A GRAPH-BASED SOLUTION FOR DUAL TRANSISTOR NETWORK GENERATION

[1]Vinicius Callegaro, [1]Leomar S. da Rosa Jr, [2]André I. Reis, [1]Renato P. Ribas

[1]Instituto de Informática – UFRGS, Porto Alegre, Brazil
[2]Nangate Inc., Sunnyvale, CA, USA
{vcallegaro, leomarjr, rpribas}@inf.ufrgs.br, are@nangate.com

ABSTRACT

Transistor network optimization is of special interest for efficient digital circuit design. In this context, a graph-based solution for dual-network generation is proposed. The algorithm is able to generate series-parallel and bridge topologies. Experimental results demonstrate significant CMOS gate design improvement in terms of the total number of transistors, when combining dual-network generation presented here with BDD-based technique.

1. INTRODUCTION

CMOS design is currently the most used and well established logic style applied by the modern industry of microelectronics. This standard, also known as complementary series/parallel (denominated here as CSP) logic style, is an arrangement of series and parallel transistors in two separated logic planes: pull-up and pull-down ones. Fig. 1 illustrates the CSP logic style. The major advantage of the CSP logic is low sensitivity to noise, good performance and low power consumption [1].

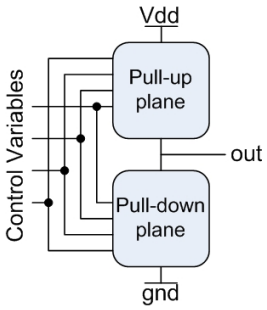


Figure 1 – Standard CMOS logic style ('CSP').

The transistor arrangement is built in a straightforward two step procedure. In the first step, a given logic function description, composed of AND and OR operators, is translated into an equivalent elements interconnection. To generate the pull-down NMOS plane, all AND operations are treated as series interconnections, while OR operations are treated as parallel associations. In the second step, the dual logic plane is obtained through a topologically complemented implementation, where all series interconnections in the first logic plane are converted to parallel interconnection in the other logic plane. The same process is done for parallel interconnections, resulting in series arrangements. Remembering that the CMOS style provides a negate logic function. Fig. 2a presents a transistor network from a CSP logic style for the equation $f = !(a*d + a*c*e + b*e + b*c*d)$. It is possible to notice that the dual logic plane is obtained directly from a series/parallel complementary association.

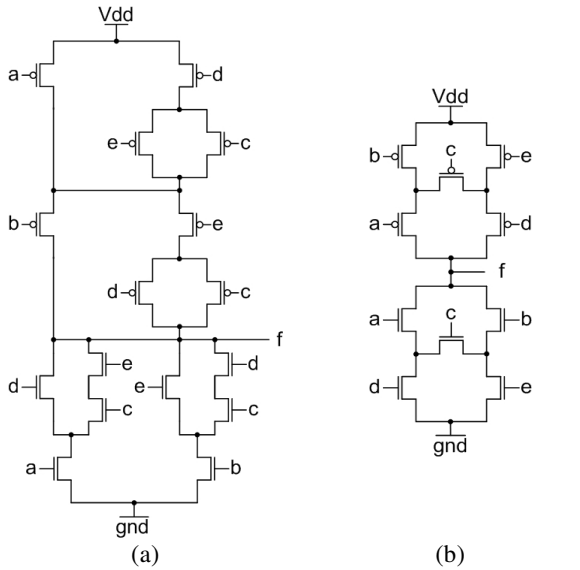


Figure 2 – Function $f = !(a*d + a*c*e + b*e + b*c*d)$, (a) CSP logic style, (b) bridge-based arrangement.

Recent works have investigated and demonstrated that different logic styles may deliver more efficient networks [2-4]. Some of these works show that circuit area, circuit delay, and circuit power consumption may be optimized if alternative logic styles are used instead of standard CMOS. In this context, some works explore the use of bridge transistor arrangements in order to minimize the transistor count in the circuits [5,6]. In such logic style, non-series parallel arrangements are done, introducing transistors that are neither in series nor in parallel with others. Fig. 2b illustrates a bridge-based implementation for the equation described before, where the transistors controlled by variable *'c'* are the responsible for a non-series/parallel arrangement. The drawback for generating this kind of network is that complementary series/parallel

operations cannot be used to obtain the dual plane, since there are some transistors that are neither in series nor in parallel with others. For that, it is necessary a dual-graph approach in order to achieve the network construction [1].

This paper presents a methodology to automatically generate dual logic planes for bridge-based arrangements. The graph theory is briefly explained and the proposed solution is presented to achieve two terminal dual networks.

2. TRANSISTOR NETWORK TO MULTIGRAPH

A transistor network description, i.e. Spice netlist, may be converted into a graph representation. In this work it is done because graph structures are easier to manipulate than Spice netlist textual descriptions. This way, all transistors are converted to edges, and the nodes of the network are converted to vertices. The *Vdd/Gnd* and *Output* nodes are kept as special vertices (this information is necessary in the proposed algorithm). In the graph theory point-of-view, the correct term for the graph obtained from this operation is multigraph.

The term multigraph refers to a graph in which multiple edges between nodes are permitted. Some references require that multigraph possess no graph loops [7], while other ones explicitly allow them [8]. For the purpose of this work, the multigraph will not possess graph loops, because in transistor networks there is no transistor connecting the same node. This situation would be a short-circuit in the network.

Another important concept to be understood is related to the graph planarity. A graph is planar if it can be drawn in a plane without graph edges crossing. This characteristic is required for the dual graph generation. Only planar graphs may deliver a dual graph [7].

In the literature, some algorithms to generate dual graph from a simple graph are available, as presented in [7]. In this approach, the 'exterior' face is converted to a single vertex in the dual graph. However, for the electrical engineering purpose, the exterior face must be considered as two vertices. For the purpose of this work these vertices are called exterior vertices. This assumption is necessary since electrical networks are composed by two terminal nodes. Fig. 3a illustrates a transistor network, and Fig. 3b shows a planar multigraph obtained from this network and its two exterior vertices dual graph (dotted lines).

In the case of generating a dual graph from a simple graph, all edges joining to the vertex *V1** (as illustrated in Fig. 3b) should be redirected to vertex *V2**. The vertex *V1** would be suppressed. Apart from that, it is important to notice that a simple graph does not allow multiple edges between two same vertices, like the edges joining *V1* and *V3* in Fig. 3b.

Since simple graph and multigraph are not sufficient to meet the requirements to generate a dual two terminal transistor network, this structure of multigraph with two exterior vertices is presented as an adequate solution.

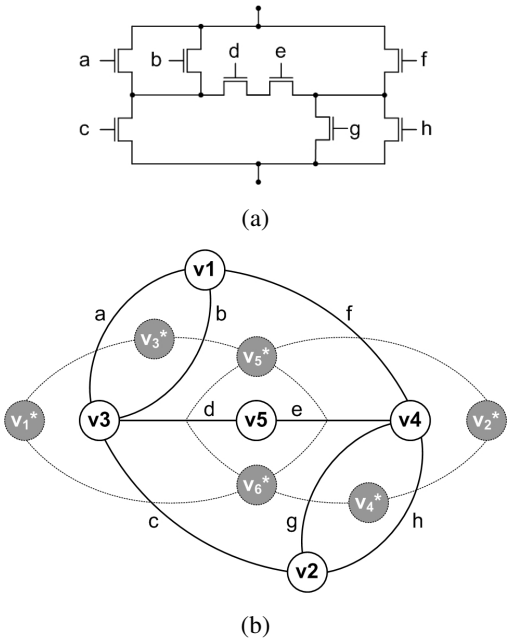


Figure 3 – Transistor network (a), and equivalent multigraph (b) representation and its dual (doted lines).

3. PROPOSED ALGORITHM

The input of the method is a Spice netlist description. This description is translated into a multigraph representation in order to be manipulated. The algorithm to generate the dual graph is performed in three main steps: compression, dual graph generation, and decompression.

3.1. Compression

In the compression step, all multiple edges joining the same two vertices (parallel edges) are merged in single edges. In the same way, the edges connecting vertices with 2-degree (series edges) are merged in single edges. Notice that this merge operation causes the suppression of the 2-degree vertex. This procedure is only allowed if the vertex is not a special vertex (*Vdd/Gnd* and *Output)*. It is done until no edge compressions are observed. This step is done to simplify the sequence of the algorithm. If the original transistor network is a series/parallel implementation, this compression will result in a graph with only one edge. It means that it is not necessary to use the next step to obtain the dual implementation, since the solution is trivial (as presented before, through series/parallel association). If the number of edges from the resulting compressed graph is different from one, it means the original transistor network is a non-series/parallel implementation. So, the next step of this algorithm is applied. Fig. 4b illustrates the graph compression step over the input multigraph described in Fig. 4a.

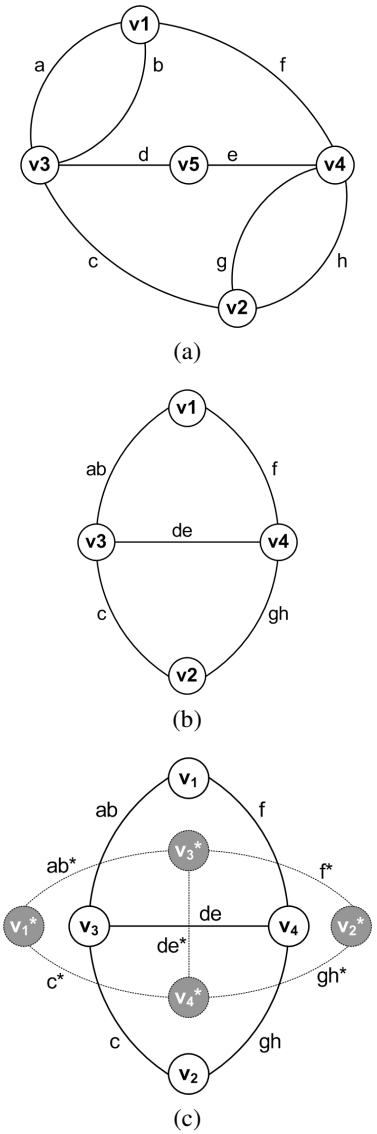


Figure 4 – (a) Input multigraph, (b) resulting simple graph after compression, and (c) dual graph obtained by applying the proposed algorithm.

3.2. Dual Graph Generation

At this point of the algorithm, all parallel edges were suppressed through the compression step, resulting in a simple graph. The dual graph generation consists in following sub steps:

- Cycle Detection and Faces Identification: a cycle is a closed walk in the graph, which is an alternating sequence of vertices and edges, beginning and ending at the same vertex. In this step, all cycles are detected. The regions defined by the cycles in a planar graph are referred as faces. The unbounded face is called exterior face [7]. In Fig. 4b the region defined by the vertices *V1*, *V3* and *V4* is a face, as well as the region defined by *V2*, *V3* and *V4* is another face. The 'exterior' face is the exterior region defined by *V1*, *V3*, *V2* and *V4*.
- Creating Exterior Vertices: two paths (*[ab, c]* and *[f, gh]*), between the special vertices (*V1* and *V2*), traversing the original exterior face are identified. These two paths define the boundary where the two external vertices for the dual graph will be created. In the graph theory this idea of two 'exterior' faces is not described, but it is necessary because these special vertices in the dual graph will be translated to *Vdd/Gnd* and *Output* nodes in the dual transistor network.
- Dual Creation: All faces in the graph that are not 'exterior' faces will receive a vertex. For two faces having an edge 'x' in common, the corresponding vertices are joined by an edge 'x*' crossing only this edge 'x'. This joining process is performed to all faces in the graph. The dual graph will be the collection of edges and vertices that were obtained after all this procedure. This is shown in Fig. 4c.

3.3. Decompression

In the decompression step, all compressed edges in the dual graph must be expanded. It is done performing the following rules:

- All edges in the dual graph that were created from a parallel compression in the original graph will be expanded in a series way. This process will reintroduce 2-degree vertices in the graph.
- All edges in the dual graph that were created from a series compression in the original graph will be expanded in a parallel way. This process will transform the simple graph in a multigraph again.

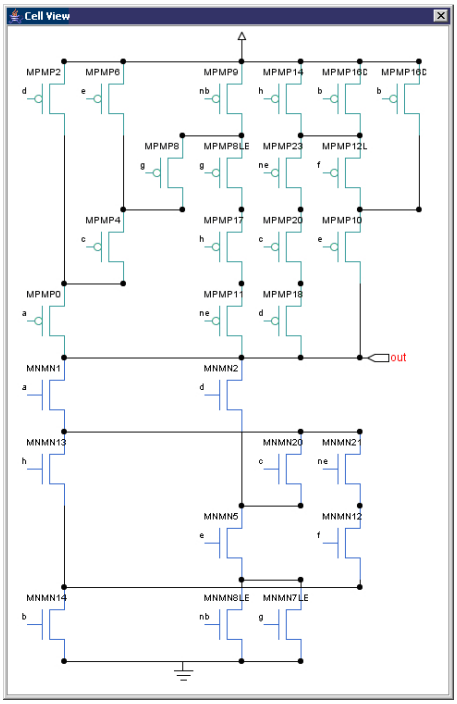
Notice that the decompression is done using the compression information stored in the first step of the algorithm. This way, the correctness of the dual graph expansion is guaranteed. The dual transistor network is obtained by a simple mapping from the multigraph to a Spice netlist.

4. RESULTS

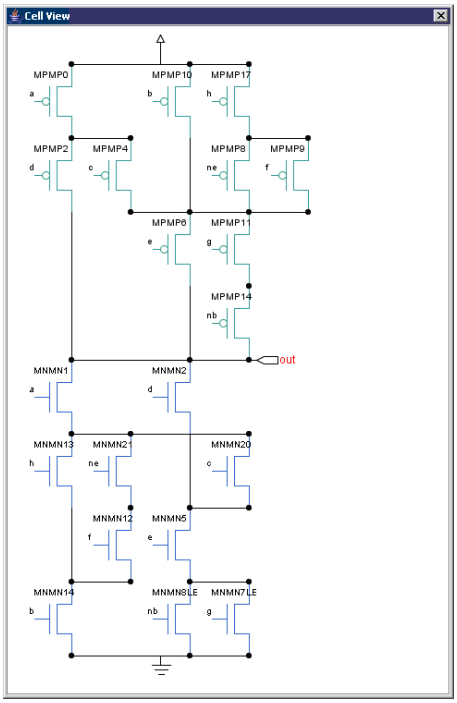
The proposed algorithm was implemented in Java language. To demonstrate the functionality of this algorithm, a Spice netlist for the function *f=!(a*b*!e*f+a*b*h+a*c*e*g+a*!b*c*e+b*c*d*!e*f+b*c*d*h+d*e*g+!b*d*e)* was generated using the BDD-based approach presented in [4], illustrated in Fig. 5a. The pull-down NMOS plane was used as reference to generate an alternative pull-up PMOS plane, according to the dual graph principle, described in this work. The result is observed in Fig. 5b.

The total transistor count is clearly minimized, since it is possible to generate the dual-network from the input transistor network plane that contains the smallest number of elements. This obtained network is logically equivalent

to the original network generated by [4]. In this case, a reduction of 26% in transistor count was verified.



(a)



(b)

Figure 5 – Transistor network for the logic function *f=!(a*b*!e*f+a*b*h+a*c*e*g+a*!b*c*e+b*c*d*!e*f+b*c*d*h+d*e*g+!b*d*e)*: (a) generated by BDD [4], (b) obtained using the proposed method.

It is important to notice that the solution proposed here does not generate transistor networks from an equation description, as it is done in state-of-the-art transistor networks generation methods. Instead, it uses a previously generated Spice netlist as input, and tries to achieve a smaller transistor count. For complementary series/parallel transistor networks the algorithm does not present any improvement, since the dual generation returns exactly the same networks.

5. CONCLUSIONS

A dual transistor network generation algorithm was presented. The algorithm is a graph-based solution that is able to generate dual networks for bridge-based arrangements. The results show the potential use of the algorithm, demonstrating that it could be used as a later optimization step to achieve efficient transistor networks. The elements count is minimized by exploring this approach, leading to logic cell implementation with small area overhead.

6. REFERENCES

[1] Weste, N.H.E. CMOS VLSI Design: A circuits and Systems Perspective. Third Edition, Pearson Education Inc, 2005.

[2] Poli, R.E.B.; Schneider, F.R.; Ribas R.P.; REIS, A.I. Unified Theory to Build Cell-Level Transistor Networks from BDDs. Symposium on Integrated Circuits and Systems Design 2003 (SBCCI'03), pp. 199 – 204.

[3] Schneider, F.R.; Ribas, R.P.; Sapatnekar, S.S.; REIS, A.I. Exact Lower Bound for the Number of Switches in Series to Implement a Combinational Logic Cell. International Conference on Computer Design 2005 (ICCD'05), pp. 357 – 362.

[4] da Rosa Jr. L.S.; Marques, F.S.; Schneider, F.; Ribas, R.P.; Reis, A.I. A Comparative Study of CMOS Gates with Minimum Transistor Stacks. Symposium on Integrated Circuits and Systems Design 2007 (SBCCI'07), pp. 93 – 98.

[5] Zhu, J.; Abd-El-Barr, M. On the Optimization of MOS Circuits. IEEE Transactions on Circuits and Systems, vol.40, n.6, pp. 412 – 422, June 1993.

[6] Kagaris, D.; Haniotakis, T. A Methodology for Transistor-Efficient Supergate Design. IEEE Transactions on VLSI Systems, vol.15, n.4, pp. 488 – 492, April 2007.

[7] Harary, F. Graph Theory. Perseus Books Group, 1994.

[8] Hartsfield, N. Pearls in Graph Theory: A Comprehensive Introduction. Second Edition, Academic Press, 1994.