Automatic Custom Instruction Generation: Phased Approach

• Challenge: HUGE design space!

• Phased approach
  – Local generation: Focus on small kernels ("hot-spots") in the code and generate custom instructions
  – Re-use: Identify opportunities for re-use in other parts of the program
  – Global selection: Select best combination of custom instructions for the entire program
Local Generation of Custom Instructions

- General idea
  - Identify **custom instruction candidates** at the BB level
  - Evaluate (rank) candidates


Custom Instruction Template Generation

Template: Sub-program that is a candidate for creating a custom instruction

c = a & 0xff; // node 1
b = b & 0xff + c; // node 2
e = d << 24; // node 3
g = f & 0xff00; // node 4
Example Illustration of Template Generation

c = a & 0xff;    // node 1
\[d = b & 0xff + c;\]    // node 2
\[e = d << 24;\]    // node 3
\[g = f & 0xff00;\]    // node 4
Example Illustration of Template Generation

c = a & 0xff;       // node 1

d = b & 0xff + c;  // node 2

e = d << 24;       // node 3

g = f & 0xff00;    // node 4
Example Illustration of Template Generation

c = a & 0xff; // node 1
\[ d = b \& 0xff + c; \] // node 2
\[ e = d \ll 24; \] // node 3
\[ g = f \& 0xff00; \] // node 4
Example Illustration of Template Generation

c = a & 0xff;       // node 1
\[\text{Basic templates}\]
d = b & 0xff + c;   // node 2
\[\text{Dependent templates}\]
e = d << 24;       // node 3
\[\text{Independent templates}\]
g = f & 0xff00;    // node 4
Reducing the search space

• A graph with N vertices has $2^N$ sub-graphs
  - Key observation: In reality, only a small fraction of sub-graphs are feasible for use as custom instructions

• How do we eliminate infeasible sub-graphs?
  - Based on a violation of the **output port** constraint
  - Based on a violation of the **convexity** constraint

---

Based on a violation of the **output port** constraint

Based on a violation of the **convexity** constraint

Pozzi et al., TCAD 2006
Efficiently Generating Feasible Set of Templates

- Given a graph, identify unique sub-graphs $S_j$ that satisfy the following conditions
  - Number of inputs of $S_j < N_{in}$
  - Number of outputs of $S_j < N_{out}$
  - $S_j$ is convex

Pozzi et al., TCAD 2006
Efficiently Generating the Feasible Set of Templates

• Nodes are numbered based on a reverse topological sort
  – Source nodes follow destination nodes

• Encode templates as a bit-vector
  – Each bit represents whether a node is in the template

• Key idea
  - Perform a branch-and-bound search to discover templates that satisfy convexity and input/output port constraints
Efficiently Generating the Feasible Set of Templates

Nodes are numbered based on a topological sort (source nodes follow destination nodes)

Example: $N_{out} = 1$

Output port constraint not satisfied!
Efficiently Generating the Feasible uned Set of Templates: Results

Number of sub-graphs generated using an output port constraint of two

Pozzi et al., TCAD 2006
Reducing Search Space: Pruning Heuristics

- Eliminating illegal templates still leaves us with a large number of candidates
  - When internal registers are allowed, no hard restriction on I/O

Sun et al., TCAD 2004
Overcoming Input/Output Port Constraints

- Example: $N_{in} = 2$, $N_{out} = 1$

Program using custom instructions

\[ \text{write } _\text{custom } _\text{reg}(1, \text{In}_3); \]

\[ \text{Out}_1 = \text{CustomInstr}(\text{In}_1, \text{In}_2); \]

\[ \text{Out}_2 = \text{read } _\text{custom } _\text{reg}(2); \]
Reducing Search Space: Pruning Heuristics

- Eliminating illegal templates still leaves us with a large number of candidates
  - When internal registers are allowed, no hard restriction on I/O
- Need to focus effort on the most promising candidates

Sun et al., TCAD 2004