Table of Contents

List of Tables

List of Figures

Purpose

The purpose of this project was to design, simulate, and partially construct a combinational circuit. This required application of combinational circuit theory and the use of a hardware trainer and software simulator.

Problem Specification

The final circuit had to correctly drive an inverted 7-segment LED display in accordance with a 4-bit Gray code input. Only 2-input NAND gates and inverters were available. After designing and simulating all 7 driver circuits, at least 3 had to be selected for physical implementation on a trainer. The selected circuits had to be drawn in schematic form to match the chip layout on the board and simulated again before wiring.

Design Process

The first step was to assign variable names. Inputs were assigned A, B, C, and D while the outputs were assigned the letters "t" through "z," corresponding alphabetically to the input pins "a" through "g" of the LED display. After assigning variables, a truth table was created showing all 4 inputs and all 7 outputs. The truth table included all input combinations, even invalid ones. **Table 1** shows that truth table.

A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	x	x	x	x	x	x	x
0	0	0	1	x	x	x	x	x	x	x
0	0	1	0	0	0	0	0	0	0	1
0	0	1	1	x	x	x	x	x	x	x
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	0	0	0	1	1	0
0	1	1	0	1	0	0	1	1	1	1
0	1	1	1	0	0	1	0	0	1	0
1	0	0	0	x	x	x	x	x	x	x
1	0	0	1	x	x	x	x	x	x	x
1	0	1	0	0	0	0	1	1	0	0
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	0	1	0	0	1	0	0
1	1	0	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	1	1	1

Table 1: Truth Table

Figure 1 shows the layout of the segments and connection pins of the LED display. The segments of the display light when that input pin receives a logic-0, so the 1 terms in the truth table indicate when that segment is dark.

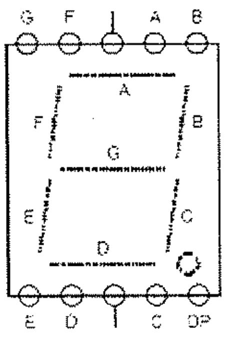

Figure 1: LED Display

With 4 input bits, there are 16 total input combinations. **Table 2** shows the 10 valid input combinations. The other 6 show up as "x" in the truth table, indicating "don't care" conditions.

Decimal	0	1	2	3	4	5	6	7	8	9
Binary	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010

Table 2: Valid Gray code combinations

From the truth table, Karnaugh maps were derived and simplified. These maps are presented in **Figure 2**.

	t			
ab \ cd	00	01	11	10
00	x0	x0	x0	0
01	1	0	0	1
11	0	1	0	0
10	x0	x1	x0	0

a.

	u			
ab \ cd	00	01	11	10
00	x0	x0	x0	0
01	0	0	0	0
11	1	1	0	0
10	x1	x1	x0	0

b.

	v			
ab \ cd	00	01	11	10
00	x	x	x	0
01	0	0	1	0
11	0	0	0	0
10	x	x	x	0

c.

	w			
ab \ cd	00	01	11	10
00	x0	x0	x0	0
01	1	0	0	1
11	0	0	1	0
10	x0	x0	x1	1

d.

	x			
ab \ cd	00	01	11	10
00	x1	x1	x0	0
01	1	1	0	1
11	1	0	1	0
10	x1	x1	x1	1

e.

	y			
ab \ cd	00	01	11	10
00	x0	x1	x1	0
01	0	1	1	1
11	0	0	1	0
10	x0	x0	x1	0

f.

	z			
ab \ cd	00	01	11	10
00	x	x	x	1
01	0	0	0	1
11	0	0	1	0
10	x	x	x	0

g.

Figure 2: Karnaugh Maps used to simplify Boolean equations

From the Karnaugh map simplifications, 7 output equations were derived:

t (A,B,C,D) = A'BD'+AC'D
u (A,B,C,D) = AC’
v (A,B,C,D) = A'BCD
w (A,B,C,D) = A'BD' + ACD + AB’C
x (A,B,C,D) = A'C' + C'D' + AB' + ACD + A'BD'
y(A,B,C,D) = A'D + A'BC + CD

Some manipulation with Boolean algebra yielded forms more suitable for NAND gate implementation:

t (A,B,C,D) = ([(A’B)’ ‘ D’]’ [(AC’)’ ‘ D]’)’
u (A,B,C,D) = (AC’)’’
v (A,B,C,D) = [(A’B)’ ‘ (CD)’ ‘]’ ‘
w (A,B,C,D) = [(A’BD’)’(ACD)’(AB’C)’]’
x (A,B,C,D) =[(A'C')’(C'D')’ (AB')’(AC D)’ (A'BD')’]’
y (A,B,C,D) = [(A’D)’(A’BC)’(CD)’]’
z (A,B,C,D) = [(ACD)’(A’CD’)’]’

Once these equations were derived, the logic diagram was drawn in LogicWorks. **Figure 3** is the diagram showing all 7 driver circuits. It would have been more efficient to reuse the “CD” signal generated in circuit v when building circuit z. For clarity and to avoid a lot of wire crossings, it was regenerated here. In the implementation, it was reused.

35+20
55 GATES
NOT THE MOST EFFICIENT.
CONSIDER SIMILARITIES IN
BOOLEAN EXPRESSIONS

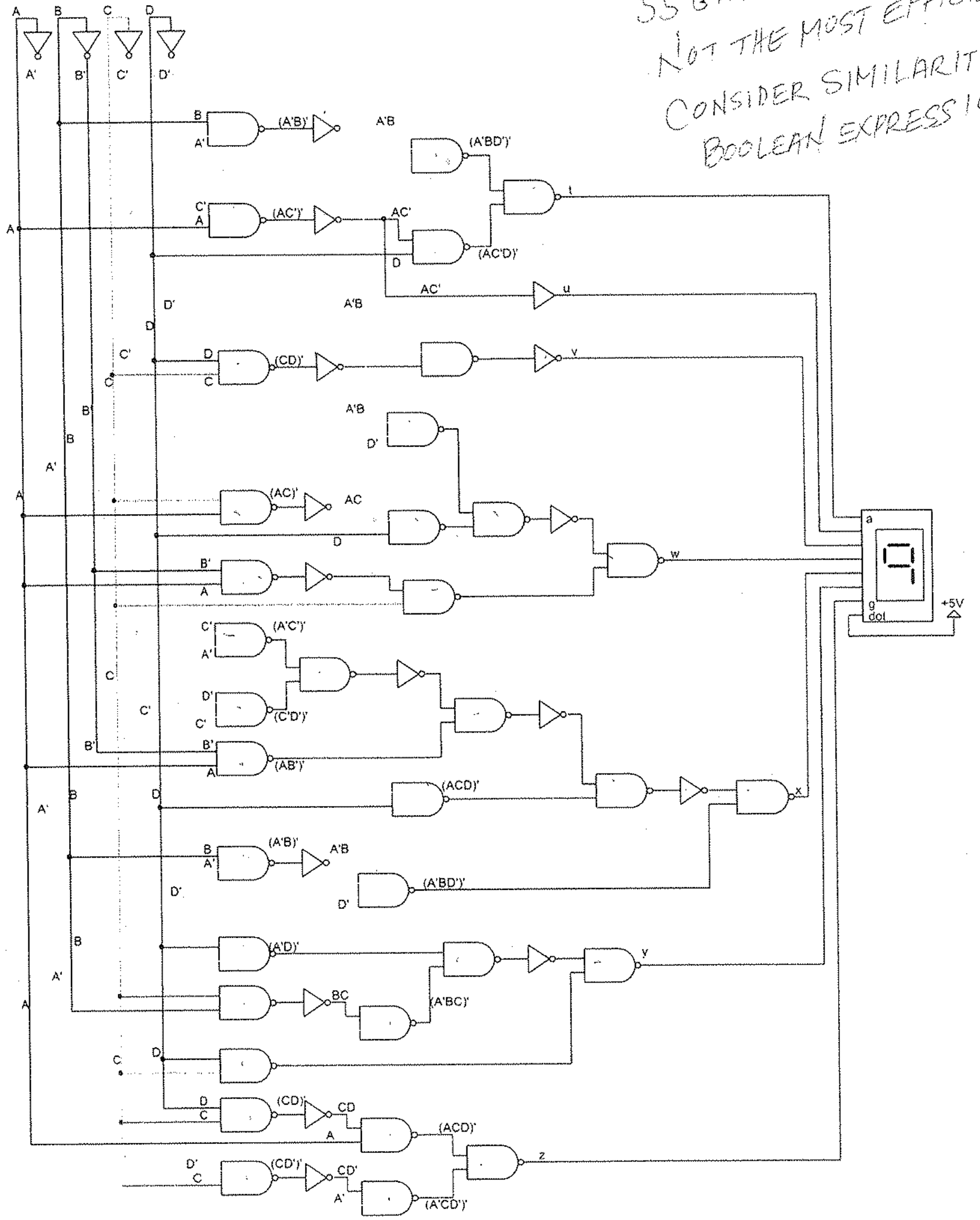

Figure 3: Gate diagram of all driver circuits

For initial testing of the logic, binary switches and probes were used. Once the circuit was proven manually, the switches and probes were removed for simulation using an input timing file. **Figure 4** is the timing window from that simulation, annotated with each correct input/output combination.

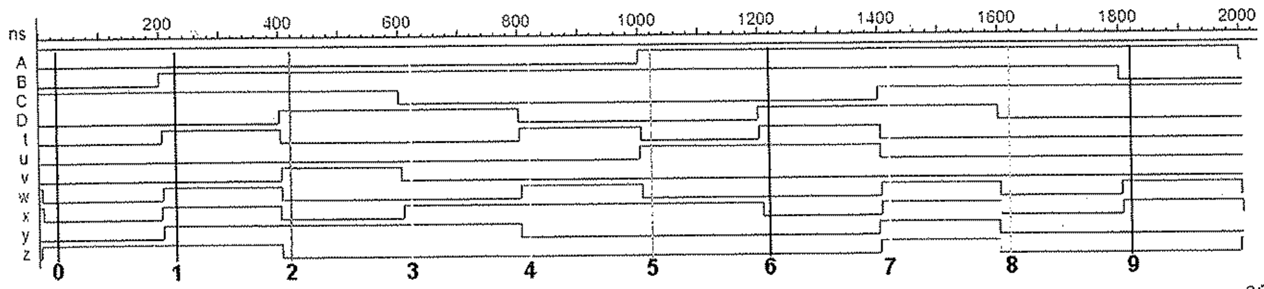

Figure 4: Gate diagram simulation timing window

GREAT LABELS

$T	$D	$O A	$O B	$O C	$O D	$O t	$O u	$O v	$O w	$O x	$O y	$O z
0	3NS	0	0	1	0	0	0	0	1	1	0	0
3NS	1NS	0	0	1	0	0	0	0	1	1	0	1
4NS	2NS	0	0	1	0	0	0	0	0	1	0	1
6NS	194NS	0	0	1	0	0	0	0	0	0	0	1
200NS	4NS	0	1	1	0	0	0	0	0	0	0	1
204NS	2NS	0	1	1	0	1	0	0	0	1	0	1
206NS	194NS	0	1	1	0	1	0	0	1	1	1	1
400NS	3NS	0	1	1	1	1	0	0	1	1	1	1
403NS	1NS	0	1	1	1	0	0	0	1	0	1	1
404NS	1NS	0	1	1	1	0	0	1	1	0	1	1
405NS	195NS	0	1	1	1	0	0	1	0	0	1	0
600NS	4NS	0	1	0	1	0	0	1	0	0	1	0
604NS	5NS	0	1	0	1	0	0	0	0	0	1	0
609NS	191NS	0	1	0	1	0	0	0	0	1	1	0
800NS	3NS	0	1	0	0	0	0	0	0	1	1	0
803NS	1NS	0	1	0	0	1	0	0	0	1	1	0
804NS	1NS	0	1	0	0	1	0	0	0	1	0	0
805NS	195NS	0	1	0	0	1	0	0	1	1	0	0
1US	3NS	1	1	0	0	1	0	0	1	1	0	0
1003NS	2NS	1	1	0	0	1	1	0	1	1	0	0
1005NS	2NS	1	1	0	0	0	1	0	1	1	0	0
1007NS	193NS	1	1	0	0	0	1	0	0	1	0	0
1200NS	2NS	1	1	0	1	0	1	0	0	1	0	0
1202NS	7NS	1	1	0	1	1	1	0	0	1	0	0
1209NS	191NS	1	1	0	1	1	1	0	0	0	0	0
1400NS	2NS	1	1	1	1	1	1	0	0	0	0	0
1402NS	2NS	1	1	1	1	1	1	0	0	0	1	0
1404NS	1NS	1	1	1	1	1	0	0	0	0	1	1
1405NS	1NS	1	1	1	1	0	0	0	0	0	1	1
1406NS	194NS	1	1	1	1	0	0	0	1	1	1	1
1600NS	2NS	1	1	1	0	0	0	0	1	1	1	1
1602NS	2NS	1	1	1	0	0	0	0	1	1	0	1
1604NS	196NS	1	1	1	0	0	0	0	0	0	0	0
1800NS	5NS	1	0	1	0	0	0	0	0	0	0	0
1805NS	2NS	1	0	1	0	0	0	0	1	0	0	0
1807NS	0	1	0	1	0	0	0	0	1	1	0	0

Table 3: Gate simulation timing export file raw data

Implementation

Once the design was completed and successfully tested, 4 driver circuits were chosen for implementation. To decide which circuits to implement, **Table 4** was created comparing the number of gates required to implement each circuit and showing any shared signals. For instance, both "t" and "v" used the signal "A'B" at some point in the logic. Since "t" was already implemented, that shared signal saved a NAND gate and an inverter in the "v" circuit. ✓

Sig.	Init. Count		Shared?	Final Count	
	NAND	INV*		NAND	INV*
t	5	4	AC'(u), A'B(v,w,x)	5	4
u	1	2	AC'(t)	0	0
v	3	4	A'B(t),CD(z)	2	1
w	8	6	A'B(t)	7	4
x	11	7	AC(w),A'B(t)	10	6
y	6	4		5	4
z	5	3	CD (v)	4	1
			Total implemented (t,u,v,z):	12	8

*includes inverted inputs

Table 4: Gate counts

It was possible to implement 4 circuits, t, u, v, and z, using only 3 quad NAND chips and 2 hex inverters. The total hardware count for these 4 circuits was 12 NAND gates and 8 inverters, requiring 3 quad NAND chips and 2 hex inverter chips.

After selecting the 4 driver circuits to implement, a chip-level schematic was developed in LogicWorks. The 5 IC chips were arranged in a row across the center of the page, with the 2 hex inverter chips all the way to the left. The 7-segment inverted LED display is placed to the right. For initial testing of the schematic, the inputs were simulated using binary switches. The first circuit built was for the "t" output, with all of the wiring on the top side of the chips. As with the gate logic diagram, each circuit was fully tested before moving on to the next. Whenever possible, signals generated previously were re-used in subsequent circuits. **Figure 5** shows the final wiring schematic.

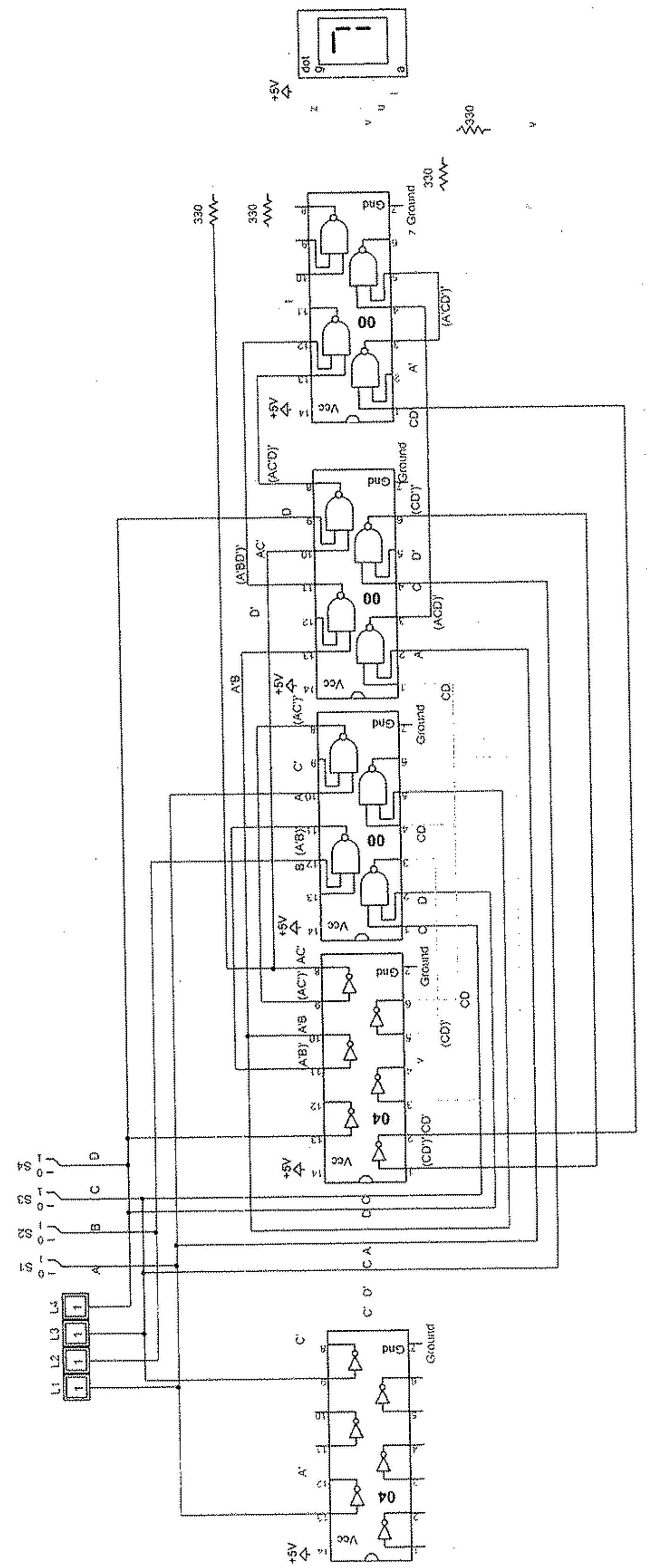

Figure 5: Wiring schematic showing all 4 implemented driver circuits

Before implementing the circuit on the breadboard, the schematic was simulated in LogicWorks. For this simulation, the binary switches were eliminated. The same input timing file used before to test the gate logic diagram was used here. **Figure 6** is the timing window from the simulation of the schematic. The 10 valid input combinations with their corresponding outputs are noted on the diagram. **Table 4** is the exported timing data.

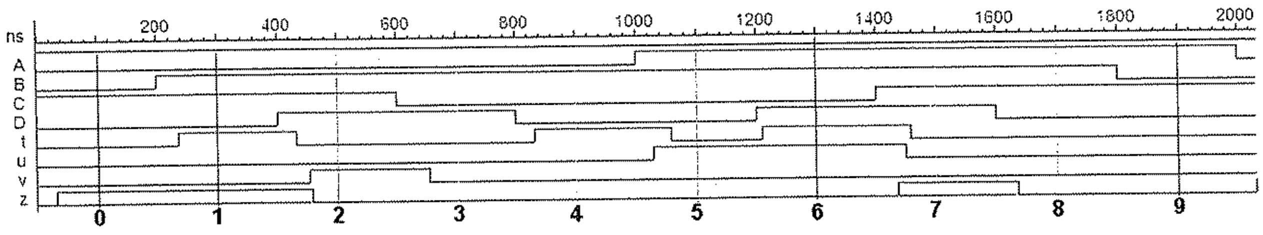

Figure 6: Schematic simulation timing window

$T	$D	$O A	$O B	$O C	$O D	$O t	$O u	$O v	$O z
0	32NS	0	0	1	0	0	0	0	0
32NS	168NS	0	0	1	0	0	0	0	1
200NS	37NS	0	1	1	0	0	0	0	1
237NS	163NS	0	1	1	0	1	0	0	1
400NS	32NS	0	1	1	1	1	0	0	1
432NS	22NS	0	1	1	1	0	0	0	1
454NS	5NS	0	1	1	1	0	0	1	1
459NS	141NS	0	1	1	1	0	0	1	0
600NS	54NS	0	1	0	1	0	0	1	0
654NS	146NS	0	1	0	1	0	0	0	0
800NS	32NS	0	1	0	0	0	0	0	0
832NS	168NS	0	1	0	0	1	0	0	0
1US	28NS	1	1	0	0	1	0	0	0
1028NS	31NS	1	1	0	0	1	1	0	0
1059NS	141NS	1	1	0	0	0	1	0	0
1200NS	10NS	1	1	0	1	0	1	0	0
1210NS	190NS	1	1	0	1	1	1	0	0
1400NS	37NS	1	1	1	1	1	1	0	0
1437NS	13NS	1	1	1	1	1	1	0	1
1450NS	9NS	1	1	1	1	1	0	0	1
1459NS	141NS	1	1	1	1	0	0	0	1
1600NS	37NS	1	1	1	0	0	0	0	1
1637NS	163NS	1	1	1	0	0	0	0	0
1800NS	200NS	1	0	1	0	0	0	0	0
2US	32NS	0	0	1	0	0	0	0	0
2032NS	0	0	0	1	0	0	0	0	1

Table 5: Schematic simulation timing export file raw data

Wiring

Once the circuit was proven in LogicWorks, it was assembled on a breadboard. As each wire was run, it was marked off on the schematic. The final output of each combinational circuit was connected to the appropriate pin of the LED display through a 330 Ω resistor to protect the components by limiting the current. When necessary, the resistor leads were insulated to prevent shorts between circuits. When each driver circuit was wired, it was tested before moving on to the next. A photograph of the final circuit is shown in **Figure 7**.

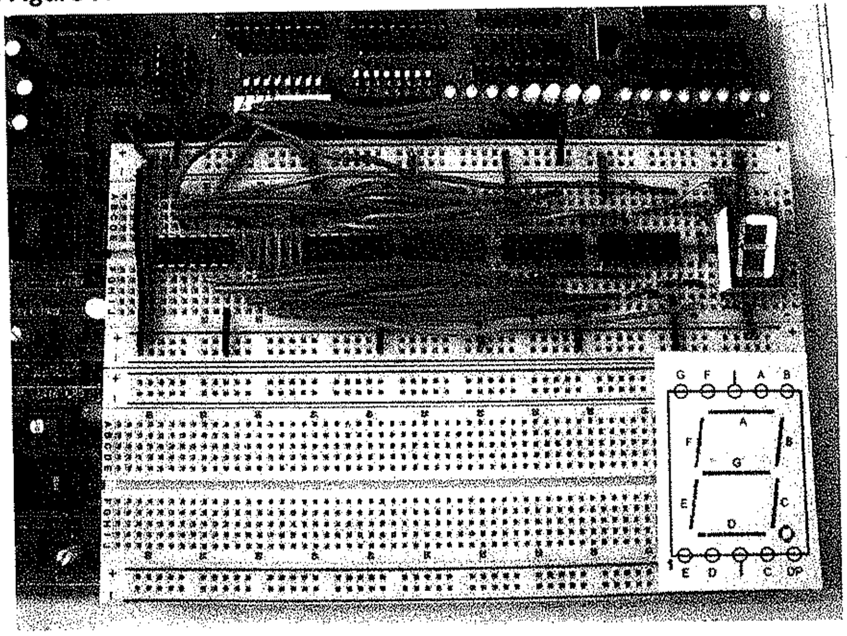

Figure 7: Final wiring

Validation

To make the validation easier, a paper copy of the LED layout, with the implemented segments highlighted, was attached to the unused breadboard. The input DIP switches were also marked. There were no problems during circuit validation. The implemented circuits functioned properly.

Conclusions

This project was a good exercise in designing and implementing a combinational circuit from a written specification. The assignment was clearly explained and well documented.

Given another attempt, or enough time to re-do the project, a different arrangement of the chips on the schematic and the breadboard may have been better. Another option would be to alternate chips, starting with a NAND chip, to spread out the inverters more. The extra space would have been better divided between all the chips, rather than gathered together in one place. The wiring would be easier to implement and to trace later. Because the schematic was fully simulated ahead of time and the wiring followed the schematic closely, the circuit worked on the first try. Had there been a mistake somewhere, this chip arrangement may have made it difficult to find.

THIS IS A GREAT REPORT.