**Chapter 22**

**Applications of Cube Calculus Machine**

**Marek Perkowski**

Many logic minimization software tools such as ESPRESSO [37], MIS, SIS [38] and EXORCISM-MV-2 [16,32] may benefit from the introduction of the CCM. David W. Foote analyzed how much the CCM can improve the performance of ESPRESSO-II in his thesis [17].

The CCM can be used to perform set operation, like set intersection, set union, set complement, and set relations such as subset as we discussed in Chapter 15. The CCM can be seen as a machine for set-theoretical problems when it is configured to process only one variable with many possible values. The Examples 15.12 to 15.15 are this kind of examples.

The CCM can also speed up the process of solving small size of the satisfiability problem and the tautology problem that are the two most fundamental combinational problems used in many research and application areas. As shown in [53], these two problems can be used to solve many other more complicated problems in CAD, Machine Learning and other fields. The point here is that we speed up the process of solving small size of these two problems by speeding up the very basic operator.

**22.1. Satisfiability Problem**

Given a product of terms, each term being a Boolean sum of literals; the satisfiability problem is to find any product of literals that satisfies all terms or prove that such product does not exist.

**Example 22.1** Find all products of literals that let function f(a,b,c,d) = $(\overbar{a}$+b) ($\overbar{b}$+c)(a+$\overbar{c}$) be 1.

**Solution.** It is very easy to rewrite this problem by multiplying out the expression as follows:

****

The Covering Problem is used in many two level logic minimization algorithms (being various improvements and extensions of the classical Quine-McCluskey algorithm [23]) and it can be reduced to the Petrick Function Minimization Problem, a special case of satisfiability problem. This means that the Covering Problem can be solved by minimizing the Petrick Function. Let us see the following example.

**Example 22.2** The following is a covering table. The problem is to find the smallest set of rows that covers all columns.



**Solution.** First, let us find all sets of rows that cover all columns. The column C1 is covered by rows R1 and R3; the column C2 is covered by rows R1and R3; the column C3 is covered by rows R2 and R4; the column C4 is covered by rows R1 and R2; the column C5 is covered by row R4; the column C6 is covered by rows R1 and R2; Therefore, the problem can be solved by solving the following Petrick function (PF):

****

The last sum shows that there are five sets of rows that cover all columns: {R1,R4}, {R1,R2,R4}, {R1,R3,R4}, {R2,R3,R4} and {R1,R2,R3,R4}. Now use the absorption law A+AB=A to simplify the decision function and obtain PF=R1R4+R2R3R4, which means {R1, R4} is the smallest set of rows that covers all columns.

The above example shows how to reduce the Covering Problem to the Petrick function problem. The CCM can be used to multiply out the function, then the function can be simplified by the software or Sorter/Absorber designed by Dr. Perkowski and his students [39]. It has to be kept in mind by the reader that CCM was designed because real life covering problems have matrices with hundreds of thousands of rows and columns. Because, however the number of variables in the CCM is limited, a large problem has to be first decomposed to many smaller problems that fit the CCM word length and can be solved by it sequentially.

**22.2 Tautology Problem**

The tautology problem is the verification a logic function to see if it is always true or not.

**Example 22.3** Is the function f(a,b,c,d)=a+b+c+d a tautology (always be 1)?

**Solution**. If the function f(a,b,c,d) = 1, this implies that 1 # f(a,b,c,d)=0, and the function f(a,b,c,d) is a tautology. If 1 # f(a,b,c,d)  0, then the function f(a,b,c,d) is not a tautology.

Because 1 # (a+b+c+d)= $\overbar{a }\overbar{b}\overbar{c}\overbar{d}$ 0, then the function f(a,b,c,d) is not a tautology. The CCM assembly program used to solve this problem is very similar to the Example 19.9.

**22.3. Future Work on CCM**

The concept of Cube Calculus, Cube Calculus Machine and a complete design of the Cube Calculus Machine have been presented in previous chapters. This is the first complete design of the CCM that has been done so far.

The design of the ILU presented above has been accomplished by many students over the years [15, 17, 18]. In the presented design some important changes have beed done (like EMPTY block, Pre-relation/pre-operation logic block).

The author’s contributions to the CCM project are outlined below in the manner in which the project pieces were completed.

* Design a complete CCM which is presented in this thesis. For the most part of ILU comes from past texts [15,17,18] all other parts have been designed by the author. The prerelation\_preoperation that were missing in past texts [15, 17, 18] have been designed by the author.
* Modeled the CCM in VHDL code and simulated it using QuickHDL tool from Mentor Graphics. This is the first simulation of the entire CCM.
* Created CCM assembly language for the CCM and realized it in the test bench. This made the VHDL model of the CCM a research tool. Future students can explore cube operations by using this model with having the knowledge of the encoding scheme of the CCM instructions, but they still need to think in terms of registers and individual instructions of the CCM. There are several complete examples given in this thesis, and they can be used as a tutorial.
* Improve and unify the descriptions of cube calculus from previous texts.
* Wrote an introduction to the DEC PeRLe-1 board.
* Derived the formula to calculate cofactor operation by using cube calculus. The result is shown in Equation 15.11.
* Mapped this design onto the PeRLe-1 board. The entire design has been captured using Xilinx Foundation software, and has been implemented in 17 XC3090A FPGA chips using Xilinx M1 software.

Suggestions are made here for future work that will build a ready-to-use Cube Calculus Machine.

* Develop the CCM runtime library (see section 19.1). DEC C++ compiler is required.
* Build several complete demo applications of practical use such as tautology, satisfiability, or set covering.
* Run the demos on benchmark functions.