- **Preponed** Current time slot before any net or variable has changed state.
- Pre-active Support for PLI callbacks.
- **Active** Blocking assignments and immediate assertions are executed in any order. NBA assignments are evaluated, but not executed.
- **Inactive** Events to be evaluated after all the active events are processed, such as blocking assignments with a #0 delay and deferred assertions.
- Pre-NBA Support for PLI callbacks
- **NBA** Nonblocking assignment region using evaluated values.
- Post-NBA Support for PLI callbacks
- Pre-Observed Support for PLI callbacks
- **Observed** Evaluation of the property expressions when they are triggered. Evaluation of concurrent assertions and queued action blocks for procedural concurrent assertions evaluated (see 4.3.4).
- Post-observed Support for PLI callbacks
- **Reactive** Action block evaluations
- Re-Inactive; Pre-Re-NBA; Re-NBA; Post-Re-NBA; Pre-Postponed
- **Postponed** Support for PLI callbacks

All scheduled events at a specific time form an event queue at that time slot. Simulation proceeds by executing and removing all events in the event queue in the current simulation time slot before moving on to the next non-empty time slot, in time order. Execution of events in the queue may re-trigger an earlier time slot. Figure 4.1-1 summarizes the SystemVerilog flow of time slots and event regions using an example.

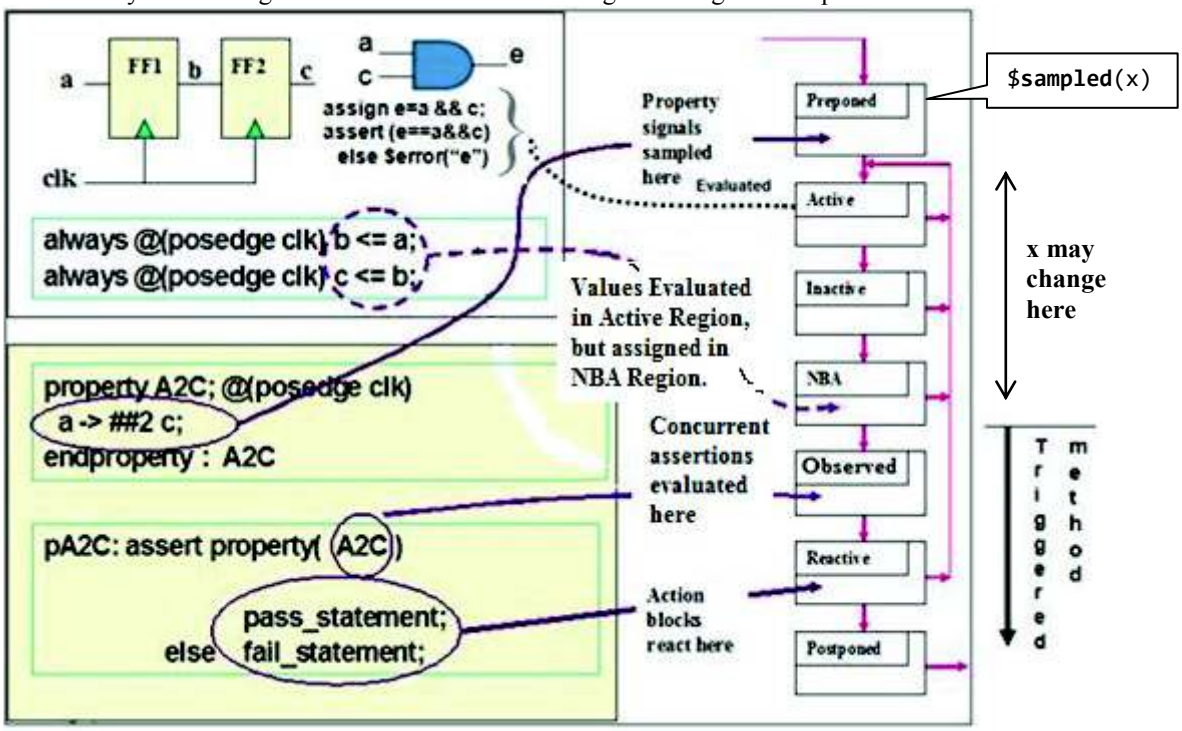

Figure 4.1-1 The SystemVerilog flow of time slots and event regions
(For immediate deferred assertions, please see 4.6.2).

SystemVerilog also allows for the specification of input and output skews. Input (or inout) signals are sampled at the designated clock event. If an input skew is specified then the signal is sampled at *skew* setup time units *before* the clock event. Similarly, output (or inout) signals are driven *skew* hold simulation time units *after* the corresponding clock event. An example of a skew definition is shown below:

```
clocking master_clock @(posedge clk);
  input #1ps address;  // setup time
  input #5 output #6 data;  // setup and hold time
endclocking : master_clock
```