DAY-81
#100DAYSRTL

"System Verilog:-Assertions Practise"

"**Aim**":-Writing the assertions for a simple up down counter

"RTL Code":-

```
module Counter(input rst,clk,en,udbar,ld,input [3:0] ld_val,
                output  reg [3:0]  cnt
               );
  always @(posedge clk) begin
    if (rst) cnt<=0;
    else begin
    if(en) begin
    if (ld) cnt<=ld_val;
    else begin
    if (udbar) cnt<=cnt+1;
    else cnt<=cnt-1;
    end
  end
 end
 end
endmodule
```

"Conditions to be checked":-

- **Reset property:-** When rst==0 then count==0;
- **Up count property:-** when udbar=1 then count=count+1;
- **Down count property:-** when udbar=1 then count=count+1;
- **Load property:-** when load==1 then count==load;
- **0 to F property:-**in case of down counter if count==0; then next value should be F
- **F to 0 property:-** in case of Up counter if count==F; then next value should be 0

“Assertions Code”:-

```
module counter_assertion(clk, rst, data, updown, load, count);
 input logic clk, rst,load_val;
 input logic updown, load;
 input logic [3:0] data, count;
 property reset_prpty;
  @(posedge clk) rst |=> (count == 4'b0);
 endproperty
 sequence up_seq;
  !load && updown;
 endsequence
 sequence down_seq;
  !load && !updown;
 endsequence
 property up_count_prpty;
  @(posedge clk) disable iff(rst)
  up_seq |=> (count == ($past(count, 1) + 1'b1));
 endproperty
 property down_count_prpty;
  @(posedge clk) disable iff(rst)
  down_seq |=> (count == ($past(count, 1) - 1'b1));
 endproperty
 property count_FtoO_prpty;
  @(posedge clk) disable iff(rst)
  (!load && updown) && (count == 4'hF) |=> (count == 4'b0);
 endproperty
 property count_OtoF_prpty;
  @(posedge clk) disable iff(rst)
  (!load && !updown) && (count == 4'b0) |=> (count == 4'hF);
 endproperty
 property load_prpty;
  @(posedge clk) disable iff(rst)
   load |=> (count == load_val);
 endproperty
 RST: assert property (reset_prpty);
 UP_COUNT: assert property (up_count_prpty);
 DOWN_COUNT: assert property (down_count_prpty);
 F2O: assert property (count_FtoO_prpty);
 O2F: assert property (count_OtoF_prpty);
 LOAD: assert property (load_prpty);
endmodule
```

“Some Insights”:-

- RTL Designers wrote the Assertions used to test the Microarchitecture Design Specification and the internal Chip level interfaces, illegal combinations.
- But they don't write it within the RTL code instead they wrote it outside the module as a separate file and then use the keyword **“bind”** to map those Assertions within RTL code.

- Sometimes RTL designers use the **“assume”** keyword as a part of Assertions to check several design-related features. Verification Engineers generally put the Assertions within the **“interface”** or inside a **“monitor”** or use a separate file as **“assertions.sv”** to increase the observability of the design.
- Verification Engineers wrote the Assertions related to inter-module interfaces, at full chip level/block level, and used it as a Checker to verify the design-related features and performance.