MITOCW | MIT6_004S17_04-02-08-04_300k

Boolean logic can be used to simplify logic equations into their simplest form.

This simplest form is known as the function's minimal sum of products representation.

Having a simplified equation allows us to implement the function using the fewest number of gates. One commonly used method for taking any boolean expression and converting it to its minimal sum-of-products equivalent is the use of Karnaugh maps. The way Karnaugh maps work is as follows.

You start with the original truth table specification of a function.

You then express each set of inputs, that makes the function equal to 1, as a basic product term. ORing all of these product terms together produces a sum of products representation for the function.

Note that this representation is not yet minimal. For our example, the sum of products is F = NOT(A) NOT(B) NOT(C) + NOT(A) B C + A NOT(B) NOT(C) + A NOT(B) C + A B C.

We then create the Karnaugh map for the function by making a 2D grid representing all possible input combinations, and ensuring that from one column or row to the next in the grid, only one single input can be changed. This is known as Gray code.

Once we label the Karnaugh map, we can fill it with 1's for each combination that produces a 1 output and 0 for each combination that produces a 0 output.

Next, we try to merge as many adjacent ones as possible into groups that are of size that is a power of 2. Note that adjacent ones can be merged across rows, along columns, and across the edges of the Karnaugh map like a taurus.

Every largest grouping of 1's that is required in order to cover all of the one's becomes one term in the minimal sum of products representation of the function.

To figure out what the term is, look at the labels of all the columns and rows that are covered by the grouping of 1's and eliminate all inputs that appear as both a 1 and 0 input.

Because these inputs are represented in both their 0 and 1 form, it means that that input is not affecting the result of that term and can thus be eliminated.

By doing this for all groupings of 1's and then ORing the results together, one produces a minimal sum of products representation for the function.

Note that having overlapping ones in your groupings is allowed and desired if it will result in a simpler product term. In our example, we circle the bottom two middle 1's which represent the term BC because A appears in both its low and high form in the grouping. The next set of ones are the two in the rightmost column. These ones represent the term A NOT(B).

Finally, our last grouping wraps around the first row to create the term NOT(B) NOT(C).

This is a minimal sum of products representation of our function.

Note, however, that instead of grouping the two one's in the rightmost column, we could have instead grouped the two rightmost ones in the bottom row.

That would have produced the term AC instead of A NOT(B).

Either combination of terms is a valid minimal sum of products representation for this function.