POTW solution 8/8 - 8/15

We will first solve the following problem algebraically:
We determine how to decompose any binary function of several binary variables as a composition of AND ($\wedge$), OR ($\vee$) and NOT ($\neg$) gates/operations.
For example:

A	B	f
0	0	0
0	1	1
1	0	0
1	1	1

We can decompose f as $(\neg A) \wedge B) \vee (A \wedge B)$
Our main strategy is to use the 'linearity' of $\vee$ operation.
By which we mean thinking of $\vee$ as + we have $T + F = T$; $T + T = T$
$T + F + F = T$ and so on.
The central idea is to write any function 'f' as a sum of 'bump' functions, analagous to a basis.

A 'bump' function is any function g which is 1 or (T) for one particular tuple of the variables and zero for all others.

For example:

A	B	g
0	0	0
0	1	1
1	0	0
1	1	0

g is the bump corresponding to (A, B) = (0, 1)

Notice that any bump function can be written as the 'product' of the variables, for eg. $g = (\neg A) \wedge (B)$ for our previous example.

It is clear that any variable X with a value of 0 in the bump tuple occurs with a '¬' in front of it in the product.

It can be seen now that any general function can be written as the 'sum' of 'bump' functions for ~~with~~ the tuples on which it assumes the value 1

for eg.

A	B	C	f	
0	0	0	1	$g_1 = (\neg A) \wedge (\neg B) \wedge (\neg C)$
0	0	1	0	
0	1	0	0	
0	1	1	1	$g_2 = (\neg A) \wedge B \wedge C$
1	0	0	0	
1	0	1	1	$g_3 = A \wedge (\neg B) \wedge C$
1	1	0	0	
1	1	1	1	$g_4 = A \wedge B \wedge C$

clearly $f = g_1 \vee g_2 \vee g_3 \vee g_4$

Notice that this is certainly not the most efficient way of decomposing f, but it gets the job done.

Now all that needs to be done is to express $\vee$, $\wedge$ and $\neg$ as a composition of $\otimes$ NANDS.
But this is clear:

$$\neg A = A \otimes A$$
$$A \wedge B = (A \otimes B) \otimes (A \otimes B) = \neg(A \otimes B)$$
$$A \vee B = (\neg A) \otimes (\neg B)$$

Finally building a NAND gate from transistors is clear from the following circuit: