Diane Shan

862148900

EE120A Section 23

Jack Huang

Lab 2 – Decoders and Muxes

**Overview**

There were two parts to this lab. In the first part, we designed and tested a 3 x 8 decoder with an enable switch for a sprinkler valve controller system. For our system, we dealt only with opening and closing of valves. We based our system on a 3 x 8 decoder, analyzed the truth table and got the function. Then we implemented it as a gate-based logic circuit and implemented the system in verilog. In the second part of the lab, we designed a 4 x 1 multiplexer that controls the flow of data with a single wire data bus. This makes it possible to use only one output pin of a microcontroller to have partial serial communication with multiple peripheral devices.

**New Concepts**

Decoder – combinational logic circuit that converts binary information from n coded inputs to a maximum of 2n unique outputs

Microcontroller Unit (MCU) – a small computer on a single metal-oxide-semiconductor integrated circuit chip. Contains one or more CPUs along with memory and programmable input/output peripherals.

Multiplexer – also known as a data selector, is a device that selects between several analog or digital input signals and forwards the selected input to a single output line.

Input/Output Pin – interface between a microcontroller and another circuit

**Analysis**

Part 1

Procedure:

1. Analyze and design a system based on a 3 x 8 decoder with an enable switch
2. Construct a truth table for the system
3. Capture the function
4. Convert to equations and/or minimize the functions
5. Implement as a gate-based logic circuit
6. Implement the behavioral and the structural modules in Verilog
7. Then do exhaustive circuit testing by copying the code into the testbench
8. Click “Run” to make sure it passes all the tests

Truth Table of Sprinkler System (‘x’ stands for “don’t care”)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | A | B | C | d0 | d1 | d2 | d3 | d4 | d5 | d6 | d7 |
| 0 | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Boolean Equation:

d0 = E . A’ . B’ . C’

d1 = E . A’ . B’ . C

d2 = E . A’ . B . C’

d3 = E . A’ . B . C

d4 = E . A . B’ . C’

d5 = E . A . B’ . C

d6 = E . A . B . C’

d7 = E . A . B . C

Part 2

Procedure:

1. Design logic circuit that will control which of the input data is present in output d
2. Implement the behavioral and structural verilog model in EDA playground
3. Implement the testbench to test the behavioral and structural models.
4. Generate the waveform
5. Make truth table, algebraic expression of the logic function, and logic circuit schematic

Truth Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| s1 | s0 | i0 | i1 | i2 | i3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

Boolean Equation

i0 = s1’+ s0’

i1 = s1’+ s0

i2 = s1 + s0’

i3 = s1 + s0

**Records**

Part 1:

Diagram

Description automatically generated

Logic circuit schematic of sprinkler system

design.sv

module and4(

input wire enable ,

input wire a,

input wire b,

input wire c,

output wire r

);

//

assign r = enable & a & b & c ;

endmodule

// structural model

module decoder\_st(

// I/0 ports

input wire enable ,

input wire a ,

input wire b ,

input wire c ,

output wire d0,

output wire d1,

output wire d2,

output wire d3,

output wire d4,

output wire d5,

output wire d6,

output wire d7

);

// Using the and4 module to set all outputs

and4 c1(enable, ~a, ~b, ~c, d0) ;

// Your code goes here (7 cases left to implement)

and4 c2(enable, ~a, ~b, c, d1) ;

and4 c3(enable, ~a, b, ~c, d2) ;

and4 c4(enable, ~a, b, c, d3) ;

and4 c5(enable, a, ~b, ~c, d4) ;

and4 c6(enable, a, ~b, c, d5) ;

and4 c7(enable, a, b, ~c, d6) ;

and4 c8(enable, a, b, c, d7) ;

endmodule

// behavioral model

module decoder\_bh(

// I/0 ports

input wire enable ,

input wire a ,

input wire b ,

input wire c ,

output reg d0,

output reg d1,

output reg d2,

output reg d3,

output reg d4,

output reg d5,

output reg d6,

output reg d7

);

// Internal wire

wire [3:0] bundle ;

assign bundle = {enable , a, b, c } ;

// Behavioral description

always @(\*) begin

d0 = 1'b0 ;

d1 = 1'b0 ;

d2 = 1'b0 ;

d3 = 1'b0 ;

d4 = 1'b0 ;

d5 = 1'b0 ;

d6 = 1'b0 ;

d7 = 1'b0 ;

// Setting the correct output

case (bundle)

4'b1000: d0 = 1'b1 ;

4'b1001: d1 = 1'b1 ;

4'b1010: d2 = 1'b1 ;

4'b1011: d3 = 1'b1 ;

4'b1100: d4 = 1'b1 ;

4'b1101: d5 = 1'b1 ;

4'b1110: d6 = 1'b1 ;

4'b1111: d7 = 1'b1 ;

default : begin

d0 = 1'b0 ;

end

endcase

end

endmodule

testbench.sv

module decoder\_tb;

// Inputs

reg enable;

reg a;

reg b;

reg c;

// Outputs

wire d0;

wire d1;

wire d2;

wire d3;

wire d4;

wire d5;

wire d6;

wire d7;

// Instantiate the Unit Under Test (UUT)

decoder\_st uut ( // change to “decoder\_bh” for testing your behavioral model

.enable(enable),

.a(a),

.b(b),

.c(c),

.d0(d0),

.d1(d1),

.d2(d2),

.d3(d3),

.d4(d4),

.d5(d5),

.d6(d6),

.d7(d7)

);

initial begin

$dumpfile("dump.vcd"); $dumpvars;

enable = 1;

a = 0;

b = 0;

c = 0;

#100; // Wait for 100 ns

$display("TC11 ");

if ( d0 != 1'b1 ) $display ("Result is wrong");

a = 0;

b = 0;

c = 1;

#100;

$display("TC12 ");

if ( d1 != 1'b1 ) $display ("Result is wrong");

a = 0;

b = 1;

c = 0;

#100;

$display("TC13 ");

if ( d2 != 1'b1 ) $display ("Result is wrong");

a = 0;

b = 1;

c = 1;

#100;

$display("TC14 ");

if ( d3 != 1'b1 ) $display ("Result is wrong");

a = 1;

b = 0;

c = 0;

#100;

$display("TC15 ");

if ( d4 != 1'b1 ) $display ("Result is wrong");

a = 1;

b = 0;

c = 1;

#100;

$display("TC16 ");

if ( d5 != 1'b1 ) $display ("Result is wrong");

a = 1;

b = 1;

c = 0;

#100;

$display("TC17 ");

if ( d6 != 1'b1 ) $display ("Result is wrong");

a = 1;

b = 1;

c = 1;

#100;

$display("TC18 ");

if ( d7 != 1'b1 ) $display ("Result is wrong");

end

endmodule

*Simulation Results (waveform):*

Structural

A picture containing text, electronics

Description automatically generated

Behavioral

A picture containing text, electronics

Description automatically generated

Part 2:



Logic circuit schematic

design.sv

module and3(

input wire a,

input wire b,

input wire c,

output wire r

);

assign r = a & b & c ;

endmodule

// structural model

module mux\_st(

// Ports I/O

input wire s1,

input wire s0,

input wire i0,

input wire i1,

input wire i2,

input wire i3,

output wire d

);

wire r1, r2, r3, r4 ;

and3 c1 ( ~s1,~s0, i0, r1 ) ;

// Your code goes here (3 cases left)

and3 c2 ( ~s1, s0, i1, r2 ) ;

and3 c3 ( s1, ~s0, i2, r3 ) ;

and3 c4 ( s1, s0, i3, r4 ) ;

assign d = r1 | r2 | r3 | r4 ;

endmodule

// behavioral model

module mux\_bh(

// Ports I/O

input wire s1,

input wire s0,

input wire i0,

input wire i1,

input wire i2,

input wire i3,

output reg d

) ;

always @(\*) begin

d = 1'b0 ;

case ( {s1,s0} )

2'b00 : d = i0 ;

// your code goes here (3 cases left)

2'b01 : d = i1 ;

2'b10 : d = i2 ;

2'b11 : d = i3 ;

endcase

end

endmodule

testbench.sv

module mux\_tb;

// Inputs

reg s1;

reg s0;

reg i0;

reg i1;

reg i2;

reg i3;

// Outputs

wire d;

// Instantiate the Unit Under Test (UUT)

mux\_bh uut ( // change to “mux\_st” for testing your structural model

.s1(s1),

.s0(s0),

.i0(i0),

.i1(i1),

.i2(i2),

.i3(i3),

.d(d)

);

initial begin

$dumpfile("dump.vcd"); $dumpvars;

i0 = 1;

i1 = 0;

i2 = 1;

i3 = 0;

s1 = 0;

s0 = 0;

#100;

$display("TC11 ");

if ( d != i0 ) $display ("Result is wrong");

s1 = 0;

s0 = 1;

#100;

$display("TC12 ");

if ( d != i1 ) $display ("Result is wrong");

s1 = 1;

s0 = 0;

#100;

$display("TC13 ");

if ( d != i2 ) $display ("Result is wrong");

s1 = 1;

s0 = 1;

#100;

$display("TC14 ");

if ( d != i3 ) $display ("Result is wrong");

// Your test cases

end

endmodule

*Simulation Results (waveform):*

Behavioral

Graphical user interface

Description automatically generated

Structural

A screenshot of a computer

Description automatically generated with medium confidence

**Discussion**

The system works according to the provided specifications for both parts of the lab. Following the diagram and adding more cases to test, we pass all the tests in both part 1 and part 2. We added code for the structural and behavioral models. Both the structural and behavioral system for the two designs works as expected. There were no problems or technical issues encountered and the system is good as is since it easily passes all the tests. There are not really any improvements to be made to this system.

**Conclusion**

For both parts of the lab, we went through the whole process of system design by doing analysis and circuit logic behavioral verification. This helped to get us more familiar with the EDA playground environment and we learned about simulation and design of controller systems based on combinational logic. We got more familiar with generating testbenches for logic design testing and verification and generated waveforms, verifying they were correct. We also learned more about combinational logic circuits and truth tables. The results revealed that the designs we made passed all the testbenches and the waveforms matched the truth tables.

**Questions**

Part 1:

1. A waveform is a visual representation of the variation of a voltage or current over time.
2. A testbench is an environment used to verify a design or model’s correctness or soundness.
3. We can replace the 4-input AND gates in the circuit with the 2-input AND gates by having 2-input AND gates connecting every input. For example, E and A will first be connected by an AND gate and then B and C will be connected by an AND gate, then they will all be connected at the end with another 2-input AND gate. The result is just, the schematic will have a lot of AND gates.

Part 2:

Explain why waveform obtained during behavioral simulation is correct.

The waveform obtained during behavioral simulation is correct because it produces the correct output and therefore passes the testbench. This means the design functions correctly and as expected. The waveform output matches the values in the truth table.