California Polytechnic State University Pomona

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Dgtl Circuit Dsgn Verilog Lab

ECE 3300L Section E01

Report # 2

Prepared by

Arvin Ghaloosian (017153422)
Vittorio Huizar (016826784)
Group H

Presented to

Mohamed Aly (Mohamed El-Hadedy)

Due - June 25th, 2025

GATE VS BEHAVIORAL IMPLEMENTATION:
Gate Code:

```
module decoder4x16_gate(
    input  [3:0] A,  // 4 bit input
    input        E,  // Enable
    output [15:0] Y  // 16 bit output
);

// Invert inputs
wire nA0 = ~A[0];
wire nA1 = ~A[1];
wire nA2 = ~A[2];
wire nA3 = ~A[3];

assign Y[0]  = E &  nA3 &  nA2 &  nA1 &  nA0;
assign Y[1]  = E &  nA3 &  nA2 &  nA1 &  A[0];
assign Y[2]  = E &  nA3 &  nA2 &  A[1] &  nA0;
assign Y[3]  = E &  nA3 &  nA2 &  A[1] &  A[0];
assign Y[4]  = E &  nA3 &  A[2] &  nA1 &  nA0;
assign Y[5]  = E &  nA3 &  A[2] &  nA1 &  A[0];
assign Y[6]  = E &  nA3 &  A[2] &  A[1] &  nA0;
assign Y[7]  = E &  nA3 &  A[2] &  A[1] &  A[0];
assign Y[8]  = E &  A[3] &  nA2 &  nA1 &  nA0;
assign Y[9]  = E &  A[3] &  nA2 &  nA1 &  A[0];
assign Y[10] = E &  A[3] &  nA2 &  A[1] &  nA0;
assign Y[11] = E &  A[3] &  nA2 &  A[1] &  A[0];
assign Y[12] = E &  A[3] &  A[2] &  nA1 &  nA0;
assign Y[13] = E &  A[3] &  A[2] &  nA1 &  A[0];
assign Y[14] = E &  A[3] &  A[2] &  A[1] &  nA0;
assign Y[15] = E &  A[3] &  A[2] &  A[1] &  A[0];

endmodule
```

Advantages:

- Direct mapping to hardware gates
- Clear visualization of the boolean logic
- Direct control over each output

Disadvantages:

- Verbose and repetitive code
- Higher chance of error
- More difficulties to scale

Behavioral Code:

```
`timescale 1ns / 1ps

module decoder4x16_behav(
    input  [3:0] A,
    input        E,
    output reg [15:0] Y
);

always @(*) begin
    Y = 16'b0;                 // ▶ RESET: clear all outputs to 0
    if (E) begin               // ▶ ENABLE: only decode when enabled
        case (A)
            4'b0000: Y = 16'b0000_0000_0000_0001; // ▶ OUTPUT: activate Y[0]
            4'b0001: Y = 16'b0000_0000_0000_0010; // ▶ OUTPUT: activate Y[1]
            4'b0010: Y = 16'b0000_0000_0000_0100; // ▶ OUTPUT: activate Y[2]
            4'b0011: Y = 16'b0000_0000_0000_1000; // ▶ OUTPUT: activate Y[3]
            4'b0100: Y = 16'b0000_0000_0001_0000; // ▶ OUTPUT: activate Y[4]
            4'b0101: Y = 16'b0000_0000_0010_0000; // ▶ OUTPUT: activate Y[5]
            4'b0110: Y = 16'b0000_0000_0100_0000; // ▶ OUTPUT: activate Y[6]
            4'b0111: Y = 16'b0000_0000_1000_0000; // ▶ OUTPUT: activate Y[7]
            4'b1000: Y = 16'b0000_0001_0000_0000; // ▶ OUTPUT: activate Y[8]
            4'b1001: Y = 16'b0000_0010_0000_0000; // ▶ OUTPUT: activate Y[9]
            4'b1010: Y = 16'b0000_0100_0000_0000; // ▶ OUTPUT: activate Y[10]
            4'b1011: Y = 16'b0000_1000_0000_0000; // ▶ OUTPUT: activate Y[11]
            4'b1100: Y = 16'b0001_0000_0000_0000; // ▶ OUTPUT: activate Y[12]
            4'b1101: Y = 16'b0010_0000_0000_0000; // ▶ OUTPUT: activate Y[13]
            4'b1110: Y = 16'b0100_0000_0000_0000; // ▶ OUTPUT: activate Y[14]
            4'b1111: Y = 16'b1000_0000_0000_0000; // ▶ OUTPUT: activate Y[15]
        endcase
    end
end

endmodule
```

Advantages:

- More readable and maintainable
- Scales better to more complicated designs
- Easier to verify

Disadvantages:

- Less explicit about the underlying hardware
- Requires synthesis interpretation

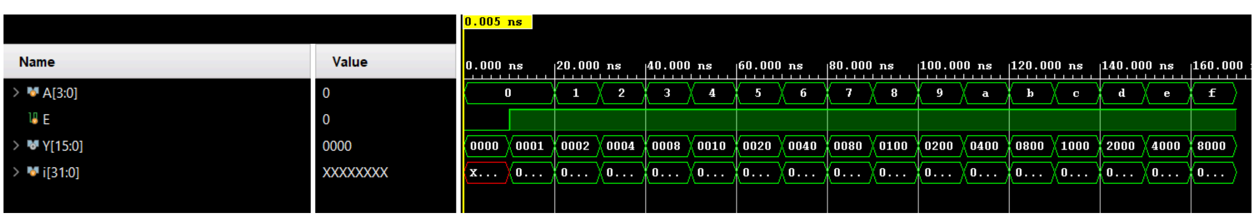
XDC Snippet:

```
# A[0]
set_property PACKAGE_PIN J15 [get_ports {A[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[0]}]

# A[1]
set_property PACKAGE_PIN L16 [get_ports {A[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[1]}]

# A[2]
set_property PACKAGE_PIN M13 [get_ports {A[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[2]}]

# A[3]
set_property PACKAGE_PIN R15 [get_ports {A[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {A[3]}]

# Enable (E)
set_property PACKAGE_PIN R17 [get_ports {E}]
set_property IOSTANDARD LVCMOS33 [get_ports {E}]

## OUTPUT LEDS

# Y[0] - LED0
set_property PACKAGE_PIN H17 [get_ports {Y[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[0]}]

# Y[1] - LED1
set_property PACKAGE_PIN K15 [get_ports {Y[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[1]}]

# Y[2] - LED2
set_property PACKAGE_PIN J13 [get_ports {Y[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[2]}]

# Y[3] - LED3
set_property PACKAGE_PIN N14 [get_ports {Y[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[3]}]

# Y[4] - LED4
set_property PACKAGE_PIN R18 [get_ports {Y[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[4]}]

# Y[5] - LED5
set_property PACKAGE_PIN V17 [get_ports {Y[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[5]}]

# Y[6] - LED6
set_property PACKAGE_PIN U17 [get_ports {Y[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[6]}]

# Y[7] - LED7
set_property PACKAGE_PIN U16 [get_ports {Y[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[7]}]
```

```
# Y[8] - LED8
set_property PACKAGE_PIN V16 [get_ports {Y[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[8]}]

# Y[9] - LED9
set_property PACKAGE_PIN T15 [get_ports {Y[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[9]}]

# Y[10] - LED10
set_property PACKAGE_PIN U14 [get_ports {Y[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[10]}]

# Y[11] - LED11
set_property PACKAGE_PIN T16 [get_ports {Y[11]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[11]}]

# Y[12] - LED12
set_property PACKAGE_PIN V15 [get_ports {Y[12]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[12]}]

# Y[13] - LED13
set_property PACKAGE_PIN V14 [get_ports {Y[13]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[13]}]

# Y[14] - LED14
set_property PACKAGE_PIN V12 [get_ports {Y[14]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[14]}]

# Y[15] - LED15
set_property PACKAGE_PIN V11 [get_ports {Y[15]}]
set_property IOSTANDARD LVCMOS33 [get_ports {Y[15]}]
```

TEST BENCH:

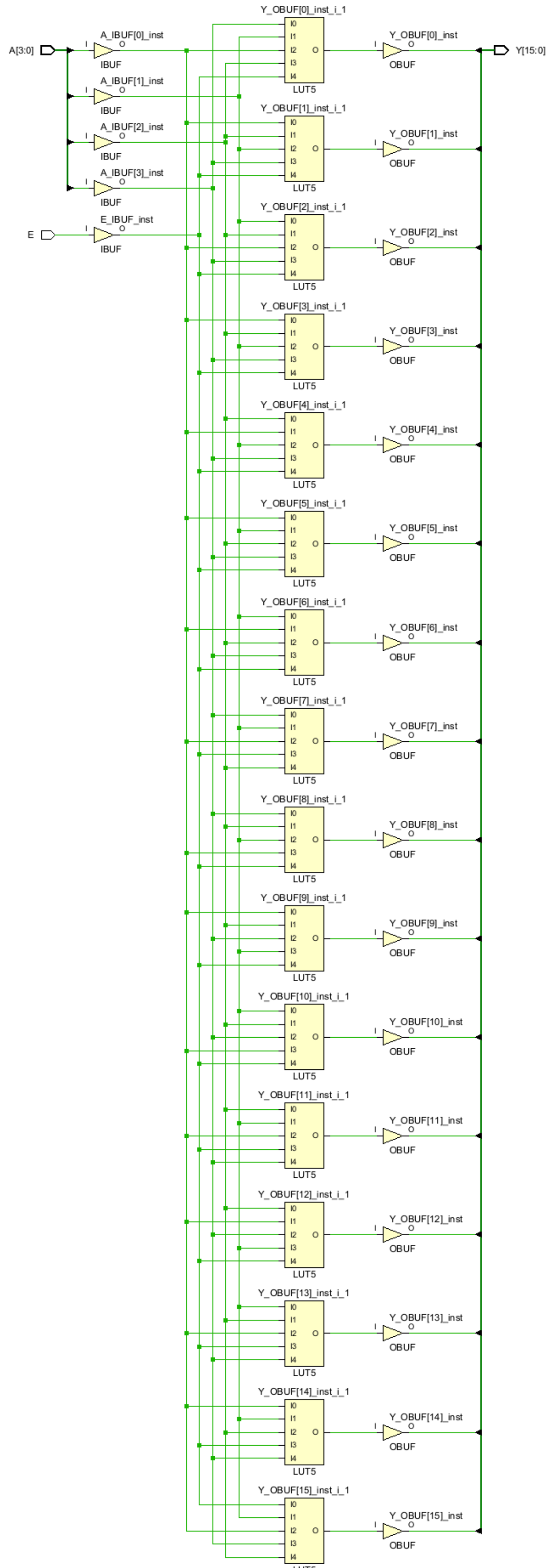
```
`timescale 1ns / 1ps

module tb_decoder4x16;

reg  [3:0] A;
reg        E;
wire [15:0] Y;

// DUT: Choose one of the following:

// Uncomment to test GATE-LEVEL version:
// decoder4x16_gate dut ( .A(A), .E(E), .Y(Y) );

// Uncomment to test BEHAVIORAL version:
decoder4x16_behav dut ( .A(A), .E(E), .Y(Y) );

integer i;

initial begin
  $display("---- Starting 4x16 Decoder Test ----");

  // Test Enable OFF
  E = 0;
  A = 4'b0000;
  #10;
  if (Y !== 16'b0) begin
    $display("FAIL: Enable OFF test failed. Y = %b", Y);
    $fatal;
  end else begin
    $display("PASS: Enable OFF outputs all zero");
  end

  // Enable ON
  E = 1;
  for (i = 0; i < 16; i = i + 1) begin
    A = i;
    #10;
    if (Y !== (1 << i)) begin
      $display("FAIL: A = %b, Expected Y = %b, Got Y = %b", A, (1 << i), Y);
      $fatal;
    end else begin
      $display("PASS: A = %b, Y = %b", A, Y);
    end
  end

  $display("---- ALL TESTS PASSED ----");
  $finish;
end

endmodule
```



```
Time resolution is 1 ps
---- Starting 4x16 Decoder Test ----
PASS: Enable OFF outputs all zero
PASS: A = 0000, Y = 0000000000000001
PASS: A = 0001, Y = 0000000000000010
PASS: A = 0010, Y = 0000000000000100
PASS: A = 0011, Y = 0000000000001000
PASS: A = 0100, Y = 0000000000010000
PASS: A = 0101, Y = 0000000000100000
PASS: A = 0110, Y = 0000000001000000
PASS: A = 0111, Y = 0000000010000000
PASS: A = 1000, Y = 0000000100000000
PASS: A = 1001, Y = 0000001000000000
PASS: A = 1010, Y = 0000010000000000
PASS: A = 1011, Y = 0000100000000000
PASS: A = 1100, Y = 0001000000000000
PASS: A = 1101, Y = 0010000000000000
PASS: A = 1110, Y = 0100000000000000
PASS: A = 1111, Y = 1000000000000000
---- ALL TESTS PASSED ----
$finish called at time : 170 ns : File "D:/verilog/ece33001/Group H/Lab2/tb/tb_decoder4x16.v" Line 69
```

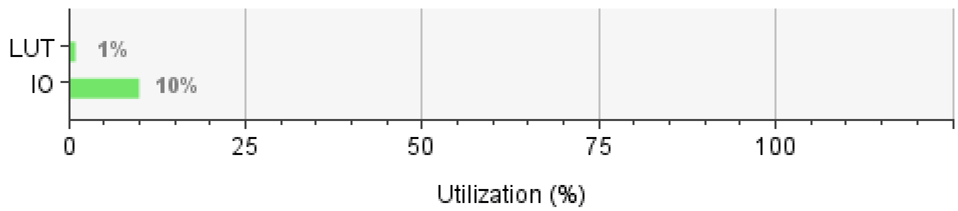
Developing this self checking testbench was initially challenging but helped us test our code a lot faster once it was written. The testbench helped catch a few early mistakes in the case statement mappings. We also realize how much faster it is to test iterations of the code with a self checking testbench, especially as projects become more complex.

SYNTHESIS SCHEMATIC:



Resource Utilization:

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



LUT Usage: 8 LUTs required, indicating efficient combinational logic implementation
I/O Usage: 21 pins = 4 inputs + 1 enable + 16 outputs
No Registers: For purely combinational logic, no flip-flops needed

CONTRIBUTIONS:

Arvin Ghaloosian (50%)

- Developed gate-level Verilog implementation
- Created and debugged self-checking testbench
- Conducted simulation verification and waveform analysis
- Performed FPGA implementation and timing analysis
- Contributed to report writing and documentation

Vittorio Huizar (50%)

- Developed behavioral Verilog implementation with color-coded comments
- Set up Vivado project and constraint file configuration
- Conducted hardware testing and demo video recording
- Analyzed resource utilization and synthesis reports
- Contributed to report writing and final presentation

Reflection:

In this lab, we successfully implemented a 4x16 decoder at a gate and behavioral level. Working with both coding styles showed the different approaches to describing hardware functionality. For example, the gate level implementation required us to consider the underlying boolean logic, writing explicitly AND and NOT gate combinations for each output. This resulted in more verbose code. In contrast to that the behavioral implementation using always@(*) blocks provided cleaner code and better maintainability.

Comments:

- We were not sure how to add color-coded comments to our verilog code in vivado so instead we opted for a clean and clear “▸ RESET”, “▸ ENABLE”, “▸ OUTPUT”, as shown in your instructions.

Demo Video:

https://youtu.be/X0b2ricZJEE