YARDstick

Automation tool for custom processor development

Nikolaos Kavvadias and Spiridon Nikolaidis

Section of Electronics and Computers, Electronics Lab, Dept. of Physics, Aristotle Univ. of Thessaloniki, Greece

E-mail: *nkavv@physics.auth.gr*

What is YARDstick?

… a building block for ASIP development, integrating *application analysis*, custom *instruction generation*, *selection* and *synthesis* with user-defined (compiler) IRs

- YARDstick provides a compiler- and simulator-agnostic infrastructure
- Separates design space exploration from compiler/simulator idiosyncrasies
- Different compilers/simulators can be plugged-in
- Both high- (ANSI C) and low-level (assembly for a machine/VM) input can be analyzed

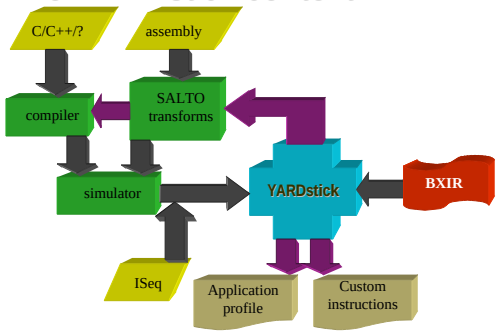
What is the need for YARDstick?

- ASIP development flows for contemporary embedded SoCs involve
  - –Architecture specification, design space exploration, profiling
  - –Exploitation of results: custom instruction (-set extension) generation and synthesis
  - –Toolchain generation/retargeting
- Certain practical issues in ASIP development are often neglected
  - –Assumptions of the IR **do affect** the solution quality
  - –The exploration infrastructure tied up to the conventions of certain SW development tools
  - –Support for low-level entry is some times desirable (reverse engineering, legacy code porting)

YARDstick has been designed with these issues in mind

YARDstick

1) The YARDstick context

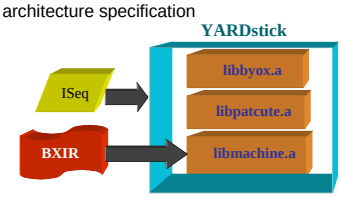


2) YARDstick architecture

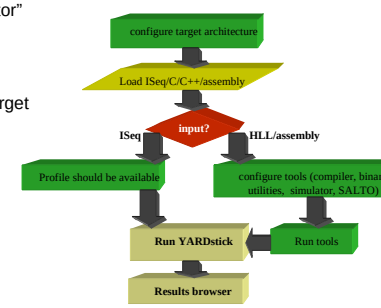
libByoX: "Bring your own Compiler-and-Simulator" kernel (target-independent)

libPatCUTE: Pattern-based Custom UniT Exploration (target-independent)

libmachine: Retargeted by a BXIR (ByoX IR) target architecture specification

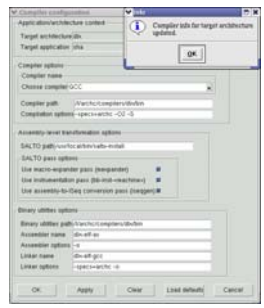


3) Using YARDstick



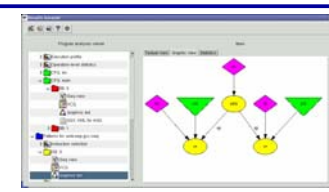
Screenshots

The main YARDstick screen

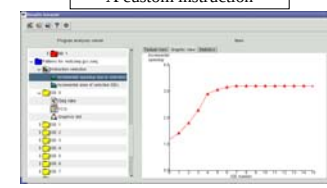


Compiler/binary tools configuration

Samples of dynamic statistics results



A custom instruction



Incremental speedup due to selected CIs

Wrap-up

- YARDstick is a retargetable application analysis and custom instruction generation/selection environment suitable for use in ASIP development
- Sample architecture targets: SUIFvm (basic/enhanced), DLX (integer subset)
- Six backends for exporting basic blocks, CFGs and CIs:
  - **ISeq (native YARDstick format)**
  - **VCG, Graphviz dot for visualization**
  - **GGX XML for graph transformations**
  - **CDFG format for translation to synthesizable RTL VHDL, ANSI C**
- Can process input in C/C++/assembly code

Acknowledgements

This work was supported by the General Secretariat of Research and Technology of Greece and the European Union.

References

[1] CDFG toolset. http://poppy.snu.ac.kr/CDFG/cdfg.html
[2] The AGG homepage. http://tfs.cs.tu-berlin.de/agg/
[3] GCC. http://www.gcc.org
[4] The COINS project. http://www.coins-project.org
[5] SALTO. http://www.irisa.fr/caps/projects/Salto/
[6] MachSUIF. http://www.eecs.harvard.edu/hube/research/machsuif.html
[7] The ArchC resource center. http://www.archc.org