Transistor reordering for low power CMOS gates using an SP-BDD representation

Dr. Alexey L. Glebov

NIISAPRAN
Academy of Sciences of Russia

Dr David Blaauw and Dr. Larry G. Jones

Semiconductor Systems Design Technology
Motorola, Inc

Abstract

In complementary CMOS circuits, the relative position or ordering of transistors in each gate has a significant impact on the power dissipation of the circuit. In this paper, we present a new approach that effectively optimizes the ordering of transistors in complex CMOS gates such that their power dissipation is minimized. The ordering algorithm uses a novel BDD representation that gives a canonical representation of both the structure and Boolean function of a CMOS gate. We present algorithms for extraction and manipulation of this BDD representation (called a SP-BDD). Using this representation, we present an approach that effectively optimizes the structure of complex gates. Experiments on a set of benchmark circuits, shows a reduction in the power consumption by 13% on average, and up to 17% for complex gates with a high reordering potential.

1. Introduction

Complementary CMOS circuits constitute a large portion of current VLSI designs. Consequently, the structure of CMOS gates has a significantly impact on the total power dissipation of todays designs, and needs to be examined for power optimization. The relative positioning of transistors in a complementary CMOS gate has a significant impact on the power dissipation of the gate. Reordering or restructuring of the transistors in a complex gate changes the capacitance that is charged/discharged when the gate output switches. It also affects which paths in the gate switch, and at which point in the path. Furthermore, the gate structure affects the delay through the gate. Finding the gate structure with a transistor ordering that minimizes the power dissipation, but meets the required delay constraint is, therefore, a non-trivial problem for complex gates.

In [1-3], a number of algorithms for transistor reordering are presented. The approaches, however, are limited to reordering along simple paths of transistors connected in series. These approaches target the pin re-assignment problem for library cells. For complex gates, pin re-assignment is often limited to a few inputs since the structure of the gate must remain constant. Only the differences in arrival times and switching frequencies of the swappable signals can be exploited and the improvement in power dissipation is unnecessarily limited. For instance, for the complex gate shown in Figure 1a, only inputs F and G can be switched by pin-reassignment. While pin-reassignment is an effective means of optimizing circuits when a fixed library is already in place, more improvement in the power consumption of circuit is obtainable if the topology of the circuit is not fixed and the circuit structure is optimized.

In the approach presented in this paper, a complete reordering of the transistors is considered where the structure of the circuit is modified. Restructuring of a gate has two advantages over input pin re-assignment. First, gate restructuring allows all inputs to be reordered to exploit the differences in arrival time and switching frequencies of all inputs. Second, restructuring the gate changes the capacitance of the nodes along the conducting paths in the gate. This can be used to minimize the total capacitance that is charged/discharged along frequently switching paths of the gate. Figure 1b shows the gate from Figure 1a after the gate restructuring approach presented in this paper has been applied. In the restructured gate, inputs B and C are exchanged, which can not be exchanged in a pin re-assignment approach. Also, the capacitance encountered along the different paths in the gate are changed from the original circuits. For instance the capacitance along path A -> C -> D in the pull-up side of the original circuit is significantly different from that along the equivalent path D -> C -> A in the restructured circuit. From the example in Figure 1b, it is clear that the effect on power of a particular circuit restructuring is difficult to anticipate. Furthermore, the impact of a circut restructuring on power is dependent on the input switching frequencies, the input arrival times, the delay requirement, and the transitor sizes. The restructuring of the pull-up and pull-down portions of the gate also affect each other. Each side changes the charge sharing effects and gate output load seen by the other side. Therefore, a restructuring approach must simultaneously optimize both sides of the pupd network and can not consider each side independently.

We present the following approach to the gate restructuring problem. First, from an initial gate topology we extract a BDD (binary decision diagram) based representation (called an SP-BDD) for the entire gate. The SP-BDD uses an input variable order, such that the BDD representation is a canonical representation of both the logic function and the structure of the gate. Using the SP-

ISLPD 95 Dana Point CA USA© 1995 ACM 0-89791-744-8/95/0M..$3.50

BDD representation, we can efficiently generate all possible circuit structures for a given gate. Since the power consumption of each possible circuit structure is highly dependent on the behavior of the signal inputs, a possible circuit structure is evaluated using switch-level simulation. The presented approach was implemented and tested on a number of benchmark circuits. The power consumption of the tested gates improved by 13% on average, and as much as 17% for some gates.

The remained of this paper is organized as follows. Section 2 presents an overview of the proposed SP-BDD representation. Section 3 presents algorithms for extraction and manipulation of the SP-BDD representation for power optimization. Section 4 presentes the algorithm for gate structure optimization. Section 5 presents the experimental results. Finally, Section 6 states some concluding remarks.

2. Series-parallel networks and their SP-BDD representation.

A large portion of digital CMOS circuits can be divided into pull-up and pull-down networks, also called pupd-networks. A pupd-network is a network with one source and one output terminal. The source terminal is connected to a node with a constant potential, VSS or VDD. The output node is connected to the gate output node and has a variable potential. Series-parallel networks (SP-networks) are widely used as pupd-networks. In this paper we limit ourselves to pupd-networks that are SP-networks, containing transistors of a single type (n or p).

Various representations of SP-network were developed for specific purposes [3,4]. However, no method represents simultaneously the structure and the Boolean function of an SP-network. We propose a representation for SP-networks and their corresponding Boolean functions based on a new type of BDD called the SP-BDD. The SP-BDD represents not only the Boolean function of a SP-network, but also its topology in a canonical form.

A transistor netlist can be represented as a non-directed graph (circuit graph) with vertices representing nodes and edges representing transistors. Let A and B be transistors in an SP-network. We define that A<B ("A is earlier than B") if there is an acyclic path in the circuit graph, starting at a source node and ending at an output node that contains A and B, such that A precedes B. It can be shown that:

1. Relation "<" is a partial ordering.

2. The SP-network topology is fully described by the partial ordering of the transistors.

Therefore, an SP-network is fully specified if and only if the list of transistors and their partial ordering are specified.

We can extend this partial ordering for an SP-network as follows: If an SP-network contains the SP-network Z=parallel(X,Y), where X and Y are also SP-networks, then either:

1. a<b for any a from X and b from Y or,

2. b<a for any a from X and b from Y.

It can be shown that this extended ordering is linear ordering for an SP-network of transistors. The extended ordering for an SP-network is compatible with both its partial ordering and one of possible partial orderings for complementary SP-network. Therefore, a single linear ordering can be found such that it fully describes the topology of two complementary SP-networks.

In addition to the topology information of a SP-network, it is desirable to represent the Boolean function of an SP-network. BDDs a provide popular and effective representation for Boolean functions [5]. There are various modifications of BDDs (OBDD, ROBDD, FBDD [6], etc.).ROBDD [6] (reduced ordered BDD) is the most usable modification, for which many effective algorithms are developed. For a given Boolean function and an arbitrary order of arguments there is unique ROBDD. Given an SP-network, we can construct an ROBDD for its Boolean function. By choosing the extended ordering of its transistors as an order for the BDD arguments, we get an SP-BDD. An SP-BDDs has two important properties:

A. An SP-BDD is minimal in size, since it contains exactly one vertex for each argument (transistor).

B. For a pair of complementary pull-up/pull-down networks, an SP-BDD is unique.

Therefore, the SP-BDD is a canonical representation for both the logic function and the topology of a complementary CMOS logic gate.

3. Algorithms for constructing an SP-BDD from a pupd network.

Each vertex v of a SP-BDD has two vertices as sons: low(v) denotes the low-son, and high(v) denotes the high-son. If X is SP-BDD then root(X) means its root vertex. Contrary to [6] we do not store in memory the leaf vertices of the SP-BDD. Instead, we make the simplifying assumption that every vertex of BDD with vertex.low=NULL has leaf vertex "0" as its low-son and every vertex with vertex.high=NULL leaf vertex "1" as its high-son. The corresponding SP-BDD consists of a single vertex v with low(v)=0 and high(v)=1.

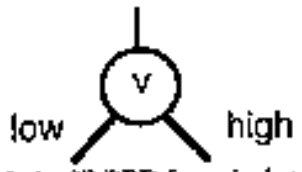

Figure 2. An SP-BDD for a single transistor.

The following operations on a BDD can be defined. Let X be an SP-BDD for an arbitrary SP-network. Let Y be a subgraph of X generated by non-leaf vertices with a.index <= vertex.index <= b.index, where a, b are non-leaf vertices of X. In this case we say that Y is an interval of X: Y=interval (X, a, b). Furthermore, let assignment statement low(Y)=v be defined by the following pseudo-code:

```
for (all vertices of Y) {
  if (low(vertex) is not in Y)
  vertex.low = &v;
}
```

Similarly, high(Y)=v means:

```
for (all vertices of Y) {
 if (high(vertex) is not in Y)
 vertex.high = &v;
}
```

For the sake of clearity, we use the the same notation for elements of an SP-BDD and the corresponding elements of an SP-network. For example, "X" is used for both an SP-network and the corresponding BDD, "a" denotes the SP-BDD vertex as well as the corresponding argument of the Boolean function and the corresponding transistor in the SP-network.

If X is a SP-BDD and v is the vertex of X, we can divide X into two BDDs Y and Z, where root(Y)=root(X) and root(Z)=v. For this operation we use the notation:

(Y,Z) = divide (X,v)

On the contrary, we can unite several BDDs A, B, C into one ordered graph with the following order of vertices: first all vertices of A, second all vertices of B, and so on. For this operation we use the notation:

X = order (A, B, C);

It should be noted that after this operation X is not yet an SP-BDD. To transform it to an SP-BDD we must make the root vertices of B and C the sons of some other vertices with lower indices.

We can also unite two BDDs X and Y into one ordered graph by inserting all vertices of Y after prescribed vertex v from X and before all subsequent vertices of X:

Z = insert (X, v, Y);

It should be noted that after this operation all connections between vertices of X are preserved.

A series connection in an SP-network Z=series(X, Y) is defined below in terms of an SP-BDD and is illustraded in Figure 3.

Z = order (X, Y);
high(X) = root (Y);

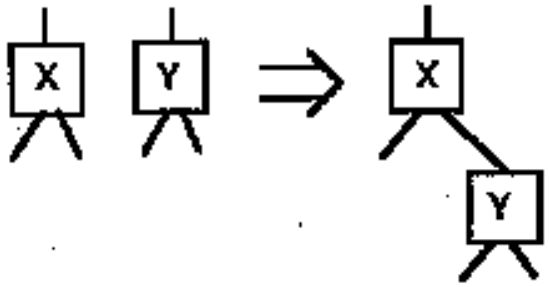

Figure 3. Z = series (X, Y);

Similarly, a parallel connection in an SP-network Z=parallel(X, Y) is defined below in terms of an SP-BDD and is illustraded in Figure 4.

Z = order (X, Y);
low(X) = root (Y);

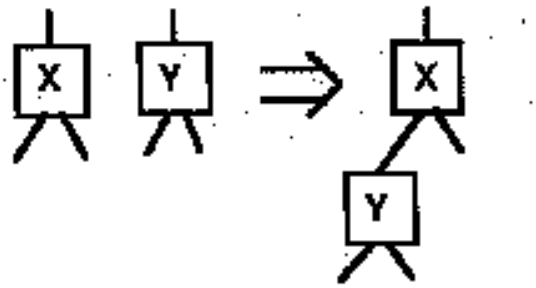

Figure 4. Z = parallel (X, Y);

Of course, the same SP-network Z=parallel(X, Y) can also be defined as follows:

Z = order (Y, X);
low(Y) = root (X);

Therefore, in terms of BDDs, parallel (X,Y) and parallel (Y,X) are two different BDDs representing single SP-network.

A circuit of multiple pupd networks is now extracted interms of SP-BDDs by the following algorithm. First, the circuit is partitioned into DC-connected components (DCCC). Then, each DCCC is processed as follows:

First, an initial path from a source node (either VSS-node or VDD-node) to any output node (node with terminal connected transistors of both types) is found. The initial path is defined by the following sequence of nodes and transistors:

no[0]-tr[0]-...-no[i]-tr[i]-...-no[n-1]-tr[n-1]-no[n].

We construct an SP-BDD for this path with node no[0] as source and node no[n] as output with the function X = make_bdd (path) as defined in pseudo-code below:

```
for (i=0; i<n; i++) {
 create (v[i]); /* create new vertex */
 v[i].transistor = tr[i];
 v[i].node = node[i];
 v[i].index = i;
 if (i > 0) {
  v[i-1].low = NULL;
  v[i-1].high = &v[i];
 }
}
v[n-1].low = NULL;
v[n-1].high = NULL;
create (X); /* create new BDD */
X.root = &v[0];
X.type = tr[0].type;
X.output_node = no[n];
```

After the initial path is found, and its corresponding SP-BDD is defined, the following sequence of steps is performed to defined the SP-BDD for the entire SP-network:

1. Search for a chord path (any path connecting two nodes of an already found path).

2. Form an SP-BDD for the chord path according the algorithm shown above.

3. Include the SP-BDD of the chord path in the SP-BDD of the total SP-network.

The SP-BDD for the chord path is included in the total SP-BDD as follows. Let X be BDD for the total SP-network containing a path with as first transistor a and as last transistor b, where a < b or a = b. Let Y be the SP-BDD for the chord path that should have X inconnected in parallel with mentioned path. It is also assumed that X does not contain any different path in parallel with the part of the mentioned path. We use the following notation for this operation: X = include (X, a, b, Y), as defined below.

```
I = interval (X, a, b);
if (b.low == NULL  &&  b.high == NULL) {
  Z = order (X, Y);
  low (I) = root (Y);
}
else {
  Z = insert (X, b, Y);
  low (Y) = low (b);
  high (Y) = high (b);
  low (I) = root (Y);
}
X = Z;
```

An example of a pupd-network and its corresponding SP-BDD built by the algorithm is shown in Figure 5.

4. Transistor reordering using SP-BDDs.

We present the following algorithm of transistor reordering for power minimization under delay constraints. Let X be SP-BDD and let v be a vertex of X. To denote the previous and the next vertex of X we use the following notation: previous(X,v), next(X,v). We consider transistor reordering in pupd-network as a sequence of elementary reorderings. Each elementary reordering, called a series_change, is performed as follows.

Let X be pupd-network (or corresponding SP-BDD) and let Y be subnetwork of X such that Y=series(A,B), where A either contains a single transistor or consists of two or more SP-networks connected in parallel (the same for B). Series_change makes the following substitution:

series(A,B) -> series(B,A).

If A is initially connected to node_a (as source_node) and to node_b (as output_node) and B is initially connected to node_b and node_c, then series_change includes the following steps:

1. Disconnect A from node_a and node_b;
2. Disconnect B from node_b and node_c;
3. Connect B to node_a and node_b;
4. Connect A to node_b and node_c.

For current state of pupd-network X, exactly n different elementary reorderings are possible, where n is the number of internal nodes of X. In terms of the SP-BDD, we perform the elementary reordering by changing places of two neighbouring intervals in the SP-BDD. An example of elementary reordering is shown by Figure 6.

The reordering algorithm performs transistor reordering in pupd-networks as a sequence of elementary reorderings. All possible elementary reorderings for the current state of pupd-network are ordered by corresponding SP-BDD. The SP-BDD representation, therefore, allows for an efficient enumeration of all possible elementary reordering for both the pull-up and pull-down sections of the gate.

It is difficult to predict exactly how each elementary reordering will affect the power and delay characteristics of the gate when changing places of two arbitrary SP-networks. Therefore, we calculate power and delays before and after every elementary reordering using a switch-level simulator. Switch-level simulation has the advantage that it directly models the switching frequencies and arrival times of the inputs and internal nodes and is still fast enough to perform an evaluation of all elementary orderings.

Elementary reordering changes node capacitances and paths of their charging and discharging. Therefore, optimal transistor sizes before and after elementary reordering are different. We perform transistor reordering for the circuit with minimal transistor sizes. Reordering is then followed by transistor resizing to meet the delay constraints.

Initially, two reordering algorithms were examined:

1. All pupd-networks are reordered to reduce power.
2. Critical pupd-networks are reordered for delay.
3. The circuit is resized to meet delay constraints.

However, extensive experiments showed that all the gain in power results from the first stage, while the second stage has almost no useful effect.

5. Experimental results.

The described reordering algorithm was implemented and tested on a number of circuit. Several benchmark circuits were selected for this purpose. These circuits contain pull-up/pull-down networks of various size. For every benchmark circuit transistor resizing for power minimization under delay and input capacitance constraints was performed before and after reordering, and the results were compared.

Values of switching frequency and arrival time for circuit inputs were obtained using a random number generator. These values were used in the power calculation by the switch-level simulation. Testing results are shown in Table 1.

Table 1 shows that the power dissipation of the tested circuits is reduced by 13%, on average. More complex gates, such as And-Or circuits and the complex gate gate7 show a significantly higher improvement in power dissipation. It should be noted that these complex gates, such as ao321, are quite common in current circuit design.

6. Conclusions

In this paper, we presented a new approach to gate reordering. The proposed reordering approach is not limited to pin-reassignment, but performs a more extensive optimization by modifying the structure of complex gates. The optimization approach uses a new BDD based representation, called an SP-BDD, which is a canonical representation of the Boolean function, as well as, the topology of the series-parallel gate. The presented algorithms were implemented and tested on a number of benchmark circuits. The experiments show that the power dissipation improved on average by 13%. For commonly used complex gates, such as an AO321, the proposed

approach improved the power dissipation by as much as 17%. The presented algorithms, therefore, are a pratical and effective approach to low power design.

References

[1] B.S.Carlson, C.Y.R.Chen. "Performance Enhancement of CMOS VLSI Circuits by Transistor Reordering", Proc. of 30th DAC, 1993, p.361-366.

[2] S.C.Prasad, K.Roy. "Circuit Optimization for Minimization of Power Consumption under Delay Constraint", Proc. of Int. Workshop on Low Power Design, 1994, p.15-20.

[3] C.H.Tan, J.Allen. "Minimization of Power in VLSI Circuits Using Transistor Sizing, Input Ordering, and Statistical Power Estimation", Proc. of Int. Workshop on Low Power Design, 1994, p.75-80.

[4] M.Boehner. "LOGEX - An Automatic Logic Extractor from Transistor to Gate Level for CMOS Technology", Proc. of 25th DAC, 1988, pp.517-522.

[5] J.P.Caisso, E.Cerny, N.C.Rumin. "A Recursive Technique for Computing Delays in Series-Parallel MOS Transistor Circuits", IEEE Trans. on CAD, 1991, v.10, n.5, pp.589-595.

[6] R.E.Bryant. "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. on Computers, 1986, v.35, n.8, pp.677691.

[7] J.Bern, J.Gergov, C.Meinel, A.Slobodova. "Boolean Manipulation with Free BDDs", Proc. of EDAC-94, pp.200-207.

[8] R.E.Bryant. "Boolean analysis of MOS circuits", IEEE Trans. on CAD, 1987, v.6, pp.634-649.

Table 1. Power resulting from reordering

Circuit	Initial Power	Final Power	Power Reductn
Nand4	15.9	13.3	17%
C17	14.1	13.8	2%
Mux	41.4	40.3	3%
AO221H	46.4	43.3	7%
AO221S	46.1	39.5	14%
AO321H	63.2	51.8	18%
AO321S	69.5	50.5	27%

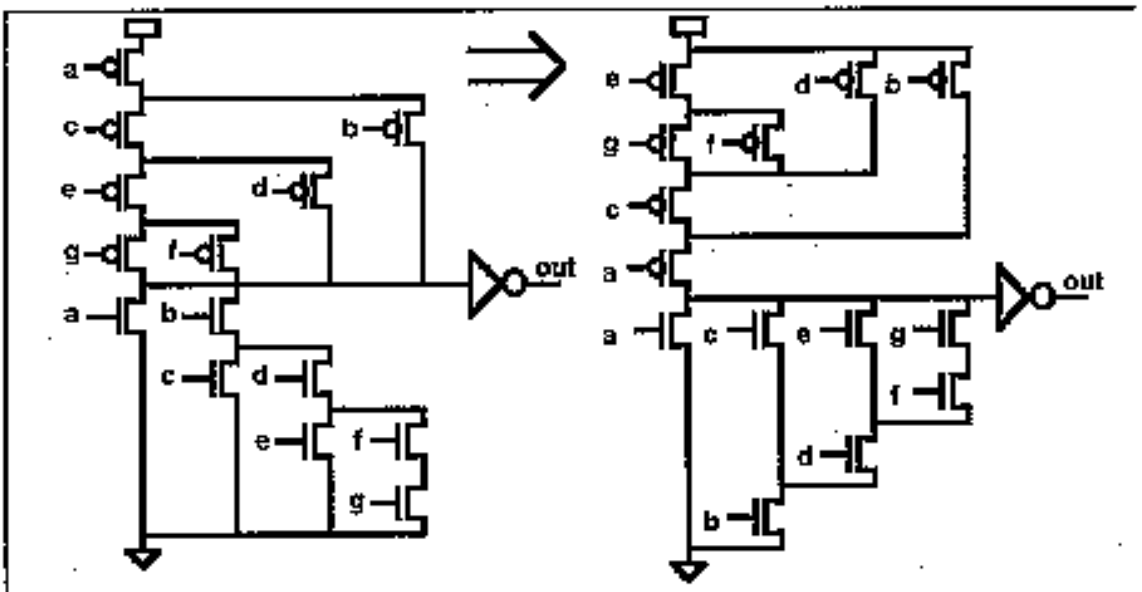

Figure 1. Complex gate before (a) and after (b) restructuring.

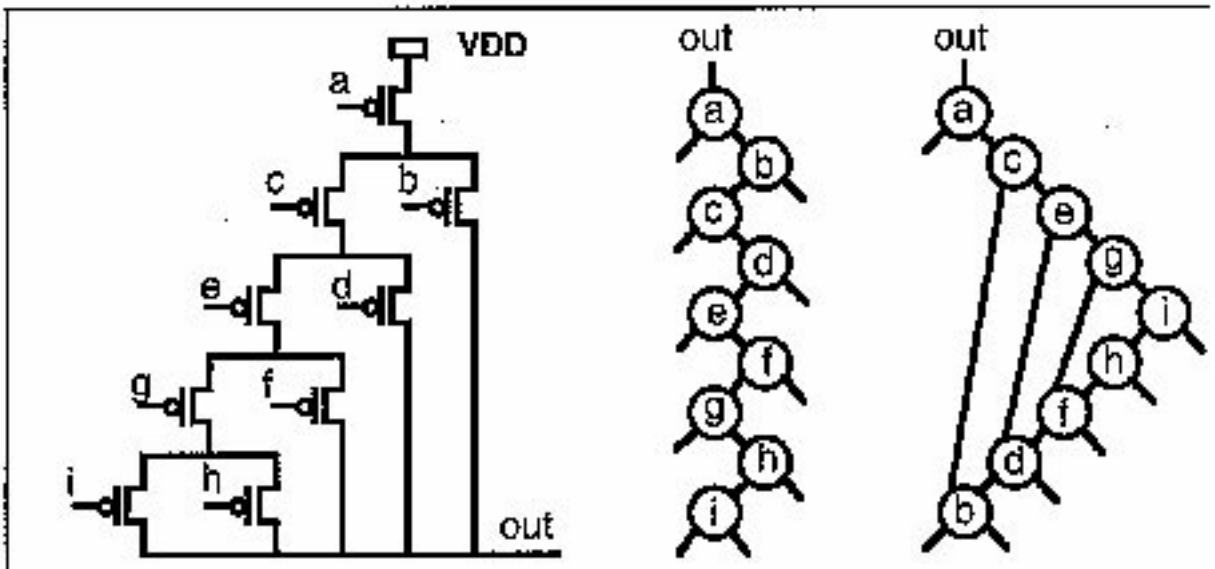

Figure 5. Pull-up network and two variants of its SP-BDD corresponding to different topologies of complementary pull-down network.

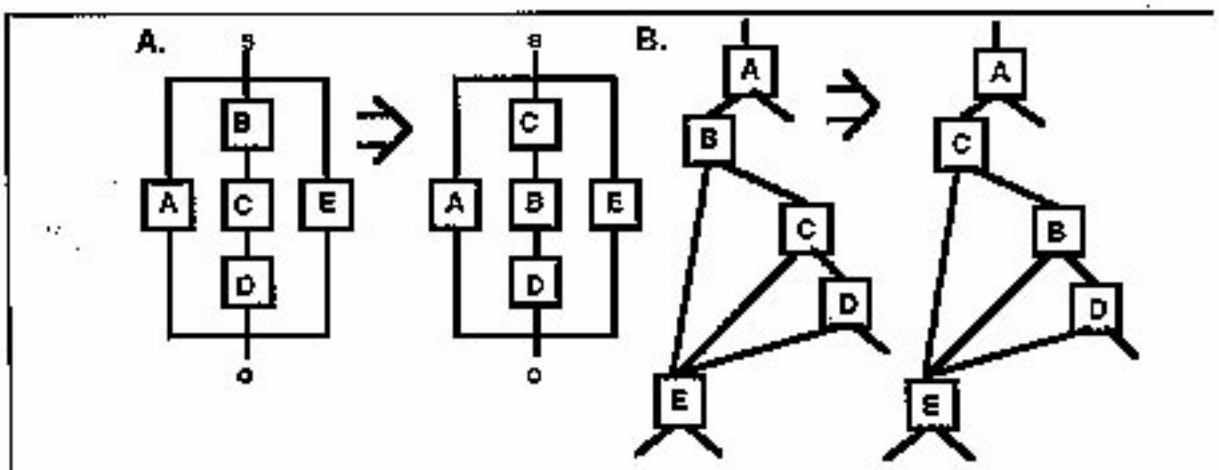

Figure 6. Elementary reordering B <-> C in terms of SP-networks (a) and in terms of SP-BDD (b). B and C are SP-networks of arbitrary complexity.