Pumping Up Test Development with Task Based, C-callable, UVM based Tests

Rich Edelman, Siemens EDA

Didan Francis, Siemens India

SIEMENS

accellera SYSTEMS INITIATIVE™

2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Logic Simulation and Verification

- Goal: Generate more stimulus with less work

- UVM is a standard and has a large ecosystem to leverage
- Run many
- Random seeds

- Many C programmers
- Other languages

UVM Testbench Design

- UVM Architects
  - UVM architecture
  - UVM sequence design
  - Task based API

- Test writers
  - Algorithms
  - Tests
  - Call the task based API

Task Based Tests

- UVM Sequences can be hard to get your head around

- Task based tests are easy to understand
  - READ
  - WRITE_BLOCK
  - WRITE_PACKET

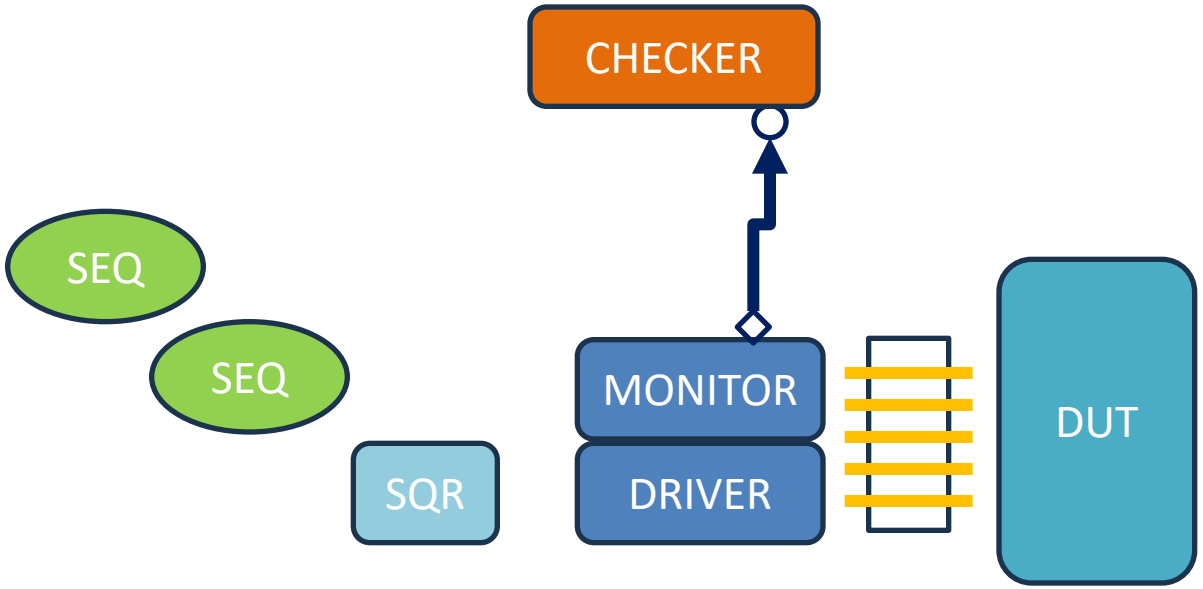
Expand Testing

- Use UVM
- Use C
- But → Make them work together

A Typical UVM Setup

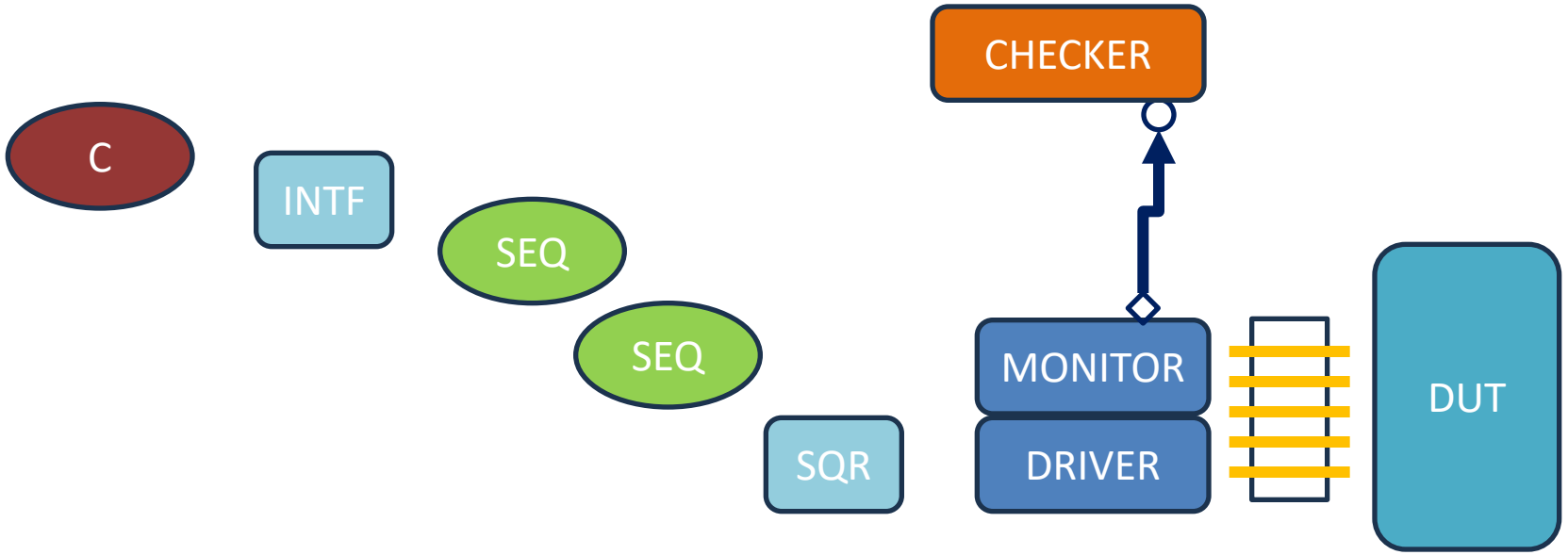
- Driver
- Monitor
- Checker
- Sequencer
- Sequences
- Interface – wires

- DUT

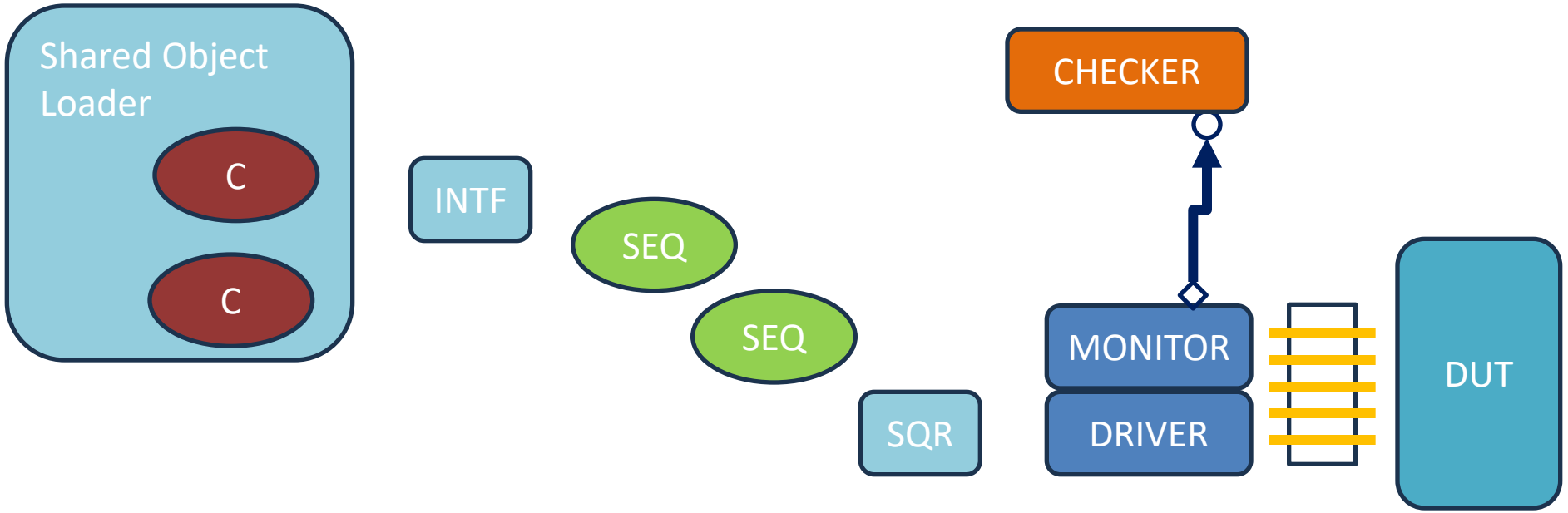


A Typical UVM Setup with C

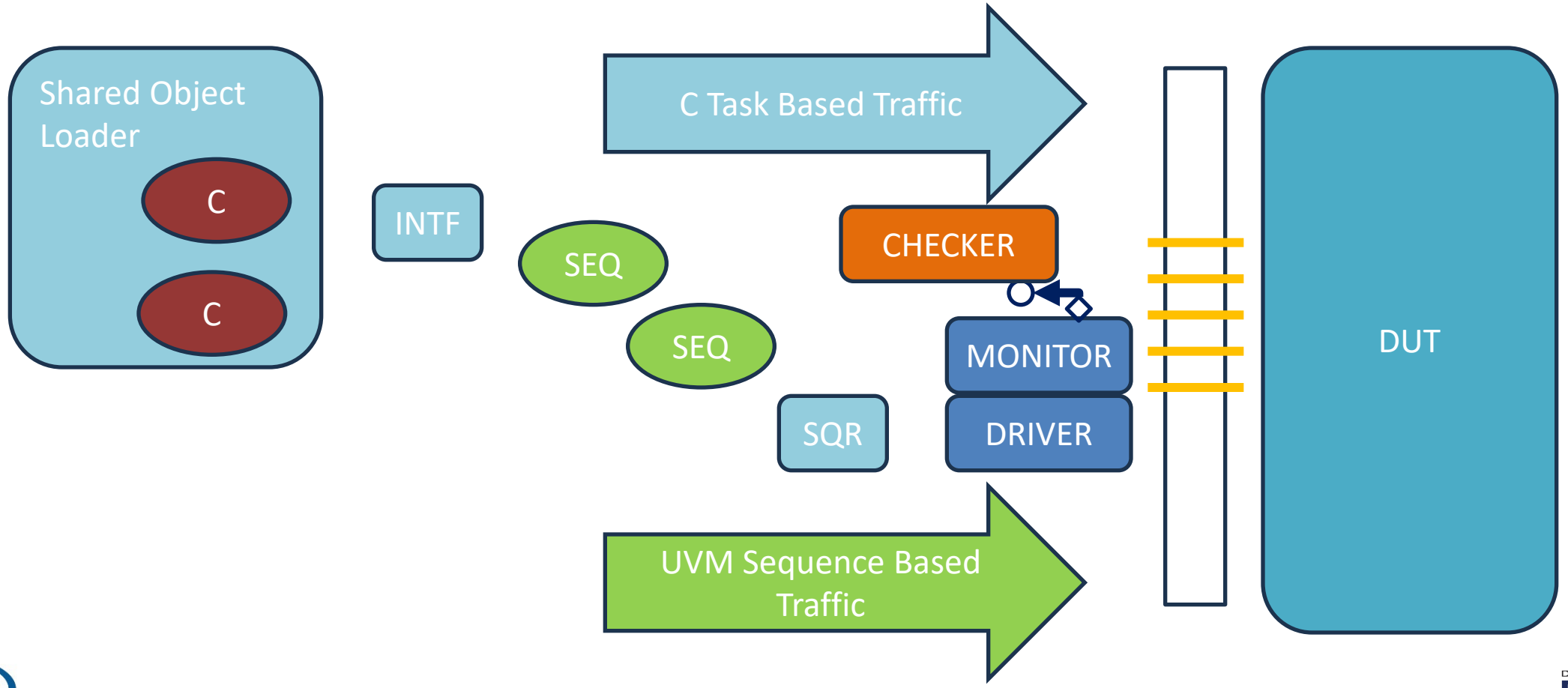
- Add C via a special kind of interface



A Typical UVM Setup many C shared objects



UVM Traffic + Task Traffic – Perfect Together



Why C?

- Lots of C programmers
- Easy to manage with SystemVerilog DPI-C
- What about other languages?
  - Rust
  - Python
  - PSS
- All you need is the ability to create a C-callable shared object → .so

Sample C Test

```
int
zinterface_start_test_program(int index, const char *name, int start_addr) {
    int ...
    ...
    // Repeat 10 times, changing the data
    for (dataloops = 0; dataloops < 10; dataloops++) {
      // Repeat 10 times - writing 10, and reading 10
      start_addr = original_start_addr;
      for (loops = 0; loops < 10; loops++) {
        for (addr = start_addr; addr < start_addr+10; addr++) {
          data = addr + 1000 + dataloops;
          SV_write(index, addr, data);
        }
        for (addr = start_addr; addr < start_addr+10; addr++) {
          SV_read(index, addr, &data);
          if (data != addr + 1000 + dataloops) {
            ...Error Checks
          }
        }
```

Write

Read

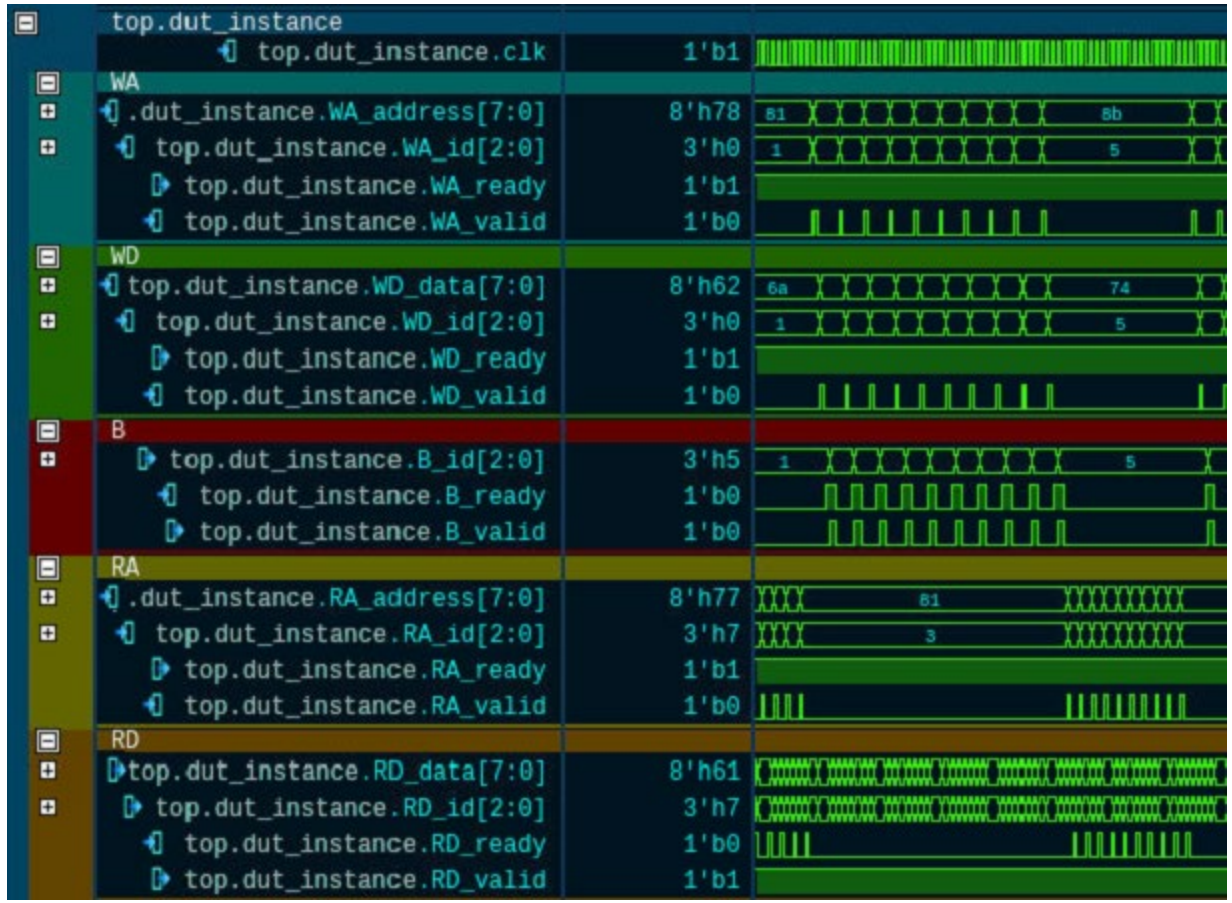
Self-checking

What about threading?

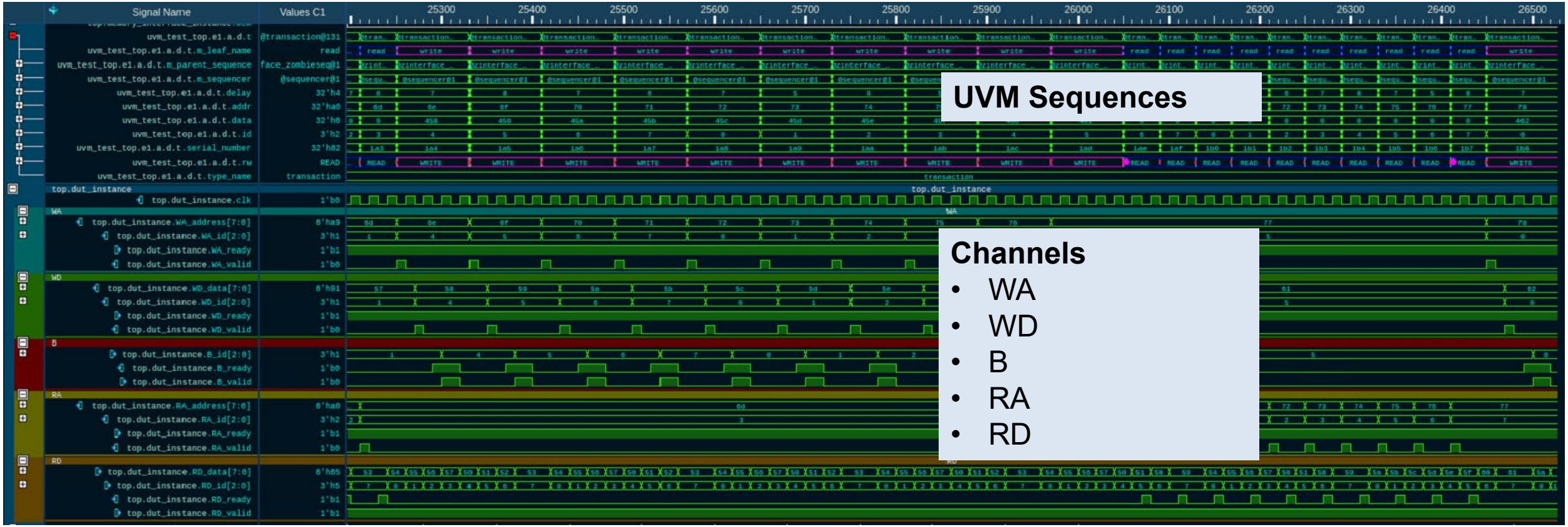
- SystemVerilog is “conceptually threaded”
  - The simulation naturally has parallel processes – threads
  - The simulator has the threads take turns as each thread “yields”
    - Hits a # delay
    - Hits a wait()
    - Hits a @
    - …
  - A thread can call C → So C code is threaded in SystemVerilog
    - In the same way initial and always blocks are threaded
  - You must write thread safe C code

Test vehicle – ready/valid

```
interface dut_interface(input clk);
  reg [7:0] WA_address;
  reg [2:0] WA_id;
  reg WA_ready;
  reg WA_valid;

  reg [7:0] WD_data;
  reg [2:0] WD_id;
  reg WD_ready;
  reg WD_valid;

  reg [2:0] B_id;
  reg B_ready;
  reg B_valid;

  reg [7:0] RA_address;
  reg [2:0] RA_id;
  reg RA_ready;
  reg RA_valid;

  reg [7:0] RD_data;
  reg [2:0] RD_id;
  reg RD_ready;
  reg RD_valid;
endinterface
```



Lots of Events



Interface Management (lower level)

- Low level pin wiggling API

```
task send_WA(input reg[2:0] id, reg[7:0]address);
  WA_address = address;
  WA_id = id;
  WA_valid = 1;
  forever begin
    @(posedge clk);
    if ( WA_ready == 1) begin
      // xfer
      WA_valid = 0;
      @(negedge clk);
      return;
    end
  end
endtask
```

```
task send_WD(input reg[2:0] id, reg[7:0]data);
  WD_data = data;
  WD_id = id;
  WD_valid = 1;
  forever begin
    @(posedge clk);
    if ( WD_ready == 1) begin
      // xfer
      WD_valid = 0;
      @(negedge clk);
      return;
    end
  end
endtask
```

Interface Management (higher level)

- Build up APIs
  - Write_packet
  - Read_packet
  - Write_BLOCK
  - Read_BLOCK

```
task write(input reg[2:0] id, reg[7:0]address,
                              reg[7:0]data);
  send_WA(id, address);
  send_WD(id, data);
  wait_B(id);
endtask

task read(input reg[2:0] id, reg[7:0]address,
                         output reg[7:0]data);
  send_RA(id, address);
  wait_RD(id, data);
endtask
```

Dynamic Object Loader

```
...Include Files

int
c_load_shared_object(const char *name) {
    void *handle;
    int (*zinterface_start_test_program)(int, char *, int);

    handle = dlopen(name, RTLD_LAZY|RTLD_GLOBAL|RTLD_NOW);
    if (!handle) {
      // …Error handing
    }
    dlerror(); // Clear any existing error.

    zinterface_start_test_program = dlsym(handle, "zinterface_start_test_program");
    zinterface_start_test_program(1, "thread1", 100);
    return 0;
}
```

Loading Shared Objects from SystemVerilog

- Load shared objects from SystemVerilog

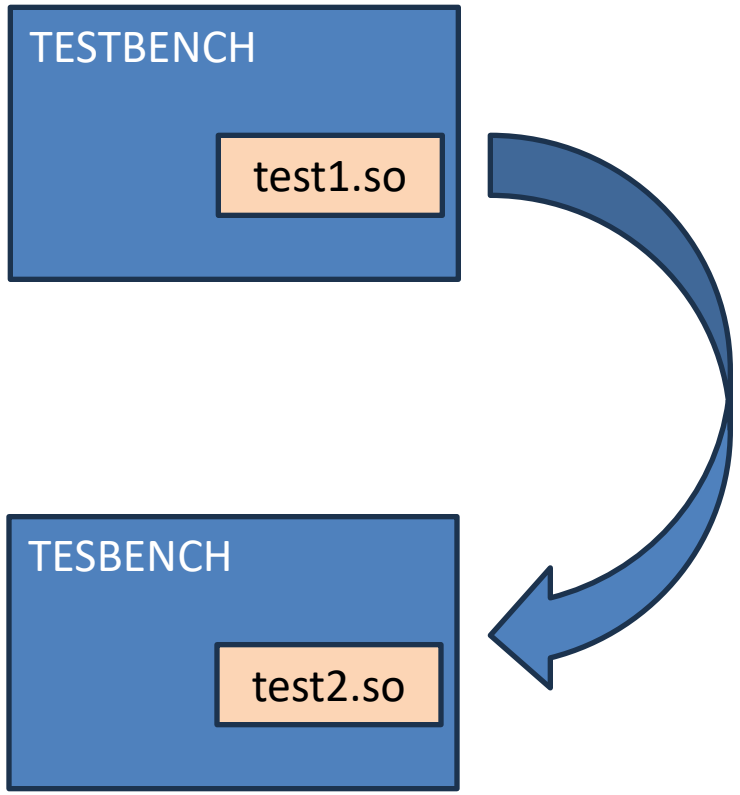
```
// LOAD & RUN
zif.sv_load_shared_object("./testprogram1.so");

// LOAD & RUN
zif.sv_load_shared_object("./testprogram2.so");
```

- Extend the features **Exercise for the reader**
  - Load via +arg command line option
  - Load from config file
  - Load from tcl script

Exercise for the reader

Restart and Checkpoint

- Standard tools for improved flows
- Restart with a NEW test
  - Load a different .so
- Checkpoint. Restore with a NEW test
  - +load=newtest.so



Questions

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 logos
- 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)