Embedded Software
Dynamic Analysis

A new life for the Virtual Platform
The Software Part of HW/SW Co-Design

- Integrated with DA flow: regression and up-to-date interfaces
- Simultaneous development and test of embedded software in-house
- Early SW enablement through Virtual Platform
  - Applicable until silicon available
  - Preference for actual HW due to speed, integration with dev tools, price
Code Analysis Continuum

Static Analysis
- Inspection
- Compilation
- Syntax, entry errors

Dynamic Analysis
- Code coverage
- Functional verification
- Error regression

Accuracy Safety

Compliance

© 2013 Altera Corporation—Public
Barriers to Dynamic Analysis

Intrusive
- Requires code change
- Not testing production code
- Impacts performance, power, safety measurements

Not Deterministic
- Reliant on process, humans
- Ensure sufficient data set

Lack of COTS
- Solutions for Windows, Web
- Less solutions for embedded

Specific to Embedded
- Logging consumes resources, I/O
- Must add code for logs
- Component failure
Higher level of quality requirements for embedded

What unsafe embedded code could expose:
- In Comms/NW
  - Proprietary, sensitive data
  - Black-outs, service outages
- In Automotive/Transportation
  - Least: map or guidance issues
  - Most: control issues such as throttles, steering
- In Industrial
  - Manufacturing quality
  - Human endangerment

<table>
<thead>
<tr>
<th>Exposure</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unauthorized access, penetration</td>
<td>3.7%</td>
</tr>
<tr>
<td>Theft of info</td>
<td>16.3%</td>
</tr>
<tr>
<td>Abuse, denial of service</td>
<td>27%</td>
</tr>
<tr>
<td>Others</td>
<td>53%</td>
</tr>
</tbody>
</table>

$ The COST $
- Product revisions
- Recall
- Truck rolls
- Catastrophic failure
Code Safety is Hard!

- Debugging requires specialty equipment
  - Cost and area for connectors on boards
  - Custom protocols may not be supported

- Inability to capture errors in run-time
  - Lack of reporting, recording
  - Errors not always visible
  - Requires large data samples for test bench

- Embedded code is often ‘C’ code
  - Highly portable, highly abstracted
  - Lack of type safety
  - Unmanaged
Instrumenting the Virtual Platform

- “Voice of the hardware” w/out debuggers
- Proactive, not reactive
- Ensures proper operation of embedded code in programmable, processor space

**Simulator Method**
- Add hooks to functional “ticks”, e.g. quantum, instruction, transaction
  - Imperas VAP Tools
  - Modify OSCI engine

**Model Method**
- SystemC functional models w/ conditional code
- API to models
  - ARM Fast Models
  - Synopsys Tools for VDK
Example: verifying resource assignments in an AMP system

1. Platform assembly
2. Verification of platform
3. Set up of basic tests
4. Add tests for AMP resources
eSW Dynamic Analysis:

- Imperas M*SDK, including Multi-Processor Debug (MPD) Verification Analysis and Profiling (VAP) API
- Custom intercept library to monitor memory accesses, ensuring separation of Linux and \( \mu C/OS II \)
  - Includes table defining memory map
  - Adds debugger command to control: enable/disable
  - No impact on performance when disabled

Platform Modeling:

- Imperas’ intercept technology works with any processor model
  - Enables heterogeneous modeling
  - With TLM wrapper, models instrumented in M*SDK have been used in OSCI environment

- VAP tools are x86 host native and independent of simulator engine
  - Same models and tools developed with M*SDK may be used in any modeling toolbench
Example Findings

- **Bug found in Linux preemptive scheduling**
  - Symptom: Boots but does not switch tasks properly
  - Cause: Incorrect register read from timer0

- **Bug found in Linux accesses of GIC registers**
  - Symptom: Unrealized
  - Cause: Using cpu base when distributor base needed

- **Bugs found in uC/OS-ii GIC register accesses**
  - Symptoms: *Unrealized*
  - Cause:
    - Access ICDICER1 to 8 when only 0 to 7 exist
    - Access ICDIPTR08 to 63 when only 00 to 55 exist
Simple Bus Master Direct

```c
simple_bus_master_direct::main_action()

int mydate[4];
bool bReadVal;

if (true)
{
    bReadVal = bus_port->direct_read(&mydata[0], m_address);
    if (bReadVal)
    {
        bus_port->direct_read(&mydata[1], m_address+4);
        bus_port->direct_read(&mydata[2], m_address+8);
        bus_port->direct_read(&mydata[3], m_address+12);
    }
    if (m_verbose)
    {
        if (bReadVal)
        {
            sb_fprintf(stdout, "%g %s : mem[%x:%x] = (%x, %x, %x, %x)\n",
            sc_time_stamp().to_double(), name(), m_address,
            m_address+15,
            mydata[0], mydata[1], mydata[2], mydata[3]);
        }
        else
        {
            sb_fprintf(stdout, "%g %s : slave at address[%x] invalid\n",
            sc_time_stamp().to_double(), name(), m_address);
        }
    }
    //--- direct BUS interface
    //---

    bool simple_bus::direct_read(int *data, unsigned int address)
    {
        if (address%4 != 0) // address not word alligned
            sb_fprintf(stdout, "BUS ERROR --> address %04X not word alligned

        return false;
    } //}
    simple_bus_slave_if *slave = get_slave(address);
    if (isSlave)
    {
        m_slaveInvalid++;
        return false;
    }
    return slave->direct_read(data, address);
}
```
eSW Dynamic Analysis:

- Expand simple bus read/write to record addresses out of range
- Custom method to monitor bus accesses, ensuring presence and proper use of FPGA-based blocks on SoC
  - Adds interface to query/control access violations: halt, accumulate & report, etc.
  - No impact on performance in run-time

Platform Modeling:

- Memory map, interrupts
  - Hard coded (example) ->Parsed from QSys sopcinfo
  - Could also validate interrupts and other run-time (DTB) assignments
- UI
  - COM component for query OSCI
  - Tcl script from Virtualizer/VDK, etc.
Comparison

Simulator Method

- Custom simulator run-time
  - Imperas
  - OSCI – forward maintenance

- Interception in one place works for all models
  - E.g. memory map

Model Method

- Supports major modeling vendors
  - Some have custom API

- Easier to target “trouble spots”, specific blocks
  - Might have to instrument all
Summary

- **Benefits of Dynamic Analysis**
  - Additional coverage, not realized w/ traditional test

- **Instrumenting the Virtual Platform**
  - Multiple methods, suit a variety of tools and practices

- **Instrumenting the Virtual Platform for Dynamic Analysis**
  - Recognize value for the entire SW lifecycle

---

Provide Safe and Secure Code for Embedded Systems


Thank You