Concurrent and Selective Logic Extraction with Timing Consideration

Peyman Rezvani, Massoud Pedram
University of Southern California
Los Angeles, CA 90089
{peyman,massoud}@zugros.usc.edu

Abstract - *We study the problem of concurrent and selective logic extraction in a Boolean circuit. We first model the problem using graph theory, prove it to be NP-hard, and subsequently formulate it as a Maximum-Weight Independent Set problem in a graph. We then use efficient heuristics for solving the MWIS problem. Concurrent logic extraction not only allows us to achieve larger literal saving and smaller area due to a more global view of the extraction space, but also provides us with a framework for reducing the circuit delay*

An efficient logic extraction algorithm, which is known as *Fast Extract*, was proposed by *J. Rajski* et al [1]. This algorithm only considers two-cube divisors and two-literal single-cube divisors. In spite of this limitation, *Fast Extract* has been shown to produce high quality extraction results and because of considering objects of size two, its time complexity is polynomial. A major drawback of this algorithm is that all extractions are "total" in the sense that once a divisor is selected, it is fully extracted from all the expressions in which it appears. Consequently, the order by which divisors are extracted has a significant impact on the quality of synthesized network. The order used in *Fast Extract* is generated by a greedy approach where divisors with highest literal savings are extracted first. In contrast, we consider in this paper the problem of selective extraction in such a way that all possible extractions (partial or total) of all candidate divisors are considered concurrently.

Each divisor may appear in a number of algebraic expressions for nodes in a network. In each case, there are two cubes that can be factored by the divisor. These two cubes are called a *Candidate Cube Pair* (CCP). Therefore, for each divisor, a set of CCPs may be found that correspond to the two-cube sub-expressions from which the divisor can be factored out. The selective extraction problem is to find which subset of the CCPs to choose and extract for each divisor. The key difficulty is that CCPs may be in conflict with each other in the sense that there exists a conflict between two CCPs when these CCPs have some cube in common.

Associated with each CCP is a *gain* which is the literal count saved by extracting the divisor out of that cube pair. For each divisor, a *cost* is defined which gives the cost of implementing the divisor itself and is simply the literal count of the divisor. The total value (or saving) of any selective extraction P of some divisor d with the CCP set Q ($P \subseteq Q$) is then given by:

$$value(P) = \sum_{CCP \in P} gain(CCP) - cost(d)$$

Formally stated, the selective logic extraction problem is to find subsets of the CCP sets for all divisors so as to maximize the total extraction value in the network while ensuring that no two selected CCPs are in conflict. This problem, which has been shown to be NP-Hard (proof is omitted for the sake of brevity), can be formulated as a Maximum Weight Independent Set (MWIS) problem in graph theory which is extensively studied and a number of effective heuristic methods for solving it exist. Details of this formulation are presented next.

A vertex-weighted graph, called a *conflict graph,* is constructed first. The vertices of the graph correspond to subsets of the CCP sets of all divisors. There exists an edge between two vertices if the two subsets have some conflicting CCPs. Notice that the vertices represent selective extractions of the candidate divisors. Each vertex is also assigned a weight, which is equal to the total value of the corresponding extraction. The goal is to find the MWIS of vertices in this conflict graph.

This extraction framework can be extended to consider timing issues while performing extraction. For example, we can limit the level increase in the network while maximizing the total extraction value in the network. More precisely, a *critical sub-network* is built out of the original network which only consists of timing critical paths in the circuit. A branch-and-bound approach is used for finding the MWIS starting with a seed solution and adding vertices one at a time. The delay cost of extraction is estimated by the level increase in the critical sub-network. An upper-bound on the literal saving and a lower-bound on the level increase are used to prune a sub-tree in the search space. This bounding technique is highly effective in reducing the runtime of the algorithm.

Experimental results on a number of benchmark circuits demonstrated an average improvement of 6% in terms of the literal count and 13% in terms of the circuit delay over *Fast Extract* with comparable runtimes.

[1] J. Rajski and J. Vasudevamurthy, "The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions", *IEEE Trans. on CAD*, June 1992.