A.P. Ershov Informatics Conference
(the PSI Conference Series, 11th edition)

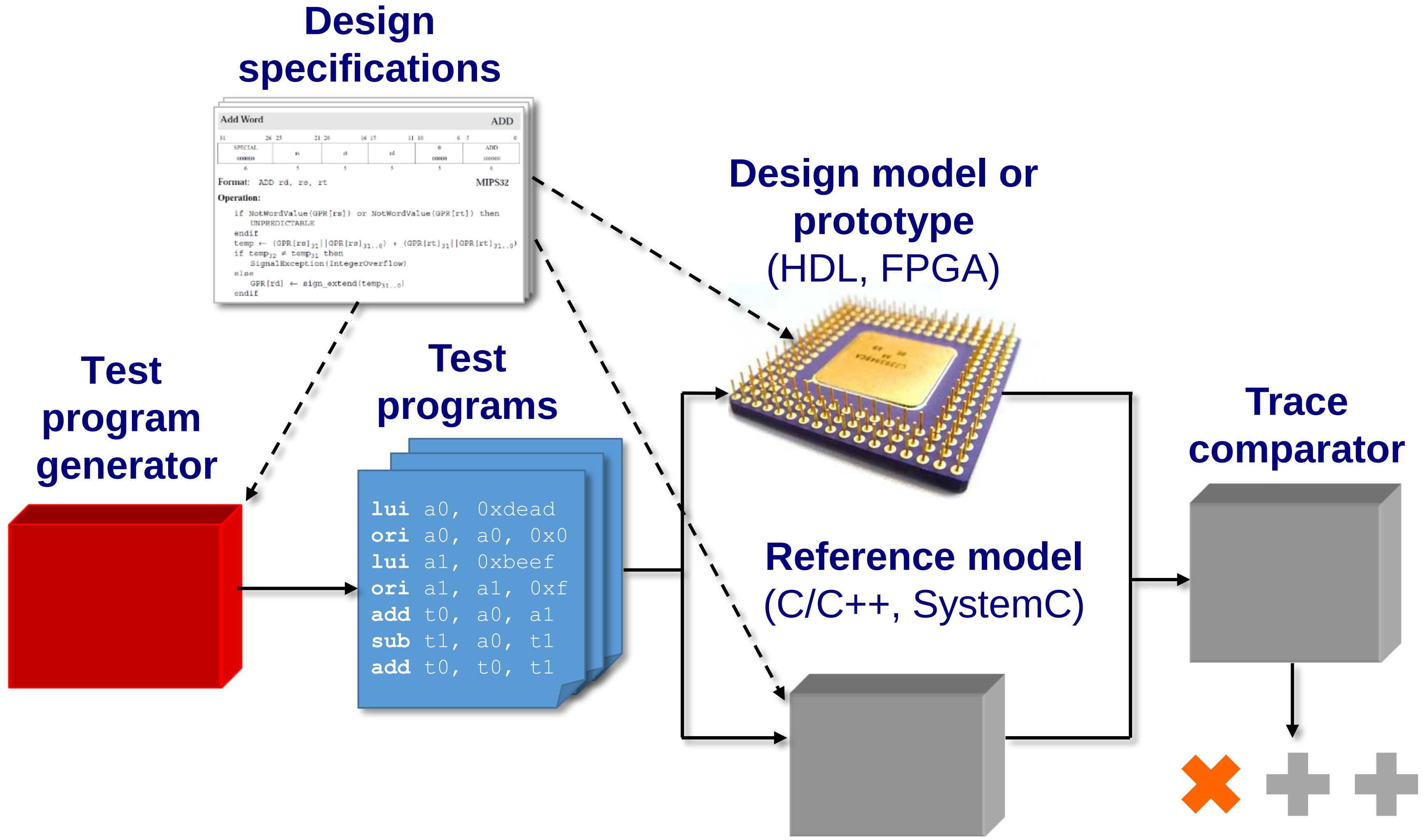
MicroTESK*: A Tool for Constrained Random Test Program Generation for Microprocessors*

Alexander Kamkin and Andrei Tatarnikov

{kamkin, andrewt}@ispras.ru

Institute for System Programming of the
Russian Academy of Sciences (ISP RAS)

Functional Verification of Microprocessors



Design specifications

Add Word ADD

31–26	25–21	20–16	15–11	10–6	5–0
SPECIAL 000000	rs	rt	rd	0 00000	ADD 100000
6	5	5	5	5	6

Format: `ADD rd, rs, rt` MIPS32

Operation:

```
if NotWordValue(GPR[rs]) or NotWordValue(GPR[rt]) then
   UNPREDICTABLE
endif
temp ← (GPR[rs]$_{31}$||GPR[rs]$_{31..0}$) + (GPR[rt]$_{31}$||GPR[rt]$_{31..0}$)
if temp$_{32}$ ≠ temp$_{31}$ then
   SignalException(IntegerOverflow)
else
   GPR[rd] ← sign_extend(temp$_{31..0}$)
endif
```

Design model or prototype
(HDL, FPGA)

Test program generator

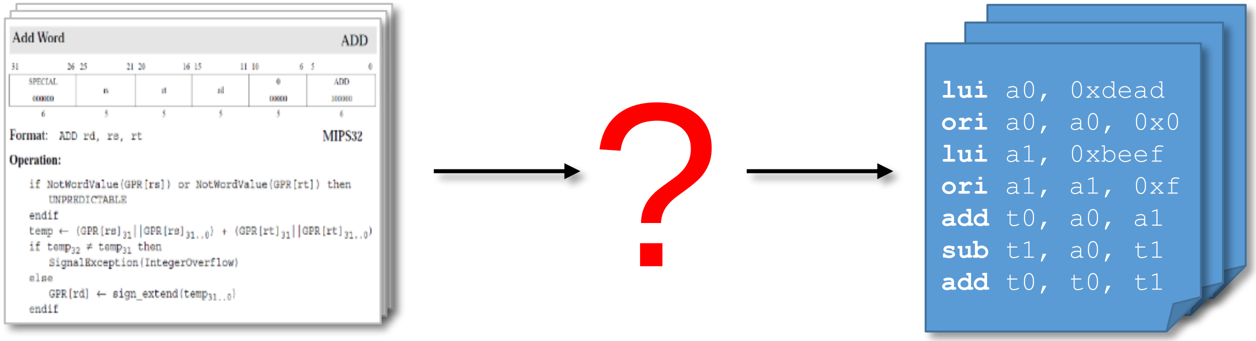
Test programs

```
lui a0, 0xdead
ori a0, a0, 0x0
lui a1, 0xbeef
ori a1, a1, 0xf
add t0, a0, a1
sub t1, a0, t1
add t0, t0, t1
```

Reference model
(C/C++, SystemC)

Trace comparator

Test Program Generation Techniques



Add Word ADD

31–26	25–21	20–16	15–11	10–6	5–0
SPECIAL 000000	rs	rt	rd	0 00000	ADD 100000
6	5	5	5	5	6

Format: ADD rd, rs, rt MIPS32

Operation:

```
if NotWordValue(GPR[rs]) or NotWordValue(GPR[rt]) then
    UNPREDICTABLE
endif
temp ← (GPR[rs]$_{31}$||GPR[rs]$_{31..0}$) + (GPR[rt]$_{31}$||GPR[rt]$_{31..0}$)
if temp$_{32}$ ≠ temp$_{31}$ then
    SignalException(IntegerOverflow)
else
    GPR[rd] ← sign_extend(temp$_{31..0}$)
endif
```

```
lui a0, 0xdead
ori a0, a0, 0x0
lui a1, 0xbeef
ori a1, a1, 0xf
add t0, a0, a1
sub t1, a0, t1
add t0, t0, t1
```

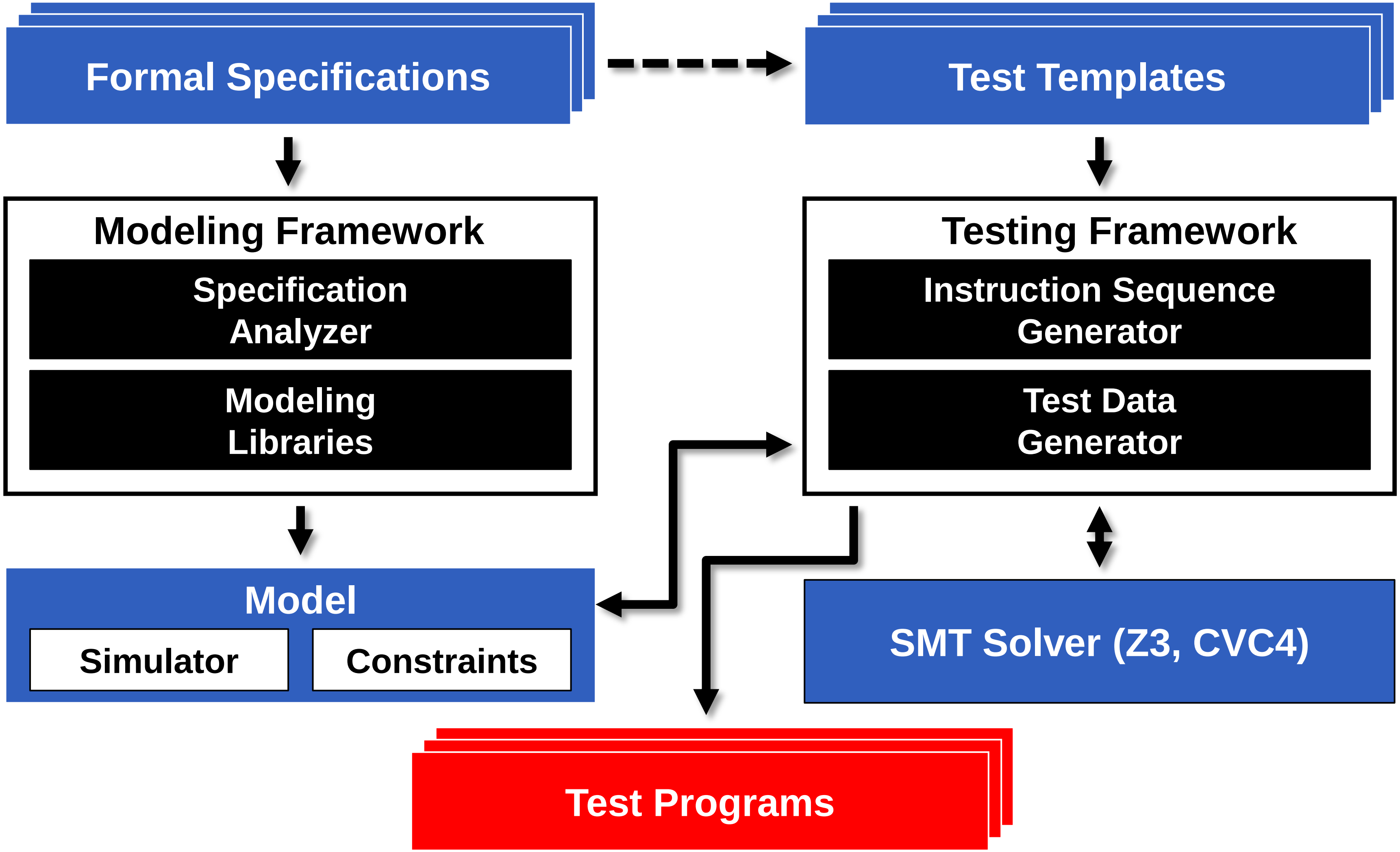
- Random
- Combinatorial
- Constraint-based
- Constrained random

MicroTESK Test Program Generator

MicroTESK Verification Technology for Microprocessors

- Describing ISA of microprocessor under test using formal specifications
- Automated extraction of constraints from formal specifications
- Description of test templates in a flexible domain-specific Ruby-based language

Architecture of MicroTESK



Formal Specifications

Test Templates

Modeling Framework

Specification Analyzer

Modeling Libraries

Testing Framework

Instruction Sequence Generator

Test Data Generator

Model

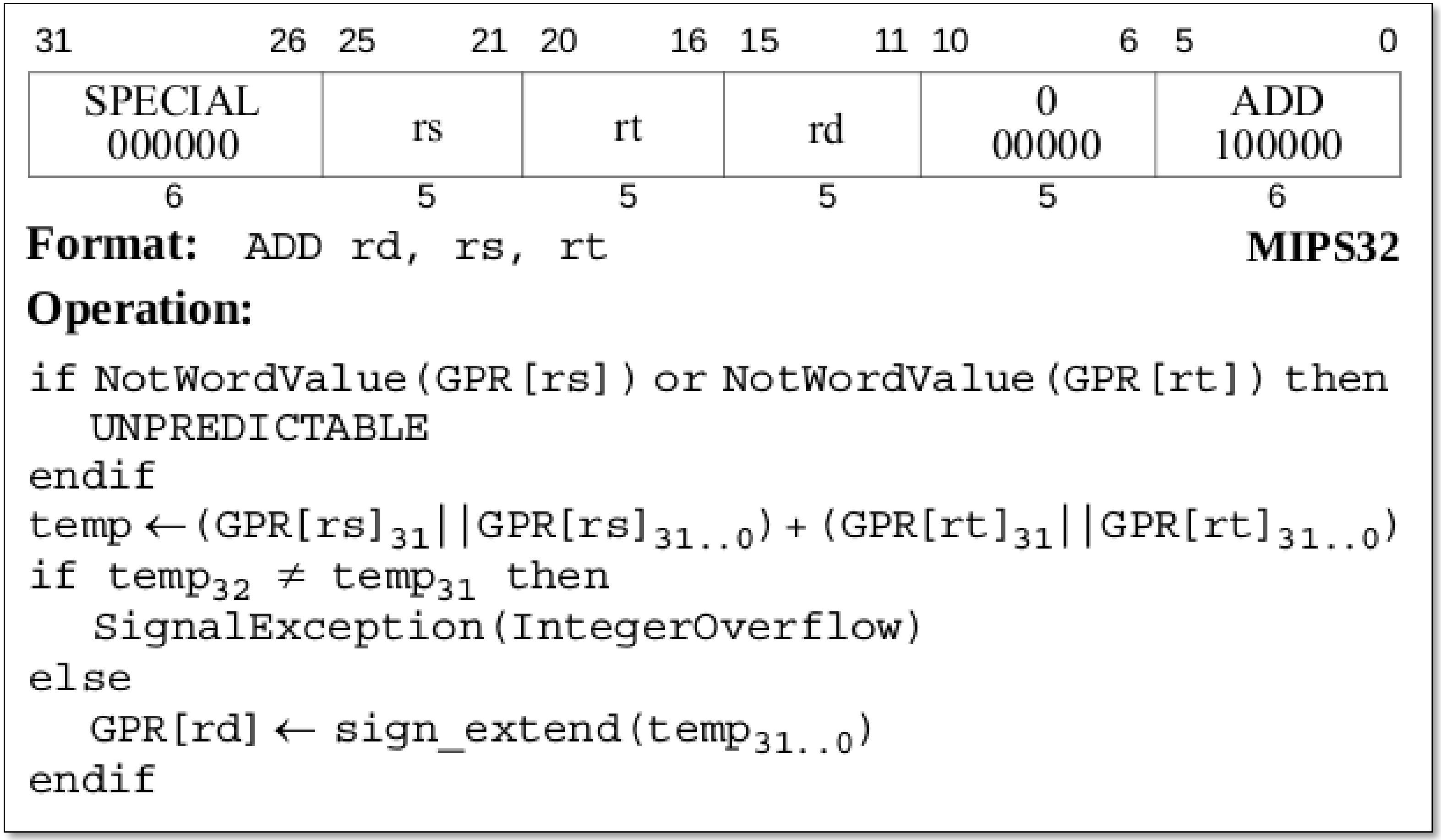
Simulator

Constraints

SMT Solver (Z3, CVC4)

Test Programs

Documentation on Instruction Set Architecture



31 26	25 21	20 16	15 11	10 6	5 0
SPECIAL 000000	rs	rt	rd	0 00000	ADD 100000
6	5	5	5	5	6

Format: `ADD rd, rs, rt` **MIPS32**

Operation:

```
if NotWordValue(GPR[rs]) or NotWordValue(GPR[rt]) then
   UNPREDICTABLE
endif
temp ← $(GPR[rs]_{31} || GPR[rs]_{31..0}) + (GPR[rt]_{31} || GPR[rt]_{31..0})$
if $temp_{32} \neq temp_{31}$ then
   SignalException(IntegerOverflow)
else
   GPR[rd] ← $sign\_extend(temp_{31..0})$
endif
```

Formal Specifications in nML

```
reg GPR [32, card(64)]
mode R (i: card(5)) = GPR[i]
   syntax = format("r%d", i)
   image  = format("%5s", i)
var temp33[card(33)]

op ADD (rd: R, rs: R, rt: R)
syntax = format("add %s, %s, %s",  rd.syntax, rs.syntax, rt.syntax)
image = format(“0000%5s%5s%5s0000100000", rs.image, rt.image, rd.image)
action = {
   if sign_extend(WORD, rs<31>) != rs<63..32> ||
     sign_extend(WORD, rt<31>) != rt<63..32> then
      unpredicted;
   endif;
   temp33 = rs<31>::rs<31..0> + rs<31>::rt<31..0>;
   if (temp33<32> != temp33<31>) then
      exception("overflow");
   else
      rd = sign_extend(DWORD, temp33<31..0>);
   endif;
}
```

Register

Addressing mode

Instruction

Precondition

Test situations

Constraints in SMT-LIB

Constraint for test situation “overflow”

```
(declare-const rs (_ BitVec 64))
(declare-const rt  (_ BitVec 64))
(declare-const temp33 (_ BitVec 33))

(define-fun IsWordValue ((value (_ BitVec 64))) Bool
   (ite (= ((_ sign_extend 31)
         ((_ extract  31 31) value)) ((_ extract 63 32) value)) true false))

(assert (IsWordValue rs))
(assert (IsWordValue rt))

(assert (= temp33 (bvadd (concat ((_ extract 31 31) rs) ((_ extract 31 0) rs))
                         (concat ((_ extract 31 31) rt) ((_ extract 31 0) rt)))))
(assert (not (= ((_ extract 32 32) temp33) ((_ extract 31 31) temp33))))

(check-sat)
(get-value (rs rt))
```

Test Templates in Ruby

Task: generate all pairs of instructions add and sub causing all possible combinations of test situations normal and overflow

```
preparator(:target => ‘R’) {
  lui target, value(16, 31)
  ori target, target, value(0, 15)
}
block(:combinator => ‘product’) {
  iterate {
    add t0, t1, t2 do situation(‘normal’) end
    add t0, t1, t2 do situation(‘overflow’) end
  }
  iterate {
    sub t3, t4, t5 do situation(‘normal’) end
    sub t3, t4, t5 do situation(‘overflow’) end
  }
}.run
```

Result: 4 sequences

```
lui r9, 0xa9c9
ori r9, r9, 0x7025
lui r10, 0xe6f6
ori r10, r10, 0x78de
lui r12, 0x5db7
ori r12, r12, 0xfa2c
lui r13, 0xfafc
ori r13, r13, 0x3700
```

```
add r8, r9, r10
sub r11, r12, r13
```

Test Program Generation Process

- Constructing abstract sequences
  - Selecting instructions and fixing their order
  - Selecting registers
  - Selecting constraints
- Constructing concrete sequences
  - Generating test data (solving constraints)
  - Constructing initialization code
- Simulating instruction sequences
- Constructing self-checks
- Printing instruction sequences to source code files

Conclusion

- Using formal specifications for configuring generator for a specific instruction set architecture
- Test program generation on the basis of Ruby templates
- Automated constraint extraction
- Industrial application of generators for ARMv8 and MIPS64

Microprocessor	ARMv8	MIPS64	PowerPC	RISC-V
Instructions specified	795	220	34	63
Size of specifications (LOC)	12220	3999	935	816
Effort (man-months)	13	4	1	0.75

Thank you!

Questions?

More information:

http://microtesk.org

http://forge.ispras.ru/projects/microtesk