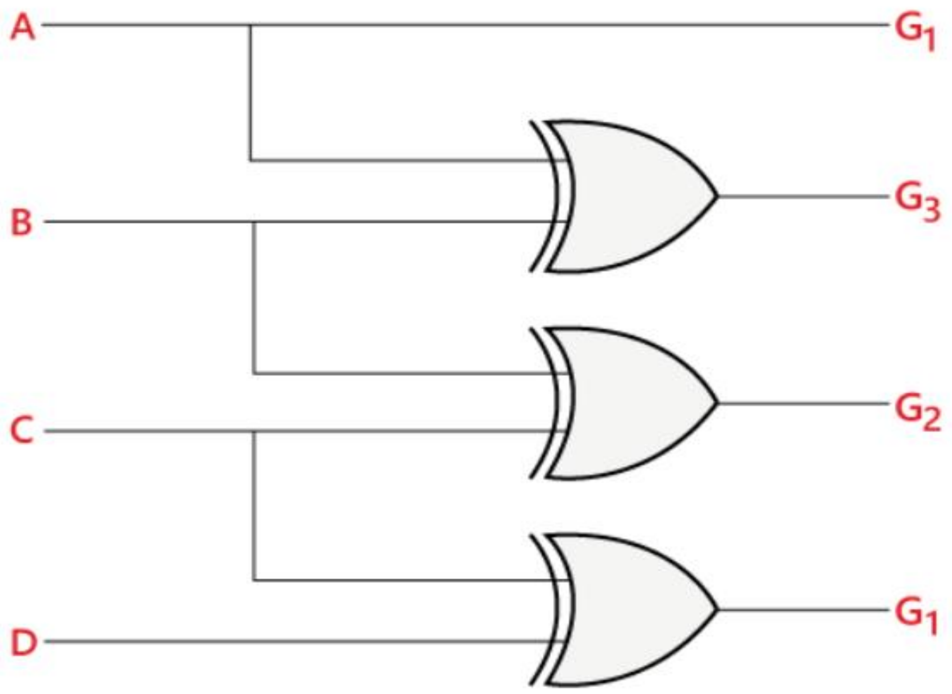
BINARY TO GRAY CODE

BINARY TO GRAY CODE:

The Binary to Gray code converter is a logical circuit that is used to convert the binary code into its equivalent Gray code. By putting the MSB of 1 below the axis and the MSB of 1 above the axis and reflecting the (n-1) bit code about an axis after 2n-1 rows, we can obtain the n-bit gray code.

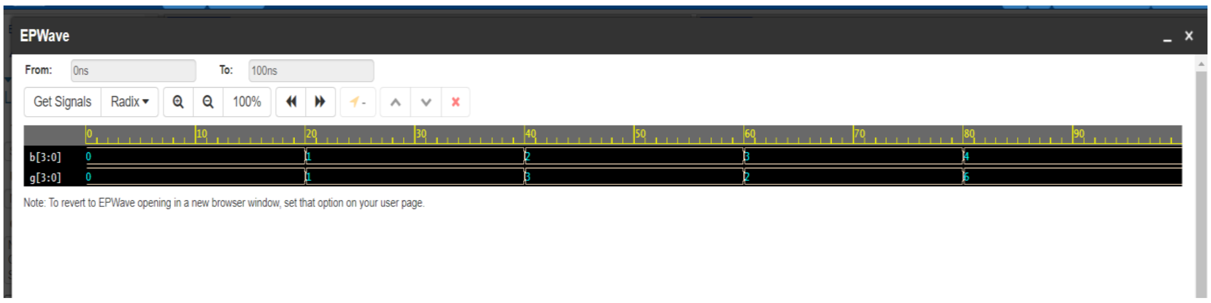


RTL CODE:

```
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
```

TEST BENCH:

```
module testbench;
  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;
      #20 b=4'b0001;
      #20 b=4'b0010;
      #20 b=4'b0011;
      #20 b=4'b0100;
    end
  initial
    begin
      #100 $finish();
    end
endmodule
```



From: 0ns   To: 100ns

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

GRAY TO BINARY CODE:

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

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

There are four inputs and four outputs. The input variable are defined as G3, G2, G1, G0 and the output variables are defined as B3, B2, B1, B0. From the truth table, combinational circuit is designed.The logical expressions are defined as :

$$G0 \oplus G1 \oplus G2 \oplus G3 = B0$$

$$G1 \oplus G2 \oplus G3 = B1$$

$$G2 \oplus G3 = B2$$

$$G3 = B3$$

RTL CODE:

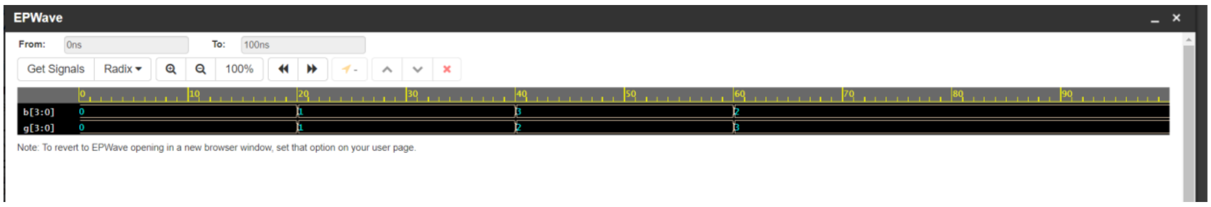
```
module graytobinary(input [3:0]g ,output[3:0]b);

  assign b[3]=g[3];

  assign b[2]=b[3]^g[2];
  assign b[1]=b[2]^g[1];
  assign b[0]=b[1]^g[0];
endmodule
```

TEST BENCH:

```
module testbench;
  reg [3:0]g;
  wire [3:0]b;
  graytobinary a1 (g,b);
  initial
    begin
      $dumpfile(".vcd");
      $dumpvars(1);
    end
  initial
    begin
      g=4'b0000;
      #20 g=4'b0001;
      #20 g=4'b0010;
      #20 g=4'b0011;
    end
  initial
    begin
      #100 $finish();
    end
endmodule
```



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