Overview of what's coming

I Review of our previous work on ARM verificaton

I Plans for new project

I Current research at Cambridge, Oxford, Utah
Overview of what’s coming

- Review of our previous work on ARM verification
- Plans for new project
- Current research at Cambridge, Oxford, Utah
Formally verified **almost all** of an ARM610
- substantial effort by Anthony Fox using HOL4
- proof used Tucker/Harman ‘algebraic’ approach

Proved instruction model abstracts ARM hardware
- collaboration with University of Leeds and ARM Ltd
- very accurate model of Instruction Set Architecture (ISA)
ARM6 is old, but real
- small: 35,000 transistors
- used in Apple Newton PDA

Most successful ARM implementation is ARM7
- hundreds of millions in cellular phones
- same instructions as ARM6
- simpler architecture (multiplication in ALU)

StrongARM and XScale
- StrongARM developed by DEC
- acquired by Intel and renamed XScale
- high performance ARM implementations (PDAs etc)
Isn’t ARM6 obsolete?

- **Key point:** ARM instruction semantics are stable
- We will use only validated instructions
- Have validated ISA against one real implementation
- ARM6 proof viewed as debugging ISA model
- ARM6 very similar to still widely deployed ARM7
- New versions of ISA backwards compatible
Current work on ARM

- Verify combinations of software and hardware

HCSS 2005 – p.6/15
New project

- Formally Validated ISA is our reference semantics
- Develop verification and synthesis tools
  - for ARM software
  - for hardware system components (co-processors, slaves)
- Apply to cryptography
  - first AES and then ECC
Formally Validated ISA is our reference semantics

Develop verification and synthesis tools
  - for ARM software
  - for hardware system components (co-processors, slaves)

Apply to cryptography
  - first AES and then ECC

Started at Utah
Formally Validated ISA is our reference semantics

Develop verification and synthesis tools
  - for ARM software
  - for hardware system components (co-processors, slaves)

Apply to cryptography
  - first AES and then ECC

Started at Utah

............... may start soon at Oxford & Cambridge
Software runs on hardware
- model of hardware is ultimate semantics
- old idea: Viper, CLI stack, Rockwell-Collins, Boyer-Yu

What level of abstraction?
- instruction set architecture (ISA)
- micro-architecture (pipelines explicit etc)
- circuit

For our project ISA is golden
- validated with respect to ARM610 micro-architecture
- micro-architecture adds details for IO and exceptions
First build symbolic simulation platform
  ▶ learn from work of others
  ▶ ARM memory not part of state – separate memory model

Next build programming logic abstraction
  ▶ too painful to reason directly about ARM executions
  ▶ verify proof rules against ARM instruction model

Implement verifier for derived programming logic
  ▶ symbolic execution, verification conditions etc
Verification platform is for post hoc code proof

Also explore correct-by-construction synthesis
- source: functional programs in TFL
- targets: ARM assembler and bespoke hardware
- goals: HW/SW partitioning (code + co-processor)

Work just starting:
- TFL to ARM assembler
- TFL to SAFL-like hardware (TPHOLs submission)
Verification and synthesis will be applied to crypto

Start with AES
- Slind has already specified in HOL
- properties proved \((\text{decrypt} \circ \text{crypt} = \text{Identity})\)
- hardware synthesis in progress (with Scott Owens)

Eventually also look at ECC
- Joe Hurd (Oxford)
- relevant finite field theories exist in HOL
Current work at Oxford

- Elliptic curves
  - infrastructure: finite fields, projective coordinates
  - define set of points on an elliptic curve
  - define the addition operation on elliptic curve points
  - prove addition satisfies the group laws

- ElGamal encryption
  - generic correctness theorem in higher order logic
  - instantiate to particular groups (e.g. elliptic curves)

- Model checking chess endgames
  - fun example using HOL4 and BDDs
  - more experience in combining deduction and checking
Current work at Cambridge

- Continuing to develop ARM ISA model
  - separate memory model: ARM6 = CPU || Memory
  - model of IO and exceptions partially ARM6 specific

- Code execution platform
  - execute instructions with IO & exceptions
  - ground and symbolic execution
  - some challenging issues (talk to Anthony)
  - goal: self-contained and usable by Oxford and Utah

- HOL to Verilog
  - goal: proof-producing compilation from TFL to FPGA
  - modified Define to hwDefine
  - used for Utah AES compilation
Current work at Utah

- AES specified and verified in HOL
  - proofs validate higher order logic specifications
  - tail-recursive version of specs derived by proof
  - proof-producing synthesis to circuits in progress

- Students theorem proving projects
  - redo Slind’s AES verification for other crypto algorithms
  - graduate class projects: IDEA, Serpent, RC6

- Proof-producing compilation to ARM assembler
  - goal: compile functional programs and produce proof
  - shares ideas with proof-producing hardware compilation
Conclusions

- Collaboration between Oxford, Cambridge, Utah
  - Utah part has officially started
  - Oxford and Cambridge are ‘jumping the gun’
Conclusions

- Collaboration between Oxford, Cambridge, Utah
  - Utah part has officially started
  - Oxford and Cambridge are ‘jumping the gun’

VISION  crypto mathematics ↔ hardware & software
Collaboration between Oxford, Cambridge, Utah

- Utah part has officially started
- Oxford and Cambridge are ‘jumping the gun’

VISION  *crypto mathematics* ↔ *hardware & software*

THE END