Proposed modifications to section 9.3 shown below

9.3 The stratified event scheduler

A compliant SystemVerilog simulator must maintain some form of data structure that allows events to be dynamically scheduled, executed, and removed as the simulator advances through time. The data structure is normally implemented as a time-ordered set of linked lists, which are divided and subdivided in a well defined manner.

The first division is by time. Every event has one and only one simulation execution time, which at any given point during simulation can be the current time or some future time. All scheduled events at a specific time define a time slot. Simulation proceeds by executing and removing all events in the current simulation time slot before moving on to the next nonempty time slot, in time order. This procedure guarantees that the simulator never goes backwards in time.

A time slot is divided into a set of ordered regions:

a) Preponed
b) Pre-active
c) Active
d) Inactive
e) Pre-NBA
f) NBA
g) Post-NBA
h) Observed
i) Post-observed
j) Reactive
k) Re-inactive
l) Pre-re-NBA
m) Re-NBA
n) Post-re-NBA
o) Pre-postponed
p) Postponed

The purpose of dividing a time slot into these ordered regions is to provide predictable interactions between the design and testbench code.

~~Except for the Observed, Reactive, and Re-inactive and the Post-observed PLI region, these~~ The Pre-active, Active, Inactive, Pre-NBA, NBA, Post-NBA, Pre-postponed and Postponed regions essentially encompass the IEEE 1364 reference model for simulation, with exactly the same level of determinism. In other words, legacy Verilog code shall continue to run correctly without modification within the new mechanism. The Postponed region is where the monitoring of signals, and other similar events, take~~s~~ place. No new value changes are allowed to happen in the time slot once the Postponed region is reached.

The Observed, Reactive, ~~and~~ Re-inactive, and Re-NBA regions are new in this standard, and events are only scheduled into these new regions from new language constructs.

The Observed region is for the evaluation of the property expressions when they are triggered. A criterion for this determinism is that the property evaluations must only occur once in any clock triggering time slot. During the property evaluation, pass/fail code shall be scheduled in the Reactive region of the current time slot. PLI callbacks are not allowed in the Observed region.

The new **`#1step`** sampling delay provides the ability to sample data immediately before entering the current time slot and is a preferred construct over other equivalent constructs because it allows the **`1step`** time delay to be parameterized. This **`#1step`** construct is a conceptual mechanism that provides a method for defining when sampling takes place and does not require that an event be created in this previous time slot. Conceptually, this **`#1step`** sampling is identical to taking the data samples in the Preponed region of the current time slot.

The code specified in the program block and the pass/fail code from property expressions are scheduled in the Reactive region. A **`#0`** control delay specified in a program block schedules the process for resumption in the Re-inactive region. The Re-inactive region is the program block dual of the Inactive region (see below). A nonblocking assignment specified in a program block schedules the process for resumption in the Re-NBA region. The Re-NBA region is the program block dual of the NBA region (see below).

The Pre-active, Pre-NBA, and Post-NBA regions are new in this standard but support existing PLI callbacks.

The Pre-re-NBA, Post-re-NBA and Post-observed regions are ~~is~~ new in this standard and have ~~has~~ been added for PLI support.

The Pre-active region provides for a PLI callback control point that allows PLI application routines to read and write values and create events before events in the Active region are evaluated (see 9.4).

The Pre-NBA region provides for a PLI callback control point that allows PLI application routines to read and write values and create events before the events in the NBA region are evaluated (see 9.4).

The Post-NBA region provides for a PLI callback control point that allows PLI application routines to read and write values and create events after the events in the NBA region are evaluated (see 9.4).

The Post-observed region provides for a PLI callback control point that allows PLI application routines to read values after properties are evaluated (in Observed or earlier region).

The Pre-re-NBA region provides for a PLI callback control point that allows PLI application routines to read and write values and create events before the events in the Re-NBA region are evaluated (see 9.4).

The Post-re-NBA region provides for a PLI callback control point that allows PLI application routines to read and write values and create events after the events in the Re-NBA region are evaluated (see 9.4).

NOTE—The PLI currently does not schedule callbacks in the Post-observed region.

The Pre-postponed region provides a PLI callback control point that allows PLI application routines to read and write values and create events after processing all other regions except the Postponed region. The flow of execution of the event regions is specified in Figure 9-1.

The Active, Inactive, Pre-NBA, NBA, Post-NBA, Observed, Post-observed, Reactive, Re-inactive, Pre-re-NBA, Re-NBA, Post-re-NBA, and Pre-postponed regions are known as the *iterative* regions.

The Preponed region provides for a PLI callback control point that allows PLI application routines to access data at the current time slot before any net or variable has changed state. Within this region, it is illegal to write values to any net or variable or to schedule an event in any other region within the current time slot.

NOTE—The PLI currently does not schedule callbacks in the Preponed region.

The Active region holds current events being evaluated and can be processed in any order.

The Inactive region holds the events to be evaluated after all the active events are processed.

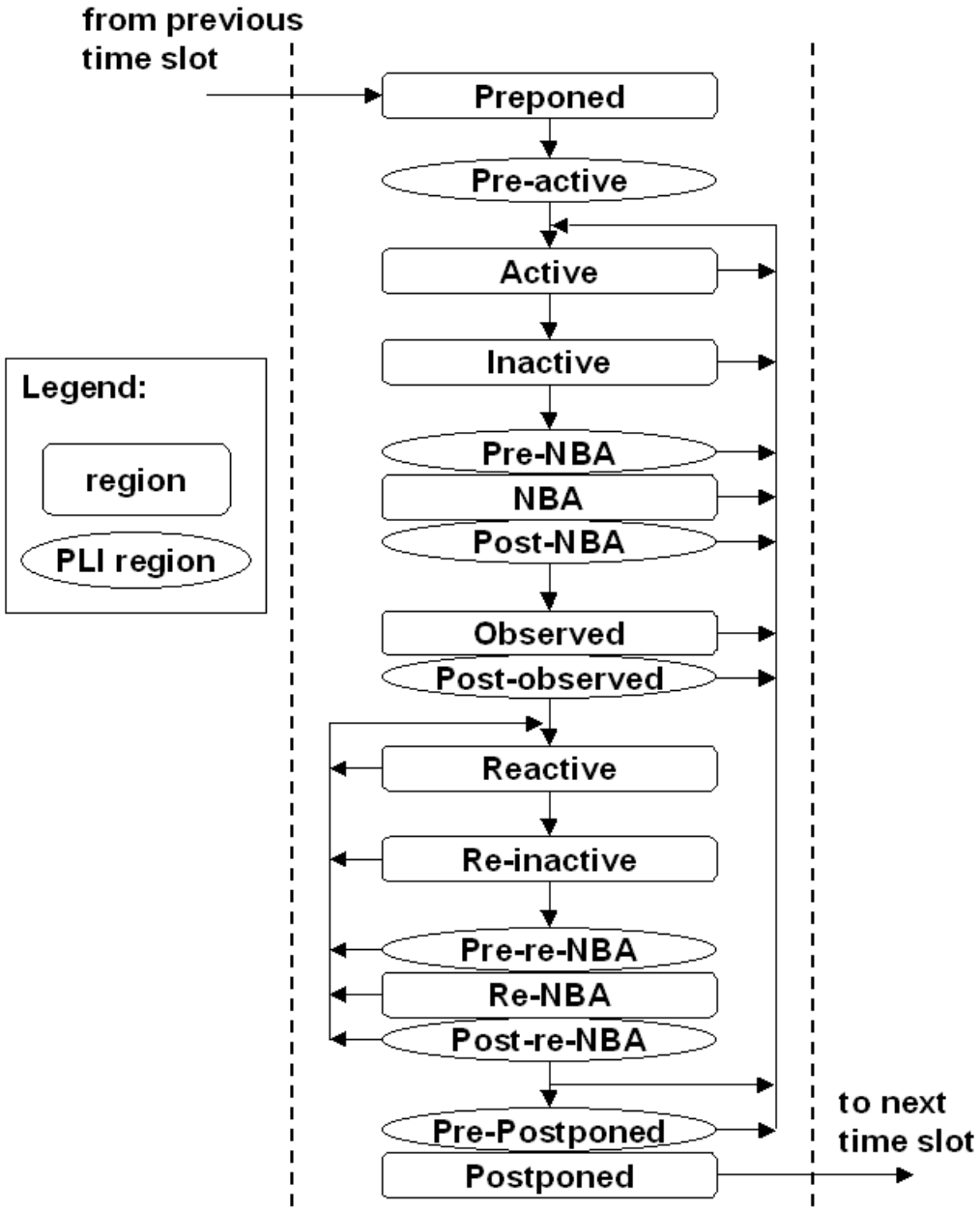
An *explicit* #0 delay control requires that the process be suspended and an event scheduled into the Inactive region (or Re-inactive for program blocks) of the current time slot so that the process can be resumed in the next inactive to active iteration.

A nonblocking assignment creates an event in the NBA region (or Re-NBA for program blocks), scheduled for the current or a later simulation time.

The Postponed region provides for a PLI callback control point that allows PLI application routines to be suspended until after all the Active, Inactive, NBA, Observed, Reactive, ~~and~~ Re-inactive, and Re-NBA regions have completed.

Within this region, it is illegal to write values to any net or variable or to schedule an event in any previous region within the current time slot.

<NEW FIG 9.1> (Added Pre-re-NBA, Re-NBA, Post-re-NBA regions to figure)



9.3.1 The SystemVerilog simulation reference algorithm

```
execute_simulation {
  T = 0;
  initialize the values of all nets and variables;
  schedule all initialization events into time 0 slot;
  while (some time slot is nonempty) {
    move to the next future nonempty time slot and set T;
    execute_time_slot (T);
  }
}

execute_time_slot {
  execute_region (preponed);
  execute_region (pre-active);
  while (any region in [active ... pre-postponed] is nonempty) {
    while (any region in [active ... post-observed] is nonempty) {
      execute_region (active);
      R = first nonempty region in [active ... post-observed];
      if (R is nonempty)
        move events in R to the active region;
    }
    while (any region in [reactive ... ~~re-inactive~~ Post-re-NBA] is nonempty) {
      execute_region (reactive);
      R = first nonempty region in [reactive ... ~~re-inactive~~ Post-re-NBA];
      if (R is nonempty)
        move events in R to the reactive region;
    }
    if (all regions in [active … ~~re-inactive~~ Post-re-NBA] are empty)
      execute_region (pre-postponed);
  }
  execute_region (postponed);
}

execute_region {
  while (region is nonempty) {
    E = any event from region;
    remove E from the region;
    if (E is an update event) {
      update the modified object;
      evaluate processes sensitive to the object and possibly schedule
      further events for execution;
    } else { /* E is an evaluation event */
      evaluate the process associated with the event and possibly
      schedule further events for execution;
    }
  }
}
```

The Iterative regions and their order are Active, Inactive, Pre-NBA, NBA, Post-NBA, Observed, Post-observed, Reactive, Re-inactive, Pre-re-NBA, Re-NBA, Post-re-NBA and Pre-postponed. As shown in the algorithm, once the Reactive, ~~or~~ Re-Inactive, Pre-re-NBA, Re-NBA or Post-re-NBA regions are processed, iteration over the other regions does not resume until these ~~two~~ five regions are empty.