Abstract

This paper presents a didactic tool for two-level logic synthesis called KARMA (KARnaugh MAps). The tool helps users that are having their first contact with two-level logic synthesis to acquire a better understanding of associated concepts; such as Boolean functions, truth-tables, equations, Karnaugh maps, cubes, minterms, prime implicants, etc. The tool has a solution mode to run Quine-McCluskey’s algorithm to get minimum equations, and a teaching mode where the user can practice his knowledge through exercises.

1. Introduction

Two level logic synthesis [1] is a very important area for those initiating on digital hardware design. Indeed, freshman students in every electrical engineering, computer engineering or computer science course are introduced to digital hardware design by learning the concepts of Karnaugh [2] Maps and the Quine-McCluskey [3, 4, 5] algorithm. Associated to these methods, are the concepts of Boolean functions, truth tables, Boolean equations, cubes, minterms, implicants, primes, don’t cares, minimum solutions, canonical forms, etc. All the concepts involved in these methods are quite abstract, and it is difficult for a teacher to provide a sufficiently large number of examples so that the students acquire sufficient confidence. The availability of a software tool that supports the student to build this self-confidence can enhance significantly the learning process.

Karma is a software tool to help students in learning Karnaugh maps and the involved concepts. It has many interesting features to accomplish this objective. These features include the visualization of prime implicants in the map, step-by-step solution of the Quine-McCluskey algorithm, and a teaching mode with questions about Karnaugh maps and logic synthesis concepts, where the student can exercise the concepts and get feedback on the correct (or wrong) solution of the proposed problems.

Karma was developed in the Java language as part of the Lagarto (LAyout GenerAtoR TOol) project [6] and is distributed as free software under GPL (General Public License).

2. Solution mode

In this mode the user can interact with a Karnaugh map and a truth-table to assign a Boolean function and obtain its minimized sum-of-products or product-of-sums, both from the on-set and from the off-set. There is also an option to factorize [7,8] the equations. The final cover can be made to minimize literals [1] or series transistors [9]. The logic function can be entered through different methods: truth-table, Karnaugh map, or Boolean equation. Two to eight variables are allowed. The Quine-McCluskey algorithm is applied step-by-step to obtain the minimized function, as shown in Fig. 1.

![Figure 1: Step-by-step run of the Quine-McCluskey algorithm](image)

As shown in Fig. 2, the truth-table and the Karnaugh map are visualized simultaneously in this mode. Both are linked, so when a value of a cell of the Karnaugh map is changed by the user, this value is updated automatically in the truth table, and vice versa. Supported logic values are zero, one and don’t-care. Also there is the possibility of changing the position of the variables of the Karnaugh map, which changes the position of minterms in the map. This feature is very useful to teach the concepts of function equivalence classes and symmetric functions.

3. Teaching mode

This mode was developed for users to review basic concepts of two-level logic synthesis through exercises. The available exercises cover the learning of adjacency of minterms, cubes, prime implicants, function cover and cube/minterm covering table.

The exercises contain a brief description of a proposed problem, and a Karnaugh map to help finding the answer in a visual way. The given solution is evaluated to check...
its correctness. If it is not correct, the correct solution is then displayed. For the exercises, the window is divided into two parts, where the left part shows the description of the exercise and the right part shows the Karnaugh map, as illustrated in Fig. 3. A description of the available exercises is given below.

**Minterms:** Two options are available to cover the concept of minterm adjacency. In one option, the user selects a minterm in the Karnaugh map, and the adjacent minterms are highlighted. In the second option, a minterm is selected and the user is asked to point its adjacent minterms. If a correct minterm is selected, it is displayed in green color, otherwise the minterm is displayed in red color.

**Groups:** To understand the concept of grouping minterms, two exercises are proposed. In the first one, the user is asked to select minterms to form a group. When the selection is finished, a message displays the equation of the formed group (or says that the selected minterms do not form a group). In the second one, the user is given a group equation and is asked to point the minterms that form it in the map. Notice that these exercises are to locate regions in the map, with no functions displayed.

**Cubes:** Two exercises are proposed to review the concept of cubes of a function. In the first one the user must select a cube of the function displayed. Information on the cube type (implicant, prime implicant or not a cube) is given. In the second exercise, some minterms are shown to the student and he has to say if the minterms form a implicant, a prime implicant or if they don’t form a cube.

**Function cover:** In this exercise a list of cubes is displayed and the user must select, for each cube, the correct answer about its state (essential, prime, not prime, not a cube) in the particular function. The students receive hints about the type of function cover, according to the given answers (redundant, irredundant, prime, not prime, not a valid cover).

**Covering table:** This option shows the covering table of a function. The student can find a cover of the function by selecting cubes in the covering table and observing which minterms are covered in the map. When a selected cube is essential prime implicant, a large red sign is displayed in the chart at the same row, otherwise small blue signs are displayed. This exercise trains the student to find the minimum cover of a function. This exercise is illustrated in Fig. 3.

**5. References**


