VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

Design and simulate a Verilog module that uses a combination of blocking (=) and non-blocking (<=) assignments along with procedural delays and intra-assignment delays. Analyze how these constructs affect the execution order, signal update timing, and final value of a variable during simulation.

```
module test1();
integer a = 0;

initial
begin
#2 a = #2 2; // at t = 4 ,a = 2
a <= #3 3; // at t = 7,a = 3
#2 a = 4; // at t = 6 , a = 4
#2 a <= 5; // at t = 8 , a = 5
#2 a = #2 6; // at t = 12, a = 6
a = #2 7; // at t = 14 , a = 7
a <= #2 8; // at t = 16,a = 8
#2 a <= #5 9; // at t = 21,a = 9
#3 a = 10; // at t = 19, a = 10
end

initial
  $monitor("at time t = %0t, a = %0d",$time, a);
endmodule
```

EDA Playground Link (Run & Observe Live):
https://www.edaplayground.com/x/hCaH

madanrajcr | madan raj c r | madanrajcr@gmail.com

VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

Analyze the time-wise value changes of a by considering blocking vs non-blocking assignments, intra-assignment delays, and parallel execution using fork…join.

```
module test2();
integer a = 0;

initial
begin
#2 a = #2 2; // at t = 4 a = 2
a <= #3 3; // at t = 7,a = 3
 fork
#1 a = 4; // at t = 5 , a = 4
#2 a <= 5; // at t = 6 , a = 5
 join
#2 a = #2 6; // at t = 10, a = 6
a = #2 7; // at t = 12 , a = 7
a <= #2 8; // at t = 14,a = 8
#2 a <= #5 9; // at t = 19,a = 9
#3 a = 10; // at t = 17, a = 10
end

initial
 $monitor("at time t = %0t, a = %0d",$time, a);
endmodule
```

EDA Playground Link (Run & Observe Live):

https://www.edaplayground.com/x/wh3w

madanrajcr | madan raj c r | madanrajcr@gmail.com

VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

Analyze the given Verilog code to determine the value of a at each time step, considering all blocking and nonblocking assignments, delays, and the execution order. Pay special attention to the begin–end block inside the fork–join, which groups multiple statements as a single parallel thread.

```
module test3();
integer a = 0;
initial begin
#2 a = #2 2; // at t = 4 a = 2
a <= #3 3; // at t = 7,a = 3
 fork
   begin
#1 a = 4; // at t = 5 , a = 4
#2 a = 5; // at t = 7 , a = 5
   end
#2 a = #2 6; // at t = 8, a = 6
 join
a = #2 7; // at t = 10 , a = 7
a <= #2 8; // at t = 12,a = 8
#2 a <= #5 9; // at t = 17,a = 9
#3 a = 10; // at t = 15, a = 10
end
initial
  $monitor("at time t = %0t, a = %0d",$time, a);
endmodule
```

EDA Playground Link (Run & Observe Live):

https://www.edaplayground.com/x/ghtL

madanrajcr | madan raj c r | madanrajcr@gmail.com

VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

Analyze the Verilog module test4 where variable a undergoes a series of blocking (=) and non-blocking (<=) assignments with specified delays (e.g., #2 a = #2 2;, #3 a <= #3 3;). Determine the value of a at each time step, considering how the delays and assignment types affect the timing.

```
module test4();
integer a = 0;
initial begin
#2 a = #2 2; // at t = 4 a = 2
#3 a <= #3 3; // at t = 10,a = 3
 fork
#1 a = 4; // at t = 8 , a = 4
#2 a <= 5; // at t = 9 , a = 5
 join
#2 a = #2 6; // at t = 13, a = 6
a = #2 7; // at t = 15 , a = 7
a <= #2 8; // at t = 17,a = 8
#2 a <= #5 9; // at t = 22,a = 9
#3 a = 10; // at t = 20, a = 10
end
initial
 $monitor("at time t = %0t, a = %0d",$time, a);
endmodule
```

EDA Playground Link (Run & Observe Live):

https://www.edaplayground.com/x/ukF9

madanrajcr | madan raj c r | madanrajcr@gmail.com

VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

Technical Tips:

Diving Deep into Verilog Timing!

When designing digital systems, mastering blocking (=) vs non-blocking (<=) assignments, along with procedural delays and intra-assignment delays, is crucial for accurate simulation and synthesis. I recently experimented with multiple Verilog modules to observe these effects firsthand.

Key Observations from the Simulation Modules:

1. **Blocking Assignments (=):**
   - Execute sequentially within a procedural block.
   - The RHS is evaluated and the LHS updated immediately, before moving to the next statement.
   - Intra-assignment delays (e.g., #2 a = #2 2;) delay the evaluation of the RHS, affecting when the value propagates.
2. **Non-blocking Assignments (<=):**
   - Schedule updates at the end of the current time step.
   - Allow multiple assignments to the same variable to be queued and updated in parallel, making them ideal for modeling register behavior.
3. **Procedural Delays (#n):**
   - Introduce specific timing gaps in simulation, essential to model real hardware delays.
   - When combined with blocking and non-blocking assignments, they influence which value appears at each simulation time step.
4. **Fork–Join Parallelism:**
   - Enables multiple threads to execute in parallel.
   - When combined with blocking assignments inside a fork–join, execution order is maintained within a thread, but parallel threads can update variables concurrently, leading to race conditions if not carefully managed.
5. **Simulation Insights (Time-wise analysis):**
   - Carefully observing $monitor outputs in EDA Playground showed that final values of a at each step depend on a combination of assignment type, delay, and execution order.

madanrajcr | madan raj c r | madanrajcr@gmail.com

VERILOG TIMING DEMYSTIFIED: REAL SIMULATION BEHAVIOR OF BLOCKING & NON-BLOCKING ASSIGNMENTS.

12/17/2025

madanrajcr | madan raj c r | madanrajcr@gmail.com

- Blocking assignments dominate the sequential thread timing, whereas non-blocking assignments schedule updates later, reflecting realistic hardware behavior.

Tips for Practitioners:

- Always distinguish when to use = vs <=; the wrong choice can introduce simulation mismatches.
- Use intra-assignment delays for precise modeling of hardware delays.
- Combine $monitor with time-stamped logs to visualize the exact timing sequence.
- Be cautious with fork–join and multiple assignments to the same variable; race conditions can mask bugs.
- Regularly simulate on tools like EDA Playground to validate timing before synthesis.

💡 **Pro Tip:**

Document time-step analysis tables alongside simulation outputs. It helps you predict the final values and makes debugging complex sequential logic much easier.

🚀 **Follow me on LinkedIn for high-quality technical content!**

If you're into **Verilog, SystemVerilog, UVM, and hardware design/verification**, I share useful tutorials, tips, and insights that can help you grow your skills and career.

👉 **Connect with me:**

🔗 https://www.linkedin.com/in/madan-raj-c-r-b6029721b/

Don’t forget to **follow** for updates and technical posts — see you there! 🙌

Try these examples live on EDA Playground:

- Module test1
- Module test2
- Module test3
- Module test4