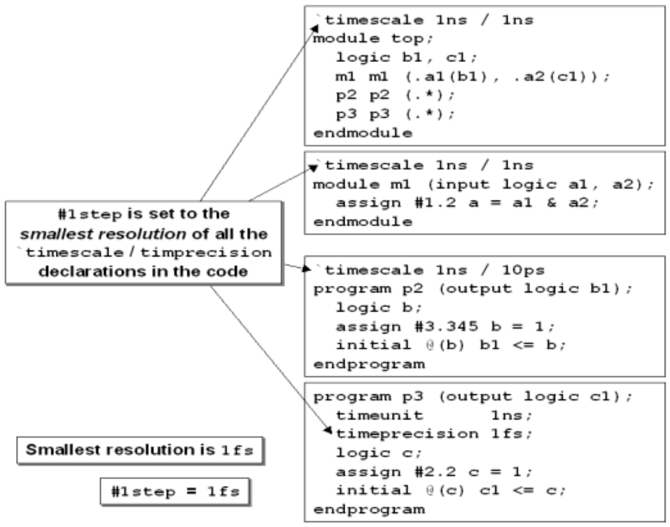
SystemVerilog event regions

Monday, November 19, 2018 11:45 AM

http://www.sunburst-design.com/papers/CummingsSNUG2006Boston_SystemVerilog_Events.pdf

#1step - what it really means??

That is, this is the smallest unit of time for which the simulator will schedule an event – there can be no activity in between #1step delays.



```
`timescale 1ns / 1ns
module top;
  logic b1, c1;
  m1 m1 (.a1(b1), .a2(c1));
  p2 p2 (.*);
  p3 p3 (.*);
endmodule
```

```
`timescale 1ns / 1ns
module m1 (input logic a1, a2);
  assign #1.2 a = a1 & a2;
endmodule
```

```
`timescale 1ns / 10ps
program p2 (output logic b1);
  logic b;
  assign #3.345 b = 1;
  initial @(b) b1 <= b;
endprogram
```

```
program p3 (output logic c1);
  timeunit      1ns;
  timeprecision 1fs;
  logic c;
  assign #2.2 c = 1;
  initial @(c) c1 <= c;
endprogram
```

#1step is set to the *smallest resolution* of all the `timescale / timprecision declarations in the code
Smallest resolution is 1fs
#1step = 1fs

System verilog event regions.

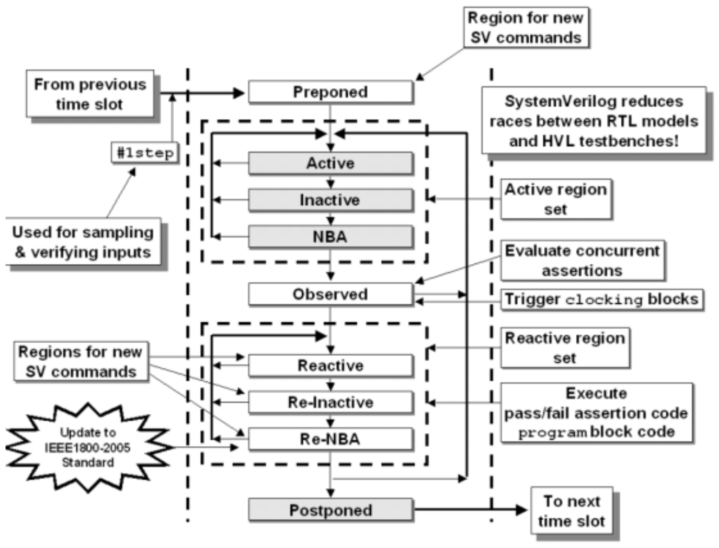

Region for new SV commands
From previous time slot
Preponed
SystemVerilog reduces races between RTL models and HVL testbenches!
#1step
Active
Inactive
NBA
Active region set
Used for sampling & verifying inputs
Evaluate concurrent assertions
Observed
Trigger clocking blocks
Regions for new SV commands
Reactive
Reactive region set
Re-Inactive
Re-NBA
Execute pass/fail assertion code program block code
Update to IEEE1800-2005 Standard
Postponed
To next time slot

Figure 5 - SystemVerilog-2005 event regions with PLI regions omitted

Regions that are designed to implement correct RTL functionality:
• Active regions (Active, Inactive and NBA regions - but avoid Inactive region events).

Regions that are designed to implement correct verification execution:
• Preponed, Reactive regions (Reactive, Re-Inactive, Re-NBA) and Postponed regions.

Regions that are designed to implement concurrent assertion checking:
• Preponed, Observed, and Reactive regions.

Region that should be avoided:
• Inactive region.

Certainly verification code can be executed in the regions devoted to RTL functionality. Prior to the implementation of SystemVerilog, engineers doing Verilog-2001 verification implemented clever methods to ensure race-free operation between RTL designs and the pseudo-RTL testbenches. The new SystemVerilog regions just make it easier to implement a race-free verification environment.

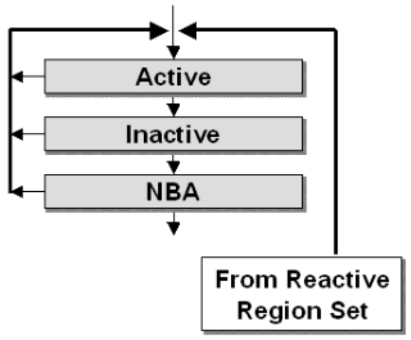
1. **Preponed Events Region**

   The stated function of this region is to sample values that are used by concurrent assertions. The Preponed region is executed only once in each time slot, immediately after advancing simulation time (there is no feedback path to re-execute the Preponed region).

   "The values of variables used in assertions are sampled in the Preponed region of a time slot, and the assertions are evaluated during the Observed region. ..."

2. **The Active Region Set (Active - Inactive - NBA)**

   The active region set is used to schedule blocking assignments and nonblocking assignments included in module code. Any task or function called from a module is also scheduled into the active region set. The intended purpose of the active region set is to schedule RTL and behavioral code activity. Testbench code can also be written as module code and indeed preexisting Verilog testbench code was written as module code, but SystemVerilog users are encouraged to place future testbench code into pr



   a. **Active Events Region**

      The Active Events Region, also commonly called the Active region, is part of the Active Region Set.

      • Execute all module blocking assignments.
      • Evaluate the Right-Hand-Side (RHS) of all nonblocking assignments and schedule updates into the NBA region.
      • Execute all module continuous assignments
      • Evaluate inputs and update outputs of Verilog primitives.
      • Execute the $display and $finish commands.

   b. **Inactive Events Region**

      The Inactive Events Region, also commonly called the Inactive region, is part of the Active Region Set. This region is where #0 blocking assignments are scheduled.

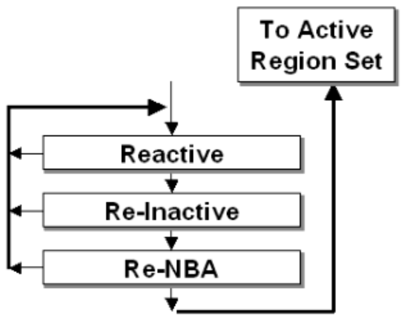
      Most engineers that continue to use #0 assignments are trying to defeat a race condition that might exist in their code due to assignments made to the same variable (The variable is read in one always block and updated the same variable in another always block at the same time)from more than one always block, which is a violation of Sunburst Design Race Avoidance Guideline

   c. **Nonblocking Assignment Events Region**

      The Nonblocking Assignment Events Region, also commonly called the NBA region, is part of the Active Region Set. In this region, RHS of all the non-blocking assignments are updated with the LHS evaluated in active event region .

3. **Observed region**
   - Observed region is designed to evaluate concurrent assertions using the values sampled in the preponed region.
   - Clocking blocks are triggered.
   - This region just evaluate pass or fail conditions of concurrent assertions, it doesn't executes the code written inside pass or fail block.
   - A process associated with the pass and fail code is scheduled inside the Reactive regions. This is because concurrent assertions are designed to behave strictly as monitors, thus, they are not allowed to modify the state of the design

4. **The Reactive Region Set (Reactive - Re-Inactive -Re-NBA).**



The reactive region set is used to schedule blocking assignments, #0 blocking assignments and nonblocking assignments included in program code. Any task or function called from a program is also scheduled into the reactive set event regions. The intended purpose of the reactive region set is to schedule testbench stimulus drivers and testbench verification checking in the same time slot after RTL code has settled to a semi-steady state.

Why the program block has been introduced?.

As it is explained above all the test bench related code is executed in reactive regions if it is written inside the program block, this way we can isolate RTL and the test bench code and avoid race conditions between RTL and test bench.

a. **Reactive Events Region**

The Reactive region is the reactive region set dual of the corresponding Active region in the same time slot

- **Execute all program blocking assignments.**
- **Execute the pass/fail code from concurrent assertions.**
- **Evaluate the Right-Hand-Side (RHS) of all program nonblocking assignments and schedule updates into the Re-NBA region.**
- **Execute all program continuous assignments**
- **Execute the $exit and implicit $exit commands.**

The reactive region events are scheduled towards the end of the time slot to ensure the followings.

- **The signals monitored/sampled by the test bench code is stable before it samples. Since all the signals which are driven by DUT are evaluated in the active regions, so they can be monitored in the reactive regions by program block without any race .**
- **It also ensures, the signals driven by the test bench to the DUT are stable enough before it is actually read by the DUT**

5. **Postponed Region**

This region is designed to execute the $strobe and $monitor commands that will show the final updated values for the current time slot. This region is also used to collect functional coverage for items that use strobe sampling.

Let's consider a simple example to understand how the events are scheduled in the different regions.

```
`timescale 1ns/100ps
module test;
  logic [7:0] d;

  initial begin
       d = 8'h33;
    #5 d = 8'hAA;
  end
endmodule
```

Example 2 - Simple test used to show sampling and #1step sample delays

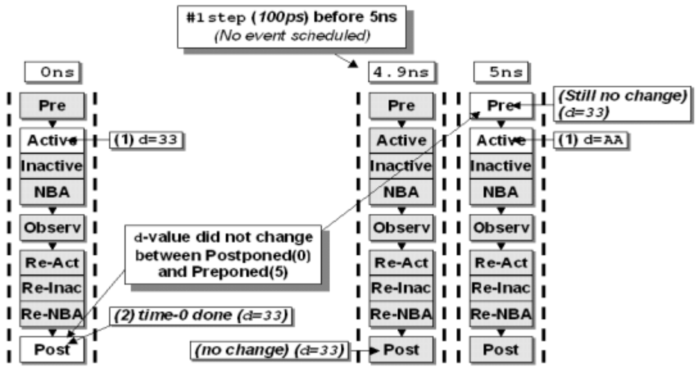


Figure 11 - How events are scheduled and sampled for the code in Example 2

What is the best or when to apply stimulus in verilog and system verilog??.

- **Verilog-2001 designs and testbenches**

  The Sunburst Design Verilog testbench methodology typically assigned stimulus on the inactive clock edge (The clock edge which is not used by the DUT), typically far away from setup and hold times for the gate-level devices. This permitted the use of efficient blocking assignments and the only activity after application of stimulus on the inactive clock edge would be input combinational logic activity.

- **System Verilog designs and testbenches**

  There are two ways to drive stimulus if the test bench is coded in system verilog.

  a. Drive stimulus on inactive clock edge similar to verilog (This method is not highly recommended over clocking block)
  b. Use the clocking blocks supported by system verilog to have configurable skew values for input and output ports of the testbench.

```
clocking ram @(posedge clk);
   input  #1 dout;
   output  #1 din,addr,ce,we;
endclocking
```

In the above example, input and output skews are defined such that, inputs are sampled by test bench 1ns before the active clock edge and all the outputs (stimulus) from test bench are driven 1 ns after the clock edge.