Verilog always block

Always block is one of the procedural blocks in Verilog. Statements inside always block are executed sequentially.

Syntax:

```
always @(event)
    [statement]
always @(event) begin
    [multiple statements]
end.
```

- Events are defined by sensitivity list.

Sensitivity list:

Is the expression that defines when the always block should be executed and is specified after the @ operator within parantheses ().

- list may contain either one or a group of signals whose value changes will execute the always block.

Ex- always block gets executed whenever the value of signals a or b change.

```
always @(a or b) begin
    [statements]
end
```

→ If there is no sensitivity list, the always block repeats continuously throughout the duration of a simulation.

Ex - always clk = ~clk;

- Even if there is no sensitivity list, there should be some type of delay simulation time is advanced by a delay statement within the always.

Ex - always #10 clk = ~clk;

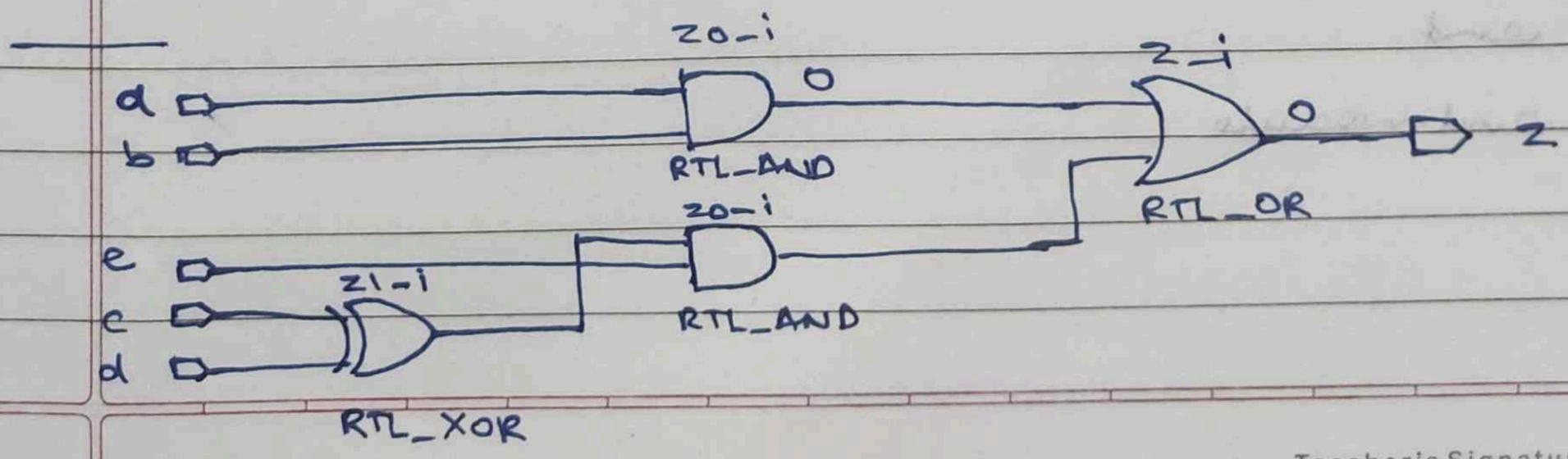
Note - Explicit delays are not synthesizable into logic gates.

Hence, real verilog design code always require a sensitivity list.

Example of simple combinational logic

(design.sv)

```
module combo (input a,b,c,d,e
              output reg z);
always @ (a or b or c or d or e) begin
    z = ((a & b) | (c ^ d) & ~e);
end
endmodule
```



Teacher's Signature:.....

(testbench)

```
module tb
  reg a, b, c, d, e;
  wire z;
  integer i;

combo uut (.a(a), .b(b), c.(c), .d(d), .e(e), .z(z));

initial begin
  a <= 0;
  b <= 0;
  c <= 0;
  d <= 0;
  e <= 0;

$monitor ("a = %0b b = %0b c = %0b d = %0b
          e = %0b z = %0b", a, b, c, d, e, z);

// As there are 5 inputs there can be 32
different input combinations.

for (i = 0; i < 32; i = i + 1) begin
  {a, b, c, d, e} = i;
  #10;
end
end
endmodule
```

Teacher's Signature:.....