The Karnaugh Map

Aims

- to describe the **Karnaugh map** technique for circuit simplification
- to apply **Karnaugh** mapping as a technique
- to extend Karnaugh maps to include ***don't care*** states

6.0 Introduction

In previous lectures truth tables have been introduced, and a method to derive a Boolean expression from them.

Consider the truth table for a simple multiplexor :

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

In ***Sum of Products*** representation the truth table can be expressed as:

Using Boolean algebra the expression can be reduced as follows:

This is relatively straightforward for simple 3-input circuits.
However, using Boolean algebra to reduce circuits can be difficult.

Therefore we would like a method that is easier – ***the Karnaugh map.***

6.1 Karnaugh Maps

The ***Karnaugh map*** is a visual way of ***detecting redundancy in a Sum of Products expression***, and can easily be used for circuits with 2, 3, or 4 inputs.

It consists of an array of cells, each representing a possible combination of inputs.

Cells are arranged so that each cell's input combination differs from adjacent cells by only a single bit.

Consider the following cell arrangements:

2-input

00	01
10	11

3-input

000	001	011	010
100	101	111	110

4-input

0000	0001	0011	0010
0100	0101	0111	0110
1100	1101	1111	1110
1000	1001	1011	1010

The cells are arranged as above but are written as empty:

2-input

		B	
		0	1
A	0		
	1		

3-input

		B C			
		00	01	11	10
A	0				

4-input

		C D			
		00	01	11	10
	00				
A B	01				
	11				
	10				

Note that **the numbers are not in binary order**, but are arranged so that *only a single bit changes between neighbours*.

- This one-bit change applies at the edges, too.
    - So cells in the same row on the left and right edges of the array also only differ by one bit.

Given an empty Karnaugh map 1's are put in all the cells that represent ***minterms*** in the *Sum of Products representation*.

- In other words, locate the 1's in the truth table output and place them in the cells corresponding to the same inputs.

Consider the truth table for a simple multiplexor:

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

		B C			
		00	01	11	10
A	0				
	1				

If there are two neighbouring **1's** in the map it means that the input bit change between the two cells has no effect on the output, and thus there is redundancy. This leads to a basic strategy:

Group adjacent 1's together in square or rectangular groups of 2, 4, 8, or 16, such that the total number of groups and isolated 1's is minimised, while using as large groups as possible

Groups may overlap so that a particular cell may be included in more than one group

Remember that adjacency wraps around the edges of the map

Applying this to the multiplexor example:

		B C			
		00	01	11	10
A	0				
	1				

Considering the grouping, notice the following:

1. **B** changes but the output does not. Therefore **B** is redundant in this group.
2. **A** changes but the output does not. Therefore **A** is redundant in this group.

The Boolean expressions are written for each group, leaving out the redundant terms.

- That is, for each group, the inputs that do not changed are written.

The multiplexor example, with two groups, produces two terms:

The whole process of converting a truth table output into a Karnaugh map can be summarised into a basic set of rules:

1. Each cell with a **1** must be included in at least one group.
2. Form the largest possible groups.
3. Form as few groups as possible.
4. Groups may be sizes that are powers of 2.
5. Groups may be square or rectangular only (including wrap-around at the map edges).
6. The larger a group is, the more redundant inputs there are:

   - A group of **1** has no redundant inputs
   - A group of **2** has **1** redundant input
   - A group of **4** has **2** redundant inputs
   - A group of **8** has **3** redundant inputs
   - A group of **16** has **4** redundant inputs

The following examples illustrate **rule 6**:

		B C			
		00	01	11	10
A	0	1	1	1	1
	1	0	0	1	0

		B C			
		00	01	11	10
A	0	0	1	1	0
	1	0	1	1	0

		C D			
		00	01	11	10
A B	00	1	1	1	1
	01	1	0	0	1
	11	1	0	0	1
	10	1	0	0	1

6.2 Examples

2-Input

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

		B	
		0	1
A	0		
	1		

3-Input

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

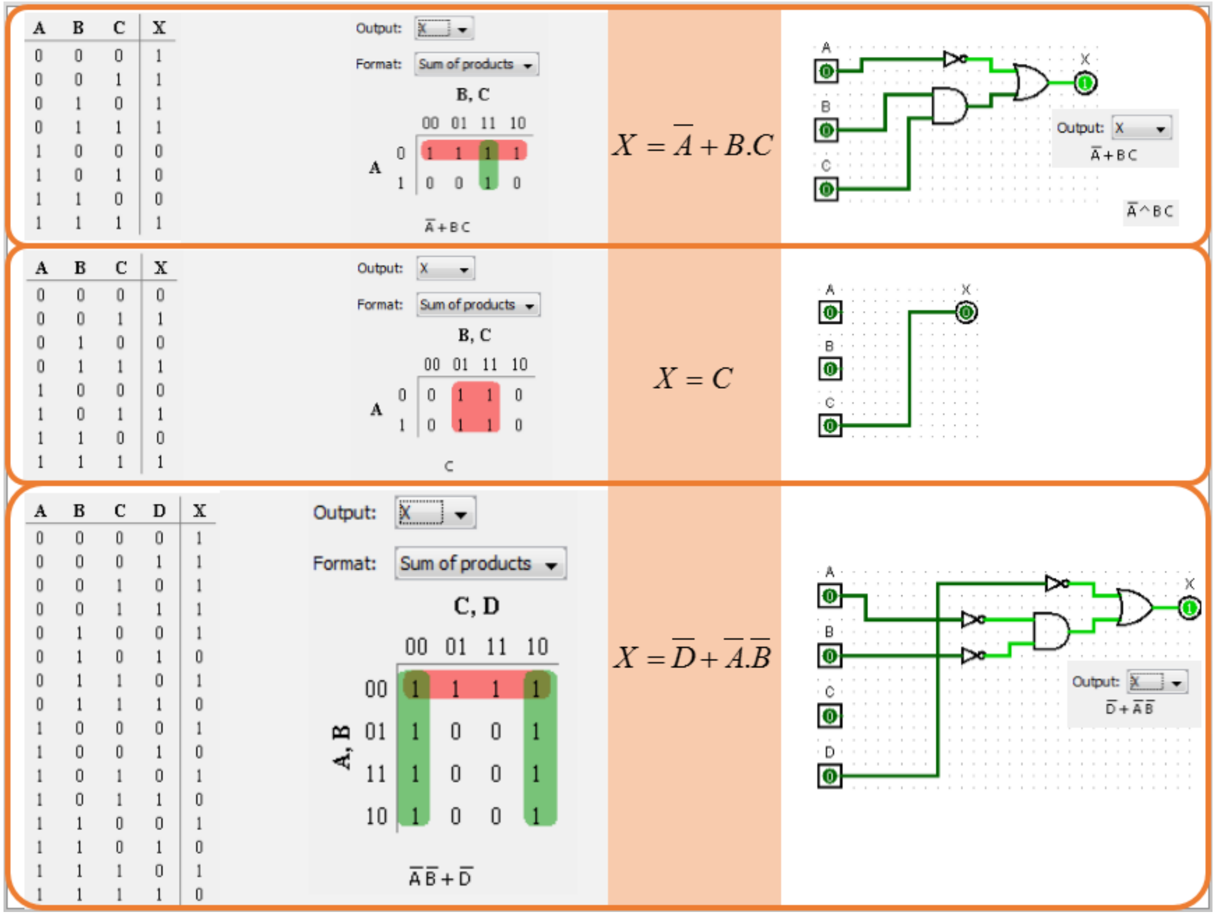
		B C			
		00	01	11	10
A	0				
	1				

4-Input

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

		C D			
		00	01	11	10
A B	00				
	01				
	11				
	10				

Further examples in the Lecture : -



SAQ 6.1

1. Which of the following rules for grouping Karnaugh maps is NOT correct?

- Every **1** must be included in at least one group.
- There should be the maximum possible number of groups.
- Each group must be as large as possible.
- A **1** may be included in more than one group if doing so would enlarge the size of a group.
- Groups must be square or rectangular with 1, 2, 4, 8 or 16 squares.

answer

Karnaugh Maps #1 – Don't Care States

Aims

- To extend Karnaugh maps to include don't care states

7.0 Introduction

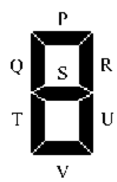
We have seen how

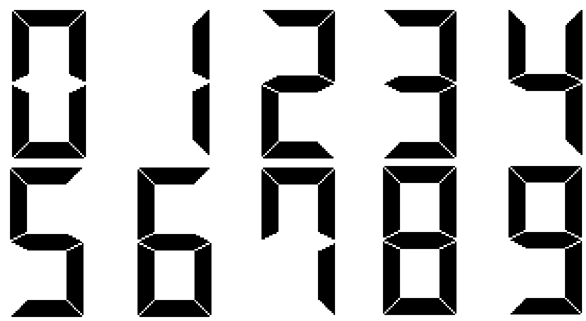
Karnaugh maps are a visual way of detecting redundancy in a Sum of Products representation

- However, there are sometimes cases where the output of a circuit does not matter
- This means that there are input combinations for which we do not care what the output is

7.1 Don't Care Logic States

Consider a seven-segment display :





On a seven-segment display only the numbers **0** to **9** can be displayed.

To represent the numbers **0** to **9** four bits are required: **0000** represents zero and **1001** represents nine.

- However, four bits can actually represents values from **0** (0000) to **15** (1111)

- Therefore, when the input bit values are from **10** (1010) to **15** (1111), we do not care what the output is

- That is, the inputs to the circuits will never have those values so we do not care what the circuit does if it does happen

So, a circuit to control a seven-segment display would have **four inputs** (the binary number representation of the decimal number to be displayed) and seven outputs.

(signals to each of the seven segments that tells them to turn on or off)

Consider the topmost segment, labelled **P**.

- Assume that a value of **1** activates the segment and a value of **0** turns it off.
- The top segment will be lit for values 0, 2, 3, 5, 6, 7, 8, 9.
    - We do not care what it does if the inputs are in the range 10 to 15.

In both the truth table and the Karnaugh map a don't care state is represented with a ***d***.

They are drawn on the Karnaugh map along with the 1's.

A	B	C	D	P
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	d
1	0	1	1	d
1	1	0	0	d
1	1	0	1	d
1	1	1	0	d
1	1	1	1	d

		C D			
		00	01	11	10
A B	00	1	0	1	1
	01	0	1	1	1
	11	d	d	d	d
	10	1	1	d	d

- When forming groups on the Karnaugh map the don't care states should be treated as 1's when they lead to the formation of larger groups.
- If they do not lead to the formation of larger groups, treat them as 0's.
- Furthermore, never form groups of just don't care states.

7.2 Example

Consider the following truth table:

A	B	C	D	X	Y	Z
0	0	0	0	1	1	1
0	0	0	1	1	0	0
0	0	1	0	0	0	1
0	0	1	1	1	1	0
0	1	0	0	0	1	1
0	1	0	1	1	1	0
0	1	1	0	0	0	1
0	1	1	1	1	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	1
1	1	0	0	d	d	d
1	1	0	1	d	d	d
1	1	1	0	d	d	d
1	1	1	1	d	d	d

		C D			
		00	01	11	10
A B	00	1	1	1	0
	01	0	1	1	0
	11	d	d	d	d
	10	0	1	1	1

		C D			
		00	01	11	10
A B	00	1	0	1	0
	01	1	1	1	0
	11	d	d	d	d
	10	1	0	1	0

		C D			
		00	01	11	10
A B	00	1	0	0	1
	01	1	0	0	1
	11	d	d	d	d
	10	1	1	1	1

SAQ 7.1

1. Under which circumstances should don't care states be treated as **1**'s?
2. Under which circumstances should don't care states be treated as **0**'s?

answer