ECE 3300L – Lab 2 4x16 Decoder Design on FPGA

By: Raj Gokidi and Priyanka Ravinder

California State Polytechnic University, Pomona

June 25th, 2025

Design

The gate-level implementation of a 4-to-16 decoder constructed using AND and NOT gates provide precise hardware control. In contrast, the behavioral implementation makes it ideal for FPGA development and rapid prototyping. Its dependence on synthesis tools may introduce variability in hardware optimization. For most applications, the behavioral approach is preferred for its simplicity and maintainability, while the gate-level method is better suited for scenarios requiring explicit hardware control.

Gate-Level implementation:

```
module decoder4x16_gate(
input [3:0] A,
input E,
output [15:0] Y
  );

  assign Y[0] = E & ~A[3] & ~A[2] & ~A[1] & ~A[0];
  assign Y[1] = E & ~A[3] & ~A[2] & ~A[1] & A[0];
  assign Y[2] = E & ~A[3] & ~A[2] & A[1] & ~A[0];
  assign Y[3] = E & ~A[3] & ~A[2] & A[1] & A[0];
  assign Y[4] = E & ~A[3] & A[2] & ~A[1] & ~A[0];
  assign Y[5] = E & ~A[3] & A[2] & ~A[1] & A[0];
  assign Y[6] = E & ~A[3] & A[2] & A[1] & ~A[0];
  assign Y[7] = E & ~A[3] & A[2] & A[1] & A[0];
  assign Y[8] = E & A[3] & ~A[2] & ~A[1] & ~A[0];
  assign Y[9] = E & A[3] & ~A[2] & ~A[1] & A[0];
  assign Y[10] = E & A[3] & ~A[2] & A[1] & ~A[0];
```

```
    assign Y[11] = E & A[3] & ~A[2] & A[1] & A[0];
    assign Y[12] = E & A[3] & A[2] & ~A[1] & ~A[0];
    assign Y[13] = E & A[3] & A[2] & ~A[1] & A[0];
    assign Y[14] = E & A[3] & A[2] & A[1] & ~A[0];
    assign Y[15] = E & A[3] & A[2] & A[1] & A[0];
endmodule
```

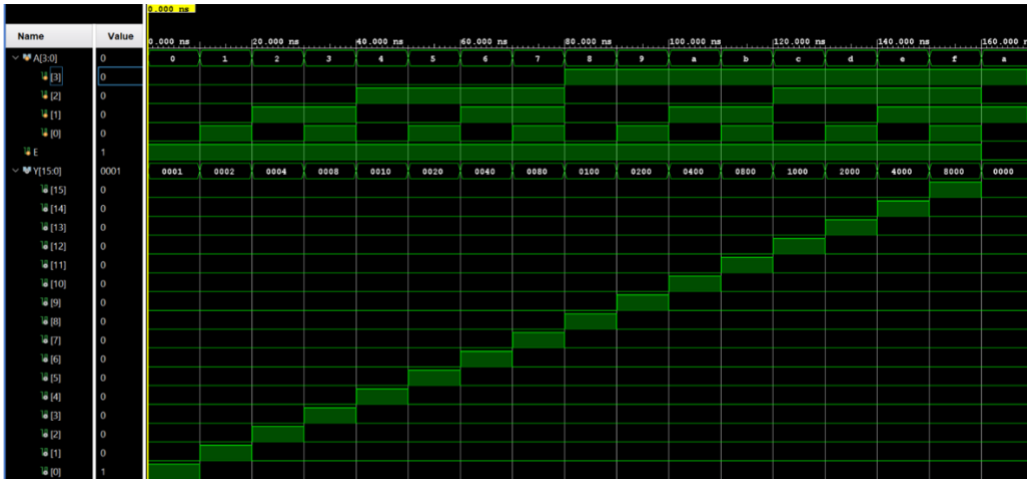
Behavior implementation:

```
module decoder4x16_behav(
input [3:0] A,
input E,
output reg [15:0] Y
    );
    always @(*) begin
    Y = 16'b0;
    if (E) begin
        case (A)
        4'b0000: Y = 16'b0000_0000_0000_0001;
        4'b0001: Y = 16'b0000_0000_0000_0010;
        4'b0010: Y = 16'b0000_0000_0000_0100;
        4'b0011: Y = 16'b0000_0000_0000_1000;
        4'b0100: Y = 16'b0000_0000_0001_0000;
        4'b0101: Y = 16'b0000_0000_0010_0000;
        4'b0110: Y = 16'b0000_0000_0100_0000;
        4'b0111: Y = 16'b0000_0000_1000_0000;
```

```
      4'b1000: Y = 16'b0000_0001_0000_0000;
      4'b0001: Y = 16'b0000_0010_0000_0000;
      4'b1010: Y = 16'b0000_0100_0000_0000;
      4'b1011: Y = 16'b0000_1000_0000_0000;
      4'b1100: Y = 16'b0001_0000_0000_0000;
      4'b1101: Y = 16'b0010_0000_0000_0000;
      4'b1110: Y = 16'b0100_0000_0000_0000;
      4'b1111: Y = 16'b1000_0000_0000_0000;
    endcase
  end
 end
endmodule
```

Testbench

The self-checking testbench in tb_decoder4x16.v verifies the 4-to-16 decoder, implemented behaviorally, by setting E = 0 and A = 4'b0000 to confirm all outputs Y[15:0] are low, then cycling A from 4'b0000 to 4'b1111 with E = 1 to ensure only one output Y[i] is high, matching the expected value (1 << i). It uses a for loop with 10ns delays to test each input, checking Y against the expected single-bit-high pattern, with $display for PASS/FAIL messages and $fatal stopping on errors. The waveform shows Y as 16'h0000 when E = 0, then a single bit high for each A (e.g., Y = 16'h0001 for A = 4'b0000, up to Y = 16'h8000 for A = 4'b1111) with clean transitions. All tests passing confirms correct decoder operation.



Implementation

Resource	Utilization	Available	Utilization %
LUT	8	63400	0.01
IO	21	210	10.00

Contributions

Raj Gokidi (016331965): 50% demo and testbench

Priyanka Ravinder (015977169): 50% gate level and behavior Verilog code

Video Link: https://youtu.be/e53nYGr-sgI