WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Nederlands
Polski
Српски / srpski
ไทย
Edit links

Espresso heuristic logic minimizer

From Wikipedia, the free encyclopedia

The **Espresso logic minimizer** is a computer program using heuristic and specific algorithms for efficiently reducing the complexity of digital electronic gate circuits.[1] Espresso was developed at IBM by Robert Brayton. Richard Rudell later published the variant Espresso-MV in 1986 under the title "Multiple-Valued Logic Minimization for PLA Synthesis".[2] Espresso has inspired many derivatives.

Contents [hide]

Introduction [edit]

Electronic devices are composed of numerous blocks of digital circuits, the combination of which performs the required task. The efficient implementation of logic functions in the form of logic gate circuits (such that no more logic gates are used than are necessary) is necessary to minimize production costs, and/or maximize a device's performance.

Designing digital logic circuits [edit]

All digital systems are composed of two elementary functions: memory elements for storing information, and combinational circuits that transform that information. State machines, like counters, are a combination of memory elements and combinational logic circuits. Since memory elements are standard logic circuits they are selected out of a limited set of alternative circuits; so designing digital functions comes down to designing the combinational gate circuits and interconnecting them.

In general the instantiation of logic circuits from high-level abstraction is referred to as Logic Synthesis, which can be carried out by hand, but usually some formal method by computer is applied. In this article the design methods for combinational logic circuits are briefly summarized.

The starting point for the design of a digital logic circuit is its desired functionality, having derived from the analysis of the system as a whole, the logic circuit is to make part of. The description can be stated in some algorithmic form or by logic equations, but may be summarized in the form of a table as well. The below example shows a part of such a table for a 7-segment driver that translates the binary code for the values of a decimal digit into the signals that cause the respective segments of the display to light up.

```
 Digit  Code        Segments
                  A B C D E F G
   0    0000      1 1 1 1 1 1 0          -A-
   1    0001      0 1 1 0 0 0 0         |   |
   2    0010      1 1 0 1 1 0 1         F   B
   3    0011      1 1 1 1 0 0 1         |   |
   4    0100      0 1 1 0 0 1 1          -G-
   5    0101      1 0 1 1 0 1 1         |   |
   6    0110      1 0 1 1 1 1 1         E   C
   7    0111      1 1 1 0 0 0 0         |   |
   8    1000      1 1 1 1 1 1 1          -D-
   9    1001      1 1 1 1 0 1 1
```

The implementation process starts with a **logic minimization** phase, to be described below, in order to simplify the function table by combining the separate terms into larger ones containing fewer variables.

Next, the minimized result may be split up in smaller parts by a factorization procedure and is eventually mapped onto the available basic logic cells of the target technology. This operation is commonly referred to as Logic Optimization.[3]

Classical minimization methods [edit]

Minimizing Boolean functions by hand using the classical Karnaugh maps is a laborious, tedious and error prone process. It isn't suited for more than 6 input variables and practical only for up to 4 variables, while product term sharing for multiple output functions is even harder to carry out.[4] Moreover, this method doesn't lend itself to be automated in the form of a computer program. However, since modern logic functions are generally not constrained to such a small number of variables, while the cost as well as the risk of making errors is prohibitive for manual implementation of logic functions, the use of computers became indispensable.

The first alternative method to become popular was the tabular method developed by Quine and McCluskey. Starting with the truth table for a set of logic functions, by combining the minterms for which the functions are active—the ON-cover—or for which the function value is irrelevant—the Don't-Care-cover or DC-cover—a set of prime implicants is composed. Finally a systematic procedure is followed to find the smallest set of prime implicants the output functions can be realised with.[5][6]

Although this Quine–McCluskey algorithm is very well suited to be implemented in a computer program, the result is still far from efficient in terms of processing time and memory usage. Adding a variable to the function will roughly double both of them, because the truth table length increases exponentially with the number of variables. A similar problem occurs when increasing the number of output functions of a combinational function block. As a result the Quine–McCluskey method is practical only for functions with a limited number of input variables and output functions.

Espresso algorithm [edit]

A radically different approach to this issue is followed in the ESPRESSO algorithm, developed by Brayton e.a. at the University of California, Berkeley.[7] Rather than expanding a logic function into minterms, the program manipulates "cubes", representing the product terms in the ON-, DC- and OFF-covers iteratively. Although the minimization result is not guaranteed to be the global minimum, in practice this is very closely approximated, while the solution is always free from redundancy. Compared to the other methods, this one is essentially more efficient, reducing memory usage and computation time by several orders of magnitude. Its name reflects the way of instantly making a cup of fresh coffee. There is hardly any restriction to the number of variables, output functions and product terms of a combinational function block. In general, e.g. tens of variables with tens of output functions are readily dealt with.

The input for ESPRESSO is a function table of the desired functionality; the result is a minimized table, describing either the ON-cover or the OFF-cover of the function, depending on the selected options. By default the product terms will be shared as much as possible by the several output functions, but the program can be instructed to handle each of the output functions separately. This allows for efficient implementation in two-level logic arrays such as a PLA (Programmable Logic Array) or a PAL (Programmable Array Logic).

The ESPRESSO algorithm proved so successful that it has been incorporated as a standard logic function minimization step into virtually any contemporary logic synthesis tool. For implementing a function in multi-level logic, the minimization result is optimized by factorization and mapped onto the available basic logic cells in the target technology, whether this concerns an FPGA (Field Programmable Gate Array) or an ASIC (Application Specific Integrated Circuit).

Software [edit]

Minilog [edit]

Minilog is a logic minimization program exploiting this ESPRESSO algorithm. It is able to generate a two-level gate implementation for a combinational function block with up to 40 inputs and outputs or a synchronous state machine with up to 256 states. It is part of the **Publicad** educational design package, that can be downloaded from the website Publicad - free Publicad toolkit including Minilog logic minimization program.

Logic Friday [edit]

Logic Friday is a free Windows program that provides a graphical interface to ESPRESSO, as well as to misII, another module in the Berkeley Octtools package. With Logic Friday users can enter a logic function as a truth table, equation, or gate diagram, minimize the function, and then view the results in both of the other two representations. Logic Friday is available at http://www.sontrak.com.

Espresso sources [edit]

The source of the original Espresso program is available from the website of the University of California, Berkeley, at Pubs/Downloads/Espresso. A version of Espresso that has been updated for modern POSIX systems, espresso-ab-1.0.tar.gz POSIX Espresso, is available at [1]

References [edit]

1. ^ Hayes, J.P. (1993), *Digital Logic Design*, Addison Wesley, ISBN 0-201-15461-7
2. ^ Rudell, Richard L. (1986-06-05), "Multiple-Valued Logic Minimization for PLA Synthesis" (PDF), *Memorandum No. UCB/ERL M86-65* (Berkeley)
3. ^ De Micheli, Giovanni (1994), *Synthesis and Optimization of Digital Circuits*, McGraw-Hill Science Engineering, ISBN 0-07-016333-2
4. ^ Lewin, Douglas (1985), *Design of Logic Systems*, Van Nostrand (UK), ISBN 0-442-30606-7
5. ^ Katz, Randy H.; Borriello, Gaetano (1994), *Contemporary Logic Design*, The Benjamin/Cummings Publishing Company, ISBN 0-8053-2703-7
6. ^ Lala, Parag K. (1996), *Practical Digital Logic Design and Testing*, Prentice Hall, ISBN 0-02-367171-8
7. ^ Brayton, Robert King; Hachtel, Gary D.; McMullen, Curtis T.; Sangiovanni-Vincentelli, Alberto L.. (1984), *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, ISBN 0-89838-164-9

Categories: Electronic design automation software | Electronics optimization