

## 1. Lab: BCD to 7-segment decoder

## [1] Structural description

Simulation result:

Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns					
out[6:0]	1110011	1111110	0110000	1101101	1111001	1011011	1011111	1110000	1111111	1110011	
in[3:0]	1001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

We can check that each input combination gives the expected output combination.

## [2] Data-flow description

Simulation result:

Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns					
out[6:0]	1110011	1111110	0110000	1101101	1111001	0110011	1011011	1011111	1110000	1111111	1110011
in[3:0]	1001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

The result is the same as [1], which is the desired result.

## [3] Behavioral description

Simulation result:

Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns					
out[6:0]	1110011	1111110	0110000	1101101	1111001	1011011	1011111	1110000	1111111	1110011	
in[3:0]	1001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

The result is the same as [1], which is the desired result.

## 2. Homework: Klingon number system decoder

## (1)-(i) Truth table:

ABCD	a	b	c	d	e	f	g
0000	1	1	1	1	1	1	0
0001	1	0	0	0	0	0	0
0010	1	0	0	0	0	0	1
0011	1	0	0	1	0	0	1
0100	0	1	0	0	0	1	1
0101	0	0	1	1	1	0	1
0110	0	1	0	0	1	0	1
0111	0	0	1	0	0	1	1
1000	0	1	1	0	1	1	0
1001	0	1	1	0	1	1	1
1010	x	x	x	x	x	x	x
1011	x	x	x	x	x	x	x
1100	x	x	x	x	x	x	x
1101	x	x	x	x	x	x	x
1110	x	x	x	x	x	x	x
1111	x	x	x	x	x	x	x

## (1)-(ii) Karnaugh map

<p><b>a</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 0 X 0</p> <p>01 1 0 X 0</p> <p>11 1 0 X X</p> <p>10 1 0 X X</p> <p><math>B</math></p> <p><math>a = A'B'</math></p>	<p><b>b</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 1 X 1</p> <p>01 0 0 X 1</p> <p>11 0 0 X X</p> <p>10 0 1 X X</p> <p><math>B</math></p> <p><math>b = A + BD' + C'D'</math></p>	<p><b>c</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 0 X 1</p> <p>01 0 1 X 1</p> <p>11 0 1 X X</p> <p>10 0 0 X X</p> <p><math>B</math></p> <p><math>c = B'C'D' + BD + A</math></p>
<p><b>d</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 0 X 0</p> <p>01 0 1 X 0</p> <p>11 1 0 X X</p> <p>10 0 0 X X</p> <p><math>B</math></p> <p><math>d = BC'D + B'CD + A'B'C'D'</math></p>	<p><b>e</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 0 X 1</p> <p>01 0 1 X 1</p> <p>11 0 0 X X</p> <p>10 0 1 X X</p> <p><math>B</math></p> <p><math>e = A + BCD' + BC'D + B'C'D'</math></p>	<p><b>f</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 1 1 X 1</p> <p>01 0 0 X 1</p> <p>11 0 1 X X</p> <p>10 0 0 X X</p> <p><math>B</math></p> <p><math>f = A + C'D' + BCD</math></p>
<p><b>g</b></p> <p><math>A</math></p> <p><math>CD \backslash AB</math> 00 01 11 10</p> <p>00 0 1 X 0</p> <p>01 0 1 X 1</p> <p>11 1 1 X X</p> <p>10 1 1 X X</p> <p><math>B</math></p> <p><math>g = B + C + AD</math></p>		

### (1)-(iii) Boolean expressions

$$a = A'B'$$

$$b = A + BD' + C'D'$$

$$c = A + BD + B'C'D'$$

$$d = BC'D + B'CD + A'B'C'D'$$

$$e = A + BCD' + BC'D + B'C'D'$$

$$f = A + C'D' + BCD$$

$$g = B + C + AD$$

### (2) Source code

For each description, inputs A, B, C and D are represented by a single array in[3:0], and the outputs a, b, c, d, e, f and g are represented by a single array out[6:0].

### (3) Simulation result

#### [1] Structural description

Simulation result:

Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns
out[6:0]	0110111	111110	1000000	1001001	0100011	0011101
in[3:0]	1001	0000	0001	0010	0011	0100

We can check that each input combination gives the expected output combination, matching the truth table.

#### [2] Data-flow description

Simulation result:

Name	Value	0 ns	200 ns	400 ns	600 ns	800 ns					
out[6:0]	0110111	1111110	1000000	1001001	0100011	0011101	0100101	0010011	0110110	0110111	
in[3:0]	1001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

The result is the same as [1], which is the desired result.

### [3] Behavioral description

Simulation result:

Name	Value	0 ns		200 ns		400 ns		600 ns		800 ns	
out[6:0]	0110111	1111110	1000000	1000001	1001001	0100011	0011101	0100101	0010011	0110110	0110111
in[3:0]	1001	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

The result is the same as [1], which is the desired result.

### (3) Discussion

The readability of the simulation result increased compared to the previous lab session, by using arrays for inputs and outputs respectively. It was significantly easier to check the output was correct, because this made it easier to compare the simulation result with the truth table. Moreover, the implementation of the test code was made simpler than the previous lab session, with the use of for loops. I no longer needed to write one line of code for every input combination, which saved a lot of time.