Motivation

- The joys of driver development
  - Poor hardware documentation
  - Poor OS documentation
  - Debugging drivers is hard

Conventional driver development

Driver synthesis with Termite
Termite is not ...

- a domain-specific language for drivers
  ```
  FEATURE device_read 
  OUTPUT char data
  poll lsr.dr until (LSR.dr == 1) data = RBR
  ERROR (LSR.oe == 0x1 || LSR.pe == 0x1);
  ```
- a register description language
  ```
  device logitech_busmouse(base : bit[8] port@{0..3})
  {
  register sig_reg = base @ 1 : bit[8];
  variable signature = sig_reg, ... : int[8];
  }
  ```
- a theoretical exercise
  - the goal is to synthesise efficient drivers for real-world devices (network, storage, audio, etc.)

Driver synthesis with Termite

- Where do specifications come from?
  - A device spec can be as complex as the driver
  - Use existing device specifications developed by hardware designers

Hardware Design Workflow

Informal specification

High-level model

Register-transfer-level description

• Low-level description: registers, gates, wires.
• Cycle-accurate
• Precisely models internal device architecture and interfaces

Driver synthesis as controller synthesis

• Captures external behaviour
• Abstracts away structure and timing
• Abstracts away the low-level interface

bus_write(u32 addr, u32 val)
```

...
Driver synthesis as controller synthesis

Driver = controller

device

Packet has been sent

send() - send a network packet

OS requests = control objective

Example: trivial network adapter

write(ctl,0)
write(ctl,1)
write(dat,...)
send

controllable transition

uncontrollable transition

GOAL

INIT

Computing the winning set

Cpre({done}) = {bsy}
Cpre({done,bsy}) = {bsy,on}

Computing the winning set

Game theory

• Game theory
  – Provides a theoretical framework for verification and synthesis of reactive systems
  – Provides a classification of games
  – Complexity bounds for various types of games
  – Algorithms for finding winning strategies
Computing the winning set

Cpre({done, bsy}) = {bsy}
write(ctl, 0)
write(ctl, 1)
write(dat, ...)
send

GOAL

Game automaton

OS specification

Game objective:
The driver must be in state 0 infinitely often (aka Büchi objective)

Synthesis with imperfect information

Synthesis with imperfect information
Strategy with imperfect information

Challenges

- State explosion (2^{32} states in the IDE controller device)
  - Classical game theory algorithms do not scale
  - Predicate abstraction (2^8 states after abstraction)
  - Symbolic state space representation (BDD size: 3226 nodes)

Demot

Key Research Outcomes (to date)

Successfully synthesised drivers:

- IDE disk controller
- W5100 Eth shield
- Aalix AX88772 USB-to-Eth adapter
- SD host controller

Termite toolkit
Synthesis of provably correct drivers

- Synthesised => Correct ??
  - Bugs in the tool
  - Bugs in the input spec
- Dealing with bugs in the synthesis tool
  - Generate proof of correctness along with the driver
- Dealing with buggy specs

Bridging the gap between hardware and driver synthesis

- Automatic device-driver synthesis:
  - correct-by-construction device drivers at a fraction of the cost of manual development
  - practical alternative to traditional driver development