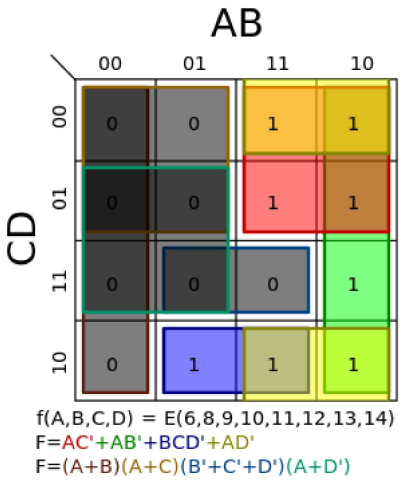
Karnaugh map

The **Karnaugh map** (**KM** or **K-map**) is a method of simplifying Boolean algebra expressions. Maurice Karnaugh introduced it in 1953[1][2] as a refinement of Edward W. Veitch's 1952 **Veitch chart**,[3][4] which was a rediscovery of Allan Marquand's 1881 *logical diagram*[5][6] aka **Marquand diagram**[4] but with a focus now set on its utility for switching circuits.[4] Veitch charts are also known as **Marquand–Veitch diagrams**[4] or, rarely, as **Svoboda charts**,[7] and Karnaugh maps as **Karnaugh–Veitch maps** (**KV maps**).

The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability.[1] It also permits the rapid identification and elimination of potential race conditions.

The required Boolean results are transferred from a truth table onto a two-dimensional grid where, in Karnaugh maps, the cells are ordered in Gray code,[8][4] and each cell position represents one combination of input conditions. Cells are also known as minterms, while each cell value represents the corresponding output value of the boolean function. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table.[9] These terms can be used to write a minimal Boolean expression representing the required logic.



An example Karnaugh map. This image actually shows two Karnaugh maps: for the function f, using minterms (colored rectangles) and for its complement, using maxterms (gray rectangles). In the image, $E()$ signifies a sum of minterms, denoted in the article as $\sum m_i$.

Karnaugh maps are used to simplify real-world logic requirements so that they can be implemented using a minimum number of logic gates. A sum-of-products expression (SOP) can always be implemented using AND gates feeding into an OR gate, and a product-of-sums expression (POS) leads to OR gates feeding an AND gate. The POS expression gives a complement of the function (if F is the function so its complement will be F').[10] Karnaugh maps can also be used to simplify logic expressions in software design. Boolean conditions, as used for example in conditional statements, can get very complicated, which makes the code difficult to read and to maintain. Once minimised, canonical sum-of-products and product-of-sums expressions can be implemented directly using AND and OR logic operators.[11]

Example

Karnaugh maps are used to facilitate the simplification of Boolean algebra functions. For example, consider the Boolean function described by the following truth table.

Truth table of a function

	A	B	C	D	$f(A,B,C,D)$
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Following are two different notations describing the same function in unsimplified Boolean algebra, using the Boolean variables A, B, C, D and their inverses.

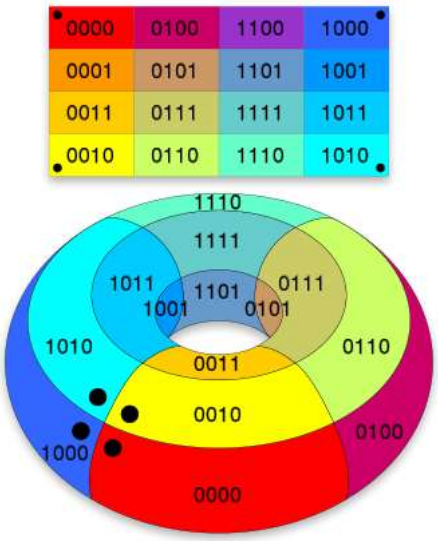
- $f(A,B,C,D) = \sum m_i, i \in \{6,8,9,10,11,12,13,14\}$ where m_i are the minterms to map (i.e., rows that have output 1 in the truth table).
- $f(A,B,C,D) = \prod M_i, i \in \{0,1,2,3,4,5,7,15\}$ where M_i are the maxterms to map (i.e., rows that have output 0 in the truth table).

Construction

In the example above, the four input variables can be combined in 16 different ways, so the truth table has 16 rows, and the Karnaugh map has 16 positions. The Karnaugh map is therefore arranged in a 4 × 4 grid.

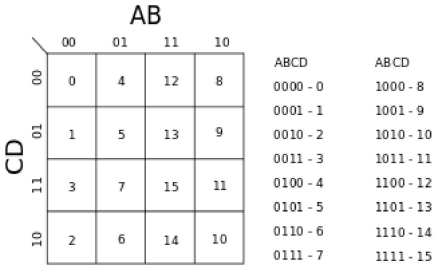
The row and column indices (shown across the top and down the left side of the Karnaugh map) are ordered in Gray code rather than binary numerical order. Gray code ensures that only one variable changes between each pair of adjacent cells. Each cell of the completed Karnaugh map contains a binary digit representing the function's output for that combination of inputs.

Grouping

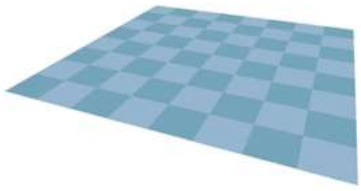


K-map drawn on a torus, and in a plane. The dot-marked cells are adjacent.

After the Karnaugh map has been constructed, it is used to find one of the simplest possible forms — a canonical form — for the information in the truth table. Adjacent 1s in the Karnaugh map represent opportunities to simplify the expression. The minterms ('minimal terms') for the final expression are found by encircling groups of 1s in the map. Minterm groups must be rectangular and must have an area that is a power of two (i.e., 1, 2, 4, 8...). Minterm rectangles should be as large as possible without containing any 0s. Groups may overlap in order to make each one larger. The optimal groupings in the example below are marked by the green, red and blue lines, and the red and green groups overlap. The red group is a 2 × 2 square, the green group is a 4 × 1 rectangle, and the overlap area is indicated in brown.



K-map construction. Instead of the output values (the rightmost values in the truth table), this diagram shows a decimal representation of the input ABCD (the leftmost values in the truth table), therefore it is not a Karnaugh map.



In three dimensions, one can bend a rectangle into a torus.

The cells are often denoted by a shorthand which describes the logical value of the inputs that the cell covers. For example, AD would mean a cell which covers the 2x2 area where A and D are true, i.e. the cells numbered 13, 9, 15, 11 in the diagram above. On the other hand, AD would mean the cells where A is true and D is false (that is, $\overline{D}$ is true).

The grid is toroidally connected, which means that rectangular groups can wrap across the edges (see picture). Cells on the extreme right are actually 'adjacent' to those on the far left, in the sense that the corresponding input values only differ by one bit; similarly, so are those at the very top and those at the bottom. Therefore, AD can be a valid term—it includes cells 12 and 8 at the top, and wraps to the bottom to include cells 10 and 14—as is BD, which includes the four corners.

Solution

Once the Karnaugh map has been constructed and the adjacent 1s linked by rectangular and square boxes, the algebraic minterms can be found by examining which variables stay the same within each box.

For the red grouping:

- A is the same and is equal to 1 throughout the box, therefore it should be included in the algebraic representation of the red minterm.
- B does not maintain the same state (it shifts from 1 to 0), and should therefore be excluded.
- C does not change. It is always 0, so its complement, NOT-C, should be included. Thus, $\overline{C}$ should be included.

- D changes, so it is excluded.

Thus the first minterm in the Boolean sum-of-products expression is $A\overline{C}$.

For the green grouping, A and B maintain the same state, while C and D change. B is 0 and has to be negated before it can be included. The second term is therefore $A\overline{B}$. Note that it is acceptable that the green grouping overlaps with the red one.

In the same way, the blue grouping gives the term $B\overline{CD}$.

The solutions of each grouping are combined: the normal form of the circuit is $A\overline{C} + A\overline{B} + B\overline{CD}$.

Thus the Karnaugh map has guided a simplification of

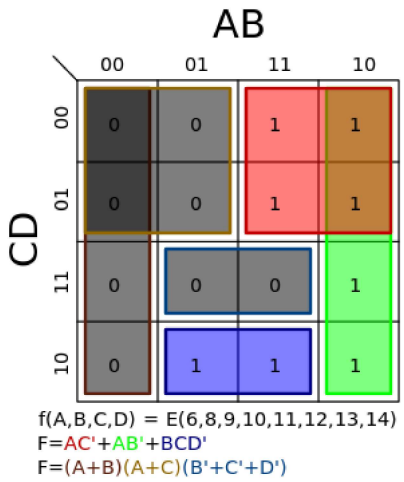


Diagram showing two K-maps. The K-map for the function f(A, B, C, D) is shown as colored rectangles which correspond to minterms. The brown region is an overlap of the red 2×2 square and the green 4×1 rectangle. The K-map for the inverse of f is shown as gray rectangles, which correspond to maxterms.

$$\begin{aligned} f(A,B,C,D) &= \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}C\overline{D} + \\ &\quad A\overline{B}CD + AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC\overline{D} \\ &= A\overline{C} + A\overline{B} + B\overline{CD} \end{aligned}$$

It would also have been possible to derive this simplification by carefully applying the axioms of Boolean algebra, but the time it takes to do that grows exponentially with the number of terms.

Inverse

The inverse of a function is solved in the same way by grouping the 0s instead.[nb 1]

The three terms to cover the inverse are all shown with grey boxes with different colored borders:

- brown: $\overline{A}\,\overline{B}$
- gold: $\overline{A}\,\overline{C}$
- blue: BCD

This yields the inverse:

$$\overline{f(A,B,C,D)} = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + BCD$$

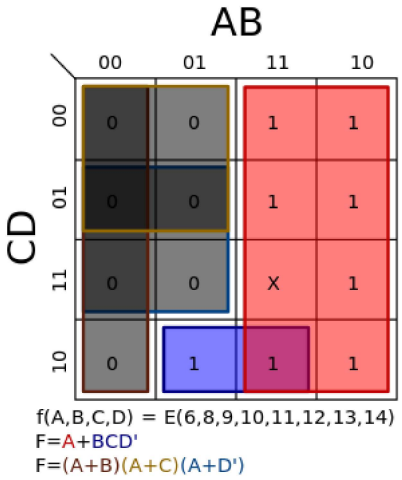
Through the use of De Morgan's laws, the product of sums can be determined:

$$
\begin{aligned}
f(A,B,C,D) &= \overline{\overline{f(A,B,C,D)}} \\
&= \overline{\overline{A}\,\overline{B} + \overline{A}\,\overline{C} + BCD} \\
&= \left(\overline{\overline{A}\,\overline{B}}\right)\left(\overline{\overline{A}\,\overline{C}}\right)\left(\overline{BCD}\right) \\
&= (A+B)\,(A+C)\left(\overline{B}+\overline{C}+\overline{D}\right)
\end{aligned}
$$

Don't cares

Karnaugh maps also allow easier minimizations of functions whose truth tables include "don't care" conditions. A "don't care" condition is a combination of inputs for which the designer doesn't care what the output is. Therefore, "don't care" conditions can either be included in or excluded from any rectangular group, whichever makes it larger. They are usually indicated on the map with a dash or X.

The example on the right is the same as the example above but with the value of $f(1,1,1,1)$ replaced by a "don't care". This allows the red term to expand all the way down and, thus, removes the green term completely.



The value of $f(A,B,C,D)$ for $ABCD = 1111$ is replaced by a "don't care". This removes the green term completely and allows the red term to be larger. It also allows blue inverse term to shift and become larger

This yields the new minimum equation:

$$f(A,B,C,D) = A + BC\overline{D}$$

Note that the first term is just A, not $A\overline{C}$. In this case, the don't care has dropped a term (the green rectangle); simplified another (the red one); and removed the race hazard (removing the yellow term as shown in the following section on race hazards).

The inverse case is simplified as follows:

$$\overline{f(A,B,C,D)} = \overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{A}D$$

Through the use of De Morgan's laws, the product of sums can be determined:

$$
\begin{aligned}
f(A,B,C,D) &= \overline{\overline{f(A,B,C,D)}} \\
&= \overline{\overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{A}\,D} \\
&= \left(\overline{\overline{A}\,\overline{B}}\right)\left(\overline{\overline{A}\,\overline{C}}\right)\left(\overline{\overline{A}\,D}\right) \\
&= (A+B)\,(A+C)\left(A+\overline{D}\right)
\end{aligned}
$$

Race hazards

Elimination

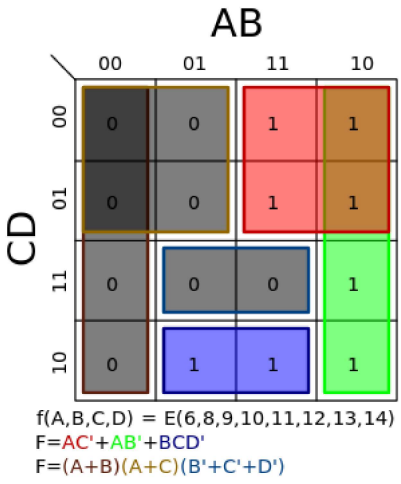
Karnaugh maps are useful for detecting and eliminating race conditions. Race hazards are very easy to spot using a Karnaugh map, because a race condition may exist when moving between any pair of adjacent, but disjoint, regions circumscribed on the map. However, because of the nature of Gray coding, *adjacent* has a special definition explained above – we're in fact moving on a torus, rather than a rectangle, wrapping around the top, bottom, and the sides.

- In the example above, a potential race condition exists when *C* is 1 and *D* is 0, *A* is 1, and *B* changes from 1 to 0 (moving from the blue state to the green state). For this case, the output is defined to remain unchanged at 1, but because this transition is not covered by a specific term in the equation, a potential for a *glitch* (a momentary transition of the output to 0) exists.
- There is a second potential glitch in the same example that is more difficult to spot: when *D* is 0 and *A* and *B* are both 1, with C changing from 1 to 0 (moving from the blue state to the red state). In this case the glitch wraps around from the top of the map to the bottom.

Whether glitches will actually occur depends on the physical nature of the implementation, and whether we need to worry about it depends on the application. In clocked logic, it is enough that the logic settles on the desired value in time to meet the timing deadline. In our example, we are not considering clocked logic.

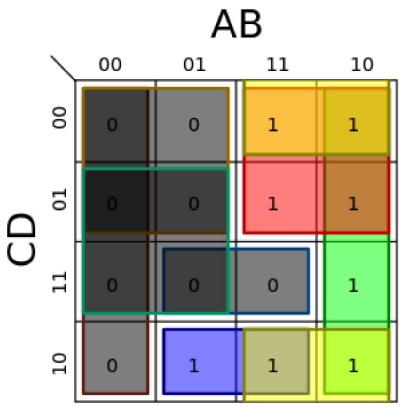
In our case, an additional term of $A\overline{D}$ would eliminate the potential race hazard, bridging between the green and blue output states or blue and red output states: this is shown as the yellow region (which wraps around from the bottom to the top of the right half) in the adjacent diagram.

The term is redundant in terms of the static logic of the system, but such redundant, or consensus terms, are often needed to assure race-free dynamic performance.



Race hazards are present in this diagram.

Similarly, an additional term of $\overline{A}D$ must be added to the inverse to eliminate another potential race hazard. Applying De Morgan's laws creates another product of sums expression for f, but with a new factor of $\left(A + \overline{D}\right)$.

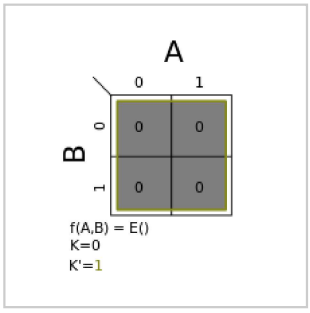


Above diagram with consensus terms added to avoid race hazards.

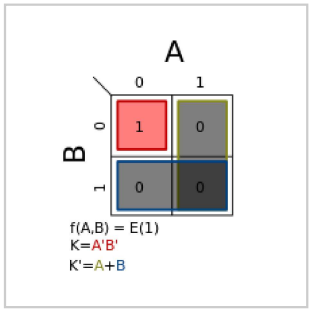
2-variable map examples

The following are all the possible 2-variable, 2 × 2 Karnaugh maps. Listed with each is the minterms as a function of $\sum m()$ and the race hazard free (*see previous section*) minimum equation. A minterm is defined as an expression that gives the most minimal form of expression of the mapped variables. All possible horizontal and vertical interconnected blocks can be formed. These blocks must be of the size of the powers of 2 (1, 2, 4, 8, 16, 32, ...). These expressions create a minimal logical mapping of the minimal logic variable expressions for the binary expressions to be mapped. Here are all the blocks with one field.

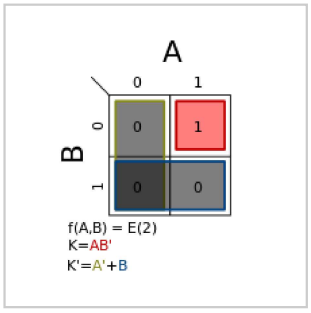
A block can be continued across the bottom, top, left, or right of the chart. That can even wrap beyond the edge of the chart for variable minimization. This is because each logic variable corresponds to each vertical column and horizontal row. A visualization of the k-map can be considered cylindrical. The fields at edges on the left and right are adjacent, and the top and bottom are adjacent. K-Maps for four variables must be depicted as a donut or torus shape. The four corners of the square drawn by the k-map are adjacent. Still more complex maps are needed for 5 variables and more.



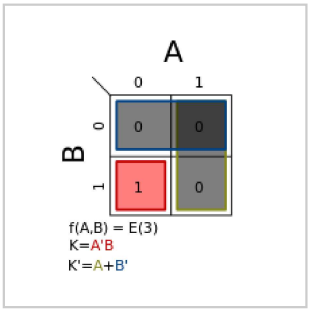
$\Sigma m(0)$; $K = 0$



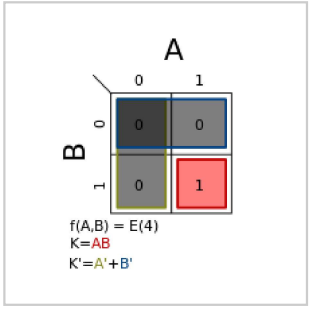
$\Sigma m(1)$; $K = A'B'$



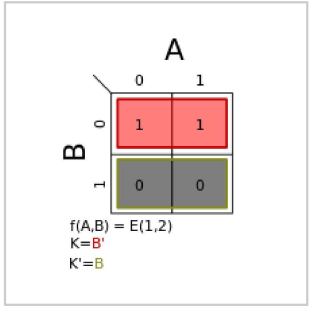
$\Sigma m(2)$; $K = AB'$



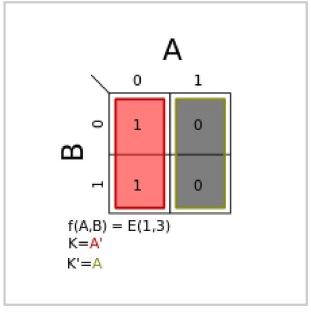
$\Sigma m(3)$; $K = A'B$



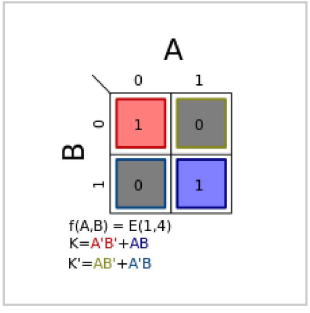
$\Sigma m(4)$; $K = AB$



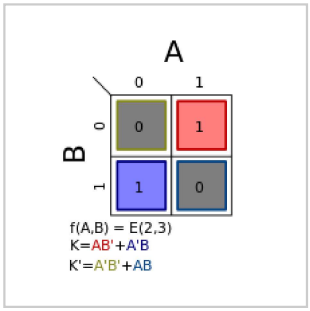
$\Sigma m(1,2)$; $K = B'$



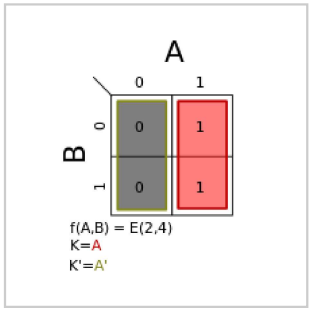
$\Sigma m(1,3)$; $K = A'$



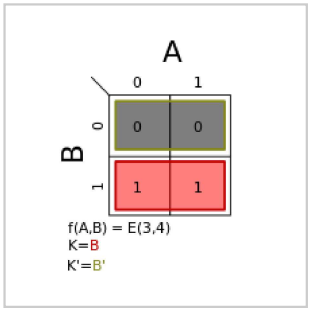
$\Sigma m(1,4)$; $K = A'B' + AB$



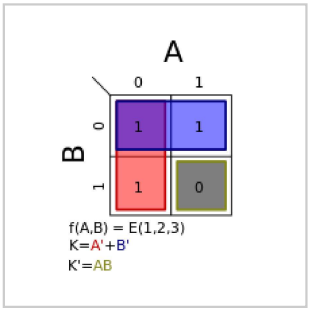
Σ*m*(2,3); $K = AB' + A'B$



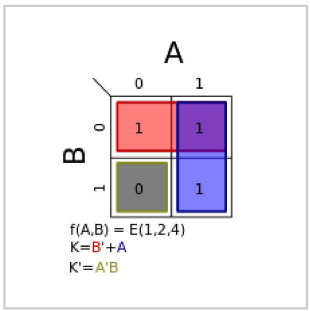
Σ*m*(2,4); $K = A$



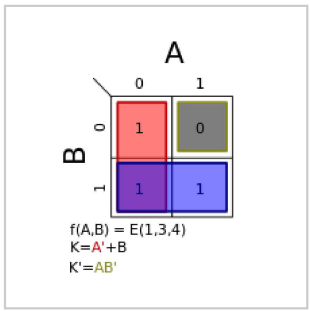
Σ*m*(3,4); $K = B$



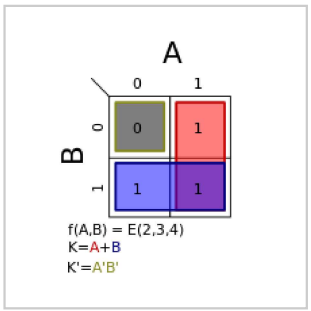
Σ*m*(1,2,3); $K = A' + B'$



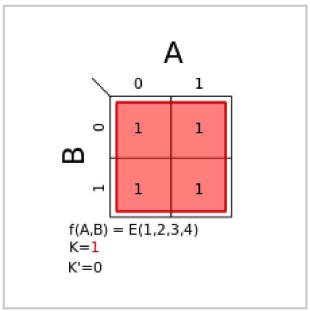
Σ*m*(1,2,4); $K = A + B'$



Σ*m*(1,3,4); $K = A' + B$



Σ*m*(2,3,4); $K = A + B$



Σ*m*(1,2,3,4); $K = 1$

Related graphical methods

Related graphical minimization methods include:

- *Marquand diagram* (1881) by Allan Marquand (1853–1924)[5][6][4]
- *Veitch chart* (1952) by Edward W. Veitch (1924–2013)[3][4]
- *Svoboda chart* (1956) by Antonín Svoboda (1907–1980)[7]
- *Mahoney map* (*M-map*, *designation numbers*, 1963) by Matthew V. Mahoney (a reflection-symmetrical extension of Karnaugh maps for larger numbers of inputs)
- *Reduced Karnaugh map* (RKM) techniques (from 1969) like *infrequent variables*, *map-entered variables* (MEV), *variable-entered map* (VEM) or *variable-entered Karnaugh map* (VEKM) by G. W. Schultz, Thomas E. Osborne, Christopher R. Clare, J. Robert Burgoon, Larry L. Dornhoff, William I. Fletcher, Ali M. Rushdi and others (several successive Karnaugh map extensions based on variable inputs for a larger numbers of inputs)
- *Minterm-ring map* (MRM, 1990) by Thomas R. McCalla (a three-dimensional extension of Karnaugh maps for larger numbers of inputs)

See also

- Algebraic normal form (ANF)
- Binary decision diagram (BDD), a data structure that is a compressed representation of a Boolean function
- Espresso heuristic logic minimizer