[20] X. Du, G. Hachtel, B. Lin, and A. R. Newton, "MUSE: A multilevel symbolic encoding algorithm for state assignment," *IEEE Trans. Computer-Aided Design.*, vol. 10, pp. 28-38, Jan. 1991.

Comparing Two-Level and Ordered Binary Decision Diagram Representations of Logic Functions

Srinivas Devadas

***Abstract*—We give an example of a class of functions with $2n + \log(n)$ inputs that have two-level or sum-of-products representations containing n^2 product terms and ordered binary decision diagram representations that have at least $\Omega(2^{n/2})$ vertices under any possible variable ordering.**

I. Introduction

Sum-of-product or two-level representations of logic functions have been used widely in the area of logic optimization and verification. There are many families of logic functions, commonly used in VLSI circuits, that have sum-of-product representations that grow exponentially with the number of inputs to the function. The Achilles Heel function [1], the parity function, and the multiply function over n inputs are popular examples of Boolean functions with exponentially growing sum-of-product representations.

Reduced ordered binary decision diagrams (OBDD's), proposed by Bryant in 1986 [2], have emerged as another useful representation of logic functions, especially so for logic verification applications. Many functions like the Achilles Heel and parity have linear-sized OBDD representations, under certain variable orderings. In [3] it was shown that a n-bit multiplier requires $\Omega(1.09^n)$ vertices, under *any* possible variable ordering.

In this paper, we give an example of a function with $2n + \log(n)$ inputs that has n^2 product terms in a sum-of-products representation, and using the theoretical framework developed in [3], we show that the function has $\Omega(2^{n/2})$ vertices in an OBDD representation, under any possible variable ordering.

II. Definitions

For standard logic synthesis and ordered binary decision diagram (OBDD) terminology, the reader is referred to [1] and [2] respectively. We define some terms relating to VLSI complexity that are necessary for our proofs. These definitions have been taken from [3] or modified from [4].

Let f denote a family of single-output Boolean functions parameterized by problem size n. The set of inputs is denoted by X. An *input assignment* x: $X \rightarrow \{0, 1\}$ is an assignment of Boolean values to the inputs. Given an input assignment x, the resulting output is denoted $f(x) \in \{0, 1\}$.

In an OBDD, all vertex labels must occur according to a total ordering of the variables. That is, for a set of input variables X, we assign each variable an "index" according to a bijection π: $X \rightarrow \{1, \cdots, |X|\}$. The variable ordering given by π is written as the sequence $[\pi^{-1}(1), \pi^{-1}(2), \cdots, \pi^{-1}(|X|)]$.

The lower bound arguments for OBDD complexity as described in [3] have a general form as follows: Some subset of inputs $Y \subseteq X$ is designated as the set of "key" inputs, and a real-valued balance parameter ω between 0 and 1 is specified. A *balanced partitioning* is defined as any partitioning of X into subsets L and R such that the fraction of Y in L equals ω. For our purposes $\omega = 0.5$.

Proving a lower bound involves showing that for every balanced partition, computing f requires that a certain amount of information about the assignment to inputs in L must be combined with a certain amount of information about the assignment to inputs in R. In the case of an OBDD, this information implies that the graph must have a certain number of arcs crossing from vertices labeled by variables in L to vertices labeled by variables in R. A lower bound on information transfer is established by creating a fooling set for the input partition; i.e., a set of input assignments no two elements of which can be computed by communicating the same information across the partition.

For a particular partitioning, define a *left* (*right*) input assignment l: $L \rightarrow \{0, 1\}$ (r: $R \rightarrow \{0, 1\}$) as an assignment of Boolean values to the inputs in $L(R)$. Let $l \cdot r$ denote the complete input assignment resulting from the left input assignment l and right input assignment r.

For OBDD's, a *fooling set* consists of left input assignments. That is, for partition (L, R), a set of left input assignments $A_{\text{OBDD}}(L, R)$ is a fooling set if it satisfies the following property: For any two distinct l and l' in A_{OBDD} there exists a right input assignment r such that $f(l' \cdot r) \neq f(l \cdot r)$. Such a right assignment is said to *distinguish* between l and l'.

The main result we use is given below:

Lemma: (Lemma 2 in [3]): If for every balanced partition (L, R), function f has a fooling set $A_{\text{OBDD}}(L, R)$ containing at least c^n elements, for some $c > 1$, then any OBDD for f must have $\Omega(c^n)$ vertices.

III. The Logic Function and Its Properties

The logic function we will analyze is given below. The function has $2n + \lceil \log(n) \rceil$ inputs, corresponding to $a_0, \cdots, a_{n-1}, b_0, \cdots, b_{n-1}$ and $mux_1, \cdots, mux_{\lceil \log(n) \rceil}$.

$f = g_i$ if $value(mux_1, \cdots, mux_{\lceil \log(n) \rceil}) = i$, where function $value(\ )$ returns the integer value of the input binary combination. The g_i are defined as follows:

$$g_i, 0 \leq i < n = \sum_{j=0}^{n-1} (a_j \cdot b_{(j+i) \bmod n}).$$

Each g_i function has n product terms. There are n such functions, and each function is ANDed by an input combination over the mux_l's, resulting in the f function having n^2 product terms.

We now show that any OBDD representation for f will have $\Omega(2^{n/2})$ vertices (under any possible variable ordering).

Theorem: Any OBDD representation of f requires $\Omega(2^{n/2})$ vertices.

Proof: We will select the key inputs Y to be the $2n$ inputs corresponding to $a_0, \cdots, a_{n-1}$ and $b_0, \cdots, b_{n-1}$. Choose $\omega = 0.5$. We therefore have a partition (L, R) wherein L has n inputs that are some a_i or b_j, and so does R. The mux_l variables may be contained in L or R or may be distributed across both.

In the sequel, we will construct a fooling set $A_{\text{OBDD}}(L, R)$ that

Manuscript received June 17, 1991; revised April 20, 1992. This paper was recommended by Associate Editor R. K. Brayton.

The author is with the Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139.

IEEE Log Number 9202996.

has $2^{n/2}$ elements. Assume that there are p variables corresponding to the a_i's and q variables corresponding to the b_j's in L. We know that $0 \le p \le n$, $0 \le q \le n$, and $p + q = n$. Further, we have q a_i variables in R and p b_j variables in R.

We will show that for any partition (L, R), we will always be able to pick a g_z such that $\ge n/2$ terms in g_z satisfy the property that the a_i variable in the term belongs to L, and the b_j variable to R, or vice versa. That is, these terms will be *split* across the partition. The number of terms in all the g_k's, where any term $a_i \cdot b_j$ has $a_i \in L$ and $b_j \in L$ is pq. Similarly the number of terms in all the g_k's $a_i \cdot b_j$ such that $a_i \in R$ and $b_j \in R$ is pq. We thus have $2pq$ terms that are not split across the partition. Given that $p + q = n$, the maximum of $2pq$ occurs at $p = q = n/2$. Since the total number of terms in f is n^2, the number of terms split across the partition is $\ge n^2 - 2(n/2)(n/2) = n^2/2$. In general for different selections of L and R, these terms will be contained in different g_k's. Given there are ng_k's we will always have at least one g_z which will have at least $(1/n)(n^2/2)$ terms split across the partition.[1]

Now that we have a selected g_z, we use the settings for the mux_l variables that make $f = g_z$. We will now show that we have a fooling set $A_{\text{OBDD}}(L, R)$ for f that is of cardinality $2^{n/2}$. The fooling set corresponds to all possible settings of the $n/2$ a_i's or b_j's in L that are contained in terms of g_z split across the partition. The a_i and b_j variables in L *not* in the $n/2$ terms that are split across the partition are set to constant 0 values for all possible settings of the variables that are in the split terms.

Assume we have two settings of the variables in L, namely l and l'. We pick a_m such that $l(a_m) \ne l'(a_m)$. The term $a_m \cdot b_s$ belongs to g_z, such that $b_s \in R$. We set the values of all a_i and b_j variables in R to 0, except for $b_s = 1$. Call this assignment (along with the mux_l variable assignment) r. Because $b_s = 1$ and all terms in g_z except $a_m \cdot b_s$ have been set to 0, $g_z(l \cdot r) \ne g_z(l' \cdot r)$. Therefore, $f(l \cdot r) \ne f(l' \cdot r)$. We can pick a b_m instead of a_m above and use similar arguments to show that two settings l and l' that differ in b_m can be differentiated. Thus, we can differentiate any two members of the constructed fooling set.

Given that we have a fooling set $A_{\text{OBDD}}(L, R)$ with $2^{n/2}$ elements invoking the lemma above gives us the desired result. □

IV. Summary

Families of logic functions having OBDD representations that grow linearly with the number of inputs (under particular orderings), but sum-of-product representations that grow exponentially with the number of inputs, have been known for a long time. We have provided an example of a function that has a quadratic-sized sum-of-product representation, but an exponential-sized OBDD representation, under *any* possible variable ordering. A larger class of functions similar to the one above, where f is a multiplexor composition of n/k functions (k is a constant) that have nk product terms and $\Omega(2^{n/2k})$ vertices in any OBDD representation, can be derived.

V. Acknowledgment

Discussions with Pranav Ashar and Abhijit Ghosh that led to the discovery of the function analyzed here are acknowledged. Thanks to Randy Bryant for suggesting the use of the theory developed in [3].

[1]In fact, the minimum corresponds to the case where *all* the g_k's have exactly $n/2$ terms split across the partition.

References

[1] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Norwell, MA: Kluwer Academic, 1984.

[2] R. Bryant, "Graph-based algorithms for Boolean function manipulation," in *IEEE Trans. Computers*, vol. C-35, pp. 677–691, Aug. 1986.

[3] R. Bryant, "On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication," in *IEEE Trans. Computers*, vol. 40, pp. 205–213, Feb. 1991.

[4] R. Lipton and R. Sedgewick, "Lower bounds for VLSI," in *Proc. ACM Symp. Theory of Computation*, pp, 300–307, June 1981.

Physical Models and Efficient Algorithms for Over-the-Cell Routing in Standard Cell Design

Jason Cong, Bryan Preas, and C. L. Liu

***Abstract*—When an over-the-cell routing layer is available for standard cell layout, efficient utilization of that routing space over the cells can significantly reduce layout area. In this paper, we present three physical models to utilize the area over the cells for routing in standard cell designs. We also present efficient algorithms to choose and to route a planar subset of nets over the cells so that the resulting channel density is reduced as much as possible. For each of the physical models, we show how to arrange inter-cell routing, over-the-cell routing, and power/ground buses to achieve valid routing solutions. Each algorithm exploits the particular arrangement in the corresponding physical model and produces provably good results in polynomial time. We tested our algorithms on two industrial standard cell designs. In these tests, this method reduces total channel density by as much as 21%.**

I. Introduction

Standard cells are widely used in the design of VLSI circuits. After the cells are placed in rows and necessary feedthroughs inserted, a channel router completes the interconnections in the channels among the cells (Fig. 1). Conventional channel routers are restricted to utilizing two routing layers in the channels for interconnections. The conventional channel routing problem has been extensively studied, and there are several channel routers which can produce solutions using at most one or two tracks more than channel density for most practical problems (for example, see [7], [23], [20], [1], [19]). To further reduce the channel routing area, some channel routers use the extra routing area over the cells for interconnections [9], [15], [21], [14], [3], [5]. These routers are

Manuscript received December 11, 1990; revised December 2, 1991. This work was partially supported by the National Science Foundation under Grants MIP 87-03273 and MIP 91-10511, by the Semiconductor Research Corporation under Contract 87-DP-109, and by a grant from the Alexander von Humboldt Foundation. This paper was recommended by Associate Editor M. Marek-Sadowska.

J. Cong is with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90024.

B. Preas was with the University of Paderborn, Germany, when the work was performed. He is currently with the Xerox Palo Alto Research Center, Palo Alto, CA 94304.

C. L. Liu is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE Log Number 9200915.