DAY-68
#100DAYSRTL

"System Verilog:- IPC: Events"

"Introduction":-

Events in System Verilog, similar to Verilog, are static entities. However, System Verilog has improved upon the basic functionalities of events. Think of events as markers or flags that help control the flow of actions in your code. They're like signals that trigger specific actions to occur in your program or hardware design. They've got a few enhancements in System Verilog, making them more versatile and powerful for managing the execution sequence in your code.

"Events":-

- ✓ Events act like flags that help two different tasks or processes in a program or hardware design to work together smoothly. They make sure these tasks stay in sync, just like a countdown timer used by two people to start an activity together.
- ✓ In System Verilog, you can 'raise' an event using "->" or "->>" to signal that something has happened, while you can 'listen' for that event using @ or wait() to make sure you don't proceed until that event has been 'raised'.

```
event ev;
-> ev;
```

- ✓ Named events can be triggered using -> or ->> operator
- ✓ Processes can wait for an event using @ operator or .triggered

Code Practising:-

```
module tb;
event event_a;
initial begin
#20 ->event_a;
  //Thread1
  $display("[%0t] Thread1: triggered event_a",$time);
end
initial begin
  //Thread2
  $display("[%0t] Thread2: waiting for trigger ", $time);
@(event_a);
  $display("[%0t] Thread2: received event_a trigger ", $time);
end
initial begin
  //Thread3
  $display("[%0t] Thread3: waiting for trigger ", $time);
  wait(event_a.triggered);
  $display("[%0t] Thread3: received event_a trigger", $time);
end
endmodule
```

Result:-

```
[0] Thread2: waiting for trigger
[0] Thread3: waiting for trigger
[20] Thread1: triggered event_a
[20] Thread2: received event_a trigger
[20] Thread3: received event_a trigger
Simulation has finished. There are no more test vectors to simulate.
```

“-> Vs ->>”:-

- (Non-blocking Event Control)->: It schedules the event to trigger, but does not wait for the event to be processed immediately. It's like leaving a note or a message about the event but continuing with other tasks without waiting for it to be handled right away.
- (Blocking Event Control) ->>: It schedules the event and then waits for it to be processed before moving on to the next task. It's like sending a message and then waiting until the receiver acknowledges it before continuing with your own tasks.
- Both -> and ->> are used to trigger events. The difference lies in whether the current process waits for the event to be processed immediately (->>) or not (->).

"@ Vs wait":-

- An event's triggered state persists throughout the time step, until simulation advances. Hence if both wait for the event and the trigger of the event happen at the same time there will be a race condition and the triggered property helps to avoid that.
- A process that waits on the triggered state always unblocks, regardless of the order of wait and trigger.

Code Practising:-

```
module tb;
event event_a;
initial begin
#20 ->event_a;
  //Thread1
  $display("[%0t] Thread1: triggered event_a",$time);
end
initial begin
  //Thread2
  $display("[%0t] Thread2: waiting for trigger ", $time);
#20 @(event_a);
  $display("[%0t] Thread2: received event_a trigger ", $time);
end
initial begin
  //Thread3
  $display("[%0t] Thread3: waiting for trigger ", $time);
 #20  wait(event_a.triggered);
  $display("[%0t] Thread3: received event_a trigger", $time);
end
endmodule
```

Result:-

```
[0] Thread2: waiting for trigger
[0] Thread3: waiting for trigger
[20] Thread1: triggered event_a
[20] Thread3: received event_a trigger
Simulation has finished. There are no more test vectors to simulate.
```

"Wait Order":-

Waits for events to be triggered in the given order, and issues an error if any event executes out of order.

wait_order(a,b,c)

Code Practising:-

```
module tb;
  event a,b,c;
 initial begin
  #10 -> a;
  #10 -> b;
  #10 -> c;
 end
initial begin
  wait_order (a,b,c)
   $display ("Events were executed in the correct order");
   else
   $display ("Events were NOT executed in the correct order !");
end
endmodule
```

Result:-

```
Events were executed in the correct order
Simulation has finished. There are no more test vectors to simulate.
```