

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.



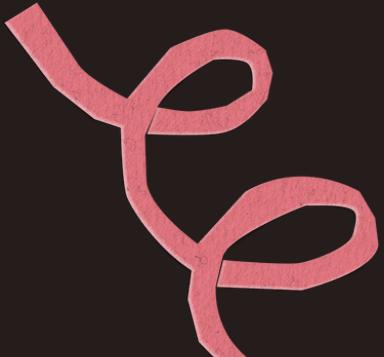


OBJECTIVE

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.



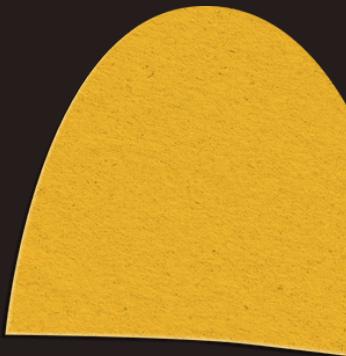
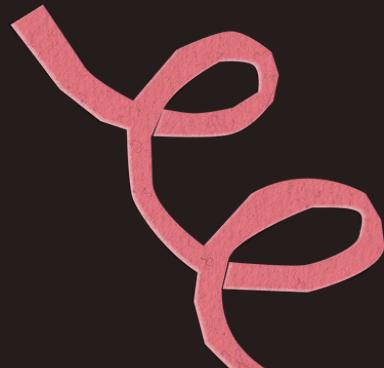
WW DESIGN.SV

```
design.sv   
  
1 // Code your design here  
2 module gray_code_counter #(parameter WIDTH = 3) (  
3     input  logic          en,  
4     input  logic          clk, // Clock signal  
5     input  logic          rst_n,  
6     output logic [2:0]    gray_code, // Gray code output  
7     output logic [2:0]    count  
8 );  
9  
10    logic [WIDTH-1:0] binary_count; // Binary counter  
11  
12    // Binary counter logic  
13    always_ff @(posedge clk) begin  
14        if (!rst_n)  
15            binary_count <= 0; // Reset binary counter  
16        else if (en)  
17            binary_count <= binary_count + 1; // Increment binary counter  
18    end  
19  
20    // Convert binary counter to Gray Code  
21    always_comb begin  
22        gray_code = binary_count ^ (binary_count >> 1);  
23    end  
24  
25        assign count = binary_count;  
26  
27    always @ (posedge clk) begin  
28        //$display ("%0d %0d %0d %0d",gray_code,en,rst_n,count);  
29    end  
30  
31 endmodule  
32
```

WW

TESTBENCH.SV

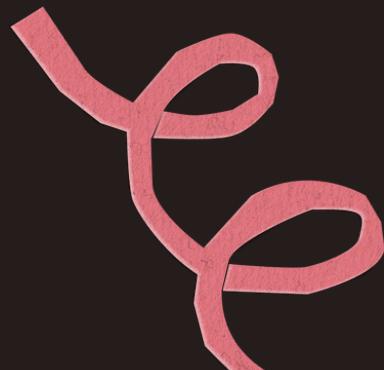
```
1 `include "interface.sv"
2 `include "test"
3 module tbench_top;
4   logic clk;
5
6   intf i_intf();
7
8   test t1(i_intf);
9
10  initial
11    clk=0;
12  always
13    #1 clk=~clk;
14
15
16  gray_code_counter #( .WIDTH(4) ) dut(
17    .en(i_intf.en),
18    .clk(clk),
19    .rst_n(i_intf.rst_n),
20    .gray_code(i_intf.gray_code),
21    .count(i_intf.count)
22  );
23
24  initial begin
25    $dumpfile("dump.vcd"); $dumpvars;
26  end
27 endmodule
```





TRANSACTION.SV

```
1 class transaction;
2     // Stimulus: Gray code and binary values
3     rand bit [3:0] reset_duration;
4     bit en=1,rst_n;
5     bit [2:0] gray_code; // Gray code value (4-bit)
6     bit [2:0] count;
7
8
9     // Function to display the values of the Gray code and its binary equivalent
10    function void display(string name);
11        $display("-----");
12        $display("Transaction Name: %s", name);
13        $display("-----");
14        $display("Gray Code = %b", gray_code);
15        $display("-----");
16        $display("Binary Value & duration : %b %0d",count,reset_duration);
17    endfunction
18 endclass
19
```



www

GENERATOR.SV

```
1 class generator;
2
3     transaction trans; //Handle of Transaction class
4
5     mailbox gen2driv; //Mailbox declaration
6
7
8
9
10    function new(mailbox gen2driv); //creation of mailbox and constructor
11        this.gen2driv = gen2driv;
12    endfunction
13
14    task main();
15
16        repeat(5)
17            begin
18                trans = new();
19                trans.randomize();
20                trans.display("Generator");
21                gen2driv.put(trans);
22            end
23
24        endtask
25
26    endclass
27
```

www DRIVER.SV

```
1 class driver;
2     virtual intf vif;
3     transaction trans;
4     mailbox gen2drv;
5
6     function new(virtual intf vif,mailbox gen2drv);
7         this.vif = vif;
8         this.gen2drv = gen2drv;
9     endfunction
10
11    task main();
12        gen2drv.get(trans);
13
14        fork
15            repeat(5) begin
16
17                vif.en     <= trans.en;
18                vif.gray_code  = trans.gray_code;
19                vif.count    = trans.count;
20
21                trans.display("Driver");
22            end
23            reset_dut(trans.reset_duration);
24        join
25    endtask
26
27
28 // Reset the DUT
29 task reset_dut(int duration);
30     vif.rst_n = 1;
31     #(duration);
32     vif.rst_n=0;
33     #1;
34     vif.rst_n=1;
35
36 endtask
37
38 endclass
39
```

WW

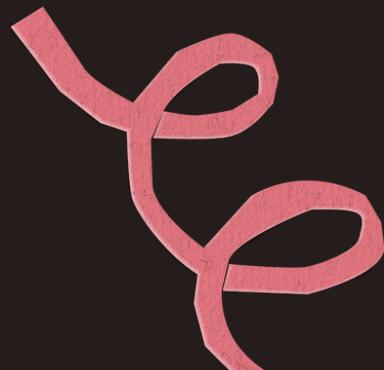
ENVIRONMENT.SV

```
1 `include "transaction.sv"
2 `include "generator.sv"
3 `include "driver.sv"
4 `include "monitor"
5 `include "scoreboard"
6
7 class environment;
8     generator      gen;
9     driver         driv;
10    monitor        mon;
11    scoreboard     scb;
12    mailbox gen2drv, mon2scb;
13
14    virtual intf vif;
15    function new(virtual intf vif);
16        //this.vif = vif;
17        gen2drv   = new();
18        mon2scb  = new();
19        gen       = new(gen2drv);
20        driv     = new(vif,gen2drv);
21        mon      = new(vif,mon2scb);
22        scb      = new(mon2scb);
23    endfunction
24
25    task test();
26        fork
27            gen.main();
28            driv.main();
29            mon.main();
30            scb.main();
31        join
32    endtask
33
34
35    task run;
36        test();
37        $finish;
38    endtask
39
40 endclass
```



INTERFACE.SV

```
1 |interface intf0;
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
```



mm 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
```

WW MONITOR

```
1 class monitor;
2
3     virtual intf vif;
4     mailbox mon2scb;
5
6     function new(virtual intf vif,mailbox mon2scb);
7         this.vif = vif;
8         this.mon2scb = mon2scb;
9     endfunction
10
11    task main();
12
13        repeat(5)
14            #3;
15            begin
16                transaction trans;
17                trans=new();
18                trans.count = vif.count;
19                trans.gray_code = vif.gray_code;
20                $display("\n===== Testing @ %0t =====", $time);
21                mon2scb.put(trans);
22                trans.display("Monitor");
23
24            end
25        endtask
26    endclass
```

WW SCOREBOARD

```
1 class scoreboard;
2     // Mailbox to receive transactions from the monitor
3     mailbox mon2scb;
4
5     // Constructor to connect the mailbox
6     function new(mailbox mon2scb);
7         this.mon2scb = mon2scb;
8     endfunction
9
10    // Main task to process transactions
11    task main;
12        transaction trans; // Transaction object
13
14        repeat(5)
15            begin
16                // Get a transaction from the monitor
17                mon2scb.get(trans);
18
19                // Check if the Gray Code is correctly derived from the binary value
20                if (trans.gray_code == (trans.count ^ (trans.count >> 1))) begin
21                    $display("Result is as Expected");
22                end else begin
23                    $error("Wrong Result: Binary = %b, Gray Code = %b", trans.count,
24                           trans.gray_code);
25                end
26
27                // Display the transaction details
28                trans.display("Scoreboard");
29            end
30        endtask
31    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 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!