Verilog Scheduling and Event Queue

Consider

```verilog
always_ff @(posedge clk) c = reset ? 0 : c + 1;
always_ff @(posedge clk) over_th = c + 1'd1 > threshold;
```

Is `over_th` computed using the new or old `c`?

(Answer: either one, and so code is unreliable.)
Terminology

**Event:**
Sort of a to-do item for simulator. May include running a bit of Verilog code or updating an object’s value.

**Event Queue:**
Sort of a to-do list for simulator. It is divided into time slots and time slot regions.

**Time Slot:**
A section of the event queue in which all events have the same time stamp.

**Time Slot Region:**
A subdivision of a time slot. There are many of these. Important ones: active, inactive, NBA.

**Scheduling:**
Determining when an event should execute. The when consists of a time slot and a time slot region.

**Update Events:**
The changing of an object’s value. Will cause *sensitive* objects to be scheduled.
Time Slot Regions

Rationale:

“Do it now!” is too vague. Need to prioritize.

SystemVerilog divides a time slot into 17 regions.

Some Regions

**Active Region:**
Events that the simulator is currently working on. Only the current time slot has this region.

**Inactive Region:**
Contains normally scheduled events. Current and future time slots have this region.
(1) Start this time slot.

(2) Execute an active event.

(3a) Schedule new events this time slot.

(3b) Schedule new events in future time slots.

(4) When active region empty, bulk move events.

(5) When all regions in this time slot empty advance to next time slot.
### Event Queue Example

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 0$</td>
<td>$t = 1$</td>
<td>$t = 1$</td>
<td>$t = 1$</td>
<td>$t = 1$</td>
</tr>
<tr>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>active</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
</tr>
<tr>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
<td>inactive</td>
</tr>
<tr>
<td>i-a.0</td>
<td>i-b.0</td>
<td>i-a.1</td>
<td>i-b.0</td>
<td>i-a.0</td>
<td>i-b.0</td>
<td>i-a.1</td>
<td>i-b.0</td>
<td>i-a.0</td>
<td>i-b.0</td>
<td>i-a.1</td>
<td>i-b.0</td>
<td>i-a.1</td>
</tr>
</tbody>
</table>

1: Verilog puts all **initial** blocks in $t = 0$’s inactive region.

2: Active region is empty, and so inactive copied to active.

3: Event **i-a.0** executes and schedules event **c** for $t = 0$ \ldots and **i-a.1** for $t = 3$.

4: Event **i-a.0** removed from active region (it is now not scheduled anywhere).

```verilog
initial begin
    // i-a.0
    a = 1;
    #3;
    // i-a.1
    a = 2;
end

initial begin
    // i-b.0
    b = 10;
    #1;
    // i-b.1
    b = a;
end

assign c = a + b; // c
```
5,6: Event \(i-b.0\) executes and schedules \(i-b.1\) for \(t = 1\).

7,8: Since active region is empty, inactive region is bulk-copied to active region.

9: Event \(c\) executes.

10-12: Since all regions in time slot 0 are empty, move to next time slot, \(t = 1\).

initial begin
  // i-a.0
  a = 1;
  #3;
  // i-a.1
  a = 2;
end

initial begin
  // i-b.0
  b = 10;
  #1;
  // i-b.1
  b = a;
end

assign c = a + b;  // c
Some More Regions

**NBA Region:**
Update events from non-blocking assignments.

**Postponed Region:**
Events scheduled using $\texttt{watch}$ system task.
Event Scheduling

*Time-Delay Scheduled*

Scheduled by a delay: \( \texttt{#4 \ a = b;} \)

Put in inactive region of a future time step.

*Sensitivity List Scheduled*

Explicit event: \( \texttt{@( \ a ), @( \ posedge \ clk ), wait( \ stop\_raining \ )} \)

Continuous assignment: \( \texttt{assign \ x = a + b;} \).

Module or primitive ports: \( \texttt{and \ myAndGate(x,a,b)} \).

Put in inactive region of current time step.

*Update Events*

Non-blocking assignment: \( \texttt{y <= a + b;} \).

Put in NBA region of current time step.
Permanently Scheduled

Watch lists: \texttt{\$watch(a)}.

Put in postponed region of every time step.
(1) Start this time slot.

(2) Execute an active event.

(3a) Schedule new events in this time slot.

(4) When active region empty, bulk move events from first non-empty region.

(5) When all regions in this time slot empty advance to next time slot.

(3b) Schedule new events in future time slots.

Time slot $t=7$

Time slot $t=9$

Time slot $t=12$
Time slot $t=7$

(1) Start this time slot.

**Active**
- Event 9
- Event 1

Execute

START

(2) Execute an active event.

SCHED

DONE

Inactive
- Event 2

NBA
- Event 12

Inactive and NBA regions are empty.

Time slot $t=19$

(1) Start this time slot.

Schedule new events in future time slots.

**Postponed**
- Watch Event 9
- Watch Event 1

Execute

START

(2) Execute a postponed event.

DONE

Initiate next time slot.
(1) Start this time slot.

**Active**
- Event 9
- Event 1

(2) Execute an active event.

**Inactive**
- Event 2

**NBA**
- Event 12

(1) Start this time slot.

**Reactive**
- Event 79
- Event 81

(2) Execute a reactive event.

**Re-Inactive**
- Event 712

**Re-NBA**
- Event 122

(2) Execute a postponed event.

Schedule new events in future time slots.

Initiate next time slot.

Execute

START

SCHED

DONE

Execute

START

SCHED

DONE