Instruction-level Abstraction (ILA) based SoC Firmware Verification

System-on-Chip Verification

SoC functionality is implemented by a combination of hardware and firmware

Verification Challenges
- Verifying the complete HW+FW design is not scalable
- Separate verification of HW and FW misses bugs

Instruction-Level Abstraction (ILA)

Key ideas
- Construct abstraction at instruction-granularity
- Better scalability
- Software verification techniques

Future Research

ILA-based SoC Verification
- Fully automated ILA synthesis
- Interfacing with software verification tool (SeaHorn)
- Interleaving semantics between processors and accelerators

Security property
- Automatic property decomposition
- Property specification language
- Interrupt-related security property

This work was supported in part by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA and a research gift from Intel Corporation.

Memory-mapped Accelerator Registers

void setToModeA(KEY keyIdx)
{
    if (keyIdx == KEY1)
    {
        MMIO_write(KEY_LOCK_BASE + keyIdx, 0xl);
    }

    MMIO_write(KEY_MODE_BASE + keyIdx, MODE_A);
}

MMIO Side Effect
- Read-only
- Write-only
- Lock-protected
- Operation triggering

Accelerator’s High-level State Machine
- Race condition

Instruction-Level Abstraction (ILA) based SoC Firmware Verification
- Fully automated ILA synthesis
- Interfacing with software verification tool (SeaHorn)
- Interleaving semantics between processors and accelerators

Processor ILA Extraction via QEMU
- One-time ILA construction for Tiny Code Generator (TCG) intermediate representation (IR)
- Generate processor ILA per instruction
- Good for heterogeneous environment in SoC

Template-based ILA synthesis
- Template: partially defined model
- Black box simulator
- CEGIS-based and parameterized algorithms
- Equivalence checking with RTL implementation

Template: Target TCG Host

QEMU Host binary

Firmware binary

TCG IR

movl (%esp,%ebx,4), %eax
add reg2, reg9, reg6
addi reg2, reg2, 4
load reg3, reg5
store reg2, reg3

Processor ILA Extraction via QEMU

Template:

op

ALU

SRC1 = choice [reg8, ..., reg7, imm]
SRC2 = choice [reg8, ..., reg7, imm]
ADD = SRC1 + SRC2
SUB = SRC1 - SRC2
MUL = SRC1 * SRC2
ALU_OUT = choice [ADD, SUB, MUL]