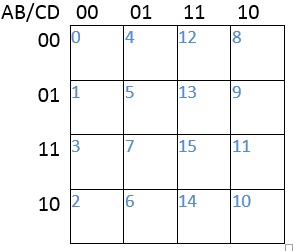
# Quine-McCluskey

* Note: The vast majority of this is taken from Prof. Steven Nowick of Columbia. <http://www.cs.columbia.edu/~cs6861/handouts/quine-mccluskey-handout.pdf>
* A briefer overview, including showing a relationship to K-maps, can be found at   
  <http://tinyurl.com/QM-dick>. It is written by our own Dr. Robert Dick.

Quine-McCluskey is a tabular method for finding the minimal sum-of-products. It is a method that is useful for two primary reasons:

1. It gives us a much more easy-to-program method than K-maps
2. It provides more formal detail about how to select prime implicants.

Steps are:

1. Generate Prime Implicants
2. Construct Prime Implicant Table
3. Reduce Prime Implicant Table (iterate until done…)
   1. Remove Essential Prime Implicants
   2. Row Dominance
   3. Column Dominance
4. Solve Prime Implicant Table

## Step 1: Generate Prime Implicants

List minterms by their “Hamming weight”—that is the # of 1s in them. Consider:



Now do a pair-wise check to see which terms can be combined with the grouping in front of them. Put a check next to each minterm that can be combined with another minterm.

Continue into column III, doing the same thing.



Column III contains a number of duplicate entries, *e.g.* (0,2,8,10) and (0,8,2,10). Duplicate entries appear

because a product in Column III can be formed in several ways. For example, (0,2,8,10) is formed by combining products (0,2) and (8,10) from Column II, and (0,8,2,10) (the same product) is formed by combining products (0,8) and (2,10). **Cross out the duplicate entries.**

That leaves: (0,2,8,10), (2,6,10,14), (5,7,13,15), (6,7,14,15), (8,10,12,14) and (12,13,14,15) or





Now cross out those rows and columns which are no longer needed.

Here’s what we have left:



### Row Dominance

We note that row 14 dominates both row 6 and row 12. That is row 14 has an X in every column where row 6 has an“X” (and, in fact, row 14 has “X”’s in other columns as well). Similarly, row 14 has in “X” in every column where row 12 has an “X”. Rows 6 and 12 are said to be *dominated by* row 14.

A *dominating* row can always be eliminated. To see this, note that every product which covers row 6 also

covers row 14. That is, if some product covers row 6, row 14 is *guaranteed* to be covered. Similarly, any

product which covers row 12 will also cover row 14. Therefore, row 14 can be crossed out.



## Iteration #2



In iteration #2 and beyond, *secondary essential prime implicants* are identified. These are implicants which will appear in *any* solution, *given* the choice of column-dominance used in the previous steps (if 2 columns co-dominated each other in a previous step, the choice of which was deleted can affect what is an “essential”at this step). As before, a row which is covered by only 1 prime implicant is called a *distinguished row*. The prime implicant which covers it is a *(secondary) essential prime implicant*.

Secondary essential prime implicants are identified and removed. The corresponding columns are crossed

out. Also, each row where the column contains an *X* is completely crossed out, since these minterms are

now covered. These essential implicants will be added to the final solution. In this example, both *CD’* and

*AD’* are secondary essentials.



## Let’s work our own problem:

Σ(A,B,C,D)=0,1,4,5,6,11,14. This example is more interesting when it comes to generating prime implicants, but much less interesting when reducing the prime implicant table.

### Step 1: Find prime implicants

### Step 2/3: Construct Prime Implicant table and reduce

# Clock dividers

Say I have a 1MHz input clock and I want a *glitch-free* 100KHz output clock. How can I use a counter and gates to achieve that goal? First worry about getting the clock divided, then worry about the glitch-free part!

**4-bit Counter**

En Q[3:0]

Clk

R

How do I change that if I want a 20% duty cycle? (Duty cycle means the percent of time that the signal is high). Still needs to be glitch free!

**4-bit Counter**

En Q[3:0]

Clk

R

And finally, how do I handle the generic problem (divide by X, high for Y cycles)?