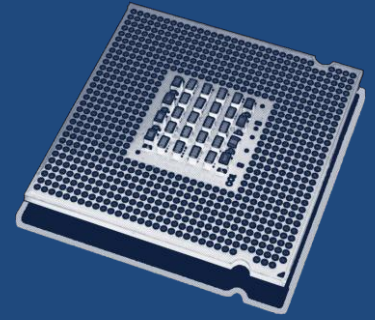
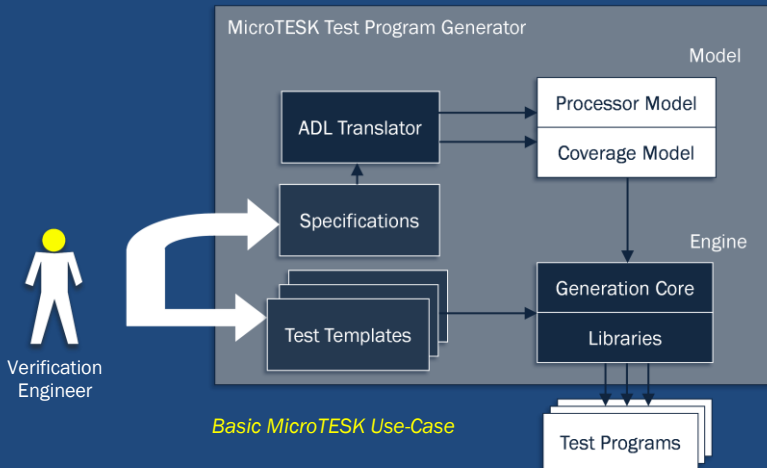


MicroTESK is a reconfigurable (retargetable and extendable) model-based test program generator for microprocessors and other programmable devices. Lightweight formal specifications customize the generator for a particular architecture and provide knowledge about situations to be covered by tests. A convenient test template framework allows rapid development of complex verification scenarios. Being retargetable, MicroTESK is able to support various RISC and CISC architectures.



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



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", ...)
image = format("%000100%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 = SPGR;
    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;
      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 Program Generation

- Test program templates in **Ruby**
 - focus on simplicity and productivity
 - transparent access to processor model
 - integration with test generators
- MicroTESK is **extendable**
 - randomized generation
 - combinatorial generation
 - model-based generation

} + custom methods