PROJECT REPORT

FUNDAMENTAL DIGITAL SYSTEMS | LEARNING EXPERIENCE B

Author : Barakaeli Lawuo (February, 2024)

1. Project Objectives

The objective of this project is to design, simulate, and implement a multiple-output combinational logic circuit that drives a seven-segment LED display on a DE10-Lite CD board based on a specified character set encoded in a four-bit binary code.

2. Requirements & Constraints

According to the assignment description below are the requirements and constraints considered while implementing the project.

- Design a character set comprising a minimum of nine and a maximum of thirteen characters.
  - The character set had 11 characters which were meant to read as "goAt drones 1".
  - There are two 'o' characters in the set but they are going to have different implementation thus each one being unique
- Each character must be represented by a unique four-bit binary code.
  - Each character was represented by a unique binary code as shown in the table below.

0000	0001	0010	0011	0100
g	o	A	t	d
0101	0110	0111	1000	1001
r	o	n	e	S
1010				
1				

- Every segment (except DP) of the seven-segment display must light at least once and remain dark at least once.
  - The Decimal Point(DP) was not considered in the implementation. The frequency of the segment being on or off at least once was considered as shown on the table below.

Character	# times ON	# times OFF
g	6	1
o	4	3
A	6	1
t	4	3
d	5	2
r	2	5
o	4	3
n	3	4
e	6	1
S	5	2
1	2	5

- The code 0000 must be used for one character.
  - Code 0000 was used for the character "g"
- Implement the circuit using structural Verilog constructs.
  - Structural Verilog was used to implement the design and a file named ssd.v containing the code will be submitted as an attachment with this project report.

- Minimize transistor count through technology mapping while considering optimization criteria such as circuit propagation delay.
  - Later on in this report, an extensive explanation of how transistor count was minimized will be explained using K-Maps.

3. Design Approach

For the character set design, I designed a set of characters to display on the seven-segment LED display. I ensured that each character was unique, covered all segments of the display, and had the appropriate assignment of four-bit binary codes as shown in the previous section of **Requirements & Constraints.**

Next, I derived minimal logic equations for each driver circuit based on the character set. I used Karnaugh maps to simplify these logic equations effectively. This will be explained in the **4th section** of the report.

In the technology mapping phase, I selected appropriate Verilog primitive gates, in the verilog code only nand and not gates were used to minimize the transistor count while optimizing the design. I considered criteria such as transistor count and circuit propagation delay to make informed decisions. This will be explained further in the **5th section**

4. **Equations Derived**

Consider the truth table for the implementation as shown below. I highlighted the high/on signal to make it easier in creating logic expressions using Sums of Products.

A	B	C	D	0	1	2	3	4	5	6	
0	0	0	0	0	0	0	0	1	0	0	g
0	0	0	1	0	0	1	1	1	0	0	o
0	0	1	0	0	0	0	1	0	0	0	A
0	0	1	1	1	1	1	0	0	0	0	t
0	1	0	0	1	0	0	0	0	1	0	d
0	1	0	1	1	1	1	1	0	1	0	r
0	1	1	0	1	1	0	0	0	1	0	o
0	1	1	1	1	1	0	1	0	1	0	n
1	0	0	0	0	0	1	0	0	0	0	e
1	0	0	1	0	1	0	0	1	0	0	S
1	0	1	0	1	0	0	1	1	1	1	1
1	0	1	1	X	X	X	X	X	X	X	
1	1	0	0	X	X	X	X	X	X	X	
1	1	0	1	X	X	X	X	X	X	X	
1	1	1	0	X	X	X	X	X	X	X	
1	1	1	1	X	X	X	X	X	X	X	

From the truth table, I obtained the following logic equations. Note below, do not confuse A-G to inputs, it is simply a way of naming the logic equations.Also, the RHS of each logic expression contains it's output

A. A'B'CD + A'BC'D' + A'BC'D + A'BCD'+A'BCD + AB'CD'= 0
B. A'B'CD + A'BC'D +A'BCD' +A'BCD + AB'C'D = 1
C. A'B'C'D + A'B'CD + A'BC'D + AB'C'D' = 2
D. A'B'C'D+ A'B'CD' + A'BC'D + AB'CD' = 3
E. A'B'C'D' + A'B'C'D + AB'C'D + AB'CD' = 4
F. A'BC'D' +A'BC'D +A'BCD' + A'BCD + AB'CD' = 5
G. AB'CD' = 6

Consider the table below that shows the transistor count. Do note the following things, the highlighted logic gates in blue, represents a logic gate whose transistors have been counted already therefore its count will not be considered, another thing is the transistor count for the not gate is

just for the inverse of the inputs. We will be not considering throughout the expressions as we will be repeating the transistor count.

At the beginning, we hand a total of 156 transistors as shown by the table below

NOT		AND		OR		OUTPUT
Term	# Transistors	Term	# Transistors	Term	# Transistors	
A'	2	AB'CD'	8			
B'	2	A'B'CD	8			
C'	2	A'BC'D'	8			
D'	2	A'BC'D	8			
		A'BCD'	8	Sum of all AND gates	14	0
		A'BCD	8			
		A'B'CD	0			
		A'BC'D	0			
		A'BCD'	0			
		A'BCD	0	Sum of all AND gates	12	1
		AB'C'D	8			
		A'B'C'D	0			
		A'B'CD	8			
		A'BC'D	8	Sum of all AND gates	10	2
		AB'C'D'	0			
		A'B'C'D	0			
		A'B'CD'	0			
		A'BC'D	0	Sum of all AND gates	10	3
		AB'CD'	0			
		A'B'C'D'	8			
		A'B'C'D	0			
		AB'C'D	0	Sum of all AND gates	10	
		AB'CD'	0			

			A’BC’D’	0				
			A’BC’D	0				
			A’BCD’	0				
			A’BCD	0	Sum of all AND			
			AB’CD’	0	Gates	12		
			AB’CD’	0		0		Grand Total
Total		8		80		68		156

5. Optimizing Equations

In Optimizing the equations, I first began by obtaining the K-Maps of logic equations . Using this website linked here. After that, I used De Morgan's Laws to Map the logic equations to NAND. Table below shows the logic expressions after K-Map and after mapping to NANDs

Output	After K-Map	After NANDs
0	B+CD+AC	((B)'(CD)'(AC)')'
1	CD+BD+BC+AD	((CD)'(BD)'(BC)'(AD)')'
2	A'B'D+A'C'D+AC'D'	((A'B'D)'(A'C'D)'(AC'D')')'
3	BD+A'C'D+B'CD'	((BD)'(A'C'D)'(B'CD')')'
4	AD+AC+A'B'C'	((AD)'(AC)'(A'B'C')')'
5	B+AC	((B)'(AC)')'
6	AC	(AC)”

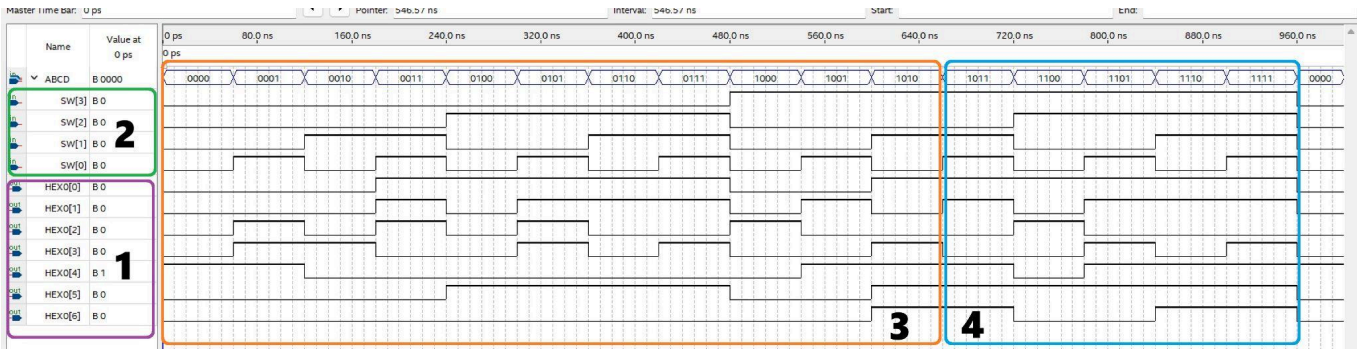
After doing the K-Maps and Mapping to NANDs, the transistor count became 84 transistors. This is a 46 percent reduction, which is good because you can be able to cut the cost but almost half while obtaining the same results.

6. Simulation Results

I coded the circuit design in the Verilog file ssd.v using structural Verilog constructs. This step was crucial for translating the design into a functional circuit.

After implementation, I simulated the circuit in Quartus to verify its behavior with all possible valid input combinations. This simulation step was essential for identifying any potential issues or bugs in the design.

Below a snapshot of the result of the simulation.



Key

Area 1 / Purple ; List of the Outputs

Area 2 / Green ; List of the Inputs

Area 3/ Orange ; Range of valid Inputs/Outputs Results

Area 4/ Blue ; Range of invalid inputs/outputs (commonly referred in this course as don't cares also indicated in the truth table as Xs)

7. Evaluation & Observation

Finally, I tested the circuit on the DE10-Lite Board, using the switches to provide valid input combinations. I was able to obtain all the desired outputs according to the input combinations. This step validated the functionality of the circuit in a real-world scenario.

8. Conclusions

The completion of this project has enhanced my understanding of digital system fundamentals, particularly in combinational logic design and implementation. It provided hands-on experience in utilizing Verilog for circuit modeling and optimization, as well as practical skills in simulation and testing. Overall, the project contributes significantly to the learning outcomes of ECE 2544.