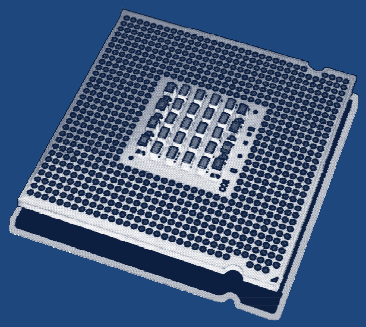
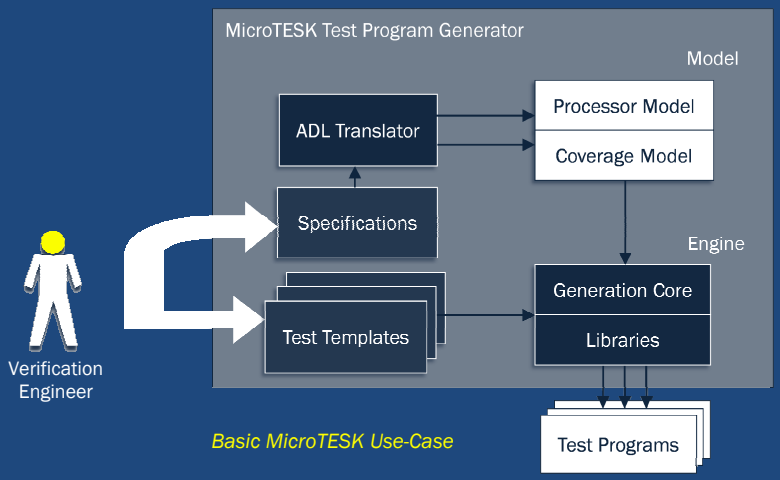
MicroTESK | Advanced Verification Program Generator for Microprocessors

MicroTESK is a reconfigurable (retargetable and extendable) model-based test program generator for microprocessors and other programmable devices. It is customized for a particular architecture by using light-weight instruction set specifications, which makes it easy to support various RISC and CISC architectures and facilitates automated extraction of knowledge about situations to be covered by tests. A convenient test template framework allows rapid development of complex verification scenarios.



Open Source | http://forge.ispras.ru/projects/microtesk



Basic MicroTESK Use-Case

Target Architectures

- MicroTESK is **retargetable**
  - RISC
  - CISC
  - DSP

  } wide range of ISA
- Primary focus on **RISC** architectures
  - ARM
  - MIPS
  - SPARC

  } + custom designs

Microprocessor Specification

- Specifications in **nML/Sim-nML** (TU Berlin/IIT Kanpur)
  - Memory structure and addressing modes
  - Behavioral description of instructions
  - Assembler/binary instruction formats
- Configurations in **domain-specific languages**
  - Memory management (TLB, L1 and L2)
  - Pipeline logic (microarchitectural networks)
  - Branch processing (prediction, etc.)

```
op ADD(c:C, s:S, r1:REG, r2:REG, d3:DATA)
  syntax = format("ADD%s%s %s, %s, %s", ... )
  image  = format("%s0000100%s%s%s%s",  ... )
  action = {
    c.action; d3.action;
    if (X == 1) then
      carry::ALU_OUT = GPR[r2] + d3;
      if (s == 1) && (r1 == 15) then
        CPSR = SPSR;
      else
        if (s == 1) then
          CPSR<31..31> = ALU_OUT<31..31>;
          if (ALU_OUT == 0) then
            CPSR<30..30> = 1;
          else
            CPSR<30..30> = 0;
          endif;
          CPSR<29..29> = carry;
          if (GPR[r2]<31..31> == d3<31..31>) &&
             (ALU_OUT != GPR[r2]<31..31>) then
            CPSR<28..28> = 1;
          else
            CPSR<28..28> = 0;
          endif;
        endif;
      endif;
      GPR[r1] = ALU_OUT;
    endif;
  }
```

Assembler format
Instruction behavior (processor model)

Testing knowledge (coverage model)

Instruction Description in Sim-nML Language

```
...
add r(1), r(2), r(3)
sub r(4), r(1), r(5) do overflow end
newline

text "// Text be placed in a test"
newline

sub r(i=rand(8..16)), r(10..20), r(i)
newline

3.times do
  ld  r(1), r(2)       do hit    end
  add r(3), r(1), r(4) do normal end
  newline
end

(1..5).each do |i|
  ori r(i), r(i+1), r(0)
  newline
end
...
```

Processor model access
Coverage model access

Test code formatting

Test randomization

Test sequence control

Test Program Template in Ruby Language

Test Template Description

- Ruby-based language
- Focus on simplicity and productivity
- Integration with test generators
- Access to processor and coverage models

Test Program Generation

- Random
- Combinatorial
- Constraint-based
- Model-based

The MicroTESK test program generator is being developed at the Software Engineering Department of the Institute for System Programming, Russian Academy of Sciences (ISPRAS). The institute performs both academic research and industrial development projects as well as provides advanced services and consulting in various areas of software engineering, information technologies and computer science.

We gratefully acknowledge Indian Institute of Technology Kanpur (IIT Kanpur) provided us with Sim-nML documentation and examples.

ISP RAS