1st Day

Encoder

- An encoder is a combinational circuit that converts binary information in the form of a 2N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time. As an example, let's consider **Octal to Binary** encoder. As shown in the following figure, an octal-to-binary encoder takes 8 input lines and generates 3 output lines.

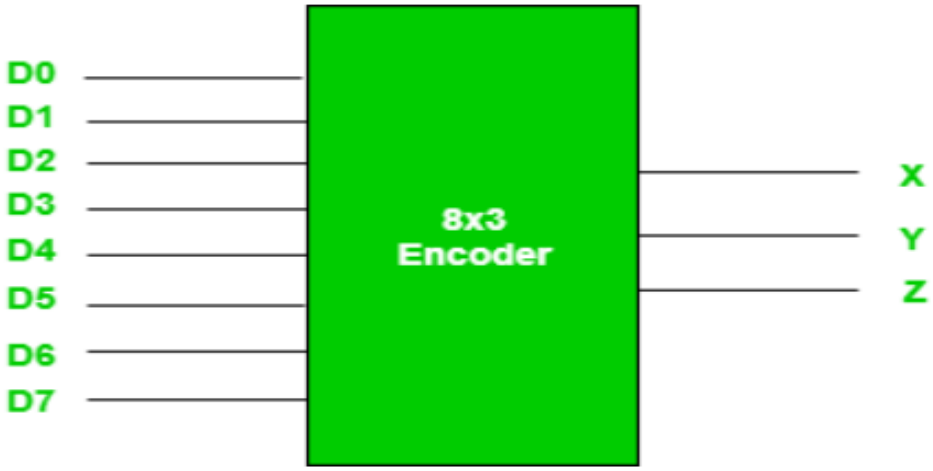
Applications

Data Transmission: Encoders are used in digital communication systems to convert parallel data into serial data for efficient transmission.

Keyboards: Convert keystrokes into digital codes for processing.

Memory Addressing: Encoders are used in memory chips to select memory locations.

Barcode Scanners & QR Code Readers: Convert scanned data into digital format.



Truth Table –

D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Design

```
module encoder_8x3 ( input[7:0]in, output reg [2:0]out);
  always @(*)
  begin
    case (in)
      8'b0000_0001: out = 3'b000;
      8'b0000_0010: out = 3'b001;
      8'b0000_0100: out = 3'b010;
      8'b0000_1000: out = 3'b011;
      8'b0001_0000: out = 3'b100;
      8'b0010_0000: out = 3'b101;
      8'b0100_0000: out = 3'b110;
      8'b1000_0000: out = 3'b111;
      default
      begin
       out = 3'bxxx;
      end
    endcase
  end
```

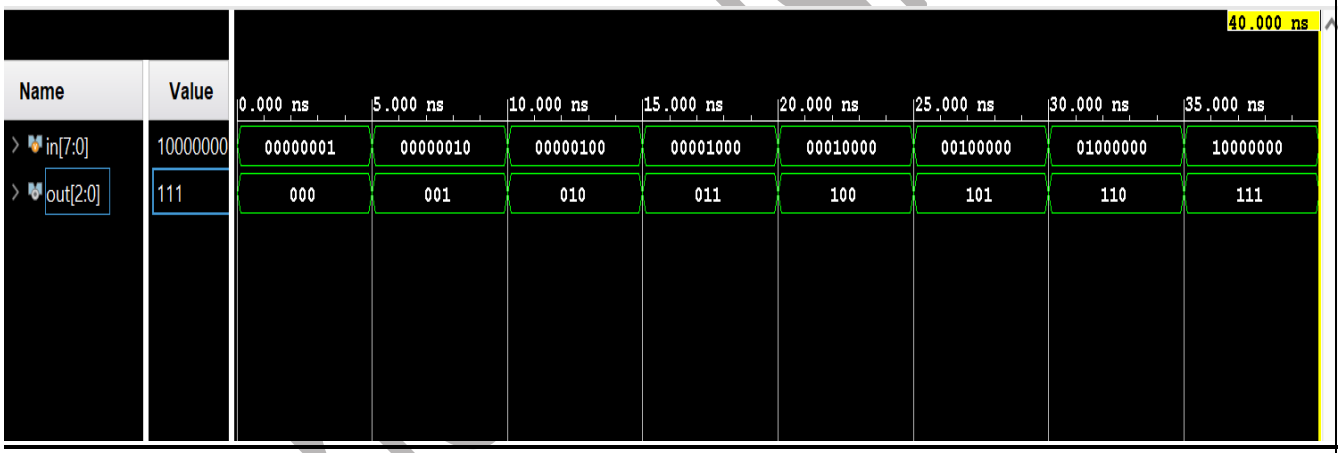
```
endmodule
```

Testbench

```
module tb_encoder_8x3;
   reg [7:0] in;
   wire [2:0] out;

   encoder_8x3 dut (.in(in), .out(out));

   initial begin
      $monitor("Input = %b | Output = %b", in, out);

      in = 8'b0000_0001; #5;
      in = 8'b0000_0010; #5;
      in = 8'b0000_0100; #5;
      in = 8'b0000_1000; #5;
      in = 8'b0001_0000; #5;
      in = 8'b0010_0000; #5;
      in = 8'b0100_0000; #5;
      in = 8'b1000_0000; #5;

      $finish;  // End simulation
   end
endmodule
```

Simulation

```
#    }
# }
# run 1000ns
Input = 00000001 | Output = 000
Input = 00000010 | Output = 001
Input = 00000100 | Output = 010
Input = 00001000 | Output = 011
Input = 00010000 | Output = 100
Input = 00100000 | Output = 101
Input = 01000000 | Output = 110
Input = 10000000 | Output = 111
$finish called at time : 40 ns : File "C:/Users/manojmsd/100_days_of_RTL/100_days_of_RTL.srcs/sources_1/new/encoder_tb.v" Line 41
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_encoder_8x3_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:11 . Memory (MB): peak = 1568.320 ; gain = 35.629
```



Priority Encoder

In case of an ordinary encoder, one and only one decimal input can be activated at any given time. But in the case of some practical digital systems, two or more decimal inputs can unintentionally become active at the same time that might cause a confusion. For example, on a keyboard, a user presses key 4 before releasing another key 2. In such a situation, the output will be corresponding to $(6)_{10}$, instead of being $(4)_{10}$ or $(2)_{10}$. This kind of problems can be solved with the help of priority encoder.

📌 **Difference?**
✓ **Normal Encoder** → Requires **only one** input to be HIGH at a time.
✓ **Priority Encoder** → Works even when **multiple inputs** are HIGH and gives priority to the highest index.

Design

```
module priority_encoder_8x3 (input [7:0] in,output reg [2:0]out);
  always @(*)
   begin
     if (in[7])
        out = 3'b111;  // Highest priority
     else if(in[6])
        out = 3'b110;
     else if(in[5])
        out = 3'b101;
     else if(in[4])
        out = 3'b100;
     else if(in[3])
        out = 3'b011;
     else if(in[2])
        out = 3'b010;
     else if(in[1])
        out = 3'b001;
```

```
        else if(in[0])
            out = 3'b000;
        else
            out = 3'bxxx;
    end
endmodule
```

Testbench

```
module tb_priority_encoder_8x3;
    reg [7:0] in;
    wire [2:0] out;

    priority_encoder_8x3 dut (.in(in), .out(out));
    initial
     begin
       $monitor("Input = %b | Output = %b", in, out);
      end
    initial
      begin
      repeat(10)
      begin
       in=$random();
       #5;
      end
       $finish;
    end
endmodule
```

Simulation

```
#    }
# }
# run 1000ns
Input = 00100100 | Output = 101
Input = 10000001 | Output = 111
Input = 00001001 | Output = 011
Input = 01100011 | Output = 110
Input = 00001101 | Output = 011
Input = 10001101 | Output = 111
Input = 01100101 | Output = 110
Input = 00010010 | Output = 100
Input = 00000001 | Output = 000
Input = 00001101 | Output = 011
$finish called at time : 50 ns : File "C:/Users/manojmsd/100_days_of_RTL/100_days_of_RTL.srcs/sources_1/new/priority_enc_tb.v" Line 4
xsim: Time (s): cpu = 00:00:08 ; elapsed = 00:00:06 . Memory (MB): peak = 1372.535 ; gain = 32.758
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_priority_encoder_8x3_behav' loaded.
```