schlansker, 1991] describe transformation in acyclic case

- Partitioning by Control Dependence
- Nodes with same set of CD: same predicate
- Predicate assignments at source of CD edges
- Initialization with false
- Topological sort order
Loops

Key observation: Represent loops compacted into a single node

- Forward control flow graph of loop nodes
- Extend by entry, exit node
- Inner loops compacted
- Proceed as in acyclic case
Loops

Example

Control flow graph

FCFG for loop with header \textit{a}

FCFG for loop with header \textit{b}
Loops

Input-data independent iteration count

- Entering header initializes predicates
- Input-data independent branch condition
- Former exit edge causes clearing of header predicate
- From then on all nodes are disabled
Composition of the SP-Graph

- Preorder traversal of the loop header tree
Experiments
Simulation Framework

1. Experimental validation of technique
2. Evaluate estimated execution cost

<table>
<thead>
<tr>
<th>Benchmark/Function</th>
<th>Max</th>
<th>SP</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm/upzero</td>
<td>96</td>
<td>125</td>
<td>1.30</td>
</tr>
<tr>
<td>compress/compress</td>
<td>3589</td>
<td>4200</td>
<td>1.17</td>
</tr>
<tr>
<td>cover/swi120</td>
<td>1570</td>
<td>2655</td>
<td>1.69</td>
</tr>
<tr>
<td>expint/expint</td>
<td>1984</td>
<td>2736</td>
<td>1.38</td>
</tr>
<tr>
<td>fir/fir_filter_int</td>
<td>2618</td>
<td>3236</td>
<td>1.24</td>
</tr>
<tr>
<td>insertsort/main</td>
<td>750</td>
<td>832</td>
<td>1.11</td>
</tr>
<tr>
<td>janne_complex/complex</td>
<td>1135</td>
<td>1381</td>
<td>1.22</td>
</tr>
<tr>
<td>lcdnum/num_to_lcd</td>
<td>36</td>
<td>190</td>
<td>5.28</td>
</tr>
<tr>
<td>lms/main</td>
<td>103839</td>
<td>150209</td>
<td>1.45</td>
</tr>
<tr>
<td>ludcmp/ludcmp</td>
<td>328187</td>
<td>415544</td>
<td>1.27</td>
</tr>
<tr>
<td>minmax/main</td>
<td>70</td>
<td>83</td>
<td>1.19</td>
</tr>
<tr>
<td>minver/minver</td>
<td>197066</td>
<td>371563</td>
<td>1.89</td>
</tr>
<tr>
<td>qsort-exam/sort</td>
<td>7253</td>
<td>12004</td>
<td>1.66</td>
</tr>
<tr>
<td>qurt/qurt</td>
<td>1838</td>
<td>2208</td>
<td>1.20</td>
</tr>
<tr>
<td>select/select</td>
<td>7196</td>
<td>11219</td>
<td>1.56</td>
</tr>
<tr>
<td>statemate/FH_DU</td>
<td>305</td>
<td>360</td>
<td>1.18</td>
</tr>
</tbody>
</table>
Summary

- Single-path conversion as graph transformation
- Single-path graph with predicated execution model
- Validated technique
- Amenable for implementation in a compiler backend
- Automated generation of predictable machine code

http://github.com/t-crest/