

SystemVerilog and UVM for the ABC system verification

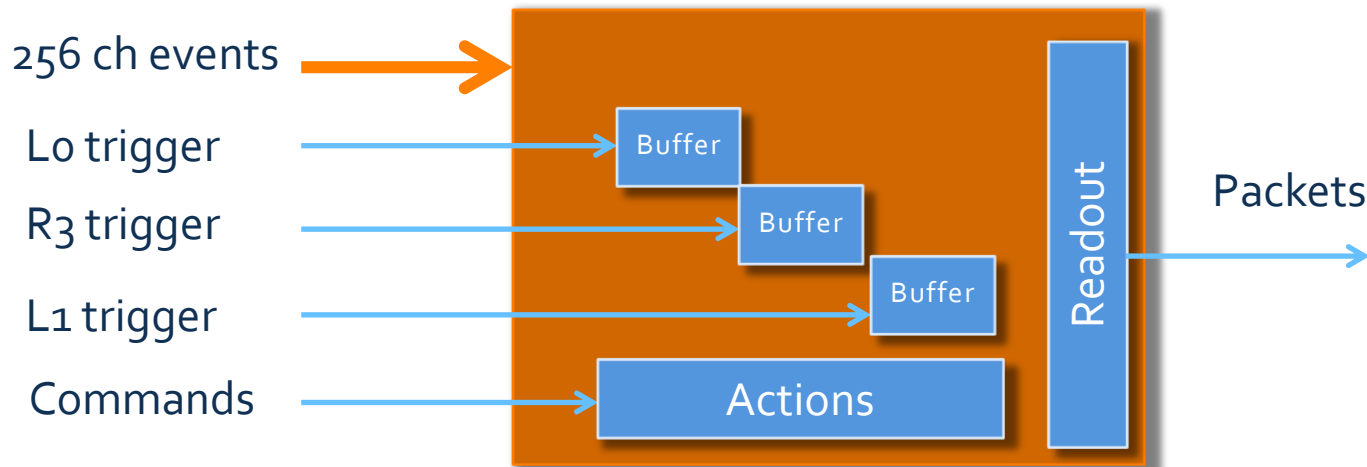
Francis Anghinolfi

OUTLINE

- + The ABC verification environments
- + SystemVerilog and UVM
- + UVM techniques for the ABC system
- + Development plans
- + SystemVerilog for ABC system?

The ABC verification environment

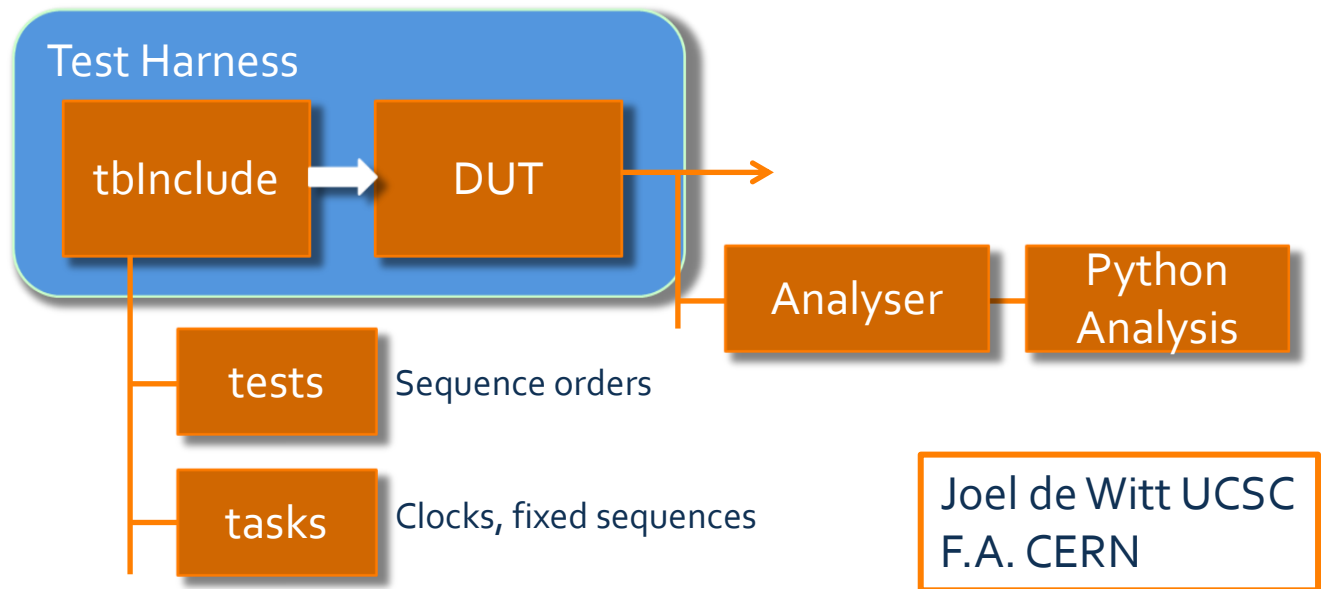
+ What is ABC function (in short)



Verification does (in short) :
Stimulation of hits, triggers, commands
Analysis of packets (in relation to Stimulations)

The ABC verification environment

- + One of the verification setup (verilog only based)



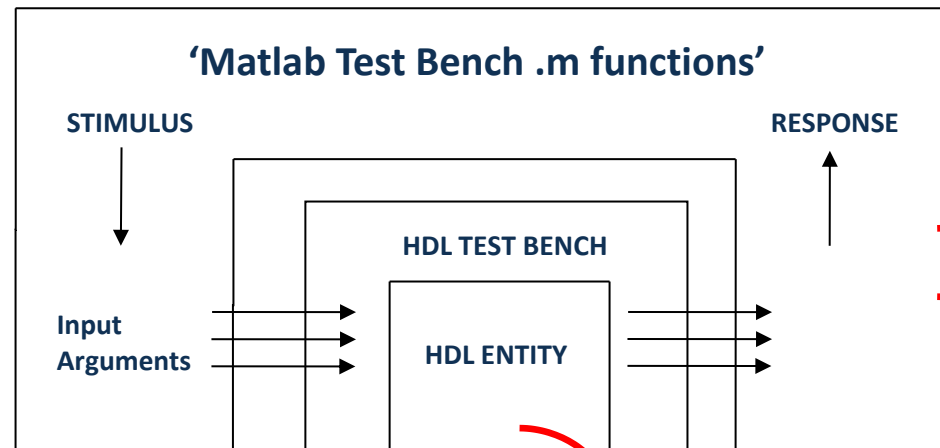
Verification does (in short) :

Stimulation of hits, triggers, commands, precoded time relations

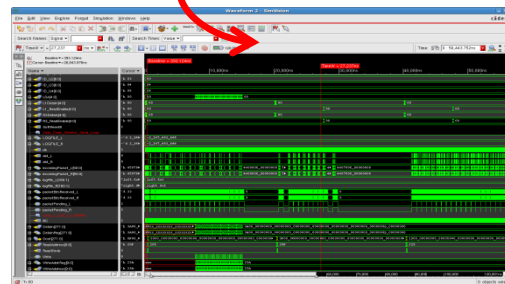
Analysis of packets (in relation to Stimulations)

The ABC verification environment

+ Algorithm Development Using Matlab and Cadence Incisive

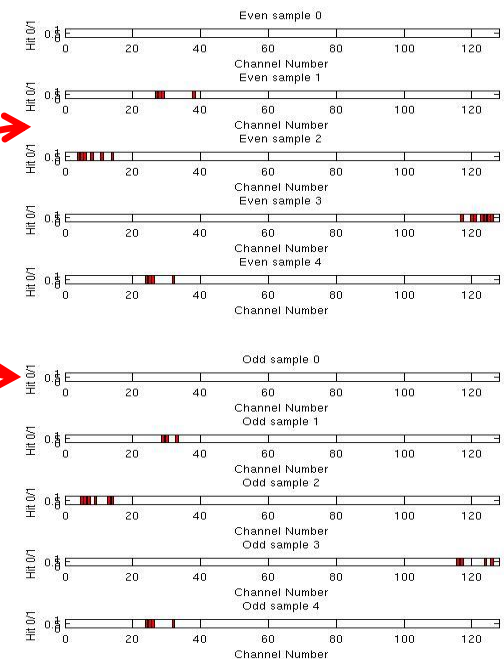


Michelle
Key-Charriere
@ RAL



Cadence Incisive Verilog Simulator

Matlab GUI output

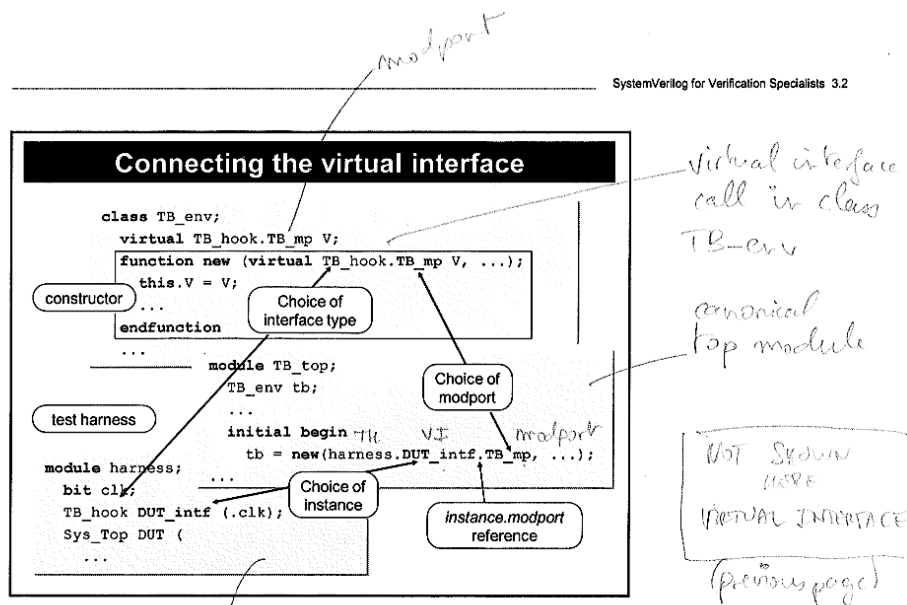


+ Object Oriented Software Trace



SystemVerilog and UVM

+ MY starting point : the SystemVerilog training course



(Sorry Mr. Fitch!
It was a
wonderful
course!)

Finally, we need to arrange that the testbench correctly hooks itself to the appropriate interface and modport when simulation begins. These code fragments show how that can be done.




module harness TH

SystemVerilog and UVM

+ And later on about UVM (from an Accelera course slide)

Studying the documentation is not enough!

↓

- **Certain UVM concepts are not straightforward** 
- **SV/UVM terminology can be a barrier** 
- **The sheer size of the UVM BCL** ?????
- **Too much choice**  ← **Backward compatibility with legacy**
- **Lack of recommended practice and naming conventions** YES !

SystemVerilog and UVM

- + At least I have seen the interest of **THIS** feature in SV/UVM :



\$RANDOM !

- ➡ In the spirit of SV, this has to do with test & functionality coverage, through generation of random data and address sets.
- ➡ For exp. systems the feature becomes naturally useful as experiments have to deal with random (physics) data AND random triggers time distributions ((with constraints ☺)...)

SystemVerilog and UVM

- + So generating random physics data set is an easy trick

```
rand int unsigned hit;  
constraint Hits (hit dist {[0,255]});
```

transacti
on

```
for (int i=0;i<256;i++)  
    begin  
        if (i == hit) hitbus[i] = 1;  
        else  
            hitbus[i] = 0;  
    end
```

driver

SystemVerilog and UVM

+ What about getting a fix pattern data ?

```
rand bit [57:0] como;  
constraint busyo {como[7:0] dist {[0:255]}; }  
constraint busy1 {como[15:8] dist {[0:255]}; }
```

transaction

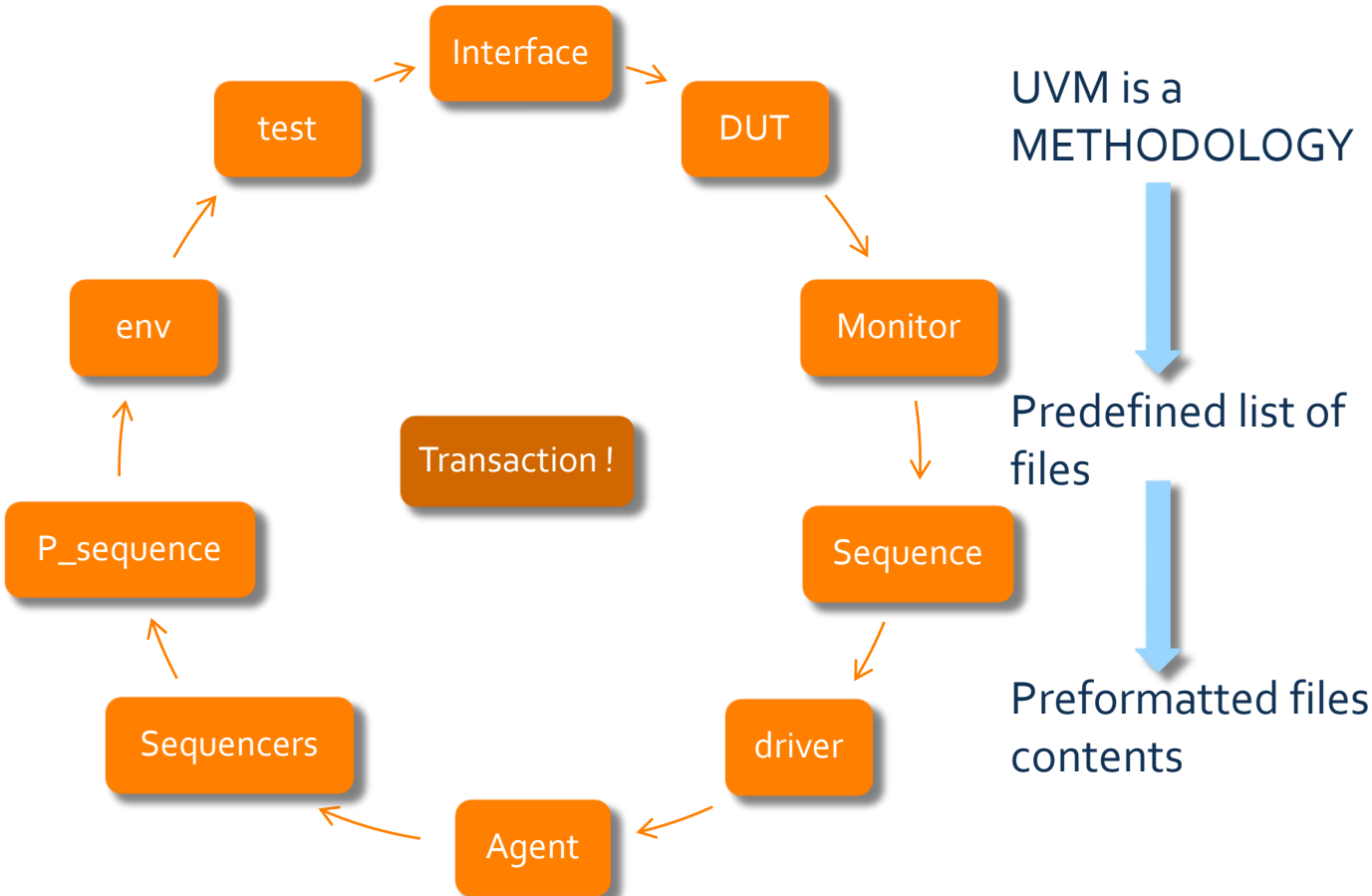
sequence
r

```
Data = { 4'h3, 4'ho, 4'ho, 4'ho, 4'h1, 4'hf, 3'ho, LEFT, 4'h1};  
'uvm_do_with (req, {como[57:0] == {HEADER, HCCField, HCCID,  
ABCID, RegAdress, WRITE, Data}; start_data < 100;})
```

Como is 58 bits word : $2^{58} = 288230376151711744$, seems beyond SV limits

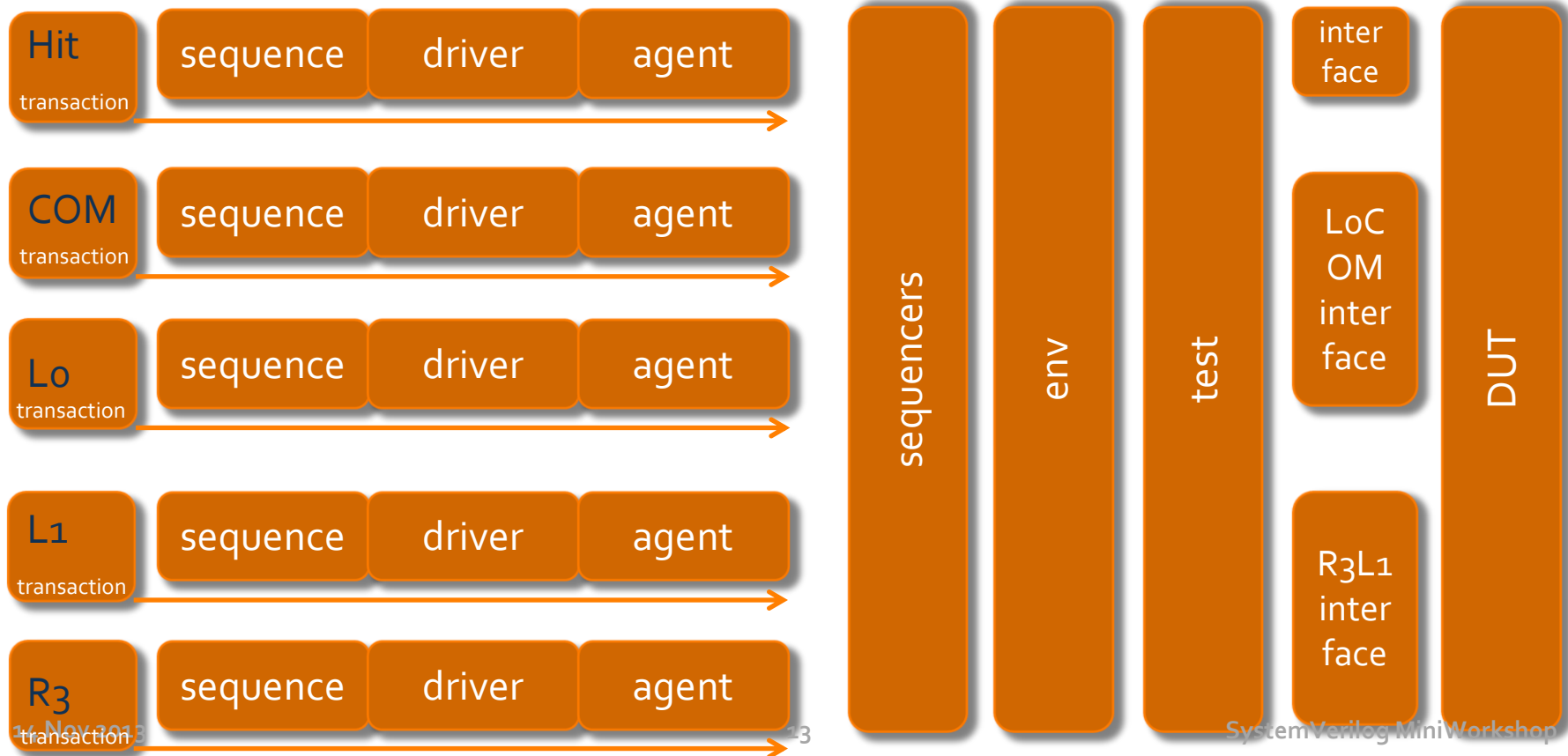
UVM for the ABC verification

What appeared is that UVM is a sort of wrapper formalism for SystemVerilog



UVM for the ABC verification

+ For ABC the interest is in running parallel transactions



UVM for the ABC verification

ABC_top.sv (module)

```
HitBus_if    si (clk);
ABCDriverCOMLo_if    comLo_si (clk);
ABCDriverR3L1_if    R3L1_si (clk);
```

ABC_test1.sv (class)

```
task main_phase(uvm_phase phase);

    phase.raise_objection(this);
    begin
        // create and start the virtual sequence
        virtual_seq vseq;
        vseq = virtual_seq::type_id::create();
        vseq.start(m_env.m_virtual_seqr);
    end
    phase.drop_objection(this);
endtask: main_phase
```

ABC_env.sv (class)

```
m_virtual_seqr.HitBus_seqr =
    m_HitBus_agent.m_sequencer;
m_virtual_seqr.COM_seqr =
    m_COM_agent.m_sequencer;
m_virtual_seqr.Lo_seqr =
    Lo_agent.m_sequencer;
m_virtual_seqr.L1_seqr =
    L1_agent.m_sequencer;
```

UVM for the ABC verification

Lo_agent.sv (class)

```
virtual ABCDriveCOMLo_if vif;  
:  
:  
m_driver.Lo_si = vif;  
m_driver.seq_item_port.connect
```

Interface decl.

Seq. to Drive connection

Same (similar) class definitions for the 3 other transactions (L1, com, Hit)

```
(m_sequencer.seq_item_export);  
Lo_Driver.sv (class)
```

```
virtual ABCDriveCOMLo_if.senderLo Lo_si;  
:  
virtual task drive_Lo_trans (ABCDriverLo_transaction tlo);  
    drive_Lo (tlo.Lo, 2*bit_period*tlo.Lo_dist);  
endtask : drive_Lo_trans  
:  
seq_item_port.get(tlo); // Get  
drive_Lo_trans(tlo); // and Drive
```

Transaction, interval

Get and Drive

UVM for the ABC verification

L1_sequence.sv (class)

```
task body; // Does L1 selection within count
```

```
for (int i=0; i<=count;i++)
```

Number of L1 sequences

```
begin
```

```
L1val = L1val+1;
```

```
$display ("L1val = \t %b, count = \t %b", L1val[7:0], i);
```

```
`uvm_do_with (req, {L1C[10:0] == {HEADER, L1val} ; L1_dist = $dist_exponential(1,400);})
```

```
end
```

11bits sequence, 3 bits Header, 8 bits binary count number,
Interval btw. sequences with exponential distribution

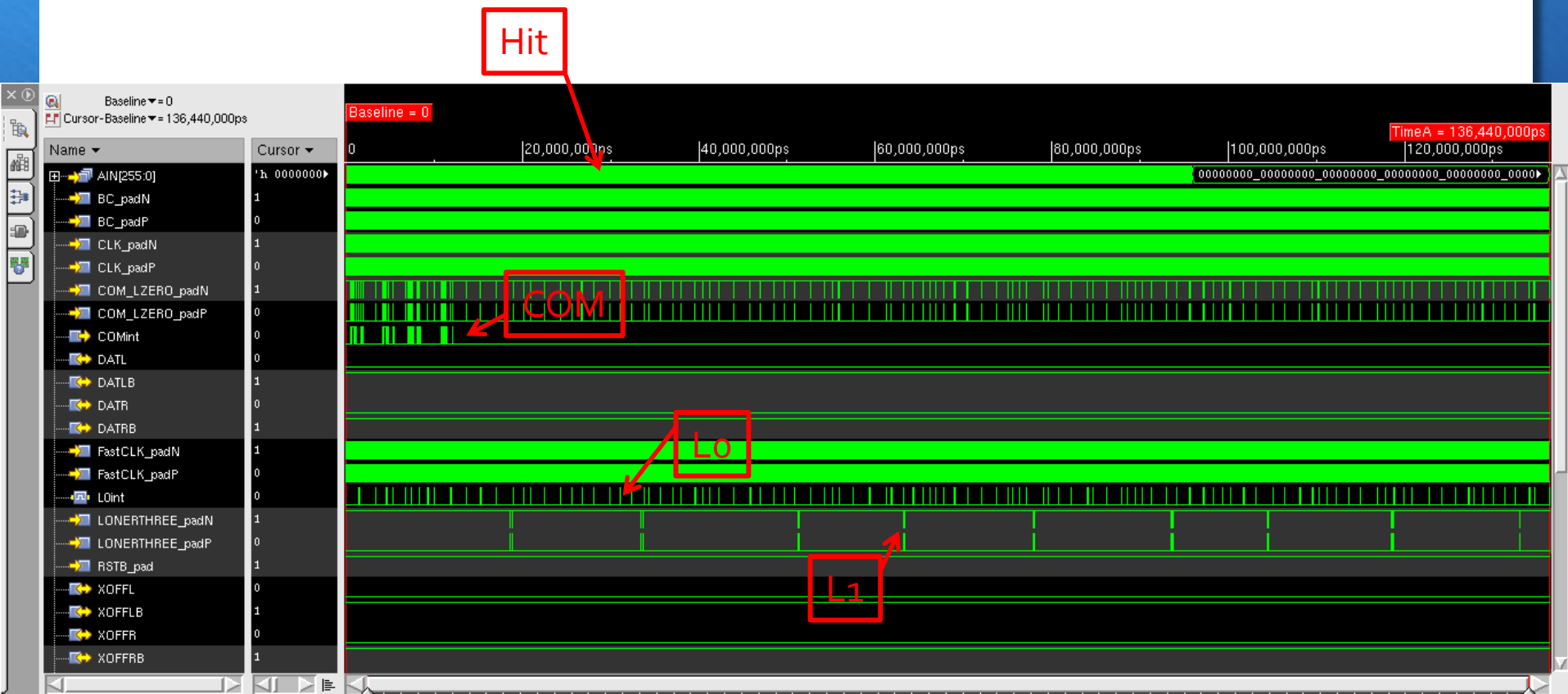
UVM for the ABC verification

L1_transaction.sv (class)

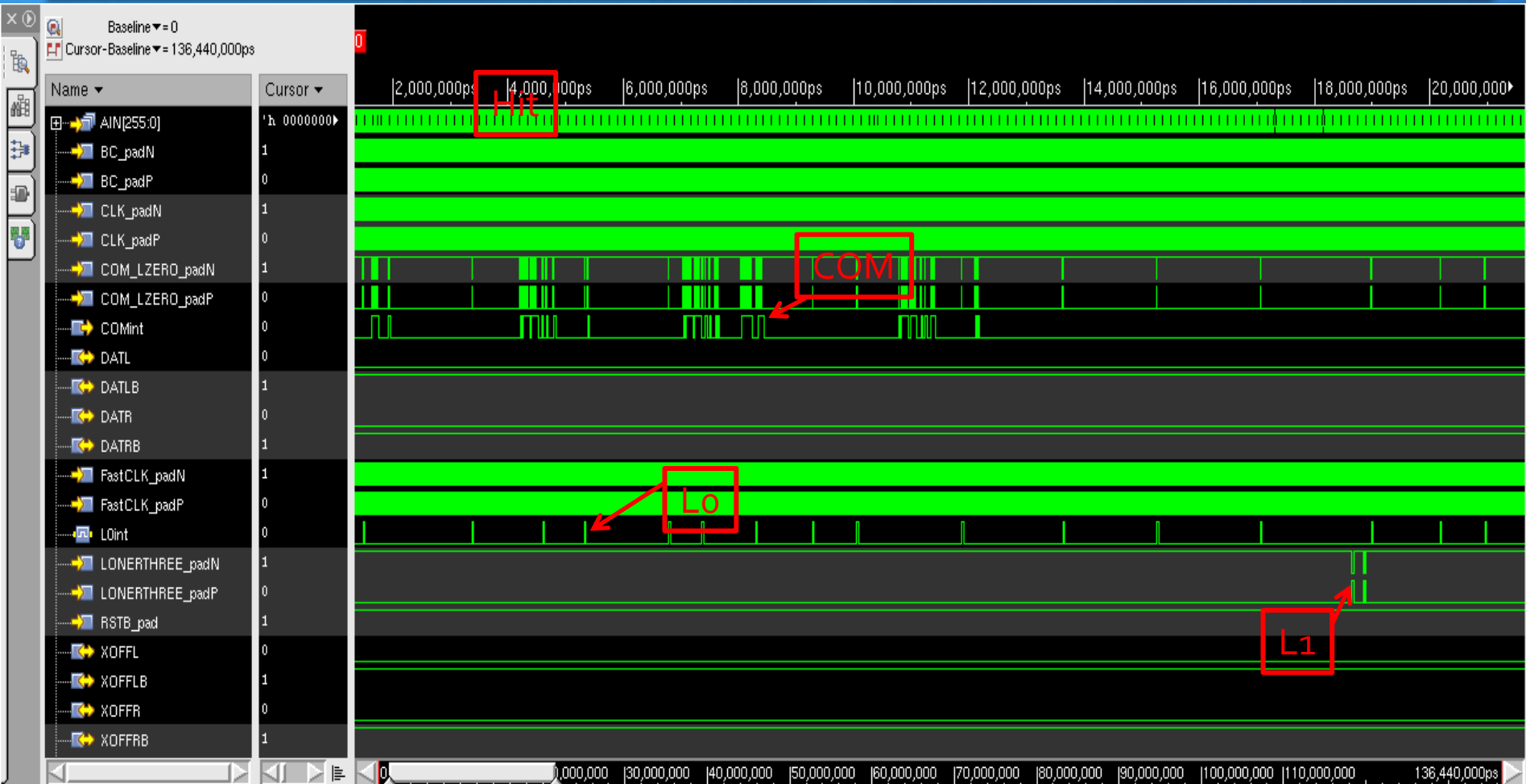
```
rand bit [10:0] L1C;  
constraint busy9 {L1C dist {[0:2047]}; }  
  
rand bit pick_L1; // Select First Level Trigger signal  
constraint busy10 {pick_L1 dist {0 :/ 9, 1 :/ 1}; } // 10% at 1  
  
rand int unsigned L1_dist;  
constraint val1 {L1_dist dist {[200:400]}; }
```

Variables declarations
with constraints

UVM for the ABC verification



UVM for the ABC verification



Development plans

- + Develop the control & input sequences close to real case
- + Join with the existing setup for ABC+HCC system, developed at RAL by M. Key-Charriere
- + Explore the random nature of stimuli to verify/validate the response of the ABC/HCC to false commands, bit errors, disordered triggers etc ...

SV/UVM for ABC system ?

- + Could I have described the same sequences with std. verilog ?
- + My experience : a veeery looooong training process
- + UVM methodology saved me : I think I could not do the job with SystemVerilog without the UVM formalism
- + The monitor/checker options did not look so attractive, however I did not do much work there
- + Pay off for the effort ?