

Ph.D. thesis research and beyond

Techniques and tools developed by Nikolaos Kavvadias
(2003-2011)

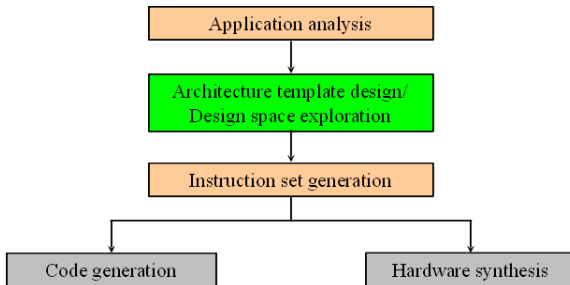
Nikolaos Kavvadias
`nikolaos.kavvadias@gmail.com`

Adjunct Lecturer,
Department of Computer Science and Technology,
University of Peloponnese, Tripoli, Greece

Thursday, 31 March 2011

Ph.D. thesis work: Development of an application-specific processor design methodology

- **Aim:** Contribution of novel methods, techniques and tools to the typical 5-stage ASIP (application-specific instruction-set processor) design flow:
 - **Orange:** New tools based on contemporary ideas/techniques
 - **Green:** Novel algorithms and/or architectures
 - **Gray:** Mainly used existing infrastructure and implemented some tools



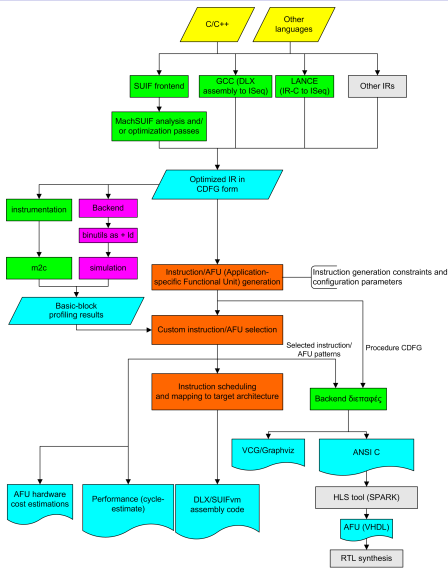
Highlights of my Ph.D. work (and beyond)

- 1 The YARDstick custom instruction generation tool
- 2 The ZOLC (Zero-Overhead Loop Controller) architecture for FSMD and programmable processors
- 3 The ByoRISC soft microprocessor family
 - The scalable register bypassing (SRB) architecture
 - Case study applications on ByoRISC
- 4 Post-Ph.D. work
 - A high-level synthesis tool under development
 - Teaching responsibilities at UOP, Greece

Highlights of my Ph.D. work (and beyond)

- 1 The YARDstick custom instruction generation tool
- 2 The ZOLC (Zero-Overhead Loop Controller) architecture for FSMD and programmable processors
- 3 The ByoRISC soft microprocessor family
 - The scalable register bypassing (SRB) architecture
 - Case study applications on ByoRISC
- 4 Post-Ph.D. work
 - A high-level synthesis tool under development
 - Teaching responsibilities at UOP, Greece

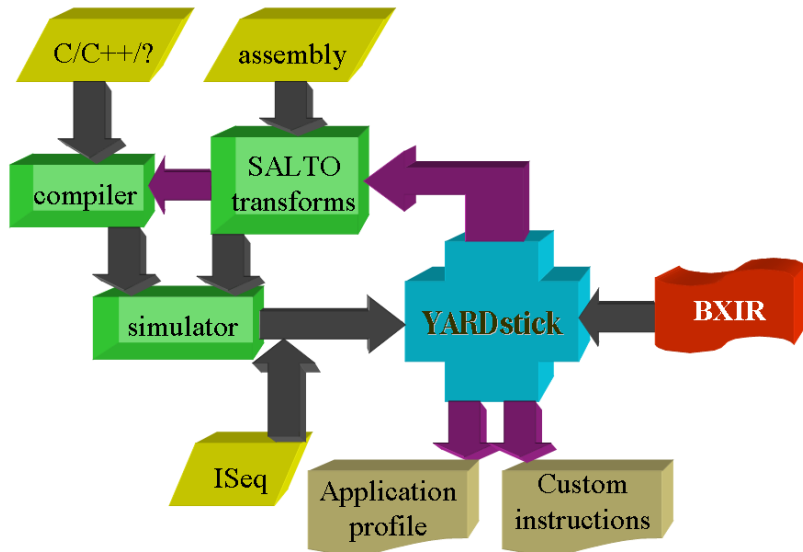
The [MELECON06] instruction generation flow



The YARDstick automation tool for custom processor development [DATE07]

- YARDstick is a building block for ASIP development, integrating application analysis, custom instruction generation, selection and synthesis with user-defined intermediate representations (compiler IRs)
- Based on previous work:
[PATMOS04, ASAP05, MELECON06]
- Provides a compiler- and simulator-agnostic infrastructure:
 - Separates design space exploration from compiler/simulator idiosyncrasies
 - Different compilers/simulators can be interfaced
 - Both high- (ANSI C) and low-level (e.g. DLX or SUIFvm assembly) input can be analyzed
- YARDstick components (static libraries)
 - libByoX: “Bring Your Own Compiler-and-Simulator” kernel
 - libPatCUtE: Pattern-based Custom UniT Exploration
 - libmachine: Retargeted by target architecture specification

Overview of the YARDstick framework



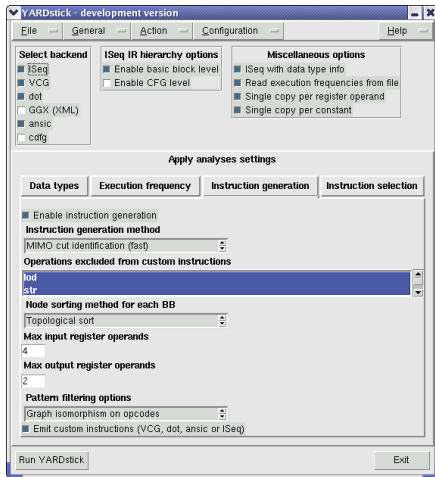
The YARDstick libraries

- libByoX implements the core YARDstick API and provides the IR (ISeg: a flat CFG form) frontend
- libPatCuTE implements instruction generation and selection
- Contemporary instruction generation methods are supported
 - MaxMISO (maximal subgraphs with a single-output node)
 - MISO exploration under user-defined constraints
 - Fast MIMO (multiple-input, multiple-output) method
- Greedy instruction selection for cycle-gain, cycle-gain-per-area priority metrics
- To extend with user-defined instruction generation methods, link to libPatCuTe.a
- libmachine provides manipulators for the BXIR (ByoX IR) target architecture specification format
- libmachine.a is the only YARDstick component that needs retargeting for a user-defined target architecture

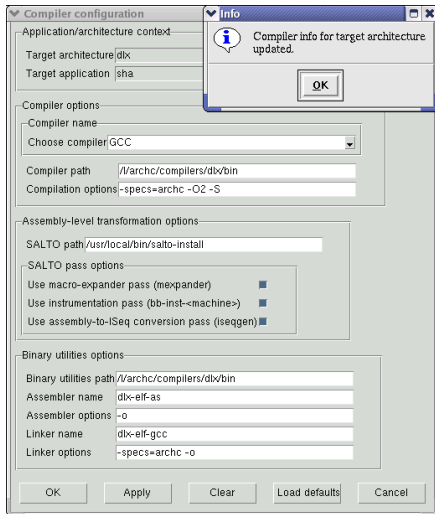
The development of YARDstick

- Initially only used with SUIF 2.2/Machine-SUIF 2.02.07.15 (prototype SUIFvm-like backend)
- On-and-off work since Sept. 2004 (ISeq frontend), heavy coding during Sept. 2006 - Apr. 2007
- Developed in C, C++, bison/flex, Tcl/Tk
- Current code base (mainline) around 20K lines (not accounting SALTO passes, GCC DLX backend etc)
- Some features require external tools/libraries:
 - Mandatory: boost (1.33.0), vlib2 (graph matching library)
 - Optional: ArchC 1.6.0, SystemC 2.0.1, SALTO 1.4.1beta3 (assembly language transformation framework), GCC, other compilers/simulators
- GUI (~5K) in Tcl/Tk 8.5.a5 (Tile, GRIDPLUS, other widgets)
- External tools as proof-of-concept (GCC/COINS DLX backends, DLX binutils/newlib/gdb port, simulators etc)

Screenshots from the YARDstick GUI (1)



Main screen

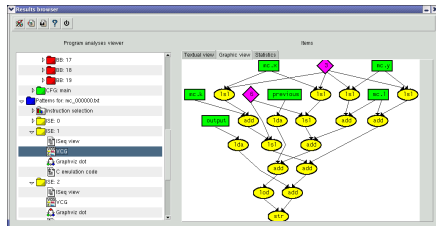


Tools configuration

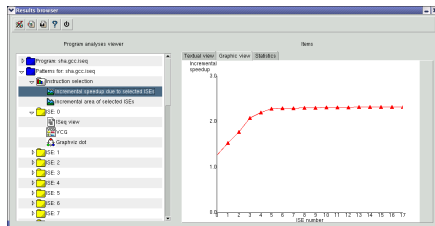
Screenshots from the YARDstick GUI (2)



Opcode utilization



A selected custom instruction (CI)



CIs speedup

Highlights of my Ph.D. work (and beyond)

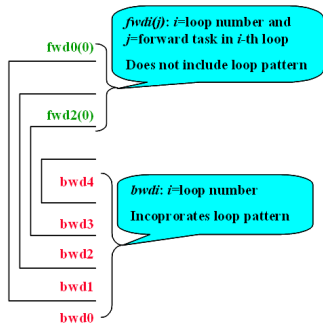
- 1 The YARDstick custom instruction generation tool
- 2 The ZOLC (Zero-Overhead Loop Controller) architecture for FSMD and programmable processors
- 3 The ByoRISC soft microprocessor family
 - The scalable register bypassing (SRB) architecture
 - Case study applications on ByoRISC
- 4 Post-Ph.D. work
 - A high-level synthesis tool under development
 - Teaching responsibilities at UOP, Greece

The Zero-Overhead Loop Controller (ZOLC) architecture [CDT05, TCOMP08]

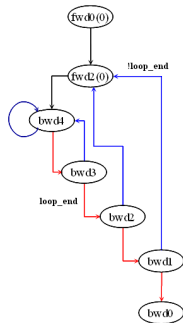
- **Problem:** Looping operations impose a significant performance bottleneck in embedded applications
 - Embedded DSPs use branch-decrement instructions and/or zero-overhead loop hardware (e.g. loop stacks)
 - Usually only special cases are handled (perfect loop nests, reducible control flow regions)
- **Solution:** The ZOLC, which eliminates the cycle overheads induced by looping
- The ZOLC concept has been incorporated into FSMMD architectures ([DSP02, CDT05]) and single-issue RISCs ([DATE05, ISCAS06, TCOMP08])
- Compared to embedded DSPs, the ZOLC supports:
 - Arbitrary number + any combination of loops in a nest
 - Multiple-entry/multiple-exit loops
 - Loop parameters (initial, final and stride index) determined at run-time

Task Control Flow Graph (TCFG) application representation

- TCFGs are extracted from procedure CDFGs
 - Comprised of DPTs: control-flow regions among loop boundaries
 - TCFG representation adopts generic control transfer expressions
 - TCFG node types: backward (including loop overhead instructions), forward (not including)



Loop nesting diagram

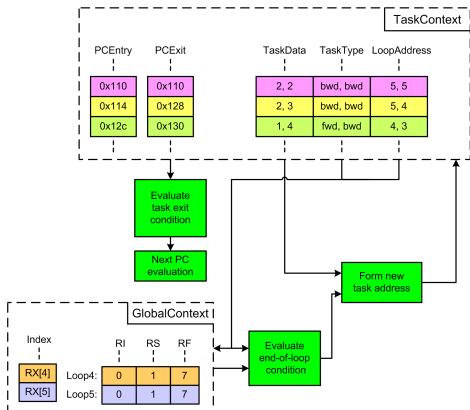


TCFG

Principles of ZOLC operation

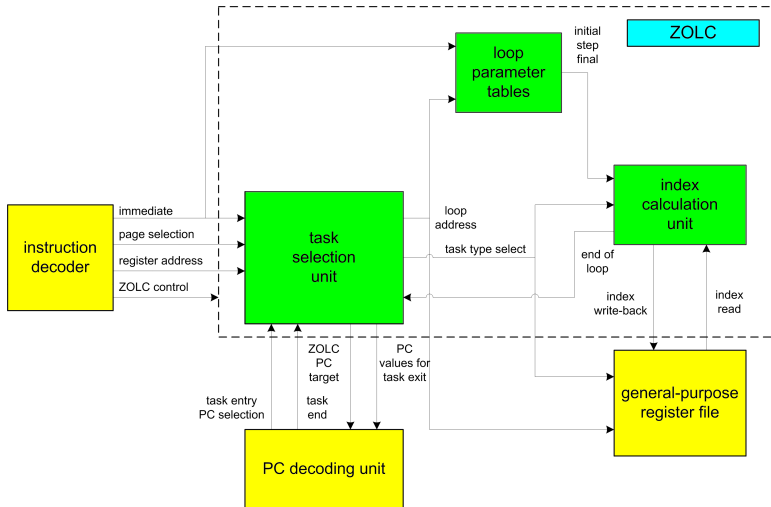
- ZOLC operation: provide a proper candidate PC address to the PC decoding unit for each substituted looping pattern
- Modes of operation
 - Initialization: ZOLC storage resources updated with static information
 - Active:
 - 1 Determine the following task
 - 2 Issue new PC and candidate exit values for multiple-exit loops
 - 3 Loop indices are updated and written back to register file
- Interface to the processor
 - PC decoding unit, instruction decoder, general-purpose register file
- ZOLC components
 - Task selection unit, loop parameters, index calculation unit

ZOLC operation for the block-based matrix multiplication algorithm



- The block-based matrix multiplication algorithm has three DPTs: fwd4(0), bwd4 and bwd5
- Actual numerical values are shown for the XiRisc soft-core
- The process is fully automated by a compilation flow including *tcfggen* (a MachSUIF pass) and *zolcgen* (a SALTO pass)

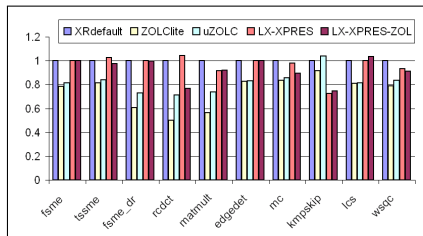
Incorporating the ZOLC architecture to programmable RISC processors



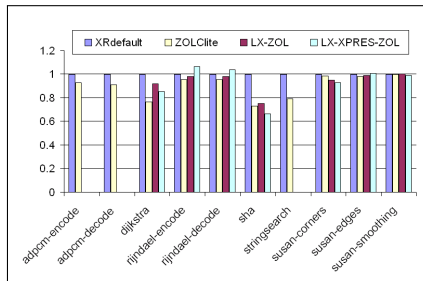
Applied to the XiRisc processor (RTL VHDL model)



Cycle performance results for the examined benchmarks



Kernel benchmarks



Mibench applications

- XRdefault: default XiRisc
- ZOLClite: 16 single-entry DPTs (8-loop structure) derived from a full-featured ZOLC
- uZOLC: Supports a single loop
- LX-ZOL: Xtensa-V4 processor with ZOL optimizations
- LX-XPRES-ZOL: Plus custom instructions
- Cycle speedup
 - 25.5% on kernel benchmarks
 - 10% on Mibench applications

Highlights of my Ph.D. work (and beyond)

- 1 The YARDstick custom instruction generation tool
- 2 The ZOLC (Zero-Overhead Loop Controller) architecture for FSMs and programmable processors
- 3 The ByoRISC soft microprocessor family
 - The scalable register bypassing (SRB) architecture
 - Case study applications on ByoRISC
- 4 Post-Ph.D. work
 - A high-level synthesis tool under development
 - Teaching responsibilities at UOP, Greece

The configurable processor concept

- Configurable/extensible processors can be tuned appropriately to speed up the execution of given applications
 - Configurability lies in tuning architectural parameters
 - Extensibility refers to instruction set extensions by incorporating application specific hardware through a tight or loose interface
- Commercial examples: ARC A4Tangent, Tensilica XTensa, the ARM OptimoDE and MIPS CorExtend technologies, Xilinx MicroBlaze, Altera Nios-II
- Limitations resulting in reduced speedups
 - Artificial restrictions due to compatibility with legacy architectures (the 3-operand instruction constraint)
 - Insufficient parameterization
 - Lack of support for both custom instruction (basic block level parallelism) and local coprocessor (function level parallelism) approaches

Development and design of a novel processor family

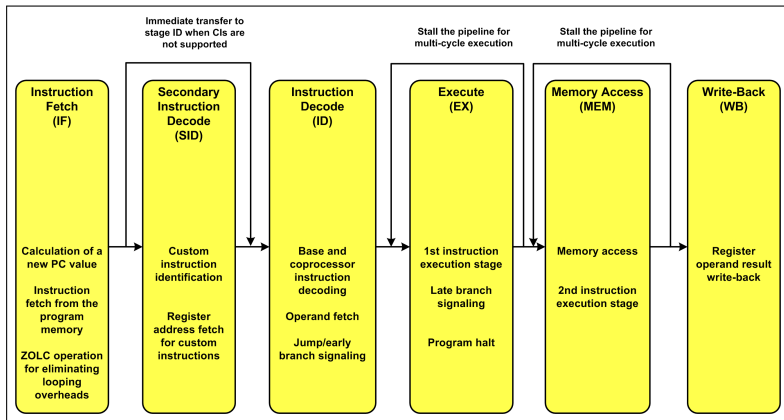
- Proposed solution: ByoRISC “Build your own RISC” configurable/extensible soft processors
- Base architecture encompasses characteristics of classical RISC architectures alongside a scalable extension architecture
- Configurable number of execution pipeline stages
- Support for multi-input, multi-output (MIMO) custom instructions without negatively affecting the ByoRISC instruction formats
- Tightly-coupled interface for custom functional units
- Configurable multi-port register file
- Custom interface to the data memory for memory accessing multi-cycle custom instructions
- Scalable data forwarding architecture

The ByoRISC architecture [VLSISOC08]

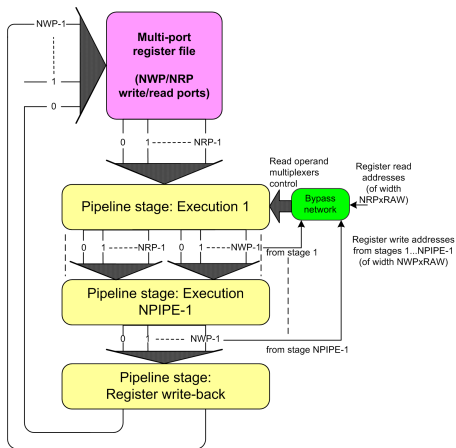


Microarchitectural organization of ByoRISC processors

Implemented synthesizable VHDL models and cycle-estimate ArchC simulators



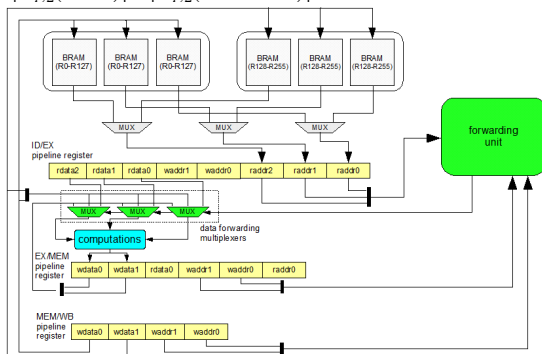
Scalable data forwarding architecture [MAM09]



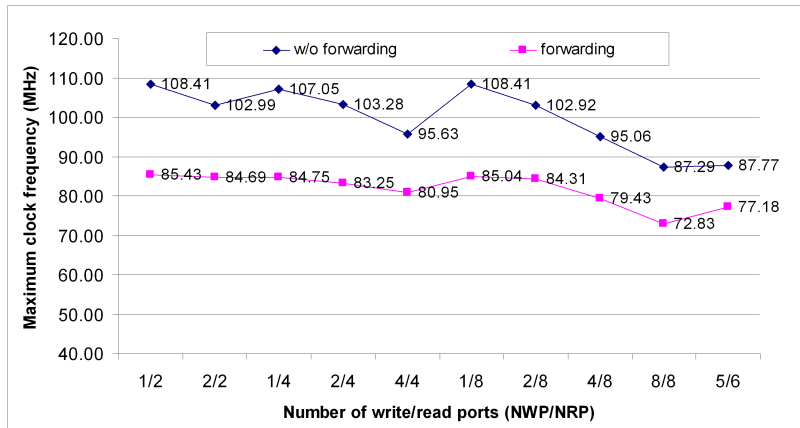
- Parameters of the SRB (Scalable Register Bypassing) architecture: **NPIPE** (Number of execution pipeline stages), **NWP/NRP** (Number of register file write/read ports)

A more detailed view of the scalable register bypassing architecture

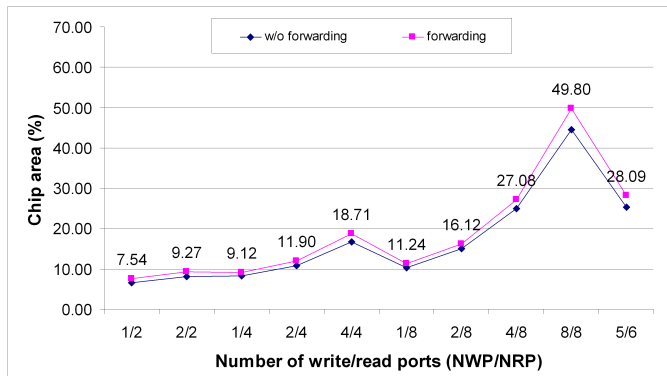
- The SRB requires:
 - NRP ($NPIPE \times NWP + 1$)-to-1 multiplexers in the first execution stage of the processor for selecting the proper forwarded datum per read port
 - $NRP \times NPIPE \times NWP$ comparators for evaluating the multiplexer control signals
 - Each of the first execution stage multiplexers require a control signal of width $\lceil \log_2(NWP) \rceil + \lceil \log_2(NPIPE + 1) \rceil$



Maximum clock frequency on Xilinx Virtex-4 FPGA (XC4VLX25)



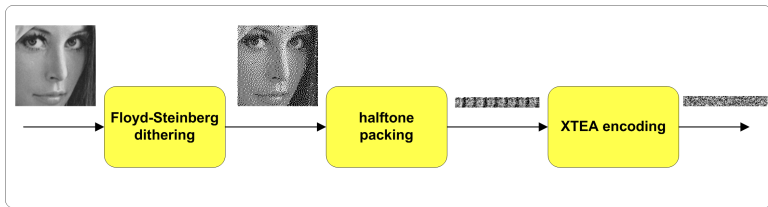
Chip area for Xilinx Virtex-4 FPGA (XC4VLX25)



$$Area = 100 - \left(\alpha \cdot (LUT_{max} - nLUT) / LUT_{max} + \beta \cdot (BRAM_{max} - nBRAM) / BRAM_{max} + \gamma \cdot (DSPX_{max} - nDSPX) / DSPX_{max} \right)$$

Case study: Image processing pipeline running on ByoRISC

- Evaluation of ByoRISC on a software pipeline consisting of image processing, data compression and cryptography kernels
- Applications:
 - fsdither: Floyd-Steinberg dithering by error diffusion to a bilevel image
 - htpack: halftone image packer for 8-fold lossless compression of a bilevel image
 - xteaenc: XTEA encryption



Custom instructions characteristics and applications speedup

CI DDG statistics												
CI	N_n	N_e	N_i	N_o	N_s	N_c	Cycle gain	Incremental speedup	SW cycles	HW cycles	HW latency	Area (MAU)
fsdither0	57	64	5	2	2	9	180224	2.69	57	13	12.38	4.56
fsdither1	9	6	3	2	2	2	32768	3.87	9	1	0.40	0.18
htpack0	78	82	3	2	2	14	35328	7.25	78	9	8.55	1.10
xteaenc0	38	38	6	5	1	7	540672	4.55	38	5	4.08	0.83

	fsdither	htpack	xteaenc
Initial cycles	250248	51421	619033
Cycles with CIs	74133	7183	158230
Speedup (sim.)	3.38	7.16	3.91
Speedup (estim.)	3.87	7.25	4.55
Cycles with CIs-ZOLC	37269	6671	141334
Speedup (sim.)	6.71	7.71	4.38

Highlights of my Ph.D. work (and beyond)

- 1 The YARDstick custom instruction generation tool
- 2 The ZOLC (Zero-Overhead Loop Controller) architecture for FSMD and programmable processors
- 3 The ByoRISC soft microprocessor family
 - The scalable register bypassing (SRB) architecture
 - Case study applications on ByoRISC
- 4 Post-Ph.D. work
 - A high-level synthesis tool under development
 - Teaching responsibilities at UOP, Greece

The high-level synthesis tool under development

- Under development since June 2009; reached first milestone: Sept. 2010
- Actually consists of two applications
 - nac2cdfg: translator from an IR notation named NAC (N-Address Code), to flat CDFGs represented in Graphviz
 - cdfg2hdl: the actual HLS tool for automatic FSMD hardware and self-checking testbench generation from Graphviz input to VHDL
- Some features
 - Multi-precision integer (fgmp) and fixed-point arithmetic
 - Supports multiple procedures and single-dimensional arrays (mappable to BRAMs)
 - Scalar and streamed (emitting a series of values over time) outputs
 - Generators: VHDL design code (FSMD), Verilog, self-checking VHDL testbench, various scripts
 - Builtin hardware operator library

Teaching (and other) responsibilities (2008-2011)

- During the last two years, I have been working as an adjunct lecturer at the Department of Computer Science and Technology of the University of Peloponnese, Greece
- Almost all content (lectures, student project assignments, exams) is in Greek
- The courses that I teach during Spring 2011 are:
 - Hardware Description Languages I: Verilog HDL (about 110 registered students)
- Past courses (Fall 2008 to Fall 2010):
 - Digital Circuit Design
 - Compilers II
 - Hardware Description Languages (VHDL)
 - Advanced Topics in Compilers
 - Advanced Topics in Theoretical Computer Science (postgraduate: “Compilers II” with 20% additional material)
- Involved in the ENOSYS FP7 EU project since 01 Jan, 2010






Summary of Ph.D. work

- During my Ph.D. research, I had introduced a number of novelties to the typical ASIP design flow
- The corresponding architectural and exploration techniques were documented and automated in the form of hardware models and software tools, respectively
 - Application analysis: Machine-SUIF based environment
 - Custom instruction generation/selection: YARDstick
 - Hardware optimizations: ZOLC, scalable register bypassing
 - A unification of the above: ByoRISC processor family and tools
 - Custom instructions with up to 8 inputs and 8 outputs
 - Synthesizable RTL VHDL / ArchC models for simulation purposes
- Other work (not shown in this presentation)
 - Early versions of the application analysis flow
 - The VGP genetic algorithm processor [MAM07] (my participation: ArchC model, Nios-II CIs)






Present and future research/development plans

- More automation for the application-specific processor design flow
- Full compilation flow (GCC, LLVM or COINS-based) for a configurable soft processor
- Heterogeneous multicore architectures integrating ByoRISC (programmable) and FSMD acceleration nodes
- High-level synthesis of ASICs and ASIPs
 - A working tool targeting ASICs is being developed
 - HLS tool plan: Register allocation, better scheduler(s), nonleaf/recursive procedures, optimized memory synthesis, graph-based optimizations, fixed/floating-point arithmetic
 - Development of a source code transformation explorer
 - Graph-based IR optimizations
 - Resource sharing optimizations utilizing graph similarity metrics
 - Extensions to the FSMD model of computation







References I

-  N. Kavvadias and S. Nikolaidis, “Parametric architecture for implementing multimedia algorithms,” in *Proceedings of the 14th International Conference on Digital Signal Processing*, vol. II, July 1-3 2002, pp. 1261–1264.
-  —, “Application analysis with integrated identification of complex instructions for configurable processors,” in *Proceedings of the 14th International Workshop on Power and Timing Modeling, Optimization and Simulation*, September 15-17 2004, pp. 633–642.
-  —, “Automated instruction-set extension of embedded processors with application to MPEG-4 video encoding,” in *Proceedings of the IEEE 16th International Conference on Application-specific Systems, Architectures and Processors*, July 23-25 2005, pp. 140–145.
-  —, “Hardware support for arbitrarily complex loop structures in embedded applications,” in *Proceedings of the Design, Automation and Test in Europe Conference*, March 7-11 2005, pp. 1060–1061.
-  —, “Zero-overhead loop controller that implements multimedia algorithms,” *IEE Proceedings on Computers and Digital Techniques*, vol. 152, no. 4, pp. 517–526, July 2005.

References II

-  —, “A flexible instruction generation framework for extending embedded processors,” in *Proceedings of the 13th IEEE Mediterranean Electrotechnical Conference (MELECON 2006)*, May 16-19 2006, pp. 125–128.
-  —, “A portable specification of zero-overhead looping control hardware applied to embedded processors,” in *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems*, May 21-24 2006, pp. 1599–1602.
-  —, “Yardstick: Automation tool for custom processor development,” in *Presented at the University Booth of the Design, Automation and Test in Europe Conference (DATE'07)*, April 16-20 2007.
-  N. Kavvadias, V. Giannakopoulou, and S. Nikolaidis, “Development of a customized processor architecture for accelerating genetic algorithms,” *Microprocessors and Microsystems*, vol. 31, no. 5, pp. 347–359, 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.micpro.2006.12.002>
-  N. Kavvadias and S. Nikolaidis, “Elimination of overhead operations in complex loop structures for embedded microprocessors,” *IEEE Transactions on Computers*, vol. 57, no. 2, pp. 200–214, February 2008.

References III

-  —, “The ByoRISC configurable processor family,” in *Proceedings of the IFIP/IEEE VLSI-SoC 2008 – International Conference on Very Large Scale Integration*, October 13–15 2008, pp. 439–444.
-  —, “Scalable register bypassing for FPGA-based processors,” *Microprocessors and Microsystems*, vol. 33, no. 7–8, pp. 441–452, October–November 2009. [Online]. Available: <http://dx.doi.org/10.1016/j.micpro.2009.07.002>
-  N. Kavvadias and K. Masselos, “Efficient hardware looping units for FPGAs,” in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI 2010)*, July 5–7 2010.
-  N. Kavvadias, Personal webpage. [Online]. Available: <http://www.nkavvadias.co.cc>
-  N. Kavvadias. Hardware looping unit. [Online]. Available: <http://www.opencores.org/project,hwlu>
-  N. Kavvadias. Rational arithmetic package. [Online]. Available: <http://www.opencores.org/project,ratpack>