# DAY-2

# #100DAYSRTL

---

**“Aim”:-** To verify the Clock divider (by clk/2n) using System Verilog

## “System Verilog Code”:-

Design :-

```
//Author:- Chinna Venkata Narayana Reddy
//Date:-28/08/2023
module ClockDividerBy2n(
    input rst, clk, en,
    output  div2, div4, div8, div16
);
    reg [3:0] count;

    always @(posedge clk or posedge rst) begin
        if (rst)
            count <= 0;
        else if (en) begin
            if (count == 4'd15)
                count <= 0;
            else
                count <= count + 1;
        end
    end
    assign div2 = count[0];
    assign div4 = count[1];
    assign div8 = count[2];
    assign div16 = count[3];
endmodule
```

Interface:-

```
interface ClkDiv_if;
  logic rst,clk,en;
  logic div2,div4,div8,div16;
endinterface
```

# Transaction:-

```
class transaction;
    rand bit rst,en;
    bit div2,div4,div8,div16;
        constraint r_n{
            en dist {0:/50,1:/50};
            rst dist{0:/50,1:/50}
            ;}
        function void display(input string name);
            $display("rst=%b,en=%b,div2=%b,div4=%b,div8=%b,div16=%b",rst,en,div2,div4,div8,div16);
endfunction
function transaction copy();
    copy =new();
    copy.rst=this.rst;
    copy.en=this.en;
    copy.div2=this.div2;
    copy.div4=this.div4;
    copy.div8=this.div8;
    copy.div16=this.div16;
endfunction
endclass
```

# Generator:-

```
class generator;
    transaction tr;
    mailbox (transaction) mbx;
    mailbox (transaction) mbxref;
    int count;
    event sconex;
    event done;

    function new(mailbox (transaction) mbx, mailbox (transaction) mbxref);
        this.mbx = mbx;
        this.mbxref = mbxref;
        tr = new();
    endfunction

    task run();
        repeat (count) begin
            for (int i = 0; i < 10; i++) begin
                assert(tr.randomize()) else $error("[GEN]:Randomization Failed");
                mbx.put(tr.copy);
                mbxref.put(tr.copy);
                tr.display("GEN");
                @(sconex);
            end
        end
        -> done;
    endtask
endclass;
```

# Driver:-

```
class driver;
  transaction tr;
  mailbox (transaction) mbx;
  virtual clkDiv_if vif;
  function new(mailbox (transaction) mbx);
    this.mbx=mbx;
  endfunction
  task reset();
    vif.rst<=1'b1;
    repeat (5) @(posedge vif.clk);
    vif.rst<=0;
    @(posedge vif.clk);
    $display("[DRV]:Reset Done");
  endtask
  task run();
    forever begin
      mbx.get(tr);
      vif.en<=tr.en;
      tr.display("DRV");
      @(posedge vif.clk);
    end
  endtask

endclass
```

# Monitor:-

```
class monitor;
    transcation tr;
    mailbox(transcation) mbx;
    virtual clkDix_if vif;
function new(mailbox #(transaction) mbx);
    this.mbx=mbx;
endfunction
task run();
    tr=new();
    forever begin
        @(posedge vif.clk);
        @(posedge vif.clk);
        tr.div2=vif.div2;
        tr.div4=vif.div4;
        tr.div8=vif.div8;
        tr.div16=vif.div16;
        mbx.put(tr);
        tr.display("MON");
    end
endtask
endclass
```

# Scoreboard:-

```
class scoreboard;
    transaction tr;
    mailbox #(transaction) mbx;
    mailbox #(transaction) mbxref;
    event sconext;
    function new(mailbox #(transaction) mbx, mailbox #(transaction) mbxref);
        this.mbx = mbx;
        this.mbxref = mbxref;
    endfunction
    task run();
        forever begin
            mbx.get(tr);
            mbxref.get(tr);
            tr.display("SCO");
            if (tr.div2 == tr.div2ref && tr.div4 == tr.div4ref && tr.div8 == tr.div8ref && tr.div16 == tr.div16ref)
                $display("[SCO] : DATA MATCHED");
            else
                $display("[SCO] : DATA MISMATCHED");
            ->sconext;
        end
    endtask
endclass
```

# Environment:-

```
class environment;
  generator gen;
  scoreboard sco;
  monitor mon;
  driver drv;
  event next;
  mailbox #(transaction) gdmbx; ///gen - drv
  mailbox #(transaction) msmbx;  /// mon - sco
  mailbox #(transaction) mbxref; ///// gen -> sco
  virtual clk_Div_if vif;
  function new(virtual clk_Div_if vif);
    gdmbx=new();
    msmbx=new();
    mbxref=new();
    gen=new(gdmbx,mbxref);
    sco=new(msmbx,mbxref);
    this.vif=vif;
    drv.vif=this.vif;
    mon.vif=this.vif;
    gen.sconext=next;
    sco.sconext=next;
  endfunction
  task pre_test();
    drv.reset();
  endtask
  task test();
  fork
    gen.run();
    drv.run();
    mon.run();
    sco.run();
  join_any
  endtask
  task post_test();
    wait(gen.done.triggered);
    $finish();
  endtask
  task run();
    pre_test();
    test();
    post_test();
  endtask
endclass
```

# Testbench Top:-

```
module tb;
  reg clk,rst;
  wire div2,div4,div8,div16;
  ClockDividerBy2n Dut(.rst(0),.clk(clk),.en(1),.div2(div2),.div4(div4),.div8(div8),.div16(div16));
  initial clk=0;
  always #5 clk=~clk;
  environment env;
  initial begin
    env=new(vif);
    env.gen.count=30;
    env.run();
  end
  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
  end
endmodule
```