1. What is Verilog?

- A **Hardware Description Language (HDL)** used to describe and model digital circuits.
- Mainly used for **RTL design** and **Testbench verification** in **FPGA** and **ASIC**.
- Easier syntax (resembles C) compared to VHDL.

---

2. Levels of Abstraction in Verilog

1. **Gate Level Modeling** → Circuit described using basic gates (AND, OR, XOR).
2. **Dataflow Modeling** → Uses continuous assignments (assign) to represent logic.
3. **Behavioral Modeling** → Uses procedural blocks (always, initial) to describe **what** the circuit does, not **how**.

---

3. Register Transfer Level (RTL)

- RTL describes **data flow between registers** and how the data is processed.
- Foundation for **digital design**.

---

4. Lexical Tokens (Building Blocks of Verilog)

- **White space** → Formatting in outputs (\t, \n).
- **Comments** → // for single line, /* */ for multi-line.
- **Numbers** → Written as <size>'<base><value>
    - Example: 4'b1010, 8'd25.
- **Operators** → Unary (~), Binary (+, &), Ternary (?:).
- **Identifiers & Keywords** → Names for modules, tasks, functions (case-sensitive).

---

5. Data Types

- **Nets (wire)** → Represent connections, continuously driven, default = Z.
- **Regs (reg)** → Represent storage, hold values until updated in procedural blocks.
- **Logic Values** → 0, 1, X (unknown), Z (high-impedance).

---

6. Module Declaration

- Structure:

```
module my_module(a, b, y);
  input a, b;
  output y;
  assign y = a & b;
endmodule
```

---

7. Always Block

- Defines **procedural behavior**.
- Triggered by changes in the **sensitivity list** (@(a or b) or @(*)).
- Used in **sequential** (flip-flops) and **combinational** circuits.

---

8. Blocking vs Non-Blocking Assignments

Blocking (=)	**Non-Blocking (<=)**
Sequential execution	Parallel execution
Good for combinational logic	Good for sequential (flip-flops)
Updates immediately	Updates after all RHS are read

---

9. Versions of Verilog

- **Verilog-1995** → Basic standard (IEEE 1364-1995).
- **Verilog-2001** → Improved readability, new constructs.
- **SystemVerilog** → Adds verification, object-oriented features.
- **Verilog-AMS** → Supports analog & mixed-signal.

🔄 **10. Verilog vs VHDL**

- **Verilog** → Easy, C-like syntax, widely used in **semiconductors**.
- **VHDL** → Complex syntax, often used in **aerospace & defense**.

✅ 11. Advantages of Verilog

- Simple, concise syntax.
- Industry standard, widely supported.
- Supports **simulation & synthesis**.
- Encourages **hierarchical design**.

⚠️ 12. Disadvantages

- Steep learning curve for beginners.
- Tool-dependent (simulation/synthesis).
- Version compatibility issues.
- Verification (testbench coverage) can be complex.

🎯 13. Applications of Verilog

- **Embedded Systems** (microcontrollers, FPGAs).
- **Digital Signal Processing (DSP)**.
- **Networking hardware** (routers, switches).
- **ASIC Design**.
- **Automotive & Autonomous vehicles** (ECU, ADAS).