



HACETTEPE UNIVERSITY
COMPUTER ENGINEERING DEPARTMENT

BBM233 LOGIC DESIGN LAB - 2022 FALL

Experiment 5 - Sequential Circuits

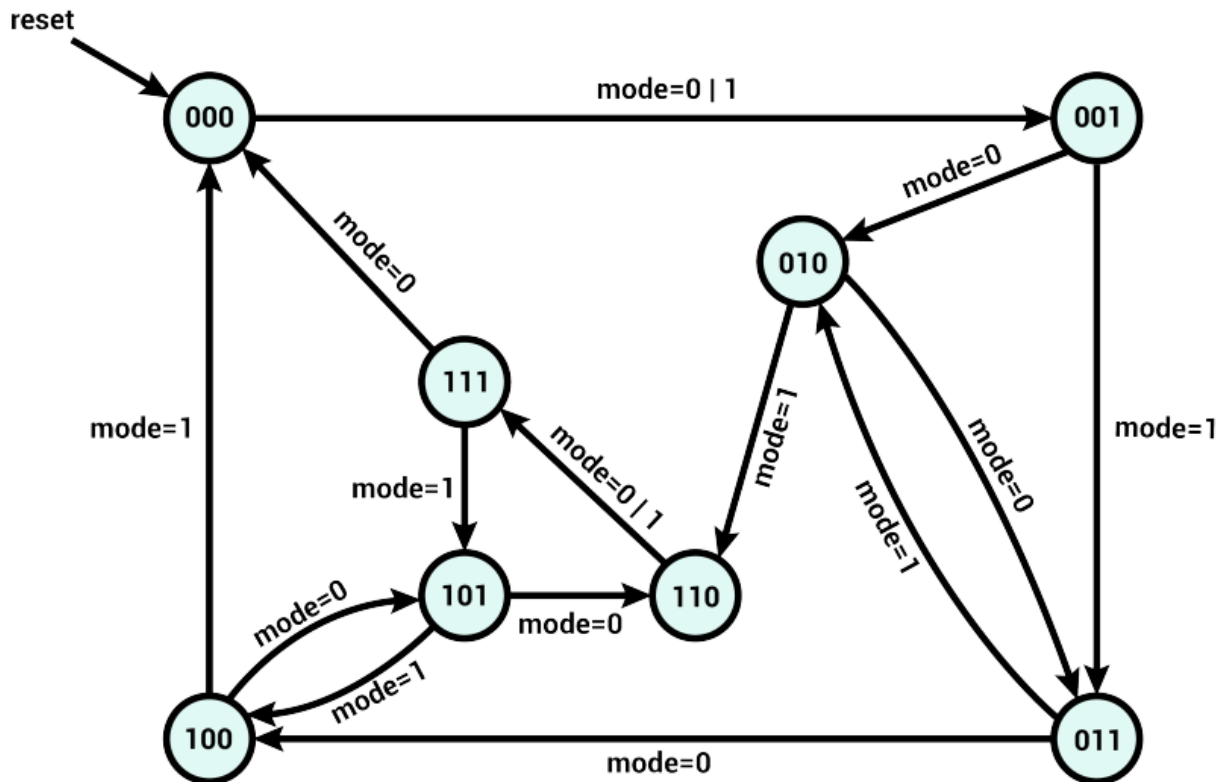
January 1, 2023

Student name:
Nurullah BAŞER

Student Number:
b2200356077

1 Problem Definition

For this experiment, I design a Binary/Gray Code Counter circuit using Verilog. The circuit should be designed to count up in binary or gray code depending on the 1-bit mode input variable, as illustrated in the state diagram below. If mode is 0, it should count in natural binary, otherwise it should count in gray code. The circuit you design should also have another 1-bit input variable reset, which resets the circuit state to the start state 000.



The counter state diagram is illustrated in the figure above. It has eight states, three 1-bit inputs, reset, mode, and clk, and a 3-bit output count that outputs the current state of the counter. Since there are eight states, we need to use three flip flops.

TABLE OF FLIP FLOPS

A	B	C	MODE	A'	B'	C'	Da	Db	Dc	Ja	Ka	Jb	Kb	Jc	Kc
0	0	0	0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	1	0	0	1	0	0	x	1	x	x	1
0	0	1	1	0	1	1	0	1	1	0	x	1	x	x	0
0	1	0	0	0	1	1	0	1	1	0	x	x	0	1	x
0	1	0	1	1	1	0	1	1	0	1	x	x	0	0	x
0	1	1	0	1	0	0	1	0	0	1	x	x	1	x	1
0	1	1	1	0	1	0	0	1	0	0	x	x	0	x	1
1	0	0	0	1	0	1	1	0	1	x	0	0	x	1	x
1	0	0	1	0	0	0	0	0	0	x	1	0	x	0	x
1	0	1	0	1	1	0	1	1	0	x	0	1	x	x	1
1	0	1	1	1	0	0	1	0	0	x	0	0	x	x	1
1	1	0	0	1	1	1	1	1	1	x	0	x	0	1	x
1	1	0	1	1	1	1	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	0	0	0	0	x	1	x	1	x	1
1	1	1	1	1	0	1	1	0	1	x	0	x	1	x	0

K-MAPS AND FORMULAS

The Karnaugh map for $D_a(A, B, C, \text{Mode})$ is shown below. The columns are labeled C, Mode with values 00, 01, 11, 10. The rows are labeled A, B with values 00, 01, 11, 10. The map contains 1s at the following positions: (01, 01), (11, 01), (11, 11), (10, 11), (00, 10), (01, 10), (11, 10), (10, 10). There are four groupings: a yellow square grouping (01, 01), (11, 01), (01, 11), (11, 11); a red square grouping (01, 01), (11, 01), (01, 10), (11, 10); a purple square grouping (01, 11), (11, 11), (01, 10), (11, 10); and a blue square grouping (11, 11), (10, 11), (11, 10), (10, 10).

		C, Mode			
		00	01	11	10
A, B	00	0	0	0	0
	01	0	1	0	1
	11	1	1	1	0
	10	1	0	1	1

$$D_a(A, B, C, \text{Mode}) = BC'\text{Mode} + A'BC\text{Mode}' + AB'\text{Mode}' + AC\text{Mode} + AC'\text{Mode}'$$

<i>Db</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	0	0	1	1
	01	1	1	1	0
	11	1	1	0	0
	10	0	0	0	1

$$Db(A, B, C, Mode) = BC' + B'CMo\text{'de}' + A'CMo\text{'de}$$

<i>Dc</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	1	1	1	0
	01	1	0	0	0
	11	1	1	1	0
	10	1	0	0	0

$$Dc(A, B, C, Mode) = A'B'Mo\text{'de}' + C'Mo\text{'de}' + ABMo\text{'de}$$

<i>Ja</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	0	0	0	0
	01	0	1	0	1
	11	-	-	-	-
	10	-	-	-	-

$$Ja(A, B, C, Mode) = BC'Mode + BCMode'$$

<i>Ka</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	-	-	-	-
	01	-	-	-	-
	11	0	0	0	1
	10	0	1	0	0

$$Ka(A, B, C, Mode) = B'C'Mode + BCMode'$$

<i>Jb</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	0	0	1	1
	01	-	-	-	-
	11	-	-	-	-
	10	0	0	0	1

$$Jb(A, B, C, Mode) = A'C + CMode'$$

<i>Kb</i>		<i>C,Mode</i>			
		00	01	11	10
<i>A,B</i>	00	-	-	-	-
	01	0	0	0	1
	11	0	0	1	1
	10	-	-	-	-

$$Kb(A, B, C, Mode) = CMode' + AC$$

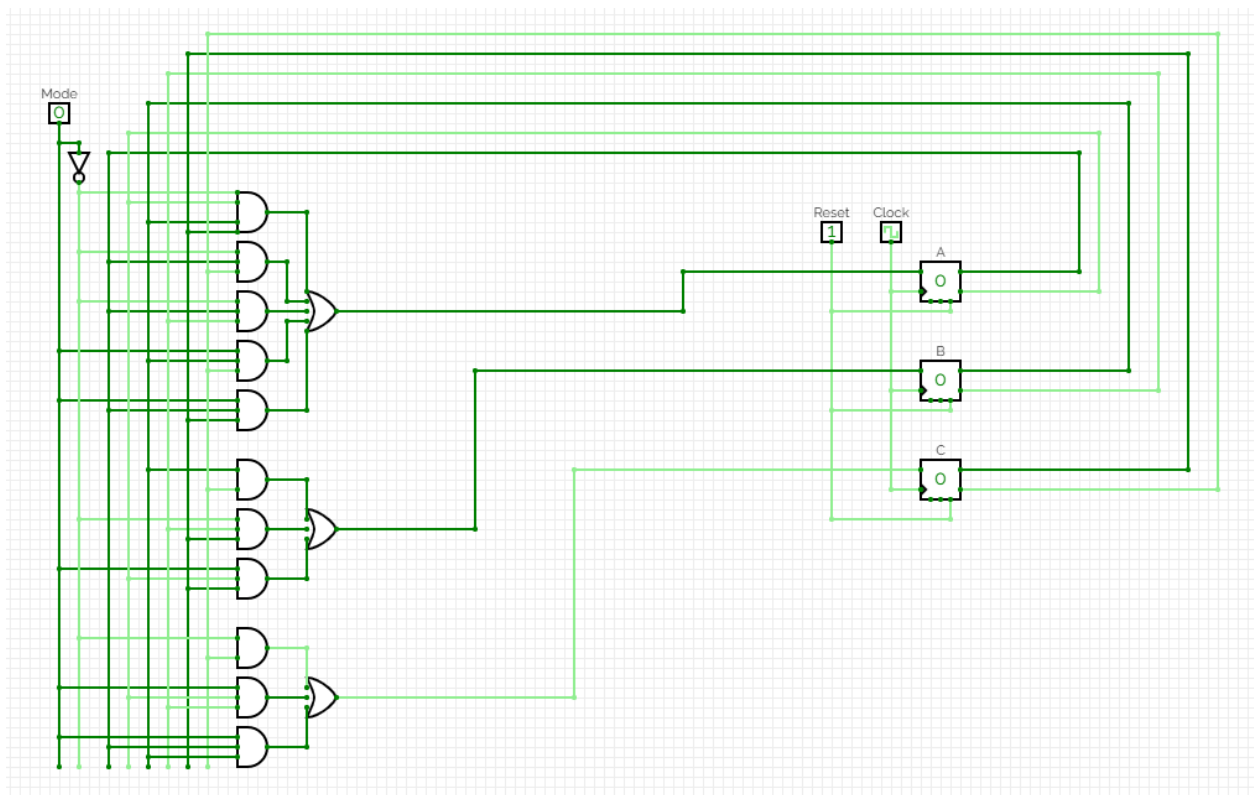
J_c		$C, Mode$			
		00	01	11	10
A, B	00	1	1	-	-
	01	1	0	-	-
	11	1	1	-	-
	10	1	0	-	-

$$J_c(A, B, C, Mode) = A'B' + Mode' + AB$$

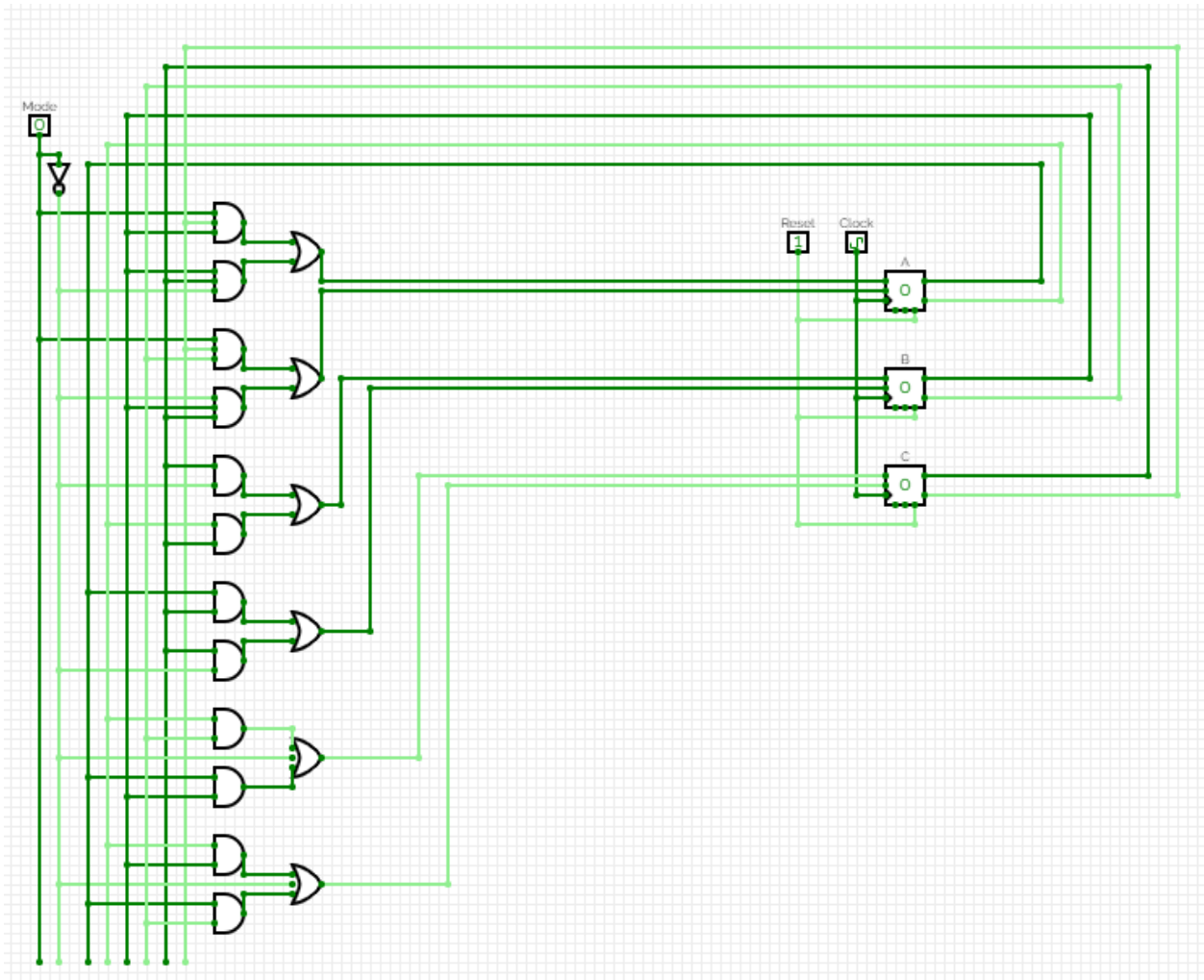
K_c		$C, Mode$			
		00	01	11	10
A, B	00	-	-	0	1
	01	-	-	1	1
	11	-	-	0	1
	10	-	-	1	1

$$K_c(A, B, C, Mode) = Mode' + A'B + AB'$$

CIRCUITS OF FLIP FLOPS



CIRCUIT OF D FLIP FLOP



CIRCUIT OF JK FLIP FLOP

2 Solution Implementation

```

1 module dff_sync_res(D, clk, sync_reset, Q);
2     input D;
3     input clk;
4     input sync_reset;
5     output reg Q;
6
7     always @(posedge clk) begin
8         if (sync_reset) // if reset is 1 then Q equals 0

```

```

9         Q <= 1'b0;
10     else // if reset is 0 than Q equals D
11         Q <= D;
12     end
13
14 endmodule

1 module jk_sync_res(J, K, clk, sync_reset, Q);
2     input J;
3     input K;
4     input clk;
5     input sync_reset;
6     output reg Q;
7
8     always @(posedge clk) begin
9         if (sync_reset) //if reset is 1 than Q = 0
10            Q <= 1'b0;
11        else begin
12            if (J && K) // 11 = complement
13                Q <= ~Q;
14            else if (J && !K) // 10 reset
15                Q <= 1'b1;
16            else if (!J && K) // 01 set
17                Q <= 1'b0;
18        end
19    end
20
21 endmodule

1 module counter_d(input reset, input clk, input mode, output [2:0] count);
2     // This module implements a 3-bit counter using D flip-flops
3     // Input determines count based on mode and current count values
4
5     // First D flip-flop in the counter
6     dff_sync_res dffA((~mode & ~count[2] & count[1] & count[0]) |
7     (~mode & count[2] & ~count[0]) |
8     (~mode & count[2] & ~count[1]) |
9     (mode & count[1] & ~count[0]) |
10    (mode & count[2] & count[0]),
11    clk, reset, count[2]);
12
13    // Second D flip-flop in the counter
14    dff_sync_res dffB((count[1] & ~count[0]) |
15    (~mode & ~count[1] & count[0]) |
16    (mode & ~count[2] & count[0]),
17    clk, reset, count[1]);
18
19    // Third D flip-flop in the counter

```

```

20     dff_sync_res dffC((~mode & ~count[0]) |
21     (mode & ~count[2] & ~count[1]) |
22     (mode & count[2] & count[1]),
23     clk, reset, count[0]);
24
25 endmodule

1 module counter_jk(input reset, input clk, input mode, output [2:0] count);
2     // This module implements a 3-bit counter using JK flip-flops
3     // J and K inputs determine count based on mode and current count values
4
5     // First JK flip-flop in the counter
6     jk_sync_res jkffA(
7         (count[1] & ~count[0] & mode) | (count[1] & count[0] & ~mode),
8         (~count[1] & ~count[0] & mode) | (count[1] & count[0] & ~mode),
9         clk,
10        reset,
11        count[2]
12    );
13
14    // Second JK flip-flop in the counter
15    jk_sync_res jkffB(
16        (count[0] & ~mode) | (~count[2] & count[0]),
17        (count[2] & count[0]) | (count[0] & ~mode),
18        clk,
19        reset,
20        count[1]
21    );
22
23    // Third JK flip-flop in the counter
24    jk_sync_res jkffC(
25        (~count[2] & ~count[1]) | (count[2] & count[1]) | (~mode),
26        (~count[2] & count[1]) | (count[2] & ~count[1]) | (~mode),
27        clk,
28        reset,
29        count[0]
30    );
31
32 endmodule

```

3 Testbench Implementation

I start with a reset at first. Then I make reset to 0 and try binary counter and gray counter. After testing and seeing the accuracy, I make reset 1 again and activate it.

```

1 'timescale 1ns/1ps
2

```

```

3  module counter_tb;
4  reg reset, clk, mode;
5  wire [2:0] count;
6  integer i = 0;
7
8  //counter_d uut(reset, clk, mode, count);
9  counter_jk c1(reset, clk, mode, count);
10
11  initial begin
12      $dumpfile("counter.vcd");
13      $dumpvars;
14      reset = 1;#50; // in the beginning reset is 1 and system is closed
15      reset = 0;#420; // reset is 0 for running code
16      reset = 1;#50; // at the beginning reset is 1 and system is closed
17      $finish;
18  end
19
20  initial begin // clock timer
21      clk = 0;
22      forever begin
23          #10;
24          clk = ~clk;
25      end
26  end
27
28  always@(posedge clk) begin //if i is below than 12 make mode 0 for running
29  //binary counter else make mode 1 for running gray counter
30      #10;
31      mode = i<12 ? 0 : 1;
32      i = i+1;
33  end
34
35  endmodule

```

4 Results

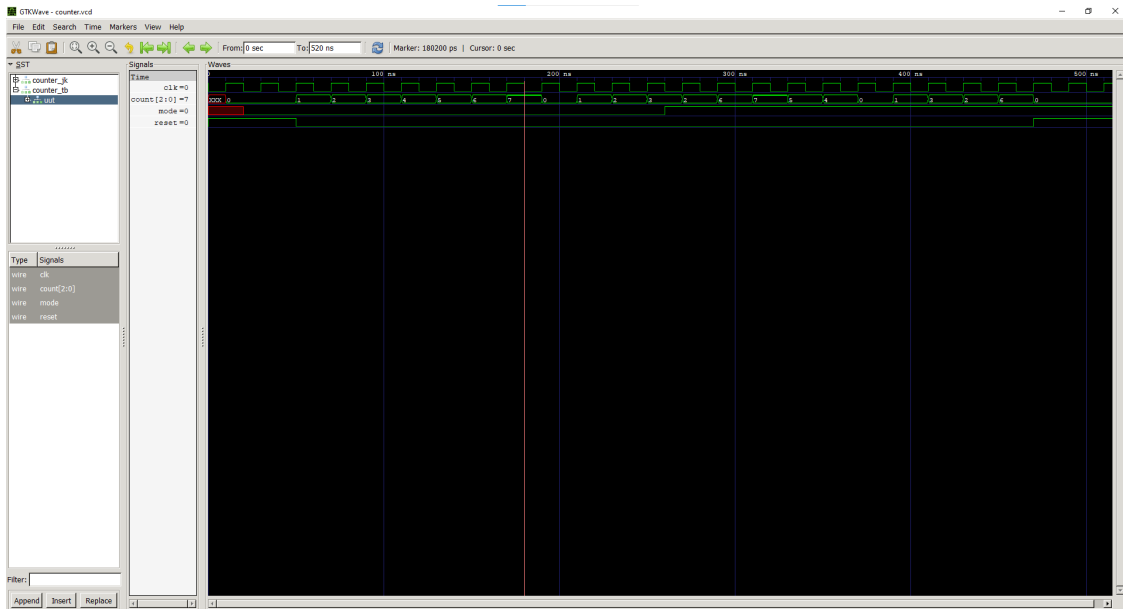


Figure 1: Resulting Waveform with D FlipFlop

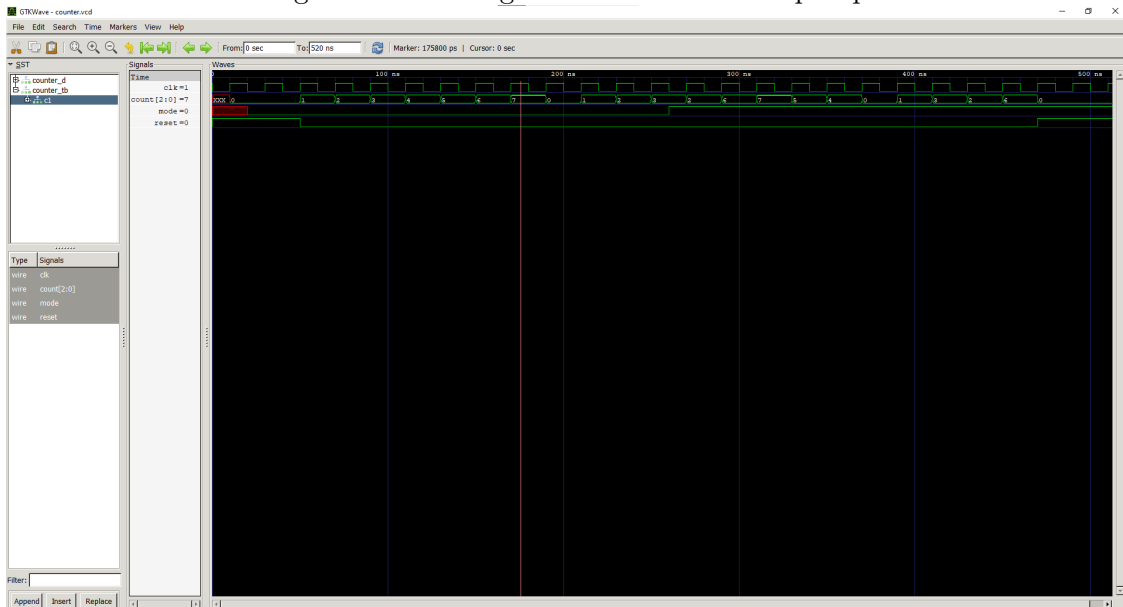


Figure 2: Resulting Waveform with J-K FlipFlop

If the reset value is 0, the system is closed. When reset is 1, we do the next state finding process according to the mode. If the mode is 0, we find it as binary, if the mode is 1, we find it as gray.

Natural Binary Counter = 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 0 - ...

Gray Counter = 0 - 1 - 3 - 2 - 6 - 7 - 5 - 4 - 0 - ...

Decimal	Gray Code	Natural Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111