Solving Satisfiability Problems on FPGAs using Experimental Unit Propagation Heuristic

Takayuki Suyama, Makoto Yokoo, and Akira Nagoya

NTT Communication Science Laboratories
2-4 Hikaridai Seika-cho Soraku-gun, Kyoto, JAPAN
{suyama, yokoo, nagoya}@cslab.kecl.ntt.co.jp

Abstract. This paper presents new results on an approach for solving satisfiability problems (SAT), that is, creating a logic circuit that is specialized to solve each problem instance on Field Programmable Gate Arrays (FPGAs). This approach has become feasible due to recent advances in Reconfigurable Computing.
We develop an algorithm that is suitable for a logic circuit implementation. This algorithm is basically equivalent to the Davis-Putnam procedure with Experimental Unit Propagation. The required hardware resources for the algorithm are less than those of MOM's heuristics.

1 Introduction

Recent reconfigurable hardware technologies enable users to rapidly create logic circuits specialized to solve each problem instance. We have chosen satisfiability problems to examine the effectiveness of this approach. In this paper, we present a new method of implementing an efficient algorithm with a powerful dynamic variable ordering heuristic on a logic circuit. This algorithm is basically equivalent to the Davis-Putnam procedure with Experimental Unit Propagation [2].

2 Design Flow

Figure 1 shows the logic synthesis flow of a logic circuit that solves a specific SAT problem.

3 Problem Definition

A satisfiability problem (SAT) for propositional formulas in conjunctive normal form can be defined as follows. A boolean *variable* x_i is a variable that takes the value true or false (represented as 1 or 0, respectively). We call the value assignment of one variable a *literal*. A *clause* is a disjunction of literals, e.g., $(x_1 + \overline{x_2} + x_3)$. Given a set of clauses $C_1, C_2, \ldots, C_m$ and variables $x_1, x_2, \ldots, x_n$, the satisfiability problem is to determine if the formula $C_1 \cdot C_2 \cdot \ldots \cdot C_m$ is satisfiable, i.e., to determine whether an assignment of values to the variables exists so that the above formula is true. This problem was the first computational task shown to be NP-complete.

4 Algorithm

The algorithm used in this paper is basically equivalent to the Davis-Putnam procedure that introduces a dynamic variable ordering with *Experimental Unit Propagation* (EUP) that is a heuristic used for the branching. The outline of the algorithm can be described as follows.

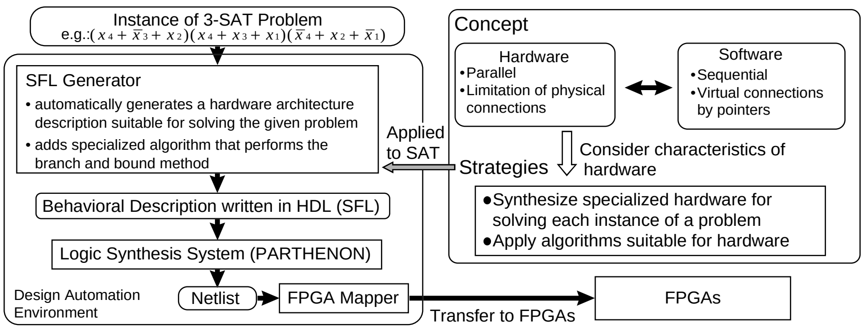
Instance of 3-SAT Problem
e.g.:$(x_4+\bar{x}_3+x_2)(x_4+x_3+x_1)(\bar{x}_4+x_2+\bar{x}_1)$
SFL Generator
• automatically generates a hardware architecture description suitable for solving the given problem
• adds specialized algorithm that performs the branch and bound method
Behavioral Description written in HDL (SFL)
Logic Synthesis System (PARTHENON)
Netlist
FPGA Mapper
Design Automation Environment
Applied to SAT
Concept
Hardware
•Parallel
•Limitation of physical connections
Software
•Sequential
•Virtual connections by pointers
Strategies
Consider characteristics of hardware
●Synthesize specialized hardware for solving each instance of a problem
●Apply algorithms suitable for hardware
Transfer to FPGAs
FPGAs

Fig. 1. Logic synthesis flow for SAT problems

Let I be the input SAT problem instance, and L be the working list containing problem instances, where L is initialized as $\{I\}$. A problem instance is represented as a pair of a set of value assignments and a set of clauses that are not yet satisfied.

1. If L is empty, then I is unsatisfiable, so stop the algorithm. Otherwise, select the first element S from L and remove S from L.
2. If S contains an empty clause, then S is unsatisfiable, go to 1.
3. If all clauses in S are satisfied, print the value assignments of S as one solution, go to 1.
4. If S contains a unit clause (a clause with only one variable), then set this variable to the value which satisfies the clause, simplify the clauses of S and go to 2.
5. **Branching:** In order to select a variable which should be assigned at the next step, count what time chain reactions of unit propagation occur when assuming each unassigned variable is set at 0 and 1. If an unsatisfied clause exists when the variable is set at 0, set it to 1, and vice versa. If this does not occur, the variable which has the largest counting number is set to 0. Go to 1.

The algorithm can be straightforwardly represented as a finite state machine.

5 Evaluation

We first evaluate the efficiency of the developed algorithm by software simulation. We use hard random 3-SAT problems as examples. The number of clauses divided by the number of variables, which is called *clause density*, is 4.3. In Figure 2, we show the log-scale plot of the average required time of over 100 problems, assuming the clock rate is 1MHz. We can see from Figure 2 that the search tree of MOM's[3] grows at the rate of $O(2^{n/17.3})$, where n is the number of variables. On the other hand, the search tree growing at the rate of the new method (EUP) is $O(2^{n/20.0})$. This shows that the new method is more efficient with a larger number of variables.

We use an ALTERA FLEX10K250 FPGA chip to implement the algorithm. We have implemented a particularly difficult 3-SAT AIM benchmark problem[1], "aim-100-2_0-no-1.cnf", which consists of 100 variables, 200 clauses. The number of Logic Cells (LCs) is 9,464 for implementing this problem. The utilized rate of LCs is 77%.

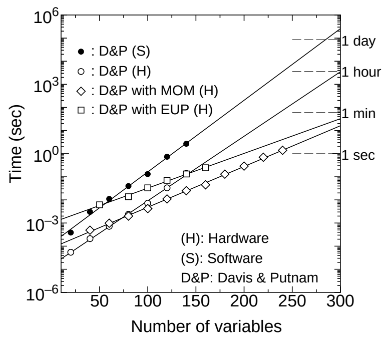

Fig. 2. Required time at 1MHz on hard random 3-SAT problems

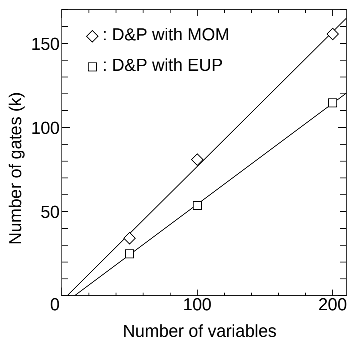

Fig. 3. Required gates for "aim-<#variables>-1_6-yes1-1.cnf"

A logic circuit simulator shows this circuit is capable of running at a clock rate of 12.07MHz.

Figure 3 shows the number of required gates for "aim-{50,100,200}-1_6-yes1-1.cnf". This shows the number of gates when the circuit is organized by primitive gates. Note that these circuits are initial ones synthesized from HDL descriptions. If these circuits are optimized, the number of gates can be reduced more with keeping the same trend. The required gates of EUP are about 30% less than that of MOM. This is because almost all the circuits of branching of EUP can be shared with the other steps. On the other hand, the MOM algorithm requires additional circuits for branching.

6 Conclusions

This paper presented new results on solving SAT using FPGAs. We developed an algorithm which is suitable for implementation on a logic circuit. This algorithm is basically equivalent to the Davis-Putnam procedure that introduces Experimental Unit Propagation. In this approach, a logic circuit specific to each problem instance is created on FPGAs. The number of gates required to implement the proposed method is 30% less than that of MOM. In addition, the order of the proposed method is better than that of MOM. We have actually implemented benchmark problems with 100 variables and it can run at 12MHz. In the future, we will refine the implementation of the algorithm on FPGAs, and perform various evaluations on implemented logic circuits.

References

1. Y. Asahiro, K. Iwama, and E. Miyano. Random generation of test instances with controlled attributes. In *Proceedings of the DIMACS Challenge II Workshop*, 1993.
2. C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proc. of 15th International Joint Conference on Artificial Intelligence*, pages 366–371, 1997.
3. T. Suyama, M. Yokoo, and H. Sawada. Solving satisfiability problems using logic synthesis and reconfigurable hardware. In *Proc. of the 31st Annual Hawaii International Conference on System Sciences Vol. VII*, pages 179–186, 1998.