YARDstick

Automation tool for custom processor development

Nikolaos Kavvadias and Spiridon Nikolaidis
Electronics Lab, Dept. of Physics, Aristotle Univ. of Thessaloniki, Greece
Webpage: http://electronics.physics.auth.gr/people/nkavv/yardstick/
E-mail: *nkavv@physics.auth.gr*

What is YARDstick?

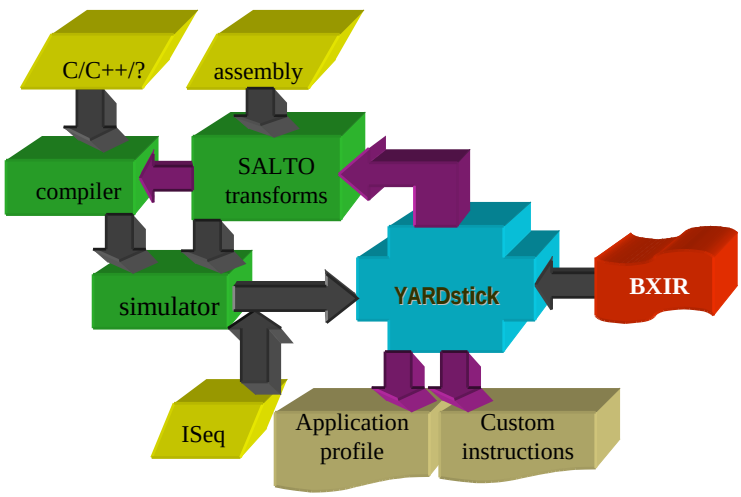
… a building block for ASIP development, integrating *application analysis*, custom *instruction generation*, *selection* and *synthesis* with user-defined (compiler) IRs

- YARDstick is a retargetable application analysis and custom instruction generation/selection environment providing a compiler-/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 an architecture/VM) input can be analyzed

- YARDstick deals with these issues of contemporary ASIP development flows
  - Assumptions of the IR affecting solution quality
  - Exploration infrastructure tied up to conventions of the SW development tools
  - Support for low-level entry

- Architecture targets: SUIFvm variants, integer DLX
- 6 backends for exporting basic blocks, control-flow graphs and custom instructions:
  - ISeq (native format)
  - VCG, Graphviz dot for visualization
  - GGX XML for graph transformations
  - CDFG format for translation to synthesizable RTL VHDL, ANSI C

A YARDstick environment

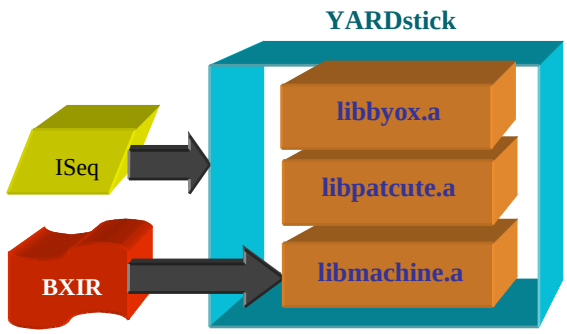


YARDstick components

libByoX: “Bring Your Own Compiler-and-Simulator” kernel (target-independent)

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

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



An example session

1) Configuration for the target architecture and the application program of interest
2) C/”assembly”/ISeq file entry
3) Run the compiler/simulator (C/assembly applications)
4) Load the resulting ISeq file (omit steps 2-3 if done on step 2)
5) Select backends, custom instruction generation/selection and additional options of your choice
6) View results within the Results Browser

The technology behind YARDstick

- libByoX implements the core YARDstick API and provides frontends/manipulators for internal data structures
- libPatCuTE implements instruction generation and selection
- Custom instruction (CI) generation methods:
- MaxMISO (maximal subgraphs with 1 output node)
- MISO exploration under user-defined constraints
- Fast MIMO (multiple-input, multiple-output) method
- Identification of redundant entire/partial CIs
- Greedy instruction selection for cycle-gain, cycle-gain-per-area priority metrics
- Custom instruction/functional unit synthesis is supported by external tools (CDFG toolset)

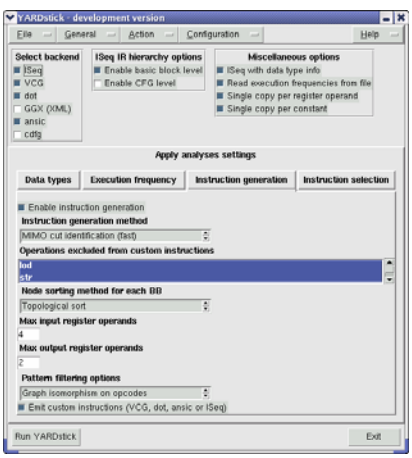
Capabilities

- Program analysis for C/assembly/ISeq applications
- Data type and operation-level statistics
- Basic block execution frequencies for identifying hotspots in applications
- Visualization (VCG, Graphviz) of BBs, CFGs and CIs
- Export to the GGX XML format which is supported by the AGG attributed graph transformation system
- ANSI C output (BBs, CIs) for use in simulators
- Application speedup due to CIs and CI selection analysis
- Interoperability with GCC and ArchC

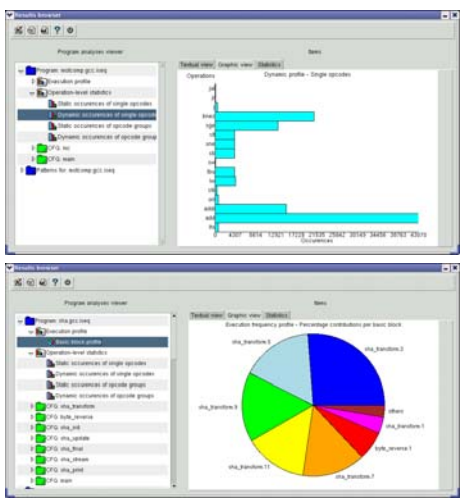
Sample benchmarks

Abbreviation	Application description	Reference	Entry in
crc32	Cyclic redundancy check	MiBench	ANSI C
deraiden	Decoding raiden cipher	raiden-cipher @ sourceforge	ISeq
enraiden	Encoding raiden cipher	raiden-cipher @ sourceforge	ISeq
idea	IDEA cryptographic kernel	--	ANSI C
sha	Secure Hash Algorithm producing an 160-bit message digest for a given input	MiBench	ANSI C
adpcmdec	Adaptive Differential Pulse Code Modulation (ADPCM) decoder	MiBench	ANSI C, ISeq
adpcmenc	Adaptive Differential Pulse Code Modulation (ADPCM) encoder	MiBench	ANSI C, ISeq
fir	FIR filter	--	ANSI C
fsme	Full-search block-matching motion estimation	--	ANSI C, ISeq
motcomp	Motion compensation	--	ANSI C, Iseq

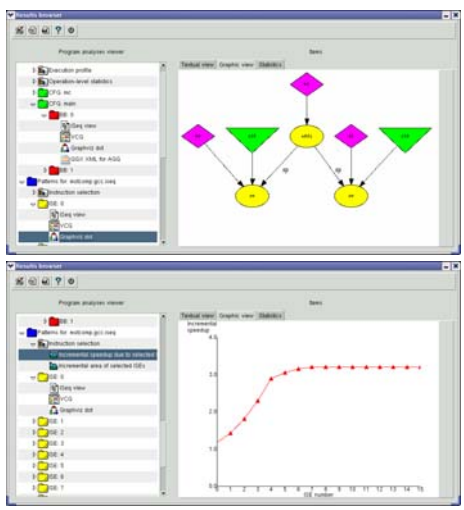
Screenshots



The main YARDstick screen



Samples of dynamic statistics results



Custom instruction analysis