Bound Set Selection and Circuit Re-Synthesis for Area/Delay Driven Decomposition

Andrés Martinelli Elena Dubrova
Royal Institute of Technology, IMIT/KTH, 164 46 Kista, Sweden
[andres,elena]@imit.kth.se

Abstract

This paper addresses two problems related to disjoint-support decomposition of Boolean functions. First, we present a heuristic for finding a subset of variables, X, which results in the disjoint-support decomposition $f(X,Y) = h(g(X),Y)$ with a good area/delay trade-off. Second, we present a technique for re-synthesis of the original circuit implementing $f(X,Y)$ into a circuit implementing the decomposed representation $h(g(X),Y)$. Preliminary experimental results indicate that the proposed approach has a significant potential.

1. Introduction

Disjoint-support decomposition of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is a representation of the form $f(X,Y) = h(g(X),Y)$ where $X \cap Y = \emptyset$, $g : \{0,1\}^{|X|} \rightarrow \{0,1,...,k-1\}$ and $h : \{0,1,...,k-1\} \times \{0,1\}^{|Y|} \rightarrow \{0,1\}$. The k-valued function g can be encoded as $f(X,Y) = h(g_1(X), g_2(X), \ldots, g_{\lceil \log_2 k \rceil}(X), Y)$ giving a decomposition with all functions being Boolean. Every set of variables X for which such a decomposition exists is called a *bound set* for f. This paper addresses two problems related to disjoint-support decomposition. First, we present a heuristic for finding a bound set which results in a disjoint-support achieving a good area/delay trade-off. Choosing a suitable bound set is important because disjoint-support decomposition does not necessarily simplify the function.

Second, we present a technique for transforming the original circuit implementing $f(X,Y)$ into a circuit implementing the decomposed representation $h(g(X),Y)$. Previous algorithms computed circuits for the decomposed representation from Binary Decision Diagrams (BDDs) of g and h, by applying various BDD-to-circuit transformation techniques. The algorithm presented in this paper uses BDDs only for *analysis* of the decomposition. The actual *synthesis* of the circuits for g and h is done by restricting the original circuit with respect to a given assignment of input variables. This guarantees that the sizes of the circuits of g and h are strictly smaller than the size of the original circuit.

2. Bound Set Selection

To find a suitable bound set X for f, we examine all linear intervals of variables of the BDD representing f. To check whether a given linear interval is a bound set, we use INTERVALCUT algorithm [1]. INTERVALCUT is very fast, because it does not require expensive BDD re-ordering.

If a bound set X with the column multiplicity $k < |X|$ is found, it is stored together with the following three parameters characterizing the associated decomposition $f(X,Y) = h(g(X),Y)$:

1. the number of outputs having X as a bound set: $s(X)$;
2. the number of outputs of g: $c(X) = \lceil \log_2 k \rceil$;
3. the difference in sizes of the bound set X and the free set Y: $d(X) = ||X| - |Y||$, $d(X) \in \{0, 1, \ldots, n-1\}$.

Let $\mathbf{X}$ be the set of bound sets computed by INTERVALCUT. The best candidate is selected from $\mathbf{X}$ as follows. First, a subset $\mathbf{X_s}$ of $\mathbf{X}$ containing all bound sets with the maximum $s(X)$ is chosen. Maximizing of $s(X)$ increases the sharing of common logic among different outputs of the circuit. Next, a subset $\mathbf{X_c}$ of $\mathbf{X_s}$ containing all bound sets with the minimum $c(X)$ is selected. Minimizing of $c(X)$ promotes the selection of bound sets with the smallest column multiplicity (more precisely, smallest $\log_2 k$). Finally, a subset $\mathbf{X_d}$ of $\mathbf{X_c}$ containing largest bound sets with the minimum $d(X)$ is obtained. Minimizing of $d(X)$ allows balancing the partitioning of logic between the functions g and h.

Any element of $\mathbf{X_d}$ is considered to be a "best" bound set for f, i.e. the one which produces a decomposition with the best area/delay trade-off. The original circuit implementing

f is transformed into the circuit implementing $h(g(X), Y)$ by applying the algorithm described in the next section.

3. Transformation Algorithm

Let X be a bound set for f and let G_g and G_h be BDDs representing the functions g and h in the decomposition $f(X, Y) = h(g(X), Y)$. These BDDs are computed by INTERVALCUT.

3.1. Constructing the circuit for h

Suppose A is an assignment of variables of X leading to the 0-terminal node in G_g. Then $g(A) = 0$, and thus $f(A, Y) = h(g(A), Y) = h(0, Y)$. Therefore, a circuit implementing the co-factor $h(0, Y)$ can be obtained from the circuit implementing f by applying the assignment A to the inputs X and propagating the constants through the circuit using the usual reduction rules. Similarly, circuits implementing co-factors $h(i, Y), i \in \{1, 2, \ldots, k-1\}$, can be obtained by propagating an assignment of variables of X leading to the i-terminal node of G_g. Recall, that g is a function of type $g : \{0, 1\}^{|X|} \rightarrow \{0, 1, ..., k-1\}$, so G_g is a multi-terminal BDD with k terminal nodes.

To maximize the sharing of common logic of the i circuits implementing co-factors $h(i, Y), i \in \{0, 1, \ldots, k-1\}$, i assignments A are chosen so that they differ in the fewest number of bit positions.

The function $h(g(X), Y)$ is obtained by combining the co-factors in a Shannon expansion as follows:

$$h(g(X), Y) = \sum_{i=0}^{k-1} g_1^{i_1}(X) g_2^{i_2}(X) \ldots g_r^{i_r}(X) h(i, Y) \quad (1)$$

where $(i_1, i_2, \ldots, i_r)$ is the binary expansion of i, $r = \lceil \log_2 k \rceil$, and the term $g_j^{i_j}$ is defined by

$$g_j^{i_j} = \begin{cases} g_j & \text{if } i_j = 1 \\ \overline{g}_j & \text{otherwise} \end{cases}$$

for $j \in \{1, 2, \ldots, r\}$.

3.2. Constructing the circuit for g

Suppose that B is an assignment of variables of Y such that $h(i, B) \neq h(j, B)$ for some $i, j \in \{0, 1, \ldots, k-1\}$, $i \neq j$. Then $f(X, B) = h(g(X), B)$ where the co-factor $h(g(X), B)$ is neither constant 0, nor constant 1, i.e. it depends of $g(X)$.

Since h is a function of type $\{0, 1, ..., k-1\} \times \{0, 1\}^{|Y|} \rightarrow \{0, 1\}$, the co-factor $h(g(X), B)$ is a function of type $\{0, 1, ..., k-1\} \rightarrow \{0, 1\}$. Note that, for $k = 2$, $h(g(X), B)$ is either an identity, or a complement. Thus, at this step, the problem of constructing the circuit for $g(X)$ is solved for $k = 2$. For larger values of k, the following strategy is used.

The k-valued function $g(X)$ can be expressed as

$$g(X) = \sum_{i=0}^{k-1} i \cdot g^i(X)$$

where $g^i : \{0, 1, \ldots, k-1\}^{|X|} \rightarrow \{0, 1\}$ are multiple-valued *literals* defined as:

$$g^i(X) = \begin{cases} 1 & \text{if } g(X) = i \\ 0 & \text{otherwise} \end{cases}$$

For a given encoding of k values of $g(K)$, each of the functions $g_1(X), g_2(X), \ldots, g_r(X)$, $r = \lceil \log_2 k \rceil$, encoding $g(X)$, can be represented as a sum of some literals $g^i(X)$'s.

Consider a decomposition chart of $h(g(X), Y)$ with columns representing k values of $g(X)$ and the rows represent all combinations of the variables of Y. Any non-constant row of $h(g(X), Y)$ represents a sum of some literals $g^i(X)$, $i \in \{0, 1, \ldots, k-1\}$.

In the best case, there exist rows in the decomposition chart corresponding directly to the encoded functions $g_1(X), g_2(X), \ldots, g_r(X)$. If $h(g(X), A) = g_j(X)$ for some assignment A of the variables of Y, then the circuit implementing $g_j(X)$ can be obtained from the circuit implementing f by applying the assignment A to the inputs Y and propagating the constants.

In the worst case, the literals $g^i(X)$, $i \in \{0, 1, \ldots, k-1\}$, need to be computed by ANDing selected rows of $h(g(X), Y)$. Afterward, the functions $g_1(X), g_2(X), \ldots, g_r(X)$ are obtained as a combination of $g^i(X)$.

4. Conclusion and Future Work

This paper has two contributions: (1) a heuristic for finding a bound set X which results in the disjoint-support decomposition with a good area/delay trade-off; (2) an algorithm which transforms the original circuit into the decomposed circuit.

Our preliminary experimental results on IWLS'02 benchmarks set show that the proposed technique usually results in a smoother trade-off between area and delay compared to the one of SIS. More experiments are needed to make a thorough evaluation.

References

[1] A. Martinelli, T. Bengtsson, E. Dubrova, and A. J. Sullivan, "Roth-Karp decomposition of large Boolean functions with application to logic design," in *Proceedings of NORCHIP'02*, (Copenhagen, Denmark), November 2002.