University of British Columbia
Electrical and Computer Engineering
Digital Design and Microcomputers CPEN312

L05: Reduction Techniques.

Dr. Jesús Calviño-Fraga. P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

January 11/16, 2019

Objectives

- Boolean function reduction using K-maps
- Product of Maxterms/Sums POS simplification with K-maps
- “Don’t Care” uses in simplification
- Fundamental gates: NAND & NOR

Reduction via Geometry

- Recall this reduction trick:
  - Y=A.B+A.B'=A.(B+B')=A
  - X=A'.B+A.B=B.(A+A')=B
- Finding these (A+A') can be easily achieved with a map.
- For Boolean functions we use Karnaugh maps or K-maps.
- I have seen it working for 2 to 6 variables.

Karnaugh Maps

- The idea is to arrange the truth table into a square grid where each cell corresponds to a truth table row. They look like this:

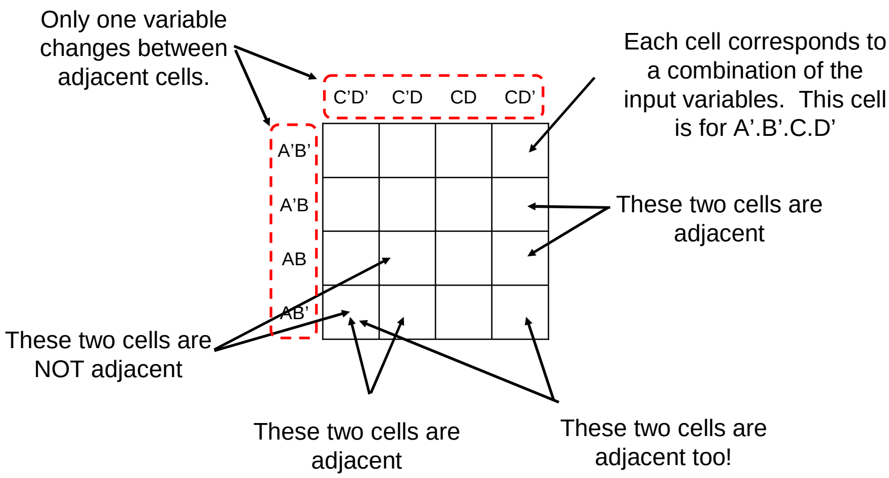
	B'	**A**
A'		
A		

	C'	**C**
A'B'		
A'B		
AB		
AB'		

	C'D'	**C'D**	**CD**	**CD'**
A'B'				
A'B				
AB				
AB'				

Two, three, and four variable Karnaugh maps.

Karnaugh Maps



Using K-Maps

- ONLY one variable changes when moving from adjacent cells. That is why we arrange the rows/columns as A'B', A'B, AB, AB'
- You can arrange the cells in any order, but the rule above must hold. A'B, AB, AB', A'B' will work also.
- Each cell represents a minterm.

Using K-Maps

- How this works? Consider the truth table below and its K-map.

In			Out
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

A'.B'.C'

A'.B'.C

A'.B.C'

	C'	C
A'B'	1	1
A'B	1	0
AB	0	0
AB'	0	0

Every output in the truth table corresponds to a cell in the K-map

Using K-Maps

- Now look at the ones in the K-map and encircle them together with all adjacent cells. The oval must contain only 1, 2, 4, 8, 16, etc. ones. An oval of 5 ones, for example, is not valid!

	C'	C
A'B'	1	1
A'B	1	0
AB	0	0
AB'	0	0

A'.B'

A'.C'

Now look at what variables remain constant within each encircle.

The final answer is the sum of the equation for each encircle:

X = A'.B' + A'.C'

Verify the Answer with Algebra…

In			Out
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

X=A’.B’.C’+A’.B’.C+A’.B.C’

X=A’.B’.(C+C’)+A’.B.C’

X=A’.B’+A’.B.C’

X=A’.(B’+B.C’)

X=A’.(B’.(C+C’)+B.C’)

X=A’.(B’C+B’C’+B.C’)

X=A’.(B’C+B’C’+B’C’+B.C’)

X=A’.(B’(C+C’)+C’(B’+B))

X=A’(B’+C’)

X=A’B’+A’C’

See why we use K-maps?

K-maps

- Try to encircle as many 1s as possible:

	C’	C
A’B’	1	0
A’B	1	1
AB	1	0
AB’	1	0

X= A’.B + C’

K-maps

- Pick the greatest number of ones on each oval. Remember the number of ones has to be 1,2,4,8,etc.

	C'D'	C'D	CD	CD'
A'B'	0	0	0	0
A'B	1	1	1	0
AB	1	1	1	0
AB'	0	0	0	0

Six ones encircled is not valid.

K-maps

- Pick the greatest number of ones on each oval. Remember the number of ones has to be 1,2,4,etc.

	C'D'	C'D	CD	CD'
A'B'	0	0	0	0
A'B	1	1	1	0
AB	1	1	1	0
AB'	0	0	0	0

Not optimal.

K-maps

- Pick the greatest number of ones on each oval. Remember the number of ones has to be 1,2,4,etc.

	C'D'	C'D	CD	CD'
A'B'	0	0	0	0
A'B	1	1	1	0
AB	1	1	1	0
AB'	0	0	0	0

Not optimal.

K-maps

- Pick the greatest number of ones on each oval. Remember the number of ones has to be 1,2,4,etc.

	C'D'	C'D	CD	CD'
A'B'	0	0	0	0
A'B	1	1	1	0
AB	1	1	1	0
AB'	0	0	0	0

Optimal!

X=B.C'+B.D

K-maps

- Remember: two cells are adjacent if only one variable changes… So for this K-map:

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	0	0	0	0
AB	1	0	0	1
AB'	1	1	1	1

X= B' + A.D'

K-maps

- I mentioned before that you can rearrange the rows/columns so far only one variable changes at a time. Then the problem from the previous slide can be redrawn as:

	C'D	CD	CD'	C'D'
A'B	0	0	0	0
AB	0	0	1	1
AB'	1	1	1	1
A'B'	1	1	1	1

X= B' + A.D'

WARNING: super easy to do in the computer (cut & paste rows/columns). A pain to do in paper by hand!

Example 1

- Simplify the following equation using a K-map:
  X=A'.D'+A.B'.D'+A'C'D+A'.C.D

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	1	1	1
AB	0	0	0	0
AB'	1	0	0	1

Cells with A'.D'

Cells with A.B'.D'

Cells with A'.C'.D

Cells with A'.C.D

Example 1

- Simplify the following equation using a K-map:
  X=A'.D'+A.B'.D'+A'C'D+A'.C.D

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	1	1	1
AB	0	0	0	0
AB'	1	0	0	1

A'

X= A' + B'.D'

Four corners: B'.D'

Example 1

- For your amusement, the previous K-map has been rolled one row and one column, just in case you can visualize the adjacency of the four corners…

	C'D	CD	CD'	C'D'
AB'	0	0	1	1
A'B'	1	1	1	1
A'B	1	1	1	1
AB	0	0	0	0

X= A' + B'.D'

Don't do this. It is a waste of time!

POS simplification with K-maps

- Instead of working with the ones work with the zeroes.
- You'll get F'.
- Use De Morgan's theorem to find F.
- Example next slide…

POS simplification with K-maps

	C'D'	C'D	CD	CD'
A'B'	1	1	0	1
A'B	0	1	0	0
AB	0	0	0	0
AB'	1	1	0	1

F'= B.D' + AB + CD

De Morgan's says

(X.Y)'=X'+Y'

Or

X.Y=(X'+Y')'

F'=(B'+D)'+(A'+B')'+(C'+D')'

F'=((B'+D).(A'+B').(C'+D'))'

F=(B'+D).(A'+B').(C'+D')

Example 2

- Get the product of sums from the K-map below:

	C'D'	C'D	CD	CD'
A'B'	1	1	1	1
A'B	1	0	0	1
AB	1	0	0	1
AB'	0	1	1	1

F'= B.D + AB'C'D'

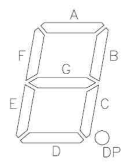
F=(B'+D').(A'+B+C+D)

Don’t Care Conditions

- If the Boolean function has un-specified outputs, you can make them either 0 or 1 at your convenience!
- One classical example is the conversion of a BCD to 7-segment displays. The inputs $(1010)_2$ to $(1111)_2$ have un-specified outputs.

Example 3: BCD to 7-Segments

Inputs				Outputs						
D	C	B	A	F_a	F_b	F_c	F_d	F_e	F_f	F_g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	?	?	?	?	?	?	?
1	0	1	1	?	?	?	?	?	?	?
1	1	0	0	?	?	?	?	?	?	?
1	1	0	1	?	?	?	?	?	?	?
1	1	1	0	?	?	?	?	?	?	?
1	1	1	1	?	?	?	?	?	?	?



Don’t Care!

Some people / books use ‘x’ or ‘d’ for don’t cares

Example 3: Solution for Segment 'a'

D	C	B	A	F_a
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C'D'	C'D	CD	CD'
A'B'	0	0	?	1
A'B	0	?	?	0
AB	0	?	?	0
AB'	1	0	?	0

F_a=A.B'.C'.D'+A'.B'.C

Example 3: Solution for Segment 'b'

D	C	B	A	F_b
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C'D'	C'D	CD	CD'
A'B'	0	0	?	0
A'B	0	?	?	1
AB	0	?	?	0
AB'	0	0	?	1

F_b=A'BC+AB'C

Example 3: Solution for Segment ‘c’

D	C	B	A	F_c
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C’D’	C’D	CD	CD’
A’B’	0	0	?	0
A’B	1	?	?	0
AB	0	?	?	0
AB’	0	0	?	0

F_c=A’.B.C’

Example 3: Solution for Segment ‘d’

D	C	B	A	F_d
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C’D’	C’D	CD	CD’
A’B’	0	0	?	1
A’B	1	?	?	0
AB	0	?	?	1
AB’	0	0	?	0

F_d=A’.B.C’+A’.B’.C+A.B.C

Example 3: Solution for Segment 'e'

D	C	B	A	F_e
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C'D'	C'D	CD	CD'
A'B'	0	0	?	1
A'B	0	?	?	0
AB	1	?	?	1
AB'	1	1	?	1

Finally!

F_e=A+B'.C

Example 3: Solution for Segment 'f'

D	C	B	A	F_f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C'D'	C'D	CD	CD'
A'B'	0	0	?	0
A'B	1	?	?	0
AB	1	?	?	1
AB'	1	0	?	0

F_f=B.C'+A.B+A.C'.D'

Example 3: Solution for Segment ‘g’

D	C	B	A	F_g
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	x
1	0	1	1	x
1	1	0	0	x
1	1	0	1	x
1	1	1	0	x
1	1	1	1	x

	C’D’	C’D	CD	CD’
A’B’	1	0	?	0
A’B	0	?	?	0
AB	0	?	?	1
AB’	1	0	?	0

F_g=B’.C’.D’+A.B.C

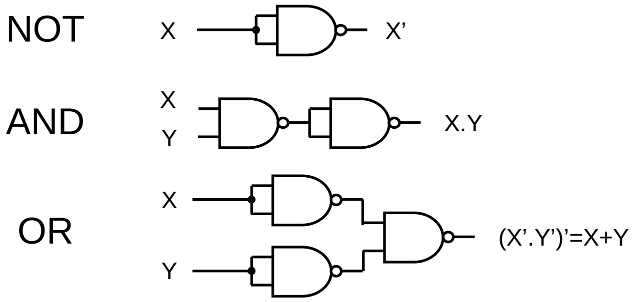
You are welcome!

Universal Gates

- An universal gate is a gate that can be used to implement any logic function. There are two universal gates:
  - NAND
  - NOR

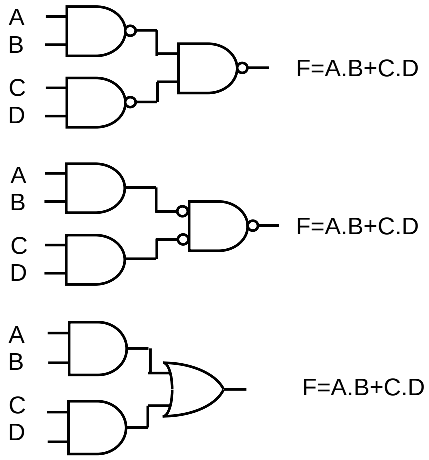
Universal Gate: NAND

- We need to show that we can implement a NOT, AND, and OR with just NAND gates:



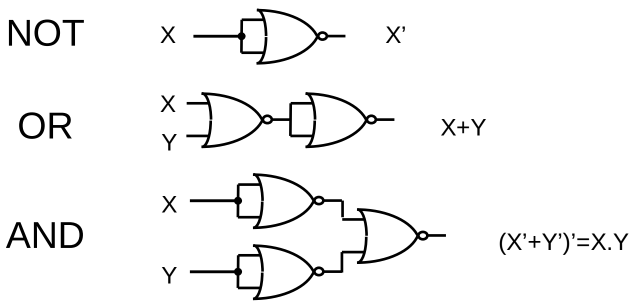
The AND-OR Circuit

These three circuits behave the same!



Universal Gate: NOR

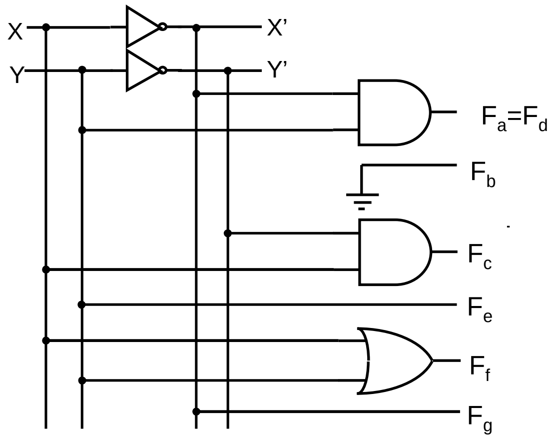
- We need to show that we can implement a NOT, AND, and OR with just NOR gates:



Example 4: All NAND Design

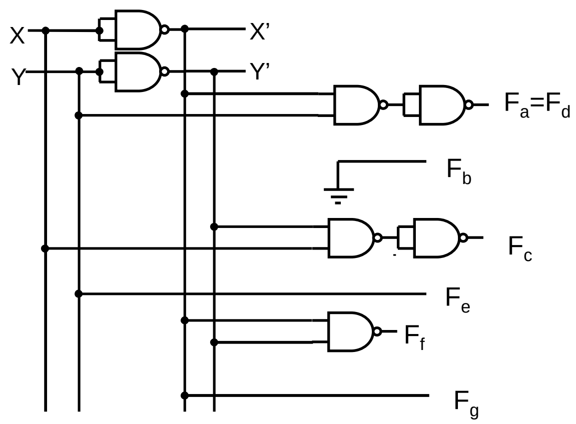
- Consider the Boolean equations for the 2-bit to 7-segment decoder from last lecture. Implement the circuit using only NAND gates. Simulate in Multisim.
    - $F_a=F_d=X'.Y$
    - $F_b=0$
    - $F_c=X.Y'$
    - $F_e=Y$
    - $F_f=X+Y$
    - $F_g=X'$

Example 4: All NAND Design



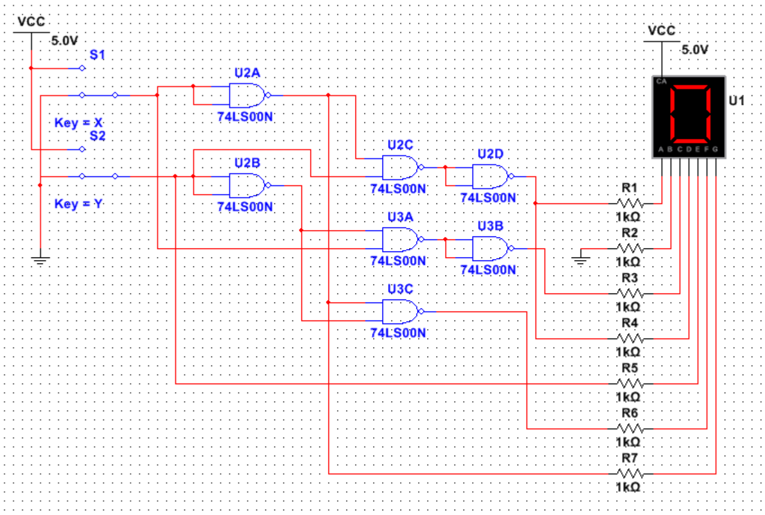
The original circuit

Example 4: All NAND Design



The all-NAND circuit

Multisim Simulation



Exercises

- Simplify the Boolean equation
  Y=L.T.P+L’T.W+L’T’P+L’.W’.T
  Using a K-map. Implement the simplified circuit of the equation above using NOR gates only.
- Design a circuit, using a K-map, that detects a number divisible by 3 in the range 1 to 15 (zero is don’t care). Implement the circuit using NAND gates only.