Analysis of Transistor Networks Generation

Leomar S. da Rosa Junior[1,2], Felipe R. Schneider[3], Renato P. Ribas[1,2], André I. Reis[3]

{leomarjr, felipers, rpribas}@inf.ufrgs.br , are@nangate.com

[1] Nangate Research Lab – Instituto de Informática – Universidade Federal do Rio Grande do Sul, Brazil
[2] Programa de Pós-Graduação em Microeletrônica – Universidade Federal do Rio Grande do Sul, Brazil
[3] Nangate Inc. – Menlo Park, CA, USA

ABSTRACT

This paper presents a comprehensive investigation of how transistor level optimizations can be used to increase design quality of CMOS logic gate networks. Different properties of transistor networks are used to explain features and limitations of previous methods. We describe which figures of merit, including the logical effort, affect the design quality of a cell transistor network. Further, we propose and compare two different approaches that generate transistor network with guaranteed theoretical minimum length transistor chains, showing it reduces significantly the logical effort of the networks.

1. INTRODUCTION

VLSI design has firmly established a dominant role in the electronics industry. Automated tools have held designers to manipulate more transistors on a design project and shorten the design cycle. In particular, logic synthesis tools have contributed significantly to reduce the cycle time. In full-custom designs, manual generation of transistor netlists for each functional block is performed, but this is an extremely time-consuming task. In this sense, it becomes comfortable to have efficient algorithms to derive transistor networks automatically.

Furthermore, some logic synthesis tools are extensively based on using Binary Decision Diagrams (BDDs) [1]. Classical BDDs based on Shannon's decomposition naturally correspond to circuits built by Shannon's decomposition algorithm. Therefore, not only BDDs can be used as compact and convenient representation of logic functions, but as a structure for direct synthesis of logic cells and circuits. Most methods for generation of transistor networks from BDDs use a non-disjoint pull-up/pull-down plane. This is the case of the methods presented in [2-7]. Some alternative methods presented in [8-11] derive disjoint planes: a pull-down composed of NMOS switches and a pull-up composed of PMOS switches. The drawback of these methods is the requirement of a special kind of BDDs with a serial/parallel structure. This way, the methods in [8-11] are not applicable to the widely used ROBDDs (Reduced and Ordered BDDs). The approaches in [12] and [13] use disjoint planes with ROBDDs, with subsequent simplifications. The simplifications performed by [12] are based on permissible functions, while the simplifications proposed in [13] are based on the unateness property of logic functions.

In this paper, we describe two different methods for transistor network generation that respect the lower bound for the number of serial connected switches in a given logic cell. In addition, six kinds of CMOS network topologies are compared using some figures of merit, including the logical effort [14] for the cell.

2. BDDS AND TRANSISTOR NETWORKS

There are several methods for deriving transistors networks from BDDs in the literature. Some of them are closely related to the use of multiplexer-based logic. These methods may require additional area because the number of necessary switches to implement a multiplexer is noticeably expensive. Other approaches use direct switches association to BDD arcs to build the logic cells. These solutions are more feasible since the total number of transistors required for the implementation can be drastically reduced if compared to multiplexer-based solution. In this context, the basic action when deriving a transistor network from a BDD is to associate a controlled switch to each arc of a BDD node. This concept is illustrated in Fig. 1, which shows a BDD node and four possible ways to associate transistor switches: CMOS pair, NMOS only, PMOS only and mixed PMOS/NMOS.

BDD arcs connecting terminal nodes (0-terminal and 1-terminal) may be connected to VDD or GND. Usually, NMOS transistors are associated to BDD arcs leading to 0-terminal node, while PMOS transistors are associated to arcs leading to 1-terminal node. The main reason for this sort of association is to guarantee a good conduction from the power sources (VDD and GND) to the internal nodes. The exception occurs in the NPTL-like logic styles (NMOS Pass Transistor Logic), which are composed of NMOS transistors only. Fig. 2 illustrates this concept, where a NPTL network (Fig. 2.b) is derived from a BDD (Fig. 2.a).

Drain inputs may be used to reduce the number of transistors in a cell derived from a BDD. When a node of a BDD represents a function corresponding to a single literal, like a or $\overline{a}$, it is not necessary to use any switch to implement the node. The electrical node corresponding to the literal may be connected directly to the drains of the transistors corresponding to arcs in the BDD that point to the literal node. This is illustrated in Fig. 2.c, where the node C generates a drain input in the network.

The drawback of drain inputs is that the fanin capacitance they add to the driving node is not constant. That means that the capacitance seen by the driving node is variable and depends on the values of other inputs controlling the transistor in a drain

input. This may cause several electrical problems resulting from the different slopes that will occur due to the variability in the fanin capacitance. Well defined input capacitances is a requirement for commercially available static timing analysis tools.

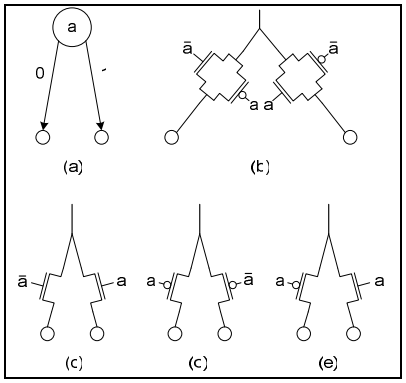

Figure 1. BDD node and associated switches.

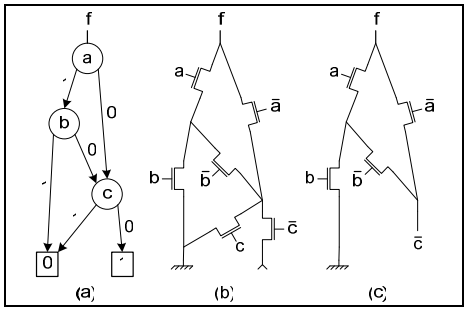

Figure 2. Drain inputs in a network derived from a BDD.

A common mistake about drain inputs is the assumption that the use of drain inputs would shorten the paths from the power sources (VDD or GND) to the output of the network. This is a pitfall, in the sense that by using drain inputs the signal driving the drain is not a power source, but a logic signal. A logic signal is always distant from the power sources by at least one switch responsible for connecting or disconnecting it to the power sources, as it may assume both logic values. This way, the number of serially connected switches between the power sources (VDD or GND) and the output of the network is not reduced by using drain inputs.

When a PTL (Pass Transistor Logic) transistor network is built with a pair of PMOS and NMOS transistors associated to BDD edges, there is the possibility to derive disjoint networks. The procedure is straightforward, as it is illustrated in Fig. 3. To build the pull-up plane PMOS transistors are associated to BDD arcs (Fig. 3.b), while to build the pull-down plane NMOS transistors are used (Fig. 3.c).

As an effect of using disjoint planes, the number of transistors into a logic cell remains the same, but the number of nodes increases. As a result, the internal capacitance per node is smaller. One of the advantages of using disjoint planes is the reduction of the capacitances of internal nodes. Another important point is that, as the number of nodes increases while the number of elements remains the same, the number of connections to be performed among elements is reduced. This effect is visible in Fig. 3, where it is clear that the use of disjoint planes implies on the reduction of connections between PMOS and NMOS active area. This is a benefit from a layout point of view. Finally, by dividing an internal node into two different ones, the switching activity probability is reduced, since there is the possibility that just one of them is switched. This is a benefit from a power consumption point of view. Apart from that, by using disjoint planes, it is possible to take advantage of the reductions presented in the next section in order to optimize the transistor networks.

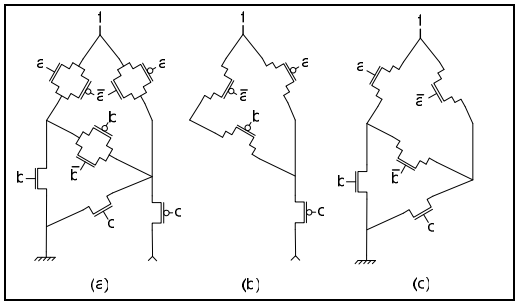

Figure 3. Network derived from a BDD using disjoint pull-up and pull-down networks.

3. MINIMUM TRANSISTOR CHAIN NETWORKS

Two methods for generating transistor networks with minimum transistor chains are discussed bellow.

3.1. BY SOP FACTORIZATION

It is a straightforward process to generate a transistor network from a SOP or from a factored form. In order to generate transistor networks coping with the lower bounds presented in [15], it is sufficient to generate minimum cube literal SOPs for the on-set and for the off-set. This will guarantee minimum length chains. Factorization is applied after computing the minimum cube literal cost SOP, as a second step to reduce the overall number of transistors by sharing some transistors among different branches. Factorization also contributes to minimize the logical effort of the input controlling the shared transistor. As the lower bound is dependent on using a minimum cube literal cost SOP, the factorization must be algebraic only. The use of Boolean factorization could lead to a different cover with non minimum cube literal cost. Boolean factorization could be used if the maximum length transistor chain is verified to remain the same after the factorization. Fig. 4 illustrates this process.

3.2. FROM BDDs

The method to generate disjoint transistor networks from BDDs is based on the methods presented in [13,16]. The basic

procedure is presented in Fig.5. The method performs, as a first step, the straightforward generation of PU and PD networks from BDDs. As a second step, short and open circuits are inserted for unate nodes, leading to the removal of some transistors, as described in [13,16]. The main difference in the algorithm proposed here is on step 3. The methods proposed in [13,16] will not perform all the short and open circuit simplifications, if sneak paths are introduced.

The algorithm presented here adopts a different strategy, in order to reduce the length of longest transistor chain to achieve the lower bound. It duplicates some nodes of the BDD to avoid the introduction of sneak paths and force the achievement of the lower bounds. The duplication of nodes in the BDD does not necessarily imply in the increase in the number of transistors, as some transistors could be removed and substituted by short or open circuits. Sometimes the lower bounds are achieved even without using all the possible unateness reduction cases, i.e. it is possible to not apply all short and open circuit simplifications and still achieve the lower bound. The reordering of the BDD in step 4 is very rarely used, as most of the times the algorithm achieves the lower bound without reordering. The effect of duplicating BDD nodes and applying unateness simplification is to make the paths in a BDD correspond to prime cubes, as BDD would normally have paths corresponding to non-prime cubes. Notice that the BDD ISOP algorithm is able to generate a prime cover from a BDD, but if this is used to generate the transistor networks it would result in the method described in the previous section (without the factorization step). Although the procedure proposed here is rather heuristic, the execution time is not prohibitive.

Step 1: Generate a minimum cube literal cost for on-set.

Step 2: Perform algebraic factorization (can be Boolean if LB is checked)

Step 3: Derive transistor network for PU

Step 4: Generate a minimum cube literal cost for off-set.

Step 5: Perform algebraic factorization (idem step 2).

Step 6: Derive transistor network for PD

Step 7: If one (or both) of the networks has a dual respecting the LB, use both planes from the same equation (chose equation with less transistors). Otherwise use one network from each equation.

Figure 4. Algorithm for generation of minimum transistor chain networks by SOP factorization.

Step 1: Generate PU and PD from BDD.

Step 2: Apply short and open circuits for unate nodes to remove transistors.

Step 3: If the circuit has sneak paths, duplicate some BDD nodes and go to step 1.

Step 4: If the circuit does not respect the lower bounds, reorder the BDD and go to step 1.

Step 5: Deliver network

Figure 5. Algorithm for generation of minimum transistor chain networks from BDDs.

4. RESULTS

A comparative experiment to show how the topology of transistor networks influences the logical effort of logic gates was performed. Six different network topologies were compared. These methods are described below.

CSP1 – CMOS series-parallel. Both on-set and off-set equations are derived and factorized, the one that respects the lower bound and give the smaller transistor count is chosen whenever possible. When there is no solution respecting the lower bounds, a solution is chosen to respect the LB in the PU network.

CSP2 – Same as CSP1, except that when there is no solution respecting the lower bounds, a solution is chosen to minimize the series transistor difference in both networks with respect to the LB.

NCSP – Non complementary series-parallel solution, using one plane from the on-set equation and one plane from the off-set equation enforce the achievement of the lower bounds. The CSP version is used whenever it respects the lower bounds. Algorithm in Fig. 2.

CMOS from BDDs – Straightforward generation of PU and PD planes from BDDs.

Optimized CMOS from BDDs – Generate pull-up and pull down planes from BDDs and simplify them using unateness properties to insert short and open circuits. Do not apply any possible optimization that introduces sneak paths. See [13,16] for details.

CMOS LB from BDDs – Force all the possible unateness reductions, duplicating nodes to guarantee the achievement of the lower bounds. Algorithm in Fig. 5.

The set of evaluated functions include all the 3982 P-classes representing the set of non-constant 4-input logic functions. For all the 3982 target functions, the six network types described above were generated. Results are reported in Table 1. The data for each generation method are described in one column. For each method, the sum of the total number of transistors, length of longest transistor chain for pull-up ($\sum$PU), length of the longest transistor chain for pull-down ($\sum$PD), logical effort (average per cell input), number of functions that do not respect the lower bounds and the number of unfeasible functions is shown. The CMOS lower bound from BDDs is a clear winner for total number of transistors, this happens because even if some nodes are duplicated, it is possible to remove some transistors, which compensates the duplication with advantages. The two methods presented in Fig. 4 and Fig. 5 respect the lower bound, so these methods have equal $\sum$PU lengths and $\sum$PD lengths, as shown in Table 1. However the total number of transistors is smaller for the CMOS LB from BDDs, which explains the advantage this method also has in terms of logical effort. Notice that the CSP1 respect the LB for the PU, as expected from its criterion of choice. The CSP2 method presents a reduced $\sum$PD length through a tradeoff that increases the $\sum$PU length compared to CSP1. This tradeoff results in an advantage of CSP2 with respect to CSP1, when logical effort is considered, as it can be observed in Table 1. All methods produce functions not respecting the lower bounds, with exception of NCSP and CMOS LB from BDDs. The CMOS from BDD method is the one that produces the highest number of functions not respecting

Table 1. Comparison of six different methods for cell level transistor network generation.

	Logic					
Figures of merit	CSP1 [8]	CSP2 [8]	NCSP [5]	CMOS from BDDs [3,7]	Optimized CMOS from BDDs [3,7]	CMOS LB from BDDs
∑ # transistors	75530	75456	75889	76774	73438	72307
∑PU length	11954	13084	11954	15538	14227	11954
∑PD length	17009	15931	14242	15538	15321	14242
Aver. logical effort	4.54	4.37	3.83	4.35	4.07	3.68
#f not respecting LB	2312	2312	0	3148	2373	0
# of unfeasible f	1546	791	0	0	0	0

the lower bounds. However, all the functions it produces have at most 4 transistor in series, and therefore they are considered feasible with a single cell. The only methods to produce networks with more than 4 transistors in series are CSP1 and CSP2. This is a result of using dual networks, which will result in excessive number of transistors in series when making a dual of a network that has many transistors in parallel. We have also observed that for networks with the same chain lengths, the one with a smaller transistor count is the winner.

5. CONCLUSIONS

We have shown that the logical effort to compute a function depends on the chosen topology. The use of minimum transistor lengths while reducing the overall number of transistors reduces significantly the logical effort of the resulting networks, a reduction of 15 to 19% is obtained when compared to CMOS series-parallel. A reduced logical effort is directly transposed to a reduction in the best achievable delay of a circuit.

6. REFERENCES

[1] C.Yang and M.Ciesielski. BDS: a BDD-based logic optimization system. IEEE Transactions on CAD, Volume 21, Issue 7, July 2002 Page(s):866 – 876.

[2] P. Buch, A. Narayan, A.R. Newton and A. Sangiovanni-Vincentelli. Logic synthesis for large pass transistor circuits. ICCAD 1997. Pages:663 – 670.

[3] C. Scholl and B. Becker. On the generation of multiplexer circuits for pass transistor logic. DATE 2000. Pp:372 – 378.

[4] P. Lindgren, M. Kerttu, M. Thornton and R. Drechsler. Low power optimization technique for BDD mapped circuits. ASP-DAC 2001. Pages:615 – 621.

[5] R.S. Shelar and S.S. Sapatnekar. An efficient algorithm for low power pass transistor logic synthesis. ASP-DAC 2002. Pages:87 – 92.

[6] M. Avci and T. Yildirim. General design method for complementary pass transistor logic circuits. Electronics Letters, Vol.: 39 , Number: 1 , 9 Jan. 2003. Pages:46 – 48.

[7] R.S. Shelar and S. Sapatnekar. BDD decomposition for delay oriented pass transistor logic synthesis. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Volume 13, Issue 8, Aug. 2005 Page(s):957 - 970.

[8] A.I. Reis, M. Robert, D. Auvergne and R. Reis. Associating CMOS transistors with BDD arcs for technology mapping. Electronics Letters, v. 31, n. 14, p. 1118-1120, 1995.

[9] S. Gavrilov, A. Glebov, S. Pullela, S.C. Moore, A. Dharchoudhury, R. Panda, G. Vijayan and D.T. Blaauw. Library-less synthesis for static CMOS combinational logic circuits. ICCAD 1997. Pages:658 – 662.

[10] C. L. Liu and J. A. Abraham. Transistor level synthesis for static CMOS combinational circuits. VLSI, 1999. Proceedings of the Ninth Great Lakes Symposium. Page(s): 172 - 175.

[11] S. Gavrilov and A. Glebiy. BDD based circuit level structural optimization for digital CMOS. Proceedings of MALOPD 1999. Pages: 45 – 49.

[12] M. Kanecko and J. Tian. Concurrent cell generation and mapping for CMOS logic circuits. ASPDAC97. Pp. 247– 52.

[13] R.E.B. Poli, F.R. Schneider, R.P. Ribas and A.I. Reis. Unified theory to build cell-level transistor networks from BDDs. SBCCI 2003. Pages: 199 – 204.

[14] I.Sutherland, B.Sproull and D.Harris, *Logical Effort: Designing Fast CMOS Circuits*, Morgan Kaufmann, 1999.

[15] F.R.Schneider, R.P.Ribas, S.S.Sapatnekar, A.I.Reis, "Exact lower bound for the number of switches in series to implement a combinational logic cell", *ICCD 2005*, pp. 357 – 362.

[16] L.S. Rosa Junior, F. Marques, T.M.G. Cardoso, R.P.Ribas, S.S.Sapatnekar, A.I.Reis, "Fast Transistor Networks from BDDs", *SBCCI 2006* proceedings.