**Module: SV for Verification**

**Section:** Threads & Interprocess Communication **Task:** Wait Fork

**Task 4**

Wait Fork

* **Code:**

module fork\_wait ();

initial begin

fork

begin

$display("Thread 1 Task Starting @ %0t",$time);

#30;

$display("Thread 1 Task Finished @ %0t",$time);

end

begin

$display("Thread 2 Task Starting @ %0t",$time);

#15;

$display("Thread 2 Task Finished @ %0t",$time);

end

join\_none

//Add code here to wait for all forked threads to finish

wait fork;

#5;

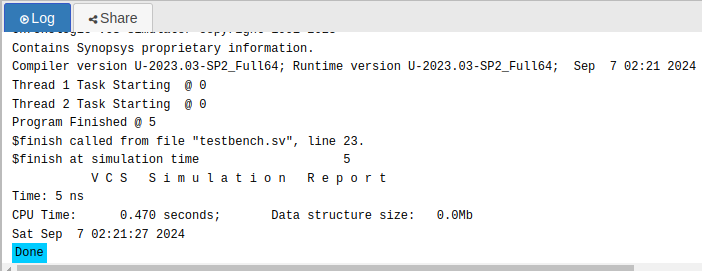
$display("Program Finished @ %0t",$time);

$finish;

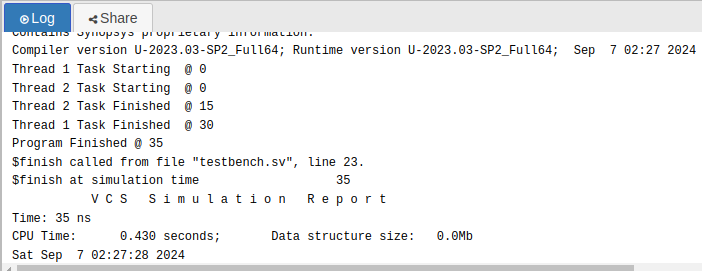
end

endmodule

* **Before Wait Fork:**
  + **Output:**



* **After Wait Fork:**
  + **Output:**



* **Observation:**

With the **wait fork;** statement, the program waits for all forked threads to complete before proceeding, ensuring that both threads finish execution before the final output is printed. In contrast, without **wait fork;**, the program moves forward immediately after the fork begins, potentially finishing before the threads complete. This leads to premature termination, with the final output being printed before either thread fully executes. In essence, **wait fork;** ensures proper synchronization, while its absence can cause incomplete execution of parallel tasks.