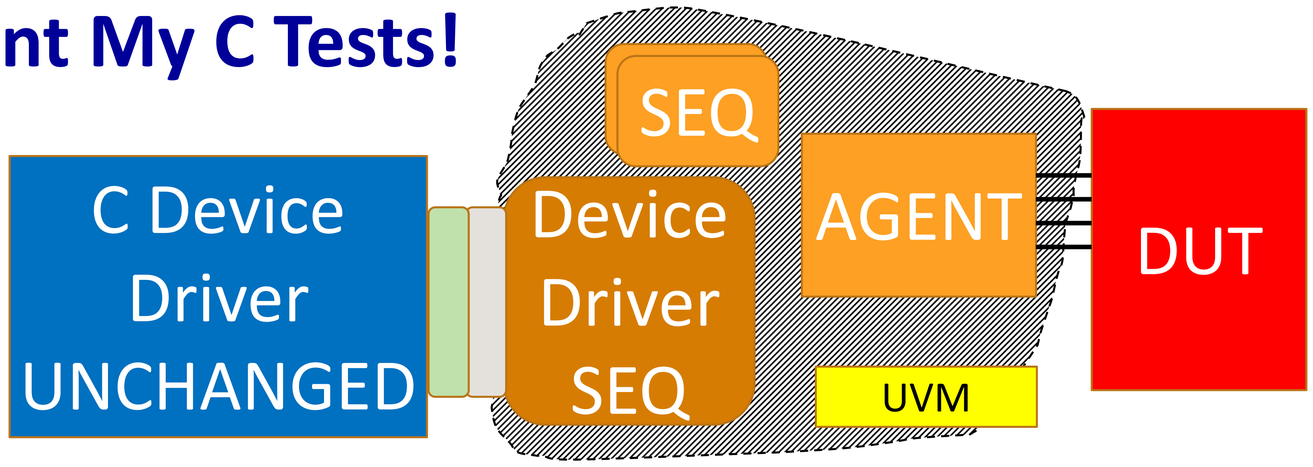
DESIGN & VERIFICATION DVCON 2014 CONFERENCE & EXHIBITION

UVM SchmooVM! – I Want My C Tests!



http://stuff.mit.edu/afs/sipb/contrib/linux/include/asm-generic/io.h

```
16 static inline u32 readl(void __iomem *addr){
18     return *(u32 __force *) addr;
19 }
20
21 static inline void writel(u32 b, __iomem *addr){
23     *(u32 __force *) addr = b;
24 }
```

http://stuff.mit.edu/afs/sipb/contrib/linux/drivers/usb/host/xhci.c

xhci.c

```
 1 /*
 2  * xHCI host controller driver
 3  *
 4  * Copyright (C) 2008 Intel Corp.
 5  *
 6  * Author: Sarah Sharp
   ...
23 #include <linux/pci.h>
24 #include <linux/irq.h>
   ...
82 /*
83  * Disable interrupts and begin the
84  * xHCI halting process.
85  */
86 void xhci_quiesce(struct xhci_hcd *xhci)
87 {
88     u32 halted;
89     u32 cmd;
90     u32 mask;
91
92     mask = ~(XHCI_IRQS);
93     halted = readl(&xhci->op_regs->status)
94         & STS_HALT;
95     if (!halted)
96         mask &= ~CMD_RUN;
97
98     cmd = readl(&xhci->op_regs->command);
99     cmd &= mask;
100    writel(cmd, &xhci->op_regs->command);
101 }
```

os_layer.c

```
 1 #include <stdio.h>
 2 #include "dpiheader.h"
 3
 4 int
 5 my_readl(int addr) {
 6   int rdata;
 7   sv_read(&addr, &rdata);
 8   return rdata;
 9 }
10
11 my_writel(int addr, int data){
12   sv_write(&addr, &data);
13 }
```

SV Gasket

```
27   export "DPI-C"          task      sv_read;
28   export "DPI-C"          task      sv_write;
   ...
39   task sv_read( addr_t addr, output data_t data);
40     DEVICE_A_SEQ_BASE c;
41     $cast(c, mapper_pkg::MapPidToClassHandle::get());
42     c.read(addr, data);
43   endtask
44
45   task sv_write( addr_t addr, data_t data);
46     DEVICE_A_SEQ_BASE c;
47     $cast(c, mapper_pkg::MapPidToClassHandle::get());
48     c.write(addr, data);
49   endtask
```

Sequence

```
65   class device_driver_sequence
66           extends DEVICE_A_SEQ_BASE;
   ...
74     task read(  addr_t addr, output data_t data);
75       mem_item t;
76       t = mem_item::type_id::create("item");
77       t.rw = 1;
78       t.addr = addr;
79       start_item(t);
80       finish_item(t);
81       data = t.data;
84     endtask
85
86     task write( addr_t addr,         data_t data);
87       mem_item t;
88       t = mem_item::type_id::create("item");
89       t.rw = 0;
90       t.addr = addr;
91       t.data = data;
92       start_item(t);
93       finish_item(t);
96     endtask

118    task body();
119      sequencerA sequencer;

122      $cast(sequencer, m_sequencer);
124      // Start two instances of the "device driver"
125      fork
126        begin
127          mapper_pkg::MapPidToClassHandle::set(this);
128          c_device_driver();
129        end
130        begin
131          mapper_pkg::MapPidToClassHandle::set(this);
132          c_device_driver();
133        end
134      join
136    endtask
137  endclass
```

```
 1 #include <stdio.h>
 2 // Device Driver Layer.
 3 // This code is not really a device
 4 // driver, but writes values to
 5 // addresses using writel() and readl()
 6
 7 char msg[4096];
 8
 9 int
10 c_device_driver() {
11   int addr, data, rdata;
12
13   for (addr = 16; addr < 100; addr+=4) {
14     data = addr<<8;
16     my_writel(addr,   data);
17     rdata = my_readl (addr);
18
...
29   }
30   return 0;
31 }
```

device_driver.c

```
 1 package mapper_pkg;
 2   import uvm_pkg::*;
 3
 4   class MapPidToClassHandle;
 5     static uvm_object class_handle_table[process];
 6
 7     static function uvm_object get();
 8       return class_handle_table[process::self()];
 9     endfunction
10
11     static function void set(uvm_object class_handle);
12       class_handle_table[process::self()] = class_handle;
13     endfunction
14
15     static function string get_full_name();
16       uvm_object class_handle = get();
17       return $sformatf("%s::%0d",
18         class_handle.get_full_name(),
19         process::self());
20     endfunction
21   endclass
22 endpackage
```

“Mapper”
Thread to Class Handle

```
139 class device_driver_test extends test;
140   `uvm_component_utils(device_driver_test)
146
147   task run_phase(uvm_phase phase);
148     device_driver_sequence
149       device_driver_seq1, device_driver_seq2;
152     phase.raise_objection(this);
153     for(int i = 0; i < 10; i++) begin
154       device_driver_seq1 =
155         device_driver_sequence::type_id::create("seq1");
156       device_driver_seq2 =
157         device_driver_sequence::type_id::create("seq2");
158       fork
159         device_driver_seq1.start(i1_agentA.sequencer);
160         device_driver_seq2.start(i2_agentA.sequencer);
161       join
162     end
163     phase.drop_objection(this);
165   endtask
166 endclass
```

The Test

Start Here

```
vlib work
vlog -dpiheader dpiheader.h ... dut.sv agentA*.sv top.sv device_driver.c os_layer.c
vsim ... +UVM_TESTNAME=device_driver_test top
```


Rich Edelman
rich_edelman@mentor.com
Raghu Ardeishar
raghu_ardeishar@mentor.com
Mentor Graphics