Multi-Stream Scenarios
Enhancing Stimulus Generation in the VMM

Sumit Dhamanwala – Verilab
Jason Sprott – Verilab
(jason.sprott@verilab.com)
Today’s Agenda

• Existing Solution for Stimulus Generation

• Challenges

• Channel Grab/Ungrab Additions

• Multi-stream Scenario Additions
Scenario Generation Today

User defined channels and scenario generators:

```
vmm_channel(Packet)
vmm_scenario_gen(Packet)
```
Scenario Generation Today

Generators all run independently
Resource Sharing Today

Limited control over flow when generators share resources
What If We Want More Control?

We might want to orchestrate scenario generation ... Enter - the Multi-Stream Scenario
The Challenges

• Backward Compatibility
• Resource Sharing
  – Grabbing channels
  – Making that hierarchy aware
• Multi-stream Scenarios (MSS)
  – Composition: other MSS; scenarios; items
  – Database of accessible resources
VMM Standard Library Modification

- **vmm_ms_scenario** *(added)*
  - Methods to establish parent/child relationship
- **vmm_ms_scenario_gen** *(added)*
  - vmm_channel registry
  - vmm_ms_scenario registry
  - vmm_ms_scenario_gen registry
- **vmm_ms_scenario_election** *(added)*
  - Selection scheme
- **vmm_scenario** *(modified)*
  - Adding parent/child relationship for channel grab
- **vmm_channel** *(modified)*
  - Grab/Ungrab
VMM Channel Grab/Ungrab

Grab owner Queue

- MSS M1
- Scenario S1
- Scenario S2
- MSS M2

M1 grabs channel

GRANTED

CHANNEL
VMM Channel Grab/Ungrab

Grab owner Queue

M1

Scenario S1

Scenario S2

M2

MSS

A grab or put of any sort blocked since the channel is grabbed.
VMM Channel Grab/Ungrab

MSS
M1

Scenario S1

Scenario S2

MSS
M2

Grab owner Queue

Grab queue updated to show S1 is now owner

GRANTED

CHANNEL

S1 grabs an already grabbed channel. Since it belongs to parent M1 it’s allowed

GRANTED

Scenario S1

Grab owner Queue

VMM Channel Grab/Ungrab
VMM Channel Grab/Ungrab

Grab owner Queue

Scenario S1

Scenario S2

MSS
M1

MSS
M2

S2 grabs are blocked because the grab owner is S1, who is not S2’s parent
VMM Channel Grab/Ungrab

- MSS M1
  - Scenario S1
  - Scenario S2
- MSS M2

Grab owner Queue

Grab queue updated to show M1 is now owner

S1 Ungrabs channel

CHANNEL
VMM Channel Grab/Ungrab

M1  S2  Grab owner Queue

MSS
M1

Scenario S1

Scenario S2

MSS
M2

 CHANNEL

GRANTED

S2 grabs allowed this time, since M1 is the parent
VMM Channel Grab/Ungrab

M1 tried to ungrab but a VMM error is issued because it is not the current owner.
VMM Channel Grab/Ungrab

Grab owner Queue

M1

Scenario S1

Scenario S2

MSS M1

MSS M2

CHANNEL

S2 ungrabs channel releasing ownership back to M1

GRANTED
VMM Channel Grab/Ungrab

MSS M1

Scenario S1

Scenario S2

MSS M2

Grab owner Queue

GRANTED

CHANNEL

Now M1 can ungrab since it is the owner
VMM Channel Grab/UnGrab

M2

Grab owner Queue

MSS
M1

Scenario S1

Scenario S2

MSS
M2

M2 grab granted

GRANTED

CHANNEL
## Grab Features In vmm_channel

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRABBED/UNGRABBED events</td>
<td>Notify events</td>
</tr>
<tr>
<td>function void reset_grabbers()</td>
<td>Removes all grabs</td>
</tr>
<tr>
<td>function bit try_grab(vmm_scenario grabber)</td>
<td>Attempts a grab, returning status (non blocking)</td>
</tr>
<tr>
<td>task grab(vmm_scenario grabber)</td>
<td>Tries to grab a channel (blocking)</td>
</tr>
<tr>
<td>function void ungrab(vmm_scenario grabber)</td>
<td>Tries to ungrab a channel. Returns a VMM error if illegal (non blocking)</td>
</tr>
<tr>
<td>function bit is_grabbed()</td>
<td>Returns 1 is the channel is grabbed</td>
</tr>
</tbody>
</table>
We have control over resources, execution of single or multi-stream scenarios and sending individual items.
Do MSS Replace Scenarios?

- MSS brings multi-stream control
- MSS can definitely replace scenarios, but ...
- They don’t have to – both play together no problem
- MSS Generator doesn’t have a registry for single scenarios
- With MSS we don’t have some features, e.g. atomic and default scenarios
## MSS vs Single Scenario

<table>
<thead>
<tr>
<th>Single Scenario</th>
<th>Multi-stream Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communicates with the channel via apply() where channel is passed as an argument</td>
<td>Looks up handle for channel in its channel registry.</td>
</tr>
<tr>
<td>Can only call scenarios that are capable of using the same channel.</td>
<td>Can call other scenarios by storing their channel in the channel registry.</td>
</tr>
<tr>
<td>Cannot call multi-stream scenarios</td>
<td>Can call other multi-stream scenarios in accessible multi-stream scenario generators.</td>
</tr>
</tbody>
</table>
// register any channels being used
dsl::msg1.register_channel("CONFIG", config_ch);
...
dsl::msg1.register_channel("INT", interrupt_ch);
MSS Generator Registry

// register MSS gens connected
msg1.register_ms_scenario_gen("MSG2", msg2);
msg1.register_ms_scenario_gen("MSG3", msg3);
// register MSS being used
msg1.register_ms_scenario("CONFIG", ms_typical);
msg1.register_ms_scenario("ATMONLY", ms_atmonly);
msg1.register_ms_scenario("PACKETS", ms_packets);
MSS Execution

MSS is fetched from registry based on election mechanism and executed

ms_typical extends vmm_ms_scenario;
...
  virtual task execute(ref int n_insts);
    // MSS actions go here
  endtask
endclass

config_sc
  apply()
  Scenario

packet_sc
  apply()
  Scenario

atmcell_sc
  apply()
  Scenario

MSS Generator

msg1

MSS, CH, MSG
Walkthrough Of MSS execute()

// Extend from the base class
class ms_typical extends vmm_ms_scenario;

... 

virtual task execute(ref int n_insts);

    int n=0;
    vmm_channel config_chan; // for single scenario gen
    vmm_channel packet_chan; // for single scenario gen
    vmm_channel atm_cell_chan; // for single scenario gen
    vmm_channel interrupt_chan; // for sending items directly
    packet_scenario packet_sc = new(this);
    atm_cell_scenario atm_cell_sc = new(this);
    config_scenario config_sc = new(this);
    interrupt_ctl my_interrupts = new(); // single item

Continued ...
Walkthrough Of MSS execute()

```java
// Get the channel handles from registry
config_chan = this.get_channel("CONFIG");
packet_chan = this.get_channel("PACKET");
atm_cellChan = this.get_channel("ATM_CELL");
imruptChan = this.get_channel("INT");
// grab the channels we want exclusive access to
configChan.grab(this);
packetChan.grab(this);
atmCellChan.grab(this);
// Do the configuration scenario first
configSc.randomize with { ... };
configSc.apply(configChan, n);
n_insts++;
```

Continued ...
Walkthrough Of MSS execute()

// Run other scenarios and send items in parallel
fork
    begin
        packet_sc1.randomize();
        packet_sc1.apply(packet_chan, n);
        n_insts++;
    end
begin
    atm_cell_sc.randomize();
    atm_cell_sc.apply(atm_cell_chan, n);
    n_insts++;
end
Continued ...
begin
    my_interrrupts.randomize() with { ... };
    interrupt_ch.put(my_interrupts);
    n_insts++;
end

join

// Release the channels
config_ch.ungrab(this);
packet_ch.ungrab(this);
atm_cell_ch.ungrab(this);
endtask: execute
...
endclass: ms_normal
Summary

• Added functionality to enhance resource locking control
• Added ability to control multiple streams of stimulus
• Augments current scenario generation
• Backward compatible
Jason Sprott (V.P. Consulting) Verilab
jason.sprott@verilab.com
www.verilab.com