Improvements in Exact Minimal Waveform Coverings of Periodic Binary Signals

Bernd Steinbach[1] and Christian Posthoff[2]

[1] Freiberg University of Mining and Technology, Institute of Computer Science, D-09596 Freiberg, Germany
[2] The University of The West Indies, St. Augustine Campus, Trinidad & Tobago

Abstract. The calculation of an exact minimal cover of a Boolean function is an NP-complete problem. In this paper we introduce the definition of this problem and its basic solution. By using a slightly modified algorithm, we get a speed-up factor of more than 10^4. The main contributions of this paper are the proof of an alternative approach mentioned in [2], and a remarkable improvement of this algorithm. In both cases operations of the XBOOLE library are used. Using the newly suggested algorithm, the time required for the calculation could be reduced by a factor of more than $8 * 10^8$ in comparison with the improved algorithm.

1 Introduction

Periodic binary signals of the length L can be created by a finite state machine. The sequential part of this machine determines both the time interval of the elementary signal step and the length of the period. The combinational part of the finite state machine determines the waveform of the signal by means of the required sequence of Boolean values.

To save circuit space and to reduce power consumption, the combinational generator circuit is typically synthesized using prime conjunctions. This paper deals with the selection of all exact minimal sets of such prime conjunctions.

We present in this paper approaches how exact minimal solutions of the unate covering problem can be found several orders of magnitude faster than in the classical approach.

2 Unate Covering - the Problem

We consider a special SAT-problem without negated variables. Such functions are called *Petrick functions* $P(\mathbf{p})$. The Petrick function defined by (1) depends on 8 variables and is given by 8 clauses:

$$\begin{array}{l}(p_4 \vee p_5 \vee p_6 \vee p_8) \wedge (p_2 \vee p_3 \vee p_4 \vee p_7 \vee p_8) \wedge (p_1 \vee p_3 \vee p_4 \vee p_7 \vee p_8) \wedge \\ (p_1 \vee p_4 \vee p_5 \vee p_7 \vee p_8) \wedge (p_1 \vee p_2 \vee p_5 \vee p_6) \wedge (p_4 \vee p_5 \vee p_6 \vee p_7 \vee p_8) \wedge \\ (p_1 \vee p_4 \vee p_5 \vee p_6 \vee p_7 \vee p_8) \wedge (p_4 \vee p_6 \vee p_7) = 1.\end{array} \quad (1)$$

The classical approach to solve the unate covering problem applies the distributive law [1] to the clauses, simplifies the created conjunctions using the idempotence law [1], and reduces the found disjunctive form using the absorption law

[1]. A remarkable speedup of more than 10^4 is reached by the IT(DIST/ABS)-algorithm which applies the distributive law for the next clause and reduces the found conjunction immediately using the absorption law.

3 Proof of the CPL(NDM(P)) - Approach

In this extended abstract we can give only a very restricted sketch of the proof. The negation according to *de Morgan's Law* applying the XBOOLE-operator `NDM` to the Petrick function creates the complement of all allowed solutions. The complement of this intermediate solution set using the XBOOLE-operator `CPL` generates all allowed solutions. This set contains the desired solutions with a minimal number of values 1 which can be selected simply by counting the values 1 in the solution vectors.

4 Improved Algorithm: DIF(S_i, NDM(P))

We divide the Boolean space into symmetric functions $S_i(\mathbf{p})$. The symmetric functions $S_i(\mathbf{p})$ are generated as lists of all Boolean vectors which include exactly i values 1; values 0 are assigned to the remaining positions. We begin the calculation of $S = \texttt{DIF}(S_i, \texttt{NDM}(P))$ with $i = 0$ and increase i by 1 if $S = \emptyset$. The first solution set S which is not empty is the desired minimal solution. It is not necessary to evaluate the solution vectors furthermore.

5 Experimental Results

The improved basic algorithm IT(DIST/ABS) solves the example (1) within 6 milliseconds instead of 83.743 seconds required for the basic approach.

Restricted by the available memory, we could solve the benchmark of 16 p-variables and 32 clauses within 299.928 seconds using IT(DIST/ABS). The same 9 solution vectors with 3 variables were found within 3 milliseconds using the $\texttt{CPL}(\texttt{NDM}(P))$-algorithm. This is an improvement of a factor of 99,976.

The implemented $\texttt{CPL}(\texttt{NDM}(P))$-algorithm needs 734.171 seconds to solve the benchmark of 32 p-variables and 256 clauses. The new $\texttt{DIF}(S_i, \texttt{NDM}(P))$-algorithm finds the same 38 solution vectors with 5 variables within 91 milliseconds. This is an improvement of a factor of 8,068. Hence, in comparison with the IT(DIST/ABS)-algorithm we reduced the runtime by a factor of more than $8 * 10^8$.

References

1. Posthoff, Ch., Steinbach, B.: Logic Functions and Equations - Binary Models for Computer Science. Springer, Dordrecht, The Netherlands, 2004.
2. Steinbach, B. and Posthoff, Ch. *Logic Functions and Equations - Examples and Exercises*. Springer Science + Business Media B.V., 2009.

This article was processed using the LaTeX macro package with LLNCS style