Shurik
Soft

High-quality software
For you

# Karnaugh Minimizer

## Using Karnaugh Minimizer Pro

Congratulations on choosing Karnaugh Minimizer Professional - the world's #1 boolean minimization software.

This advanced minimization tool quickly generate a high-quality solution. You can convert boolean function to **VHDL** or **Verilog** code or create schematic with NAND gates. Also, Karnaugh Minimizer has a report generation tool that you can use to create reports of minimization process

This guide contains links to third-party Web sites that are not under the control of ShurikSoft, and ShurikSoft is not responsible for the content on any linked site. If you access a third-party Web site mentioned in this guide, then you do so at your own risk. ShurikSoft provides these links only as a convenience, and the inclusion of the link does not imply that ShurikSoft endorses or accepts any responsibility for the content on those third-party sites.

First Edition: February 2004

ShurikSoft.
Yazikova St.
Donetsk, Ukraine 83101

# CONTENTS

# CHAPTER 1
Introduction

## About Karnaugh Minimizer Professional

Boolean Algebra assistant program is an interactive program extremely easy to use. A must-have tool for the freshmen electrical engineering student.

The program is intended for the developers of small digital devices and radioamateurs, also for those who is familiar with Boolean algebra, mostly for electrical engineering students.

Karnaugh Minimizer program is called to facilitate minimization of Boolean function by a method of Karnaugh maps. The program has the simple and convenient interface, evident image of received results of minimization.The program draws the so-called largest circle and displays the prime implicant solution.

With Karnaugh Minimizer, a proven boolean minimization tool, you can effeciently design small circuits and logical schemas, Karnaugh Minimizer handles the widest variety of boolean function representations, letting you to choose convenient way of presenting your boolean functions.

With Karnaugh Minimizer Professional, you can now do all of the following:

- Receive output in either SOP(DNF) or POS(CNF) format
- Find and eliminate redundant terms
- Handle don't care conditions
- Formula to schematic tool convert boolean formula to NAND or AND/OR schematic. supported European and USA gate systems
- Use Quine Mc Cluskey method minimization tool to handle 4-23 variables
- Convert boolean formula to **VHDL** or **Verilog** code
- Allows you to click on a term in a given expression and view it on the map. The program will indicate the corresponding term with a fill cells
- Draw 2 - 8 variable Karnaugh Map
- Simplifies expressions that you type in
- Fill map with several ways Truth table, Sets, Boolean formula
- Export Map to HTML table to easyer creation of synthesis reports
- Automated creation of reports after analyze

# What’s new in Karnaugh Minimizer Professional version

- Absolutely new user interface + WinXP themes support
- Size of installation package was reduced in about two times
- More user friendly than ever
- Increased enhancement potential
- Converter tool allow you to convert Boolean function between different representations of it (K-map, Sets, Truth table)

## Sources of information

Useful links to information about Karnaugh minimization method:

- Theory of Karnaugh Maps
  http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karnaugh.html
- Book about Boolean algebra and Karnaugh Minimization
  http://www.maxmon.com/kmaps1.htm
- Minimization of Boolean expressions using Karnaugh maps
  http://www.cs.usm.maine.edu/~welty/karnaugh.htm
- Logic Minimization
  http://www.cs.byu.edu/courses/cs143/reading/karnaugh.html

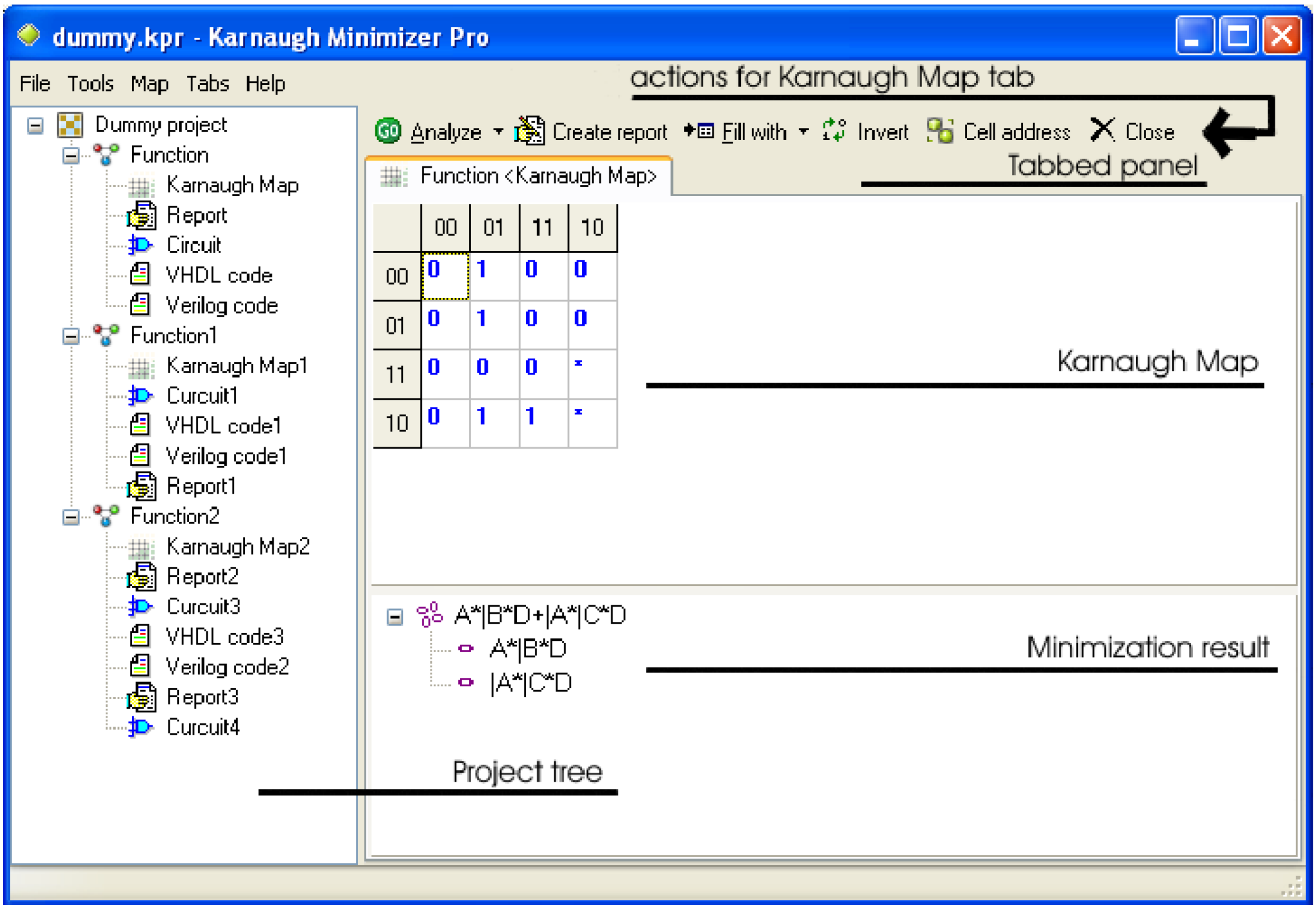
# CHAPTER 2
User Interface description

## Using project tree

User interface of Karnaugh Minimizer Pro consist of two main parts:

- Project tree - allow you to manage content of your project, you can add any number of boolean functions to project tree using Ctrl-Alt-F shortcut or through main menu “File | Add function”. Each function node of project tree can contain unlimited number of sub nodes with fixed types like Report, Circuit, VHDL code, Verilog code and only one node of type Karnaugh Map.

- Tabbed panel - allow you to browse through project tree nodes, Use F3 shortcut to add content of selected node to tabbed panel. Each tab has appropriate set of actions Which can help you to operate with selected node content, for example Karnaugh Map tab has Analyze, Create report, Fill with, Invert, Cell address, Close actions.

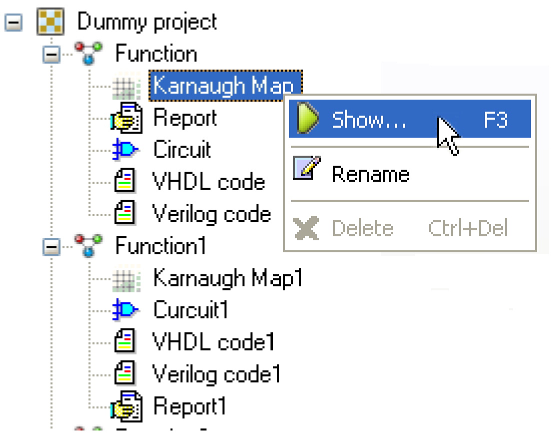


|  | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | * |
| 10 | 0 | 1 | 1 | * |

A*|B*D+|A*|C*D

A*|B*D

|A*|C*D



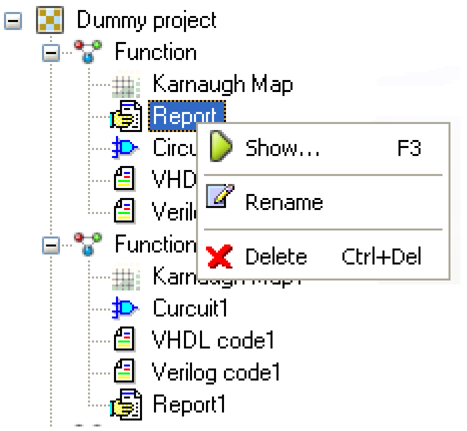
Right click on project tree brings up popup menu.

You can  Show, Rename or Delete tree nodes.

**Show** - adds content of selected node to tabbed panel if it not already there, otherwise it will set corresponding tab as active.

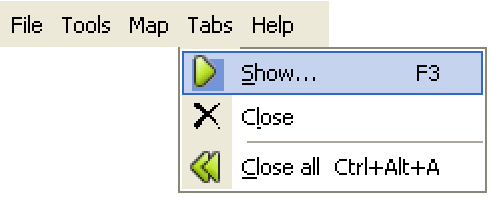
Also you can do this by double clicking on tree node.

**Rename** - When you add nodes to tree, they will be named with default names,  you can rename it for better understanding and quick search purposes.



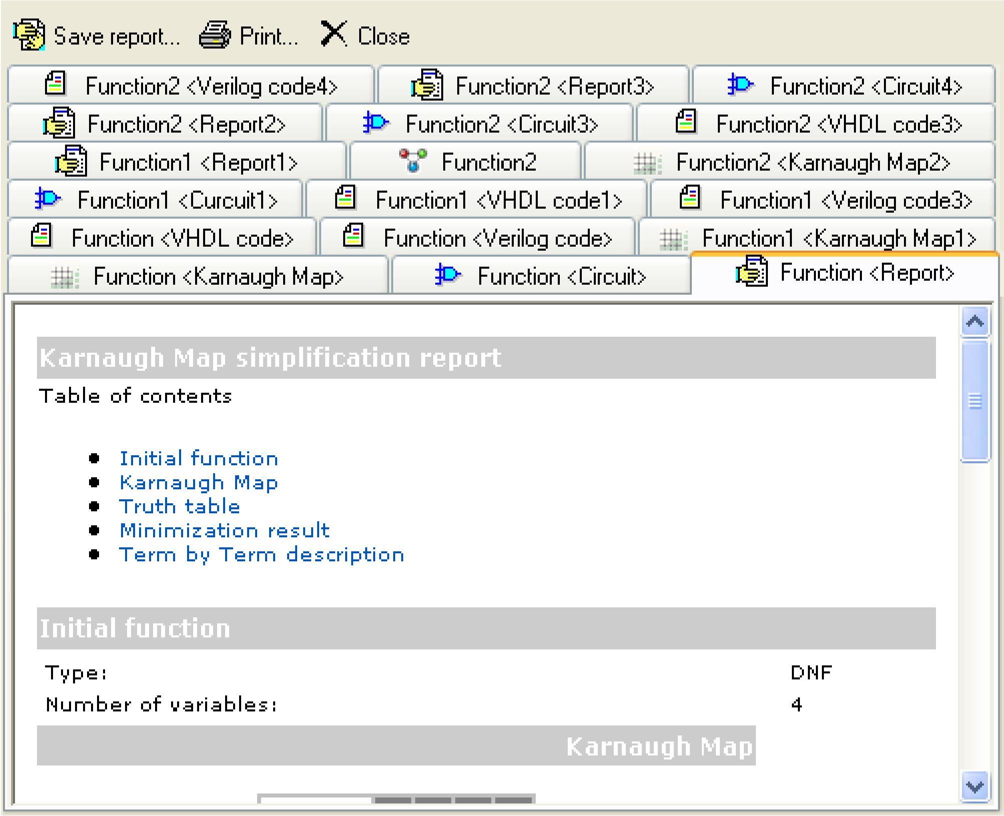
**Delete** - allows  you  to remove unnecessary nodes from tree. Corresponding tab will be closed when you delete node with opened tab.

# Using tabbed panel



Tabbed panel - can simultaneously  display as many tabs as you wish.

Use project tree popup menu to add content of selected node to tabbed panel.  Each tab has "**Close**"  action which  allow you to close active  tab. Also you can close all tabs if number of opened tabs too many.



Karnaugh Map simplification report

Table of contents

- Initial function
- Karnaugh Map
- Truth table
- Minimization result
- Term by Term description

Initial function

Type: DNF

Number of variables: 4

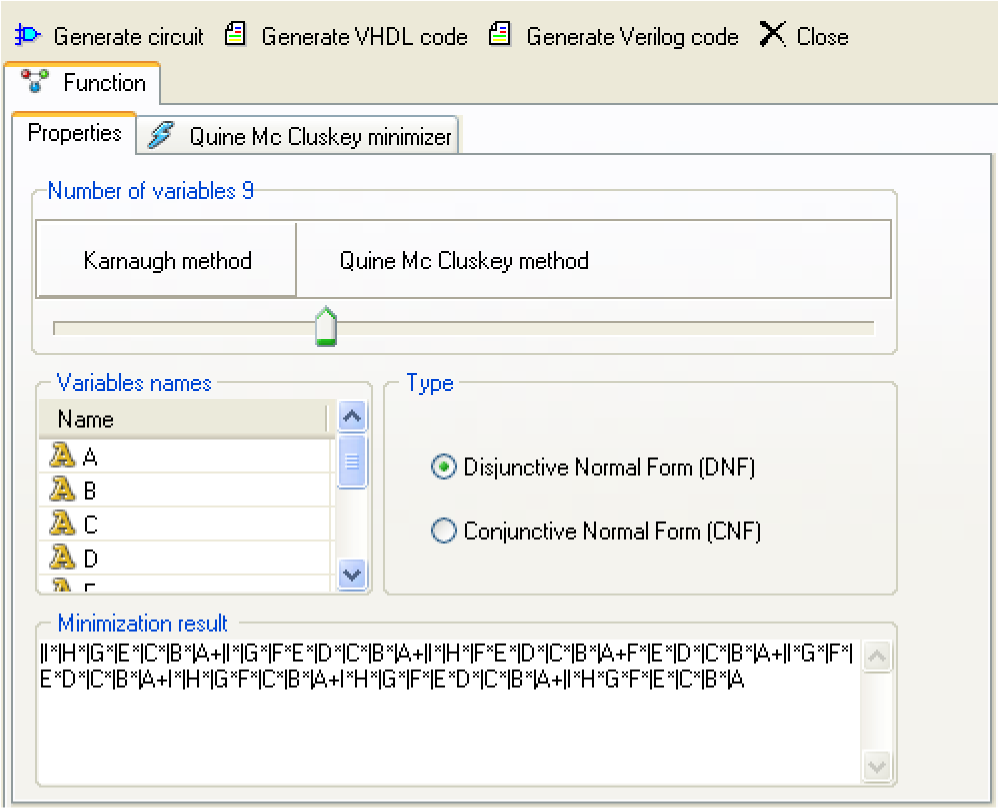
Karnaugh Map

# CHAPTER 3
Tabs description

## Function tab

Function tab represent properties of boolean function such as: variable count, variable names, type of function.  This tab more important then others, because it define which minimization algorithm will be used for particular boolean function, it depends on how many Input variables used in function.

- Karnaugh map method is used when number of input variables between 2 and 8
- Quine Mc Cluskey method is used when number of input variables gain 9 and below 23



Number of variables 9

| Name |
|---|
| A |
| B |
| C |
| D |
| E |

● Disjunctive Normal Form (DNF)

○ Conjunctive Normal Form (CNF)

Minimization result

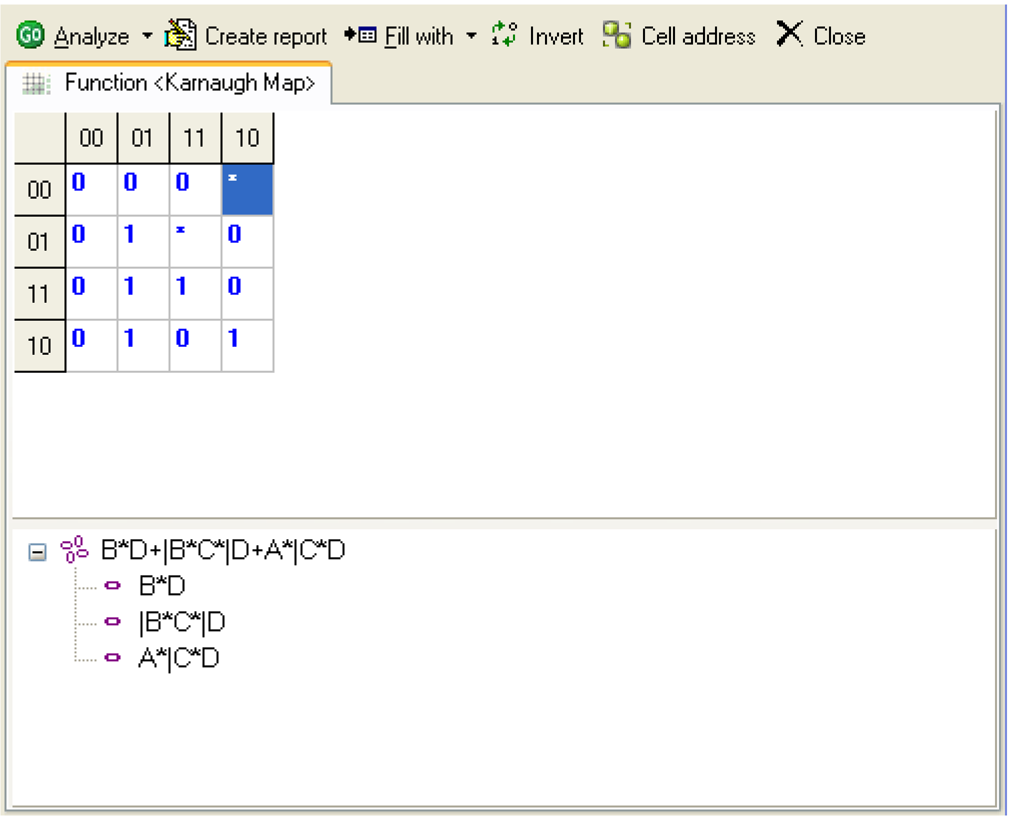
```
|I*|H*|G*|E*|C*|B*|A+|I*|G*|F*E*|D*|C*|B*|A+|I*|H*|F*E*|D*|C*|B*|A+F*|E*|D*|C*|B*|A+|I*G*|F*|
E*D*|C*|B*|A+I*|H*|G*F*|C*|B*|A+I*H*|G*|F*|E*D*|C*|B*|A+|I*H*G*F*|E*|C*|B*|A
```

After minimization process, function tab will display “Minimization result” section, then you will be able to use actions for generating Circuit, VHDL or Verilog codes based on result of minimization.

# Karnaugh Map tab

Karnaugh Map tab is used for minimization of boolean function by graphical method Karnaugh maps. This tab will help you to visualize minimization process and improve your understanding of this method of analyze.

Karnaugh map can be filled by several ways: with Truth table, Sets, Logic formula or with manual input of function values directly to map. Direct input can be done through keyboard or with mouse clicks.



| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | * |
| 01 | 0 | 1 | * | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 0 | 1 |

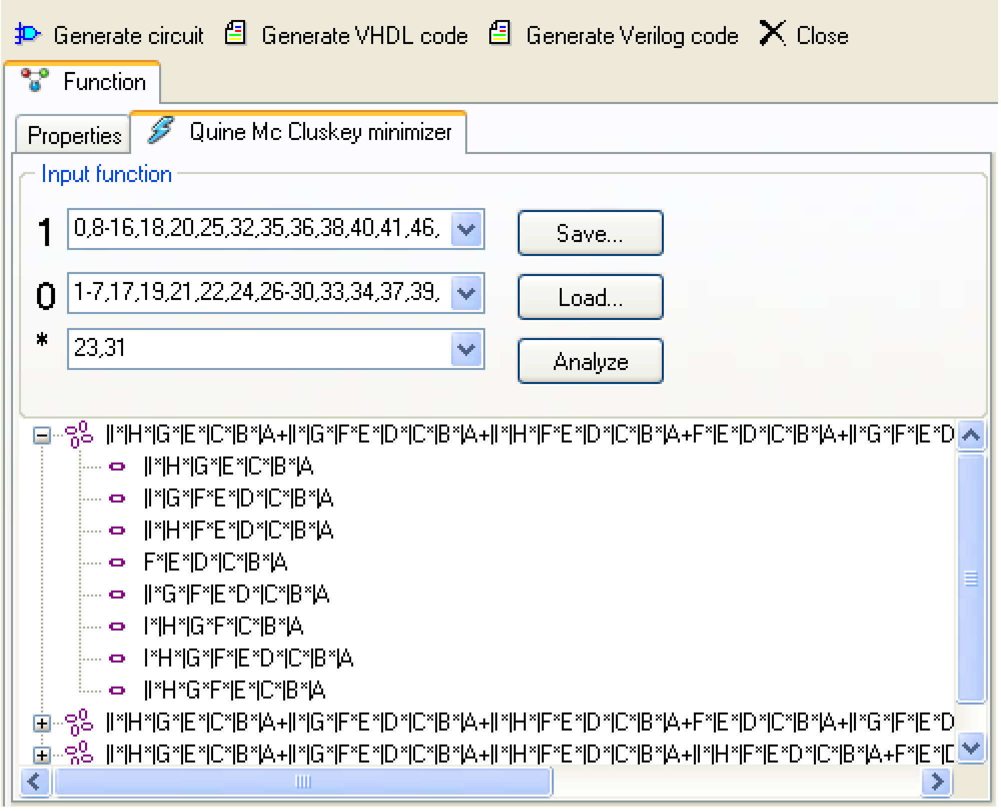
B*D+|B*C*|D+A*|C*D
- B*D
- |B*C*|D
- A*|C*D

After analyze of Karnaugh Map you will be able to use “Create report” action, which quickly and easily make detailed report of minimization process. Also you can open appropriate function tab and generate Circuit, VHDL or Verilog code.

# Quine Mc Cluskey tab

This tab appears only when you move variables count slider to values more then 8. With Quine Mc Cluskey minimization algorithm you can handle boolean functions with number of Input variables from 9 to 23.

Boolean function can be defined with Sets representation only, because it is compactest way to store function.  You will be able to Save, Load and Analyze boolean function. This tab will be accessible to registered users only.



1 0,8-16,18,20,25,32,35,36,38,40,41,46,

0 1-7,17,19,21,22,24,26-30,33,34,37,39,

* 23,31

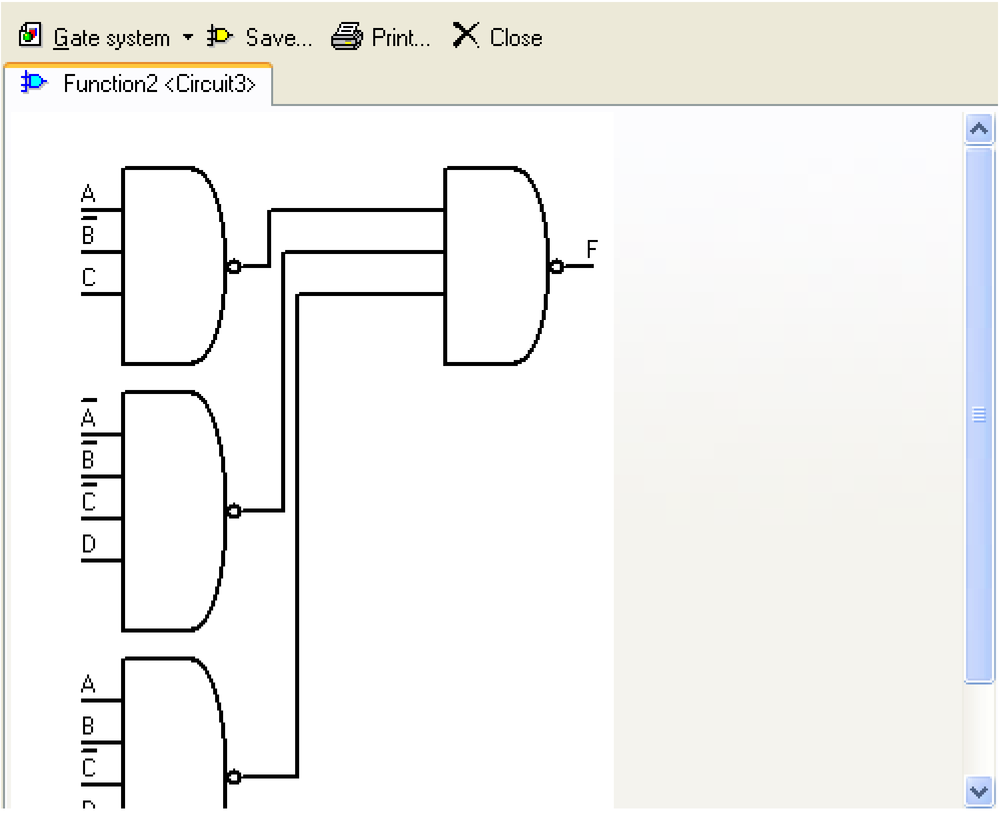
```
⊟ |I*|H*|G*|E*|C*|B*|A+|I*|G*|F*E*|D*|C*|B*|A+|I*|H*|F*E*|D*|C*|B*|A+F*|E*|D*|C*|B*|A+|I*G*|F*E*D
    |I*|H*|G*|E*|C*|B*|A
    |I*|G*|F*E*|D*|C*|B*|A
    |I*|H*|F*E*|D*|C*|B*|A
    F*|E*|D*|C*|B*|A
    |I*G*|F*|E*D*|C*|B*|A
    I*|H*|G*F*|C*|B*|A
    I*H*|G*|F*|E*D*|C*|B*|A
    |I*H*G*F*|E*|C*|B*|A
⊞ |I*|H*|G*|E*|C*|B*|A+|I*|G*|F*E*|D*|C*|B*|A+|I*|H*|F*E*|D*|C*|B*|A+F*|E*|D*|C*|B*|A+|I*G*|F*|E*D
⊞ |I*|H*|G*|E*|C*|B*|A+|I*|G*|F*E*|D*|C*|B*|A+|I*|H*|F*E*|D*|C*|B*|A+|I*|H*|F*|E*D*|C*|B*|A+F*|E*|D
```

After minimization process, function tab will display “Minimization result” section, then you will be able to use actions for generating Circuit, VHDL or Verilog codes based on result of minimization.

# Circuit tab

Circuit tab is designed for schematic drawing  purposes only, it is possible to draw circuit with NAND gates and save it as JPG or BMP image. Also you can do switching  between USA and European systems of gates .

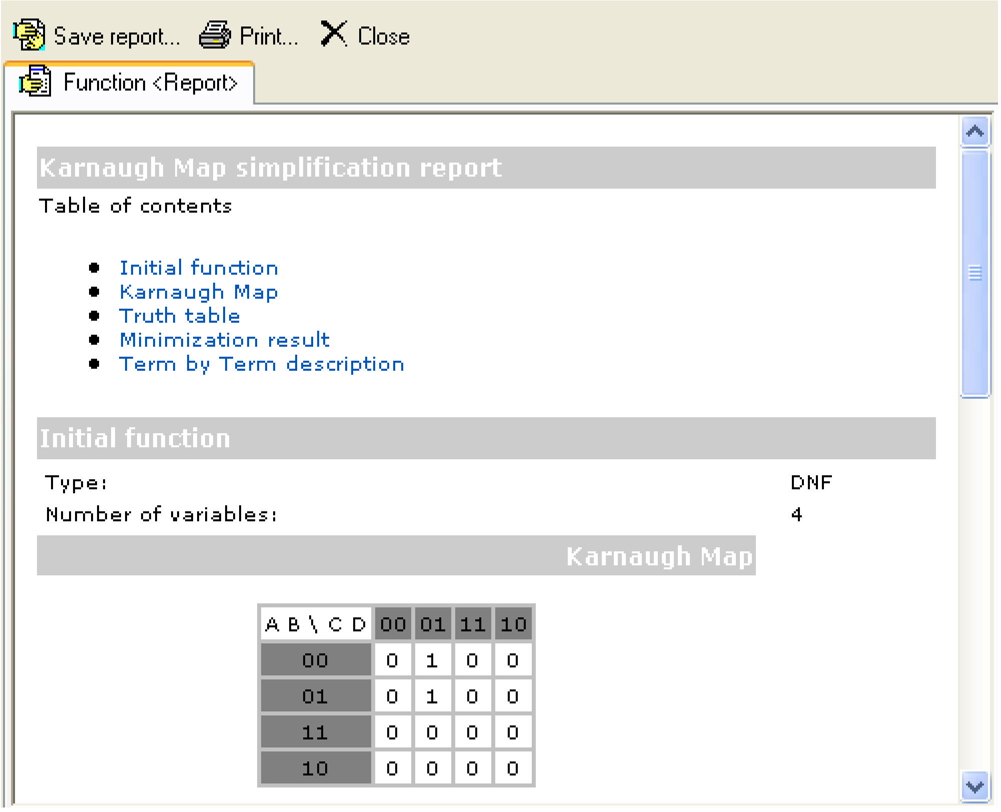
You can generate circuit after minimization process from Function tab



# Report tab

Report tab allow you to view, print and save reports of minimization process produced by Karnaugh Map tab

Report it is HTML document which you can easily edit with Word processor and add Circuit picture or VHDL or Verilog code.



Karnaugh Map simplification report

Table of contents

- Initial function
- Karnaugh Map
- Truth table
- Minimization result
- Term by Term description

Initial function

| | |
|---|---|
| Type: | DNF |
| Number of variables: | 4 |

Karnaugh Map

| A B \ C D | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

# VHDL code tab

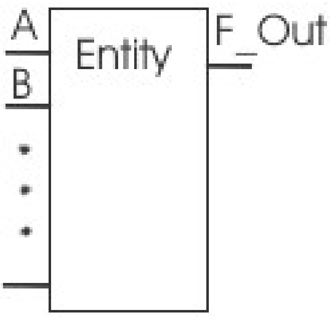
VHDL code tab is used to display generated by Function tab code.

VHDL is (Very high speed integrated circuit Hardware Description Language) code.

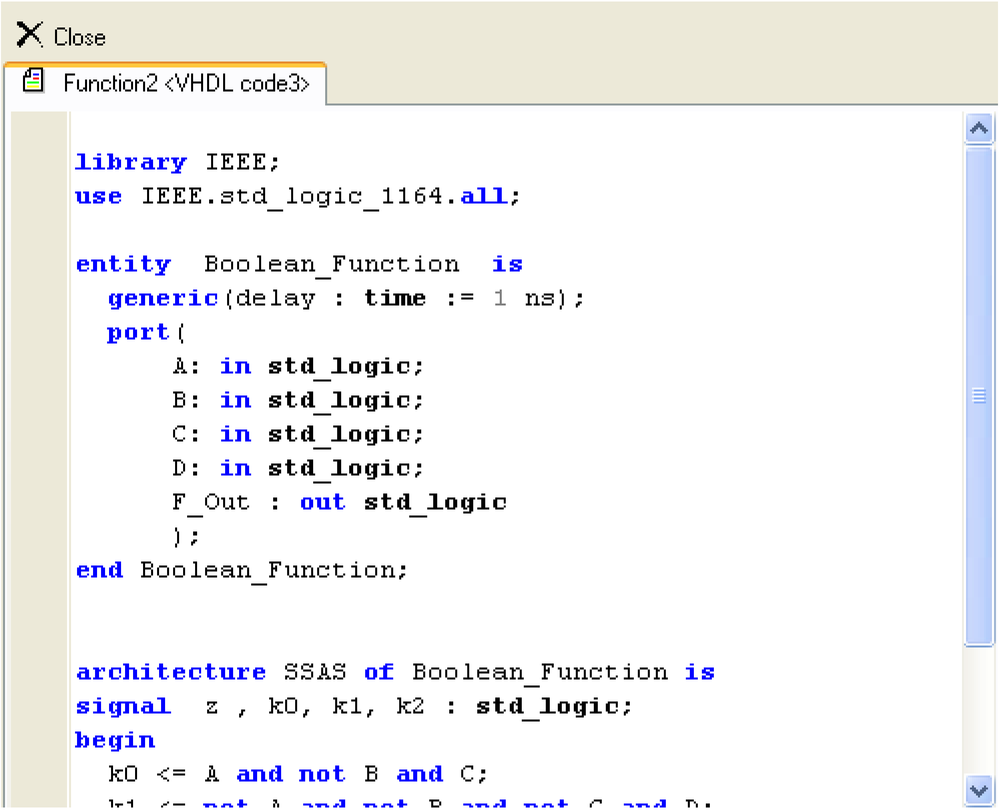
This codes created by Karnaugh Minimizer Pro can be used with any EDA Design/Verification tool include Active-HDL by Aldec inc.

VHDL code is used for verification and testing results of synthesis.

Created VHDL code has following structure:



Architecture for this entity is created with Simple Signal Assignment Statement (SSAS).



```
library IEEE;
use IEEE.std_logic_1164.all;

entity  Boolean_Function  is
  generic(delay : time := 1 ns);
  port(
      A: in std_logic;
      B: in std_logic;
      C: in std_logic;
      D: in std_logic;
      F_Out : out std_logic
      );
end Boolean_Function;


architecture SSAS of Boolean_Function is
signal  z , k0, k1, k2 : std_logic;
begin
  k0 <= A and not B and C;
  k1 <= not A and not B and not C and D;
```

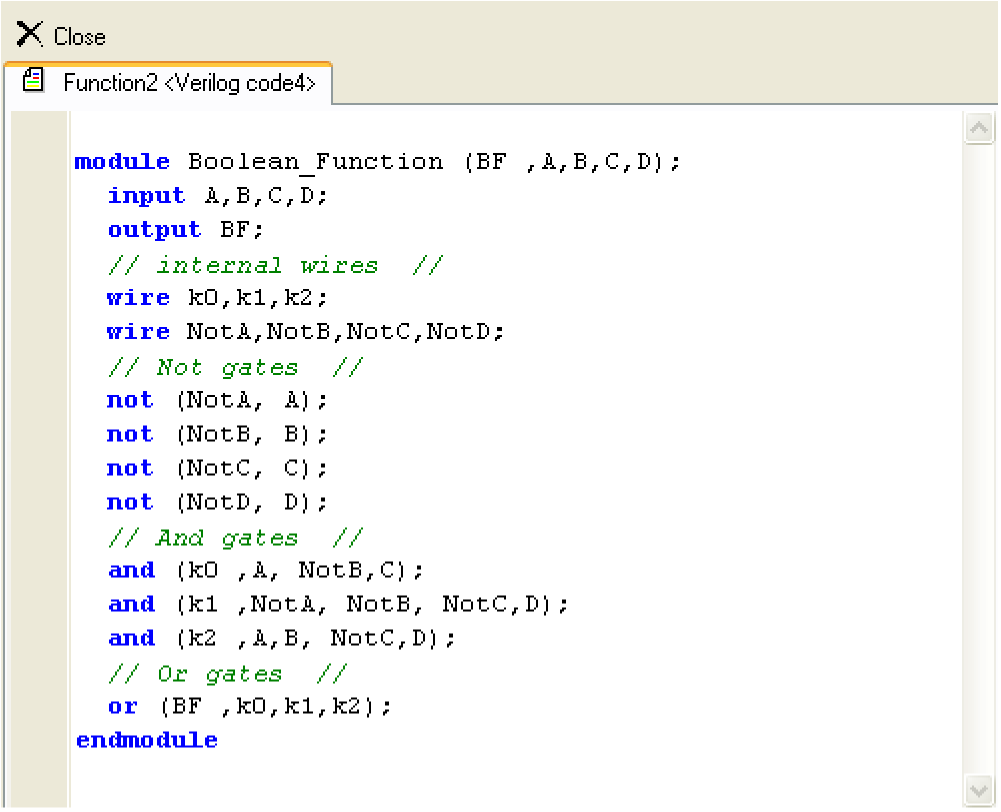
You can paste generated code directly to HDL design IDE without major changes in it structure.

# Verilog code  tab

Verilog  code tab is used to display generated by  Function tab  code.

Verilog (Verilog HDL - developed in 1984-1985 by Philip Moorby) code.
This codes created by Karnaugh Minimizer can be used with any EDA Design/Verification tool include Active-HDL by Aldec inc

Verilog code is used for verification and testing results of synthesis.



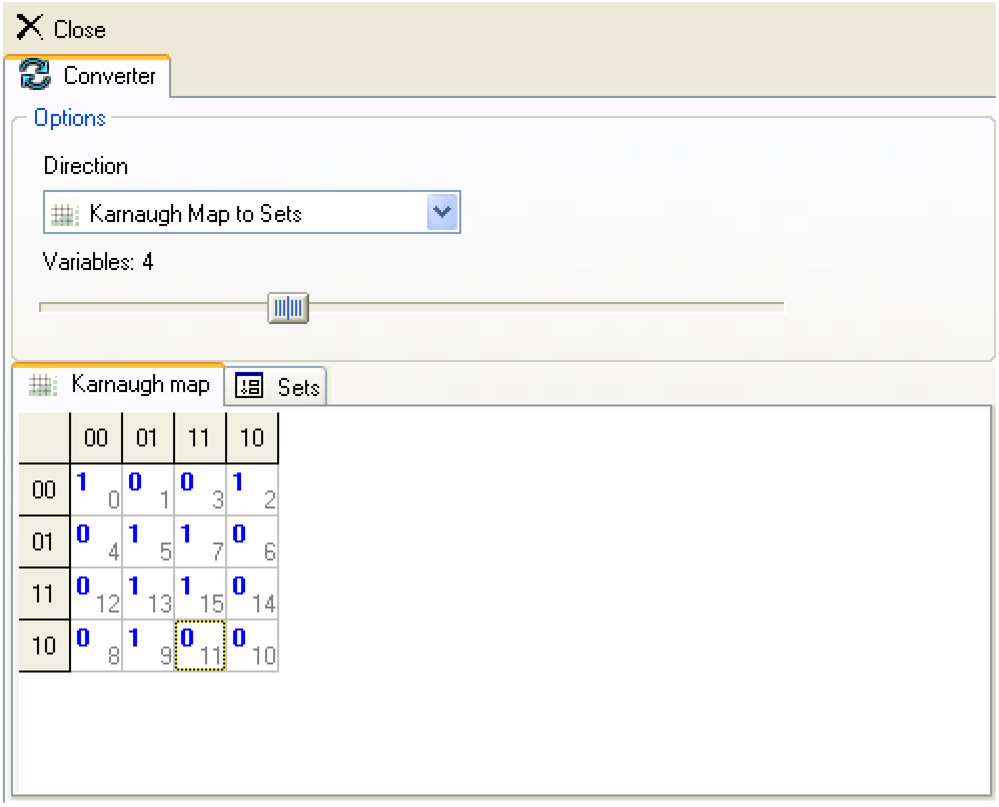
```
module Boolean_Function (BF ,A,B,C,D);
  input A,B,C,D;
  output BF;
  // internal wires  //
  wire k0,k1,k2;
  wire NotA,NotB,NotC,NotD;
  // Not gates  //
  not (NotA, A);
  not (NotB, B);
  not (NotC, C);
  not (NotD, D);
  // And gates  //
  and (k0 ,A, NotB,C);
  and (k1 ,NotA, NotB, NotC,D);
  and (k2 ,A,B, NotC,D);
  // Or gates  //
  or (BF ,k0,k1,k2);
endmodule
```

You can paste generated code directly to HDL design IDE without major changes in it structure.

# Converter tab

Converter tab is used for conversion purposes only not for minimization.

Converter tool allow you to convert Boolean function between different representations of it (Karnaugh map, Sets, Truth table) and it is limited to 8 variables because
Karnaugh map method can handle maximum 8 variables.

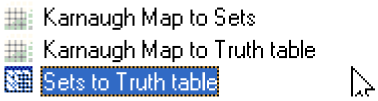


Direction: Karnaugh Map to Sets

Variables: 4

|  | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1<br>0 | 0<br>1 | 0<br>3 | 1<br>2 |
| 01 | 0<br>4 | 1<br>5 | 1<br>7 | 0<br>6 |
| 11 | 0<br>12 | 1<br>13 | 1<br>15 | 0<br>14 |
| 10 | 0<br>8 | 1<br>9 | 0<br>11 | 0<br>10 |

First of all select direction of conversion, then adjust number of variables.
Next fill one of two parts and switch to another, you can do two way conversions with this tool.



# CHAPTER 4
Theory

## SOP and POS notations

Boolean function can be represented in one of this two forms:

- **SOP** (**S**um-**o**f-**P**roducts) or **DNF** (**D**isjunctive **N**ormal **F**orm) focuses on the 1's in the output.

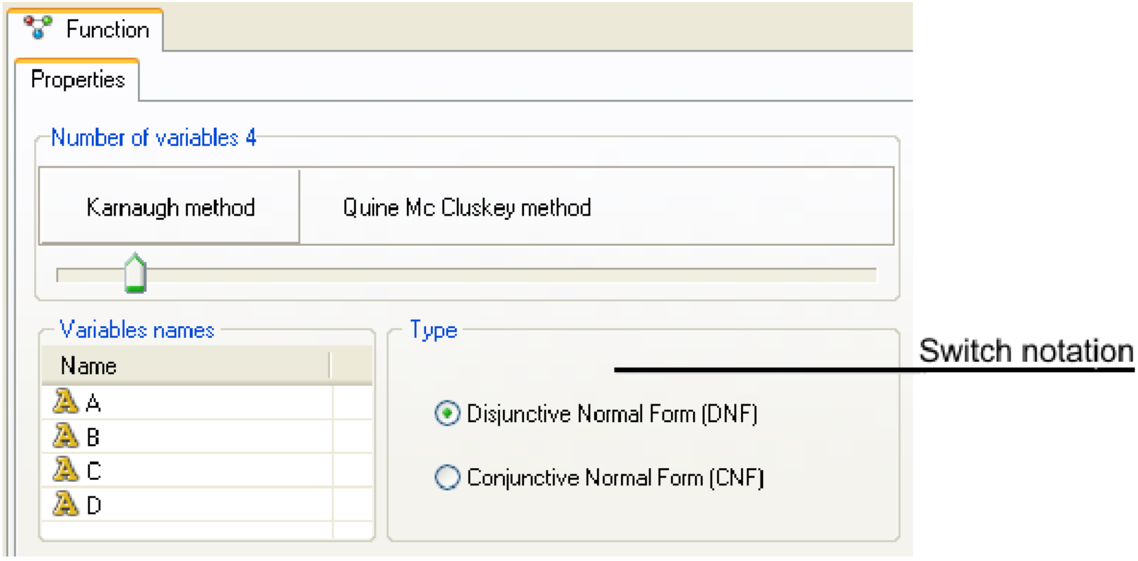
- **POS** (**P**roduct-**o**f-**S**ums) or **CNF** (**C**onjunctive **N**ormal **F**orm) focuses on the 0's in the output.

In **SOP** The direct meaning variable corresponds logic "1" and inverse logic "0". In this form the terms united with "+"(Disjunction) and variable in a term united with “*"(Conjunction).

**Example:** A*B*C + |A*B*|C + B*C

In **POS** The direct meaning variable corresponds logic "0" and inverse logic "1". In this form terms united with "*"(Conjunction) and variable in a term united with "+"(Disjunction).

**Example:** (A+B+C)*(|A+B+|C)*(B+C)



Number of variables 4

| Name |
|---|
| A |
| B |
| C |
| D |

Type

● Disjunctive Normal Form (DNF)

○ Conjunctive Normal Form (CNF)

Switch notation

You can manage this notations on boolean function properties page.

# Decimal notation

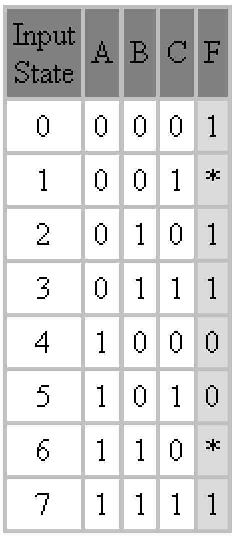
Decimal notation is a short hand for specifying boolean functions. An example of decimal notation is:

SOP/DNF:
F = E(0,2,3,7), D(6,1).

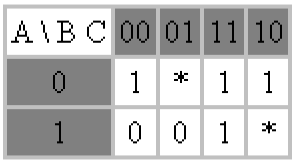
POS/CNF:
F = A(4,5), D(6,1).

**Note**: D - Don't care

A function is specified either in SOP or POS. Sum(E) indicates SOP and Pi(A) indicates POS, D specifies don't care conditions (all don't care conditions are specified with sum). The best way to understand decimal notation is to show how to get it. We'll use the following system:

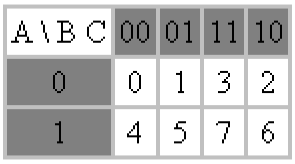


| Input State | A | B | C | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | * |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | * |
| 7 | 1 | 1 | 1 | 1 |



| A \ B C | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | * | 1 | 1 |
| 1 | 0 | 0 | 1 | * |

*K-map for function F*



| A \ B C | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 3 | 2 |
| 1 | 4 | 5 | 7 | 6 |

*K-map for input states(Sets)*

The inputs are A, B and C. The output is F. The key to decimal notation is a well defined arrangement of the inputs.

Each input is mapped to a bit in a binary number. In this case, C is bit 0, B is bit 1 and A is bit 2. The values of a particular input state are then represented by a binary number.

A=0, B=1, C=1 Input State = 011 (binary), 3 (decimal)

# What is RegKey ?

**RegKey** - Registration Key is used to personalize copy of Karnaugh Minimizer similar with Serial Number (S/N) or Product code.

You can find your RegKey in main menu of Karnaugh Minimizer under "Help | About".
RegKey have folowing structure: KMX-YYYYYYYY-YYYYYYYY where
X = S is Karnaugh Minimizer Standard edition.
X = P is Karnaugh Minimizer Professional edition.
YYYYYYYY-YYYYYYYY - personal number for every copy of the program.

The RegKey is stored in Windows registry so if you ReInstall Windows you will lose your personal number and program will generate new one. If you are registered user simply mail new RegKey to support team of Karnaugh Minimizer and you will receive new license file.