YARDstick - custom processor development toolset

YARDstick (http://www.nkavvadias.com/yardstick) is a design automation tool for custom processor development flows that focuces on the hard part: generating and evaluating application-specific hardware extensions. YARDstick is a powerful building block for ASIP (Application-Specific Instruction-set Processor) development, since it integrates application analysis, ultra-fast algorithms for custom instruction generation and selection with user-defined compiler intermediate representations. It integrates retargetable compiler features for the targeted IRs/architectures.

1. Features

Important features of YARDstick are as follows:

- retargetable to used-defined IRs by machine description
- can be targeted to low-level compiler IRs, assembly-level representations of virtual machines, or assembly code for existing processors
- fully parameterized custom instruction generation and selection engine
- code selector for multiple-input multiple-output patterns
- virtual register assignment for virtual machine targets
- an extensive set of backends including:
  - assembly code emitter
  - C backend
  - visualization backend for Graphviz (http://www.graphviz.org)
  - visualization backend for VCG (http://rw4.cs.uni-sb.de/~sander/html/gsvcg1.html) (or aiSee: http://www.absint.com/aisee/)
  - an XML format backend amenable to graph rewriting.

YARDstick comes along with a cross-platform GUI written in Tcl/Tk 8.5 (http://wiki.tcl.tk).

2. Aim

The ultimate goal of YARDstick is to liberate the designer's development infrastructure from compiler and simulator idiosyncrasies. With YARDstick, the ASIP designer is empowered with the freedom of specifying the target architecture of choice and adding new implementations of analyses and custom instruction generation/selection methods.

Typically, 2x to 15x speedups for benchmark applications (ANSI C optimized source code) can be fully automatically obtained by using YARDstick depending on the target architecture. Speedups are evaluated against a scalar RISC architecture.

3. Detailed look

1. Analysis engines generating both static and dynamic statistics:
   - Data types
   - Operation-level statistics
   - Basic block statistics (ranking)
   - Performance estimations with/without custom instructions.
2. Generation of CDFGs (Control-Data Flow Graphs).
3. Backend engines:
   - ANSI C
   - dot (Graphviz)
   - VCG (GDL, aiSee)
   - XML (GGX for the AGG graph rewriting tool): http://www.user.tu-berlin.de/o.runge/AGG/
   - Retargetable assembly emitter for entire translation units
   - CDFG formats for various RTL synthesis tools
4. Custom instruction engines:
   - Full-parameterized MIMO custom instruction generation algorithm
   - Fast heuristic
   - Configurable number of inputs
   - Configurable number of outputs
5. Custom instruction selection:
   - Based on priority metrics
6. Graph (and graph-subgraph) isomorphism features for eliminating redundant patterns. Multiple algorithms supported.
7. Visualization of custom instructions, basic blocks, control-flow graphs and control-data flow graphs (basic block nodes expanded to their constituent instructions).

8. Basic retargetable compiler features:
    - Code selector for MIMO instructions (tested with large cases).
    - Virtual register assignment (allocation for a VM).
    - Hard register allocator in the works.
9. Miscellaneous features:
    - single constant multiplication optimizer
    - elimination of false data-dependences in assembly-level CDFGs.
    - beautification options for visualization
    - interfacing (co-operation) with external tools such as peephole optimizers, profilers, code generators etc.
    - features related to the custom processor architecture named ByoRISC: (http://arxiv.org/abs/1403.6632)

4. Benchmarks

Here is a list of application benchmarks that have been tested with YARDstick:

- ADPCM encoder and decoder (typically: 4x speedup)
- Video processing kernels: full-search block-matching motion estimation, logarithmic search motion estimation, motion compensation
- Image processing kernels: steganography (hide/uncover), edge detection, matrix multiplication
- Cryptographic kernels: crc32, rc5, raiden (7x speedup, 12x for unrolled version)

At the YARDstick homepage (http://www.nkavvadias.com/yardstick/) you can find some additional materials:

- A0-poster
- 2-page brochure
- a more extended presentation on YARDstick