# Building a Hardware Prototype for Thread Level Speculation Using Programmable Logic

**Radu Teodorescu and Josep Torrellas**

[http://iacoma.cs.uiuc.edu/](http://iacoma.cs.uiuc.edu/)

---

## Goals
- Design and implement a hardware prototype of a TLS processor, using FPGA (Field Programmable Gate Arrays) technology
- Use the system to support a comprehensive debugging infrastructure
- Lightweight, on-the-fly debugging of production runs
- TLS supports rollback and replay of buggy sections of the program
- Experiment with real workloads, including OS kernel

## Debugging System
- Applications run in four possible execution states:
  - **Normal** – minimal checking
  - **Suspicious** – intensive checking
  - **Speculative** – TLS is enabled, can undo execution
  - **Erroneous** – re-execution, bug characterization, repair

## Hardware Support
- Speculative state is kept in the cache
- Speculative lines are not sent to memory
- If cache full or IC, begin new speculative window
- Speculative mode entry
  - Checkpoint register file and program counter
  - Mark new dirty lines as speculative and do not displace
- Speculative mode exit
  - Merge speculative and non-speculative state
  - Rollback
  - Invalidate speculative cache lines
  - Restore register file and program counter
  - Re-execute from checkpoint

## FPGA Technology
- Field Programmable Gate Arrays
  - Fast, complex, reprogrammable devices
  - Ideal platform for rapid prototyping
  - Large number of gates
  - Implements complex logic functions
  - Can be reprogrammed quickly
- Xilinx Virtex-II XC2V3000
  - 5 million gates
  - 3,584 logic blocks
  - Large array of programmable elements
    - CLB – configurable logic blocks
    - IOB – input/output blocks
    - SelectRAM – configurable memory cells
    - Programmable interconnect logic
  - Configurable logic blocks
    - Function generators
      - 16-bit RAM
      - 16-bit shift register
      - 4 input lookup table
    - Storage elements
    - Multiplexers

## Processor System
- SPARC V8 processor
- Simple 32 bit instructions
- 32 register window
- Coprocessor support
- Single issue, 5-stage pipeline
- Up to 4-way set associative, configurable caches
- Memory controller
- PCI, Ethernet interfaces
- Open source VHDL code, from Gaisler Research

## Design Process
- **Design Entry**
- **Hardware Description Language**
- **HEL Simulation**
- **Software**
- **Synthesis Tools**
- **Timing and Constraints, Analysis**
- **Deployment**
- **Development Board**
- **FPGA Debugging**

## Simulation
- **Application Code**
- **Breakpoint Manager**
- **ROM Monitor**
- **Single Core Model**
- **Memory Model**
- **VHDL Simulator**
- **Processor RTL Implementation**
- **Simulation Infrastructure**

## Deployment
- **Processor Image**
- **FPGA Programming Tool**
- **Communication Tool**
- **PCI Interface**

## Results
- Implementation of a cache controller that supports speculative versions and commit/rollback sequences
- Support for special instructions that enable/disable the speculative execution mode
- Implementation of a rollback sequence in hardware that restores the cache to its pre-speculative state
- Extensive testing of the implementation on RTL simulations
- Complete synthesis with timing analysis
- FPGA deployment and testing with C programs

## Future Directions
- Phase 1: Uniprocessor with TLS support
- Phase 2: SMT implementation
  - 2 cores – share L1 cache
- Phase 3: Chip multiprocessor
  - 2-4 cores, cache coherence
  - Second level of cache