

Optimizing Logic Design Using Boolean Transforms

Pramit Chavda*

Tata-IBM, Bangalore 560017, India
pchavda@vnet.ibm.com

James Jacob†

Nortel, Ottawa K1Y 4H7, Canada
jacobj@nortel.ca

Vishwani D. Agrawal

Bell Labs, Murray Hill, NJ 07974, USA
va@research.bell-labs.com

Abstract

When a Boolean function is transformed by exclusive-OR with a suitably selected transform function, the new function is often synthesized with significantly reduced hardware. The transform function is separately synthesized and the original function is recovered as an exclusive-OR of the two functions. We select the transform to reduce the number of cubes in the function to be synthesized. The function is represented as a Shannon expansion about selected variables. A transform function is constructed such that a selected set of cofactors is complemented to minimize the overall number of cubes. Examples of single-output functions show an average area reduction of 19%. For a multiple-output function, transformations can be customized for each output.

1 Introduction

The size of the implementation for a logic function is often measured in terms of gates or, for a specific technology, as chip area. Other relevant characteristics are delay, testability and power consumption. The present work deals with the size alone. Most complex Boolean functions have about half of their minterms as true [1]. For arbitrary multiple output functions, the average multi-level implementation size has been empirically studied by Cheng and Agrawal [2]. Most previous work is based on the minterm representation of functions. Large functions, however, are generally described more compactly by cubes or product terms of their minimized sum of products representation. Therefore, we will use the cube description.

In a recent paper [3], we proposed the use of exclusive-OR transforms on input variables for adder and comparator circuits. Similar transforms are also used in spectral methods [4]. The basic disadvantage is that a common transform must be found for all output functions. In the output transform method, as discussed in the present work, each output function can have a customized transform that best suits it.

*Formerly with Indian Institute of Science, Bangalore 560012, India, pramit@protocol.ece.iisc.ernet.in

†On leave from Indian Institute of Science, Bangalore 560012, India, james@ece.iisc.ernet.in

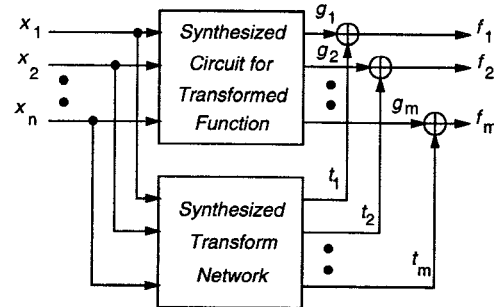


Figure 1: Synthesis architecture

2 The Proposed Synthesis Architecture

Figure 1 shows an m output function (f_1, f_2, \dots, f_m) of n variables x_1, x_2, \dots, x_n . A set of transform functions, (t_1, t_2, \dots, t_m) , is derived from the sum of products description of the specified outputs. These are obtained by a function-specific transform network whose inputs are the primary input variables. The functions to be synthesized are derived as:

$$g_i = f_i \oplus t_i \quad (1)$$

Each f_i is recovered by an exclusive-OR gate:

$$f_i = g_i \oplus t_i \quad (2)$$

The cost of the transformation is m exclusive-OR gates at the output and the network to derive t_i s. The maximum number of exclusive-OR gates needed is m . The combined size of the entire network including the transform network and the output exclusive-ORs should require fewer gates than a conventionally designed multi-level circuit. The contribution of this paper is a method of finding t_i s such that the complexity of g_i s is significantly reduced, leading to an overall reduction in the size of the complete implementation.

In general, the transform network is simpler and can be synthesized with lower delay than the network that realizes g_i s. Since the transform simplifies g_i s with respect to f_i s, we can expect g_i s also to have lower delays. However, there are added delays due to extra fanouts at primary inputs and the exclusive-OR gates at the outputs. In the

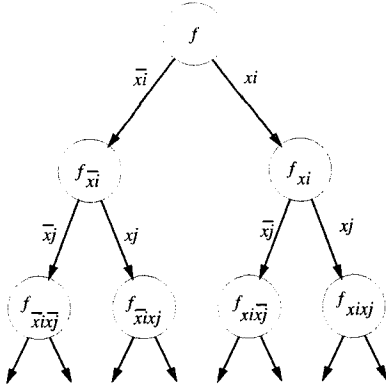


Figure 2: Recursive application of Shannon expansion

examples, we have only emphasized area optimization. Still, delay penalties were either very small or none.

3 Theory

We restrict our discussion to single output functions. This procedure can be applied successively to each function of a multi-output circuit.

In experiments with random functions we observed a good correlation between the number of cubes in the minimum sum of products (MSOP) form of a function and the area of synthesized multi-level implementation [5]. Since our synthesis architecture indirectly implements f using g and t , we shall aim to obtain a low cost transform function t , which on exclusive-ORing with f will produce a g with minimum number of cubes. We give a heuristic based on recursive Shannon expansion to obtain a transform t which attempts to satisfy the above criteria.

Consider a Boolean function $f(X) = f(x_1, x_2, \dots, x_n)$ of independent variables x_i s. The Shannon expansion [6] of a function around variable x_i is obtained as:

$$f(x_1, x_2, \dots, x_i, \dots, x_n) = \bar{x}_i f(x_1, x_2, \dots, 0, \dots, x_n) + x_i f(x_1, x_2, \dots, 1, \dots, x_n) \quad (3)$$

This may be expressed compactly as:

$$f(X) = \bar{x}_i f_{\bar{x}_i} + x_i f_{x_i} \quad (4)$$

We will call $f_{\bar{x}_i}$ and f_{x_i} as *cofactors*. The expansion can be applied recursively to divide a function into subfunctions, depicted as a binary tree in Figure 2.

We obtain the Shannon expansion of f about a selected subset of input variables. The choice of the input variables and their order of application will be discussed in the next subsection. Having expanded f about the first variable x_i , we minimize the two subfunction in the current level of the expansion tree using ESPRESSO to obtain the MSOP forms for their ON and OFF sets. If the MSOP form of the OFF set of a subfunction has lower

cost (number of cubes), we would like to complement that subfunction. Suppose, we decide to complement f_{x_i} . We will transform f to g according to Equation 1, using x_i as the transform function. Thus,

$$g = f \oplus x_i = (\bar{x}_i f_{\bar{x}_i} + x_i f_{x_i}) \oplus x_i \quad (5)$$

On simplification, we get

$$g = \bar{x}_i f_{\bar{x}_i} + x_i \bar{f}_{x_i} \quad (6)$$

Thus, the Shannon expansion of g about x_i has the same cofactors as those of f except the cofactor with respect to x_i has been complemented. A generalization of this procedure becomes evident by a later example.

In Figure 2, suppose the expansion about x_i does not provide an adequate reduction in cubes. We then postpone the selection of t and proceed to the next level of expansion, about the variable x_j . Now we have four cofactors. Any of these can be complemented to reduce the number of cubes. For example, $f_{x_i \bar{x}_j}$ will be complemented if t is chosen as $x_i x_j$. Similarly, it can be shown that more than one cofactor will be complemented if the transform function is a disjunction of the individual transform functions.

At any stage in the expansion, the total number of cubes for g has an *upper bound* that is the sum of cubes in all cofactors in that stage. After we minimize this upper bound by selectively complementing cofactors, the corresponding transform function t is constructed and ESPRESSO (-Dxor option) is used to find the MSOP form for $f \oplus t$. Once the reduction in cubes satisfies some user specified threshold, we construct the transform function. Otherwise, we continue the search for a better transform function, through expanding f further about remaining variables. The process is continued until the reduction in the number of cubes exceeds the specified threshold (15% in our experiments) or until the cost of t increases to a point that does not justify further Shannon expansion.

Finding the optimal subset of variables for Shannon expansion is a non-trivial problem. In our experience, we observed that the most suitable input variables have the least number of *don't cares* (X) with a balanced distribution of zeros and ones in the MSOP form. We have employed the following heuristic non-binate-ness measure (NM) as a figure of merit to choose between the various input variables of f . We define, for each x_i :

$$NM(i) = 2 \times N_x(i) + |N_1(i) - N_0(i)| \quad (7)$$

where $N_x(i)$, $N_0(i)$ and $N_1(i)$ denote, respectively, the number of *don't cares* (X), 0s and 1s in the i^{th} column of the MSOP form of f . Variable with the smallest NM is empirically found to give good results. This generally has a small number of Xs and hence corresponds to a

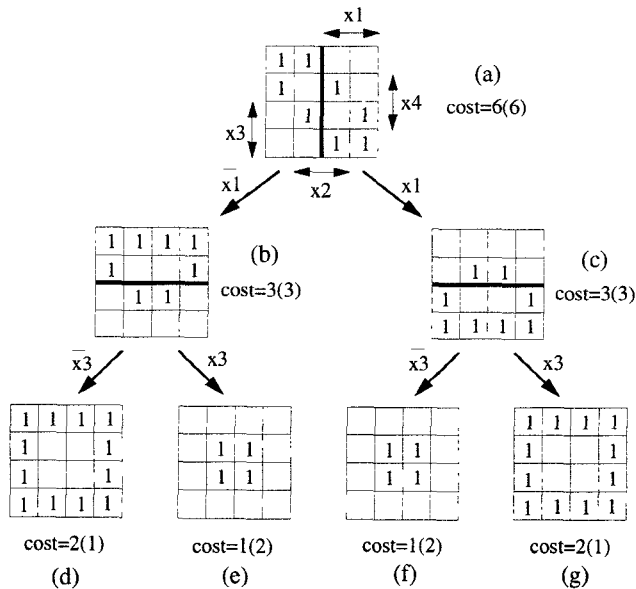


Figure 3: Transform function for f , $t = x_1.x_3 + \bar{x}_1.\bar{x}_3$. Costs in parentheses are for complemented functions.

variable that the function strongly depends on. Having a symmetric distribution of 0s and 1s is likely to make the two subfunctions after Shannon expansion, to be similar in complexity. Experiments show the usefulness of this heuristic. We minimize f using ESPRESSO to obtain its MSOP form and list the input variables in ascending order of their non-binateness measure NM . Shannon expansion will use the above ordering of input variables.

As an example, consider the Boolean function $f(x_1, x_2, x_3, x_4)$ in the Karnaugh map (a) of Figure 3. Its MSOP form is shown in Table 1. The figures of merit for input variables, as computed from Equation 7, are: $NM(1) = 0$, $NM(2) = 4$, $NM(3) = 0$ and $NM(4) = 4$. Hence the ordering of variables for Shannon expansion will be x_1, x_3, x_2 and x_4 . We first expand the function around variable x_1 . This gives the two subfunctions $\bar{x}_1 f_{\bar{x}_1} = (000X, 0X00, 0111)$ and $x_1 f_{x_1} = (1101, 101X, 1X10)$ as shown on either side of the bold vertical line in the center of the Karnaugh map (a) of Figure 3. The subfunctions $f_{\bar{x}_1} = (X00X, XX00, X111)$ and $f_{x_1} = (X101, X01X, XX10)$ are shown on the Karnaugh maps (b) and (c) of Figure 3. The cost of each subfunction as well as its complement is three cubes. Since no cube reduction is possible by complementing any subfunction, we proceed to the next level of recursion and obtain Shannon expansions around variable x_3 , for each subfunction. We now have four subfunctions as shown in Karnaugh maps (d) through (g) of Figure 3. Clearly, the cost of subfunctions $f_{\bar{x}_1 \bar{x}_3}$ in (d) and $f_{x_1 x_3}$ in (g) can be reduced by taking the complement of these subfunctions. Those complements have a single cube each. This is done by exclusive-ORing with $\bar{x}_1 \bar{x}_3$ and $x_1 x_3$, respec-

Table 1: Example function f

x_1	x_2	x_3	x_4	f
0	0	0	X	1
0	X	0	0	1
0	1	1	1	1
1	1	0	1	1
1	0	1	X	1
1	X	1	0	1

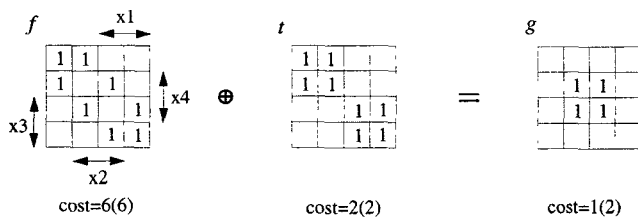


Figure 4: Derivation of a transformed function

tively. Hence the transform function will have two cubes; i.e., $t = (0X0X, 1X1X)$. The function g as obtained from Equation 1 is shown in Figure 4. Notice that function g is obtained by complementing those minterms of f for which t is 1. Since we have achieved a significant reduction (6 cubes to 1 cube), we accept this transform function. The transform based realization is shown in Figure 5. The transform function $t = \bar{x}_1 \bar{x}_3 + x_1 x_3$, is a single exclusive-NOR gate.

4 Experimental Results

We synthesized several PLAs from the ESPRESSO benchmarks [7]. First, an output function was minimized (as single output function) using the *-Dso_both* option of ESPRESSO and the set having the smaller number of cubes among ON and OFF sets was chosen as a reference. We then derive a transform function t and obtain the transformed function g with a lower cost than the reference. Only those output functions of PLAs that yield reasonable reduction in multi-level area using the transform approach, are reported in Table 2. The PLA name, the number of inputs (I) and the output number ($O \#$) denoting the output function that was optimized by our approach are given in columns 1 to 3, respectively. Under *# Cubes*, we list the number of cubes in the MSOP form for f , t and g . Then f , t and g were synthesized using SIS Version 1.2 [8] and technology mapped using the *mcnc.genlib* standard cell library supported by the package. The synthesis script (which uses *script.rugged*) of SIS is given below:

```
read_pla "pla_name"
resub -a
simplify -d
sweep; eliminate -1
simplify -m nocomp
```

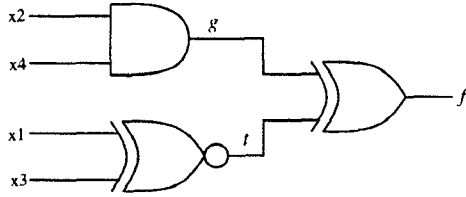


Figure 5: Transform based realization

Table 2: Experimental results for customized transform

PLA Name	I	O #	Cubes			Multi-level Area				
			f	t	g	f	t	g	Tot %red	
alu2	10	1	5	2	2	19	3	8	16	15.8
alu2	10	6	17	2	13	56	5	42	52	7.1
alu3	10	1	4	1	2	14	3	3	11	21.4
b9	16	1	9	1	7	55	3	40	48	12.7
dc2	8	4	12	1	9	67	3	49	57	14.9
dist	8	4	40	1	34	228	2	189	196	14.0
dist	8	5	43	1	36	209	0	181	186	11.0
ex7	16	1	9	1	7	55	3	40	48	12.7
exps	8	9	25	1	21	186	6	131	142	23.7
f51m	8	1	23	1	11	99	0	43	48	51.5
f51m	8	2	18	1	8	68	0	40	45	33.8
f51m	8	3	14	1	6	51	0	28	33	35.3
f51m	8	5	5	1	2	20	0	9	14	30.0
lin.rom	7	1	24	3	19	133	11	102	118	11.3
lin.rom	7	11	22	2	19	111	5	93	103	7.2
lin.rom	7	13	19	1	15	95	0	69	74	22.1
lin.rom	7	21	11	1	8	54	3	42	50	7.4
max128	9	18	25	1	20	123	0	103	108	12.2
mlp4	8	4	36	2	31	200	10	145	160	20.0
prom2	9	3	15	1	19	134	3	106	114	14.9
intb	15	4	90	1	80	301	0	218	223	25.9
Ave.	-	-	22	1	18	109	3	80	88	19.0

```

eliminate -1
sweep; eliminate 5
simplify -m nocomp
resub -a
fx
resub -a; sweep
eliminate -1; sweep
full_simplify -m nocomp
read_library mcnc.genlib
map -m0 -AF
print_map_stats

```

The multi-level area after synthesis and mapping of these functions is reported in columns labeled *Multi-level Area*. The *Tot* area is that required for the transform based implementation including the areas of *t* and *g* and the cost of a two-input Exclusive-OR gate (5 units in the *mcnc.genlib* library). When *t* has one cube with a single variable, the area of *t* is zero as that variable becomes a direct input to the output exclusive-OR gate. Finally, the percentage reduction, $\% red = 100(f - tot)/f$, in the multi-level area relative to the reference is reported in the

last column. Averages for cubes and areas for 21 PLAs are given in the last row. We notice that the average saving in area is 19% and that on an average the transform function *t* required only one cube.

We did not always succeed in obtaining multi-level area reduction, even in cases where we get significant reduction in the number of cubes. Further, we could not always find a transform function that can achieve reduction in number of cubes above the specified threshold (15% in this case). However, our experiments with random functions revealed that in more than 60% of cases where cube reduction in excess of 10% was obtained, we could also achieve reduction in multi-level area by the transform approach.

5 Conclusion

The novel synthesis approach using Boolean transforms optimizes a logic design beyond what is possible with present-day synthesis tools. Our focus was the reduction of area. We outlined a heuristic procedure based on recursive Shannon expansion of a function, for the design of a suitable transform function. Experimental results on several benchmark PLAs reveal the potential of the transform based synthesis approach. Further work is required to devise transform functions to optimize other relevant performance parameters such as delay, power consumption and testability.

References

- [1] R. W. Cook and M. J. Flynn, "Logical Network Cost and Entropy," *IEEE Trans. Computers*, vol. C-22, pp. 823-826, September 1973.
- [2] K. T. Cheng and V. D. Agrawal, "An Entropy Measure for the Complexity of Multi-Output Boolean Functions," in *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 302-305, 1990.
- [3] J. Jacob, P. S. Sivakumar, and V. D. Agrawal, "Adder and Comparator Synthesis with Exclusive-OR Transform of Inputs," in *Proc. 10th International Conf. VLSI Design*, pp. 514-515, 1997.
- [4] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*. London: Academic Press, 1985.
- [5] P. Chavda, "Boolean Transforms in Logic Synthesis," Master's thesis, Indian Institute of Science, Dept. of ECE, Bangalore, 1997.
- [6] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw Hill, 1978.
- [7] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston: Kluwer Academic Publishers, 1984.
- [8] E. M. Sentovich *et al.*, "SIS: A System for Sequential Circuit Synthesis," Memorandum UCB/ERL M92/41, University of California, Berkeley, CA, May 1992.