

## Verilog Scheduling and Event Queue

Consider

```
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*.

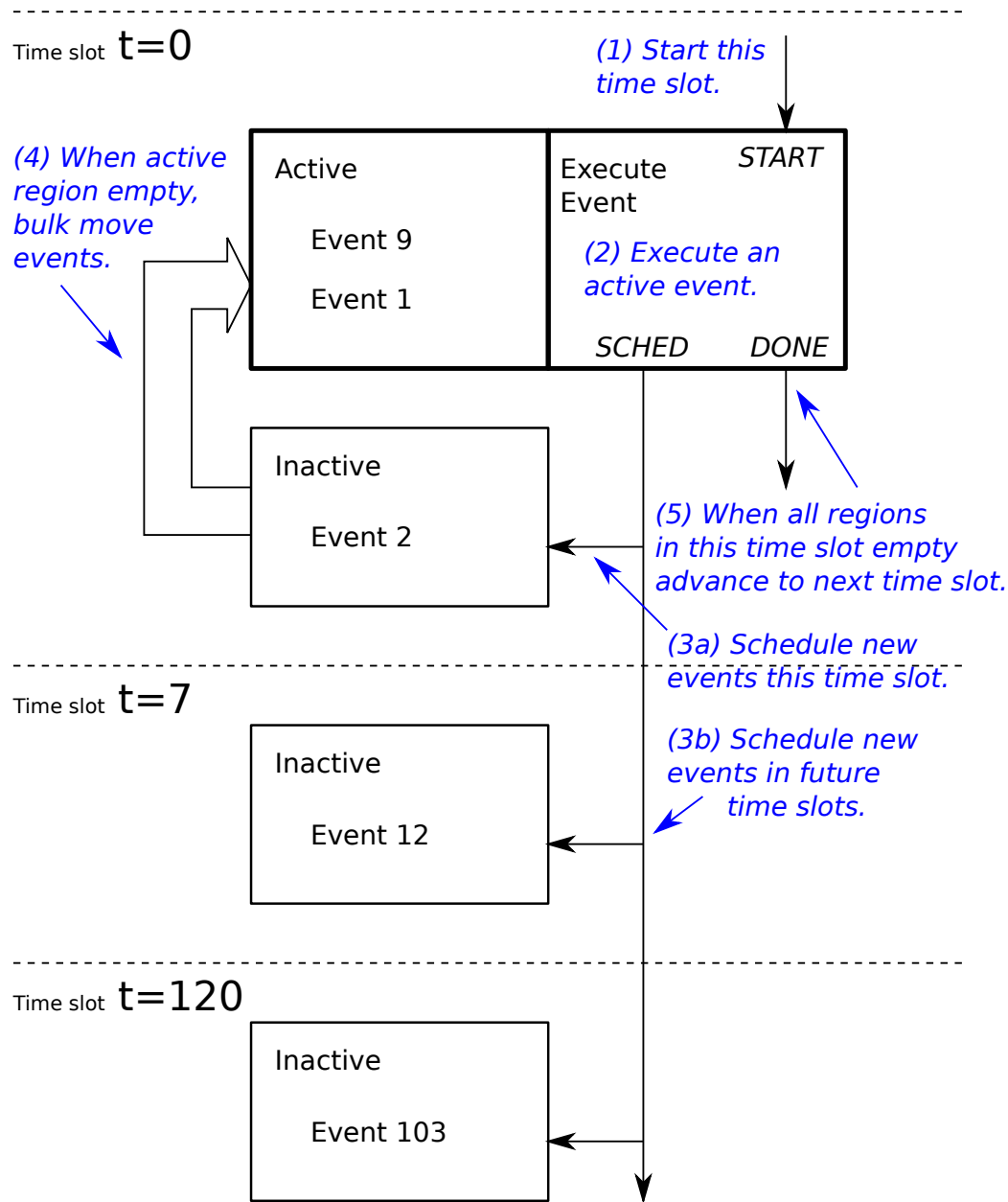
### Some Regions

#### *Active Region:*

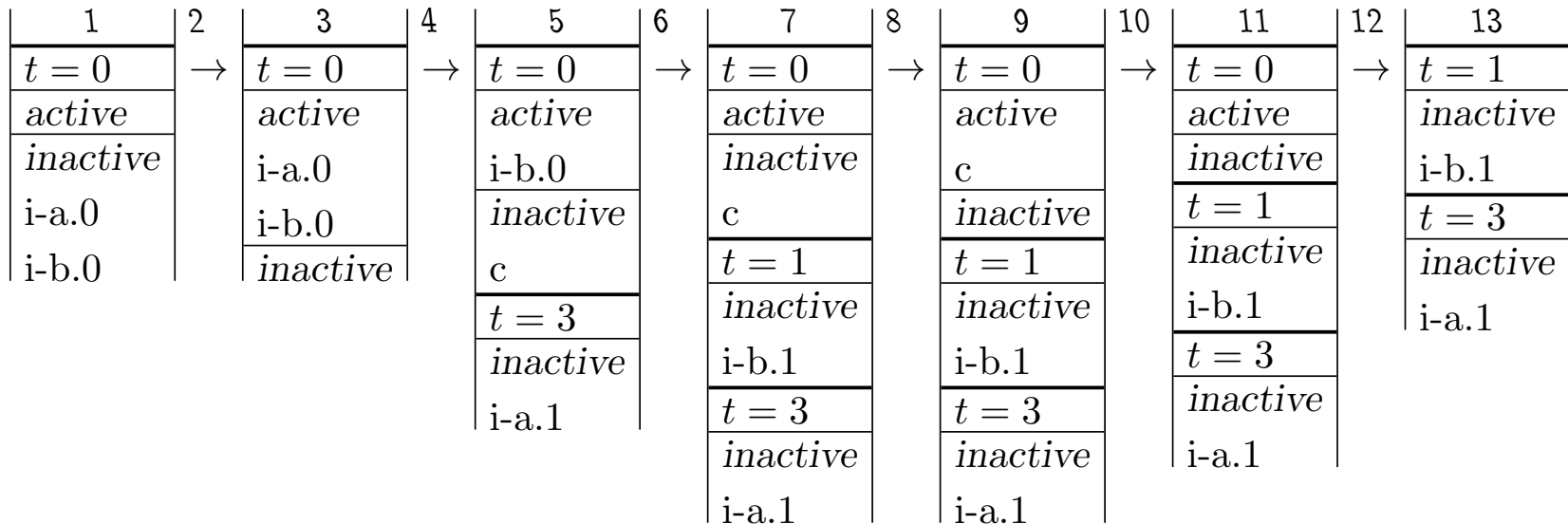
Events that the simulator is currently working on. Only the current time slot has this region.

#### *Inactive Region:*

Contains normally scheduled events. Current and future time slots have this region.



## Event Queue Example



- 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  $t = 0$  ...  
... and `i-a.1` for  $t = 3$ .
- 4: Event `i-a.0` removed from active region (it is now not scheduled anywhere).

```
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
```

## Event Queue Example

1	2	3	4	5	6	7	8	9	10	11	12	13
$t = 0$	→	$t = 0$	→	$t = 0$	→	$t = 0$	→	$t = 0$	→	$t = 0$	→	$t = 1$
active		active		active		active		active		active		inactive
inactive		i-a.0		i-b.0		inactive		c		inactive		i-b.1
i-a.0		i-b.0		inactive		c		inactive		$t = 1$		$t = 3$
i-b.0		inactive		c		$t = 1$		$t = 1$		inactive		inactive
				$t = 3$		inactive		inactive		i-b.1		i-a.1
				inactive		i-b.1		i-b.1		$t = 3$		
				i-a.1		$t = 3$		$t = 3$		inactive		
						inactive		inactive		i-a.1		
						i-a.1		i-a.1				

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 `$watch` system task.

## Event Scheduling

### *Time-Delay Scheduled*

Scheduled by a delay: `#4 a = b;`

Put in inactive region of a future time step.

### *Sensitivity List Scheduled*

Explicit event: `@( a ), @( posedge clk ), wait( stop_raining )`

Continuous assignment: `assign x = a + b;.`

Module or primitive ports: `and myAndGate(x,a,b) .`

Put in inactive region of current time step.

### *Update Events*

Non-blocking assignment: `y <= a + b;.`

Put in NBA region of current time step.



## *Permanently Scheduled*

Watch lists: `$watch(a)`.

Put in postponed region of every time step.

