GRAY CODE COUNTER VERIFICATION

PRANATHI S [PES2UG22EC096]
KAUSHIK S [PES2UG22EC112]

GRAY CODE

A Gray code is a binary numeral system in which two successive values differ in only one bit. This property makes Gray code particularly useful in applications where minimal changes between consecutive states are desirable, such as in analog-to-digital converters, rotary encoders, or digital counters.

EXAMPLE

DECIMAL	BINARY	GRAY CODE
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

GRAY CODE COUNTER

A Gray Code Counter is a digital counter that counts in Gray code rather than in regular binary. This counter increments through successive Gray code values, ensuring that only one bit changes between any two consecutive numbers. This behavior helps to minimize errors or glitches that can occur when multiple bits change simultaneously, which is a common problem in traditional binary counters.

OBJECTIVE

ENSURE ONE-BIT CHANGE BETWEEN CONSECUTIVE STATES

EACH TRANSITION BETWEEN TWO CONSECUTIVE GRAY CODE VALUES SHOULD DIFFER BY ONLY ONE BIT. THIS IS THE DEFINING CHARACTERISTIC OF GRAY CODE.

EXAMPLE: BETWEEN GRAY CODE 000 (0) AND 001 (1), ONLY THE LEAST SIGNIFICANT BIT CHANGES.

DESIGN.SV

design.sv

```
// Code your design here
module gray_code_counter #(parameter WIDTH = 3) (
    input  logic                    en,
    input  logic                    clk,// Clock signal
    input  logic                    rst_n,
  output logic [2:0]       gray_code, // Gray code output
  output logic [2:0]    count
);

    logic [WIDTH-1:0] binary_count; // Binary counter

    // Binary counter logic
    always_ff @(posedge clk) begin
        if (!rst_n)
            binary_count <= 0; // Reset binary counter
      else if (en)
            binary_count <= binary_count + 1; // Increment binary counter
    end

    // Convert binary counter to Gray Code
    always_comb begin
        gray_code = binary_count ^ (binary_count >> 1);
    end

     assign count = binary_count;

  always @(posedge clk)begin
    //$display (" %0d %0d %0d %0d",gray_code,en,rst_n,count);
  end

endmodule

```

TESTBENCH.SV

```
`include "interface.sv"
`include "test"
module tbench_top;
  logic clk;

  intf i_intf();

  test t1(i_intf);

  initial
    clk=0;
  always
    #1 clk=~clk;


  gray_code_counter #(.WIDTH(4)) dut(
    .en(i_intf.en),
        .clk(clk),
        .rst_n(i_intf.rst_n),
        .gray_code(i_intf.gray_code),
     .count(i_intf.count)
    );

  initial begin
    $dumpfile("dump.vcd"); $dumpvars;
  end
endmodule
```

TRANSACTION.SV

```
class transaction;
    // Stimulus: Gray code and binary values
  rand bit [3:0] reset_duration;
    bit en=1,rst_n;
  bit [2:0] gray_code; // Gray code value (4-bit)
  bit [2:0] count;


    // Function to display the values of the Gray code and its binary equivalent
    function void display(string name);
        $display("------------------------------");
        $display("Transaction Name: %s", name);
        $display("------------------------------");
        $display("Gray Code  = %b", gray_code);
        $display("------------------------------");
      $display("Binary Value  & duration   : %b %0d",count,reset_duration);
    endfunction
endclass

```

GENERATOR.SV

```
class generator;

  transaction trans; //Handle of Transaction class


  mailbox gen2driv;  //Mailbox declaration



  function new(mailbox gen2driv);  //creation of mailbox and constructor
    this.gen2driv = gen2driv;
  endfunction


  task main();

    repeat(5)
      begin
        trans = new();
        trans.randomize();
        trans.display("Generator");
        gen2driv.put(trans);
      end

  endtask

endclass
```

DRIVER.SV

```
class driver;
    virtual intf vif;
    transaction trans;
    mailbox gen2drv;

  function new(virtual intf vif,mailbox gen2drv);
        this.vif = vif;
        this.gen2drv = gen2drv;
      endfunction

    task main();
        gen2drv.get(trans);

      fork
        repeat(5) begin


        vif.en      <= trans.en;
        vif.gray_code      = trans.gray_code;
        vif.count     = trans.count;

          trans.display("Driver");
        end
        reset_dut(trans.reset_duration);
      join
    endtask

    // Reset the DUT
    task reset_dut(int duration);
        vif.rst_n = 1;
        #(duration);
        vif.rst_n=0;
        #1;
        vif.rst_n=1;

    endtask

endclass

```

ENVIRONMENT.SV

```
`include "transaction.sv"
`include "generator.sv"
`include "driver.sv"
`include "monitor"
`include "scoreboard"

class environment;
  generator     gen;
  driver        driv;
  monitor       mon;
  scoreboard    scb;
  mailbox gen2drv, mon2scb;

  virtual intf vif;
  function new(virtual intf vif);
    //this.vif = vif;
    gen2drv   = new();
    mon2scb = new();
    gen  = new(gen2drv);
    driv = new(vif,gen2drv);
    mon  = new(vif,mon2scb);
    scb  = new(mon2scb);
  endfunction

  task test();
    fork
      gen.main();
      driv.main();
      mon.main();
      scb.main();
    join
  endtask


  task run;
    test();
    $finish;
  endtask

endclass
```

INTERFACE.SV

```
1  interface intf();
2
3     logic en;
4     logic rst_n;
5     logic [2:0] reset_duration;
6     logic [2:0] gray_code;
7     logic [2:0] count;
8
9
10 endinterface
```

TEST

```
1  `include "environment.sv"
2
3  program test(intf i_intf);
4    environment env;
5
6    initial
7      begin
8        env = new(i_intf);
9        env.run();
10     end
11
12 endprogram
```

MONITOR

```
class monitor;

  virtual intf vif;
  mailbox mon2scb;

  function new(virtual intf vif,mailbox mon2scb);
        this.vif = vif;
        this.mon2scb = mon2scb;
    endfunction

    task main();

      repeat(5)
        #3;
        begin
            transaction trans;
            trans=new();
            trans.count = vif.count;
            trans.gray_code = vif.gray_code;
            $display("\n============ Testing @ %0t ======", $time);
            mon2scb.put(trans);
            trans.display("Monitor");

        end
    endtask
endclass
```

SCOREBOARD

```
class scoreboard;
    // Mailbox to receive transactions from the monitor
    mailbox mon2scb;

    // Constructor to connect the mailbox
    function new(mailbox mon2scb);
        this.mon2scb = mon2scb;
    endfunction

    // Main task to process transactions
    task main;
        transaction trans; // Transaction object

      repeat(5)
        begin
            // Get a transaction from the monitor
            mon2scb.get(trans);

            // Check if the Gray Code is correctly derived from the binary value
          if (trans.gray_code == (trans.count ^ (trans.count >> 1)))            begin
                $display("Result is as Expected");
            end else begin
              $error("Wrong Result: Binary = %b, Gray Code = %b", trans.count,
trans.gray_code);
            end

            // Display the transaction details
            trans.display("Scoreboard");
        end
    endtask
endclass

```

OUTPUT

```
------------------------------
Transaction Name: Generator
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
Transaction Name: Generator
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 15
------------------------------
Transaction Name: Generator
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
Transaction Name: Generator
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 1
------------------------------
Transaction Name: Generator
------------------------------
Gray Code  = 000
------------------------------
```

```
Binary Value  & duration   : 000 12
------------------------------
Transaction Name: Driver
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
Transaction Name: Driver
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
Transaction Name: Driver
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
Transaction Name: Driver
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8
------------------------------
```

```
Transaction Name: Driver
------------------------------
Gray Code  = 000
------------------------------
Binary Value  & duration   : 000 8

========== Testing @ 15 ======
------------------------------
Transaction Name: Monitor
------------------------------
Gray Code  = 010
------------------------------
Binary Value  & duration   : 011 0
Result is as Expected
------------------------------
Transaction Name: Scoreboard
------------------------------
Gray Code  = 010
------------------------------
Binary Value  & duration   : 011 0
```

THANK YOU!