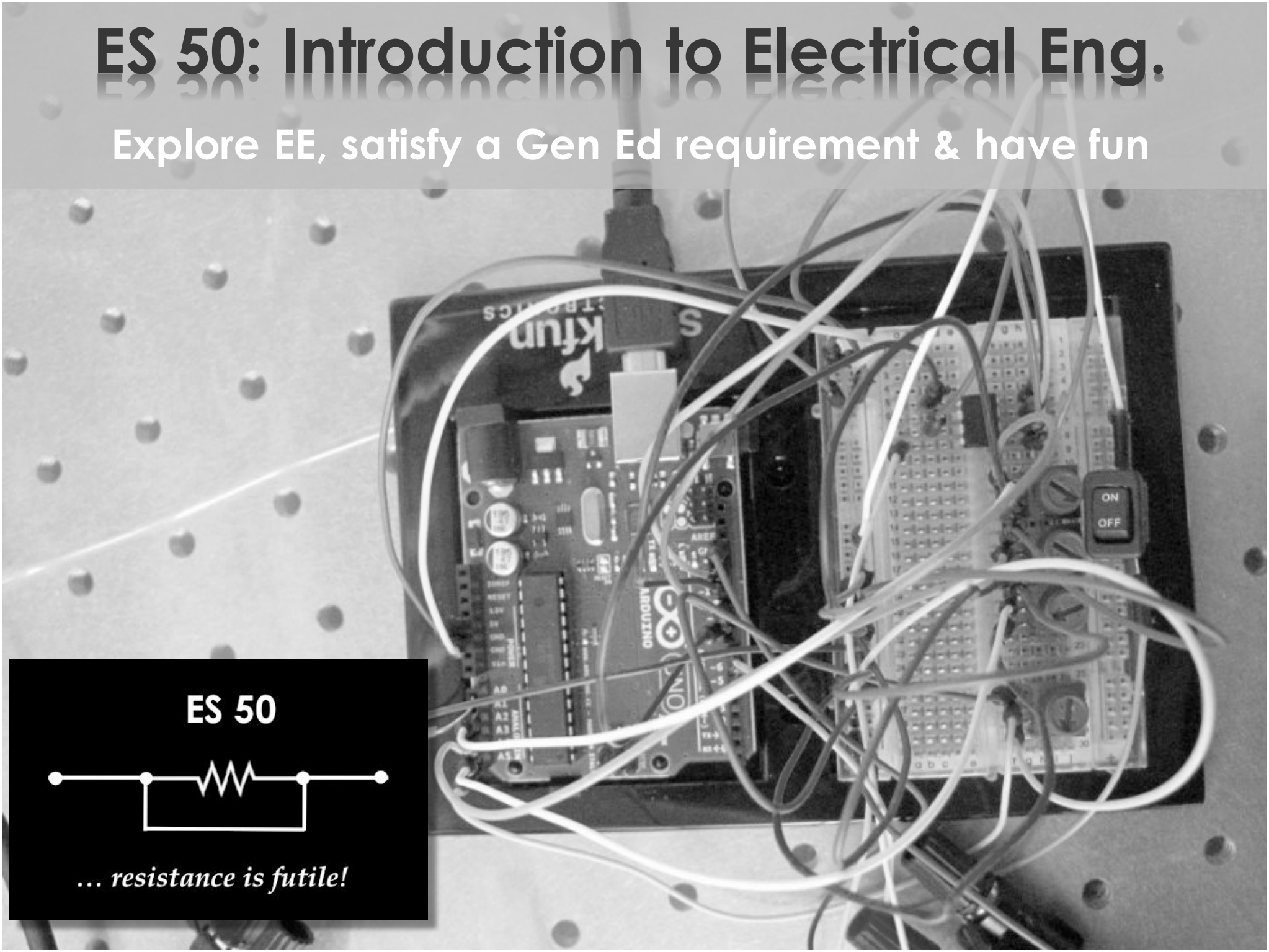
ES 50: Introduction to Electrical Eng.

Explore EE, satisfy a Gen Ed requirement & have fun



ES 50

… resistance is futile!

Lecture 11: Combinational Logic

October 21, 2015

HARVARD

Upcoming Events

Lab 7 - Project Proposal

- Not a typical lab, please choose your final project team (3-4 people)
- You will brainstorming and beginning to write you final project proposal
- You may go to an alternative lab time so your group can meet together BUT email the lab TFs for the lab time you intend to go to ahead of time

Make up Labs

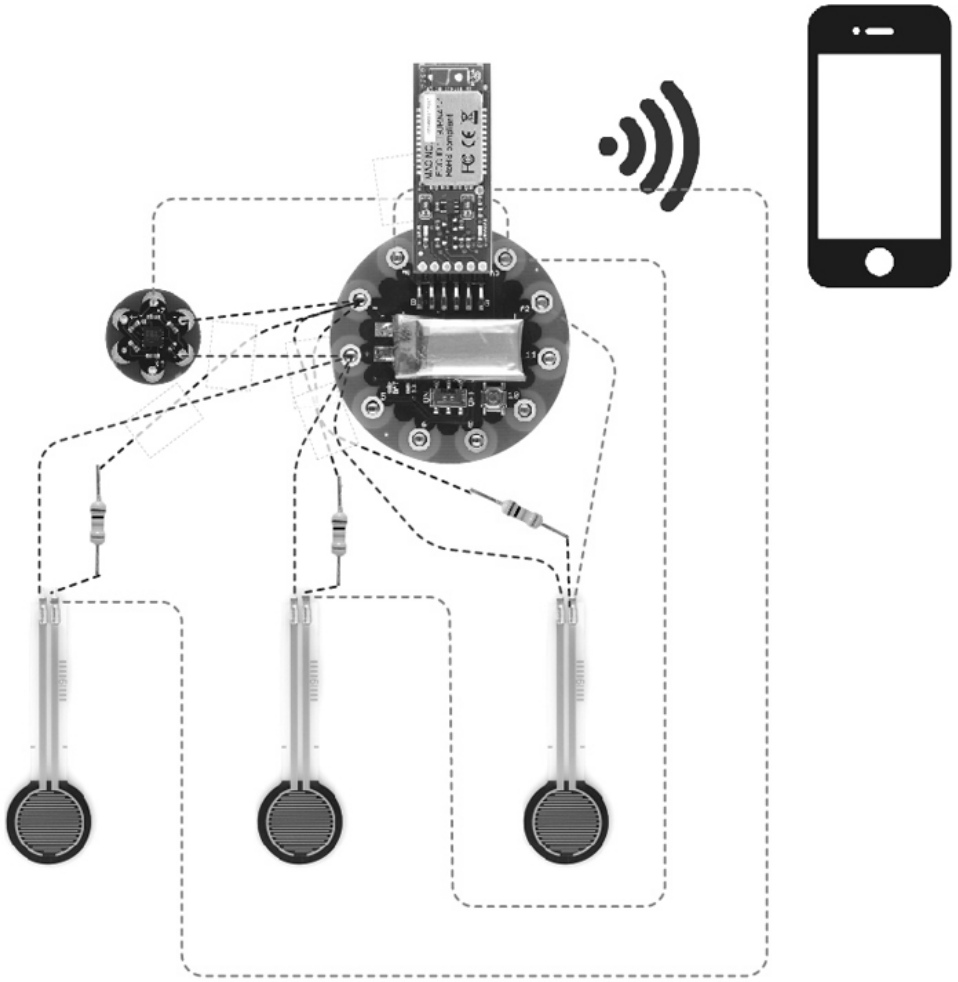
- Able to make up labs 1-6 during the week of Oct 26-30
- Google form will be sent out via CANVAS - http://goo.gl/forms/3lwQ3X6dXN
- You must sign up in order to make up a lab

Class Wed Oct 28

- Guest Speaker, Katie Cagen '14, Microsoft Hardware
- Former EE student and ES 50 TF

HARVARD

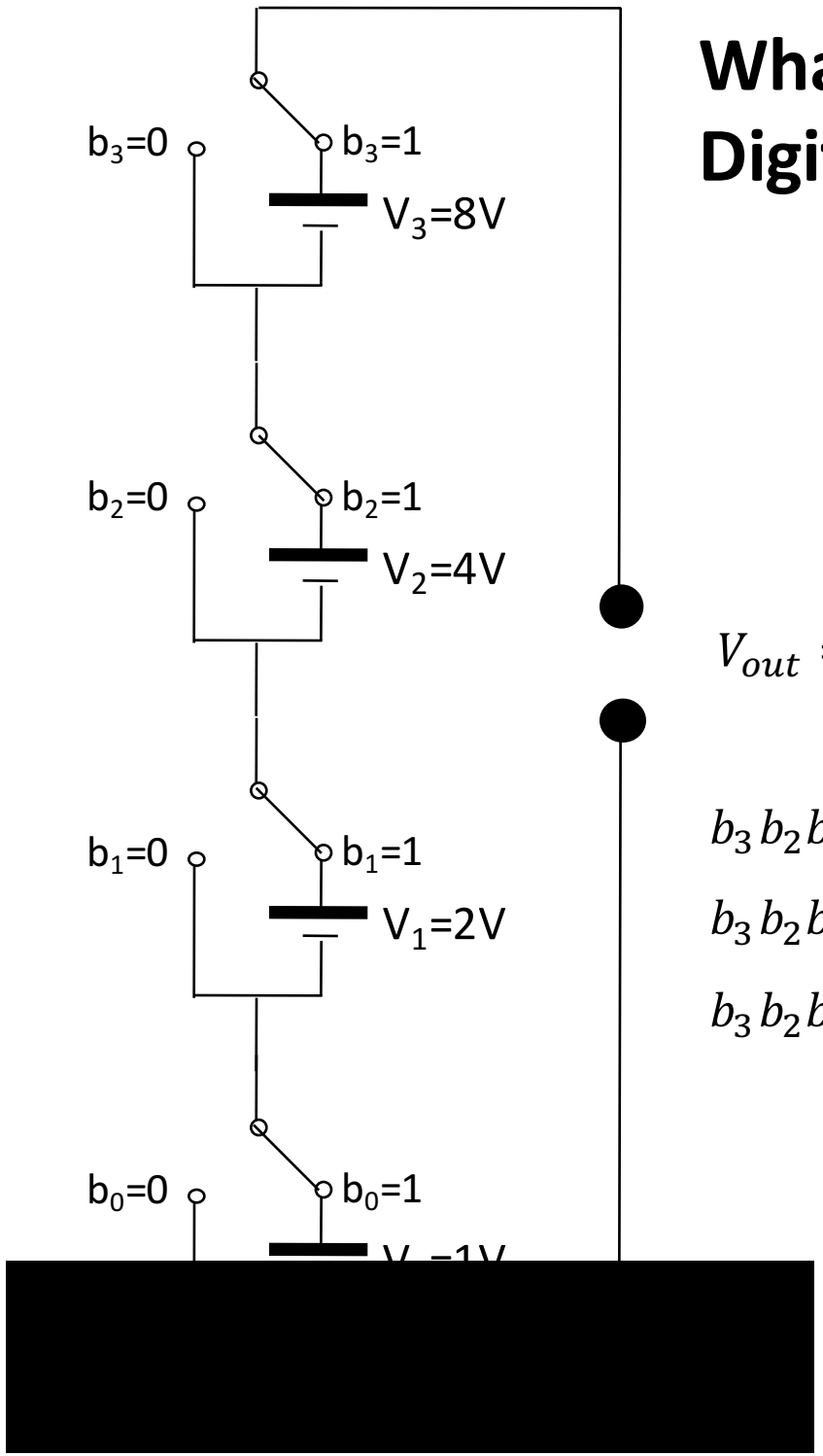
Final Project Inspriation



https://vimeo.com/108109673

http://cargocollective.com/lesiatrubat/E-TRACES-memories-of-dance

HARVARD



What about **Digital to Analog** Converter **(DAC) ?**

$$V_{out} = b_3 V_3 + b_2 V_2 + b_1 V_1 + b_0 V_0$$

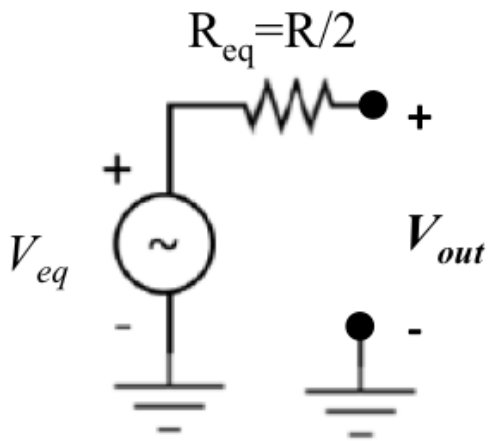
$b_3 b_2 b_1 b_0 = 1001 \rightarrow V_{out} = 8\,V + 1\,V = 9\,V$, YES!

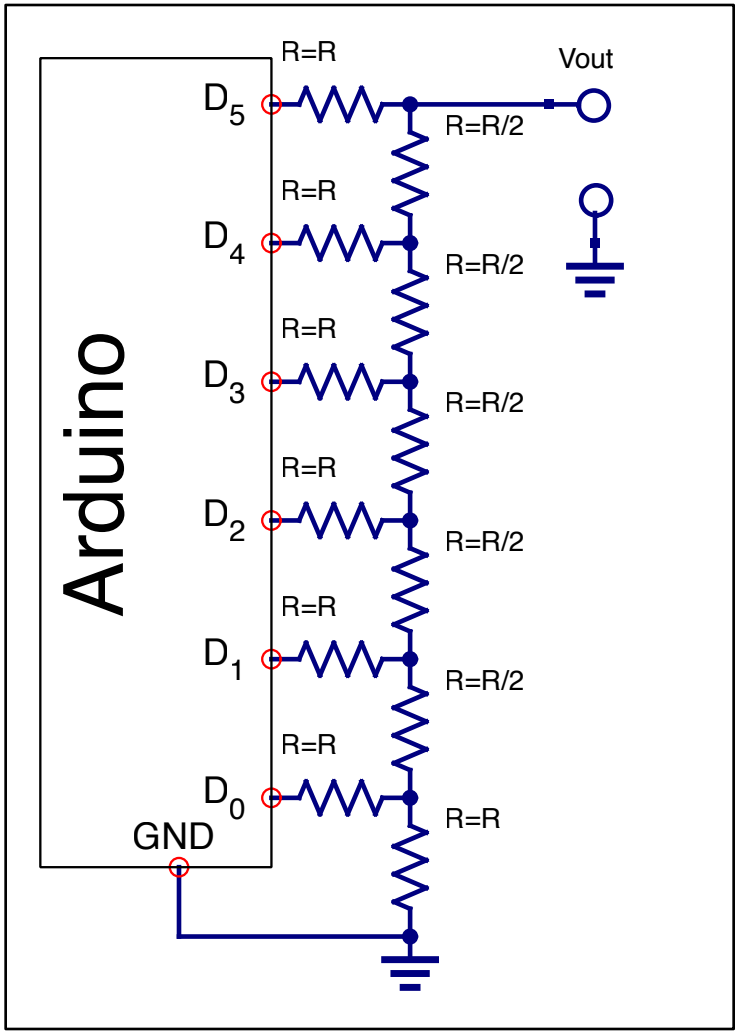
$b_3 b_2 b_1 b_0 = 0011 \rightarrow V_{out} = 2\,V + 1\,V = 3\,V$, YES!

$b_3 b_2 b_1 b_0 = 1111 \rightarrow V_{out} = 8\,V + 4V + 2\,V + 1\,V = 3\,V$, YES!

Simpler Digital to Analog Conversion!

$$V_{eq} = \frac{1}{2^6}(D_5 2^5 + D_4 2^4 + D_3 2^3 + D_2 2^2 + D_1 2^1 + D_0 2^0)$$





HARVARD

Truth Tables for Complex Circuits

- A truth table lists all possible combinations of the input variables and specifies the output for each input combination
- If a logic circuit has N inputs, how many possible combinations are there?
  - The most systematic way to arrange the table is to arrange inputs in numerical order starting with 0.
  - Once the truth table is known, it is EASY to figure out equation that describes the dependence of the outputs (they may be many of them) on the inputs.

How do we find circuit that represents this table?

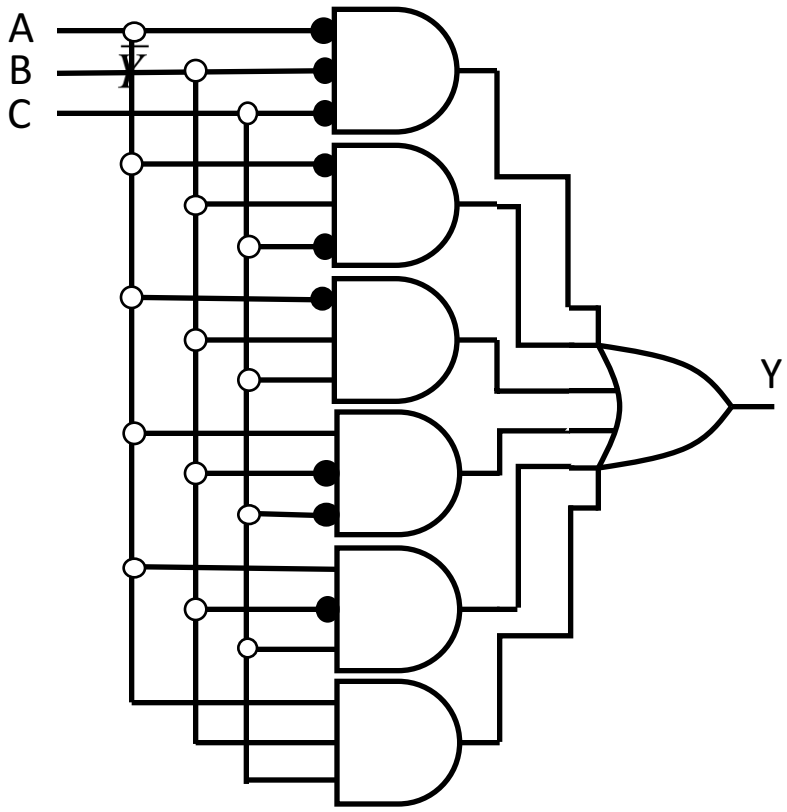
1. Find all combinations that result in output $Y = 1$
2. Since $Y = 1$ when ***any*** of these combinations is present, we join them using OR gate!

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$$Y = A'B'C' + A'BC' + A'BC + AB'C' + AB'C + ABC$$

HARVARD

And the circuit that does this is



$$Y = A'B'C' + A'BC' + A'BC + AB'C' + AB'C + ABC$$

Cool! But is this the only/optimal solution? Can we use fewer gates?

- Well, our systematic approach does not always give the simplest solution. We can simply the answer using the rules we just learned

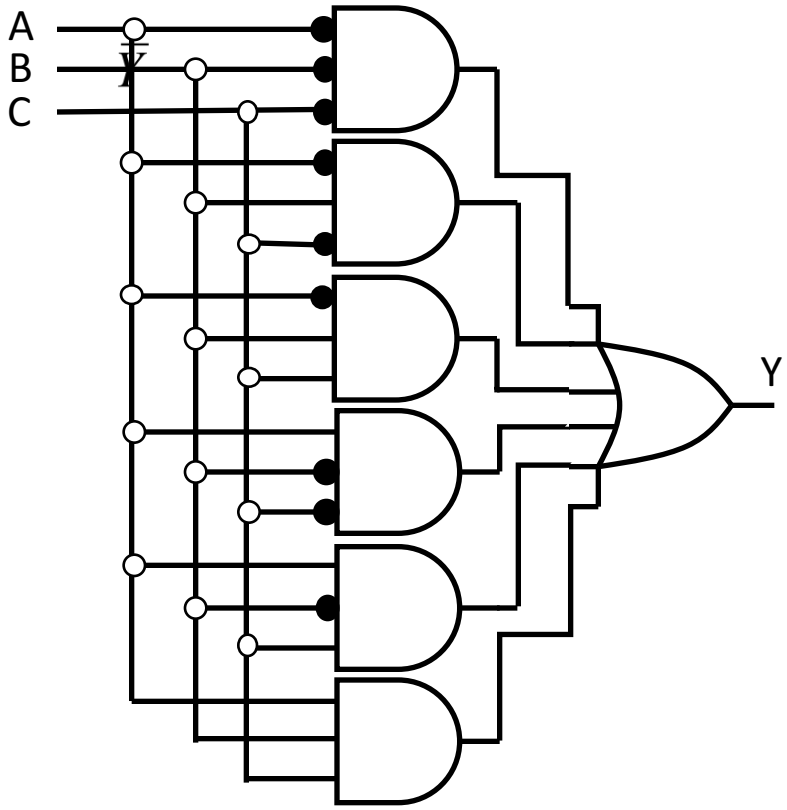
Example: Consider the complement

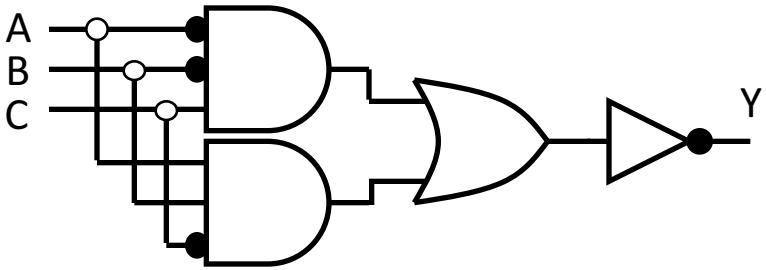
A	B	C	Y	Y'
0	0	0	1	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

$$Y' = A'B'C + ABC'$$

HARVARD

Simpler circuit





$$Y' = A'B'C + ABC'$$

In ES 50 we will use Boolean Algebra to simplify the logic expressions and obtain simpler circuits

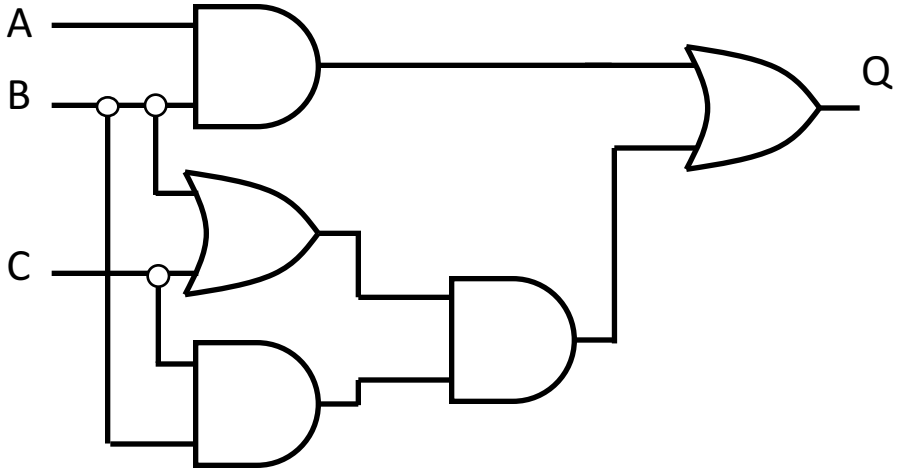
- This, however, does not guarantee that we will find the optimal solution!

$$Y = A'B'C' + A'BC' + A'BC + AB'C' + AB'C + ABC$$

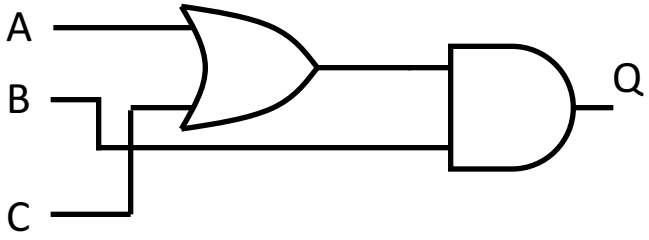
HARVARD

There is a systematic way to find the optimal solution: Karnaugh map (check wikipedia, as it is beyond this course)

An Example



$Q = AB + (B + C)BC = B(A + C)$



Find Q in terms of A, B and C

$$Q = AB + (B + C)BC$$

Simplify the expressions using the Boolean Algebra rules

- Distribute BC

1. $Q = AB + BBC + BCC$
   - $BB = B$ & $CC = C$
2. $Q = AB + BC + BC$
   - Factor out B
3. $Q = B(A + C + C)$
   - $C + C = C$
4. $Q = B(A + C)$
   - This is only 2 gates instead of 5!!!

VE RI TAS HARVARD

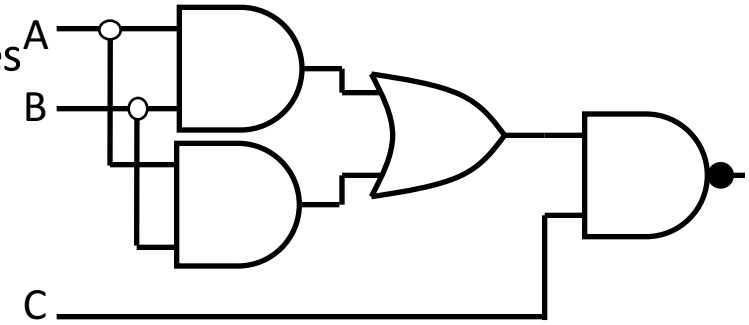
Example: 2011 Quiz

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

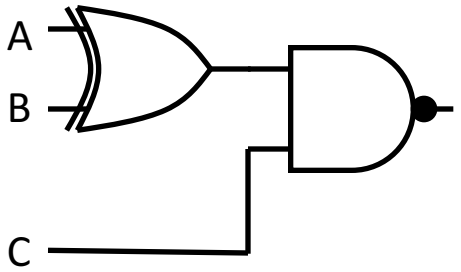
Consider the logic function described by the truth table:

1. Find & simplify the expression for Y in terms of A, B, & C
   - $Y = A'B'C' + A'B'C + A'BC' + AB'C' + ABC' + ABC$, but we can do better
   - $Y' = A'BC + AB'C$
   - $Y = (A'BC + AB'C)'$

1. Implement the expression using only 2-input logic gates
   - $Y = (A'BC + AB'C)' = [(A'B + AB')C]'$



2. Implement the above expression using ONLY TWO 2-input logic gates of any kind
   - $Y = [(A'B + AB')C]' = [(A \oplus B)C]'$



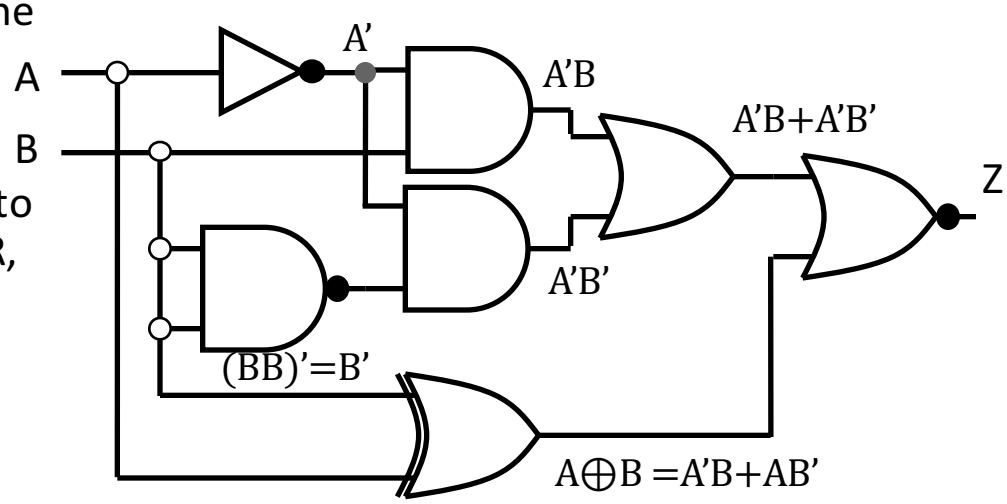
HARVARD

Example: 2011 Quiz

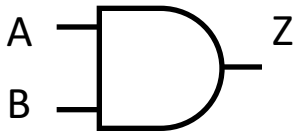
For the logic circuit shown:

a) Find Z as a function of A & B. If you like you may write the equation directly (no need to show the truth table).
   - $Z = \overline{(A'B + A'B') + (A'B + AB')}$

b) Simplify the equation. Hint: it can be reduced to ONE of the following gates AND, OR, NAND, OR, XOR, XNOR.
   - $Z = \overline{A'B + A'B' + A'B + AB'}$
     - $A'B + A'B = A'B$
   - $Z = \overline{A'B + A'B' + AB'} = \overline{A'(B + B') + AB'}$
     - $B + B' = 1$
   - $Z = \overline{A' + AB'} = \overline{A'}\left(\overline{AB'}\right) = A(A' + B)$
     - DeMorgan's Theorem
   - $Z = AA' + AB$
     - $AA' = 0$
   - $Z = AB$



$$Z = AB$$



HARVARD

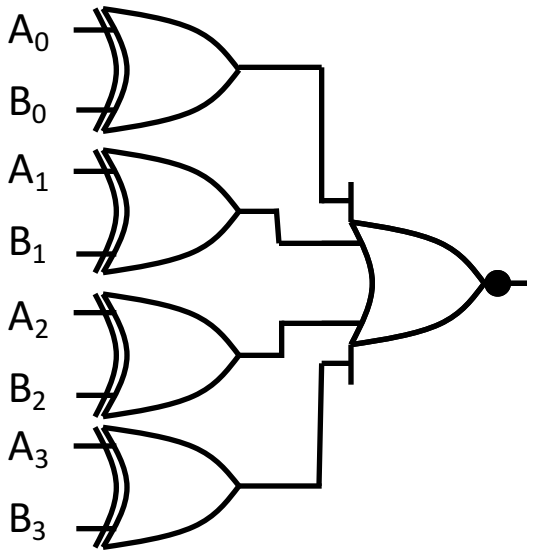
4 bit Simple Comparator

Each bit is compared by an XOR gate

- If bits are different XOR goes to 1

XORs are compared but a NOR gate

- If any input of NOR is 1 , the output is 0
- Only if all inputs are 0, is output 1



A_n	B_n	XOR
0	0	0
0	1	1
1	0	1
1	1	0

HARVARD

Let's add single-digit binary numbers

decimal	binary
0+0=0	0+0=0
0+1=1	0+1=1
1+0=1	1+0=1
1+1=2	1+1=10

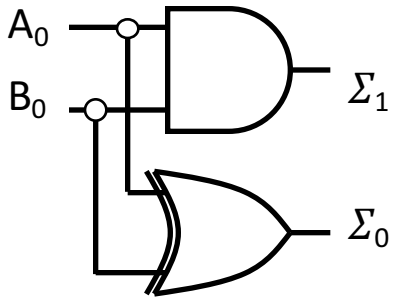
Another way to understand this addition is to consider following trust table that shows all possible outcomes when two binary numbers are added:

$$A_0 + B_0 = \Sigma_1 \Sigma_0$$

A_0	B_0	Σ_1	Σ_0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

It seems that:

- Bit Σ_0 of the sum is 1 ONLY when A_0 and B_0 are different
- Bit Σ_1 of the sum is 1 ONLY when A_0 and B_0 are both 1



So what does this look like:

- $\Sigma_0 = A_0 \oplus B_0$ A_0 XOR B_0
- $\Sigma_1 = A_0 B_0$ A_0 AND B_0

HARVARD

Amazing!!! We can use switches to do binary math!!!
But does this scale?

2-bit Adder

Let's try adding two 2-digit numbers:

$$A_1A_0 + B_1B_0 = \Sigma_2\Sigma_1\Sigma_0$$

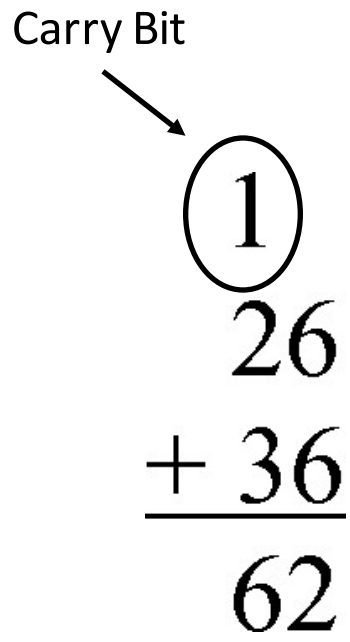
Okay, so this works, but it is painful… so there must be an easier way?

- Of course there is, but we need to learn a few more things!

A_1	A_0	B_1	B_0	Σ_2	Σ_1	Σ_0	DEC
0	0	0	0	0	0	0	0+0=0
0	0	0	1	0	0	1	0+1=1
0	0	1	0	0	1	0	0+2=2
0	0	1	1	0	1	1	0+3=3
0	1	0	0	0	0	1	1+0=1
0	1	0	1	0	1	0	1+1=2
0	1	1	0	0	1	1	1+2=3
0	1	1	1	1	0	0	1+3=4
1	0	0	0	0	1	0	2+0=2
1	0	0	1	0	1	1	2+1=3
1	0	1	0	1	0	0	2+2=4
1	0	1	1	1	0	1	2+3=5
1	1	0	0	0	1	1	3+0=3
1	1	0	1	1	0	0	3+1=4
1	1	1	0	1	0	1	3+2=5
1	1	1	1	1	1	0	3+3=6

HARVARD

n-bit Adder



Now let's try adding 2 3-bit numbers now (more painful)

- $A_2A_1A_0 + B_2B_1B_0 = \Sigma_3\Sigma_2\Sigma_1\Sigma_0$

But this can be written in another form with "carry bits"

- $C_1C_0C_{-1}$, What are carry bits?
  - Think back to elementary school addition long addition
- Now let's see how we add any two bits with a carry bit

C_{n-1}	A_n	B_n	C_n	Σ_n
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\Sigma_n = A_n{'}B_n{'}C_{n-1} + A_n{'}B_nC_{n-1}{'} + A_nB_n{'}C_{n-1}{'} + A_nB_nC_{n-1}$$

$$C_n = A_n{'}B_nC_{n-1} + A_nB_n{'}C_{n-1} + A_nB_nC_{n-1}{'} + A_nB_nC_{n-1}$$

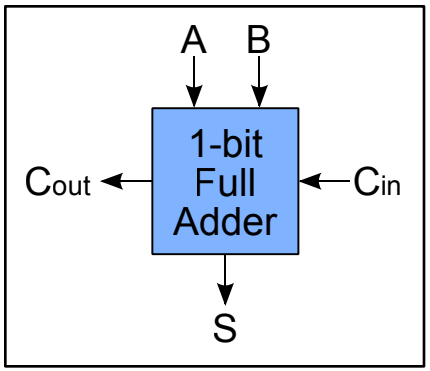
HARVARD

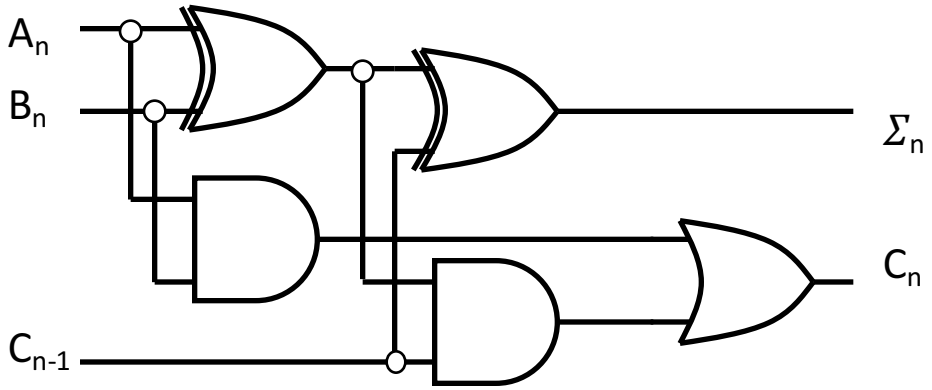
n-bit Adder

Let's simplify these expressions:

- $\Sigma_n = A_n{}'B_n{}'C_{n-1} + A_n{}'B_nC_{n-1}{}' + A_nB_n{}'C_{n-1}{}' + A_nB_nC_{n-1}$
  - $\Sigma_n = (A_n{}'B_n + A_nB_n{}')C_{n-1}{}' + (A_n{}'B_n{}' + A_nB_n)C_{n-1}$
  - $\Sigma_n = (A_n \oplus B_n)C_{n-1}{}' + (A_n \oplus B_n)'C_{n-1}$
  - $\Sigma_n = A_n \oplus B_n \oplus C_{n-1}$

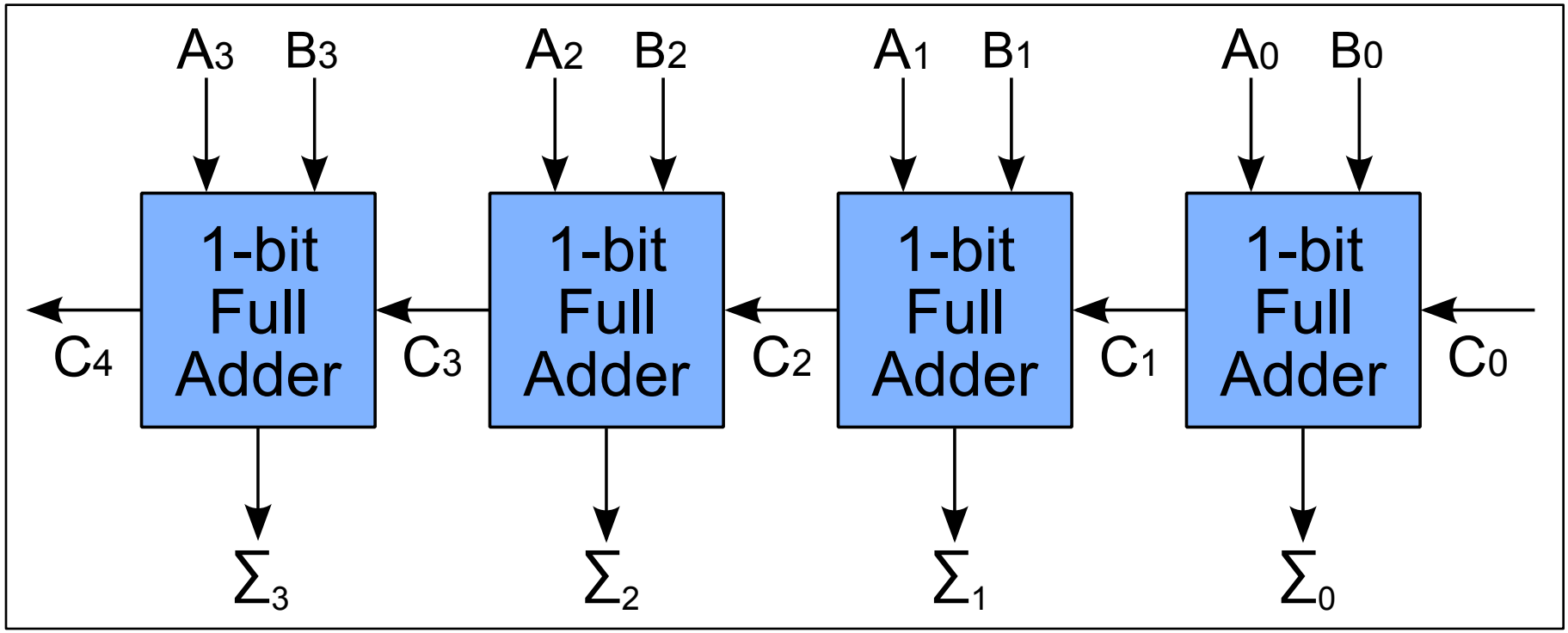
- $C_n = A_n{}'B_nC_{n-1} + A_nB_n{}'C_{n-1} + A_nB_nC_{n-1}{}' + A_nB_nC_{n-1}$
  - $C_n = A_nB_n(C_{n-1} + C_{n-1}{}') + (A_n{}'B_n + A_nB_n{}')C_{n-1}$
  - $C_n = A_nB_n + (A_n \oplus B_n)C_{n-1}$





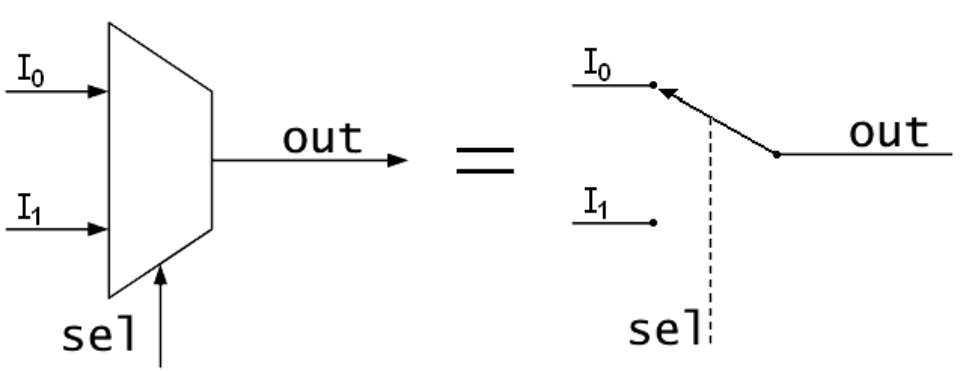
HARVARD

4-bit Adder

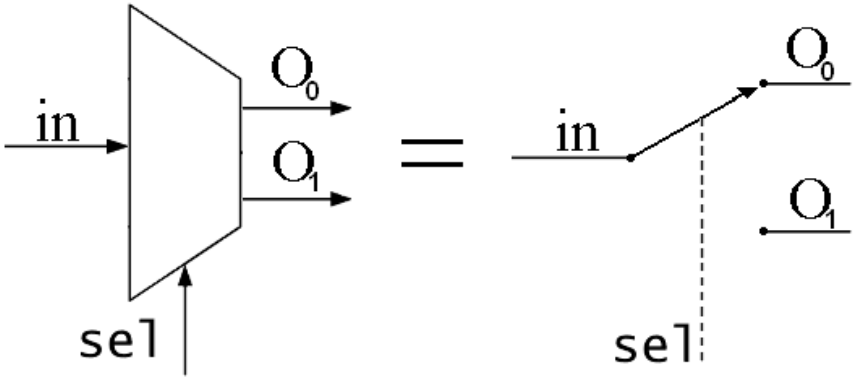


HARVARD

Multiplexer & De-Multiplexer



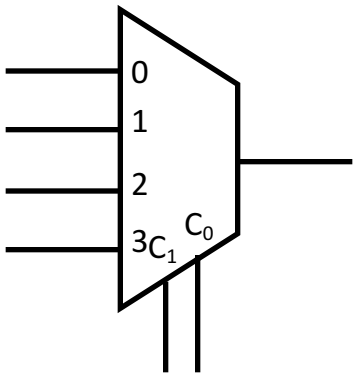
2 – to – 1 multiplexer (MUX)



1 – to – 2 demultiplexer (DeMUX)

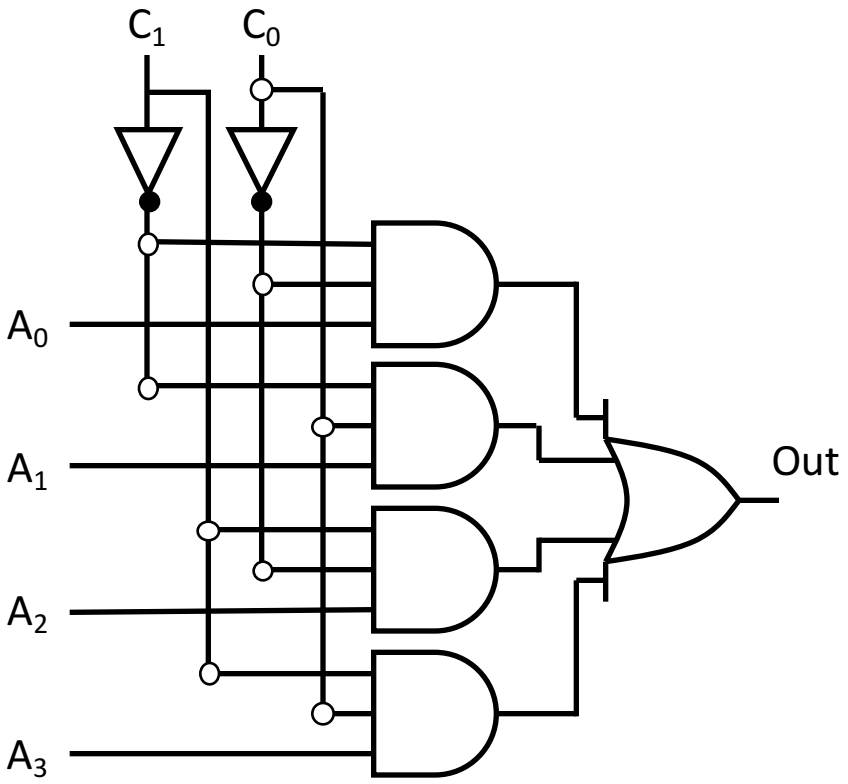
HARVARD

4 to 1 MUX



A_0, A_1, A_2 and A_3 are inputs
C_1 and C_0 are control signals

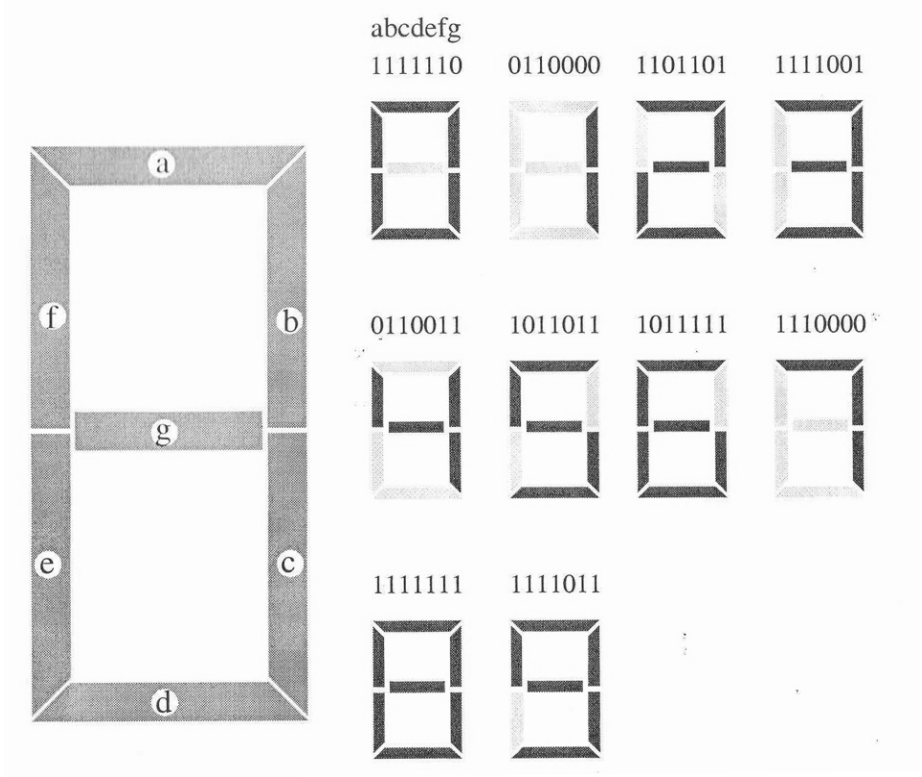
C_1	C_0	Out
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3



$$Out = C_1'C_0'A_0 + C_1'C_0A_1 + C_1C_0'A_2 + C_1C_0A_2$$

HARVARD

7-segment Display



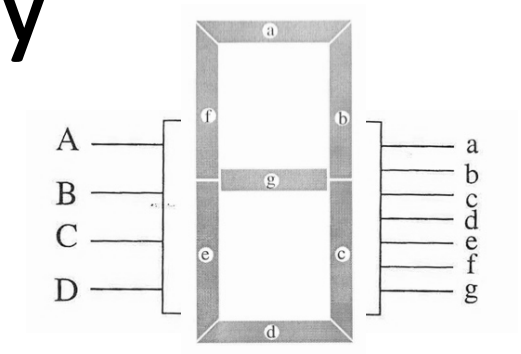
1. Create a 7-segment pattern that can be used to form all the digits from 0 to 9
2. Label those segments a – g; associate a digital code to each of the letters: either 0 (off) or 1 (on)

3. Consider the BINARY REPRESENTATIONS of those digits

Base 10	Base 2				Output						
Dec	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

HARVARD

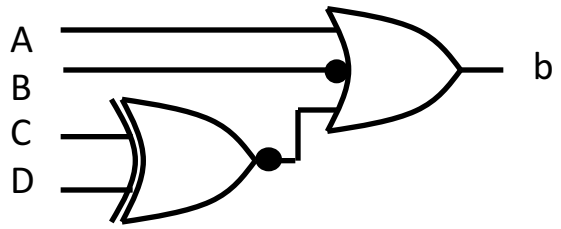
7-segment Display



How do we make a logical association between the binary number and the segments a-g?

- How do we 'turn on' the segments a-g to give a representation of 0-9?
- Let's use segment b for an example…

$$\overline{b} = A'BC'D + A'BCD'$$
$$\overline{b} = A'B(C'D + CD') = A'B(C \oplus D)$$
$$b = \overline{A'B(C \oplus D)} = A'' + B' + (C \oplus D)'$$
$$b = A + B' + \overline{C \oplus D}$$



Base 10	Base 2				Output						
Dec	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

HARVARD