University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150
Spring 2000

J. Wawrzynek
D. Pertula

LUTs, Truth Tables, and Configuration Bits

Here I (DP) will explain the relationship between LUTs, truth tables, and configuration bits, and I will clarify the example I showed in my Monday section. I'll use mostly 2-input LUTs for simplicity. In real life, 2-LUTs are not an efficient use of resources; LUTs have 4 or 5 inputs.

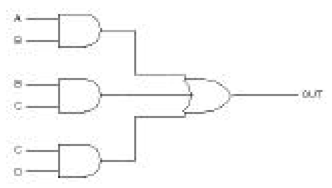
The relationship between logic, truth tables, and LUT configuration bits

Here is the same function expressed four ways:

In words:

Are there any adjacent ones in the inputs a, b, c, and d?

As logic:



As a truth table:

A	B	C	D	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

The output here could come from evaluating the logic to the left or directly from the problem specification.

As configuration bits in a LUT:

0 0 0 1 0 0 1 1

0 0 0 1 1 1 1 1

The LUT will use its inputs as a binary number to look up the answer in the list

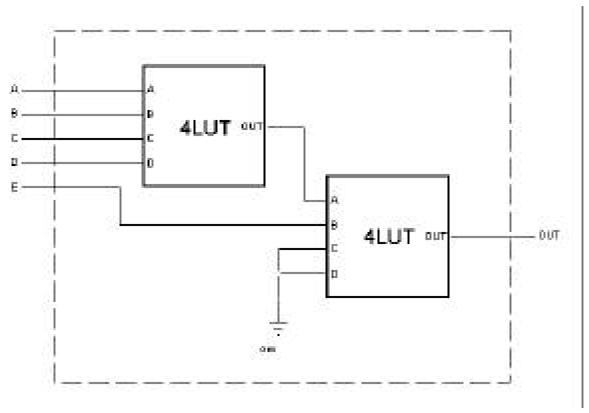
LUT sizes

An n-input function needs 2^n rows in its truth table. The LUT that performs the function will have 2^n (corrected by DP 2/11/2000) configuration bits.

The number of functions an n-input LUT can perform is $2^{\#\ config\ bits}$, because each different bit combination represents a different set of outputs.

5-LUT from 4-LUTs

One homework question asks for a 5-input LUT made of 4-LUTs. Here is a *wrong* answer:



This takes 5 inputs, has one output, and is made completely from programmable 4-LUTs. But it is not a complete 5-LUT; it cannot perform any function of 5 inputs.

To see this, consider a function that has the beginning of a truth table like this:

A	B	C	D	E	OUT
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	0	1	0

Think about the first LUT's output signal. It is a function of A, B, C, and D only, which means that it has the same value for truth table rows 1 and 2. Suppose it's a 0 for those inputs.

Next, consider rows 2 and 3. The first LUT's output has to be constant for these rows, too, because they also share ABCD values. The intermediate output also has to be the opposite for these rows, because the output behavior is different. If the intermediate output was the same for ABCD=0000 and ABCD=0001, the output could not possibly be different for inputs 00000 and 00010. Suppose the intermediate output is a 1 for those inputs.

Notice that the output is not dependent on E for any of the first 4 rows. So whether the first LUT outputs a 0 or 1, the second LUT must not change when E changes. And we know that the first LUT will output both those values for rows 1-4, because it needs to output two different values for the first two row pairs to be any different.

But now one of rows 5 and 6 is impossible! The first LUT output is necessarily the same for those rows, because ABCD don't change. It looks like the output will have to change with E. This would be a contradiction in the second LUT, because we already confirmed that its output does not change with E.

Another way to prove that this approach doesn't work is to count configuration bits. We know that a real 5-LUT needs 32 configuration bits. My circuit here has 16 bits in the first LUT, but only uses 4 bits of the second LUT. The second LUT may have 32 configuration bits, but since C and D are tied to ground, any truth table row with non-zero C or D is never going to be used, so its output bit will never matter. There are only 4 rows in the 4-LUT truth table with CD=00, so there are only 4 useful configuration bits. My design has 20 configuration bits that affect its output. That means it can perform 2^{20} different functions, which isn't enough. A real 5-LUT can perform 2^{32} different functions (each of 32 input combinations can make one of 2 outputs).