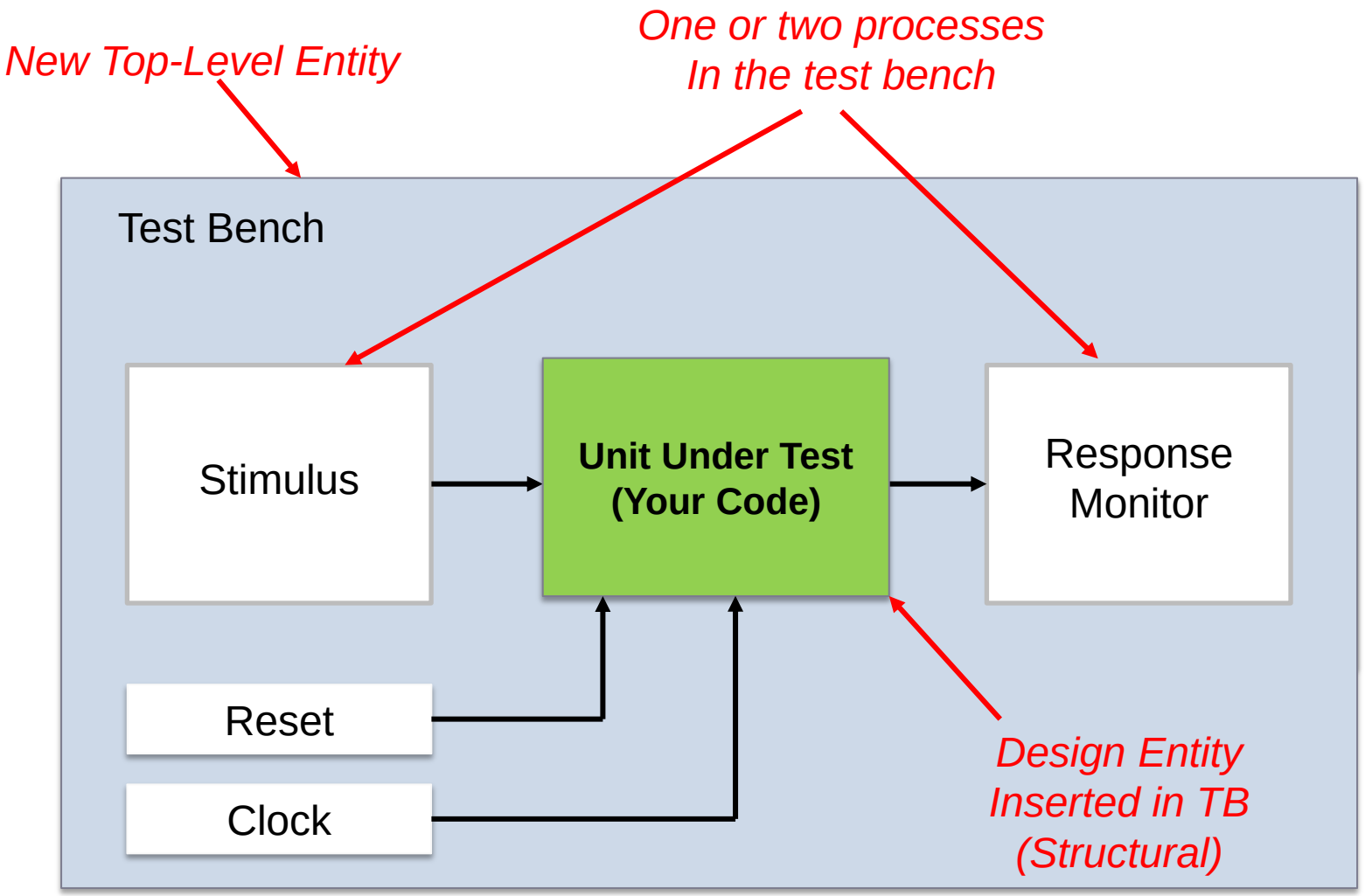
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

PETER HØGH MIKKELSEN
29. JANUARY, 2015

SEQUENTIAL TEST BENCHES

TEST BENCH BASICS



CLOCK GENERATION

```
-- Clock Generator with
-- adj phase & duty cycle

genClk: process
  begin
    clk <= 0;
    wait for phase;
    loop
      clk <= 1;
      wait for time_high;
      clk <= 0;
      wait for time_low;
    end loop
end process addProc;
```

```
-- Basic concurrent
-- Clock Generator

signal clk : std_logic := '1';
signal clk2 : std_logic := '1';
  ...
  constant period : time := 20 ns;
begin  -- waveform

  clk <= not clk after period/2;

  clk2 <= not clk2 after period/2
          when end_sim = false
          else unaffected;
```

SEQUENTIAL TESTBENCH

```
entity anderClocked is
  port (
    clk      : in  std_logic;
    reset    : in  std_logic;
    inA      : in  std_logic;
    inB      : in  std_logic;
    anderOut : out std_logic);
end anderClocked;

architecture add of anderClocked is

begin  -- ander
addProc: process (clk, reset)
  begin  -- process addProc
    if reset = '0' then
      anderOut <= '0';
    elsif rising_edge(clk) then
      anderOut <= inA and inB;
    end if;
  end process addProc;
end add;
```

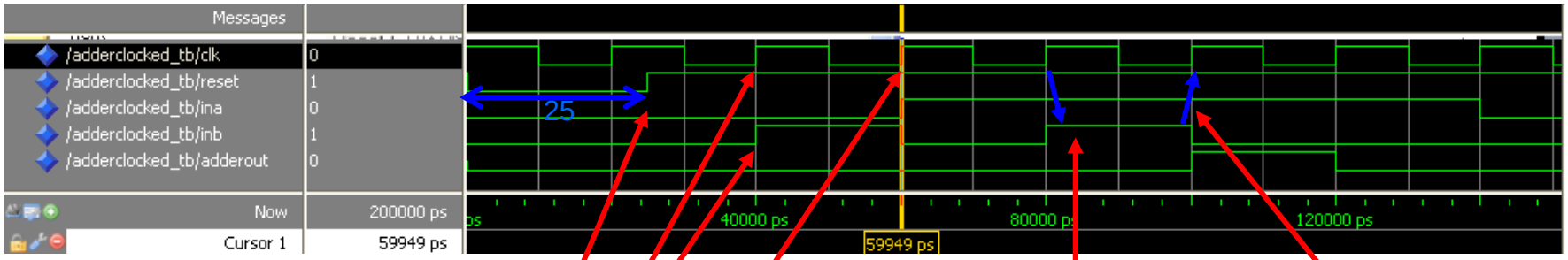
```
signal clk : std_logic := '1';
  ...
  constant period : time := 20 ns;

begin  -- waveform
  UUT: anderClocked
    port map (clk => clk, ... );

  clk <= not clk after period/2;
  reset <= '0', '1' after 25 ns;

  WaveGen_Proc: process
  begin
    wait until reset = '1';
    wait until clk = '1';

    inA <= '0';
    inB <= '1';
    wait for period;
    inA <= 1;
    ...
```

STIMULUS SYNCHRONIZATION

```
wait until clk = '1';
assert count = "01" report "missing 01" severity error;

...

wait until clk = '1';
wait for 1 ns;
assert count = "01" report "missing 01" severity error;
```

› Remember to wait until signal has settled, before testing it.

SEQ. TB WAVEFORM

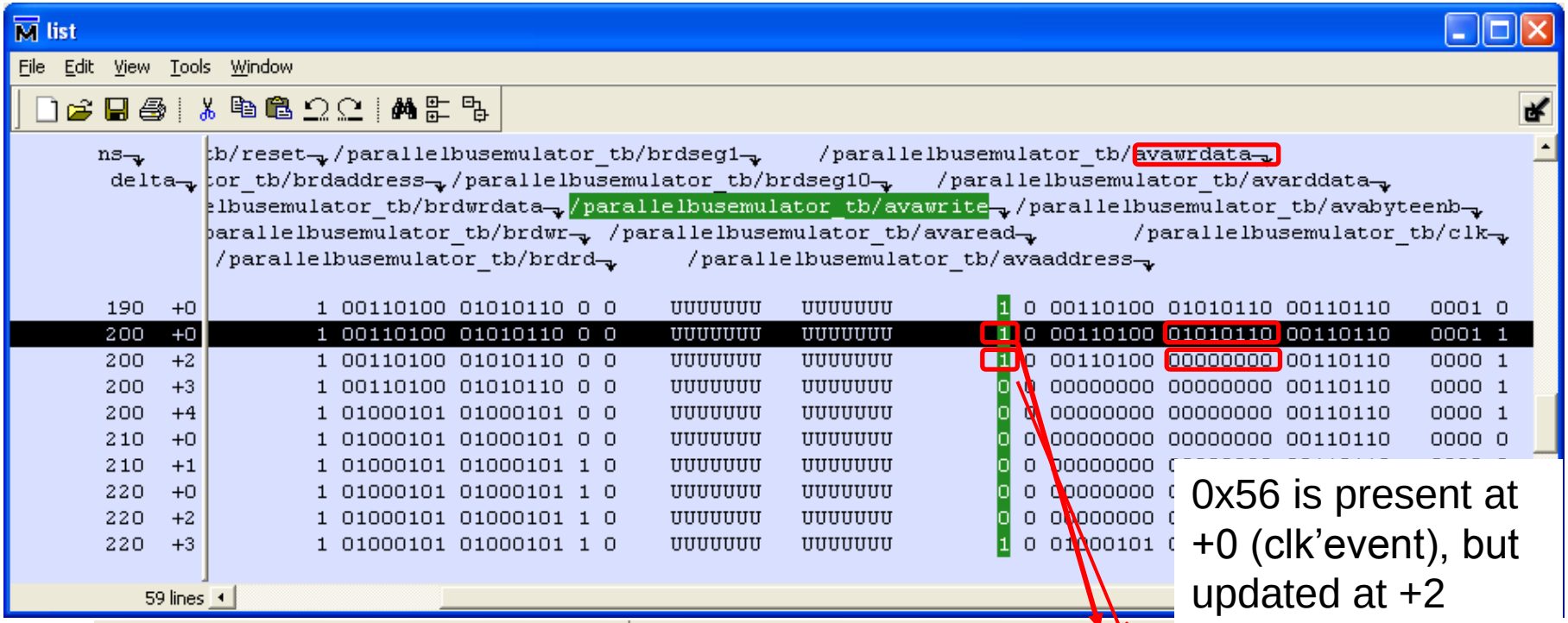



```
WaveGen_Proc: process
 begin
   wait until reset = '1';
   wait until clk = '1';

   inA <= '0';
   inB <= '1';
   wait for period;

   inA <= '1';
   inB <= '0';
   wait for period;
```

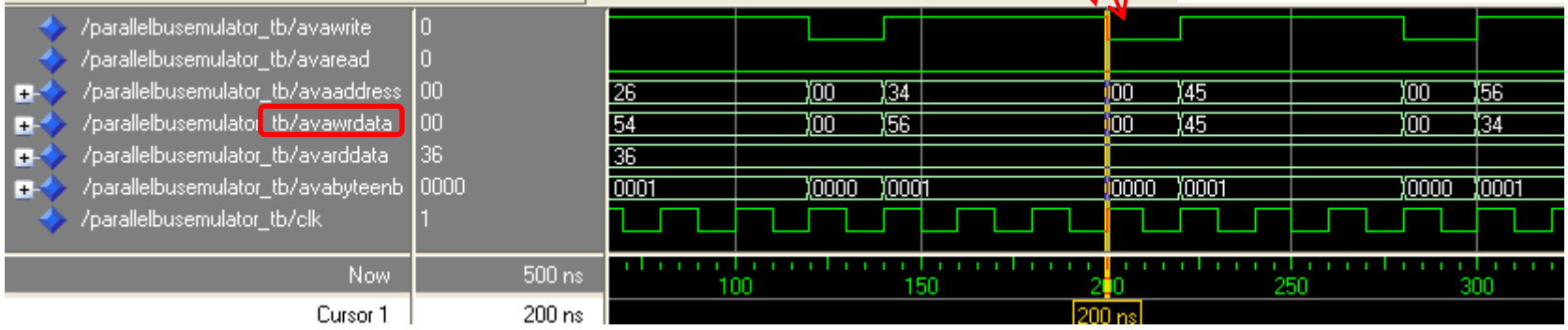
Signal is read by UUT at clk edge (Δ0)

Signal is set in TB at clk edge (Δ+1)

MODELSIM LIST WINDOW



0x56 is present at +0 (clk'event), but updated at +2



STIMULUS PROCEDURES

```
-- purpose: Simulus Generator
procedure doAnd (
  constant inp : in std_logic_vector(1 downto 0);
  signal clk : in std_logic;
  signal inA : out std_logic;
  signal inB : out std_logic) is
begin  -- doAnd
  inA <= inp(0);
  inB <= inp(1);
  wait until clk = '1';
  wait for 2 ns;

end doAnd;
```

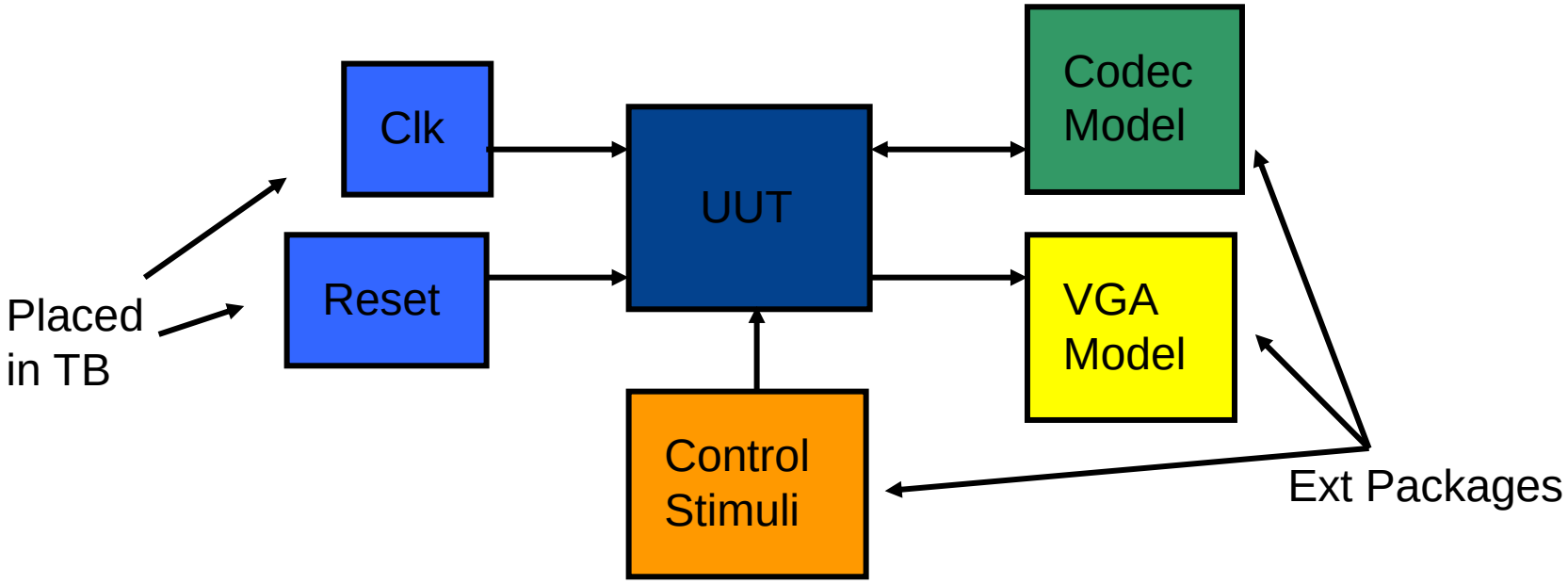
```
-- Stimuli Process
...
doAnd("10", clk, inA, inB);
doAnd("11", clk, inA, inB);
doAnd("01", clk, inA, inB);
...
```

› Can only be called from processes with NO sensitivity list

RESPONSE MONITOR

```
monitorAnder: process
variable a, b : std_logic;
begin
  a := inA;
  b := inB;                                -- Latch inputs
  wait until clk = '1';                    -- Pos clk edge
  wait for 1 ns;
  assert anderOut = (a and b)
    report "addition error: " & std_logic'image(a) &
    " and " & std_logic'image(b) & " != " &
    std_logic'image(anderOut) severity error;

end process monitorAnder;
```

SIMULATION PACKAGES



› Packages for external models simplifies design and eases re-use
› Models for external components may be available

SIMULATION PACKAGE EXAMPLE

```
package andTester is
  constant : thold : time= 2 ns;
  procedure doAnd (
    constant inp : in std_logic_vector(1 downto 0);
    signal clk : std_logic;
    signal inA : out std_logic;
    signal inB : out std_logic);
end andTester;

package body andTester is
procedure doAnd (...) is
  begin  -- doAnd
    inA <= inp(0);
    inB <= inp(1);
    wait until clk = '1';
    wait for thold;
  end doAnd;
end andTester;
```

```
library ieee;
use ieee.std_logic_1164.all;
use work.andTester.all;
...
 WaveGen_Proc: process
  begin
    ...
    doAnd("11", clk, inA, inB);
```

RECORDS IN PROCEDURE CALLS

```
 type AvalonMmBus_type is record
    csi_clockreset_clk  : std_logic;
    avs_s1_writedata : std_logic_vector(7 downto 0);
    avs_s1_readdata : std_logic_vector(7 downto 0);
  end record;

 procedure MonAvalonRdWr (
    constant NbrCs     : in integer;
    signal AvalonBus   : in AvalonMmBus_type);
...
  assert AvalonBus.avs_s1_read = '0'
         report "Read Set during Write cycle";
```

- Records ease the transfer of signals through hierarchal designs and makes them more robust to changes (Think: A class)
- NOT supported by SOPC tools (Qsys/SOPC Builder)!!!