DAY-65
#100DAYSRTL

"System Verilog: Fork & Join "

"Introduction":-

System Verilog provides support for parallel or concurrent threads through fork-join construct. Multiple procedural blocks can be spawned off at the same time using fork and join. There are variations to fork-join that allow the main thread to continue executing the rest of the statements based on when child threads finish.

"Fork-Join":-

- The join statements are executed only after completing all the fork statements
- The statements in the join and fork executed concurrently

"Code Practising":-

```
module tb;
 task first();
  $display("Task 1 Started at %0t",$time);
  #20;
  $display("Task 1 Completed at %0t",$time);
 endtask
 task second();
  $display("Task 2 Started at %0t",$time);
  #30;
  $display("Task 2 Completed at %0t",$time);
 endtask
 task third();
  $display("Reached next to Join at %0t",$time);
 endtask
 initial begin
 fork
  first();
  second();
 join
  third();
end
endmodule
```

“Result”:-

```
Task 1 Started at 0
Task 2 Started at 0
Task 1 Completed at 20
Task 2 Completed at 30
Reached next to Join at 30
Simulation has finished.
```

“Fork-Join any”:-

- The join statements are executed if at least one statement in the fork is executed completely

“Code Practising”:-

```
module tb;
 task first();
  $display("Task 1 Started at %0t",$time);
  #20;
  $display("Task 1 Completed at %0t",$time);
 endtask
 task second();
  $display("Task 2 Started at %0t",$time);
  #30;
  $display("Task 2 Completed at %0t",$time);
 endtask
 task third();
  $display("Reached next to Join at %0t",$time);
 endtask
 initial begin
 fork
  first();
  second();
 join_any
  third();
end
endmodule
```

“Result”:-

```
Task 1 Started at 0
Task 2 Started at 0
Task 1 Completed at 20
Reached next to Join at 20
Task 2 Completed at 30
Simulation has finished.
```

“Fork-join none”:-

- The join statements are executed without depending on fork statements

“Code Practising”:-

```
module tb;
 task first();
  $display("Task 1 Started at %0t",$time);
  #20;
  $display("Task 1 Completed at %0t",$time);
 endtask
 task second();
  $display("Task 2 Started at %0t",$time);
  #30;
  $display("Task 2 Completed at %0t",$time);
 endtask
 task third();
  $display("Reached next to Join at %0t",$time);
 endtask
 initial begin
 fork
  first();
  second();
 join_none
  third();
end
endmodule
```

“Result”:-

```
Reached next to Join at 0
Task 1 Started at 0
Task 2 Started at 0
Task 1 Completed at 20
Task 2 Completed at 30
Simulation has finished.
```