

DOULOS Delivering KnowHow www.doulos.com

Getting Started with SystemVerilog and UVM

Presenter: Dr David Long
Principal Member of Technical Staff

Copy

Notes

Getting Started with SystemVerilog

- What is SystemVerilog?
- SystemVerilog Classes
- Use of Interfaces
- Constraints

Notes

What is SystemVerilog?



- The world's first HDVL, Hardware Design and Verification Language
 - IEEE Std 1364-2005 Verilog and
 - IEEE Std 1800-2005 SystemVerilog
- merged to form
- IEEE Std 1800-2009 SystemVerilog
 - IEEE Std 1800-2012 SystemVerilog
 - IEEE Std 1800-2017 SystemVerilog
-
- **SystemVerilog RTL**, aka concise RTL
 - **SystemVerilog Assertions**, aka SVA
 - **SystemVerilog Test Bench**, or class-based verification

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

SystemVerilog Language Features

RTL + programming

- C-style data types & control - enum, struct, typedef, ++, break, return
- Synthesis-friendly "concise" RTL notation
- Packages
- Interfaces

Assertions

- SystemVerilog Assertions

Test Bench


- Clocking blocks (synchronization between DUT and test bench)
- Object-oriented programming - classes
- Constrained random stimulus generation
- Functional coverage
- Dynamic processes, dynamic arrays, queues, mailboxes, semaphores

- Direct Programming Interface (DPI) - calling C from SystemVerilog
- Extensions to VPI

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Caveats



- C-like control constructs and data types
- Concise RTL
- VHDL-like package and import
- Assertions

A better Verilog

- Non-portable constructs

Ill-defined

- Classes
- Constraints and coverage based on classes
- Built-in types - strings, queues, maps
- Virtual interfaces

Class-based verification

Used by standard verification methodologies

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

SystemVerilog Classes



```
package Bus_pkg;
typedef logic [15:0] T_addr;
typedef logic [15:0] T_data;
typedef enum bit {dir_Rd, dir_Wr} T_dir;
class Bus_trans;
    int ID;
    T_dir dir;
    T_addr addr;
    T_data data;
    function void print;
        string kind = (dir==dir_Rd) ? "Read" : "Write";
        $display("%s cycle #%0d: A=%h, D=%h",
            kind, ID, addr, data);
    endfunction : print
endclass : Bus_trans

endpackage : Bus_pkg
```


Always put classes in a package

Class defines a transaction object

Data members or class properties

Method

Notes



Object = Instance of Class

```

module use_Bus_trans;
import Bus_pkg::*;
Bus_trans t1, t2;
initial begin
    t1 = new;
    t1.data = 16'h1234;
    t2 = t1;
    t2.data = 16'habcd;
    t1.print();
    ...
        
```

references

t1

t2

Bus_trans **object**

ID

dir

addr

data

Method call Read cycle #0: A=xxxx, D=abcd

- 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

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Initializing Objects



```
module use_Bus_trans;  
import Bus_pkg::*;  
Bus_trans t3 = new;
```

Create object with default initial values

- **new** allocates memory and calls default constructor

```
class Bus_trans;  
...  
function new;  
    addr = 0;  
    dir = dir_Wr;  
    $write("Created new ");  
    print();  
endfunction : new  
...
```

Explicit constructor new. No return type

Call a method from within the class

print myself

Copyright © 2015-2022 Doulos. All Rights Reserved

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;
```

ERROR

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

Read

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

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

```
Bus_trans t5 = new;
```

Read

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

Write

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

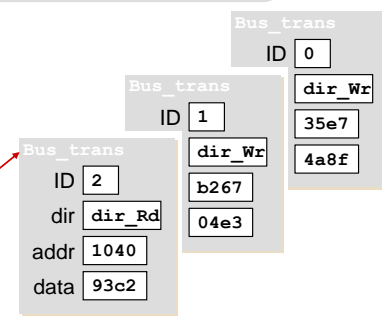
Randomized Data Members

```

class Bus_trans;
  static int next_ID;
  const int ID;
  rand T_dir dir;
  rand T_addr addr;
  rand T_data data;
  function new;
    ID = next_ID++;
  endfunction : new
  function void print; ...
endclass : Bus_trans

Bus_trans tR;
repeat (3) begin
  tR = new;
  void'( tR.randomize() );
  tR.print();
end

```

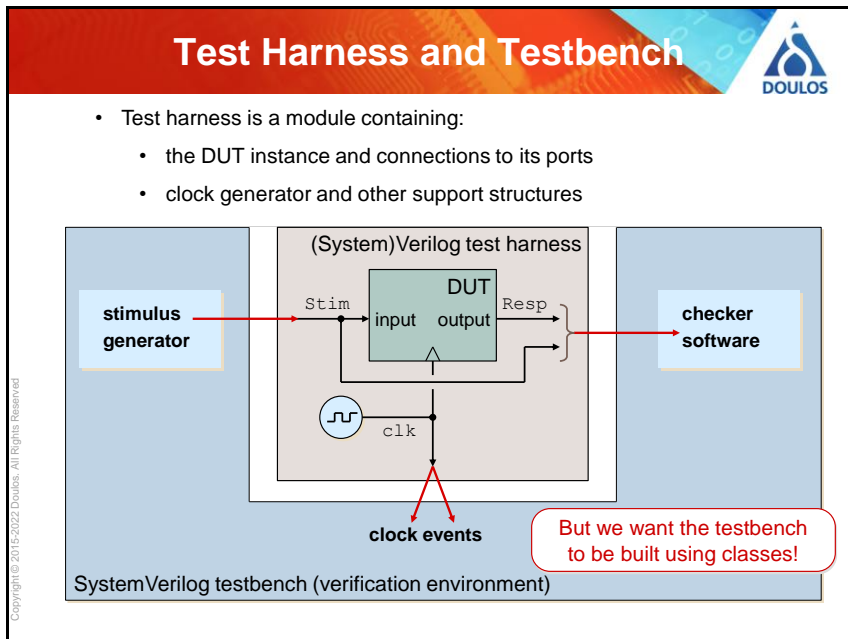


any data member can be declared rand
randomize an existing object

Write cycle #0: A=35e7, D=4a8f
 Write cycle #1: A=b267, D=04e3
 Read cycle #2: A=1040, D=93c2

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



Notes

Lifetime and Persistence



- Module, interface and program instances:
 - created at elaboration, before simulation begins
 - hierarchy structure controlled by parameters
 - structure/instances cannot be changed dynamically

- Objects of class type:
 - created dynamically, during simulation, using new
 - structure controlled by run-time activity
 - can be created and destroyed at any time

verification environment

- typically constructed at time zero
- structure probably remains unchanged throughout simulation

transaction data objects

- created in large numbers during the simulation
- destroyed after use (unless logged)

Notes

Creating the Testbench

```

module TB_top;
  import TB_pkg::*;
  TB_env tb;
  initial begin
    tb = new;
    tb.run();
  end
endmodule : TB_top

```

```

module harness;
  logic Stim, Resp;
  bit clk;
  Sys_Top DUT (.*)
  ...
endmodule

```

```

class TB_env;
  ...
  task drive_Stim(input bit data);
    @(posedge harness.clk)
    harness.Stim <= data;
  endtask
  ...

```

Test harness module

- Our entire testbench class is hard-coded for the name of the test harness!

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Virtual Interface

```

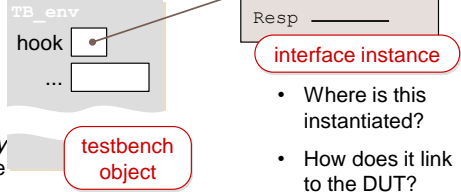
class TB_env;
  virtual TB_hook hook;
  function new(virtual TB_hook h);
    hook = h;
    ...
  endfunction : new
  task drive_Stim(input bit data);
    @(posedge hook.clk)
      hook.Stim <= data;
  endtask
  ...

```

```

interface TB_hook;
  logic Stim, Resp;
  bit clk;
endinterface

```



interface instance

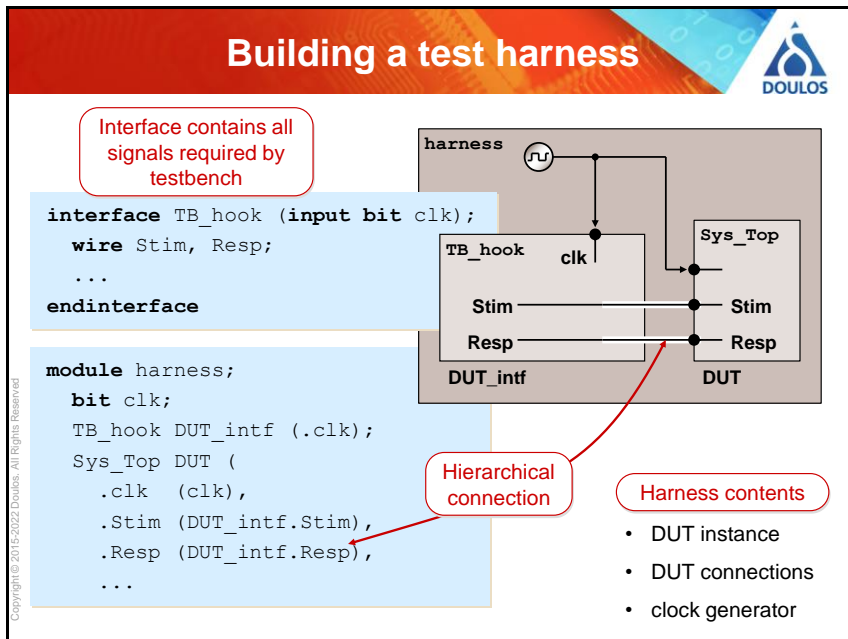
- Where is this instantiated?
- How does it link to the DUT?

- Testbench class now OK for *any* instance of a TB_hook interface
- Link TB object to interface instance at runtime

testbench object

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



Notes

Connecting the virtual interface

```
class TB_env;
  virtual TB_hook V;
  function new (virtual TB_hook V, ...);
    this.V = V;
    ...
  endfunction
  ...
endclass

module TB_top;
  TB_env tb;
  ...
  initial begin
    tb = new(harness.DUT_intf, ...);
  end
endmodule

module harness;
  bit clk;
  TB_hook DUT_intf (.clk);
  Sys_Top DUT (
    ...
  )
endmodule
```

constructor (points to `function new`)

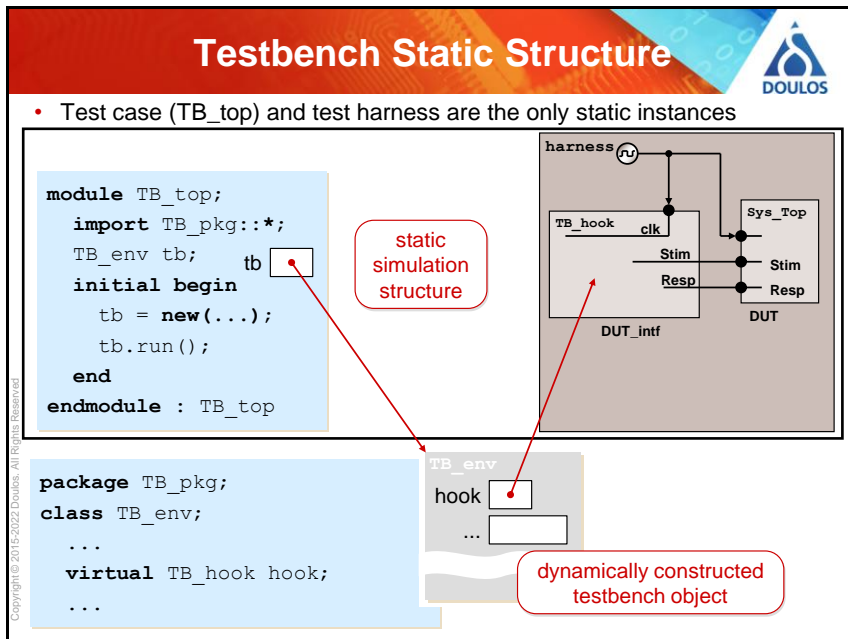
Choice of interface type (points to `virtual TB_hook V`)

test harness (points to `module harness`)

Choice of instance (points to `new(harness.DUT_intf, ...)`)

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



Notes

Constrained randomization

```

class Bus_trans;
  rand T_dir dir;
  rand T_addr addr;
  rand T_data data;
  constraint rom_area {
    dir == dir_Rd; addr <= 16'h7FFF;
  }
  ...
endclass : Bus_trans

```

System has ROM at low addresses

		dir	
		Rd	Wr
addr	FFFF	X	X
	8000	X	X
	7FFF	✓	X
	0000	✓	X

- But now there will be *no* access to high addresses!
- Solution: use an *implication constraint*

```

constraint low_adrs_is_ROM {
  (addr <= 16'h7FFF) -> (dir == dir_Rd);
}


```

		dir	
		Rd	Wr
addr	FFFF	✓	✓
	8000	✓	X
	7FFF	✓	X
	0000	✓	X

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Creating an Extended Class



```
class Bus_trans;
  rand T_dir dir;
  rand T_addr addr;
  rand T_data data;
```

General, re-usable

```
constraint low_adrs_is_ROM {
  (addr <= 16'h7FFF) -> (dir == dir_Rd);
}
```

Specific to the current DUT

- 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

Better not to mix these together...

Notes

Inheriting Class Members

- What you write

```

class Bus_trans {
  addr: T_addr
  data: T_data ...
  new()
  copy(): Bus_trans
  psprint(): string
}

class Mem_map_trans {
  area: enum{ROM,RAM,IO}
  constraint ...
}

Mem_map_trans inherits Bus_trans
          
```

- What you get

```


class Mem_map_trans ... is a Bus_trans {
  addr: T_addr
  data: T_data ...
  area: enum{ROM,RAM,IO}
  new()
  copy(): Bus_trans
  psprint(): string
  constraint ...
}
          
```

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

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Control Knobs and Constraints



```

class Mem_map_trans extends Bus_trans;

  rand enum { ROM, RAM, IO } area;

  constraint address_map {
    area == ROM -> addr inside {['h0000:'h7FFF]};
    area == RAM -> addr inside {['h8000:'hDFFF]};
    area == IO -> addr inside {['hFF80:'hFFFF]};
  }

  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 };
  }

```

"control knob" variable

All enum values covered - no other addresses will be used

implication allows "control knob" to enable other constraints

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

Getting Started with UVM

- UVM Hello World
- DUT Interface
- Sequencer-Driver Communication

Notes

What is UVM?



- The Universal Verification Methodology for SystemVerilog
- Supports constrained random, coverage-driven verification
- An open-source (Apache 2.0) base class library
- An Accellera standard and the IEEE Standard 1800.2
- Supported by all major simulator vendors

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes


Why UVM?

- **Best practice**
 - Consistency, uniformity, don't reinvent the wheel, avoid pitfalls
- **Reuse**
 - Verification IP, verification environments, tests, people, knowhow

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

Versions of UVM



- UVM originally based on OVM 2.1.1
- UVM-1.2, June 2014

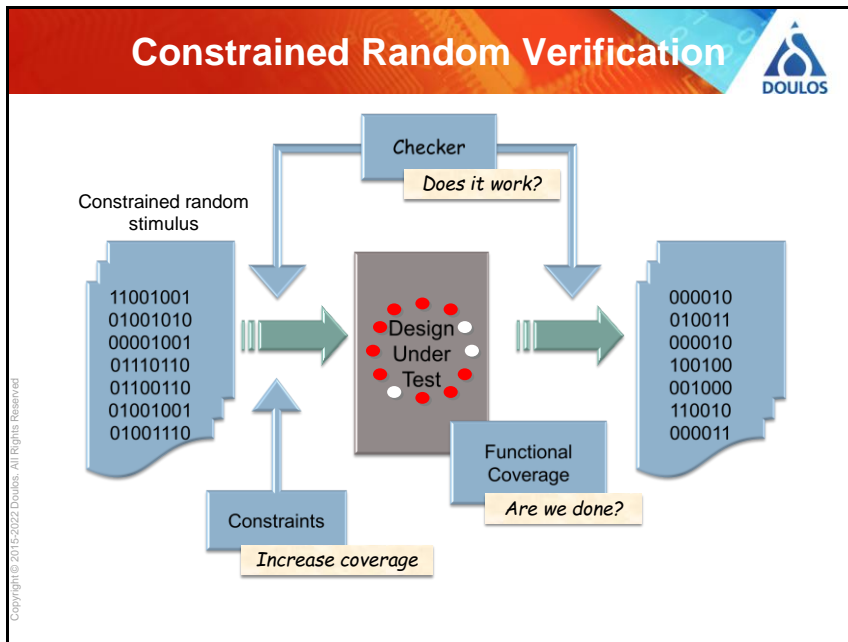
- IEEE Std 1800.2-2017 (Free from IEEE)
- UVM 2017-1.0 Reference Implementation

- IEEE Std 1800.2-2020 (Free from IEEE)
- UVM 2020-1.1 Reference Implementation

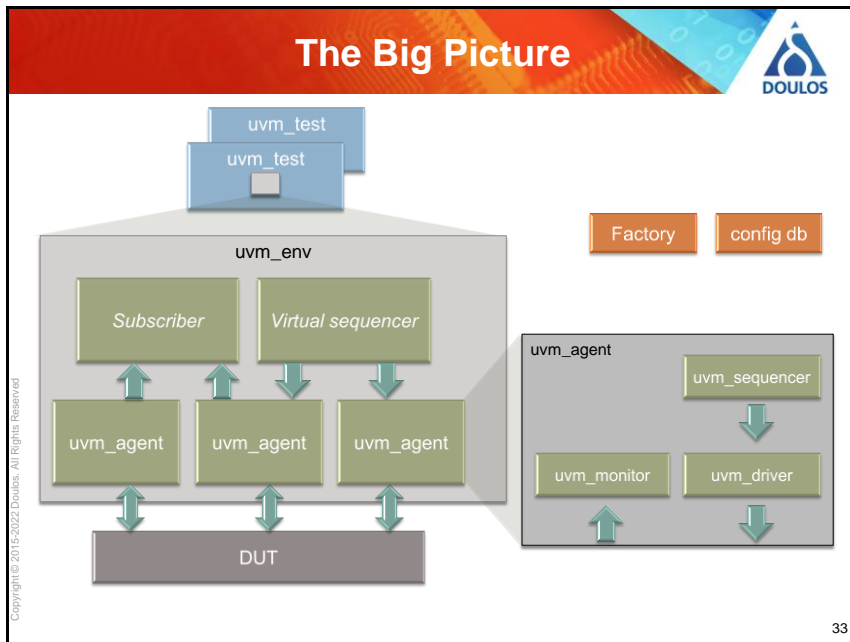
<http://www.accellera.org/downloads/standards/uvm/>
<https://ieeexplore.ieee.org/document/9195920>

Copyright © 2015-2022 Doulos. All Rights Reserved

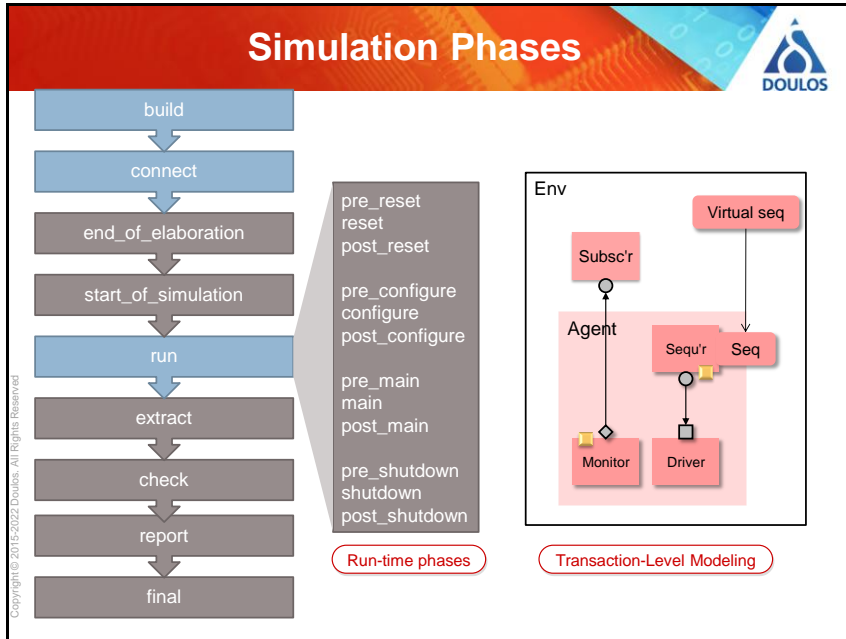
Notes



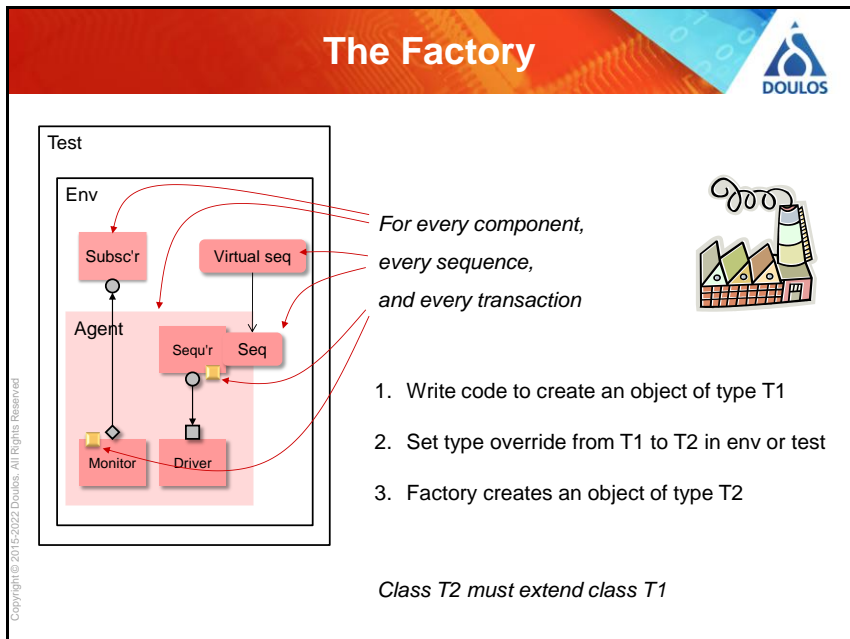
Notes



Notes



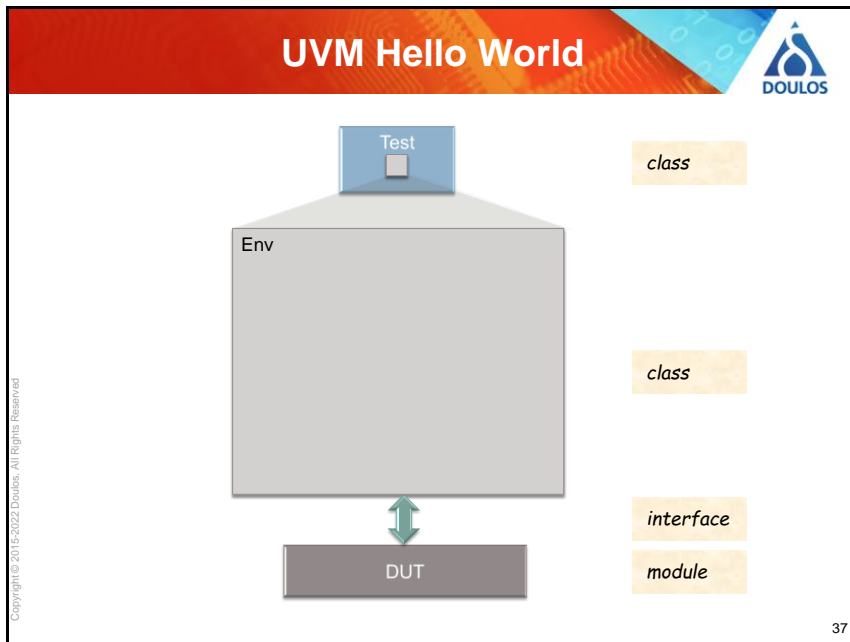
Notes



Getting Started with UVM

- ⇒ • UVM Hello World
- DUT Interface
- Sequencer-Driver Communication

Notes



Notes

Interface and DUT



```
interface dut_if;  
endinterface
```

```
module dut(dut_if dif);  
endmodule
```


```
module top;  
...  
dut_if dut_if1 ();  
dut dut1 ( .dif(dut_if1) );  
...  
endmodule
```

Copyright © 2015-2022 Doulos. All Rights Reserved

38

Notes

The Env



```
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

The Test (1)



```
class my_test extends uvm_test;

  `uvm_component_utils(my_test)

  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction

  my_env m_env;


  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);
    ...
  endtask
endclass
```

UVM Factory

Notes

The Test (2)



```
...  
    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

Classes in a Package



```
`include "uvm_macros.svh"

package my_pkg;

    import uvm_pkg::*;

    class my_env extends uvm_env;
        `uvm_component_utils(my_env)
        ...
    endclass

    class my_test extends uvm_test;
        `uvm_component_utils(my_test)
        ...
    endclass


endpackage
```

Copyright © 2015-2022 Doulos. All Rights Reserved

42

Notes

Running the Test



```
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

Hello World Source Code

```
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
```

```
package my_pkg;
import uvm_pkg::*;

class my_env extends uvm_env;
    `uvm_component_utils(my_env)

    function new(string name, uvm_component parent);
        super.new(name, parent);
    endfunction
endclass

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);
        #10;
        `uvm_info("", "Hello World", UVM_MEDIUM)
        phase.drop_objection(this);
    endtask
endclass
endpackage: my_pkg
```

Copyright © 2015-2022 Doulos. All Rights Reserved

44

Notes

UVM Simulation Output



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 *****



<https://www.edaplayground.com/x/GjxC>

Copyright © 2015-2022 Doulos. All Rights Reserved

45

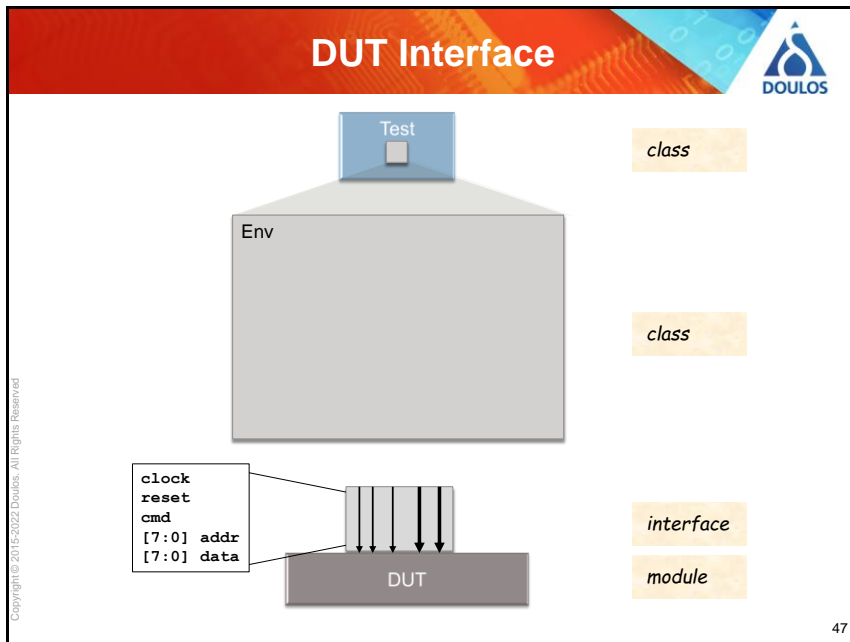
Notes

Getting Started with UVM

- UVM Hello World
- • DUT Interface
- Sequencer-Driver Communication



Notes



Notes

Interface and DUT



```
interface dut_if;
    logic clock, reset;
    logic cmd;
    logic [7:0] addr;
    logic [7:0] data;
endinterface
```

```
module top;
    import uvm_pkg::*;
    import my_pkg::*;

    dut_if dut_if1 ();
    dut dut1 ( .dif(dut_if1) );

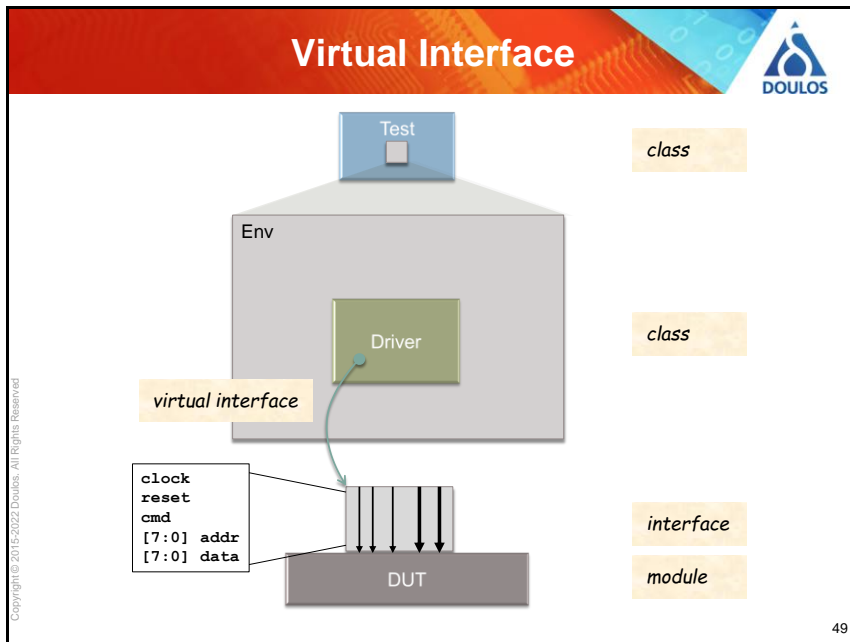
    initial
    begin
        ...
        run_test("my_test");
    end
endmodule
```

```
module dut(dut_if dif);
    import uvm_pkg::*;

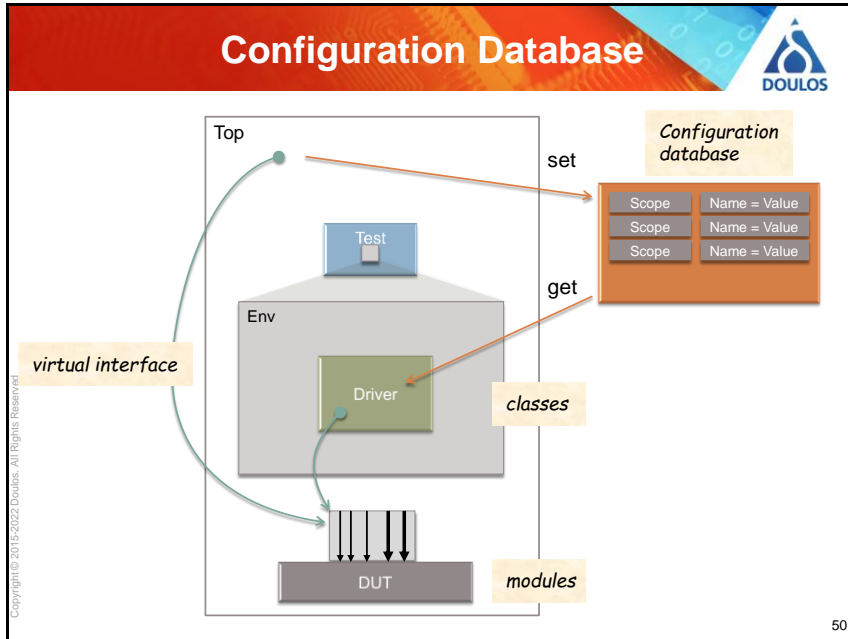
    always @(posedge dif.clock)
    begin
        `uvm_info("", $sformatf("DUT received cmd=%b, addr=%d, data=%d",
            dif.cmd, dif.addr, dif.data), UVM_MEDIUM)
    end
endmodule
```

Dummy implementation

Notes




Notes



Notes

Config Set



Configuration database

```

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

```

Scope	Name = Value
Scope	Name = Value
Scope	Name = Value

Type	Caller	Path	Name	Value
↓	↓	↓	↓	↓

Copyright © 2015-2022 Doulos. All Rights Reserved.

51

Notes

Config Get



```
class my_driver extends uvm_driver;
  `uvm_component_utils(my_driver)
  virtual dut_if dut_vi;

  function new(string name, uvm_component parent);
    super.new(name, parent);
  endfunction

  function void build_phase(uvm_phase phase);

    if ( !
        Type      Caller Path Name Value
        ↓         ↓     ↓   ↓   ↓
        uvm_config_db #(virtual dut_if)::get(this, "", "dut_if", dut_vi)
        )
      `uvm_error("", "uvm_config_db::get failed")


    endfunction
    ...
  endclass
```

Copyright © 2015-2022 Doulos. All Rights Reserved

52

Notes

Clock Generator



```
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

Pin Wiggling



```
class my_driver extends uvm_driver;
    `uvm_component_utils(my_driver)


    virtual dut_if dut_vi;
    ...

    task run_phase(uvm_phase phase);
        forever
            begin
                @(posedge dut_vi.clock);
                dut_vi.cmd <= $urandom;
                dut_vi.addr <= $urandom;
                dut_vi.data <= $urandom;
            end
        endtask
    endclass
```

Wiggle pins of DUT

Notes

The Test



```
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

UVM Simulation Output



```
-----
UVM-1.2.Synopsys
(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 design.sv(17) @ 5: reporter [] DUT received cmd=x, addr= x, data= x
UVM_INFO design.sv(17) @ 15: reporter [] DUT received cmd=1, addr=173, data= 18
UVM_INFO design.sv(17) @ 25: reporter [] DUT received cmd=1, addr= 1, data= 13
UVM_INFO design.sv(17) @ 35: reporter [] DUT received cmd=1, addr=147, data= 75
UVM_INFO design.sv(17) @ 45: reporter [] DUT received cmd=1, addr=245, data= 23
UVM_INFO design.sv(17) @ 55: reporter [] DUT received cmd=1, addr=170, data=131
UVM_INFO design.sv(17) @ 65: reporter [] DUT received cmd=1, addr= 57, data= 54
UVM_INFO design.sv(17) @ 75: reporter [] DUT received cmd=1, addr=226, data= 81
UVM_INFO /apps/vcsmx/vcs/Q-2020.03-SP1-1/etc/uvm-1.2/src/base/uvm_objection.svh(1276) @
80: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
UVM_INFO /apps/vcsmx/vcs/Q-2020.03-SP1-1/etc/uvm-1.2/src/base/uvm_report_server.svh(894) @
80: reporter [UVM/REPORT/SERVER]
--- UVM Report Summary ---
...
[TEST_DONE] 1
```

<https://www.edaplayground.com/x/UDv>

56

Notes

Getting Started with UVM

- UVM Hello World
- DUT Interface
- ➔ • Sequencer-Driver Communication



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
```

Copyright © 2015-2022 Doulos. All Rights Reserved

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
```

Copyright © 2015-2022 Doulos. All Rights Reserved

``uvm_do(req)` creates and sends a randomized transaction to the driver

`req` inherited from `uvm_sequence`

Notes

Sequence versus Sequencer



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

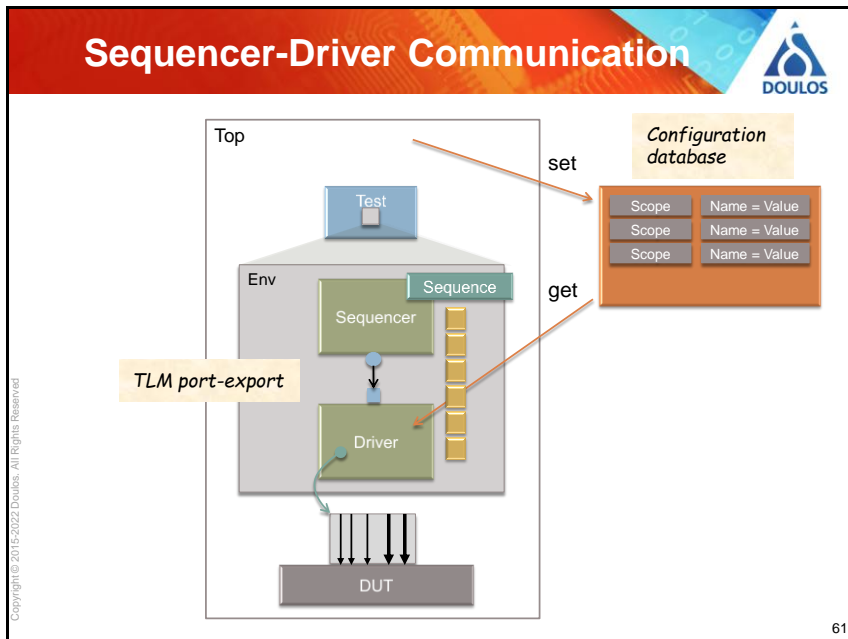
```
typedef uvm_sequencer #(my_transaction) my_sequencer;
```

A sequence runs on a sequencer

```
uvm_sequence extends uvm_sequence_item extends uvm_object  
uvm_sequencer extends uvm_component extends uvm_object
```

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes



Notes

Driver Run Phase



```
class my_driver extends uvm_driver #(my_transaction);  
  
...  
  
task run_phase(uvm_phase phase);  
  forever  
  begin  
    seq_item_port.get(req);  
  
    @(posedge dut_vi.clock);  
    dut_vi.cmd = req.cmd;  
    dut_vi.addr = req.addr;  
    dut_vi.data = req.data;  
  
  end  
endtask  
  
endclass
```

Pull sequence item from sequencer

Drive DUT through virtual interface

Notes

Sequencer-Driver Connection



```
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

Starting the Sequence



```
class my_test extends uvm_test;
  ...
  my_env m_env;
  ...

  task run_phase(uvm_phase phase);
    my_sequence seq;
    seq = my_sequence::type_id::create("seq");

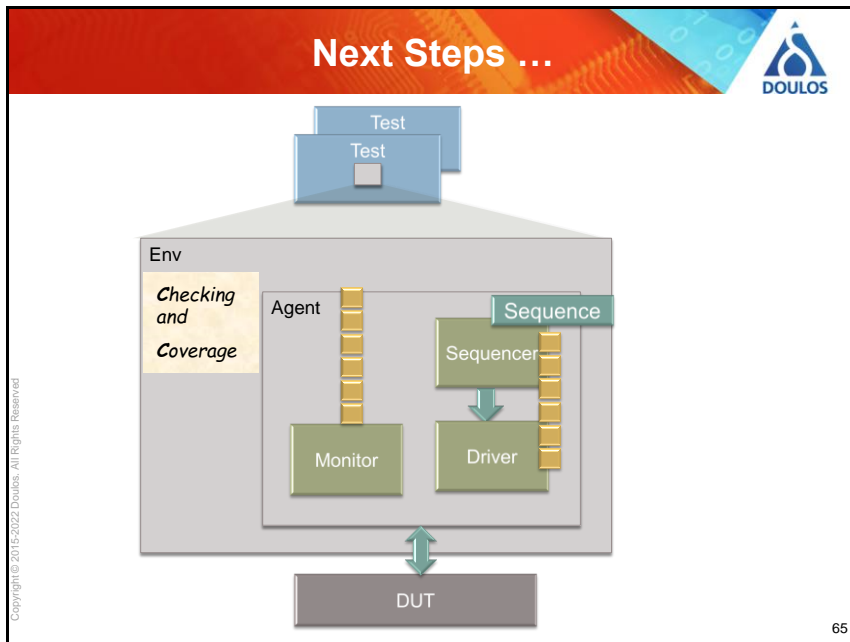
    if( !seq.randomize() )
      `uvm_error("", "Randomize failed")

    seq.set_starting_phase(phase);
    seq.set_automatic_phase_objection(1);

    seq.start( m_env.m_seqr );
  endtask
endclass
```

Run phase ends when all objections have been dropped


Notes



Notes

Doulos – Easier UVM

- Coding guidelines – "One way to do it" Free and open
- Automatic code generator Apache 2.0 license
- Help individuals and project teams
 - learn UVM and avoid pitfalls
 - become productive with UVM (saves ~ 6 weeks)
 - use UVM consistently
- Reduce the burden of supporting UVM code



easier
UVM

https://www.doulos.com/knowhow/sysverilog/uvm/easier_uvm/

Copyright © 2015-2022 Doulos. All Rights Reserved

Notes

DOULOS Delivering KnowHow www.doulos.com

- SoC Design & Verification**
 - » SystemVerilog » UVM
 - » SystemC » TLM-2.0 » Arm
- FPGA & Hardware Design**
 - » VHDL » Verilog » Perl » Tcl
 - » Xilinx » Intel (Altera)
- Embedded Software**
 - » Emb C/C++ » Emb Linux » Yocto
 - » RTOS » Security » Arm Cortex
- Python & Deep Learning**
 -  

SystemVerilog OUT OF THE BOX UVM SYSTEMC LINUX YOCTO RTOS INTEL SECURE ENVELOPE ARM

Notes