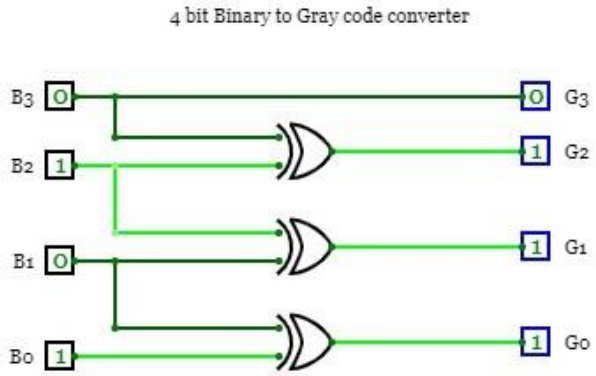
BINARY TO GRAY CODE

Binary to Gray conversion :

The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index



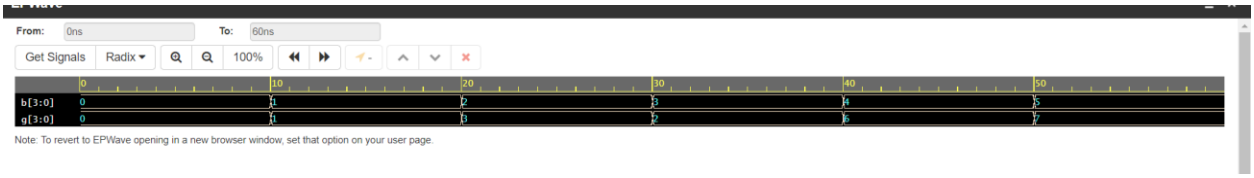
RTL CODE:

```
module binarytogray(input[3:0]b,output[3:0]g);
  assign g[3]=b[3];
  assign g[2]=b[3]^b[2];
  assign g[1]=b[2]^b[1];
  assign g[0]=b[1]^b[0];
endmodule
```

TESTBENCH

```
module test;
  reg [3:0]b;
  wire [3:0]g;
  binarytogray a1(b,g);
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
  initial
    begin
      b=4'b0000;
      #10 b=4'b0001;
      #10 b=4'b0010;
      #10 b=4'b0011;
      #10 b=4'b0100;
      #10 b=4'b0101;
      #10 b=4'b0110;
      #10 b=4'b0111;
      #10 b=4'b1000;
      #10 b=4'b1001;
      #10 b=4'b1010;
      #10 b=4'b1011;
      #10 b=4'b1100;
      #10 b=4'b1101;
```

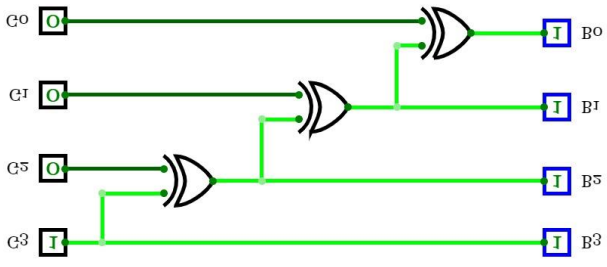
```
    #10 b=4'b1110;
    #10 b=4'b1111;
  end
 initial begin
  #60 $finish();
 end
endmodule
```



GRAY TO BINARY CODE

The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.

Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the invert of the previous binary code bit.



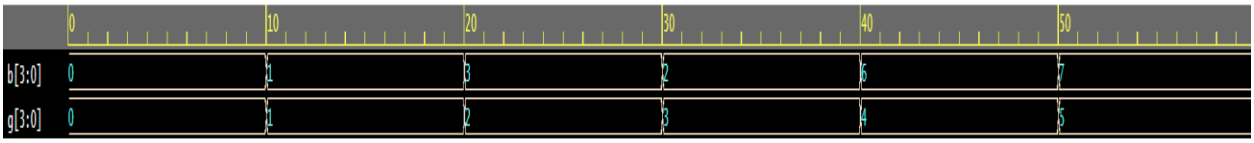
RTL CODE:

```
module graytobinary(input[3:0]g,output[3:0]b);
  assign b[3]=g[3];
  assign b[2]=g[3]^g[2];
  assign b[1]=g[2]^g[1];
  assign b[0]=g[1]^g[0];
endmodule
```

TESTBENCH:

```
module test;
  reg [3:0]g;
  wire [3:0]b;
   graytobinary a1(g,b);
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
  initial
    begin
      g=4'b0000;
      #10 g=4'b0001;
      #10 g=4'b0010;
      #10 g=4'b0011;
      #10 g=4'b0100;
      #10 g=4'b0101;
      #10 g=4'b0110;
      #10 g=4'b0111;
      #10 g=4'b1000;
```

```
    #10 g=4'b1001;
    #10 g=4'b1010;
    #10 g=4'b1011;
    #10 g=4'b1100;
    #10 g=4'b1101;
    #10 g=4'b1110;
    #10 g=4'b1111;
  end
 initial begin
  #60 $finish();
 end
endmodule
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.