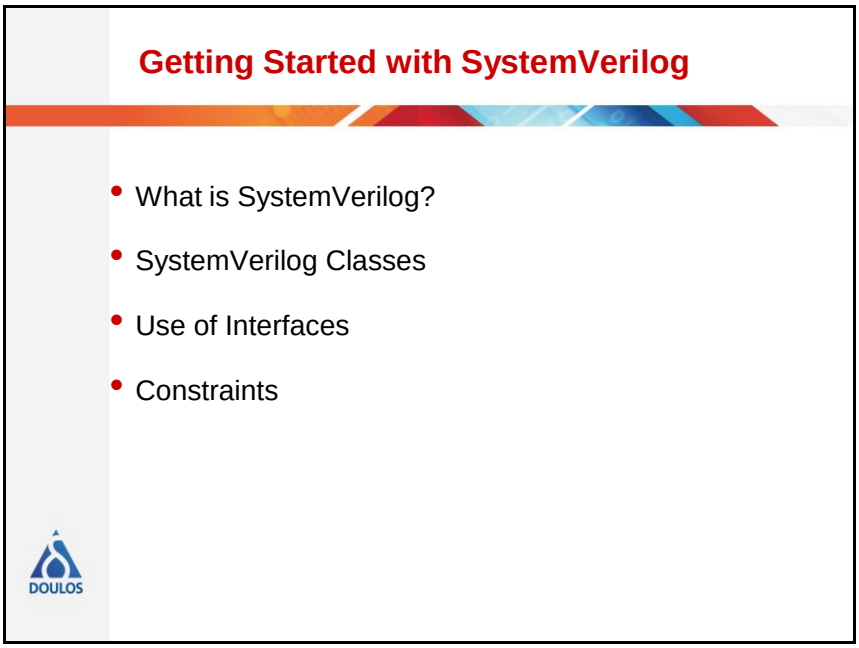
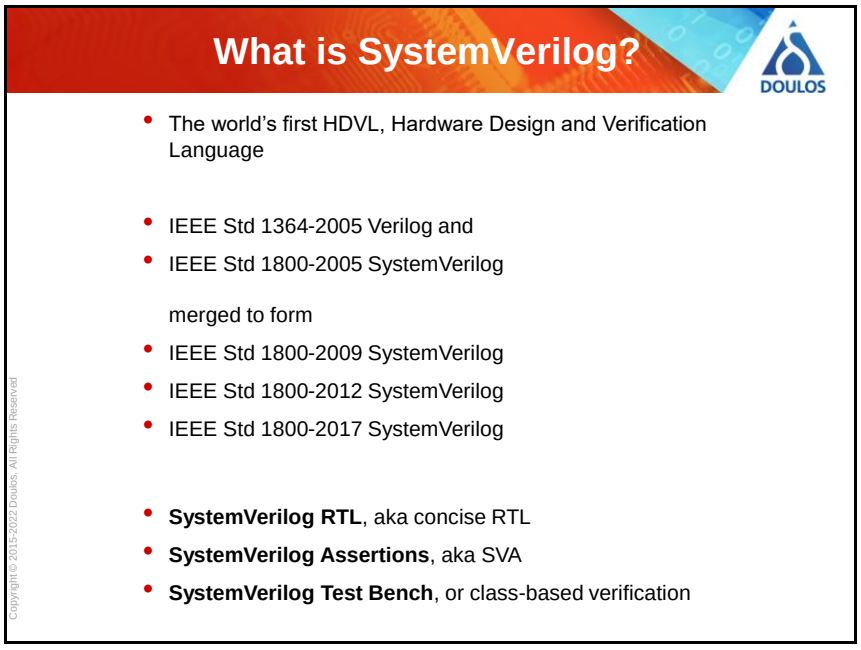
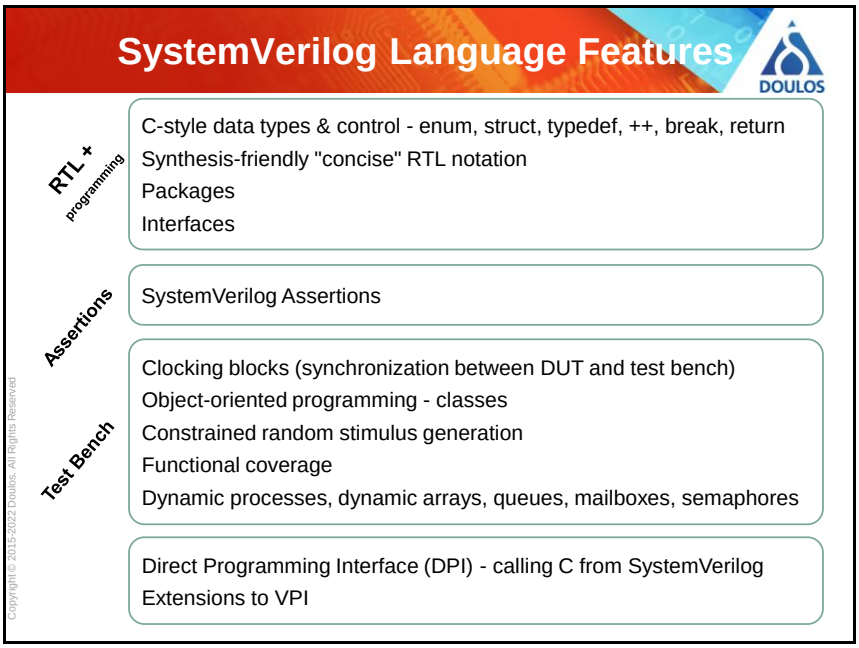
Notes



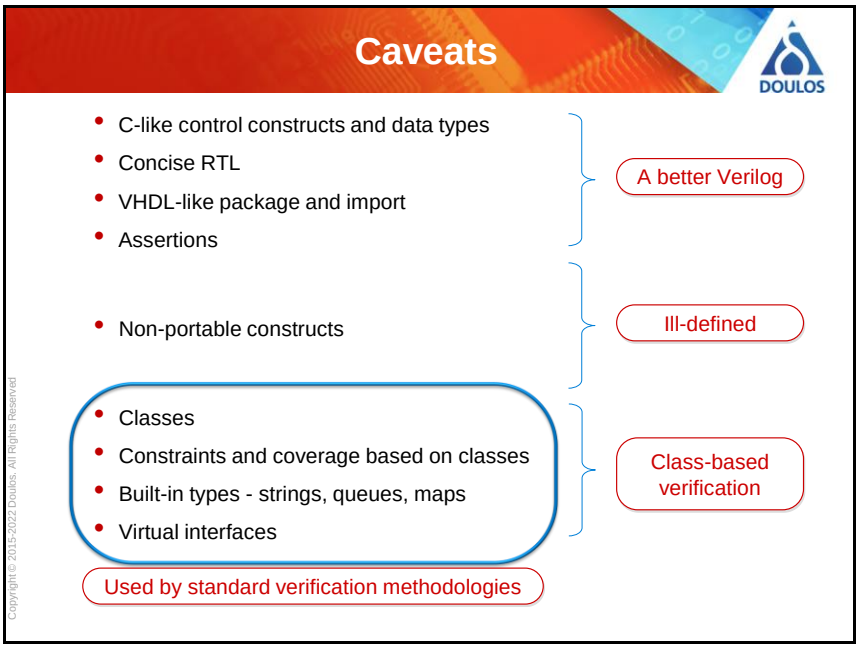
Notes



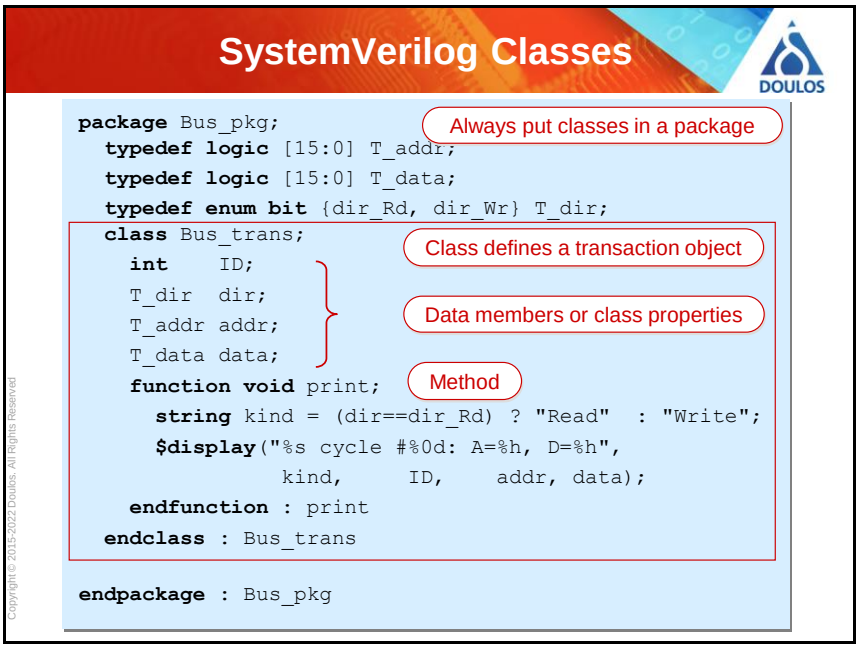
Notes



Notes

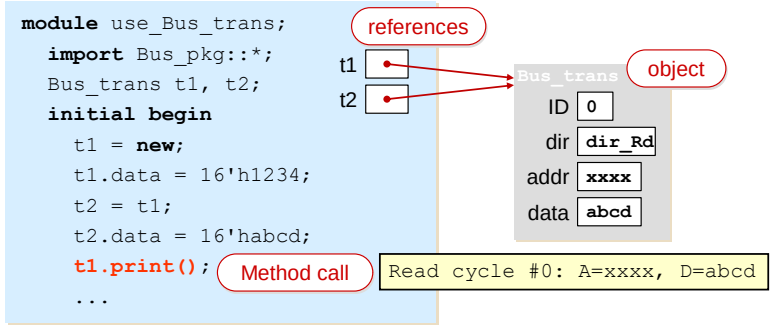


Notes



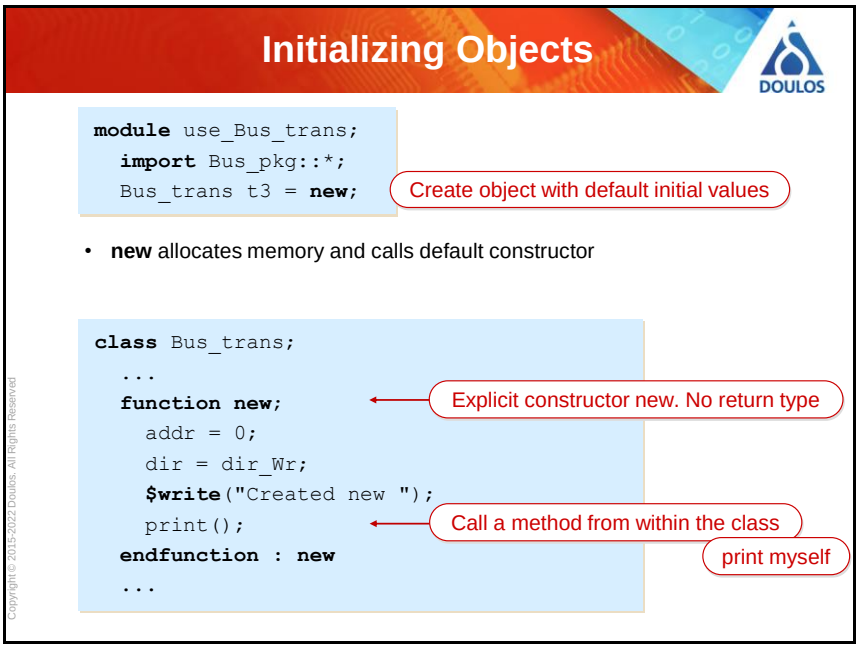
Notes

Object = Instance of Class



- Variables of class type store *references* (handles) to real objects
  - Initialised to `null` (reference to no object)
- Create objects using `new` - data members get their usual default values
- Access data members, and call methods, in existing objects using dot notation
  - Methods act on the object through which they were called

Notes



Notes

Constructor Arguments

- Like any function in SystemVerilog, constructors may have arguments

```
function new (T_dir direction);
  addr = 0;
  dir = direction;
  $write("Created new "); print();
endfunction : new
```

```
Bus_trans t4 = new;
Bus_trans t4 = new(dir_Rd);
```

ERROR

Read

- Arguments may have default values (no overloading, though)

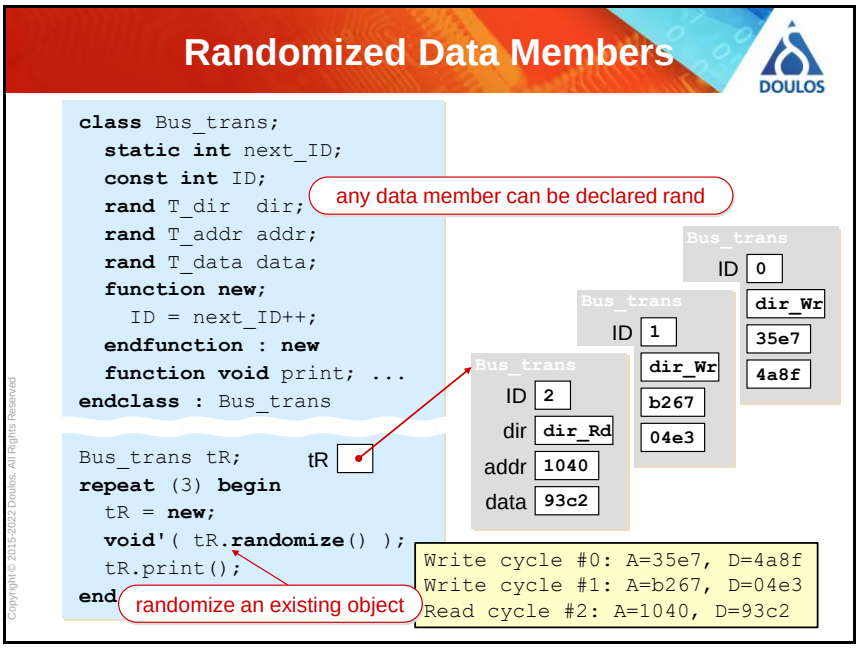
```
function new (T_dir direction = dir_Rd); ...
```

```
Bus_trans t5 = new;
Bus_trans t6 = new(dir_Wr);
```

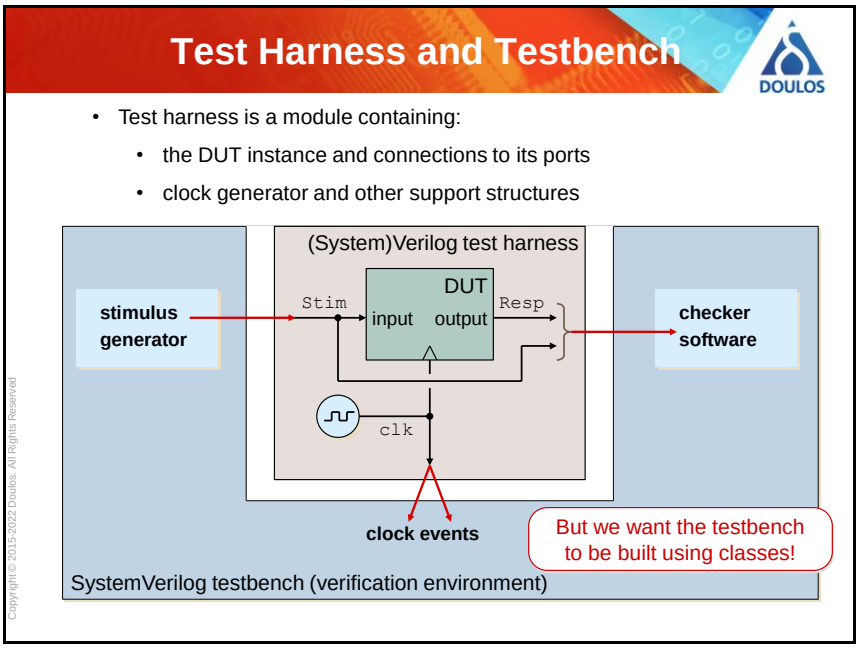
Read

Write

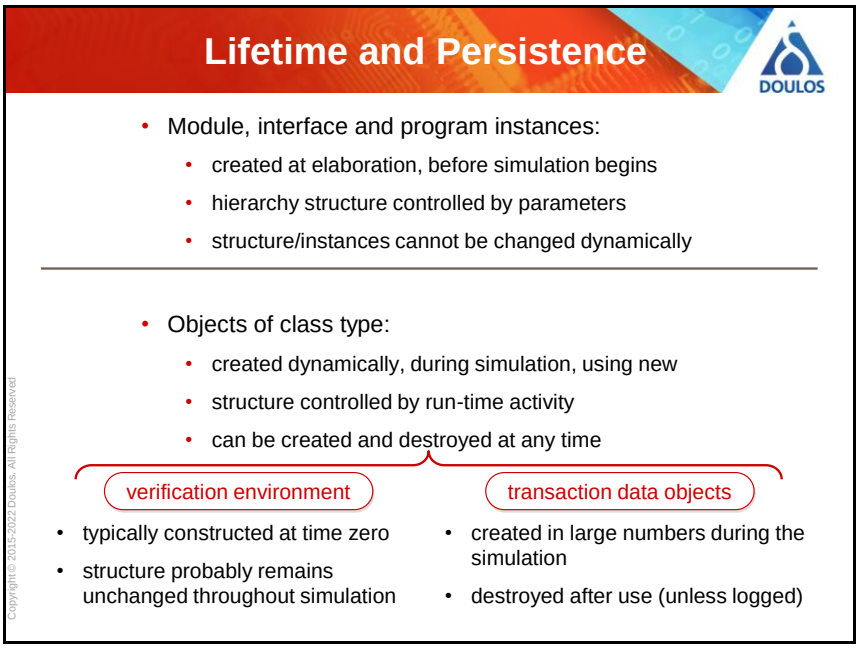
Notes



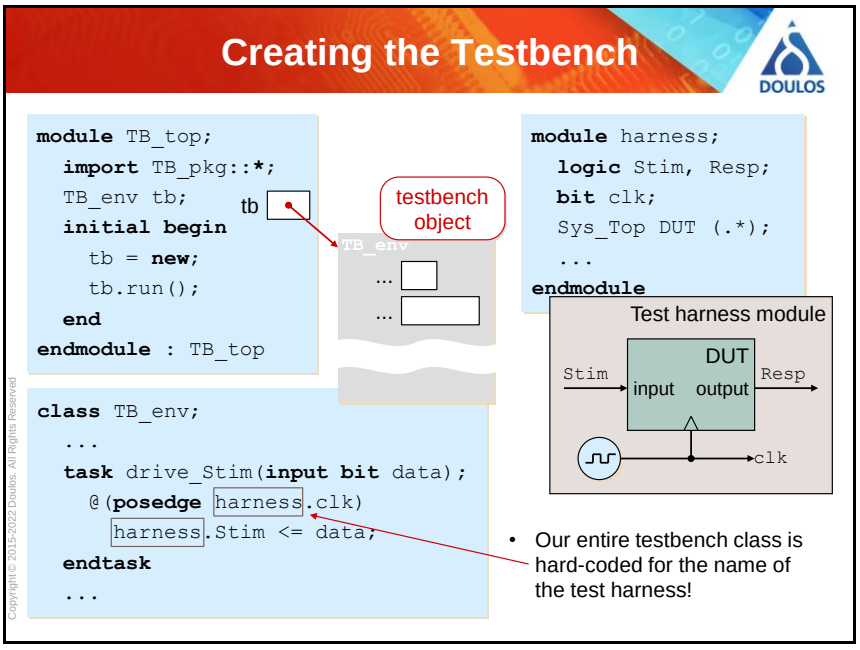
Notes


Test Harness and Testbench
DOULOS
Test harness is a module containing:
the DUT instance and connections to its ports
clock generator and other support structures
(System)Verilog test harness
stimulus generator
Stim
DUT
input
output
Resp
checker software
clk
clock events
But we want the testbench to be built using classes!
SystemVerilog testbench (verification environment)
Copyright © 2015-2022 Doulos. All Rights Reserved

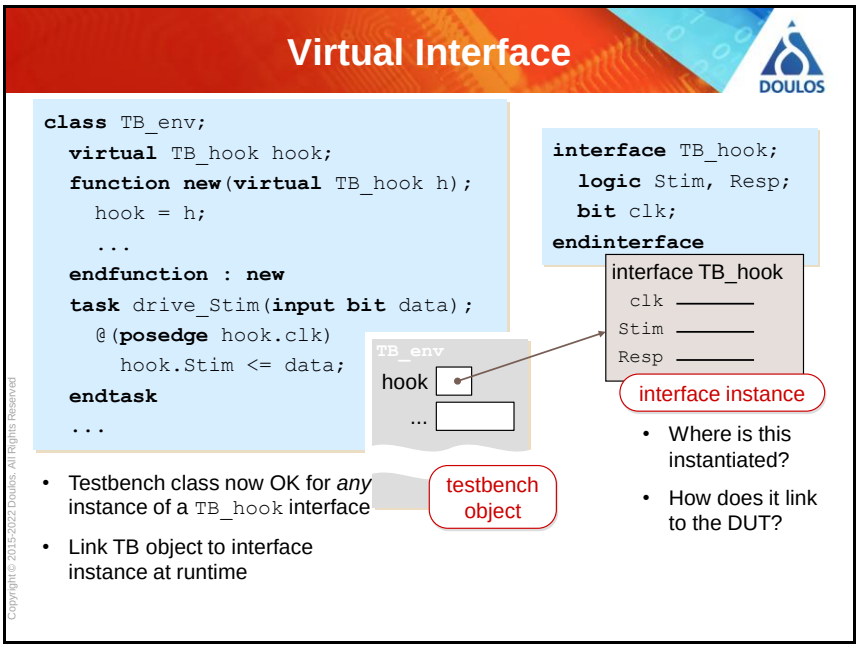
Notes



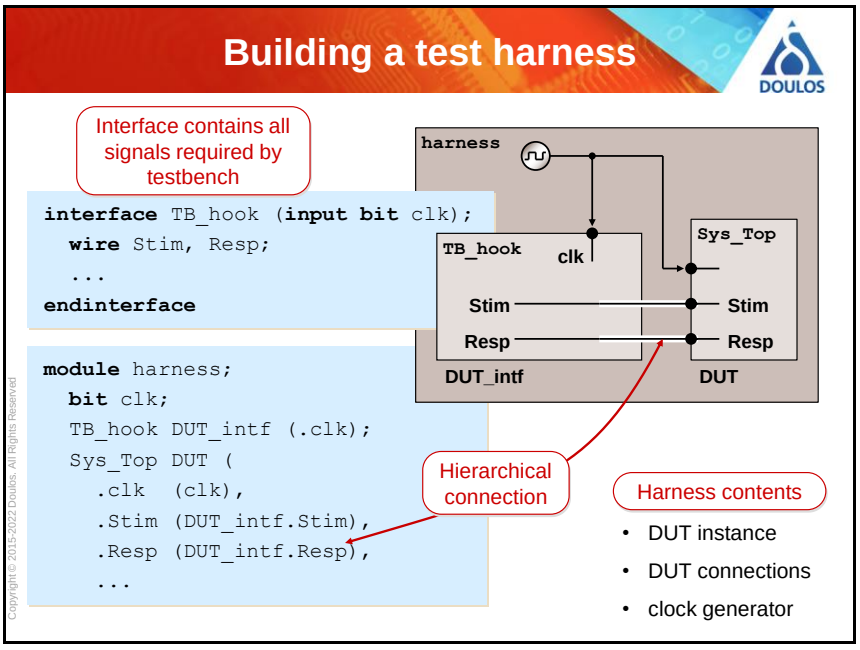
Notes


Creating the Testbench
DOULOS
module TB_top;
import TB_pkg::*;
TB_env tb;
initial begin
tb = new;
tb.run();
end
endmodule : TB_top
tb
testbench object
TB_env
...
...
module harness;
logic Stim, Resp;
bit clk;
Sys_Top DUT (.*);
...
endmodule
Test harness module
DUT
Stim
input
output
Resp
clk
class TB_env;
...
task drive_Stim(input bit data);
@(posedge harness.clk)
harness.Stim <= data;
endtask
...
• Our entire testbench class is hard-coded for the name of the test harness!
Copyright © 2015-2022 Doulos. All Rights Reserved

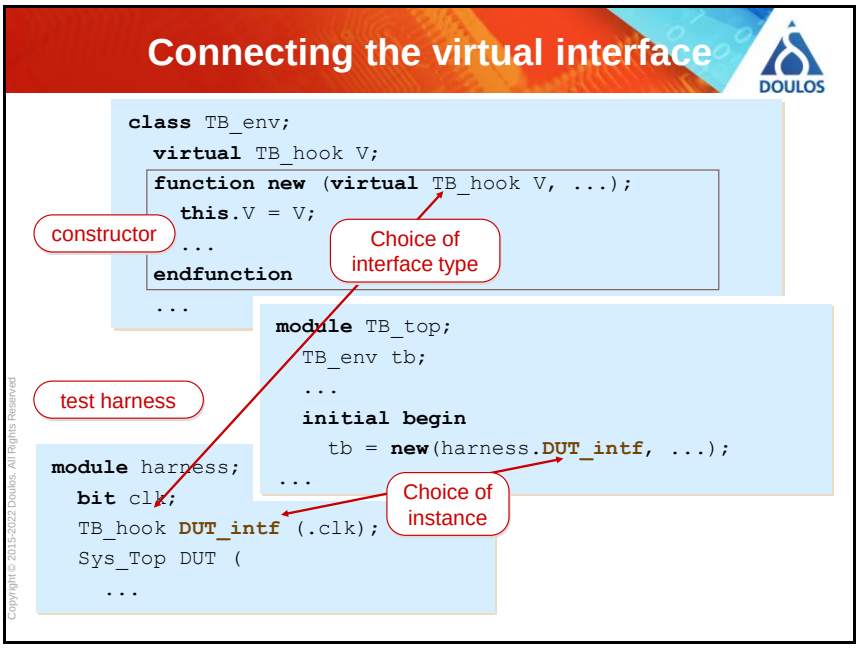
Notes



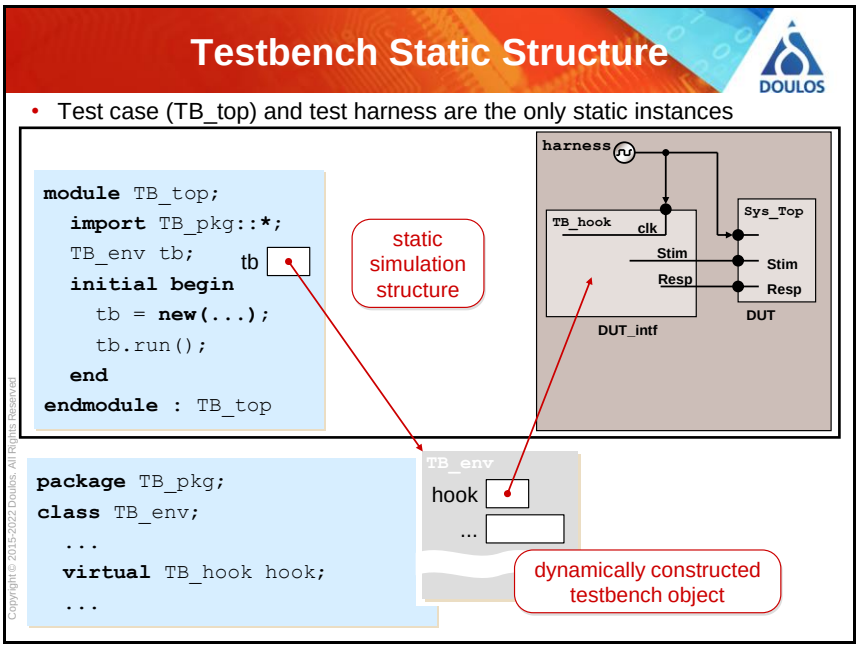
Notes


Building a test harness
DOULOS
Interface contains all signals required by testbench
interface TB_hook (input bit clk);
wire Stim, Resp;
...
endinterface
module harness;
bit clk;
TB_hook DUT_intf (.clk);
Sys_Top DUT (
.clk (clk),
.Stim (DUT_intf.Stim),
.Resp (DUT_intf.Resp),
...
harness
TB_hook
clk
Stim
Resp
DUT_intf
Sys_Top
Stim
Resp
DUT
Hierarchical connection
Harness contents
• DUT instance
• DUT connections
• clock generator
Copyright © 2015-2022 Doulos. All Rights Reserved

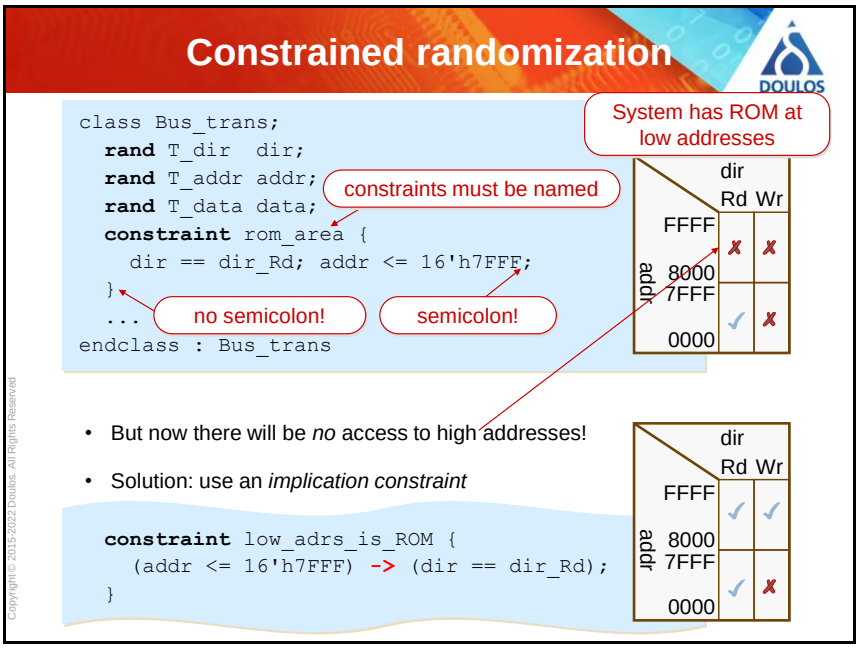
Notes



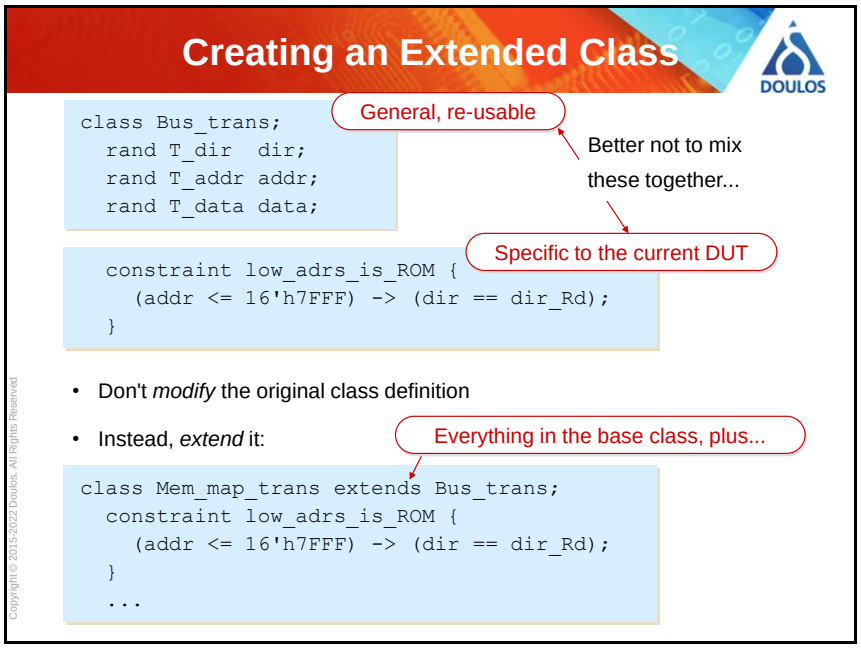
Notes


Testbench Static Structure
DOULOS
• Test case (TB_top) and test harness are the only static instances
module TB_top;
import TB_pkg::*;
TB_env tb;
initial begin
tb = new(...);
tb.run();
end
endmodule : TB_top
tb
static simulation structure
harness
TB_hook
clk
Stim
Resp
Sys_Top
Stim
Resp
DUT_intf
DUT
package TB_pkg;
class TB_env;
...
virtual TB_hook hook;
...
TB_env
hook
...
dynamically constructed testbench object
Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



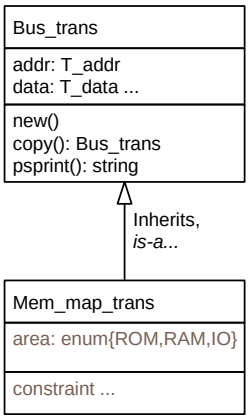
Notes


Creating an Extended Class
DOULOS
class Bus_trans;
rand T_dir dir;
rand T_addr addr;
rand T_data data;
General, re-usable
Better not to mix these together...
Specific to the current DUT
constraint low_adrs_is_ROM {
(addr <= 16'h7FFF) -> (dir == dir_Rd);
}
• Don't modify the original class definition
• Instead, extend it:
Everything in the base class, plus...
class Mem_map_trans extends Bus_trans;
constraint low_adrs_is_ROM {
(addr <= 16'h7FFF) -> (dir == dir_Rd);
}
...
Copyright © 2015-2022 Doulos. All Rights Reserved

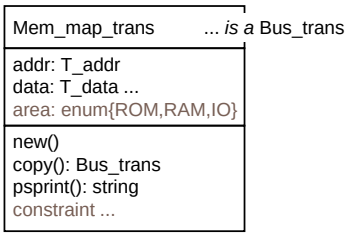
Notes

Inheriting Class Members

- What you write

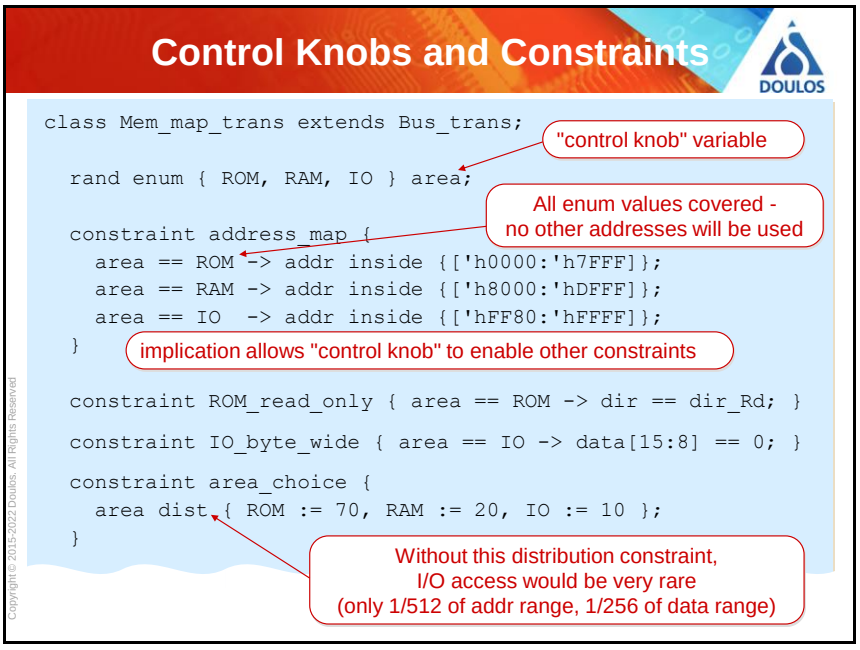


- What you get

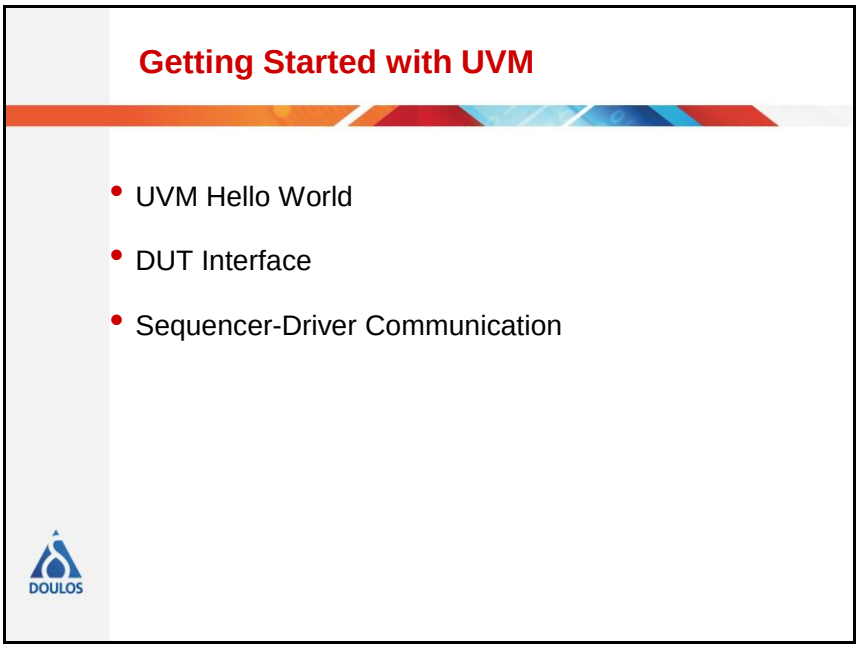


- A *Mem_map_trans* object can be used anywhere a *Bus_trans* object is appropriate

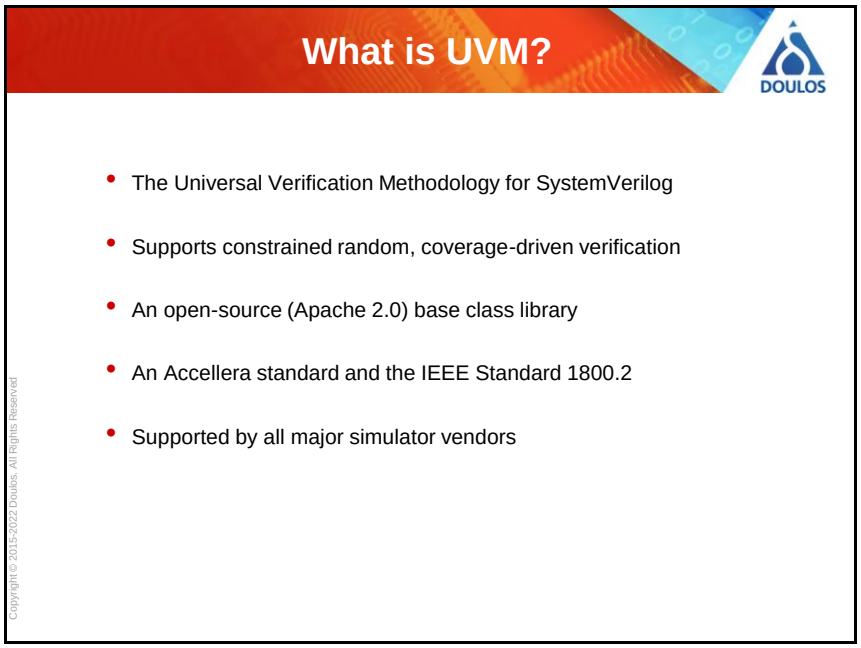
Notes


Control Knobs and Constraints
DOULOS
class Mem_map_trans extends Bus_trans;
"control knob" variable
rand enum { ROM, RAM, IO } area;
All enum values covered -
no other addresses will be used
constraint address_map {
area == ROM -> addr inside {['h0000:'h7FFF]};
area == RAM -> addr inside {['h8000:'hDFFF]};
area == IO -> addr inside {['hFF80:'hFFFF]};
}
implication allows "control knob" to enable other constraints
constraint ROM_read_only { area == ROM -> dir == dir_Rd; }
constraint IO_byte_wide { area == IO -> data[15:8] == 0; }
constraint area_choice {
area dist { ROM := 70, RAM := 20, IO := 10 };
}
Without this distribution constraint,
I/O access would be very rare
(only 1/512 of addr range, 1/256 of data range)
Copyright © 2015-2022 Doulos. All Rights Reserved

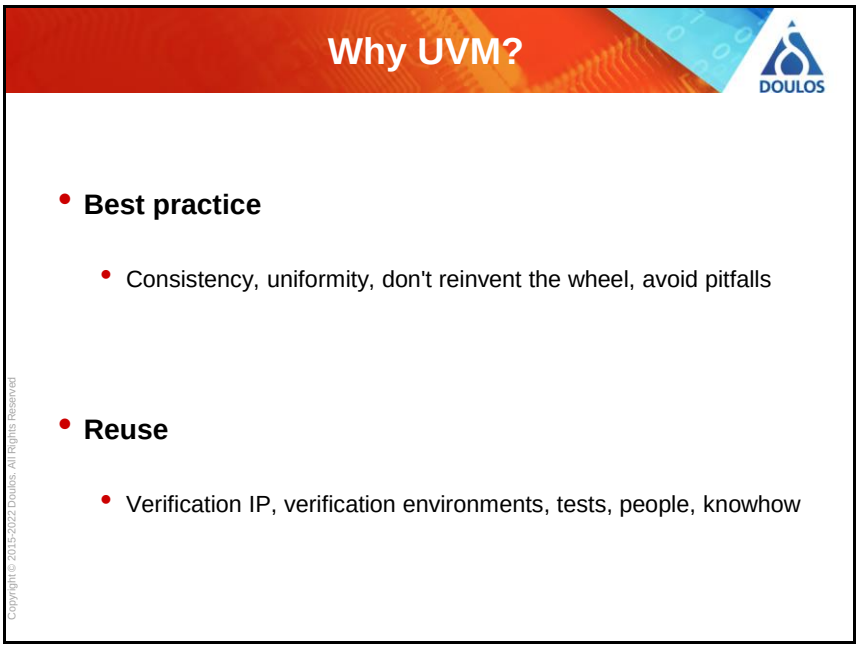
Notes



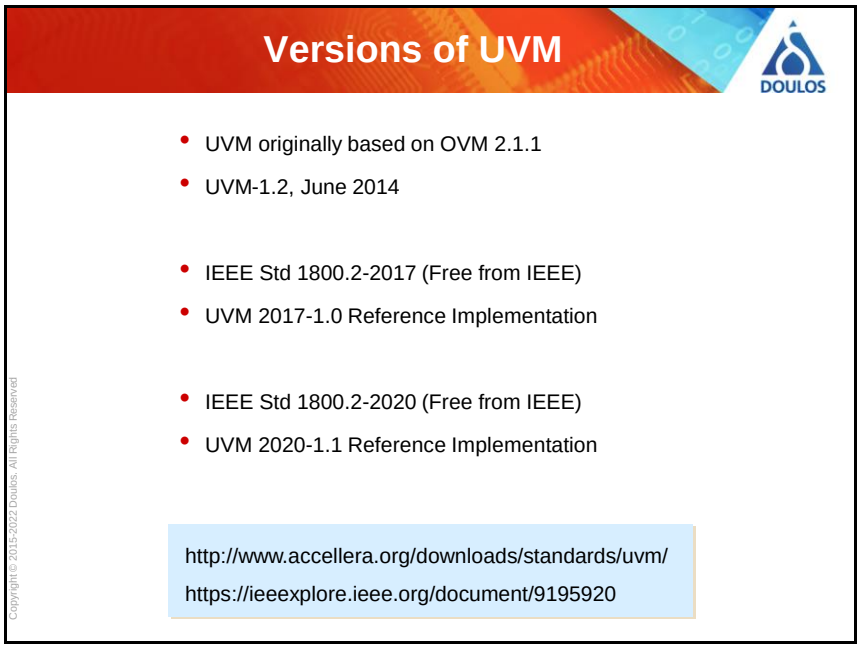
Notes



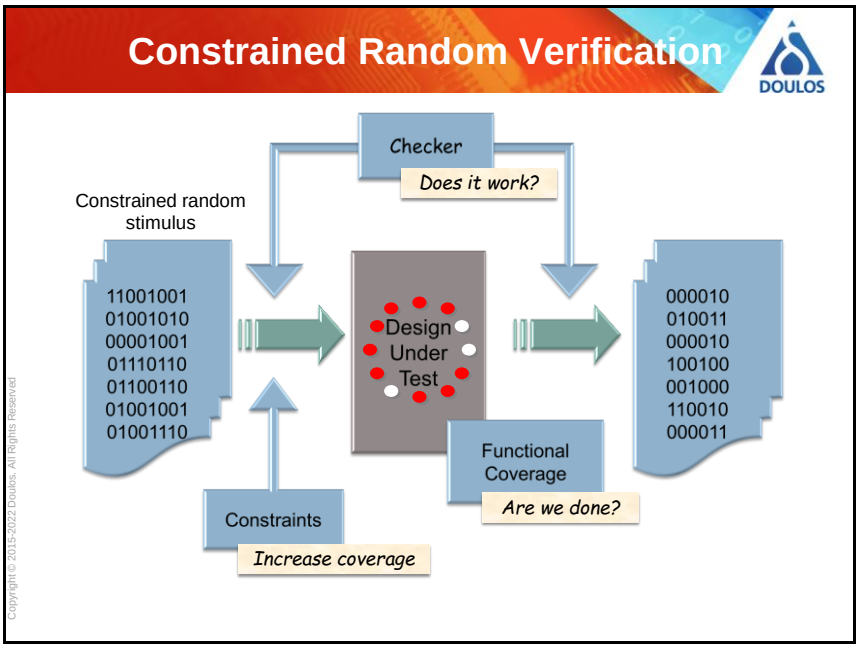
Notes



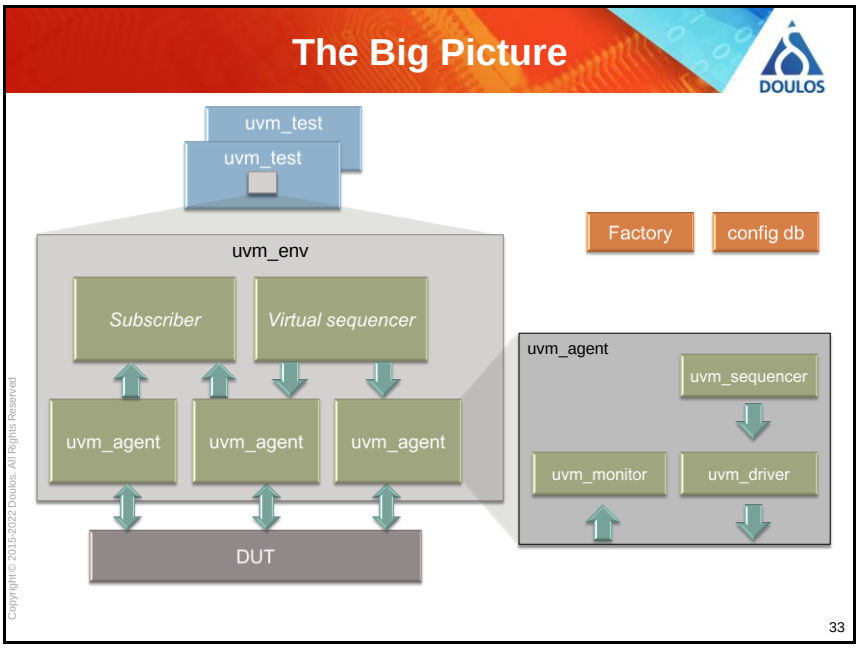
Notes



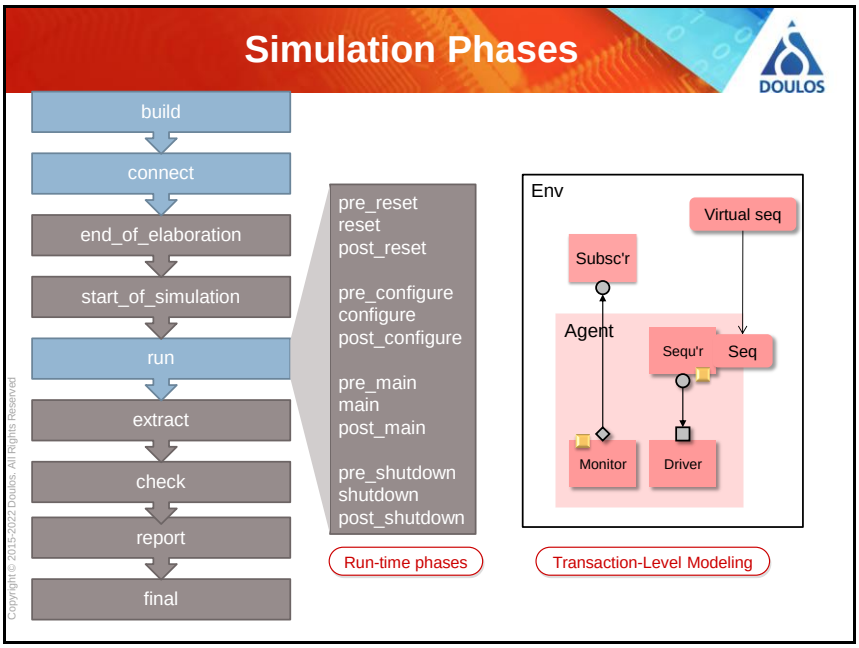
Notes



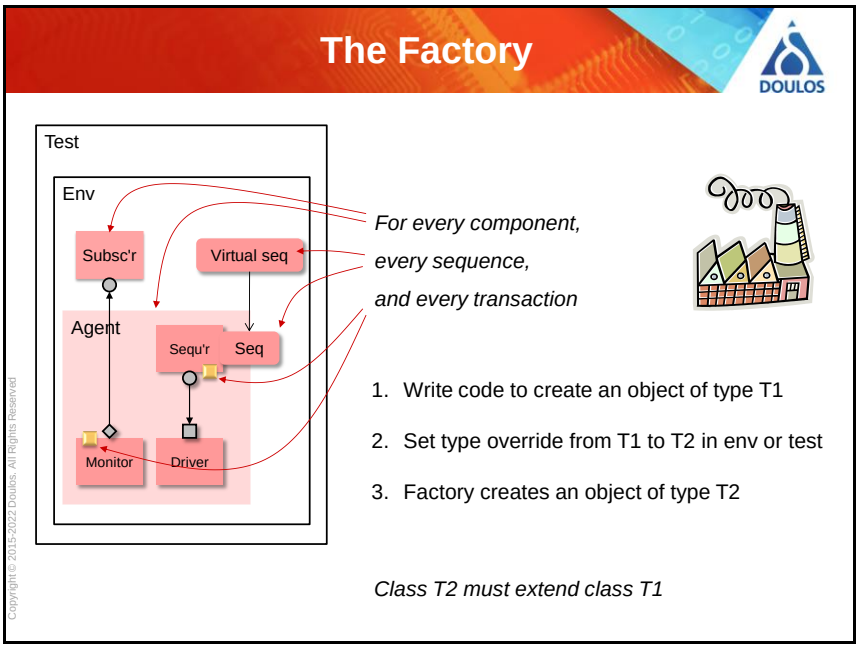
Notes


The Big Picture
DOULOS
uvm_test
uvm_test
Factory
config db
uvm_env
Subscriber
Virtual sequencer
uvm_agent
uvm_agent
uvm_agent
uvm_agent
uvm_sequencer
uvm_monitor
uvm_driver
DUT
Copyright © 2015-2022 Doulos. All Rights Reserved
33

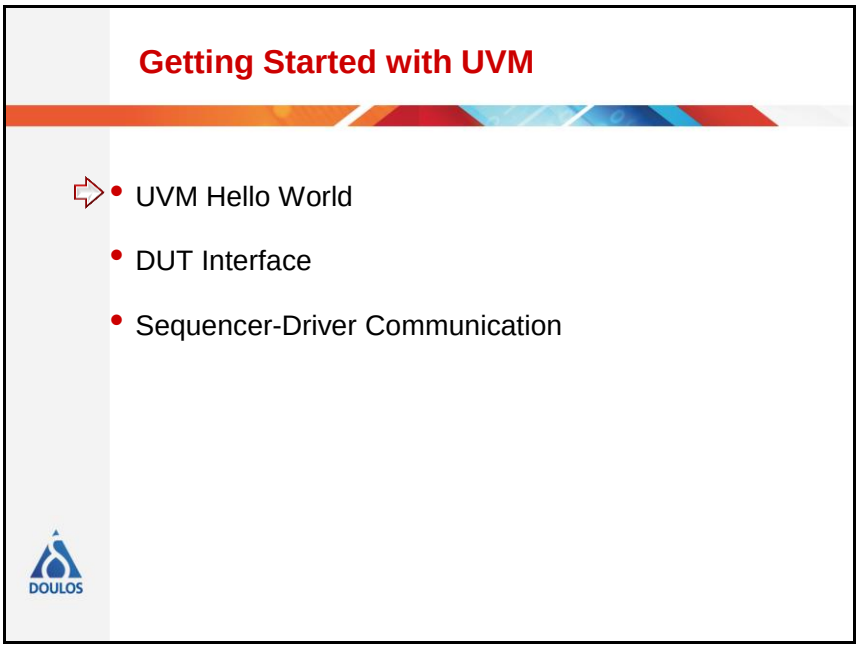
Notes



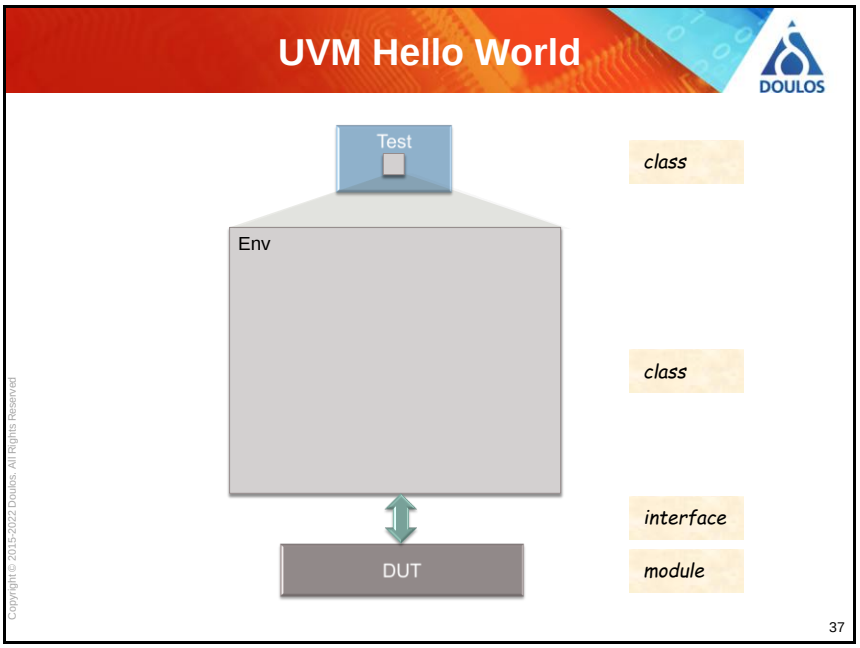
Notes


The Factory
DOULOS
Test
Env
Subsc'r
Virtual seq
Agent
Sequ'r
Seq
Monitor
Driver
For every component,
every sequence,
and every transaction
1. Write code to create an object of type T1
2. Set type override from T1 to T2 in env or test
3. Factory creates an object of type T2
Class T2 must extend class T1
Copyright © 2015-2022 Doulos. All Rights Reserved

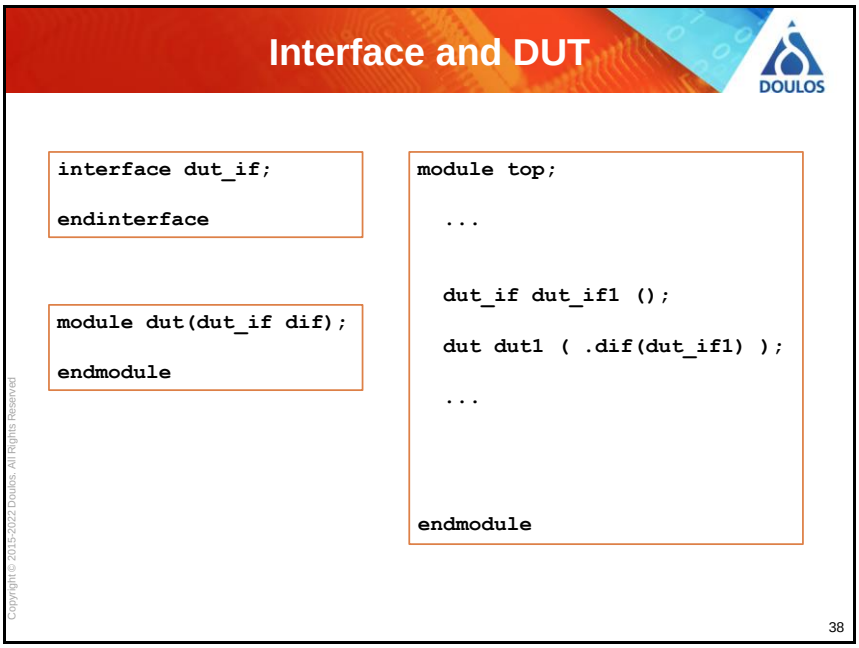
Notes



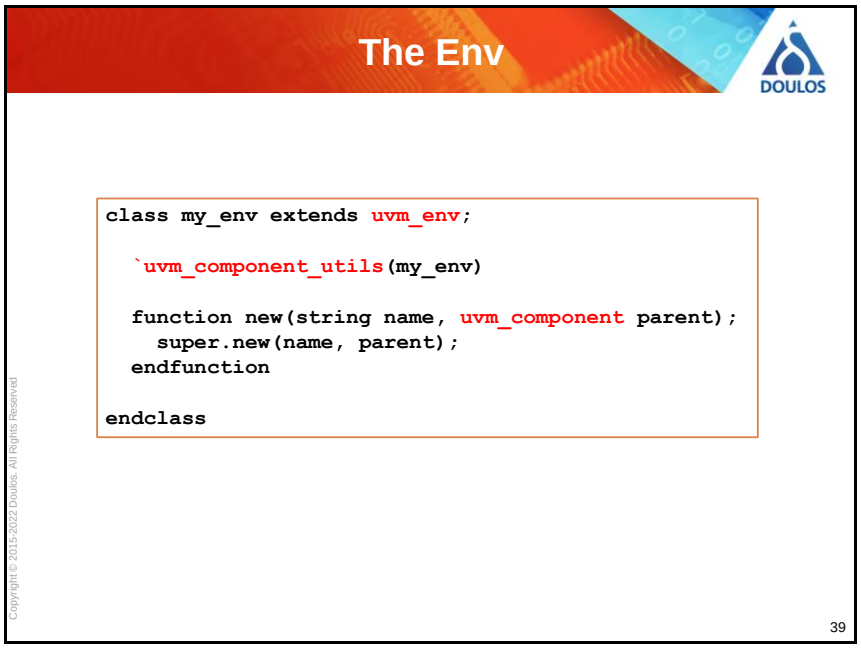
Notes


UVM Hello World
DOULOS
Test
Env
DUT
class
class
interface
module
Copyright © 2015-2022 Doulos. All Rights Reserved
37

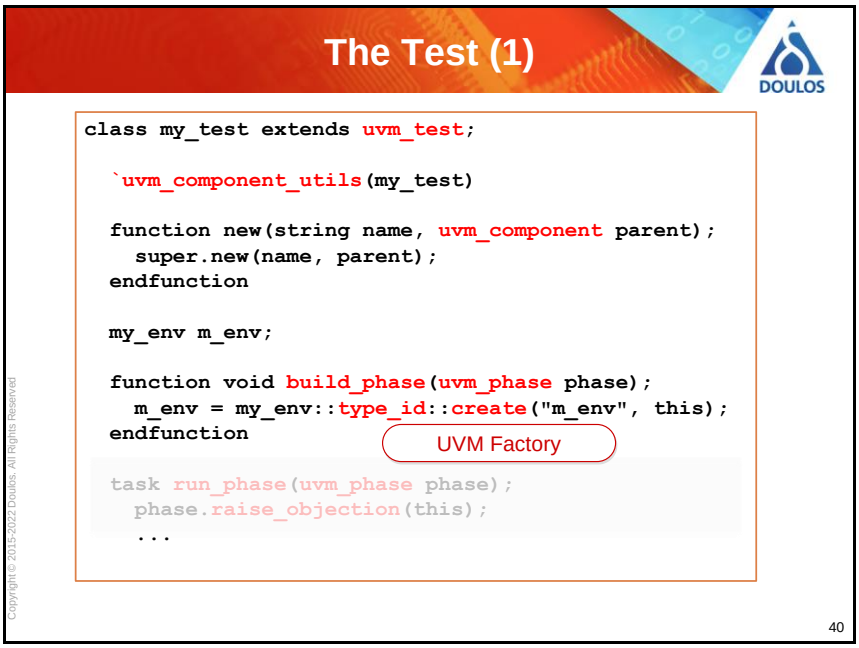
Notes



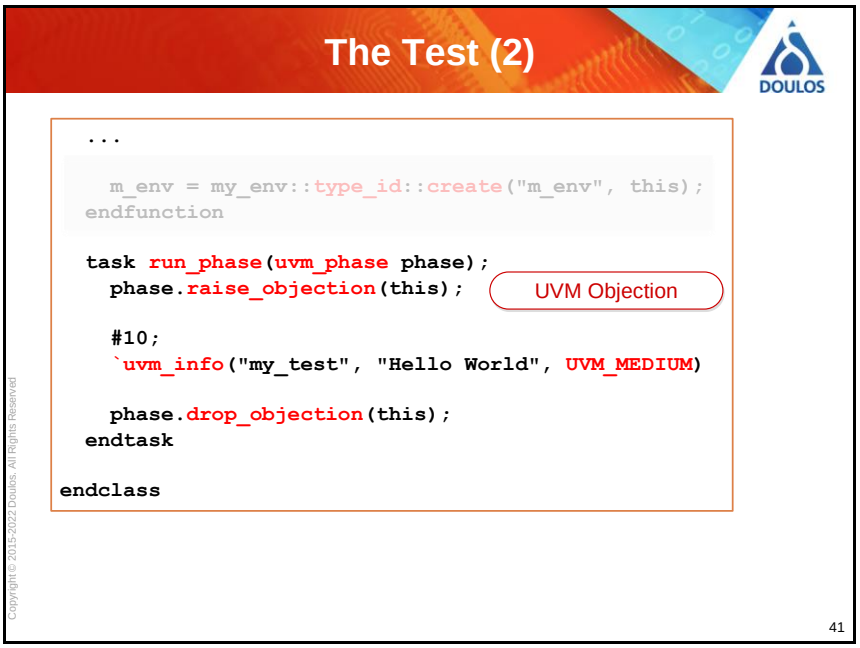
Notes


The Env
DOULOS
class my_env extends uvm_env;
`uvm_component_utils(my_env)
function new(string name, uvm_component parent);
super.new(name, parent);
endfunction
endclass
Copyright © 2015-2022 Doulos. All Rights Reserved
39

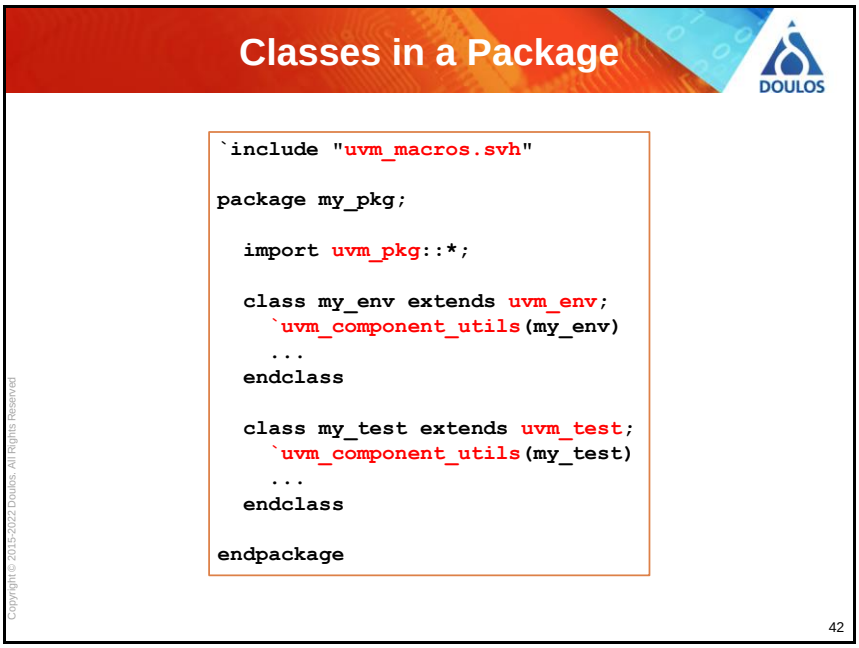
Notes



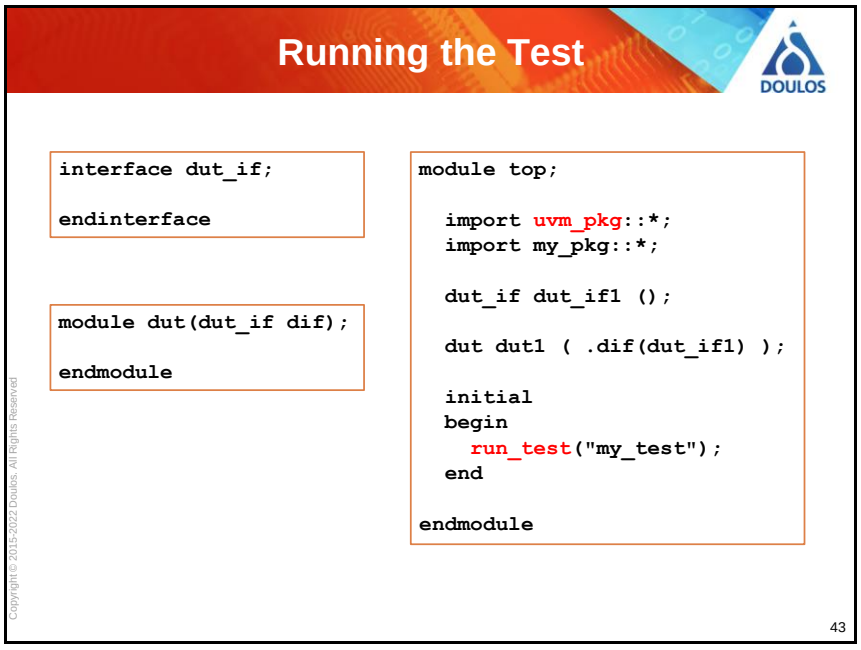
Notes


The Test (2)
DOULOS
...
m_env = my_env::type_id::create("m_env", this);
endfunction
task run_phase(uvm_phase phase);
phase.raise_objection(this);
UVM Objection
#10;
`uvm_info("my_test", "Hello World", UVM_MEDIUM)
phase.drop_objection(this);
endtask
endclass
Copyright © 2015-2022 Doulos. All Rights Reserved
41

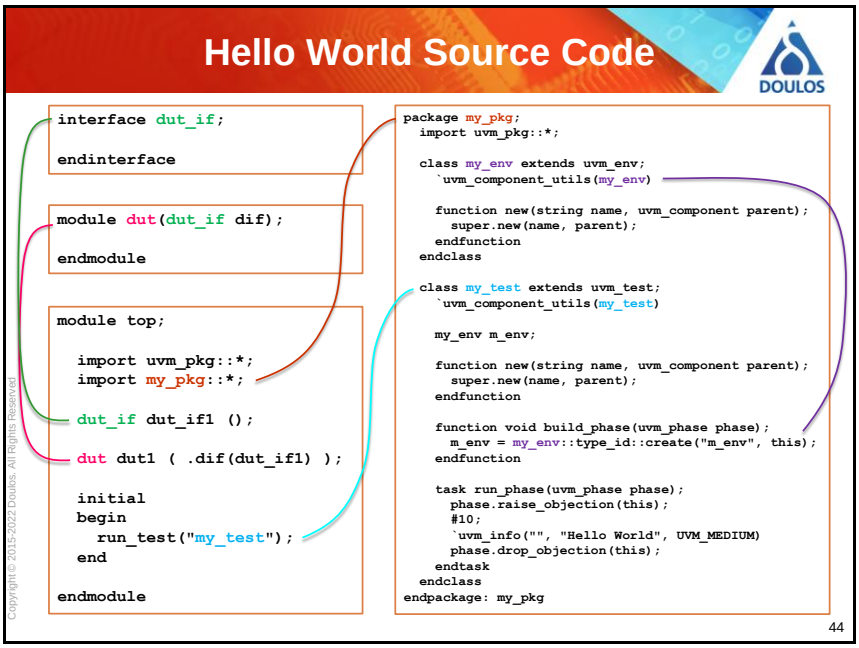
Notes



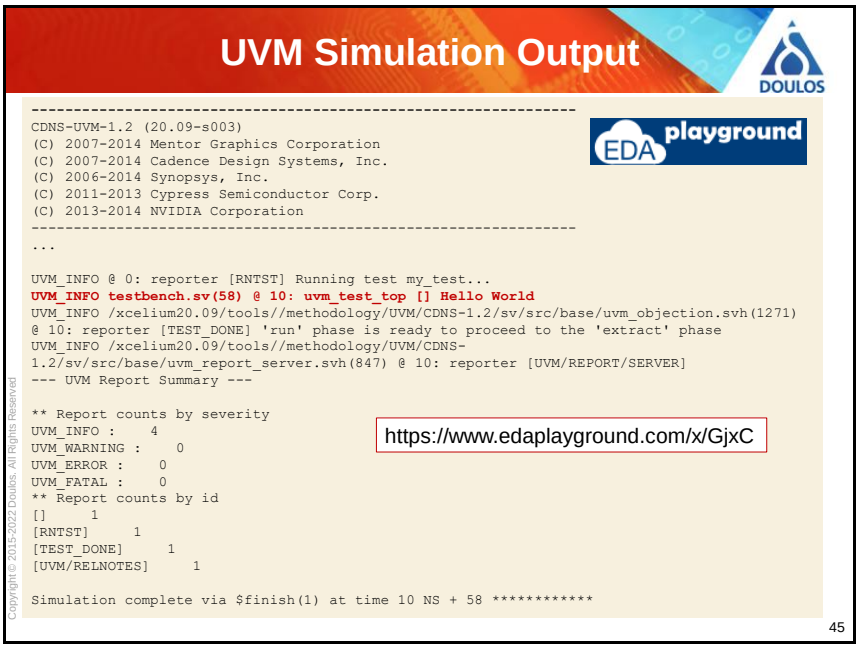
Notes


Running the Test
DOULOS
interface dut_if;
endinterface
module dut(dut_if dif);
endmodule
module top;
import uvm_pkg::*;
import my_pkg::*;
dut_if dut_if1 ();
dut dut1 (.dif(dut_if1));
initial
begin
run_test("my_test");
end
endmodule
Copyright © 2015-2022 Doulos. All Rights Reserved
43

Notes

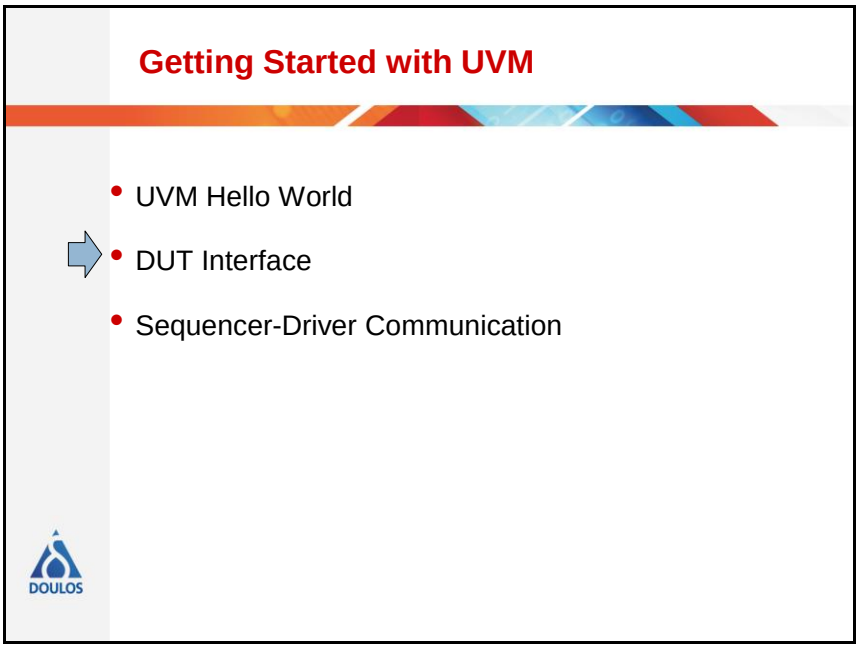


Notes

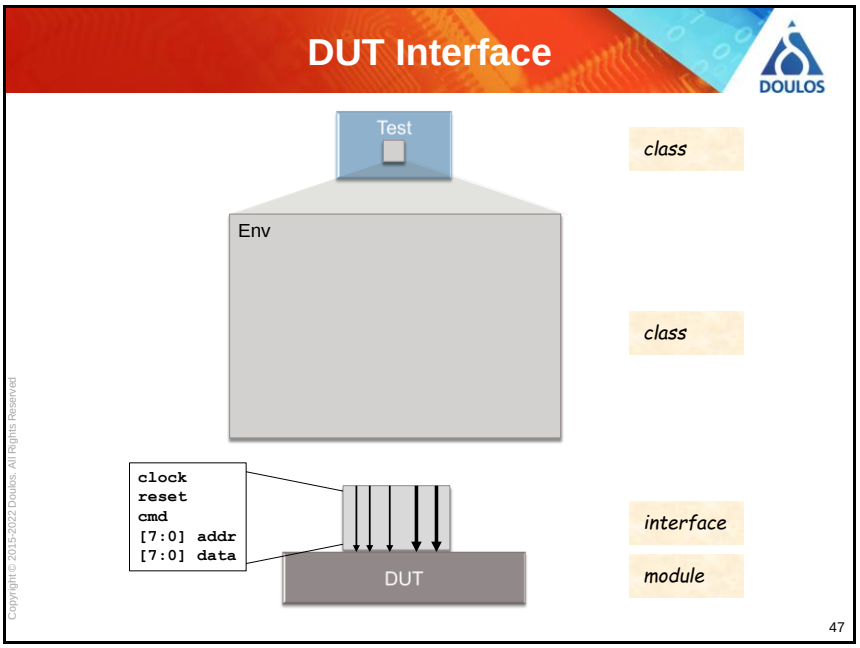



```
----------------------------------------------------------------
CDNS-UVM-1.2 (20.09-s003)
(C) 2007-2014 Mentor Graphics Corporation
(C) 2007-2014 Cadence Design Systems, Inc.
(C) 2006-2014 Synopsys, Inc.
(C) 2011-2013 Cypress Semiconductor Corp.
(C) 2013-2014 NVIDIA Corporation
----------------------------------------------------------------
...

UVM_INFO @ 0: reporter [RNTST] Running test my_test...
UVM_INFO testbench.sv(58) @ 10: uvm_test_top [] Hello World
UVM_INFO /xcelium20.09/tools//methodology/UVM/CDNS-1.2/sv/src/base/uvm_objection.svh(1271)
@ 10: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO /xcelium20.09/tools//methodology/UVM/CDNS-
1.2/sv/src/base/uvm_report_server.svh(847) @ 10: reporter [UVM/REPORT/SERVER]
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :    4
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[]     1
[RNTST]     1
[TEST_DONE]     1
[UVM/RELNOTES]     1

Simulation complete via $finish(1) at time 10 NS + 58 ************
```

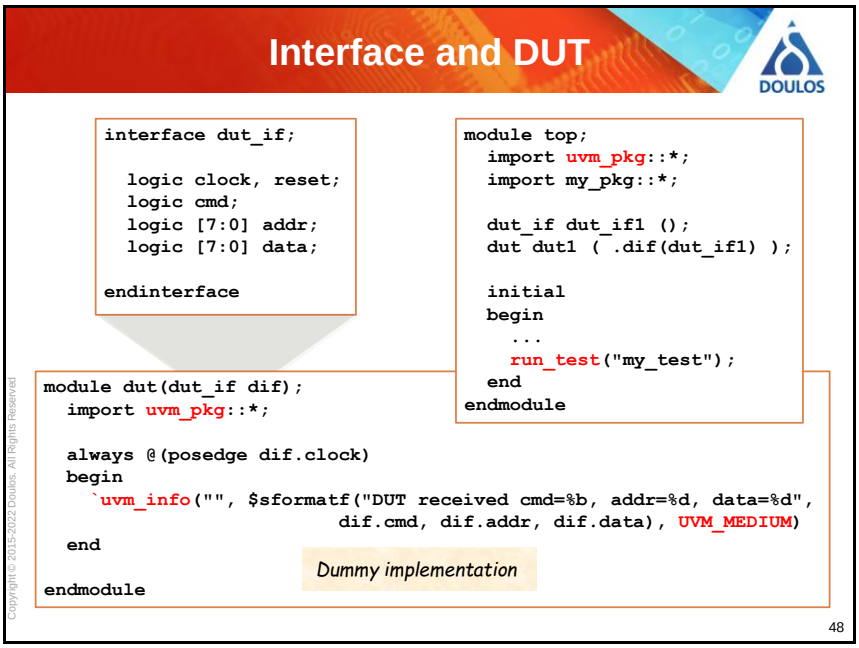

Notes



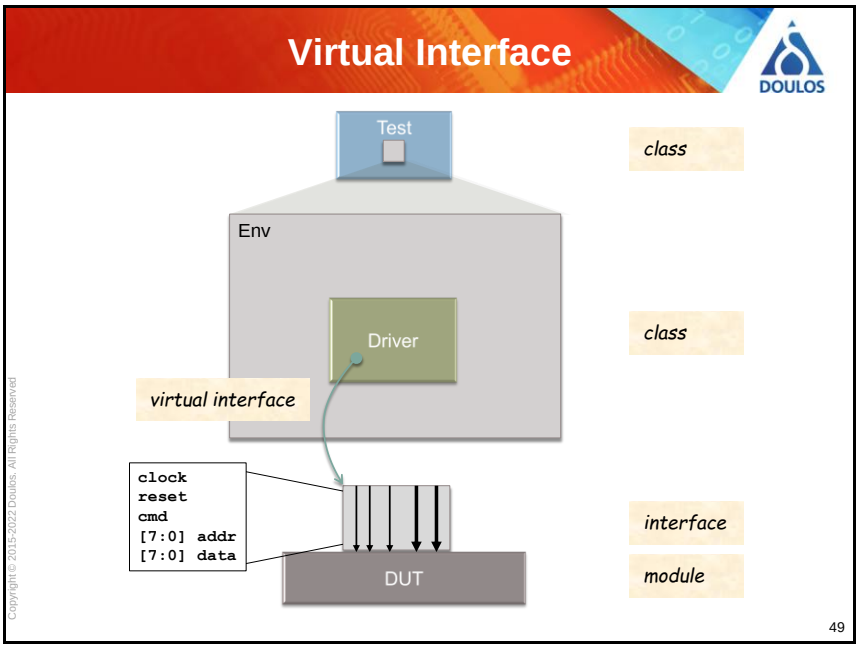
Notes


DUT Interface
DOULOS
Test
class
Env
class
clock
reset
cmd
[7:0] addr
[7:0] data
interface
DUT
module
47
Copyright © 2015-2022 Doulos. All Rights Reserved

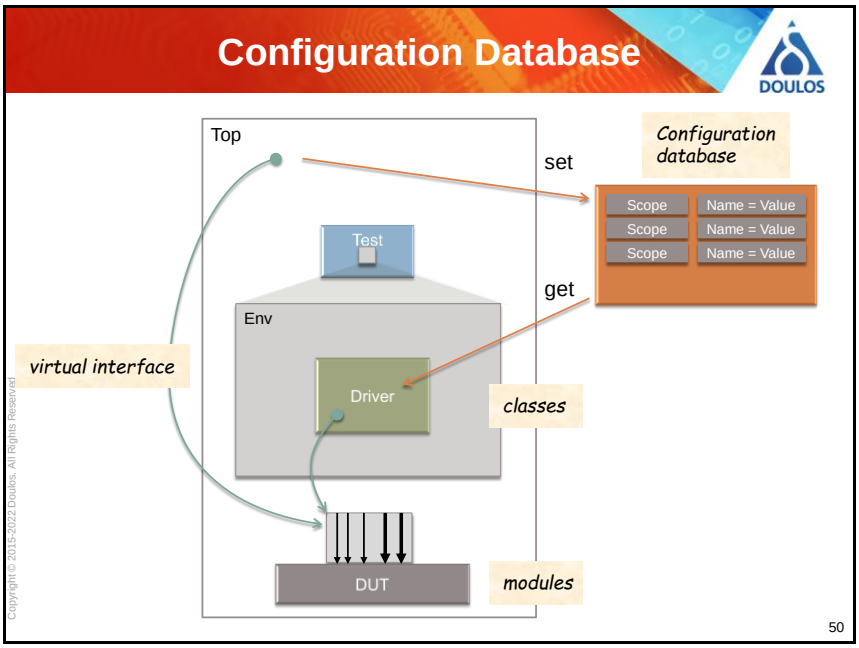
Notes



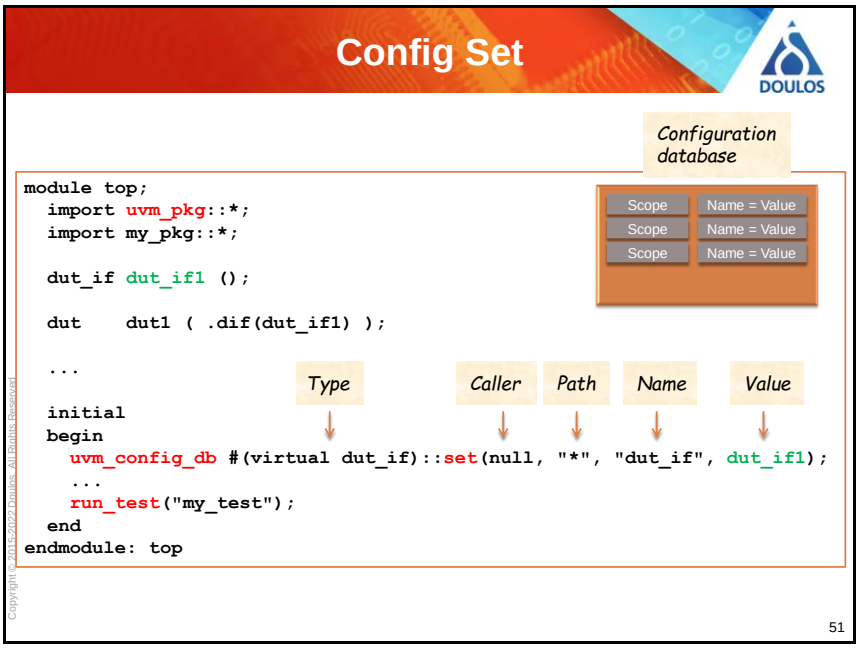
Notes


Virtual Interface
DOULOS
Test
class
Env
Driver
class
virtual interface
clock
reset
cmd
[7:0] addr
[7:0] data
interface
DUT
module
Copyright © 2015-2022 Doulos. All Rights Reserved
49

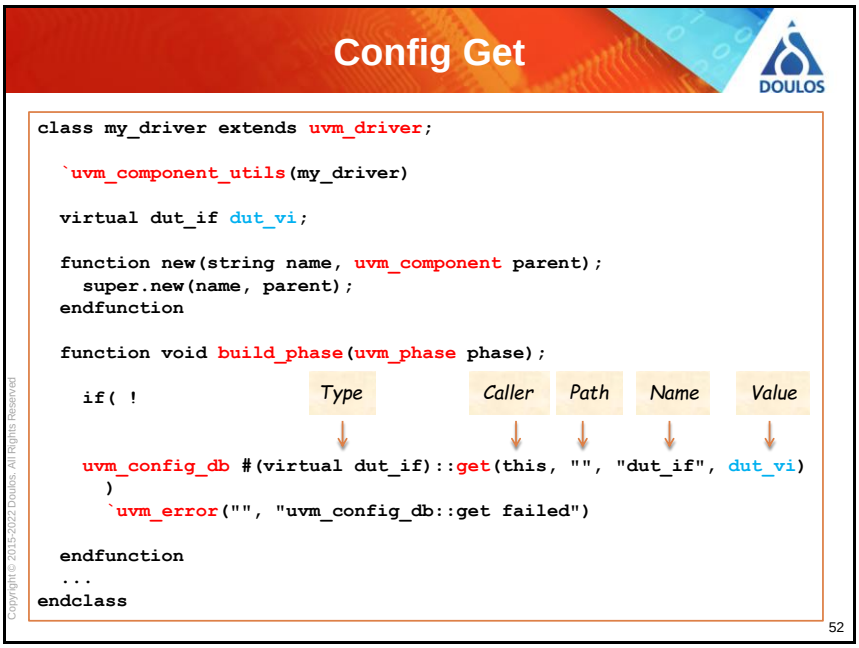
Notes



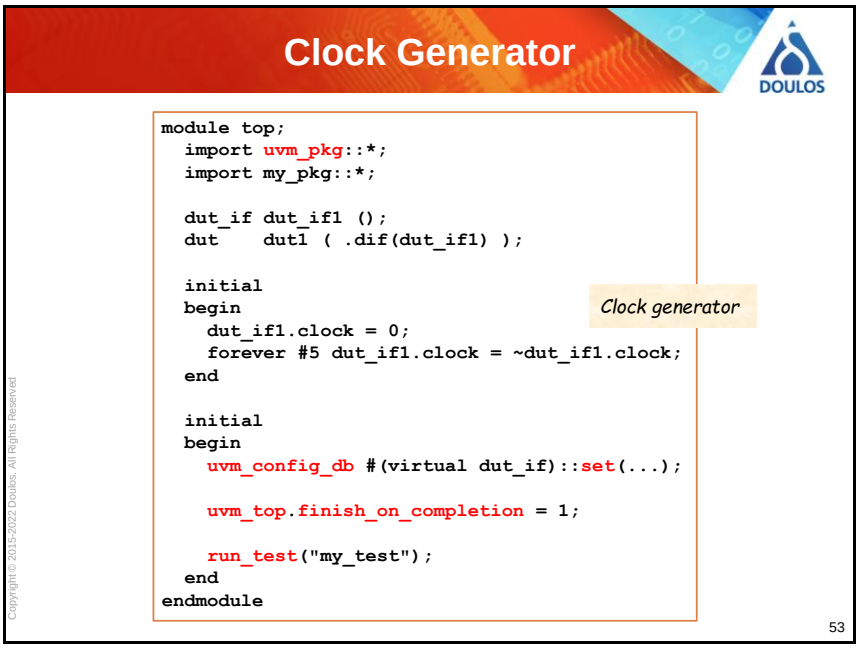
Notes


Config Set
DOULOS
Configuration database
Scope
Name = Value
Scope
Name = Value
Scope
Name = Value
module top;
import uvm_pkg::*;
import my_pkg::*;
dut_if dut_if1 ();
dut dut1 (.dif(dut_if1));
...
initial
begin
uvm_config_db #(virtual dut_if)::set(null, "*", "dut_if", dut_if1);
...
run_test("my_test");
end
endmodule: top
Type
Caller
Path
Name
Value
Copyright © 2015-2022 Doulos. All Rights Reserved
51

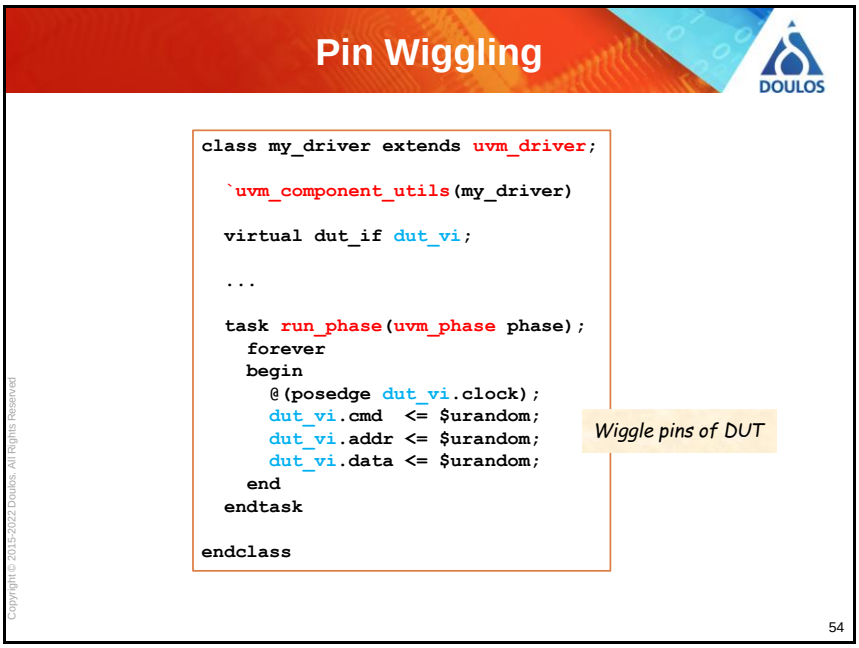
Notes



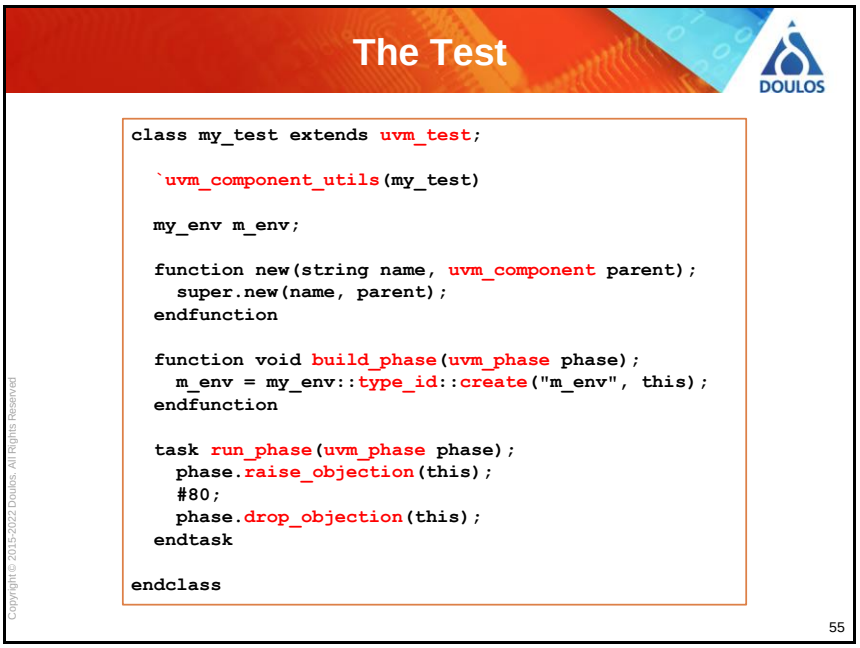
Notes


Clock Generator
DOULOS
module top;
import uvm_pkg::*;
import my_pkg::*;
dut_if dut_if1 ();
dut dut1 (.dif(dut_if1));
initial
begin
dut_if1.clock = 0;
forever #5 dut_if1.clock = ~dut_if1.clock;
end
initial
begin
uvm_config_db #(virtual dut_if)::set(...);
uvm_top.finish_on_completion = 1;
run_test("my_test");
end
endmodule
Clock generator
Copyright © 2015-2022 Doulos. All Rights Reserved
53

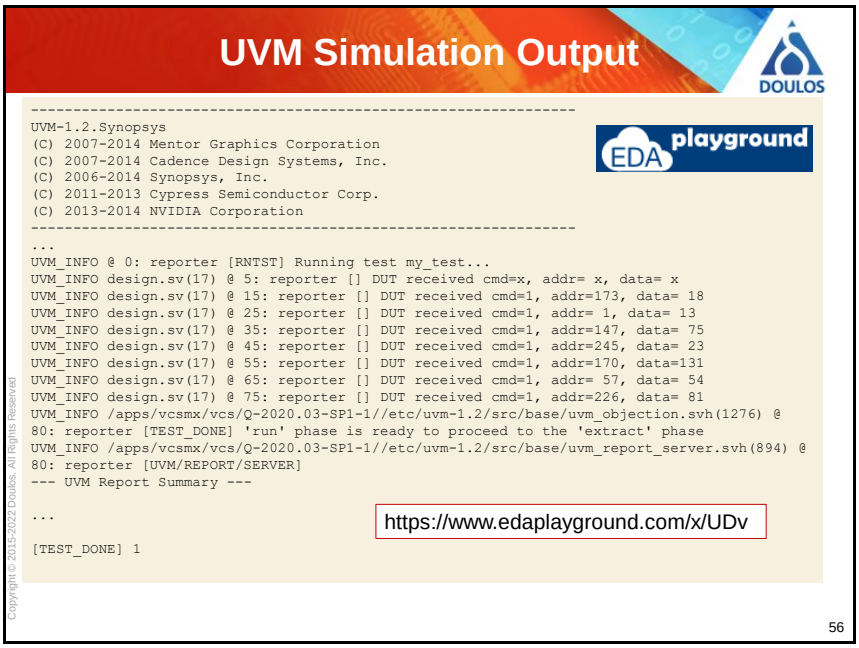
Notes



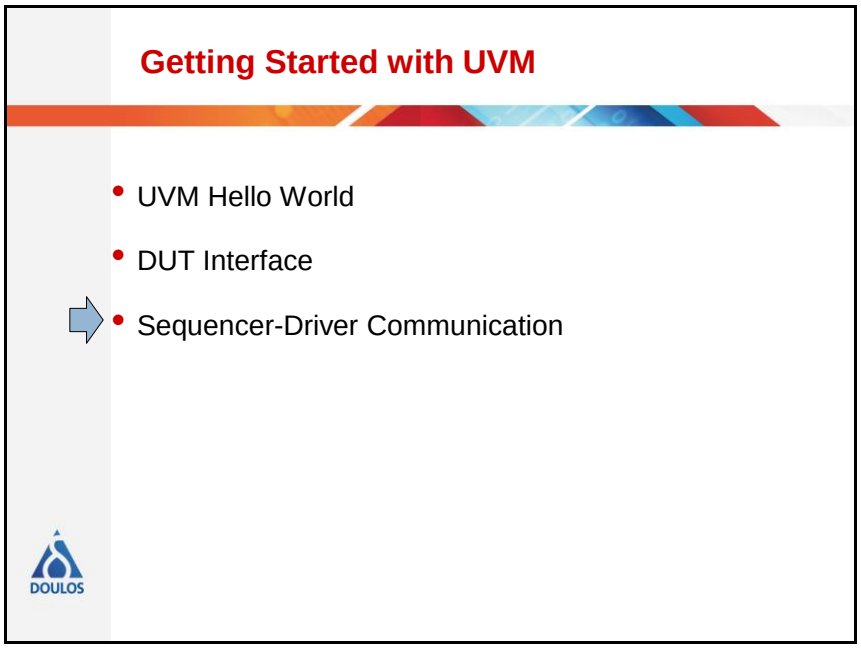
Notes


The Test
DOULOS
class my_test extends uvm_test;
`uvm_component_utils(my_test)
my_env m_env;
function new(string name, uvm_component parent);
super.new(name, parent);
endfunction
function void build_phase(uvm_phase phase);
m_env = my_env::type_id::create("m_env", this);
endfunction
task run_phase(uvm_phase phase);
phase.raise_objection(this);
#80;
phase.drop_objection(this);
endtask
endclass
Copyright © 2015-2022 Doulos. All Rights Reserved
55

Notes



Notes



Notes

Sequence Item Class

```
class my_transaction extends uvm_sequence_item;

  rand bit cmd;
  rand bit [7:0] addr, data;

  function new (string name = "");
    super.new(name);
  endfunction

  `uvm_object_utils_begin(my_transaction)
    `uvm_field_int(cmd, UVM_DEFAULT)
    `uvm_field_int(addr, UVM_DEFAULT)
    `uvm_field_int(data, UVM_DEFAULT)
  `uvm_object_utils_end

endclass
```


Notes

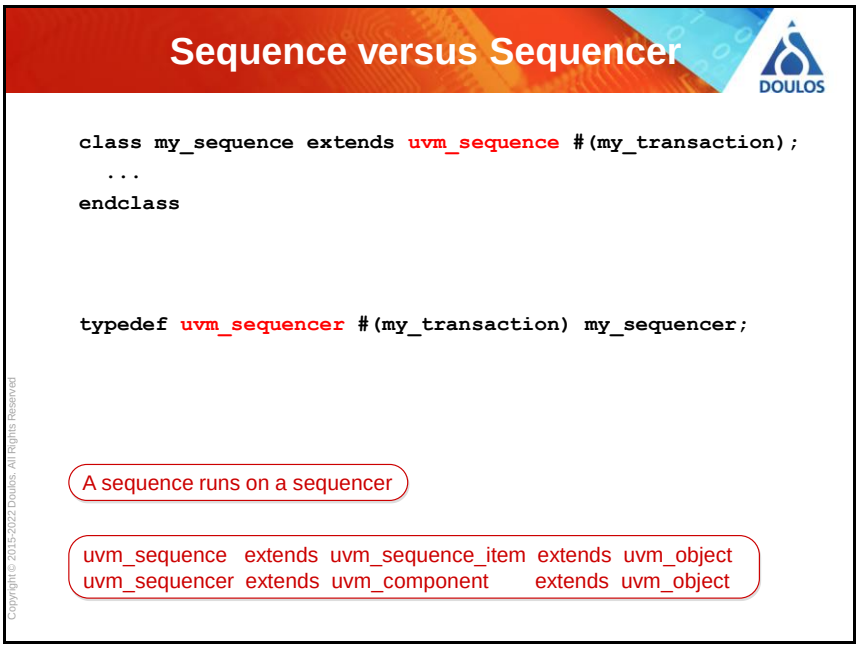
Sequence Class

```
class my_sequence extends uvm_sequence #(my_transaction);

 `uvm_object_utils(my_sequence)

  function new (string name = "");
    super.new(name);
  endfunction

  task body;
    repeat(8)
    begin
      `uvm_do(req)
    end
  endtask

endclass
```

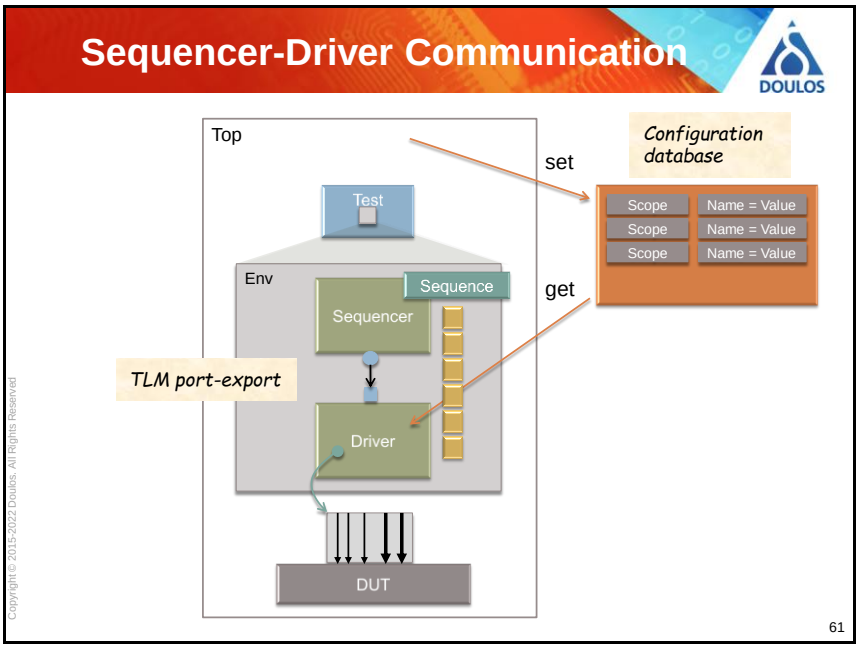
`uvm_do creates and sends a randomized transaction to the driver

req inherited from uvm_sequence

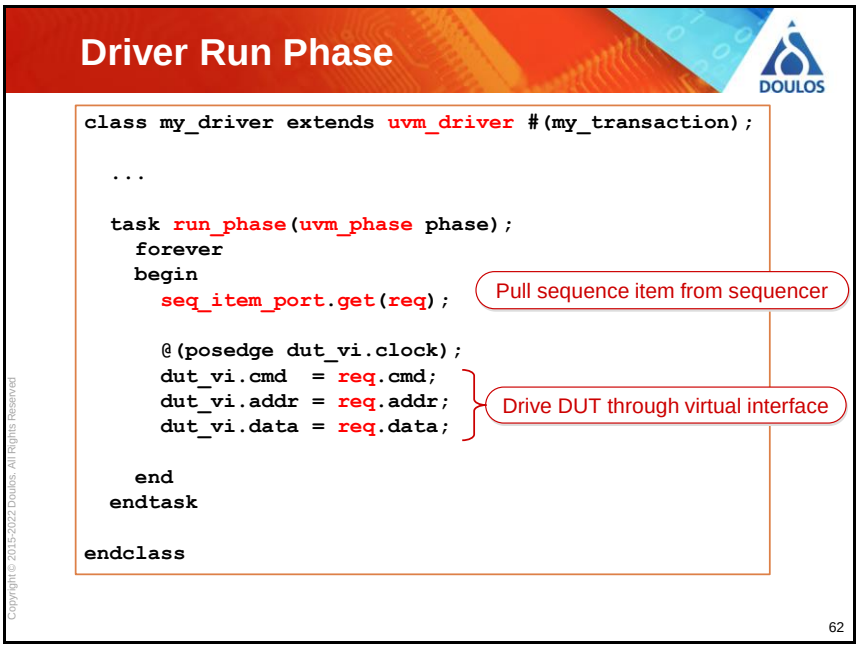
Notes



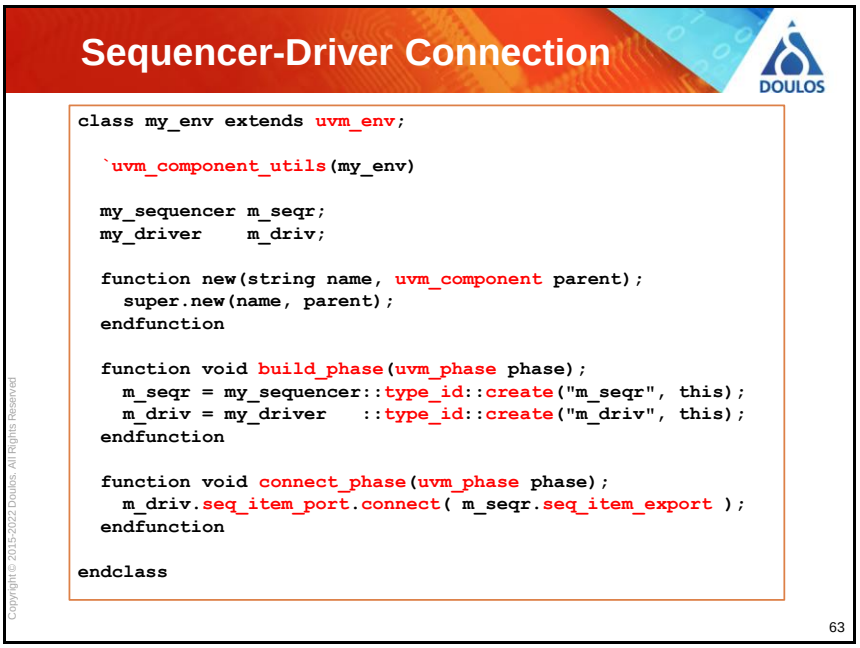
Notes


Sequencer-Driver Communication
DOULOS
Top
Test
Env
Sequence
Sequencer
Driver
DUT
TLM port-export
set
get
Configuration database
Scope
Name = Value
Scope
Name = Value
Scope
Name = Value
Copyright © 2015-2022 Doulos. All Rights Reserved
61

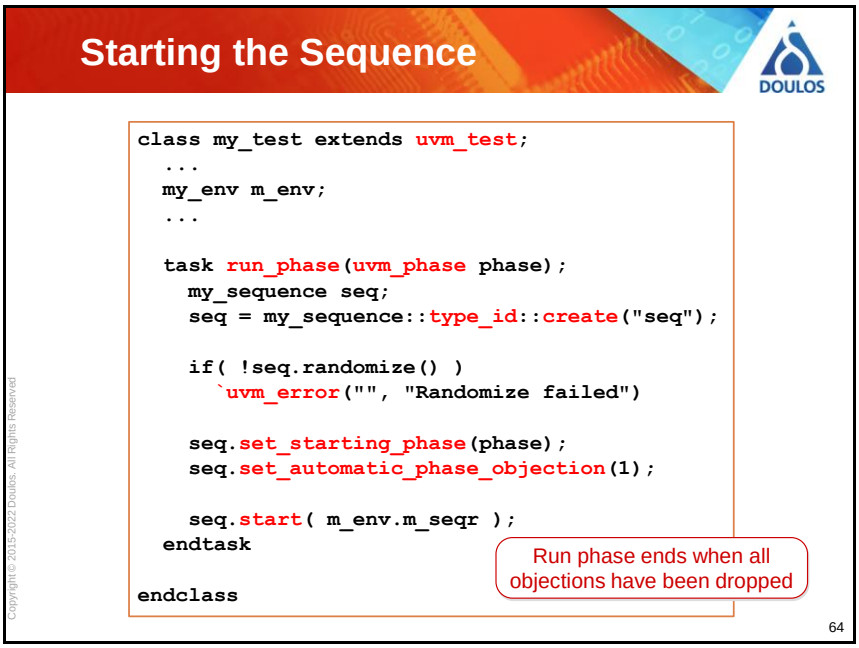
Notes



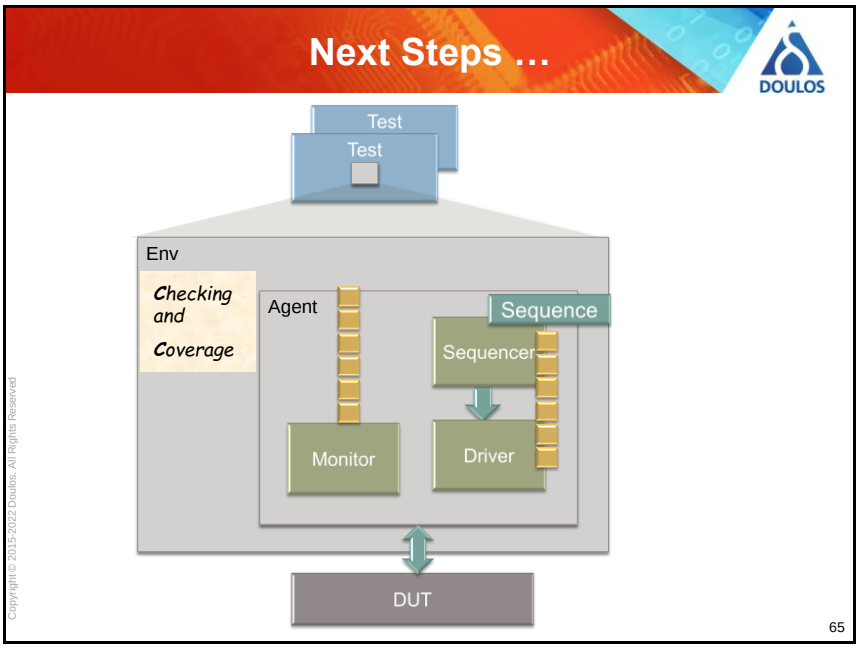
Notes


Sequencer-Driver Connection
DOULOS
class my_env extends uvm_env;
`uvm_component_utils(my_env)
my_sequencer m_seqr;
my_driver m_driv;
function new(string name, uvm_component parent);
super.new(name, parent);
endfunction
function void build_phase(uvm_phase phase);
m_seqr = my_sequencer::type_id::create("m_seqr", this);
m_driv = my_driver ::type_id::create("m_driv", this);
endfunction
function void connect_phase(uvm_phase phase);
m_driv.seq_item_port.connect(m_seqr.seq_item_export);
endfunction
endclass
Copyright © 2015-2022 Doulos. All Rights Reserved
63

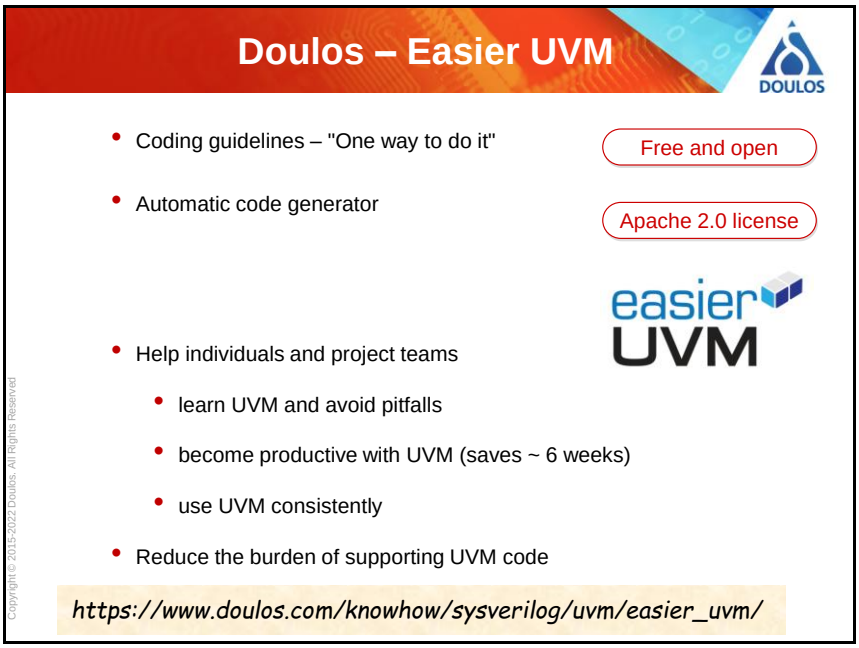
Notes



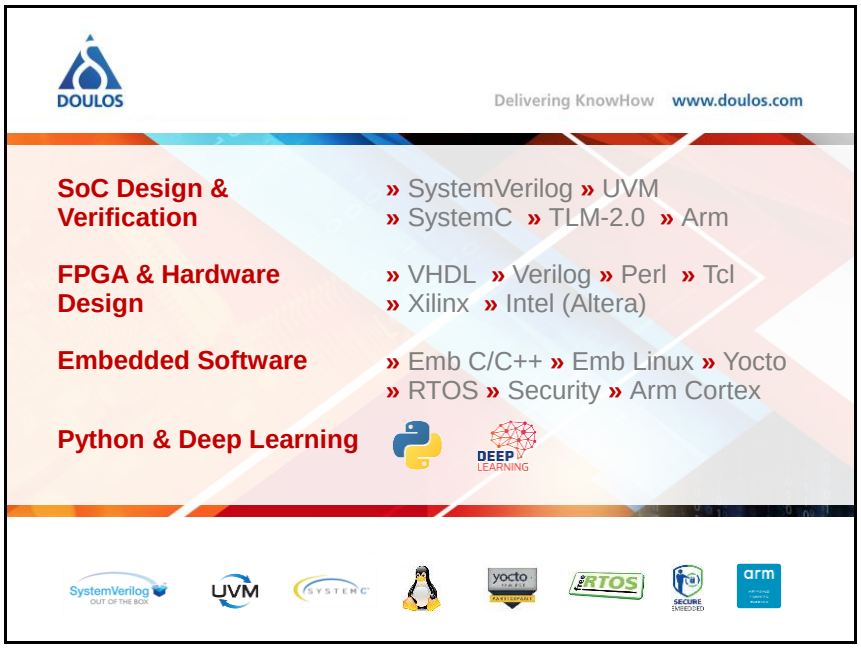
Notes


Next Steps …
DOULOS
Test
Test
Env
Checking and Coverage
Agent
Sequence
Sequencer
Monitor
Driver
DUT
65
Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



Notes



Notes