UVM based Hardware/Software Co-Verification of a HW Coprocessor using Host Execution Techniques

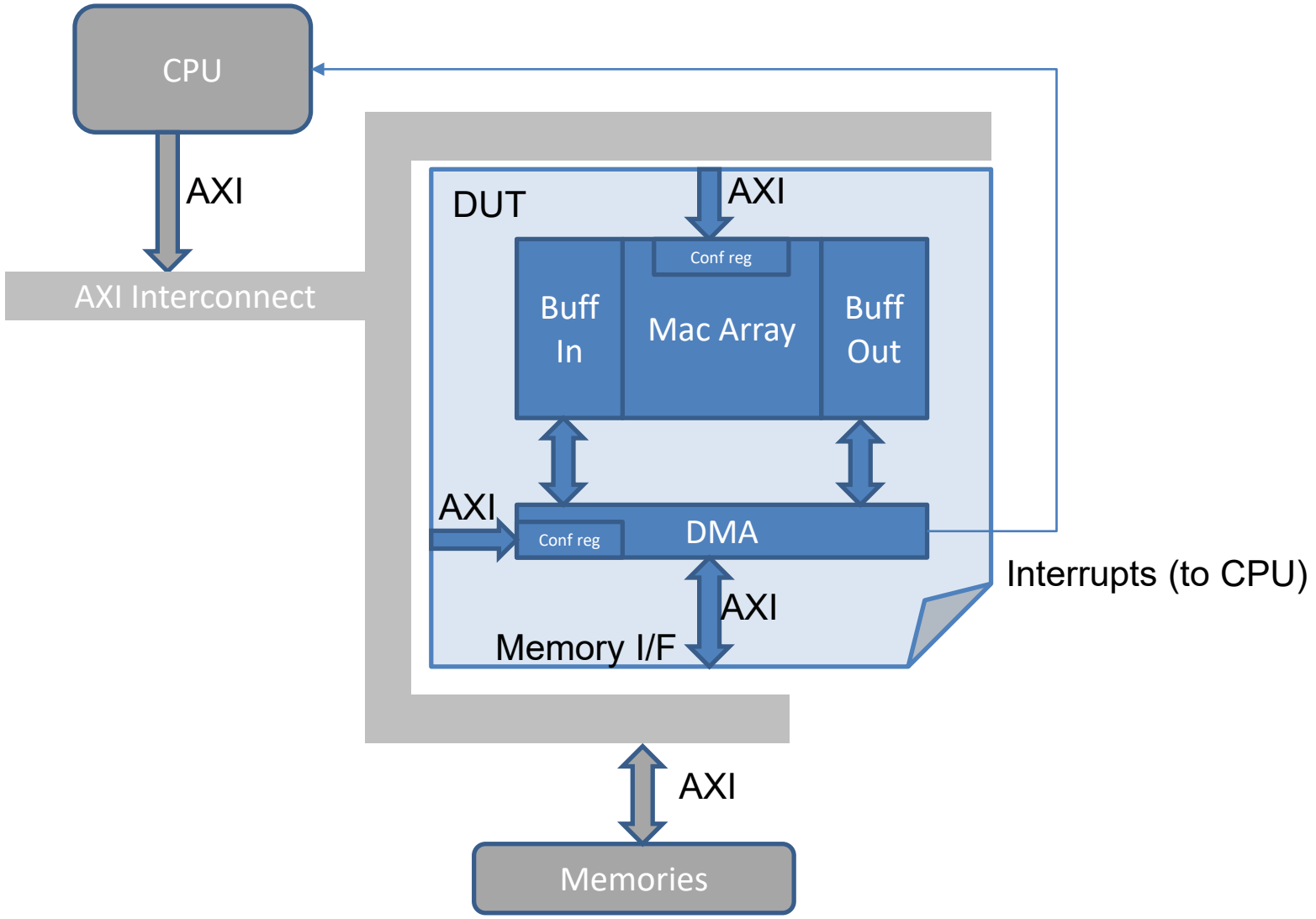
François Cerisier, Christian Rivier, Andrea Battistella
AEDVICES Consulting

Arnaud Grasset, Thales Research & Technologies

UVM Based vs Software-Driven testing

- UVM Environment
  - Test Sequences
  - Strong ability for constrained random testing
  - Advanced verification using monitors, scoreboards, assertions
  - Little vertical reuse
  - No horizontal reuse
- Software Driven Testing
  - C testing, running (or as if) on the core processor
  - Use of the Software API stack
  - Little random testing, no constrained random
  - No vertical reuse
  - Horizontal reuse
- How to get the best of both with minimum limitations and setup complexity ?

The DUT



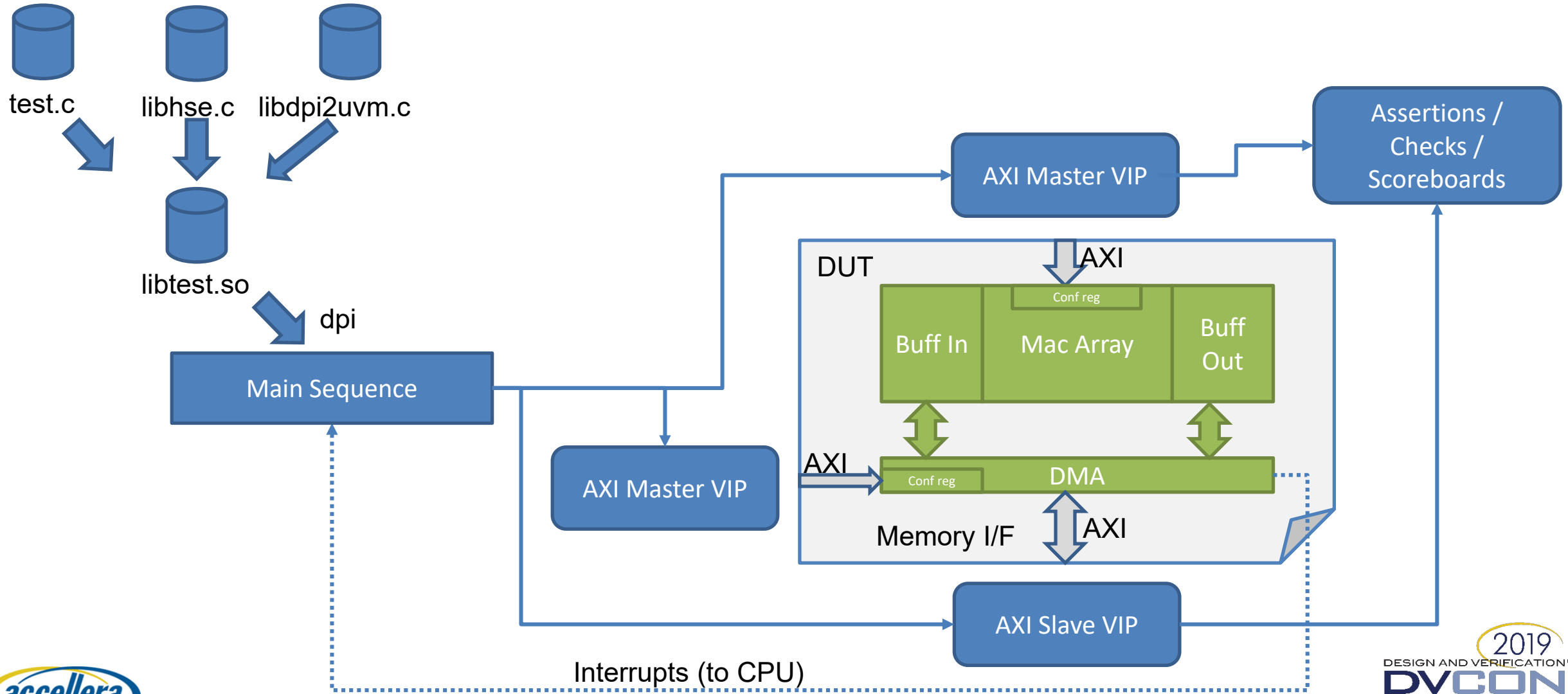
The Verification Goals and Constraints

- Verify the signal processing of the design
- Verify the control and synchronization
- No virtual CPU to simulate, No ISS (easily) available
- Existing UVM setup using AXI QVIP from Mentor
- Need either to use software drivers or redevelop the corresponding sequences
- Need simulations to ease debug
- Final proof is the FPGA with the CPU, not the simulation.

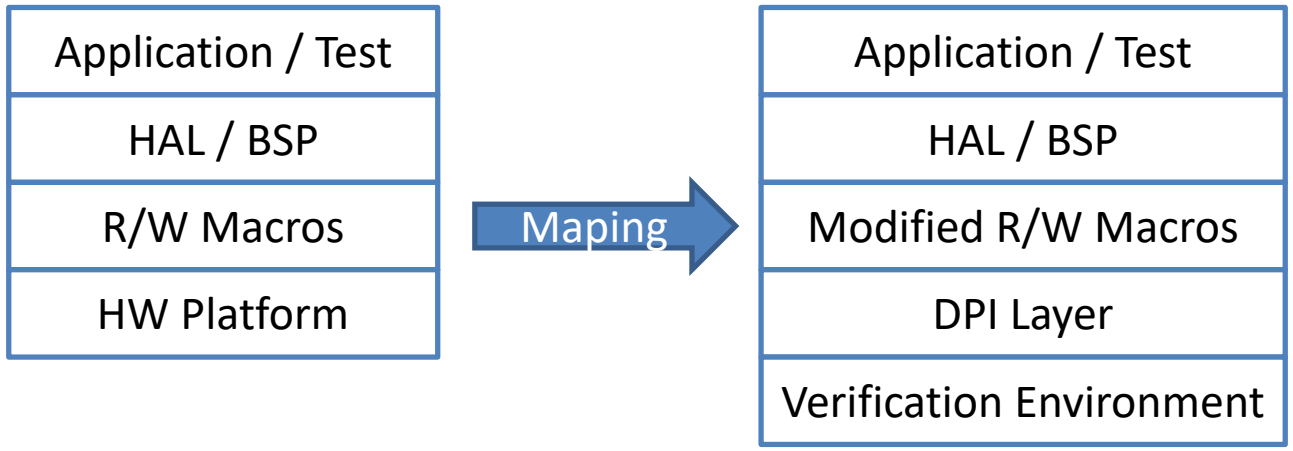
Software Driven UVM verification environment

- Use the advantages of UVM, while still doing software driven tests
- Still need UVM and SystemVerilog for
  - scoreboarding,
  - assertions,
  - AXI bus driving/monitoring with on-the-shelf AXI VIP.

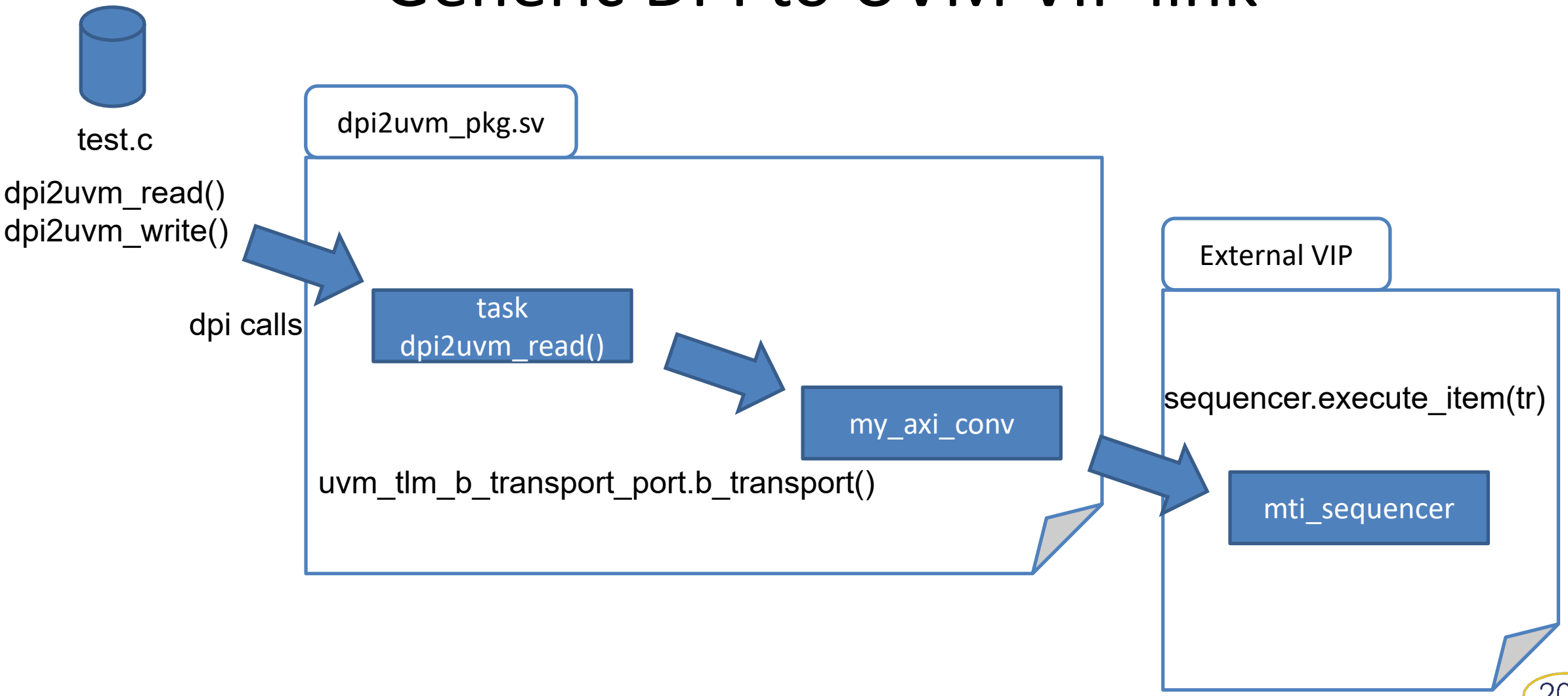
Software Driven UVM environment



Mapping Software to Verification



Generic DPI to UVM VIP link



DPI to UVM Code Snippet

```
extern void dpi2uvm_read16 (int,uint16_t*);
void main() {
   dpi2uvm_read16( address , data );
```

C

```
virtual class dpi2uvm_tlm_conv_container extends dpi2uvm_tlm_conv_base_container;
  /// TLM Conversion Tasks to VIP Single Access Sequence
  pure virtual task b_transport(dpi2uvm_trans tr, uvm_tlm_time delay);
```

inherits from

```
export "DPI-C" task      dpi2uvm_read16;
uvm_tlm_b_transport_port #(dpi2uvm_trans) transport;

task dpi2uvm_read(dpi2uvm_address_t addr,
                  output byte unsigned read_val[],
                  input int size);

    uvm_tlm_generic_payload trans;
    trans.set_address(addr);
    trans.set_read();
    trans.set_data_length(size);

    transport.b_transport(tr,tt);

    trans.get_data(read_val);
```

SystemVerilog DPI calls

```
class dpi2uvm_demo_vip_custom_conv extends dpi2uvm_tlm_conv_container;

  /// User Conversion of the demo_dpi_trans to the actual User Sequence Call
  virtual task b_transport(uvm_tlm_generic_payload tr, uvm_tlm_time delay);
    axi_trans = new("AXI_TRANS");

    // Alternate non-random code:
    if ( tr.is_write() )
      axi_trans.data    = {tr.m_data[3],tr.m_data[2],tr.m_data[1],tr.m_data[0]};
    axi_trans.address = tr.m_address;
    axi_trans.direction = (tr.is_write() ? WRITE : READ);

    // Send the transaction to the sequencer
    this.sequencer.execute_item(cvtd_trans);
  endtask
```

DPI to UVM Link

DPI to UVM Limitations

- Need to have read/write calls
  - Macros: can be redefined
  - Functions: can be linked to another driver
- First driver from Xilinx was ok.
- Other Software contained pointers

test.c

p* = 12;
Val = &p;

How to deal with such software drivers ?

Host Code Execution Principles

- Define memory address space as « protected »
- Pointers to memories will hit a segmentation fault
- Implement a fault handler that catches the memory access
- Call the memory read/write functions

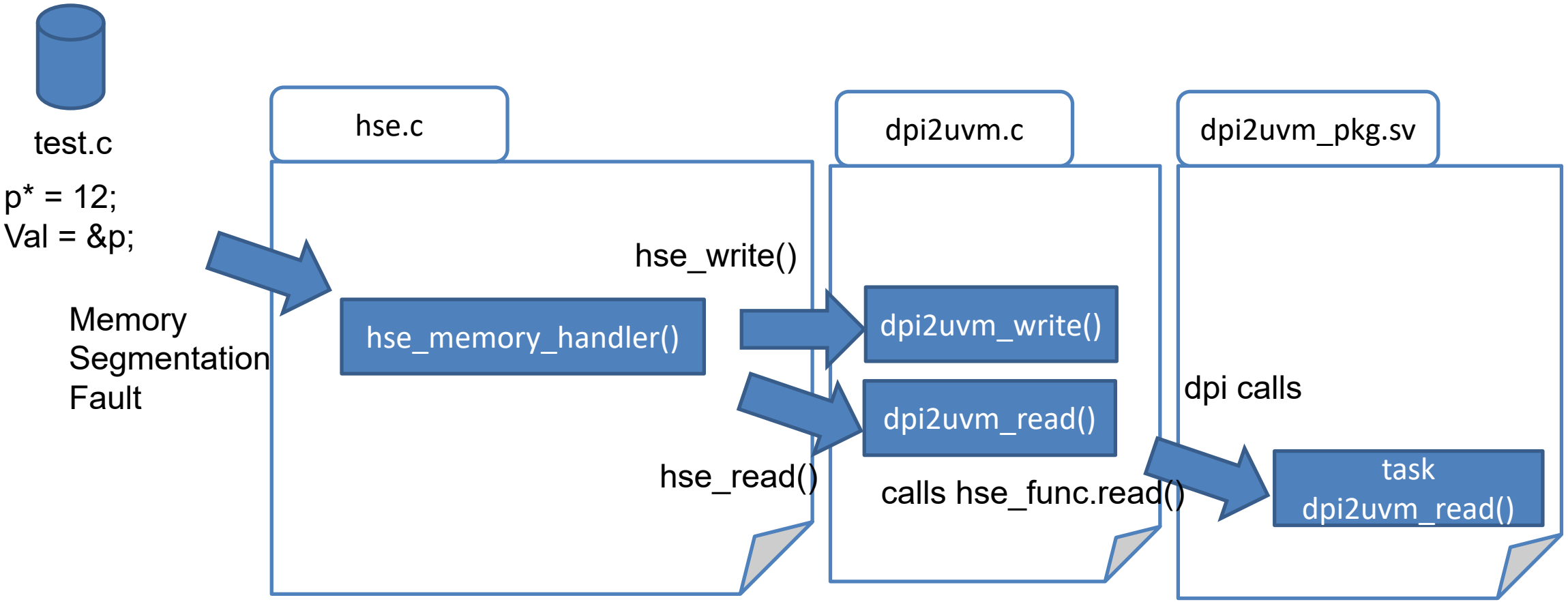
Linux Memory Map Fault Handler

```
void hce_protect_memory_map(char *__buffer)
{
  int pagesize = sysconf(_SC_PAGE_SIZE);
  _buffer = (char *) mmap ( (void*)addr,
                            pagesize,
                            PROT_READ | PROT_WRITE | PROT_EXEC ,


  if (mprotect(__bu
    hce_fatal("mpro
}
```

```
// Handler links to external read/write functions
static void hse_memory_handler(){
        uint8_t temp = 0x0;  hse_read8(hhs.g_addr, &temp);
}
void hse_read8(int addr, char* data) {
    if ( hse_func.read8 != NULL && hse_init_done != 0) {
        hse_func.read8(addr,data);
    }
}
```

Generic Host Software/Code Execution



Generic Software Driven UVM

C Code	HSE function	DPI Calls	VIP action taken (equivalence through the TLM conversion)
P* = 12;	hse_write(&p , 12)	dpi2uvm_write (addr, 12)	`uvm_do_with (REQ , { REQ.address == local::address; REQ.data == 12; REQ.direction == WRITE; })
val = &p;	hse_read(&p , &val)	dpi2uvm_read(addr, data)	`uvm_do_with (REQ , { REQ.address == local::address; REQ.direction == READ; }) return REQ.data;

Considered Alternatives

Techniques	Pros	Cons
ISS	Instruction Accurate Transaction Accurate (also on the instruction fetch) Potentially Cycle Accurate	Development time On-the-shelf component availability On-the-shelf component cost
QEMU	Instruction Accurate Transaction Accurate (also on the instruction fetch) Potentially Cycle Accurate	Availability for the given processor No easy/generic integration
Simulated RTL	Cycle Accurate	Slow Encrypted FPGA soft core (no RTL available)
FPGA Only	Use the real CPU	Little debug No advanced verification (assertions, scoreboards, …)

Results

- Integration of the generic DPI2UVM : 1/2 day
- Getting the Xilinx DMA software driver working: < 1 day
- Bugs:
  - Synchronization issue in RTL.
  - Software bug in HAL

Questions

Zzzzz !

Guidelines (1)

- Please keep the default font size for main lines at 28pt (or 26pt)
  - And use 24pt (or 22pt) font size for the sub bullets
- Use the default bullet style and color scheme supplied by this template
- Limited the number of bullets per page.
- Use keywords, not full sentences
- Please do not overlay Accellera or DVCon logo's
- Check the page numbering

Guidelines (2)

- Your company name and/or logo are only allowed to appear on the title page.
- Minimize the use of product trademarks
- Page setup should follow on-screen-show (4:3)
- Do not use recurring text in headers and/or footers
- Do not use any sound effects
- Disable dynamic slide transitions
- Limit use of animations (not available in PDF export)

Guidelines (3)

- Use clip-art only if it helps to state the point more effectively (no generic clip-art)
- Use contrasting brightness levels, e.g., light-on-dark or dark-on-light. Keep the background color white
- Avoid red text or red lines
- Use the MS equation editor or MathType to embed formulas
- Embed pictures in vector format (e.g. Enhanced or Window Metafile format)