The Exhaust Search for NP-Complete Using FPGA

Youhei Yamagishi (110123)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 2003

Keywords: NP-Complete, enumeration problem, exhaust search, FPGA, ${}_nC_k$ counter.

The *Satisfiability problem(SAT)* asks to check whether a Boolean formula $f(x_1, x_2, \ldots, x_n)$ is satisfiable. The *Enumeration problem* asks to list all the possible truth assignments that satisfy the formula f. For instance the Enumeration problem lists the solution $(x_1, x_2, x_3) = \{(0, 1, 0)\}$, when evaluating the Boolean formula $f(x_1, x_2, x_3) = (\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_3)$.

One can use heuristic approaches to solve the enumeration problem. Since, heuristic approaches usually do not list all possible solutions we need to continuously apply heuristics such that all possible solution are obtained. Eventually, these approaches run in exponential time to solve the problem.

Our approach is to build a circuit which generates all the possible assignments for the Boolean formula f. A simple exhaustive search can be applied to check if each generated assignment satisfies f. To speed up the verification process, we have implemented a fast circuit that performs the exhaustive search. More specifically, we use binary counter that generates all the combinations of possible truth assignments and a circuit that verifies if it satisfies f for all assignments. The binary counter generator and the verification circuits have been implemented on FPGAs(*Field Programmable Gate Array*).

The number of truth assignments is 2^n, and the exhaustive search takes a lot of time. Thus we consider the SAT with restricted truth assignment.

Concretely exhaustive search enumerates all satisfying assignment of the SAT problem with restricted truth assignments which assign true to exactly k variables. Therefore, all the possible $\binom{n}{k}$ assignments are verified if they satisfy f. For this purpose, we propose the ${}_nC_k$ counter algorithms that generate all the combination of "n choose k".

we designed the ${}_nC_k$ counters, implemented using FPGA, and evaluated the implemented circuits by the frequency and the number of total gates of FPGA.

Farther, we propose various counter algorithms, ${}_nC_{[0,k]}$counter that generates combinations of n choose at most k , ${}_nC_{[k,n]}$counter that generates combinations of n choose at least k and ${}_nC_{[i,j]}$counter that generate combinations of n choose between i and j. These counters have been also implemented in the FPGA and their performance have been evaluated.

These counter application is for examples, VERTEX-COVER problem and INDEPENDENT-SET, etc.

VERTEX-COVER is defined as follows: A *vertex cover* of an undirected graph $G = (V, E)$ is a set of vertices $U \subseteq V$ such that every edge in E is adjacent to some vertex in U. The *vertex cover problem* asks to determine, given $G = (V, E)$ and $k \geq 0$, whether there exists a vertex cover U in G of cardinality at least k. We use ${}_nC_{[0,k]}$counter to solve VERTEX-COVER.

INDEPENDENT-SET is defined as follows: An *independent set* of an undirected graph $G = (V, E)$ is a subset U of V such that $U^2 \cap E = \emptyset$, i.e.. no two vertices in U are connected by an edge in E. The *independent set problem* asks to determine, given $G = (V, E)$ and $k \leq 0$, whether G has an independent set U of cardinality at least k. We use ${}_nC_{[k,n]}$counter to solve INDEPENDENT-SET.

As we have seen above, the counters are very important ingredients for exhaustive search. Our purpose is accelerating counter or combination generator. Our main contribution is ${}_nC_k$ counter, because it is fast and simple.