# drm

# How Can You Trust Formally Verified Software?

Alastair Reid

Arm Research @alastair\_d\_reid



# **Formal verification**

### **Of libraries and apps**





### **Of compilers**

COMPCERT





### **Of operating systems**





arm

Fonseca et al., An Empirical Study on the Correctness of Formally Verified Distributed Systems, Eurosys '17



Fonseca et al., An Empirical Study on the Correctness of Formally Verified Distributed Systems, Eurosys '17



# Takeaway #1: 3 key questions to ask

- 1. What specifications does your proof rely on?
- 2. Why do you trust those specifications?
- 3. Does anybody else use these specifications?

## **Takeaway #2: Specifications must have multiple uses**

### **Takeaway #2: Specifications must have multiple uses**



# How can you trust formally verified software?

### How can you trust formally verified software?

- Specifications are part of the TCB
- 3 key questions
- Specifications must have multiple users
- How can you trust formal specifications?
  - Testing specifications
  - Verifying processors
  - Verifying specifications

How can you trust formally verified software?

"Trustworthy Specifications of the ARM v8-A and v8-M architecture," FMCAD 2016

# Creating trustworthy specifications

arm

© 2017 Arm Limited

# **Arm Architecture Reference Manual (ARMARM)**



32-bit / 64-bit Instructions Exceptions / Interrupts Privilege / Security Virtual Memory System registers Debug / Trace Profiling

...

2018

# **English prose**

R<sub>JRJC</sub>

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.

R<sub>VGNW</sub>

Entry to lockup from an exception causes:

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending or active.
- The PC to be set to 0xEFFFFFE.
- EPSR.IT to be become UNKNOWN.

### **Pseudocode**

#### Encoding A1 ARMv4\*, ARMv5T\*, ARMv6\*, ARMv7

ADC{S}<c> <Rd>, <Rn>, <Rm>{, <shift>}

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

cond	0 0	0	0 1	0	1	S	Rn	Rd	imm5	type	0	Rm
------	-----	---	-----	---	---	---	----	----	------	------	---	----

```
if Rd -- '1111' && S -- '1' then SEE SUBS PC, LR and related instructions;
d - UInt(Rd); n - UInt(Rn); m - UInt(Rm); setflags - (S -- '1');
(shift_t, shift_n) - DecodeImmShift(type, imm5);
if ConditionPassed() then
    EncodingSpecificOperations();
    shifted - Shift(R[m], shift_t, shift_n, APSR.C);
    (result, carry, overflow) - AddWithCarry(R[n], shifted, APSR.C);
    if d -- 15 then // Can only occur for ARM encoding
        ALUWritePC(result); // setflags is always FALSE here
    else
        R[d] - result;
        if setflags then
            APSR.N - result<31>;
```

```
APSR.Z = IsZeroBit(result);
```

```
APSR.C - carry;
```

```
APSR.V - overflow;
```

# **Arm Architecture Specification Language (ASL)**

Indentation-based syntax

Imperative

First-order

Strongly typed (type inference, polymorphism, dependent types)

**Bit-vectors** 

Unbounded integers

Infinite precision reals

Arrays, Records, Enumerations

#### Exceptions



# **Architectural Conformance Suite**

### Processor architectural compliance sign-off

### Large

- v8-A 11,000 test programs, > 2 billion instructions
- v8-M 3,500 test programs, > 250 million instructions

### Thorough

• Tests dark corners of specification



arm

# **v8-M**



# Measuring architecture coverage of tests

Untested: op1\*op2 == -3.0, FPCR.RND=-Inf

	b	its(N) FPRSqrtStepFused(bits(N) op1, bits(N) op2)
TESTED		assert N IN {32, 64};
TESTED		bits(N) result;
TESTED		op1 = FPNeg(op1); // per FMSUB/FMLS
TESTED		<pre>(type1,sign1,value1) = FPUnpack(op1, FPCR);</pre>
TESTED		<pre>(type2,sign2,value2) = FPUnpack(op2, FPCR);</pre>
TESTED		(done,result) = FPProcessNaNs(type1, type2, op1, op2, FPCR);
TESTED	TESTED TESTED	if !done then
TESTED		<pre>inf1 = (type1 == FPType_Infinity);</pre>
TESTED		inf2 = (type2 == FPType_Infinity);
TESTED		zero1 = (type1 == FPType_Zero);
TESTED		zero2 = (type2 == FPType_Zero);
TESTED	TESTED TESTED	if (inf1 && zero2)    (zero1 && inf2) then
TESTED		result = FPOnePointFive('0');
	_	elsif inf1    inf2 then
TESTED		result = FPInfinity(sign1 EOR sign2, N);
		else
		// Fully fused multiply-add and halve
TESTED		result_value = $(3.0 + (value1 * value2)) / 2.0;$
TESTED	UNEXECUTED TESTED	if result_value == 0.0 then
	_	// Sign of exact zero result depends on rounding mode
UNEXECUTED		<pre>sign = if FPCRRounding() == FPRounding_NEGINF then '1' else '0';</pre>
UNEXECUTED		result = FPZero(sign, N);
		else
TESTED		result = FPRound(result_value, FPCRRounding());
TESTED		return result;



"End to End Verification of ARM processors with ISA Formal," CAV 2016

# **Formal verification**

# of processors

arm

© 2017 Arm Limited

# Checking an instruction





The Architecture for the Digital World®



#### **ARM**Research

# Checking an instruction













arm



#### **ARM**Research

The Architecture for the Digital World®

# **Arm CPUs verified with ISA-Formal**

A-class	R-class	M-class
Cortex-A53	Cortex-R52	Cortex-M4
Cortex-A32	Next generation	Cortex-M7
Cortex-A35		Cortex-M33
Cortex-A55		Next generation
Next generation		Cambridge Projects

#### **Rolling out globally to other design centres**

Sophia, France - Cortex-A75 (partial)

Austin, USA - TBA

Chandler, USA - TBA



"Who guards the guards? Formal Validation of ARM v8-M Specifications" OOPSLA 2017

# Formal validation of specifications



© 2017 Arm Limited



Last year: audited all accesses to privileged registers

- Specification: Added missing privilege checks
- Testsuite: Added new tests to test every privilege check
- Formal testbench: Verify every check

This year: add new instruction but accidentally omit privilege check

How many tests in the test suite will fail on new specification?

# Can we formally verify specification?

# Specification of the specification

- **Disallowed behaviour**
- Invariants
- **Cross-cutting properties**

# Tools that can prove properties of ASL specifications

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.



• Preemption by a higher priority exception.



#### R<sub>JRJC</sub>

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.



29

Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.







Exit from lockup is by any of the following:

- A Cold reset.
- A Warm reset.
- Entry to Debug state.
- Preemption by a higher priority exception.

rule lockup\_exit
 assume Fell(LockedUp);
 Called(TakeColdReset)
 v Called(TakeReset)
 v Rose(InDebugState())
 v Called(ExceptionEntry);

# **Converting ASL to SMT**

**Functions** Local Variables Statements Assignments If-statements Exceptions Arithmetic operations **Boolean operations Bit Vectors** Arrays

**Functions Local Variables Statements Assignments** If-statements **Exceptions** Arithmetic operations **Boolean operations Bit Vectors** Arrays

### **Formally Validating Specifications**



arm

## **Formally Validating Specifications**



arm

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending or active.
- The PC to be set to 0xEFFFFFE.
- **EPSR**.IT to be become UNKNOWN.

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending or active.
- The PC to be set to 0xEFFFFFE.
- **EPSR.IT** to be become UNKNOWN.

```
rule lockup entry
   assume Rose(LockedUp);
   assume ¬Called(TakeReset);
   property a HaveMainExt() ⇒ CFSR != 0;
   property b1 Stable(ExnPending);
   property b2 Stable(ExnActive);
   property c PC == 0xEFFFFFE;
   property e HFSR.FORCED == 0;
```

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending or active.
- The PC to be set to 0xEFFFFFE.
- **EPSR.IT** to be become UNKNOWN.

```
rule lockup entry
    assume Rose(LockedUp);
    assume ¬Called(TakeReset);
                   HaveMainExt() \Rightarrow CFSR != 0;
    property a
    nronerty h1 Stahle(EvnDending).
                   Stable(EvnActive).
    nronerty h?
                   PC)== 0xEFFFFFE;
    property c
                   HECD FODCED --
    property e
                Stable(HFSR.FORCED);
```

- Any Fault Status Registers associated with the exception to be updated.
- No update to the exception state, pending of active.
- The PC to be set to 0xEFFFFFE.
- EPSR.IT to be become UNKNOWN.

In addition, HFSR.FORCED is not set to 1. is not changed.

rule lockup entry
 assume Rose(LockedUp);
 assume ¬Called(TakeReset

Debug view of









# Public release of machine readable Arm specification

Enable formal verification of software and tools

Releases

April 2017: v8.2

July 2017: v8.3

Working with Cambridge University REMS group to convert to SAIL

Backends for HOL, OCaml, Memory model, (hopefully Coq too)

Tools: <u>https://github.com/alastairreid/mra\_tools</u>

Talk to me about how I can help you use it

# **Potential uses of processor specifications**

Verifying compilers

Verifying OS page table / interrupt / boot code

Verifying processor pipelines

Verification and discovery of peephole optimizations

Automatic generation of binary translators

Automatic generation of test cases

Decompilation of binaries

Abstract interpretation of binaries

etc.

# How can you trust formally verified software?



© 2017 Arm Limited

# How can you trust formal specifications?

Test the specifications you depend on

Ensure specifications have multiple uses

Create meta-specifications





https://xkcd.com/1416/

arm

Thank You! Danke! Merci! 谢谢! ありがとう! Gracias! Kiitos!

# @alastair\_d\_reid

arm

"Trustworthy Specifications of the ARM v8-A and v8-M architecture," FMCAD 2016 "End to End Verification of ARM processors with ISA Formal," CAV 2016 "Who guards the guards? Formal Validation of ARM v8-M Specifications" OOPSLA 2017