30 Days of RTL Coding

-By T Tushar Shenoy

Day 5

Problem Statement: Implementing a 4-Bit Code Converter which can perform 4 different code conversions based on the select value using case statement.

Theory:

A code converter is a logic circuit that changes data presented in one type of binary code to another type of binary code, such as binary to gray code, gray code to binary, BCD to excess-3 code and excess-3 code to BCD.

The group of symbols is called as code. The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as binary code. Binary codes can be classified into two types and they are weighted codes and non-weighted codes. If the code has positional weights, then it is said to be weighted code. Otherwise, it is an unweighted code. Weighted codes can be further classified as positively weighted codes and negatively weighted codes.

Coding is the process of translating the input information which can be understandable by the machine or a particular device. Coding can be used for security purpose to protect the information from steeling or interrupting. Code converters are used to convert the information into the desired code.The four types of codes which are included in this project are Excess-3, BCD, Gray code and Binary.

A 4-bit binary code converter is a circuit that can convert a 4-bit binary code represent from one form to another.

Description of Codes

The **Excess-3 code (or XS3)** is a non-weighted code used to express code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code. Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit.

BCD code or Binary coded Decimal codes. It is a numeric weighted binary codes, where every digit of a decimal number is expressed by a separate group of 4-bits. There are various BCD codes like 8421, 2421, 5211, etc. The BCD code is also known as the 8421 code.

Gray code is a non-weighted code. The successive gray code differs in one bit position only that means it is a unit distance code. It is also referred as cyclic code. It is not suitable for arithmetic operations. It is the most popular of the unit distance codes. It is also a reflective code.

Binary code in electronics refers to the representation of data and information using only two digits, 0 and 1. In digital electronics, binary codes are used to store, process and transmit information in computers and other digital devices. The binary system uses only two digits, 0 and 1, to represent all types of data, including numbers, letters, and special characters. The binary code is converted into electrical signals that can be processed and manipulated by electronic circuits. The binary system is a fundamental concept in digital electronics and forms the basis of modern computing.

Code Conversion

Binary to Gray code conversion: The Binary to Gray code conversion involves converting a binary number into a gray code number. The gray code is a non-weighted code where only one bit changes between consecutive values. The conversion is performed by XORing the binary number with its right shift by 1 position.

BCD to Excess-3 conversion: BCD to Excess-3 conversion involves converting a BCD number into an Excess-3 number. In the Excess-3 code, the decimal equivalent of each code is 3 more than the BCD code. For example, if the BCD code is 1001, its Excess-3 code will be 1100.

Gray code to Binary conversion: The Gray code to Binary conversion involves converting a gray code number into its equivalent binary number. The conversion in general is performed by starting with the least significant bit (LSB) and XORing it with the previous bit. This process is repeated for each bit, starting from the LSB, to get the equivalent binary number.

Excess-3 to BCD conversion: The Excess-3 to BCD conversion involves converting an Excess-3 code into its equivalent BCD code. The conversion in general is performed by subtracting 3 from each digit in the Excess-3 code. For example, if the Excess-3 code is 0100, its BCD code will be 0001.

Verilog Code:

```
module Code_Converter(code_in,select,code_out);
input [3:0]code_in;
input [1:0]select;
output reg[3:0]code_out;

always@(code_in,select)
    begin
        case(select)
            //Binary to Gray
            2'b00:begin

code_out={code_in[3],code_in[3]^code_in[2],code_in[2]^cod
e_in[1],code_in[1]^code_in[0]};
                    end

             //BCD to Excess-3
            2'b01:begin
                  if(code_in<4'b1010)
                     code_out=code_in+4'b0011;
                  else
                     code_out=4'bxxxx;
                  end

            //Gray Code to Binary
```

```
        2'b10:begin

code_out={code_in[3],code_in[3]^code_in[2],code_in[3]^code_in[2]^code_in[1],code_in[3]^code_in[2]^code_in[1]^code_in[0]};
            end

        //Excess-3 to BCD
        2'b11:begin
            if(code_in>4'b0010&&code_in<4'b1101)
              code_out=code_in-4'b0011;
            else
              code_out=4'bxxxx;
            end
        endcase

    end

endmodule
```

Testbench Code:

```
module Code_Converter_tb();


reg [3:0]code_in;
reg [1:0]select;
wire [3:0]code_out;


Code_Converter
dut(.code_in(code_in),.select(select),.code_out(code_out));


initial begin
   code_in=4'b1111;     //Initializing the Code_in to 1111


// Performs Binary to Gray code
    select=2'b00;
      stimulus();
// Performs BCD to Excess-3 code
    select=2'b01;
      stimulus();
// Performs Gray Code to Binary
    select=2'b10;
      stimulus();
// Performs Excess-3 code to Binary
    select=2'b11;
     stimulus();
     $finish;
end
```

```
task stimulus;
    integer i;
        for(i=0;i<16;i=i+1)
            begin
                code_in=code_in+1'b1;
                #5;
            end
endtask
endmodule
```

Truth Table:

Table 1.1 Binary to Gray Code

BINARY INPUT				GRAY CODE OUTPUT			
A	B	C	D	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Table 1.2 BCD to Excess-3

BCD INPUT				EXCESS-3 OUTPUT			
A	B	C	D	C_3	C_2	C_1	C_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Table 1.3 Gray Code to Binary

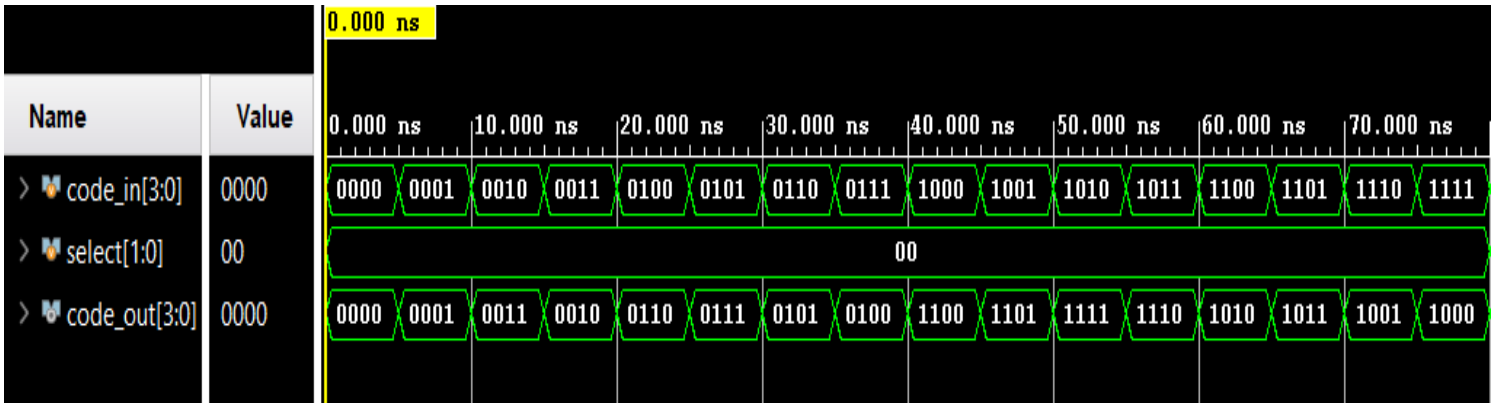
GRAY CODE INPUT				BINARY OUTPUT			
A	B	C	D	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

Table 1.4 Excess-3 to BCD

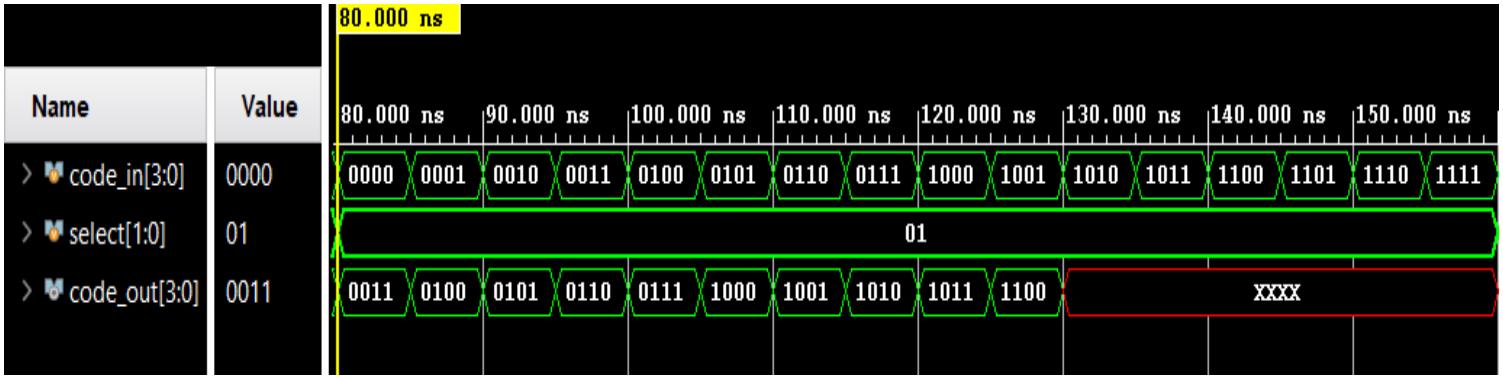
EXCESS-3 INPUT				BCD OUTPUT			
A	B	C	D	C_3	C_2	C_1	C_0
0	0	0	0	X	X	X	X
0	0	0	1	X	X	X	X
0	0	1	0	X	X	X	X
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Simulation Output:

- **Binary to Gray Code** Conversion, select=00

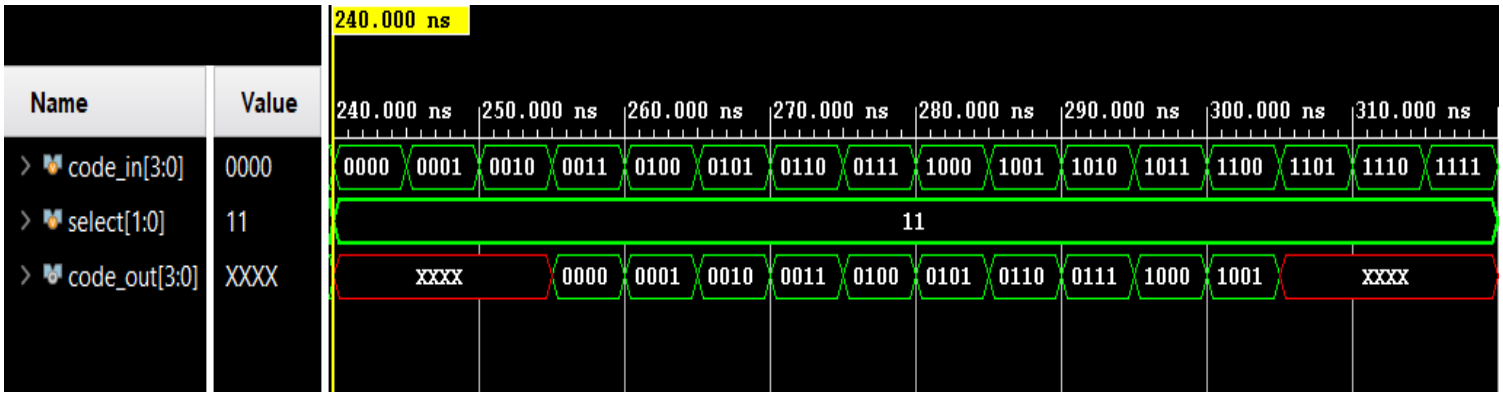


- **BCD to Excess-3** Conversion, select=01

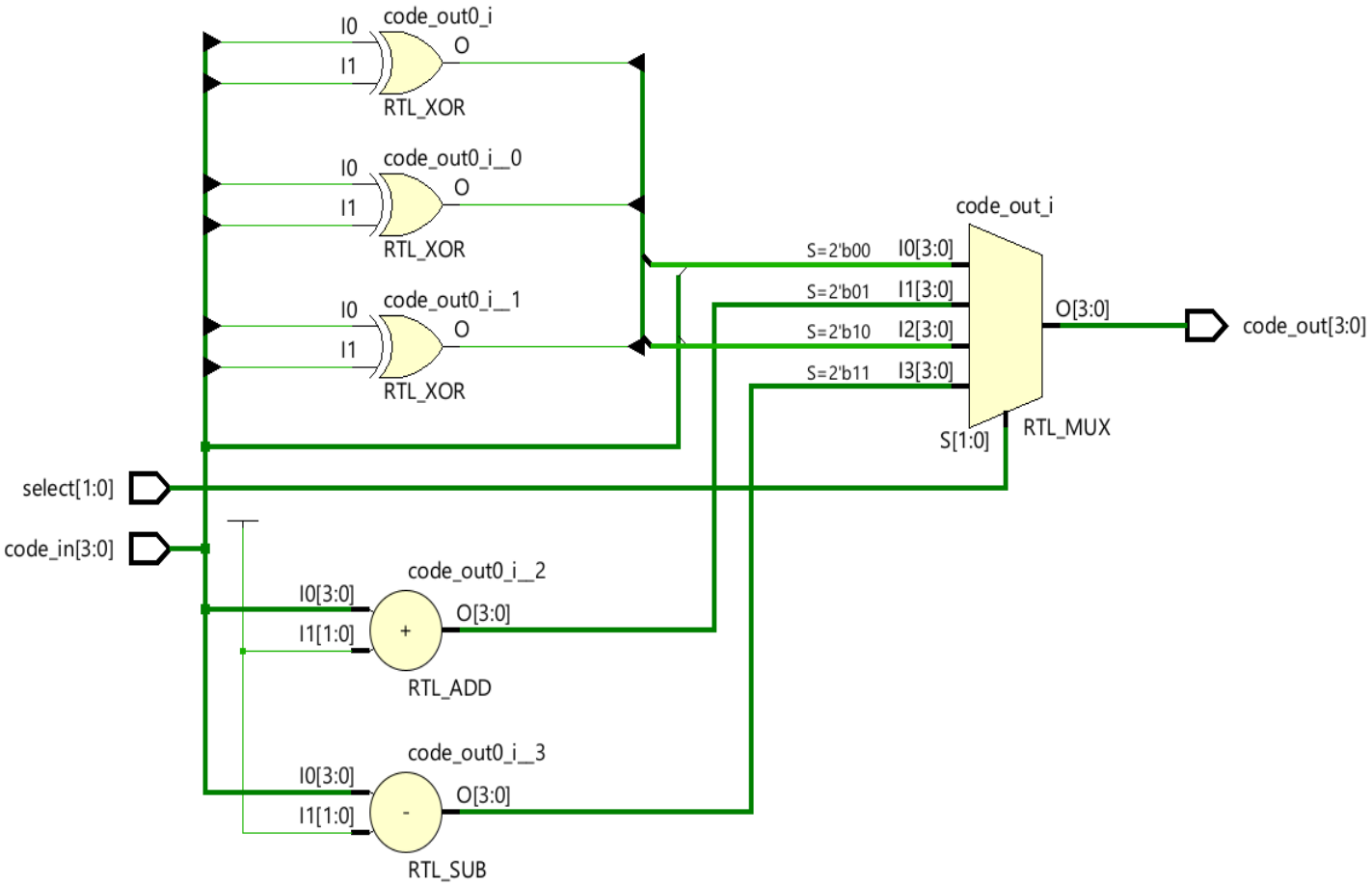


- **Gray Code to Binary** Conversion, select=10

- **Excess-3 to BCD** Conversion, select=11



Schematic:



GitHub Repository URL: - https://github.com/tusharshenoy/RTL-Day-5-Code-Converter