RESSL UVM Sequences to the Mat

Jeff McNeal & Bryan Morris
Verilab

SNUG San Jose
2015

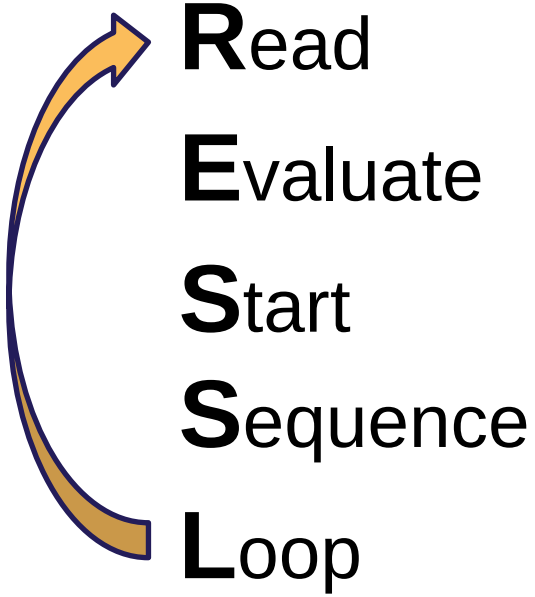
Agenda

- RESSL?
- Commands
- Uses
- Questions

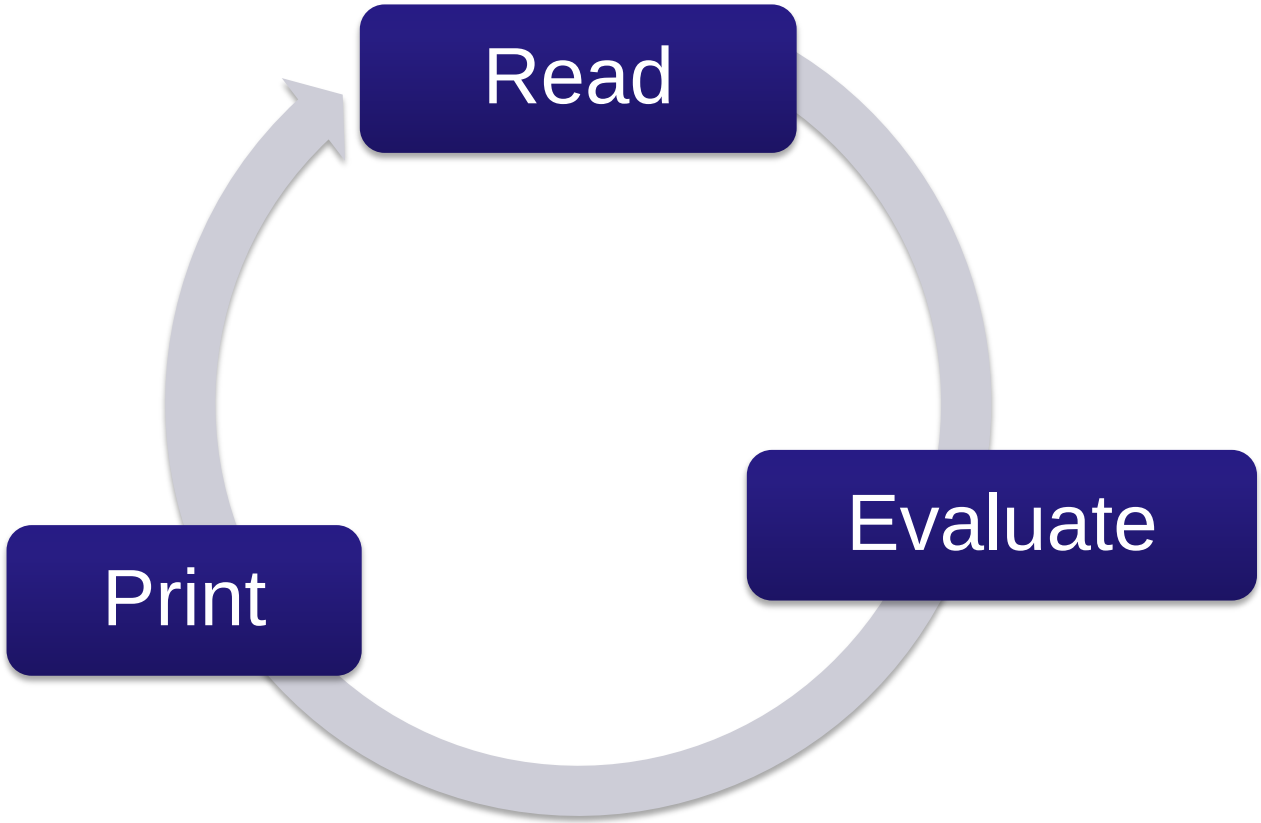
RESSL

Command Line Interpreter UVM/System Verilog

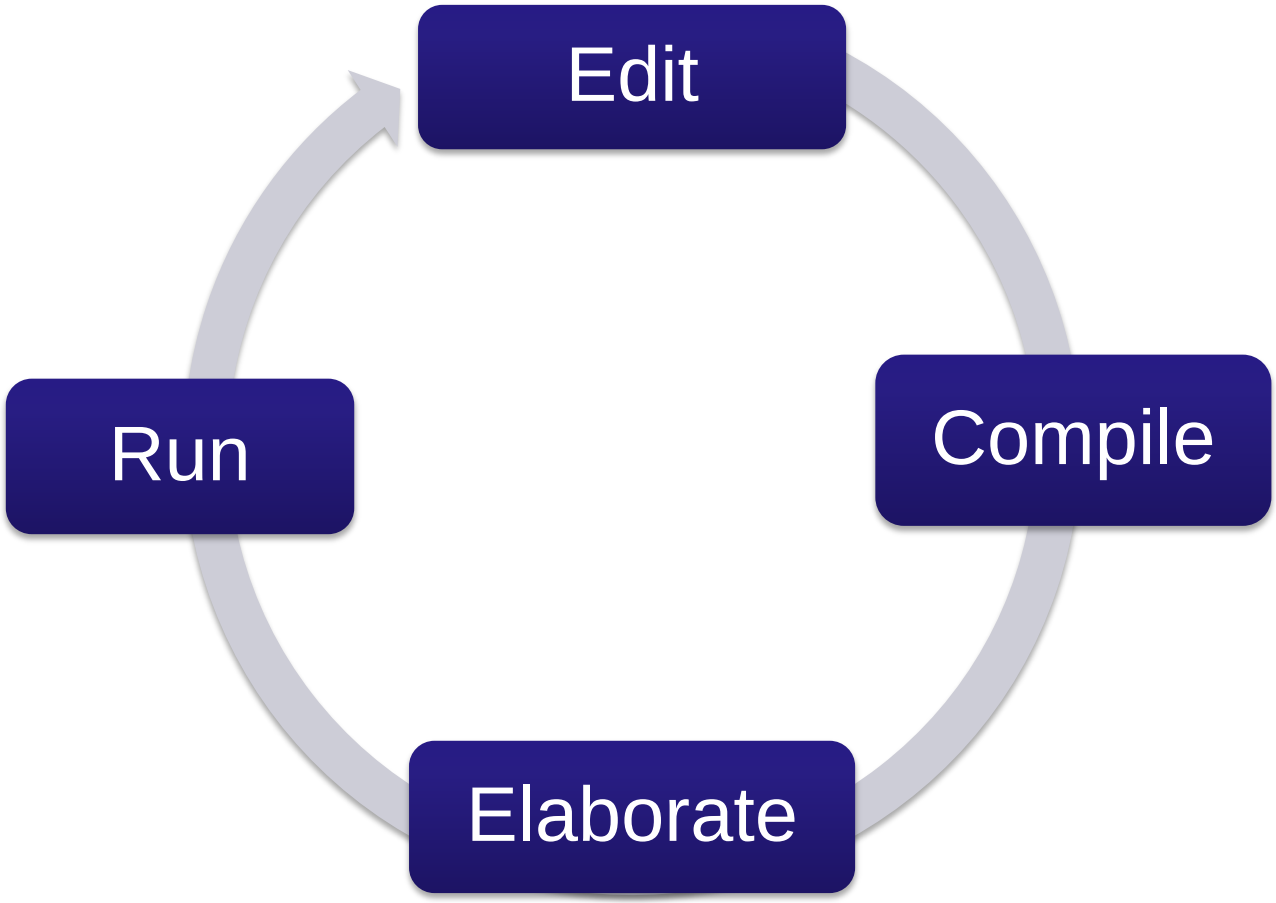
Interactive development using the UVM factory to create sequences



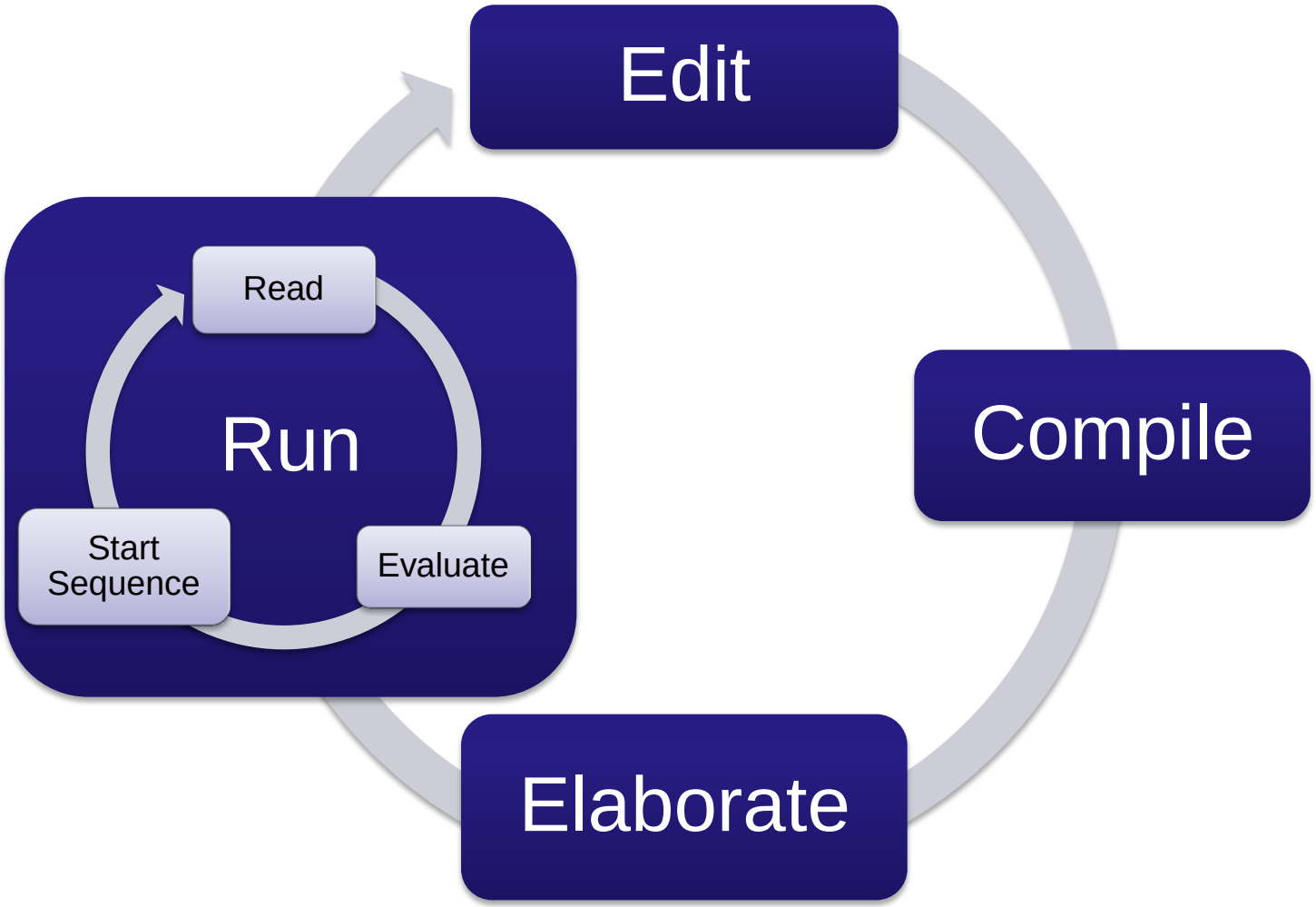
Lisp REPL



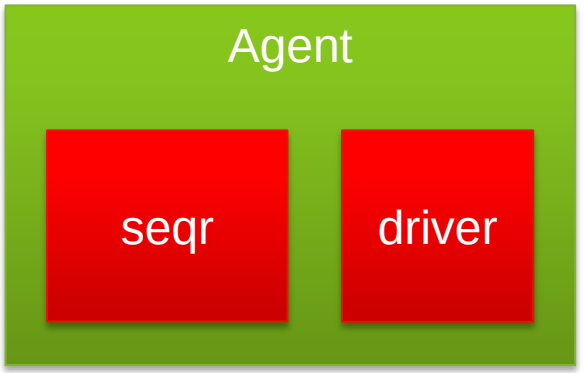
System Verilog Loop



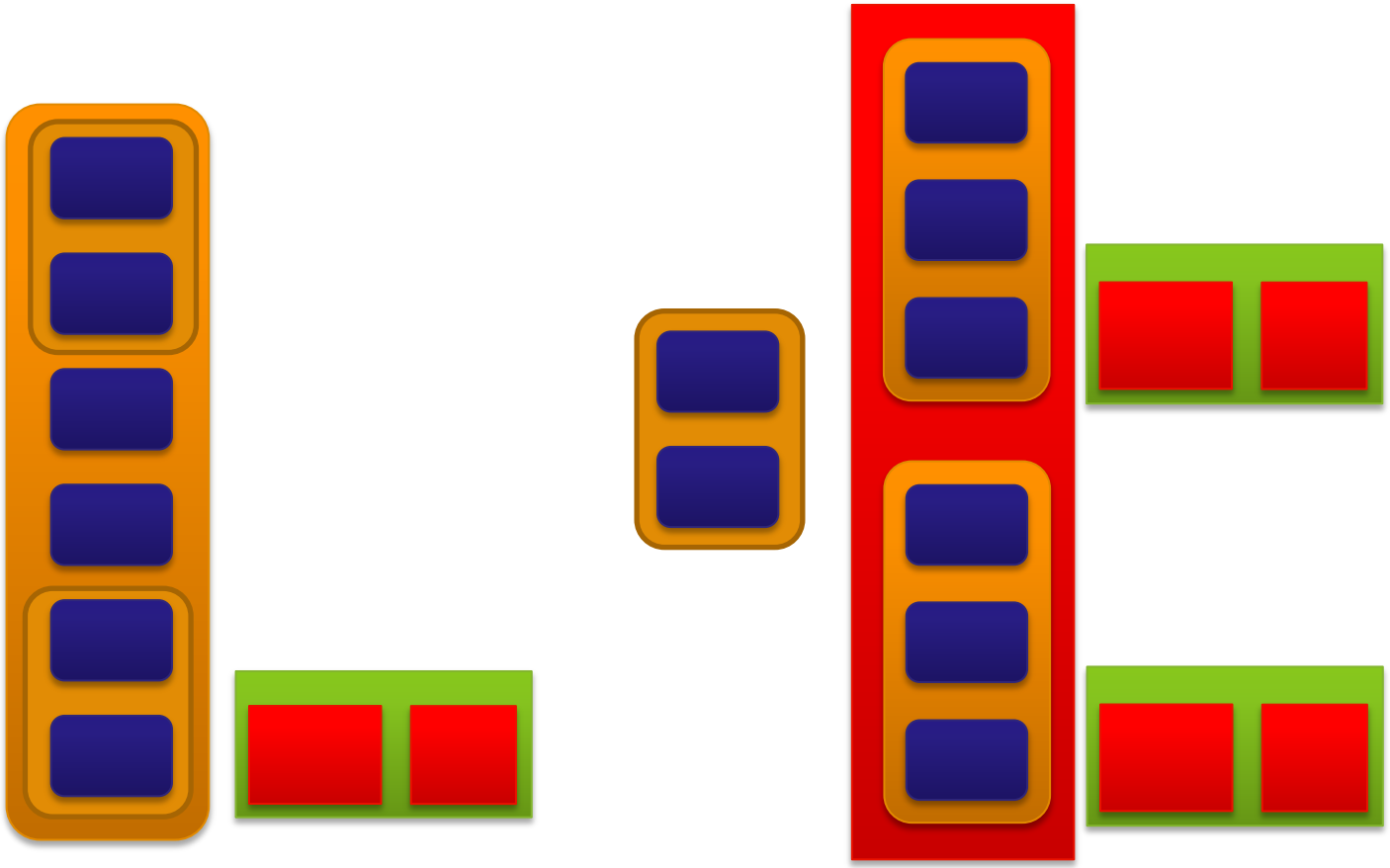
System Verilog RESSL



Current UVM

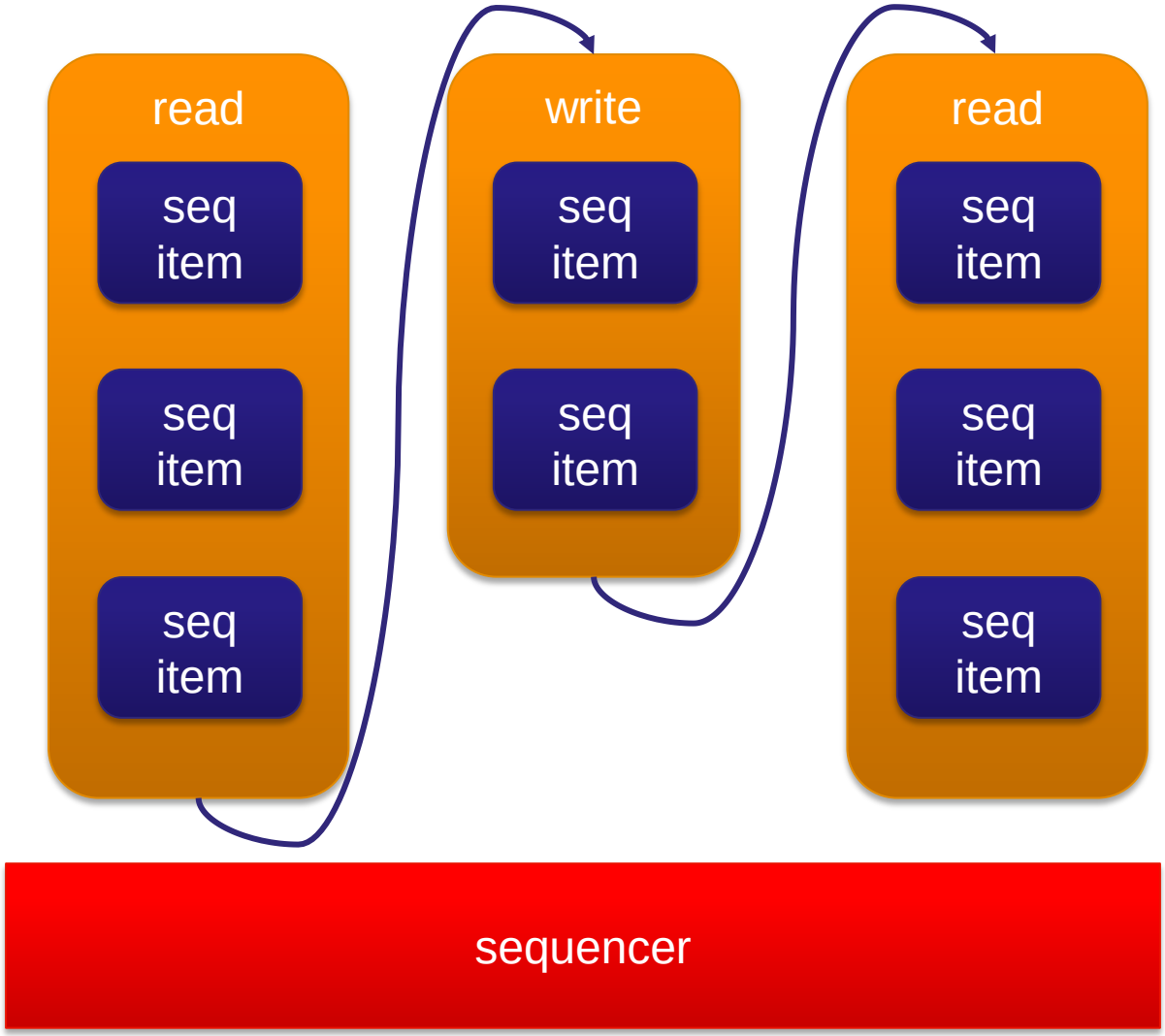


Complex Sequences



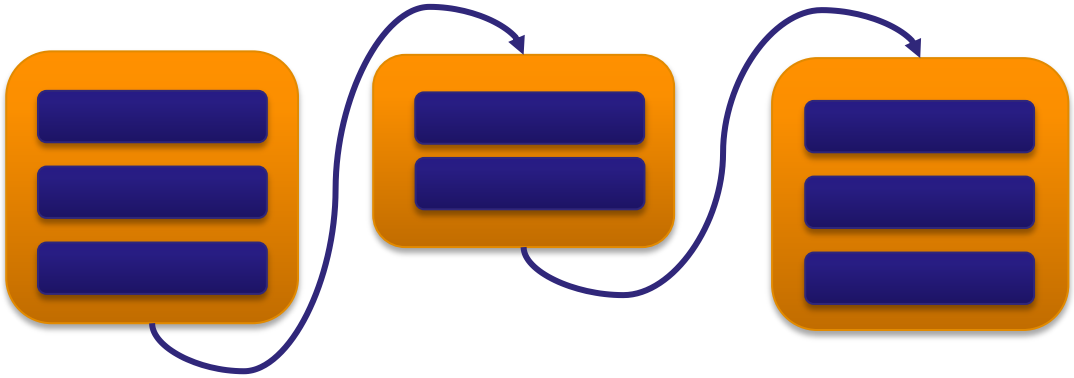
RESSL Sequence Registry

“rwr_seq” →



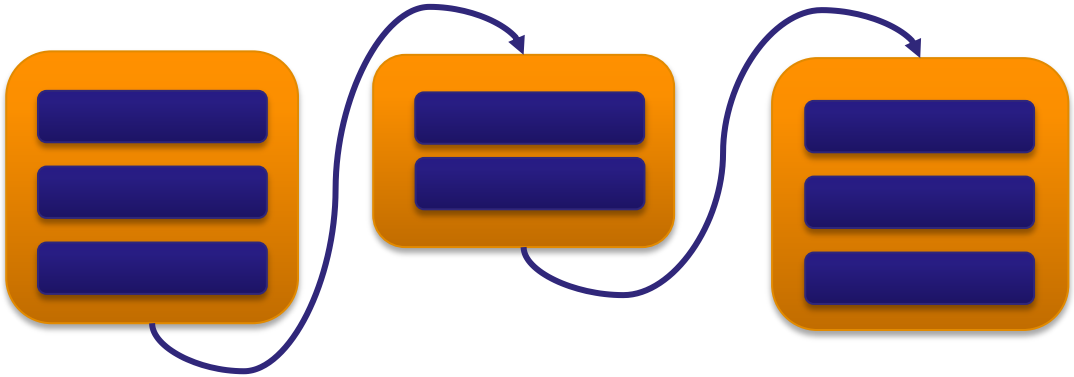
RESSL Sequence Registry

"rwr_seq" →



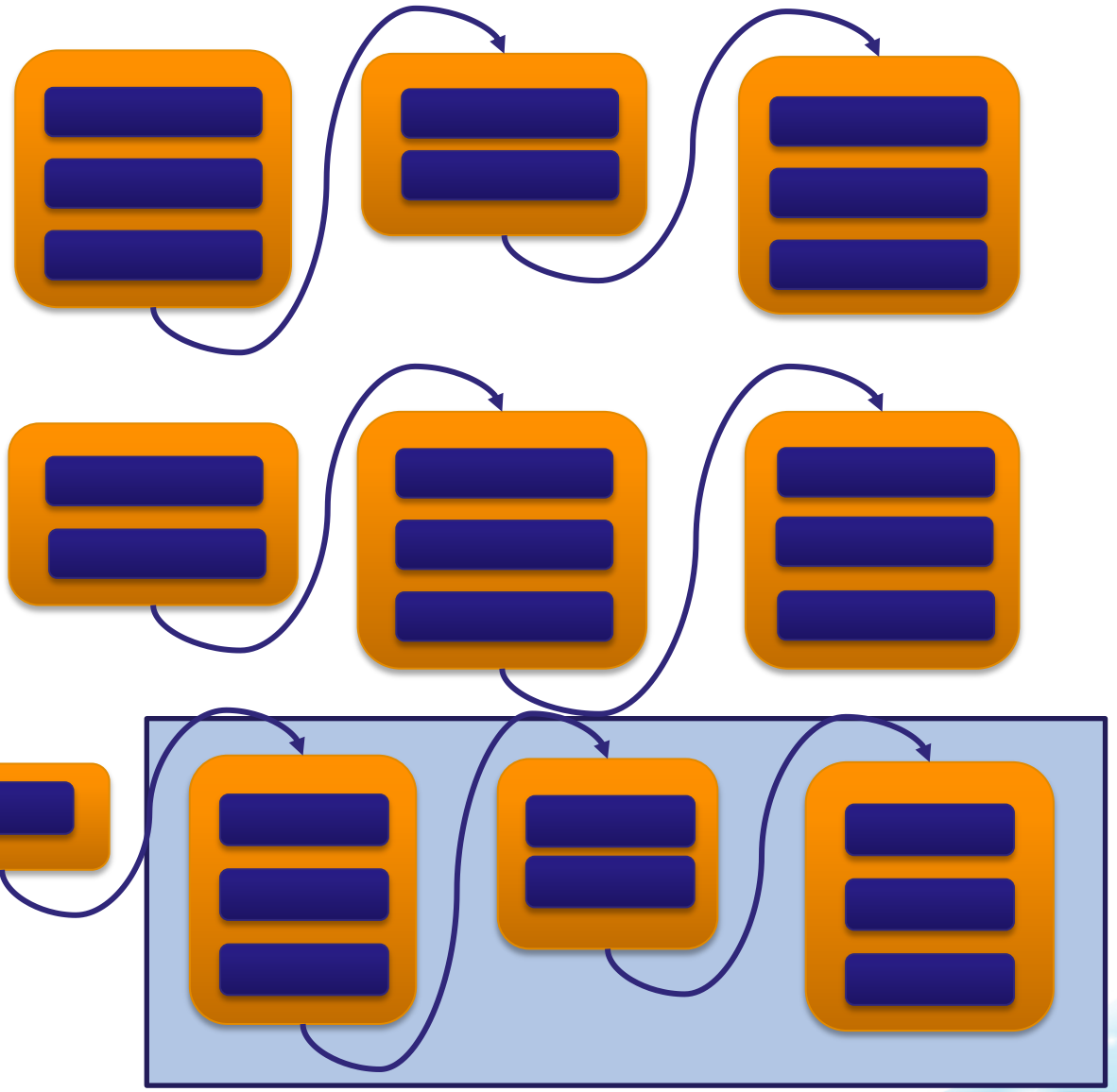
RESSL Sequence Registry

“rwr_seq” →



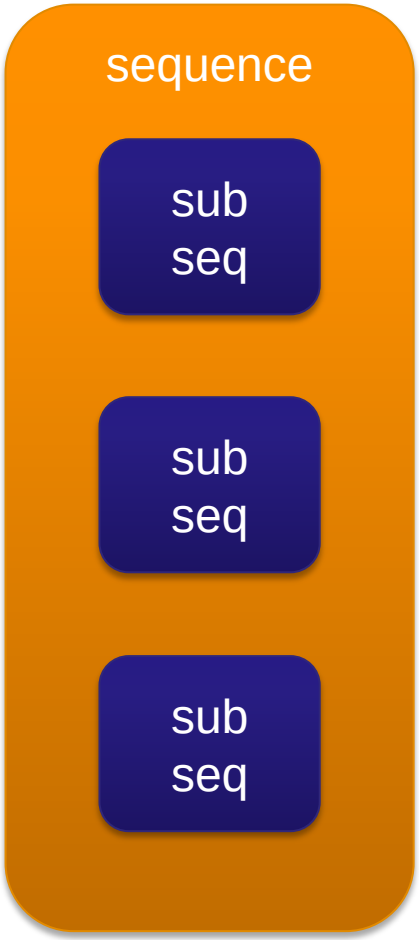
“wrr_seq” →

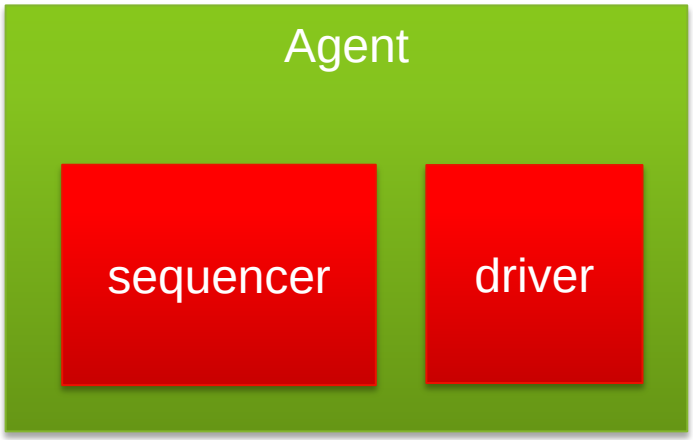
RESSL Sequence Registry



RESSL UVM

[*]>>





RESSL Select

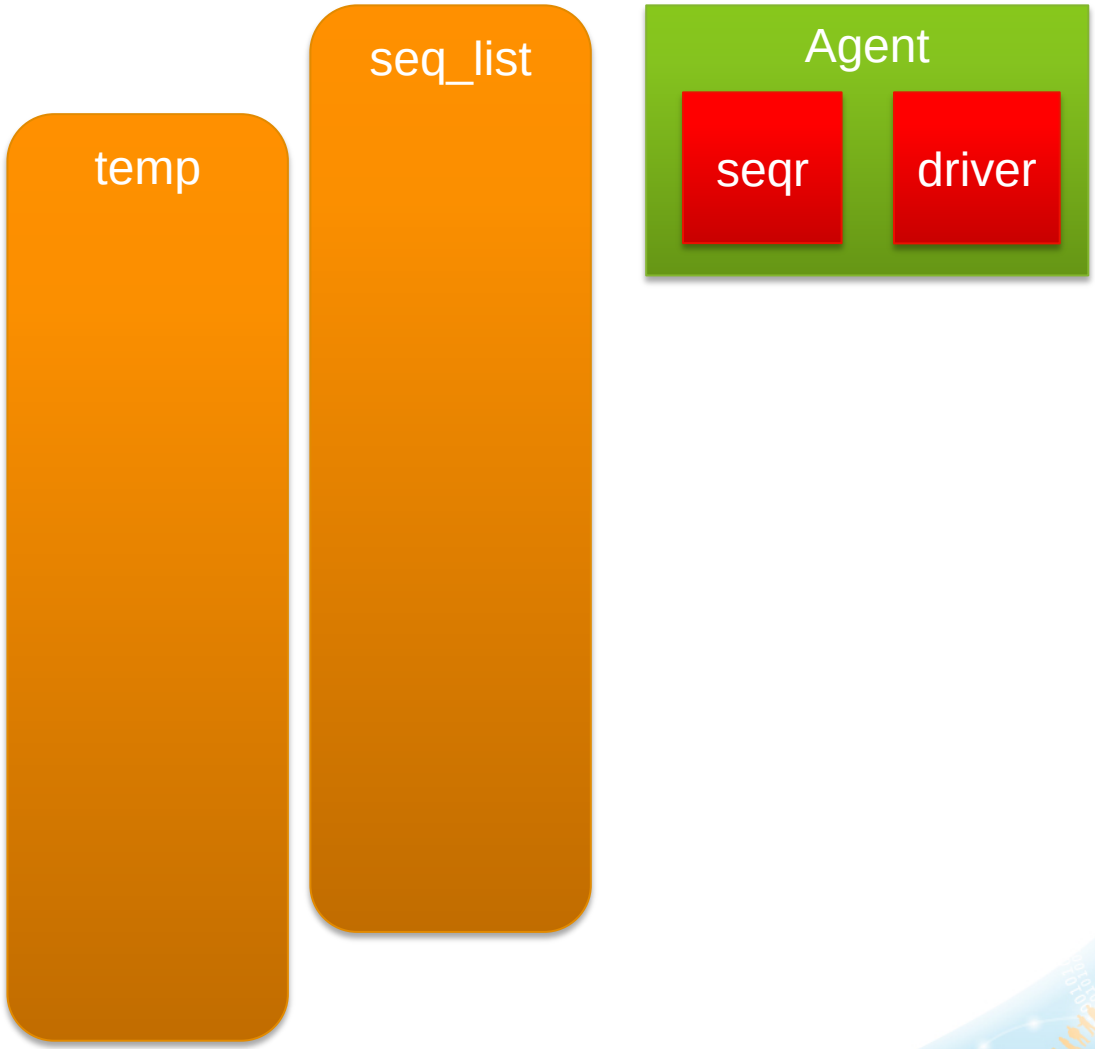
[*] >> **select** seq_list
Selecting seq_list (new)

[seq_list]>>

[seq_list]>> **select** temp
Selecting temp (new)

[temp] >>

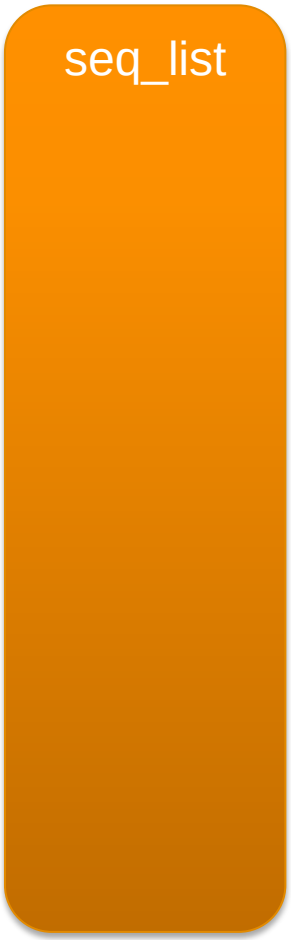
[temp] >> **select** seq_list

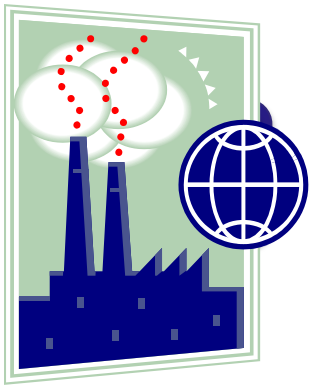


RESSL Create

[list]> **create** SEQ write_byte_seq wbs
seq wbs (type=write_byte_seq) added.

[list]> **create** SEQ read_byte_seq rbs
seq rbs (type=read_byte_seq) added.

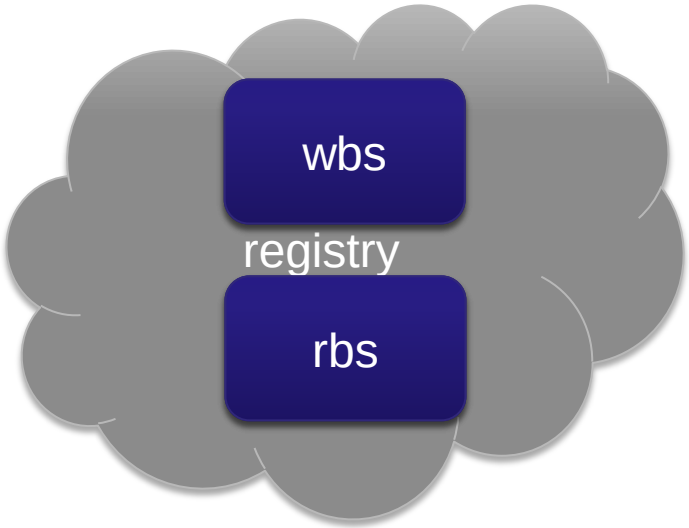


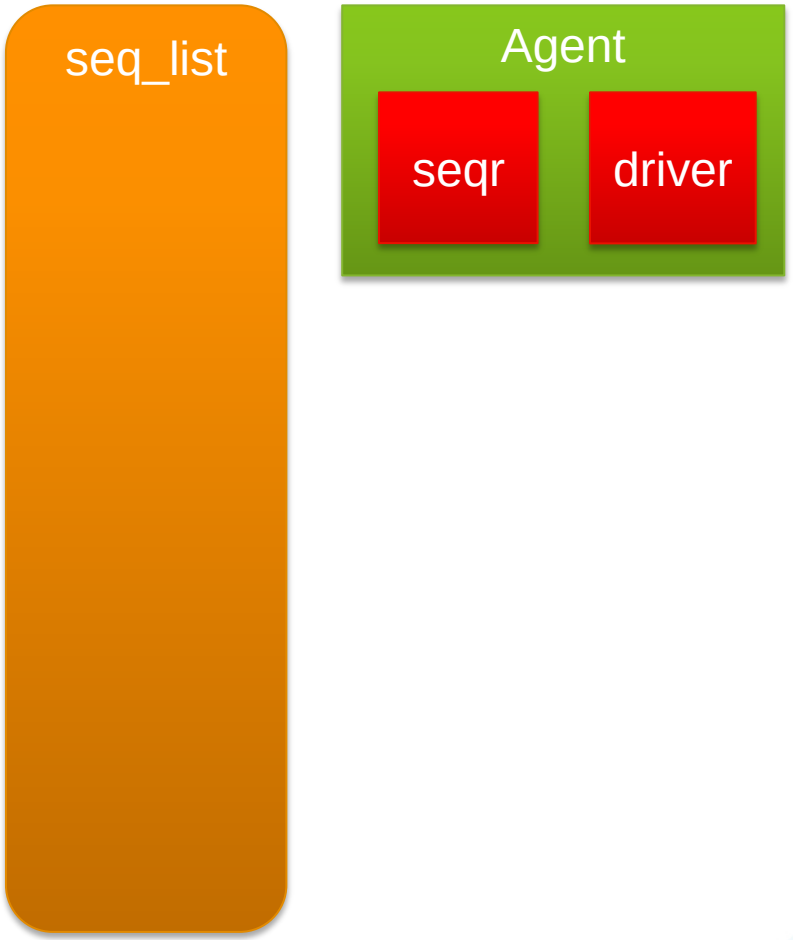


RESSL Add

[seq_list]>> **add** wbs

[seq_list]>> **add** rbs



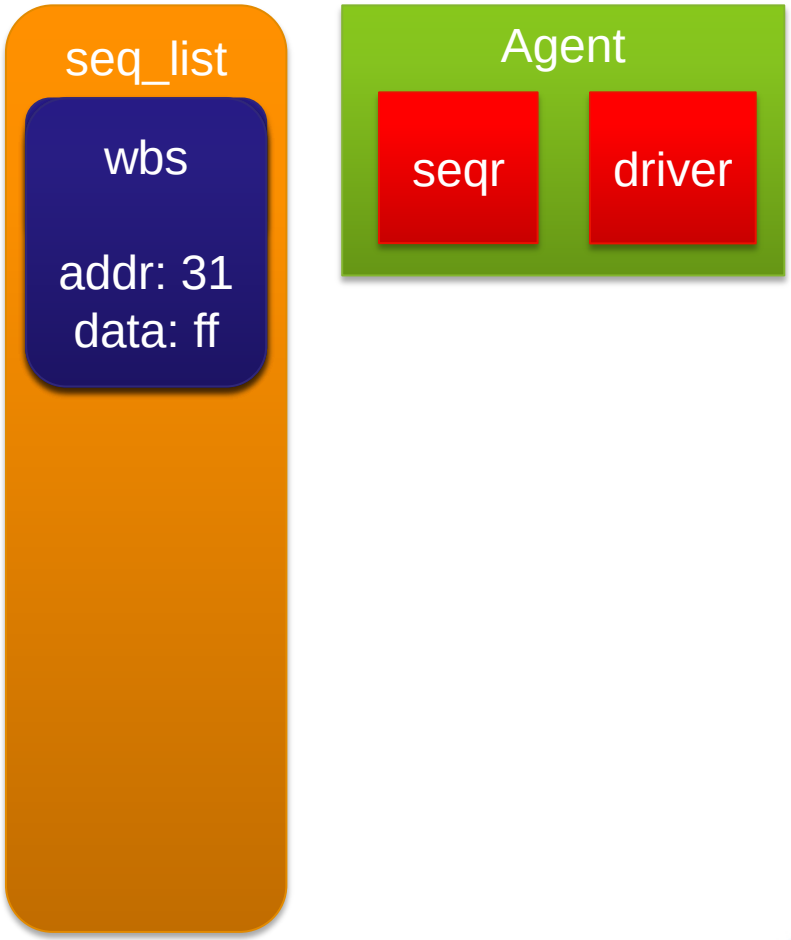


RESSL Set Describe

```
[seq_list]>> describe 0
[0] sequence: wbs (write_byte_seq) …
Fields:
    Field: addr = 00
    Field: data = 0

[seq_list]>> set 0 addr 31
[0] sequence: wbs (write_byte_seq) …
Fields:
    Field: addr = 31
    Field: data = 0

[seq_list]>> set 0 data ff
[0] sequence: wbs (write_byte_seq) …
Fields:
    Field: addr = 31
    Field: data = ff
```

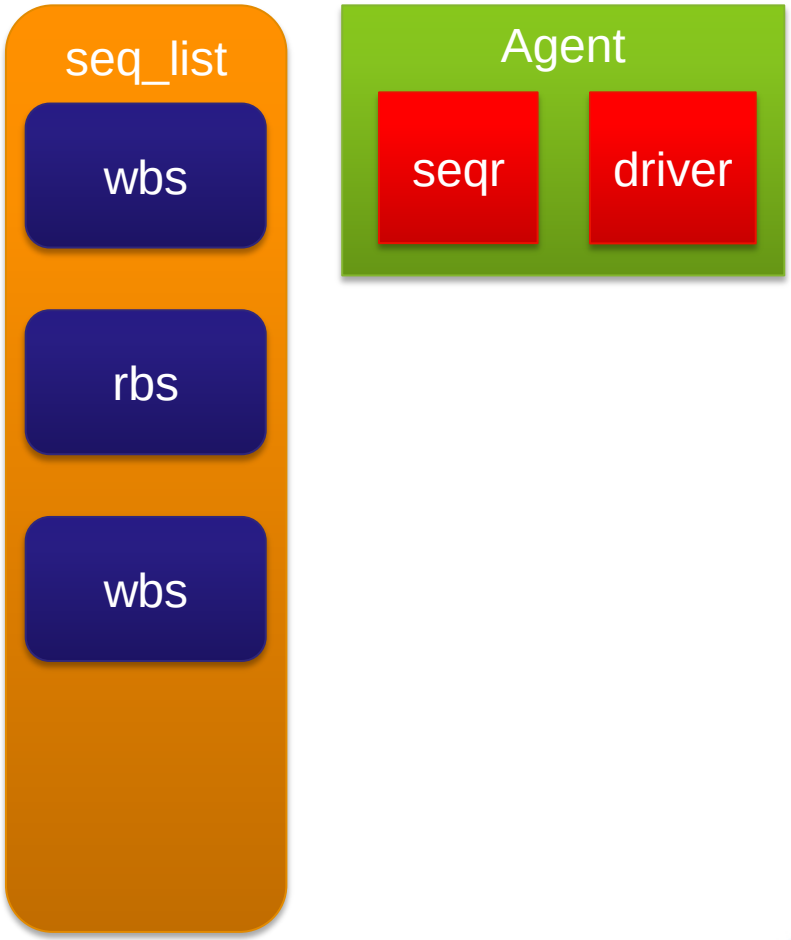


RESSL Start

[seq_list]>> **add** rbs

[seq_list]>> **add** wbs

[seq_list]>> **start**



RESSL Start

[seq_list]>> **add** rbs

[seq_list]>> **add** wbs

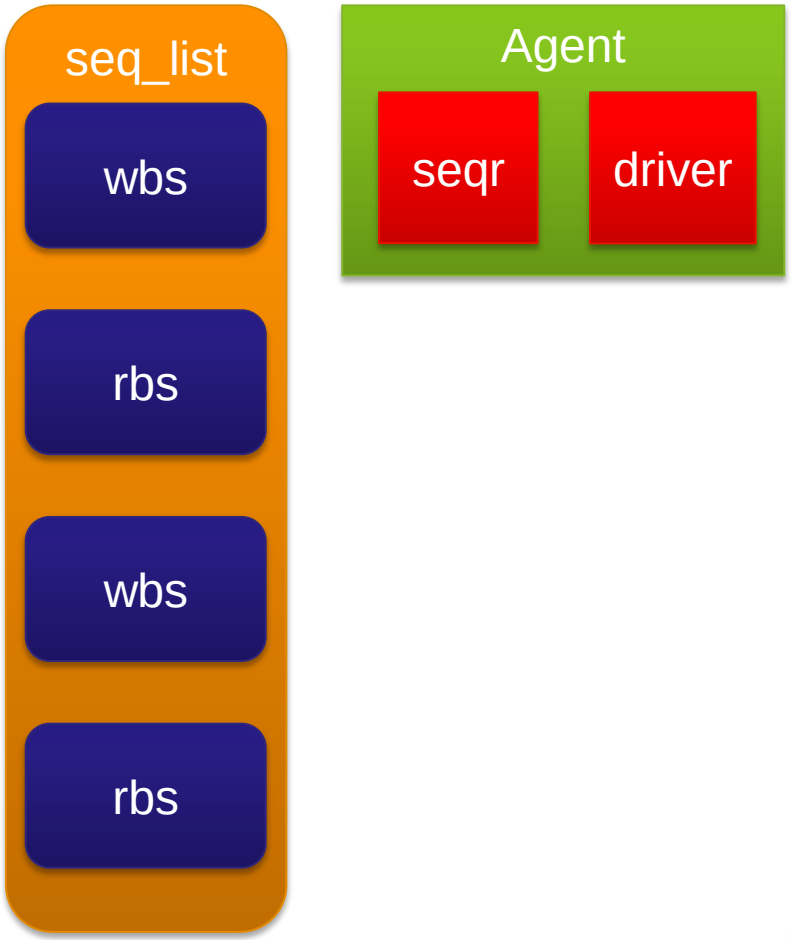
[seq_list]>> **start**

[seq_list]>>

[seq_list]>> **add** rbs

[seq_list]>> **start** 3

[seq_list]>> **quit**



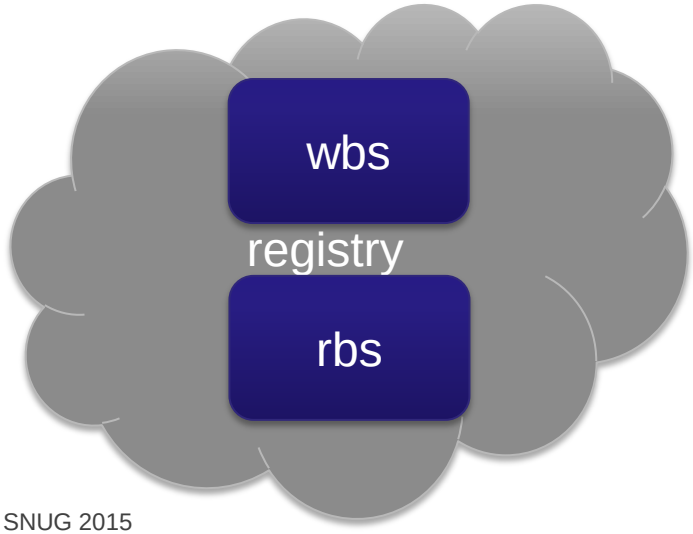
RESSL Sequencers

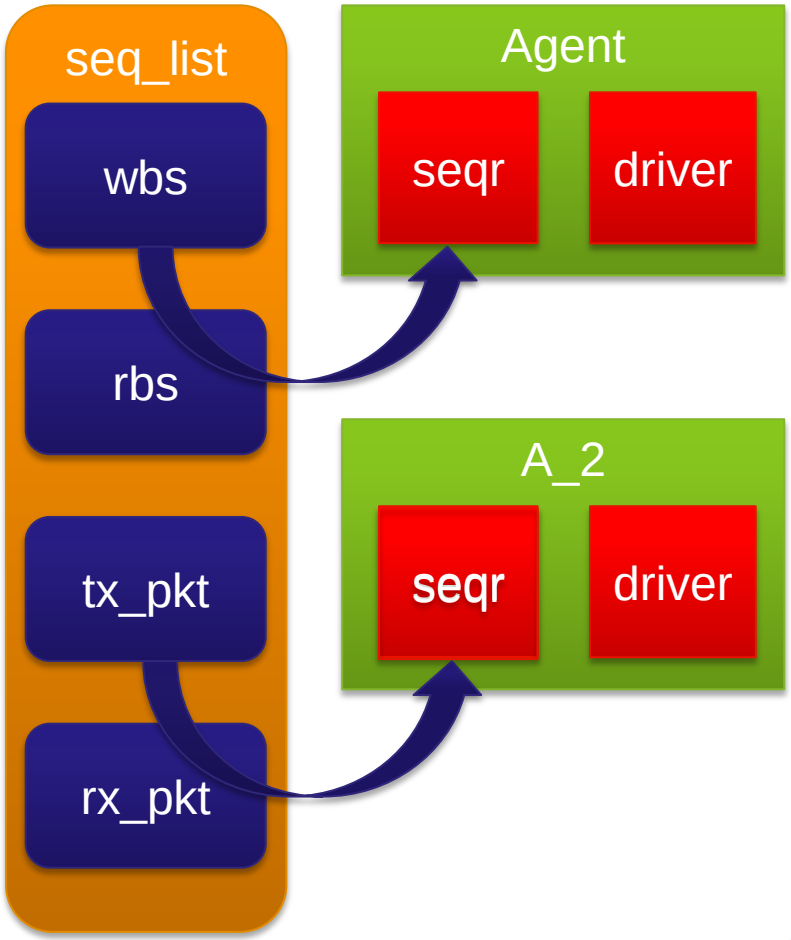
[seq_list]>> **create** SEQR A_2.seqr

[seq_list]>> **attach** A_2.seqr tx_pkt

Overriding attachment on
seq: tx_pkt to
seqr: A_2.seqr

[seq_list]>> **start**



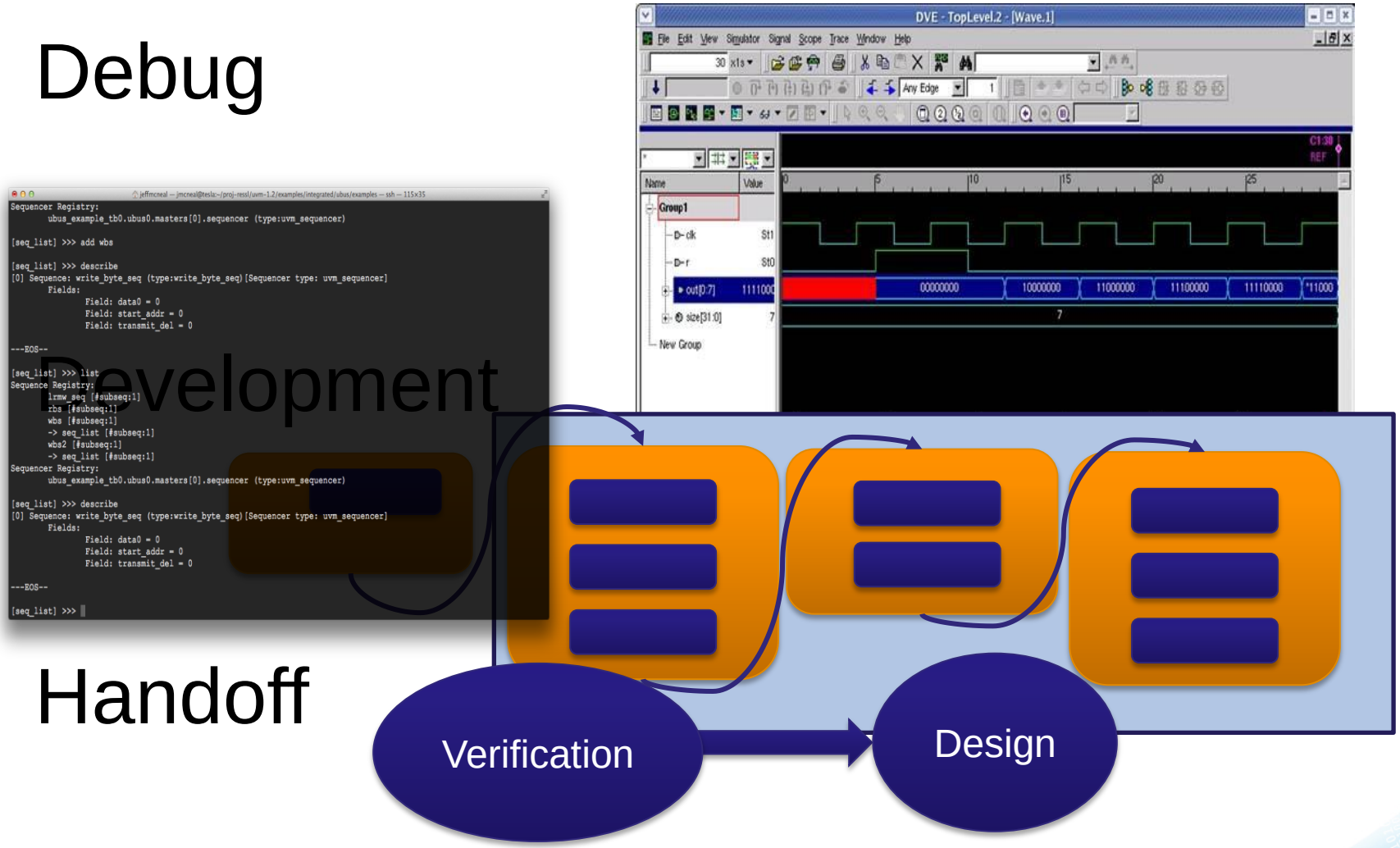


What else can I do?

`add`	Add a sub-sequence from the repository
`attach`	Associate a sequence with a sequencer
`create`	Create a sub-sequence from the factory
`copy`	Copy sub-sequence
`delete`	Remove a sub-sequence
`move`	Move a sub-sequence up or down
`describe`	Print sequence or sub-sequence fields and type
`set`	Change sub-sequence member variables
`randomize`	Randomize sequence or sub-sequence variables
`start`	Run the sequence
`save`	Store the current session to a command file
`load`	Load a command file
`select`	Select or create new sequence
`list`	List containers and sequences

What would I use this for?

- Debug
- Development
- Handoff



I'm sold, how do I use RESSL

1. Download RESSL, svlib and UVM patch (for introspection) from Verilab:
   http://www.verilab.com/resources/papers-and-presentations/
2. Modify scripts to compile RESSL and svlib with your build
3. Declare and instantiate the RESSL object

   ressl ubus_ressl;

   ubus_ressl = ressl::type_id::create("ubus_ressl");
4. Disable main sequence
5. Call RESSL instead

   ubus_ressl.go();

What's the Catch

Introspection: the ability of a program to examine the type or properties of an object at runtime.

Introspection requires modification to UVM

- Changes to UVM 1.2
  - Add two functions to uvm_object (get_field() set_field())
  - uvm_field_macros

Get/Set	Via
integers	`uvm_field_int
enums	`uvm_field_enum
string	`uvm_field_string
real	`uvm_field_real

- Plan to submit changes to Accellera

What's the Catch Continued

RESSL can't add fields

RESSL can't add constraints

No performance impact, but stalls waiting for the user

Future Developments

- Tcl Command Line Interpreter
- Python Command Line Interpreter
- Export sequences to SystemVerilog code
- Others… (any suggestions)

Conclusion

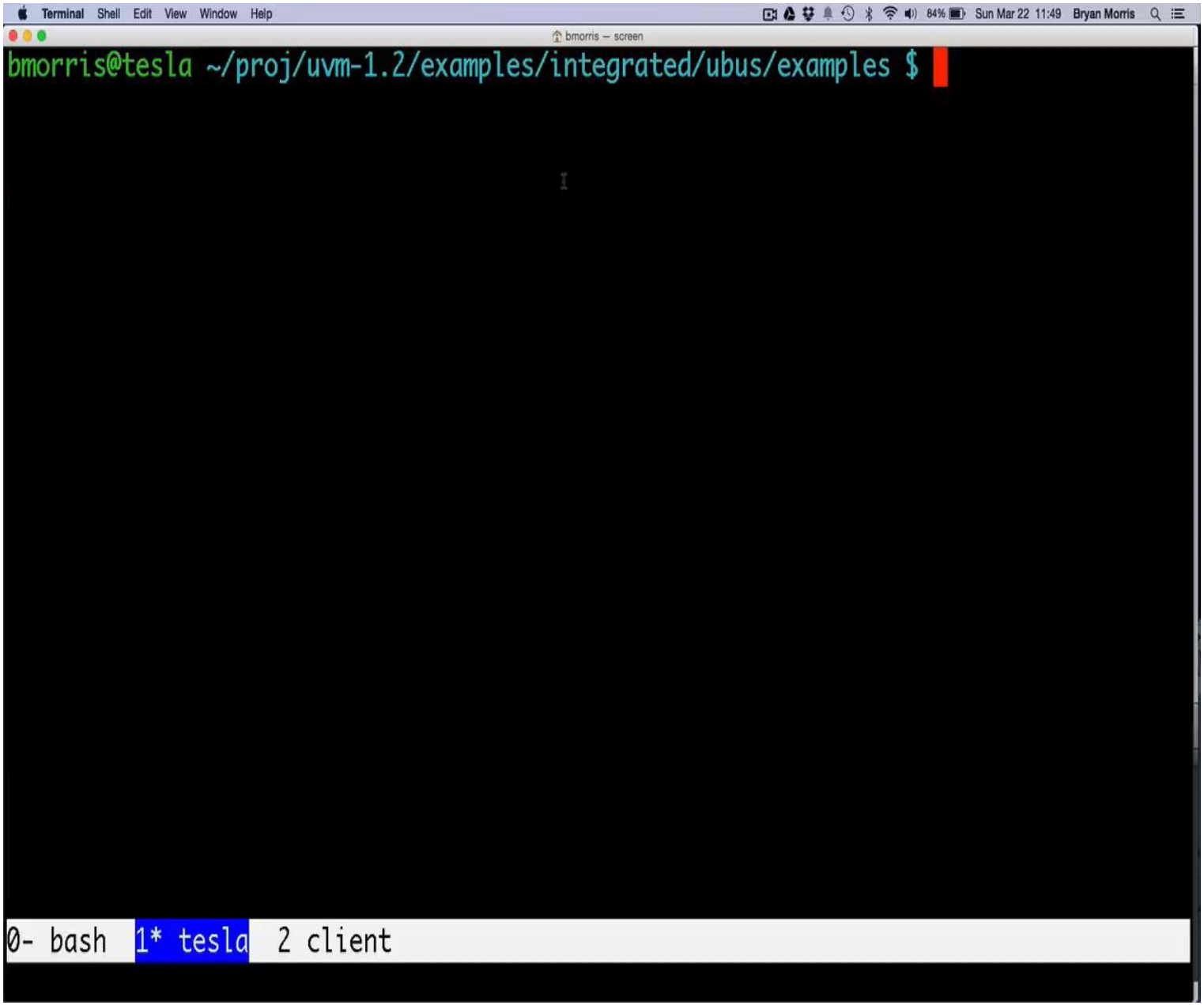
- RESSL provides interactive UVM development
  - Quickly explore “what if” scenarios
  - Debug RTL with minimal UVM knowledge
  - Test scenario development by non UVM users
  - Short circuit the compile-elaborate-run loop

Thank You

Questions

- jeff.mcneal@verilab.com
- bryan.morris@verilab.com

Backup Slides


Terminal Shell Edit View Window Help
84% Sun Mar 22 11:49 Bryan Morris
bmorris — screen
bmorris@tesla ~/proj/uvm-1.2/examples/integrated/ubus/examples $
0- bash 1* tesla 2 client

[seq_list] >>> help

--- RESSL Help ---

add <sequence name> [repeat_count]

attach <seqr> <to_seq> [ALL | <seq_index>]

create <SEQ | SEQR> <type> <name>

describe <sequence name>

help [verb]

list

load <filename>

move <from_index> <to_index>

quit = exits RESSL and continues the simulation.

randomize <seq_index>

save <filename>

select <seq_path>

set <seq_index> <field> <value>

start [repeat_count] = Execute the currently selected sequence [repeat_count] times.

```
class registry#(type KEY = string, type T = seq_info) extends uvm_object;
  local T registry_[KEY][$];
  local int unsigned current_q_index_[KEY];

  // keys: really used for iterating thru the registry.
  `uvm_object_param_utils(registry)

  // Function: add
  //   Adds the supplied item_type into the registry, keyed by key.
  //   If key does not exist, a new queue of Ts is created and the
  //   item_type added to the front.
  //   If key already exists in the registry, the item_type is added to
  //   the back of the queue.
  virtual function void add(
    input KEY key,
    input T   item
  );
    this.reserve(key);
    this.registry_[key].push_back(item);
  endfunction : add
```