EXACT MINIMIZATION OF NETWORKS WITH COMPLEX GATES USING TERMINAL SUPPRESSED BINARY DECISION DIAGRAMS AND DISSECTED PAIRS

Craig Files +, Andre Reis, Marek Perkowski +, Michel Robert, Daniel Auvergne
L.I.R.M.M. University of Montpellier II - CNRS
Montpellier, France.
+ on sabbatical leave from Portland State University,
Department of Electrical Engineering, USA

1 Introduction

This paper describes a top-end minimization algorithm/program called COMPLEX that is used to optimize a complex gate network. The basis for this program, is the use of a library-free layout generator, instead of a standard pre-characterized library, such as the traditional layout synthesis method of standard-cells. The library-free cells are made up of static CMOS complex gates(SCCGs), where the quality of the final layout is based on the initial network (input to the layout generator). That is, if the initial network is given in a form that cannot be easily implemented in complex gates, then the final layout will not be good. In this paper, a method that produces "high-quality" initial networks, using a new approach based on minimizing the complex gates is presented.

This paper is organized as follows: Section 2 discusses the technology mapping in TABA. In Section 3, the algorithm for minimizing the complex gate network is given. Section 4 shows the initial experimental results of this method, and Section 5 concludes the paper.

2 Technology Mapping in TABA

The Library Free Technology Mapping (LFTM) tool of LIRMM, TABA (Transistor Assignment with BDDs representing AOIs), is able to read a functional or structural network containing simple gates (AND, NAND, OR, NOR, XOR, NXOR and INVERTERs) and output to TROPIC (an automatic layout generator from LIRMM) a hierarchical SPICE netlist containing SCCGs. Just like the tools presented in [5, 6] TABA is a technology mapping tool where the final result depends on the quality of the initial network.

2.1 Binary Decision Diagrams

A BDD is a rooted directed acyclic graph whose vertex set is V. Each non-leaf or non-terminal vertex has as attributes a pointer $index(v) \in \{1, 2, ..., n\}$ to an input variable in the set $\{x_1, x_2, ..., x_n\}$, and two children $low(v), high(v) \in V$. A leaf or terminal vertex v has only one attribute value $value(v) \in B$, where B is the binary set {0,1}. In this paper, every arc with head v and tail $low(v)$ is a $S0$ arc. Similarly, every arc with head v and tail $high(v)$ is a $S1$ arc.

Two special kinds of BDDs have been introduced, the Reduced Ordered BDDs (ROBDDs), Bryant [7, 8], and the Zero-suppressed BDDs (ZBDDs), Minato [9]. An OBDD is a BDD where any non-terminal vertex pair $\{v, low(v)\}$ and $\{v, high(v)\}$ obeys the variable ordering condition $index(v) < index(children(v))$. A ROBDD is an OBDD where there is no vertex v with $low(v) = high(v)$, nor any pair $\{v, u\}$ such that the sub-graphs rooted in v and in u are isomorphic. In order to get a ROBDD from an OBDD, it is necessary to perform the following:

1. Eliminate all the redundant vertices whose edges point to the same vertex.

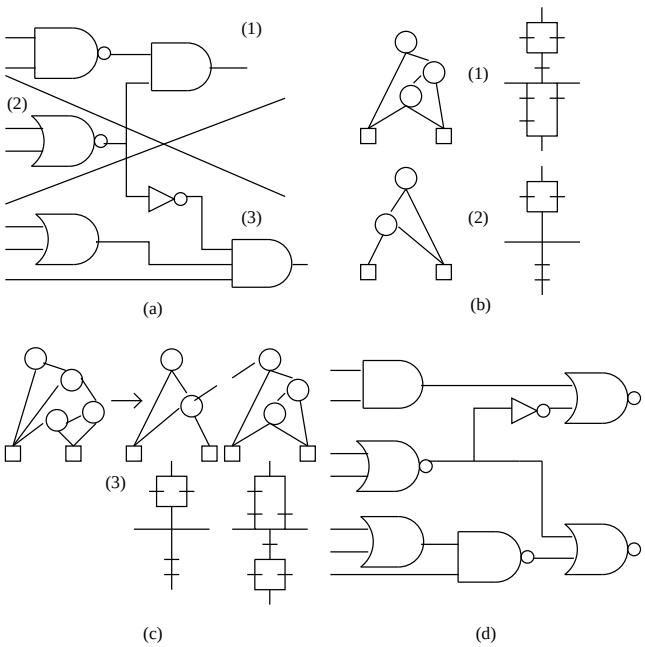

Figure 1: Solving the function in Example 1.

2. Share all the isomorphic sub-graphs.

2.2 Terminal Suppressed BDDs

Terminal suppressed BDDs (TBDDs) are briefly presented in [3, 4]. TBDDs are appropriate for CMOS technology mapping because a TBDD will always match a SCCG and the transistor topology of the PMOS and NMOS transistor networks can be obtained by graph isomorphism with S0 and S1 arcs, respectively.

A TBDD is defined as a BDD that has the following four properties:

1. Only the $S0$ arcs connect the 1-terminal vertices, so they are suppressed.
2. Only the $S1$ arcs connect the 0-terminal vertices, so they are suppressed.
3. All arcs having the same tail are either $S0$ or $S1$.
4. There is a path that passes through all non-terminal vertices (vertex precedence condition).

In brief, if a BDD obeys these four conditions, then it is possible to associate CMOS transistors directly in a one-to-one association with TBDD arcs to obtain SCCGs.

Example 1.

The stages of technology assignment with TBDDs using a virtual library of SCCGs is shown in Figure 1. The input is a Boolean netlist of simple gates that is broken up into logic cones, where each logic cone is connected to one output. In Figure 1a, the netlist is broken into three logic cones numbered (1), (2), and (3). TBDDs are constructed to represent the given logic cones. The TBDDs for (1) and (2) are shown in Figure 1b, where the TBDDs can be mapped directly into a SCCG, i.e. TBDD arcs are associated to transistors on the SCCG. There is a limitation to the number of serial arcs allowed in each BDD (number of serial transistors in a SCCG). This is shown in Figure 1c, where the number of serial arcs is greater than four, thus the TBBD is cut into two separate TBBDs. The final result is shown in Figure 1d, with a reduction of transistors from the original netlist of 30 transistors to the new netlist with 22 transistors.

3 Complex Gate Minimization

In this section, a SCCG minimization algorithm is presented for single output, completely specified functions. Note, that this method has been extended to multi-output, multi-level functions, realized in limited complex gates using a method called Dissected Pairs [2].

Complex Gate Minimization Algorithm.

1. Find the complement $\overline{f}$ of function f. This reduces the size of the function since an inverter is not needed to make the output of the SCCG positive.
2. Search through all possible minterms in $\overline{f}$. If a minterm has only one prime implicant covering it, the implicant is essential. When this occurs, the minterm and all minterms included in the essential prime are removed from $\overline{f}$ by changing their values to don't cares. The essential implicant is placed in the set MI($\overline{f}$). This is implemented in COMPLEX by using a BDD to represent $\overline{f}$.
3. The remaining set of minterms is a set that has multiple primes covering them, and thus, the primes are non-essential. All possible primes for a minterm are placed in a *prime implicant covering table*. The prime covers found from this covering table, and from MI($\overline{f}$) produce a set of sufficient implicants that represent $\overline{f}$.

4. For each sufficient implicant, store the value in terms of all positive variable terms, or *kernels*. A kernel is defined as the set of positive literals in a given implicant. For example, given the implicant $x_1\overline{x_2}x_3\overline{x_4}$, the kernel is x_1x_3.

5. Combine all sufficient implicants that have the same kernel. For example, given the implicants with kernel x_1, $x_1\overline{x_2}\ \overline{x_3}$ and $x_1\overline{x_2}\ \overline{x_4}$, the implicants can be combined as $x_1(\overline{x_2}(\overline{x_3}+\overline{x_4}))$. Or it can be placed in the *initial complex implicant notation* of $(x_1)\overline{(x_2+x_3x_4)}$.

6. Each sufficient implicant, in *initial complex implicant notation*, is placed in the set UF($\overline{f}$). The set UF($\overline{f}$) is then added to the *complex implicant covering table* which is used to find the SCCG with the "highest quality". This covering table is composed of positive kernels from the best cover as columns, and complex implicants that correspond to the positive kernels as rows.

7. Do an item by item *combine* of all the elements in the set UF to create *secondary complex implicants*. The combination of two implicants is done through a Boolean factorization method. In this method, a pair of implicants is combined by ORing the two kernels and ANDing the terms included in the OFF-terms of each implicant. For example, combining $x_4\overline{(x_1+x_3)}$ and $x_1x_3\overline{(x_4)}$, ORing the kernels gives $(x_4+x_1x_3)$ and ANDing the terms in the OFF-terms gives $\overline{(x_4(x_1+x_3))}$. Thus, the newly created *secondary complex implicant* is $(x_4+x_1x_3)\overline{(x_4(x_1+x_3))}$. There are cases, however, when this factorization method will not work. This occurs when there isn't a negative term that can cover all the OFF-terms and none of the ON-terms in both of the implicants. Thus, this factorization is only executed if the secondary complex term is equivalent to the union of the initial complex terms from which it was created.

This procedure is recursive, and will complete when there are no new combinations possible. (This happens either when all possible kernels have been combined or when the elements that remain cannot be combined.) At each level of recursion place the new set of *secondary complex implicants* in the *complex implicant covering table*.

8. Solve the *complex implicant covering table* by evaluating the cost of each solution as a weighted function of a set of parameters. These parameters

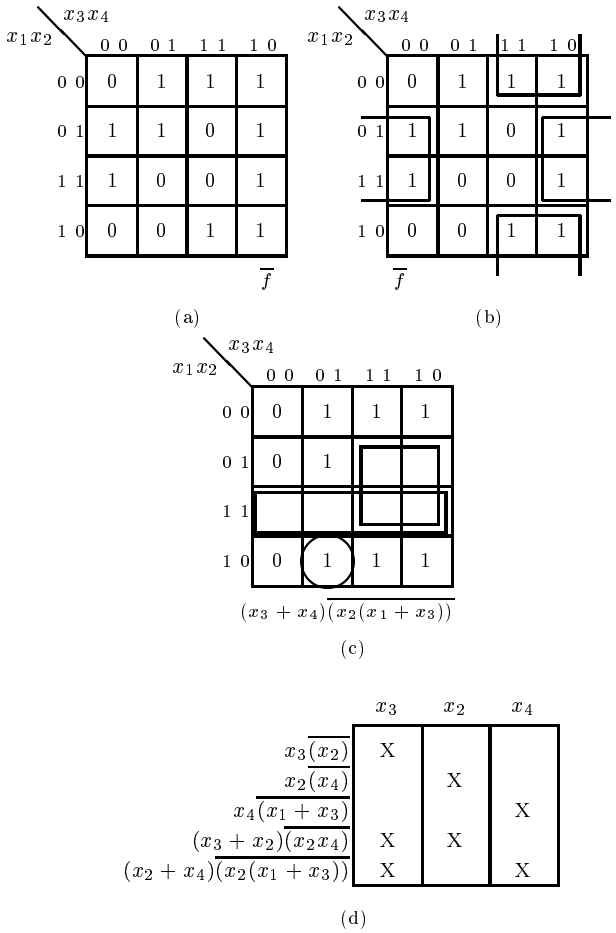

Figure 2: Solving the function in Example 2.

are defined throughly in [1], but are simply a set of parameters used to influence respective design parameters that can be extracted from TROPIC. This means, select a subset of rows that covers all columns and minimizes the weighted cost function, of the form $\sum_{i=1}^{9}\alpha_iP_i$ (there are 9 parameters to evaluate).

9. End. (Possibly repeat all steps using f instead of $\overline{f}$, because a reduced set of sufficient implicants will lead to a better solution.)

Example 2.

Given the function $\overline{f}(x_1,x_2,x_3,x_4) = \sum(1,2,3,4,5,6,10,11,12,14)$, shown in Figure 2a. (2) Find all the essential implicants of $\overline{f}$. The minterms 1011 and 1100, produce the respective essential primes, $\overline{x_2}x_3$ and $x_2\overline{x_4}$, as shown in

Figure 2b. (3) From the remaining two minterms the cover $\overline{x_1}\ \overline{x_3}x_4$ is selected. (4) The kernels for $\overline{f}$ are $\{x_3, x_2, x_4\}$. (5) There are no implicants with the same kernel, so nothing needs to be done here. (6) Initially, the set UF contains the elements: $(x_3)\overline{(x_2)}$, $(x_2)\overline{(x_4)}$, $(x_4)\overline{(x_1 + x_3)}$ which are placed in the *complex implicant covering table* shown in Figure 2d. (7) By combining the elements in UF, the *secondary complex implicants* are found as $(x_2 + x_3)\overline{(x_2x_4)}$ and $(x_2 + x_4)\overline{(x_4(x_1 + x_3))}$. Notice, that the kernels x_3 and x_4 could not be combined. In this case, if x_3 and x_4 are combined the resulting *secondary complex implicant* would be $(x_3 + x_4)\overline{(x_2(x_1 + x_3))}$ and is shown in Figure 2c. Thus, this does not cover the same set of minterms as the *initial complex implicants*.

(8) The *complex implicant covering table* is shown in Figure 2d. The resulting minimized network is given by the function
$\overline{f} = (x_2 + x_3)\overline{(x_2x_4)} + (x_4)\overline{(x_1 + x_3)}$.

4 Results

After running COMPLEX on a number of test functions, the following set was chosen because of their complexity. That is, functions that we feel are a fairly good test of COMPLEX. Most of the functions are eight variable functions, the only exception is the parity function of four variables. The reason for not using an eight variable representation is that COMPLEX generates a huge covering table, 128 kernels by 1000 rows. Even though many of the solutions are exactly the same, COMPLEX is not able to handle this size of covering table, yet. Note, the methodology of this algorithm is 2-level synthesis, and our goal is to only use this method on small functions. Thus, in the future, using it to evaluate functions that have been minimized by using other multi-level methods. This is the reason for evaluating the COMPLEX program on small functions, and does not cause us that much concern that COMPLEX is not able to minimize the eight variable parity function.

The results were generated by running COMPLEX on a test function and passing the best result of the function to the program TABA/TROPIC. For comparison, SIS, using a multi-level minimization method, was also used as an input to the TABA/TROPIC system. The results are shown in Table 1. In the table, the columns under each program are 3 of the 9 parameters used to evaluate the weighted cost function of COMPLEX. Parameter 1 or P1 is the total number of transistors. It is assumed that the overall number of transistors is a first order estimator for both area and power. Parameter 2 or P2 is the total number of SCCGs, this parameter is also related to both area and power. Parameter 5 or P5 is the number of serial SCCGs. This parameter is related to the depth of the net list, and its speed. Delay is the estimated timing delay, in nano-seconds, of the resulting function after layout. The other parameters are not shown because the information found within each of these parameters does not give values that can be used for comparison. The logic net-list generated by COMPLEX and SIS is converted to a transistor net-list with the number of serial transistors set to a maximum of four serial transistors [3, 4]. This transistor net-list is then used as the input to TABA/TROPIC.

Table 1: Comparison of results

	SIS multi-level				COMPLEX			
	P1	P2	P5	Delay	P1	P2	P5	Delay
ch22f0	40	9	5	1.88	40	3	3	2.14
ch47f0	94	17	9	4.58	78	6	4	2.65
or_and_chain	28	6	7	2.05	22	4	5	1.75
substr1	54	10	6	2.06	46	6	5	2.04
parity(4 var)	52	10	5	1.90	66	7	5	2.33
ch30f0	66	12	7	3.62	58	5	5	2.63
ch74f1	120	20	10	4.66	82	8	5	3.07
modulus2	96	18	8	3.10	76	10	5	2.70
rnd_m10	148	27	9	3.25	160	21	7	3.68
pal	160	28	7	2.84	320	36	10	6.06

The results show that COMPLEX performed very well compared to the multi-level minimization method SIS. The only function that COMPLEX didn't perform well on was the *pal* function which is a completely symmetric function, much like parity. This is expected since SIS should always beat COMPLEX on these type of functions. Also, for the function *rndm10*, COMPLEX returned a larger transistor count but the number of SCCGs was smaller.

5 Conclusion

A technique to minimize two-level networks from complex gates has been discussed.

A new factorization operation was introduced that finds good secondary complex implicants without creating all possibilities, and thus leads to "high quality" results. However, contrary to the method presented in [1, 2], we are not able to prove that the exact minimum of the first or second level is achieved. One method of improving this approach would be to repeat the algorithm for any set of prime kernels from a minimal prime cover. The comparison of the approximate and exact methods should be performed.

Moreover, the method assumes a two-level circuit of complex gates. Therefore, it is not competitive on large multilevel circuits with high EXOR components, such as parity. The method should be used for "high quality" minimization of functions in a window moved through a multi-level network created with EXOR synthesis methods. If no cost improvement is obtained for certain locations of the window, the original circuit remains and the window is moved to a new location.

We have extended the concept of a dissected pair [1] from a pair of a minterm and a false minterm to a pair of arbitrary one-cube and zero-cube. This permits applying the methods to larger functions, represented as arrays of arbitrary cubes. In particular, minimal implicants and minimal implicates [2].

References

[1] A. Reis, C.Files, M. Perkowski, M. Robert, and D. Auvergne, "Introduction to Exact Minimization of Networks with Complex Gates for High-Performance Layout Generator "TROPIC"" *Non published report*, LIRMM, Montpellier, France, April 1996.

[2] C. Files, R. Tinder, A. Reis, and M. Perkowski, "Dissected Pairs, Minimal Implicants and Synthesis of Minimum Multi-Level Multi-Output Networks from Complex Gates," Continuation of this paper, in preparation. Includes multi-level, multi-output design and extensions to power minimization and testability.

[3] A. Reis, M. Robert, D. Auvergne, and R. Reis, "Associating CMOS transistors with BDD arcs for technology mapping," *IEE Electronics Letters*, Vol. 31, No.14, 6th July 1995, pp. 1118-20.

[4] A. Reis, M. Robert, D. Auvergne, and R. Reis, "Extensive using of CMOS complex gates with terminal suppressed BDDs" Journal of the Brazilian Computer Society, No. 2, Vol. 2, November 1995. pp.63-75.

[5] E. Detjens, G. Gannot, R. Rudell, A. L. Sangiovanni-Vinccentelli, and A.Wang, "Technology mapping in MIS," *Proc. of ICCAD,* Santa Clara, pp. 116-119, 1987.

[6] K. Keutzer, "DAGON: Technology Binding and Local Optimisation by DAG Matching," *Proc. of 24th DAC,* pp. 341-347, 1987.

[7] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 677-691, August 1986 .

[8] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," *Proc. of 27th DAC*, pp. 272-277, 1990.

[9] S. Minato, "Zero-Suppressed BDDs for set manipulation in combinatorial problems," *Proc. of 30th DAC*, pp. 40-45, 1993.