30 Days of RTL Coding

-By T Tushar Shenoy

Day 14

Problem Statement: Implementing a 2 to 4 Decoder, 3 to 8 Decoder and 4 to 16 Decoder using Gate/Structural Implementation.

Theory:

The combinational circuit that change the binary information into 2^N output lines is known as **Decoders.** The binary information is passed in the form of N input lines. The output lines define the 2^N-bit code for the binary information. In simple words, the **Decoder** performs the reverse operation of the **Encoder**. At a time, only one input line is activated for simplicity. The produced 2^N-bit output code is equivalent to the binary information.

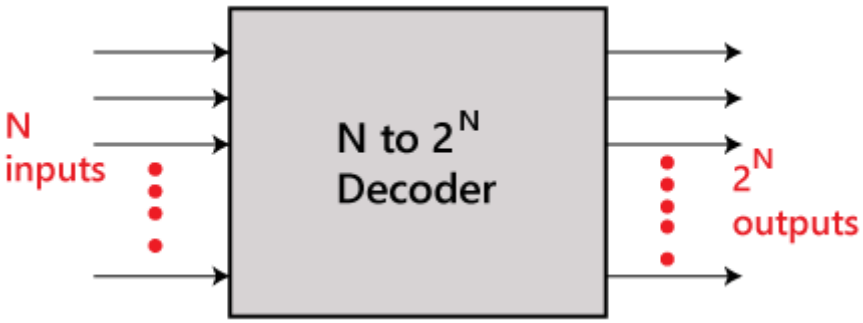


FIG: Decoder Block Diagram

1. 2 to 4 line decoder:

In the 2 to 4 line decoder, there is a total of three inputs, i.e., A_0, and A_1 and E and four outputs, i.e., Y_0, Y_1, Y_2, and Y_3. For each combination of inputs, when the enable 'E' is set to 1, one of these four outputs will be 1.

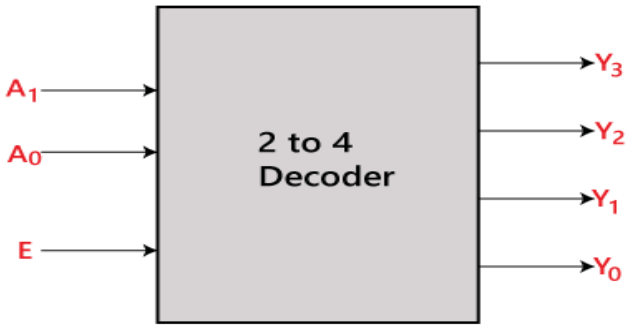


FIG: 2 to 4 Decoder Block Diagram

Truth Table:

Enable	INPUTS		OUTPUTS			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

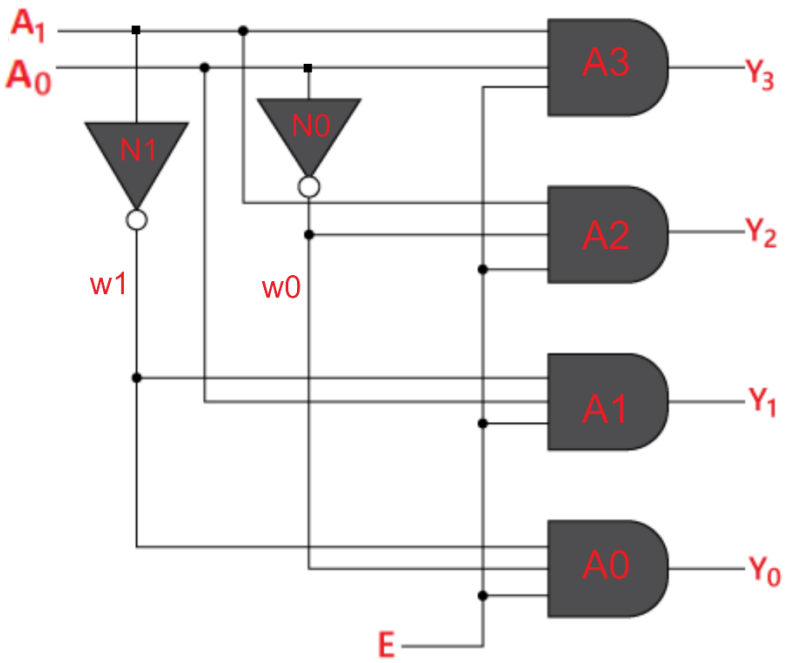


FIG: 2 to 4 Decoder Logic Diagram

Verilog Code:

```
//Verilog code for 2 to 4 Decoder
module Decoder2to4(A,E,Y);

input [1:0]A;
input E;
output [3:0]Y;

wire [1:0]w;
not N0(w[0],A[0]);
not N1(w[1],A[1]);


and A3(Y[3],A[0],A[1],E);
and A2(Y[2],A[1],w[0],E);
and A1(Y[1],A[0],w[1],E);
and A0(Y[0],w[0],w[1],E);

endmodule
```

Testbench Code:

```
// Testbench Code for 2 to 4 Decoder
module Decoder2to4_tb();

reg [1:0]A;
reg E;
wire [3:0]Y;

Decoder2to4 dut(.A(A),.E(E),.Y(Y));

initial begin
   E=1'b0;
   A=2'bxx;
#5 E=1'b1;
   A=2'b00;
#5 A=2'b01;
#5 A=2'b10;
#5 A=2'b11;
#5 $finish;
end

endmodule
```

Schematic:

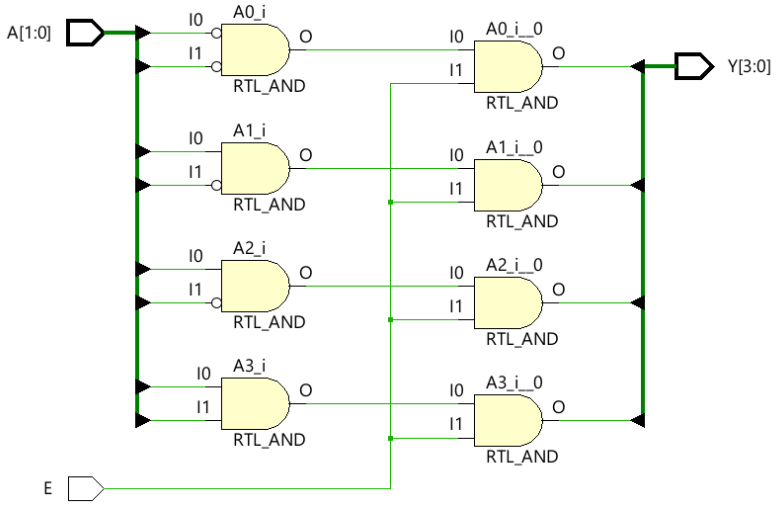
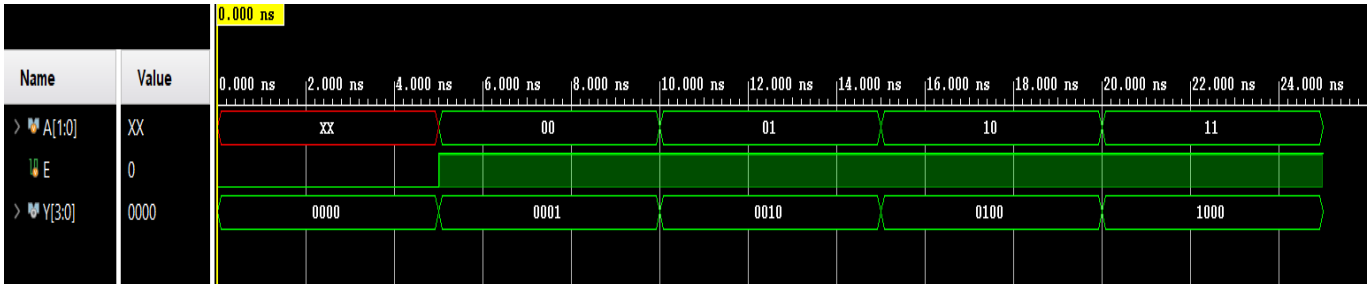


FIG: 2 to 4 Decoder Schematic

Simulation Output:



2. 3 to 8 line decoder:

The 3 to 8 line decoder is also known as binary to Octal Decoder. In a 3 to 8 line decoder, there is a total of eight outputs, i.e., Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, and Y_7 and three outputs, i.e., A_0, A1, and A_2. This circuit has an enable input 'E'. Just like 2 to 4 line decoder, when enable 'E' is set to 1, one of these four outputs will be 1.

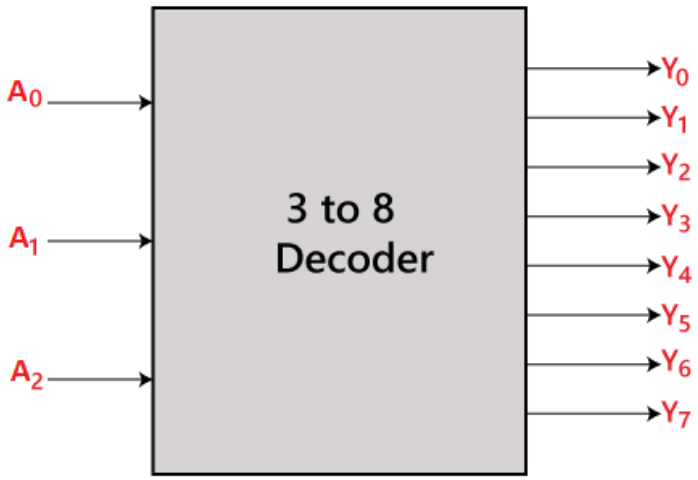


FIG: 3 to 8 Decoder Block Diagram

Truth Table:

Enable	INPUTS			Outputs							
E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

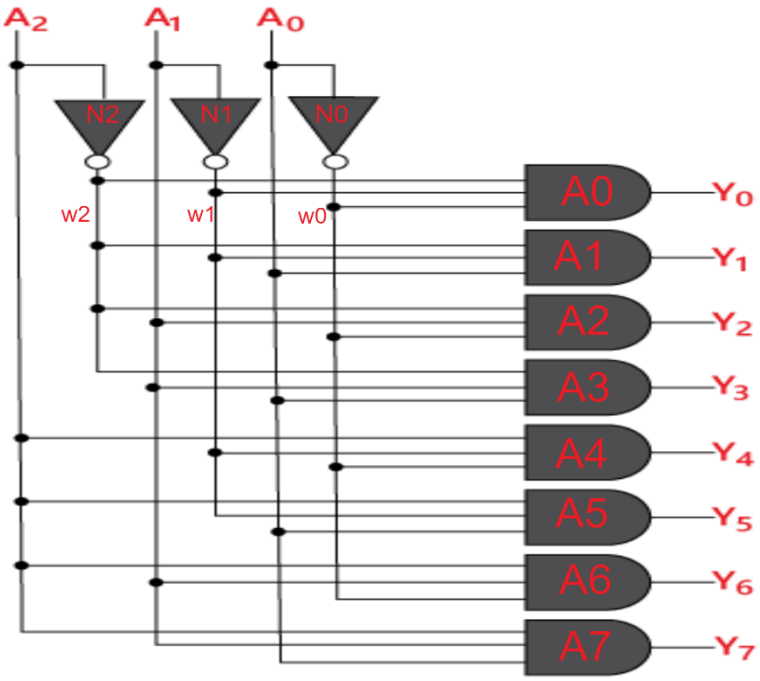


FIG: 3 to 8 Decoder Logic Diagram

Verilog Code:

```
// Verilog Code for 8x3 Encoder
module Encoder8x3(Y,A);

input [7:0]Y;
output [2:0]A;

or O2(A[2],Y[7],Y[6],Y[5],Y[4]);
or O1(A[1],Y[7],Y[6],Y[3],Y[2]);
or O0(A[0],Y[7],Y[5],Y[3],Y[1]);

endmodule
```

Testbench Code:

```
//Testbench for 8x3 Encoder
module Encoder8x3_tb();

reg [7:0]Y;
wire [2:0]A;

Encoder8x3 dut(.Y(Y),.A(A));

initial begin
   Y=8'b00000001;
#5 Y=8'b00000010;
#5 Y=8'b00000100;
#5 Y=8'b00001000;
#5 Y=8'b00010000;
#5 Y=8'b00100000;
#5 Y=8'b01000000;
#5 Y=8'b10000000;
#5 $finish;

end
endmodule
```

Schematic:

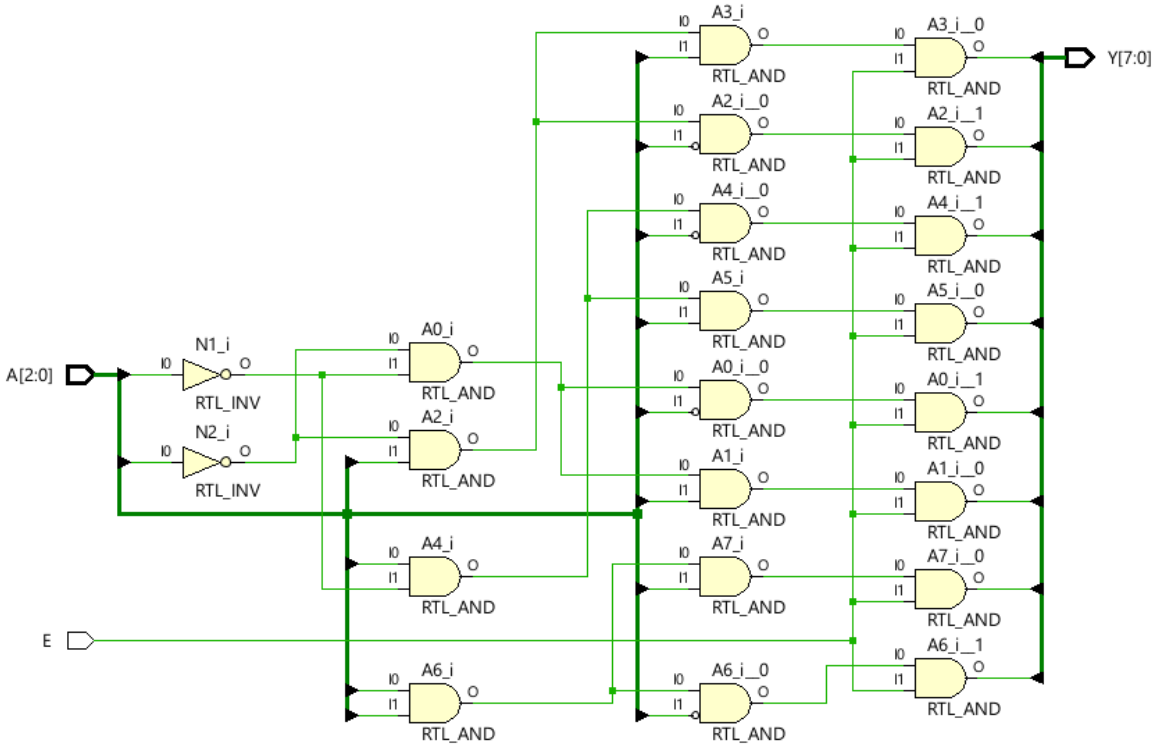
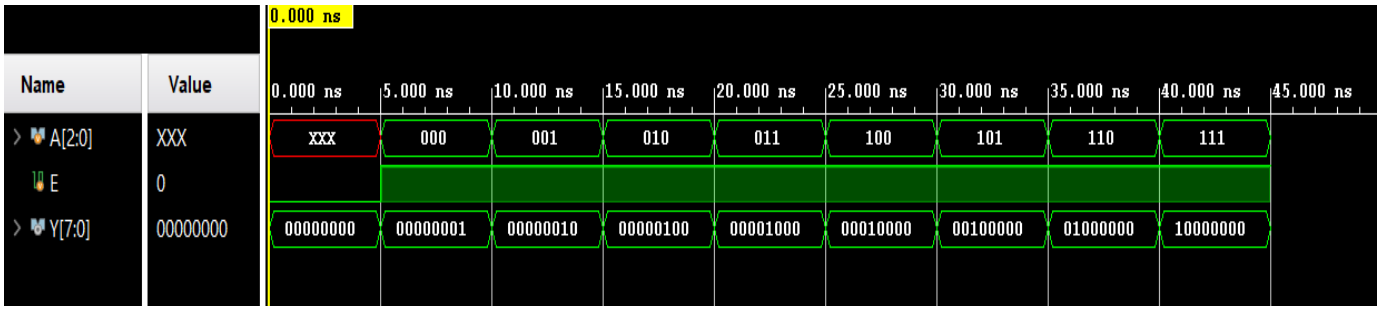


FIG: 3 to 8 Decoder Schematic

Simulation Output:



3. **4 to 16 line decoder** In the 4 to 16 line decoder, there is a total of 16 outputs, i.e., Y_0, Y_1, Y_2,……, Y_{16} and four inputs, i.e., A_0, A1, A_2, and A_3. The 3 to 16 line decoder can be constructed using either 2 to 4 decoder or 3 to 8 decoder.

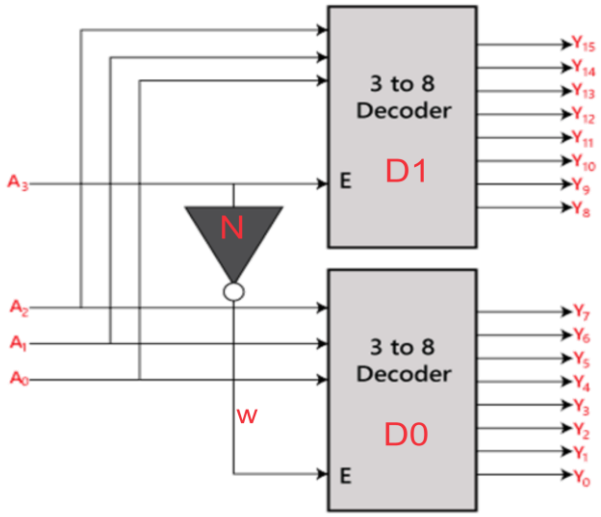


FIG: 4 to 16 line decoder Block Diagram

Truth Table

INPUTS				OUTPUTS															
A_3	A_2	A_1	A_0	Y_{15}	Y_{14}	Y_{13}	Y_{12}	Y_{11}	Y_{10}	Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

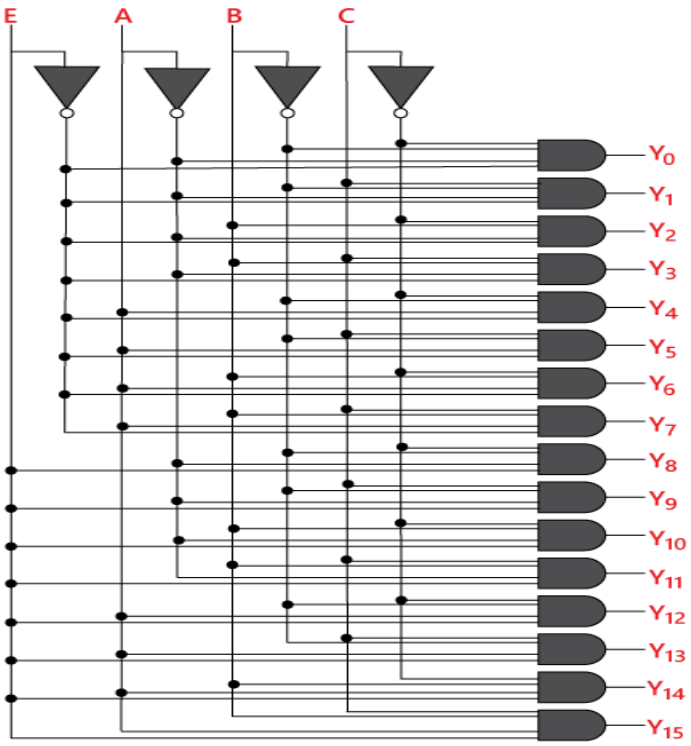


FIG: 4 to 16 line decoder Logic Diagram

Verilog Code:

```
// Verilog Code for 4 to 16 Decoder
// Structural  Style using Decoder3to8 Structure
module Decoder4to16(A,Y);

input [3:0]A;

output [15:0]Y;

Decoder3to8 D1(.A(A[2:0]),.E(A[3]),.Y(Y[15:8]));
wire w;
not N(w,A[3]);
Decoder3to8 D0(.A(A[2:0]),.E(w),.Y(Y[7:0]));
endmodule
```

Testbench Code:

```
// Testbench Code for 4 to 16 Decoder
module Decoder4to16_tb();

reg [3:0]A;
wire [15:0]Y;
integer i;
Decoder4to16 dut(.A(A),.Y(Y));

initial begin
   A=4'b1111;
  for(i=0;i<16;i=i+1)
  begin
  A=A+1;
  #5;
  end
$finish;
end

endmodule
```

Schematic:

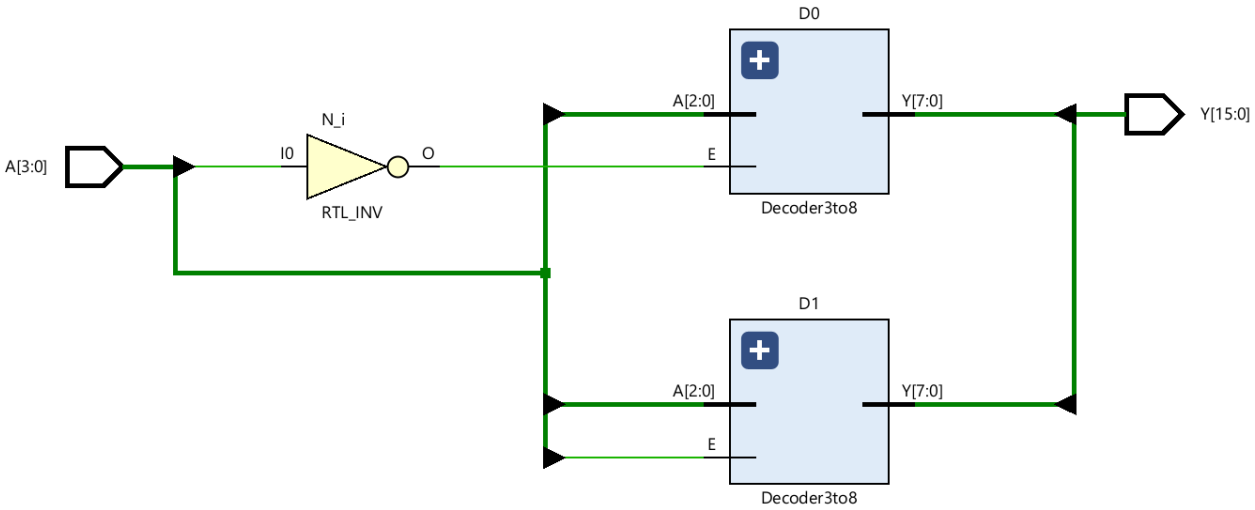


FIG: 4 to 16 Decoder Schematic

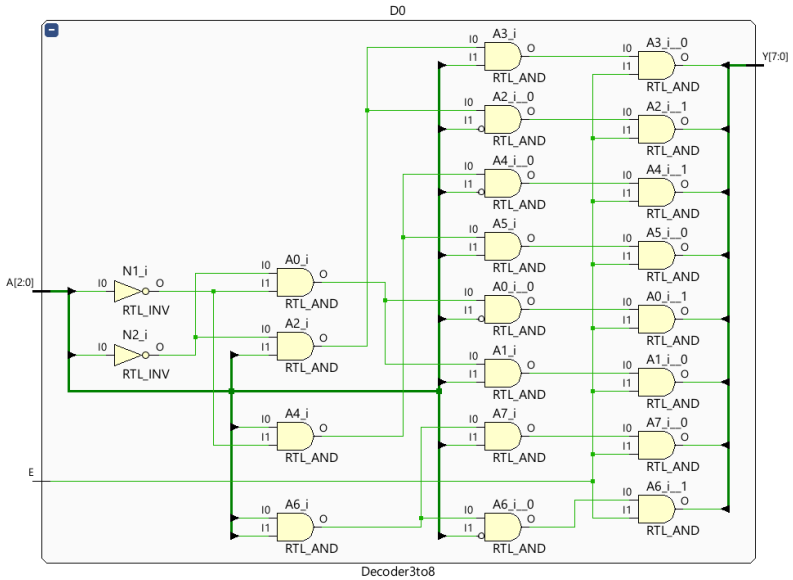
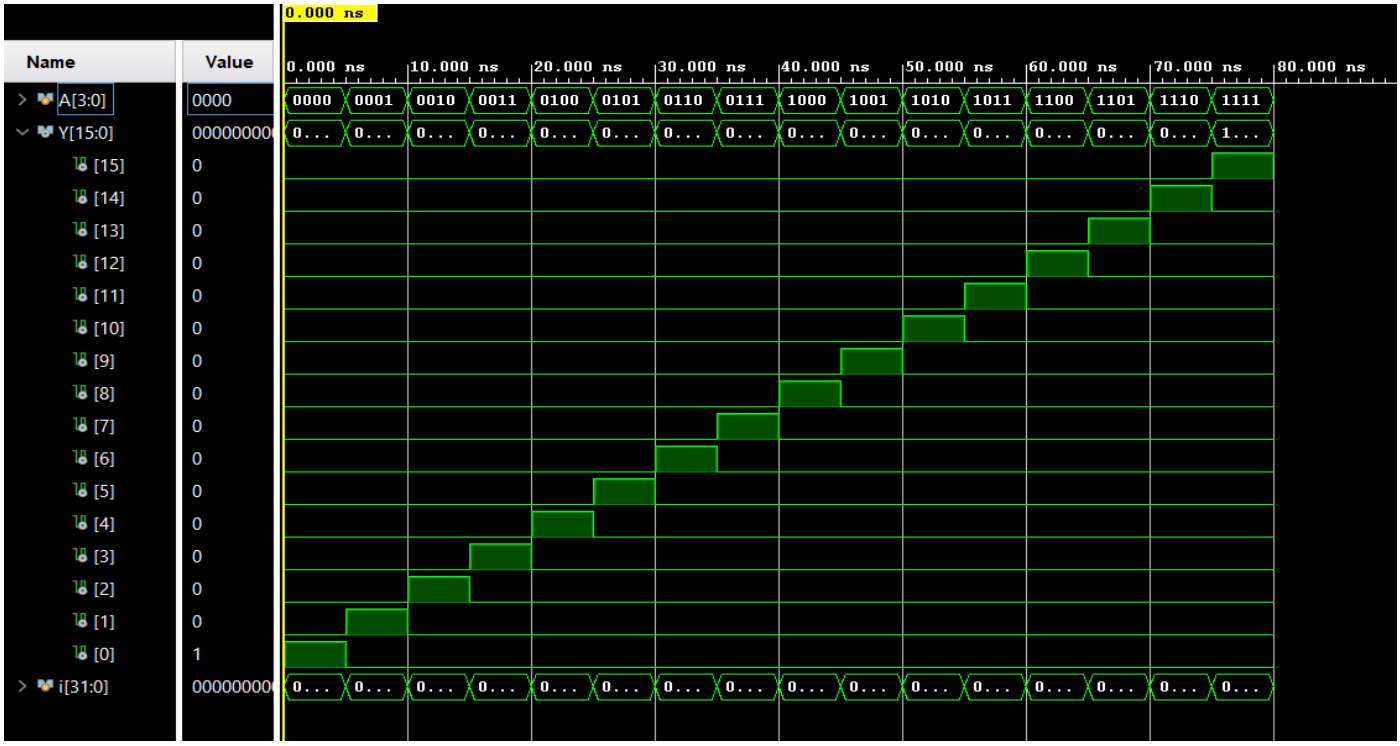


FIG: 3 to 8 Decoder Schematic

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-14-Decoder