UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT

Combinational Static CMOS Networks

J.A. Brzozowski
M. Yoeli

CS-85-42

December, 1985

COMBINATIONAL STATIC CMOS NETWORKS*

J.A. Brzozowski

Department of Computer Science
University of Waterloo
Waterloo, Ontario
Canada

M. Yoeli

Department of Computer Science
Technion
Haifa, Israel

ABSTRACT

We develop mathematical switch-level models for static combinational CMOS networks. In contrast to other available MOS models and theories, our models capture design principles that are special to CMOS, such as the use of transmission gates. First we study networks consisting of cascade connections of CMOS cells realizing negative functions. We then extend this model to incorporate transmission gates. Finally, we develop a more complex CMOS graph model which includes a ternary transient analysis and is capable of handling some unconventional, but commercially used, combinational networks. Such designs cannot be properly explained by presently available theories. Also, we discuss several general design approaches.

1. Introduction

It is now generally accepted that conventional switching theory based on gate networks is not suitable for digital MOS networks, and that a switch-level model is required in order to represent the logic aspects of digital MOS circuits. Some early work on switch-level models of static NMOS and CMOS can be found in [BY 76, Appendix D]. Models covering both static and dynamic MOS have been developed by Bryant [B] and Hayes [H]. However, these "unified" models

---

* This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. A0871.

treat NMOS and CMOS in a similar way and do not take into account the significant differences between the design principles of CMOS and those of NMOS [WE]. For example, the unified models do not distinguish between "good" 2-transistor switches, i.e. transmission gates, and "poor" single-transistor switches [WE, p. 28].

Since CMOS technology is becoming the dominant VLSI technology, switch-level models particularly tailored towards digital CMOS networks are of considerable importance. One such model was proposed in [YB 85]. An early version appeared as [YB 84]. The model will reject as not "well-formed" any network which violates the rules of "good" static CMOS design, whereas many such networks would be accepted by the unified models and corresponding simulators. However, further studies of CMOS design revealed the model of [YB 85] to be somewhat too restrictive. It will reject some commercially used static CMOS designs, e.g. the 6-transistor CMOS XOR [WE, p. 317]. (Incidentally, this design will also be rejected by the unified models [WE, p. 256].) Consequently, this paper develops improved switch-level models for static, combinational CMOS networks. Actually, the paper proposes a hierarchy of switch-level models. The top level of this hierarchy, the "C-graph" of Section 5 is not required for the modeling of most static combinational CMOS circuits. However, the complexity of the C-graph model cannot be avoided, if certain "tricky" combinational CMOS designs are to be suitably verified. Some limitations of our switch-level model are discussed in Section 6. The concepts developed in this paper are intended to form a basis for a sound design methodology, as illustrated in Section 7.

2. Cells

The fundamental building blocks of CMOS networks are the N-channel and P-channel enhancement mode transistors [WE]. In Figure 2.1 we show commonly used symbols for the N-channel and P-channel transistors, along with the graph symbols that we will use in this paper. The graph of Figure 2.2 shows a typical simple CMOS network, namely a two-input NOR gate. In that figure 1 represents voltage VDD which is the positive end of the power supply, and 0 represents voltage VSS which is the negative end of the power supply. The symbols 0 and 1 also represent the customary logical values, using positive logic.

Refer to Figure 2.1(a). An N-transistor behaves like a switch between terminals t_1 and t_2; in particular, when $A = 1$ the switch is ON and when $A = 0$ it is OFF. Such a transistor is labeled A_N in Figure 2.1(b). An N-transistor which is ON will transmit 0-signals well, but will perform poorly in the case of a 1-signal [WE]. Similarly, in Figure 2.1(c) a P-transistor is ON when $A = 0$ and OFF when $A = 1$. A P-transistor which is ON transmits 1's well and 0's poorly.

Consider now the NOR gate of Figure 2.2. Let $A = 1$ and $B = 0$. There is a "good" connection between the input node labeled 0 and the output node Z via the N-transistor labeled A_N, which is ON since $A = 1$. Note also that there is no connection of any kind to the input node labeled 1. Therefore the output node Z becomes 0 reliably. The reader will verify that the graph of Figure 2.2 represents a NOR gate that behaves reliably for all input

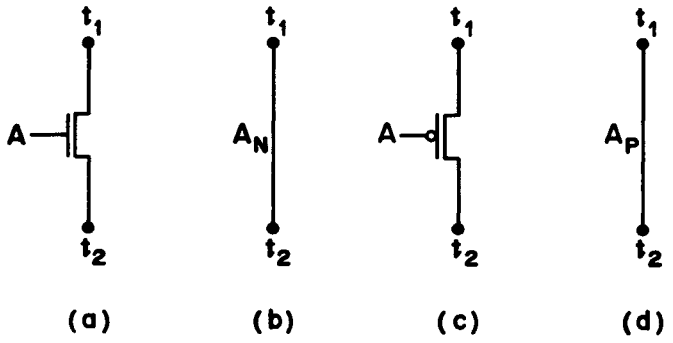

Figure 2.1 Transistor symbols: (a) N-channel transistor; (b) graph notation for (a); (c) P-channel transistor; (d) graph notation for (c).

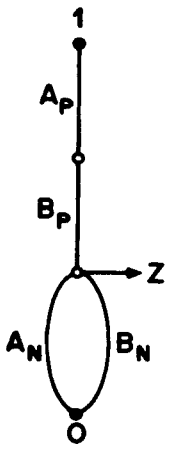

Figure 2.2 CMOS NOR gate.

combinations.

We now formalize this idea of CMOS logic gates such as NOR gates by introducing the concept of cell. A *cell* is a finite, labeled, undirected graph with three types of nodes:

(a) *Input nodes*, shown as black dots, and labeled by either a constant input (0 or 1) or by an input letter *(A, B,* etc.).

(b) *Internal nodes* shown as white dots.

(c) Exactly one *output node* - this is a special internal node, indicated by a small outgoing arrow.

Every edge in a cell graph must be labeled with a label of the form A_N or A_P where A is an input letter.

A ***binary input state*** of a cell is an assignment of 0's and 1's to the input letters. An *N-path* (*P-path*) in a cell graph is any path of N-transistors (P-transistors) which are ON. We define the following boolean functions on the input letters in any cell:

$g_0 = 1$ iff there exists an N-path from the output node to an input node with value 0;

$g_1 = 1$ iff there exists a P-path from the output node to an input node with value 1;

$t_0 = 1$ iff there exists any path of transistors which are ON from the output node to an input node with value 0;

$t_1 = 1$ iff there exists any path of transistors which are ON from the output node to an input node with value 1.

The following definition is motivated by good design rules for combinational static CMOS. We say that the output Z of a cell is 0 only if there is a "good" connection, i.e. an N-path, to 0 and no connection to 1. We define $Z = 1$ in a similar way. We use the value $Z = 2$ to indicate that the output node is isolated, to model tri-state outputs. Finally $Z = 3$ denotes all the other situations which are considered undesirable under stable conditions. Formally, a 4-valued *output function* Z over the binary input variables is defined as follows:

$Z = 0$	iff $g_0 = 1$ and $t_1 = 0$;
$Z = 1$	iff $g_1 = 1$ and $t_0 = 0$;
$Z = 2$	iff $t_0 = t_1 = 0$;
$Z = 3$	otherwise.

The cell is ***boolean*** if the output value is always either 0 or 1. It is ***tri-state*** if $Z \in \{0,1,2\}$, and it is not boolean.

We now illustrate the above definitions by several examples. For the cell of Figure 2.2 one verifies that

$$g_0 = t_0 = A+B, \quad g_1 = t_1 = A'B' = g_0', \text{ and } \quad Z = (A+B)'.$$

In the example of Figure 2.3 we find g_0 by enumerating all the N-paths from Z to 0, namely

$$g_0 = AD+BE+ACE+BCD.$$

Similarly, by considering all the P-paths from Z to 1 we find

$$g_1 = A'B'+D'E'+A'C'E'+B'C'D'.$$

The reader may also note that the graph consiting of the P-transistors is the dual of the graph consisting of the N-transistors. The construction of such duals is

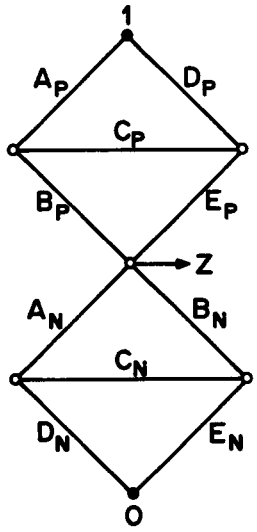

Figure 2.3 A "bridge-type" cell.

described in [C]. The cells of Figures 2.2 and 2.3 are boolean. One verifies that in any boolean cell

$$g_1 = g_0', \quad g_0 = t_0, \quad g_1 = t_1, \quad \text{and} \quad Z = g_1.$$

In our third example, in Figure 2.4, we find $g_0 = t_0 = AB$ and $g_1 = t_1 = A'B'$. Note that $g_0 \neq g_1'$ but $g_0 g_1 = 0$. Here the cell is tri-state:

$Z = 0$	iff $g_0 = 1$;
$Z = 1$	iff $g_1 = 1$;
$Z = 2$	otherwise.

Our final example is shown in Figure 2.5. To compute g_0, note that there are two paths to be considered: the path labeled $A_N B_N$ from 0 to Z and the path labeled C_N from B to Z. The latter becomes a path from 0 to Z when $B = 0$. Altogether $g_0 = AB + B'C$. To compute t_0 we must also consider paths other than N-paths. If $A = 0$ there is a path labeled B_P; this contributes the product $A'B'$ to t_0. Also, when $B = 0$, there is the path labeled A_P, but this contributes the same product $A'B'$. Thus $t_0 = g_0 + A'B'$. Similarly one verifies that $g_1 = A'B + AB'$ and $t_1 = g_1 + BC$. Note that, when $A = B = C = 1$, the output is $Z = 3$. Thus this cell is neither boolean nor tri-state. Nevertheless this cell can be useful as will be shown in Section 6.

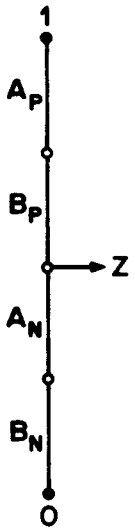

Figure 2.4 A tri-state cell.

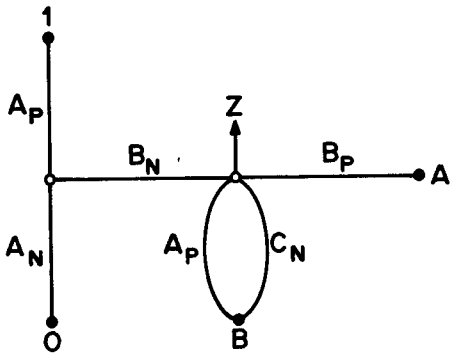

Figure 2.5 A cell that is neither boolean nor tri-state.

3. Boolean and Tri-State Cells

In this section we study the structure of cells which are either tri-state or boolean. A cell is said to be *redundant* if some edge can be removed or shorted without changing the output function; otherwise the cell is *irredundant*.

Proposition 1. If an irredundant cell is tri-state or boolean, then all of its input nodes are labeled by constant inputs.

Proof: Suppose the cell has an input node labeled by input variable A. Suppose also that there is an N-path t from Z to A for some input state.

Assign the value 1 to all the cell inputs. Then N-path t still exists and $t_1 = 1$, but $g_1 = 0$ since no P-transistors can be ON. Thus $Z = 3$ which contradicts our assumption. Therefore there cannot be any N-paths from Z to A. By a similar argument we show that there cannot be any P-paths from Z to A. Consequently, removing node A and its incident edges cannot destroy any P-paths or N-paths from Z to any input node. Thus one verifies that the output function remains unchanged if node A and its edges are removed. But this implies that the cell is redundant. □

A boolean function f is *positive* iff there exists a sum of products of uncomplemented variables for f. A boolean function is *negative* iff its complement f' is positive. The following propositions show the importance of positive and negative functions in CMOS design.

Proposition 2. If a cell is tri-state or boolean then g_0 is positive and g_1 is negative.

Proof: If the cell has an input node labeled by an input variable, that node and its edges can be removed as in the proof of Proposition 1 without affecting the output function. Thus we may assume that all the input nodes are labeled by constant inputs. Consider a path t consisting of N-transistors with labels $A_N^1, \ldots, A_N^k$, from Z to an input node labeled 0. The path t is an N-path iff $A^1A^2 \cdots A^k = 1$. The function g_0 can be denoted by the boolean sum of all such path products, and is therefore positive. By a similar argument g_1 is negative. □

A commonly used method of implementing any negative function is based on the structure shown in Figure 3.1 where the P-part (N-part) consists only of P-transistors (N-transistors). We call this structure a *separated* cell. Clearly a separated cell is tri-state iff $g_0\, g_1 = 0$ and $g_0 + g_1 \neq 1$. It is boolean iff $g_0 = g_1'$. One easily verifies that any negative function can be implemented by a separated boolean cell. However, an implementation of a negative function is not necessarily separated, even if it is an irredundant implementation. The cell of Figure 3.2 is an implementation of the NAND function $Z = (AB)'$, that is not separated. No transistor can be removed or shorted without changing the output function; hence the cell is irredundant. On the other hand, this cell is not minimal, since the separated implementation of the NAND function requires only 4 transistors. In general, we have the following result.

Proposition 3. For any negative boolean function there exists a separated cell implementation with the minimal number of transistors (i.e. no non-separated cell could have fewer transistors).

Proof: Consider any implementation of a negative boolean function by some non-separated cell G. Let G_P (G_N) be the subgraph of G which consists of all the P-transistors (N-transistors) of G. Now construct a separated cell with G_P as P-part and G_N as N-part. Clearly the output function and the transistor count are the same. □

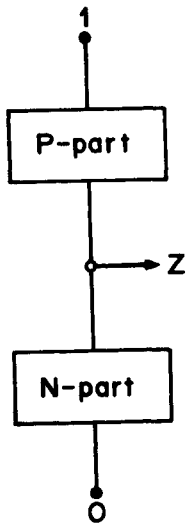

Figure 3.1 General form of separated cell.

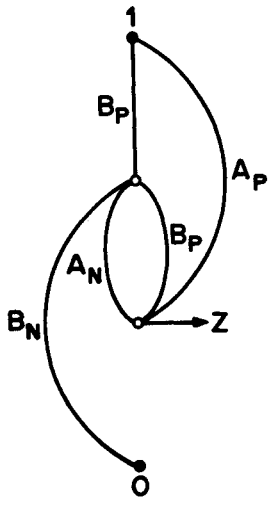

Figure 3.2 A non-separated NAND cell.

One also verifies that the following result holds; the proof is similar to the proof of Proposition 3.

Proposition 4. For any tri-state cell there exists a separated cell implementation with the minimal number of transistors (i.e. no non-separated implementation can have fewer transistors).

In general, a tri-state function is not necessarily realizable by a single cell. In fact the following result is easily verified.

Proposition 5. Let f be a tri-state function, and let f_0 and f_1 be boolean functions defined by $f_0 = 1$ iff $f = 0$ and $f_1 = 1$ iff $f = 1$. Then f is realizable by a single cell iff f_0 is positive and f_1 is negative.

The problem of designing separated cells with the minimal number of transistors is closely related to the design of minimal contact networks [C]. The minimization of the transistor count in the N-part is equivalent to the minimization of the corresponding contact network, which uses only normally open contacts. Similar remarks apply to the P-part.

4. Cascades

Arbitrary boolean functions may be implemented as cascade compositions of negative functions, such as NOT and (multi-input) NAND or NOR functions. Thus, any boolean function may be implemented as a cascade of suitably interconnected separated cells. Figure 4.1(a) shows such an implementation of the XOR function, consisting of two separated cells Γ^1 and Γ^2, where Γ^1 implements the function $Z^1 = (A+B)'$ and Γ^2 implements $Z^2 = (Z^1+AB)'$. Thus $Z^2 = A \oplus B$.

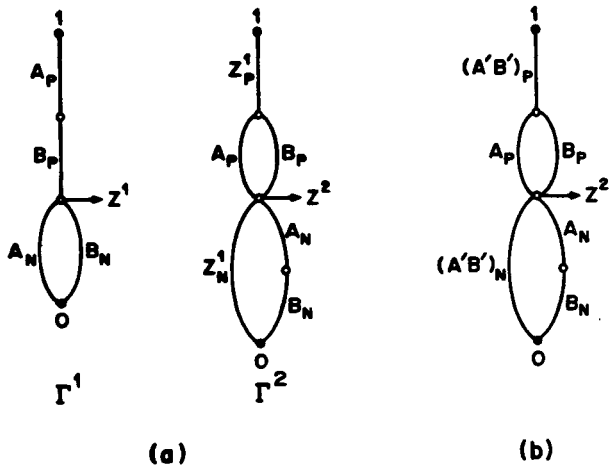

Figure 4.1 (a) XOR with separated cells; (b) $\tilde{\Gamma}^2$.

The concept of "cascade of suitably interconnected cells" will be formalized in this section. Moreover, our formal concept of "cascade" is also intended to model combinational networks containing transmission gates [WE]. A 2-cell

implementation of the XOR-function using transmission gates is shown in Figure 4.2(a). Γ^1 is an inverter, producing $Z^1 = A'$. Γ^2 contains two transmission gates, one controlled by A and the other by $Z^1 = A'$. Note, however, that cell Γ^2, considered by itself, is neither boolean nor tri-state. For example, when $A = B = Z^1 = 1$, we have $Z^2 = 3$. Nevertheless, the composition of Γ^1 and Γ^2 is a suitable implementation of $Z^2 = A \oplus B$.

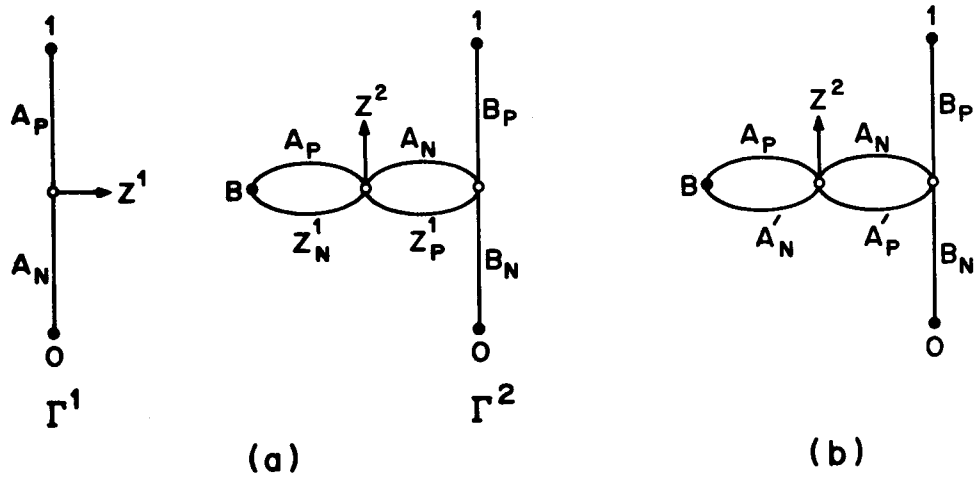

Figure 4.2 (a) XOR implementation using transmission gates; (b) $\tilde{\Gamma}^2$.

The above considerations motivate the following formal definitions.

Let $\mathbf{A} = A^1, \ldots, A^n$ be a vector of input letters. A *cascade over* $\mathbf{A}$ is a finite sequence $\Gamma = \Gamma^1, \ldots, \Gamma^k$ of cells having the following property: Any input letter of Γ^j is either in $\mathbf{A}$ or in the set $\{Z^1, \ldots, Z^{j-1}\}$, where Z^i denotes the output node of Γ^i. However, any input letter Z^i $(1 \leq i < j)$ may only appear as an edge label of Γ^j, and not as an input node label. The input vector of Γ is defined to be $\mathbf{A}$, and its output is Z^k.

An *extended cell* is similar to a (basic) cell as defined in Section 2 except that the edge labels are of the form f_N or f_P where f is a boolean function on the input variables. An edge labeled f_N (f_P), again representing an N-transistor (P-transistor), is ON when $f = 1$ ($f = 0$) and OFF when $f = 0$ ($f = 1$). The concept of output function defined for basic cells in Section 2 is generalized to extended cells in the obvious way.

To illustrate these definitions consider the two-cell cascade of Figure 4.1(a). Analyzing the first cell we find $Z^1 = A'B'$. We now replace Z^1 in all the relevant labels of the second cell by $(A'B')$, thus obtaining the extended cell of Figure 4.1(b). Now Z^2 is 1 iff $(A'B') = 0$ and either A or B is 0, i.e.

$$Z^2 = (A+B)(A'+B') = A \oplus B.$$

Alternatively, from Figure 4.1(a),

$$Z^2 = (Z^1+AB)' = (A'B'+AB)' = A \oplus B$$

As a second example consider the two-cell cascade of Figure 4.2(a). In Figure 4.2(b) we show the extended version of the second cell Γ^2 obtained by substituting A' for Z^1. Now one verifies that $Z^2 = A \oplus B$. Note, however, that the alternative method which we used for Figure 4.1 does not apply here because Γ^2 is neither boolean nor tri-state. A problem occurs when $A = B = Z^1 = 1$, giving $Z^2 = 3$. However, if Γ^2 is used in cascade with Γ^1, $Z^1 = A'$ and the offending input combination does not occur under steady-state conditions.

To analyze any cascade proceed from Γ^1 to Γ^k using step-by-step substitutions to obtain the appropriate extended cells $\tilde{\Gamma}^1 = \Gamma^1, \tilde{\Gamma}^2, \ldots, \tilde{\Gamma}^k$. If each $\tilde{\Gamma}^i$, $1 \leq i < k$, whose output is used in subsequent cells is boolean and $\tilde{\Gamma}^k$ is boolean (tri-state), then the cascade Γ is said to be *boolean (tri-state)*.

Note that the cascade of Figure 4.2 has fewer transistors than the cascade of separated cells in Figure 4.1. One can verify that any cascade of separated cells that realizes the XOR function requires at least 10 transistors. If we restrict our attention to cascades of separated cells, our next example shows that using the minimal number of cells does not always result in a minimal transistor count. Consider the function $f = (ABC)' \oplus D$. The minimal number of cells for this function is 2, as can be verified by applying Muroga's algorithm [M]. In this case the algorithm results in a unique solution, namely a cell realizing the function $Z^1 = (ABCD)'$ and a cell realizing the function $Z^2 = [Z^1 (D+ABC)]'$. One verifies that any cell realizing Z^1 must use at least 8 transistors, and Z^2 requires at least 10 transistors. Thus the minimal 2-cell solution uses 18 transistors. The following 3-cell solution uses only 16 transistors:

$$Z^1 = (ABC)',$$
$$Z^2 = (Z^1+D)',$$
$$Z^3 = (Z^1D+Z^2)'.$$

5. C-Graphs

So far we have been concerned with the steady-state behavior of CMOS networks, disregarding any transient phenomena. However, we show later in this section that the study of transient behavior is essential for the proper understanding of CMOS networks, even if we restrict our attention to combinational networks only. Furthermore, the new model of combinational CMOS networks to be introduced in this section, will enable us to make essential distinctions between networks implementing the same boolean function. This distinction is a novel feature of CMOS switching theory and is not applicable to conventional gate networks.

To be more specific, let us consider the two cascades of Figure 5.1. One can verify that they are both implementations of the AND function $Z = AB$.

We wish to study the behavior of the two networks, under the assumption that $B = 0$, but the value of A is unknown. "Unknown" is to be understood as indicating that the voltage representing A could be VDD or VSS or some intermediate value between VDD and VSS. In the network of Figure 5.1(a), when $B = 0$, there is a P-path from Z^1 to 1 and no path to 0. Consequently $Z^1 = 1$ and $Z = 0$. In the network of Figure 5.1(b), if A is unknown, then so is Z^1. If one assumes that A is unknown but binary, i.e. has the value either 0 or 1, then the output Z is well defined in each case, and has the value 0 if $B = 0$. However, if A should take on some intermediate voltage between VDD and VSS, then the output Z is not well defined. Note that, under the same conditions, the output of Figure 5.1(a) *is* well defined.

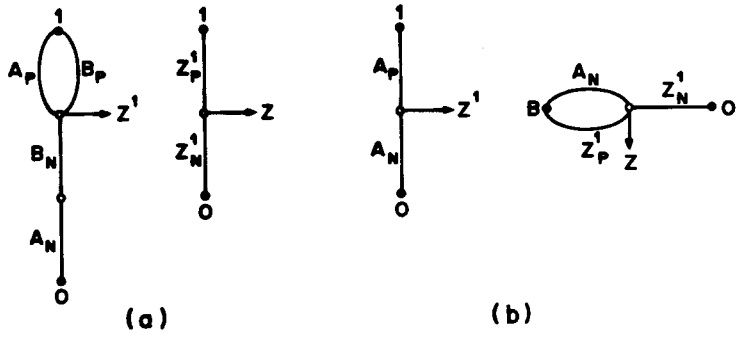

Figure 5.1 Two implementations of the AND function.

In order to model the different behavior of the two networks of Figure 5.1, we introduce a third value X (sometimes also denoted by ½). Ternary models have been previously used for gate networks [BY 79] and switch-level MOS simulators [B]. The ternary behavior of the two networks of Figure 5.1 is described in Figure 5.2. The model we are about to introduce will distinguish between these two types of behavior.

Our next example illustrates a feature of CMOS networks that we have not considered so far. Figure 5.3 shows a widely used CMOS implementation of the XOR function [WE]. Here an internal node (Q^1) controls one of the transistors (labeled Q^1_N). Assume first that $A = 0$ and $B = 1$. Then Q^1 has a P-path to 1, and no path to 0. Thus Q^1 becomes 1 reliably, and Z is also 1. If now A changes from 0 to 1 we reach a configuration that cannot be analyzed correctly by means of our previous models. This is because Q^1 has an N-path to 0 (through A_N), but also another N-path (B_N,Q^1_N to $B = 1$) to 1. However, we will show shortly that this N-path to 1 is a "non-conducting" path and should be ignored. A similar situation arises if we assume that Q^1 is initially unknown, i.e. has the value X. When $A = B = 1$, Q^1 has an N-path (A_N) to 0 and a

A \ B	0	X	1
0	0	0	0
X	0	X	X
1	0	X	1

(a)

A \ B	0	X	1
0	0	0	0
X	X	X	X
1	0	X	1

(b)

Figure 5.2 (a) Ternary table network of Figure 5.1 (a);
(b) Ternary table for network of Figure 5.1 (b).

potential N-path to 1 (B_N,Q_N). We will show that this potential path can be ignored also.

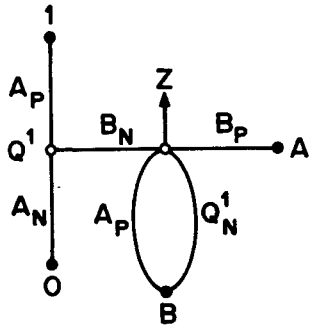

Figure 5.3 6-transistor implementation of the XOR function.

To explain the concept of "non-conducting" paths we first make the following formal definitions. We generalize the concept of a cell by permitting internal node variables to control any edges as in the example of Figure 5.3. A C-*graph* is a finite, labeled, undirected graph with four types of nodes:

(a) *Input nodes* shown as black dots, and labeled by either a constant input (0 or 1) or by an input letter (A,B, etc).

(b) *Internal nodes* shown as white dots.

(c) *Key nodes*, which are special internal nodes, each of which is labeled by a different *key letter* $Q^1, \ldots, Q^k$.

(d) One *output node* which is a special internal node shown by a small outgoing arrow.

Every edge in a C-graph is labeled with a label of the form Q_N or Q_P, where Q is either an input letter or a key letter. Each key letter appears as some edge label. A (ternary) *input-key state* is an assignment of 0, 1, or X to each input letter and each key letter. Similarly, a *total state* is an assignment of 0, 1, or X to each input letter and to each internal node.

Given a C-graph Γ and a ternary input vector for Γ, we define the concept of *self-dependent* path as follows. Let Q be a key-node of Γ and let e be an edge of Γ labeled Q_N. A path π containing both node Q and edge e is called self-dependent iff either π has a node Z labeled 0 and Q is between edge e and node Z, or π has a node U labeled 1 and Q_N is between Q and U. Similarly, a path containing Q and an edge labeled Q_P is self-dependent if either Q is between Q_P and 1 or Q_P is between Q and 0. These four situations are illustrated in Figure 5.4.

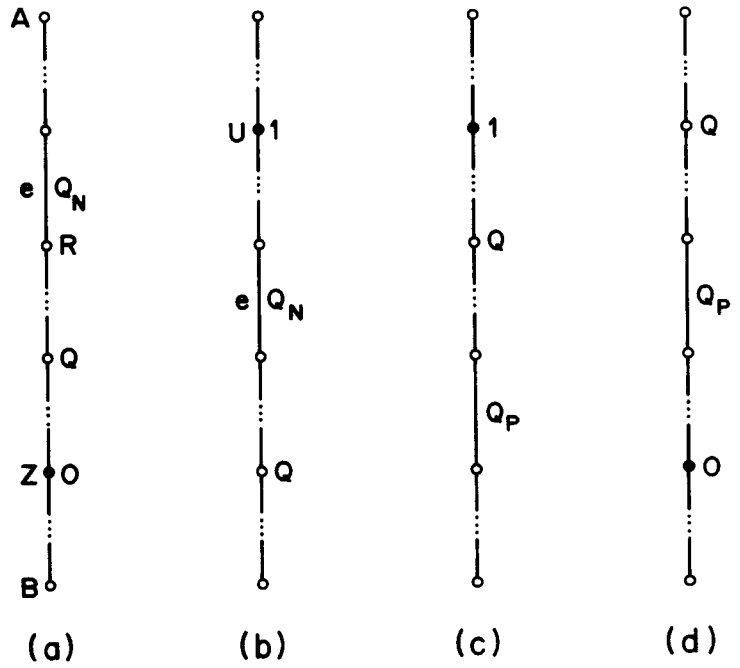

Figure 5.4 Self-dependent paths.

When analyzing C-graphs one should ignore self-dependent paths for the following reasons. Suppose we have a path of the type shown in Figure 5.4(a). If this path were to conduct current from node A to node B, the voltage at node Q would be close to the voltage at node R which is the source of Q_N. Thus the transistor labeled Q_N has a gate-to-source voltage that is essentially 0 (in fact, a small negative voltage). Thus Q_N is OFF, contradicting the assumption

that the path conducts. A similar argument holds for cases (b), (c), and (d) in Figure 5.4. In summary, self-dependent paths from any node to 0 or 1 will have no effect on that node.

Returning to our mathematical model, for each ternary input-key state of a C-graph Γ we define the following boolean functions for each internal node Q^i of Γ :

$g_0 = 1$	iff there is an N-path that is not self-dependent from Q^i to 0;
$g_0 = 0$	otherwise;
$t_0 = 1$	iff there is a path, which is not self-dependent, consisting of transistors that are either ON or labeled X from Q^i to 0;
$t_0 = 0$	otherwise.

The functions g_1 and t_1 are defined similarly (using P-paths from Q^i to 1 for g_1, etc.).

The excitation $E(Q^i)$ of node Q^i for a given ternary input-key state is defined as follows:

$E(Q^i) = 1$	iff $g_1 = 1$ and $t_0 = 0$;
$E(Q^i) = 0$	iff $g_0 = 1$ and $t_1 = 0$;
$E(Q^i) = X$	otherwise.

A key-node is *stable* in a given ternary input-key state iff $E(Q^i) = Q^i$. An input-key state is *stable* iff each key-node is stable.

To analyze the behavior of any C-graph started in a given ternary key state for a fixed ternary input vector, we define the relation ρ on $\{0,1,X\}^k$, which will determine the set of possible successor key states, as follows:

For $\mathbf{Q} = Q^1, \ldots, Q^k \in \{0,1,X\}^k$, define $E(\mathbf{Q}) = E(Q^1), \ldots, E(Q^k)$. Now

if	$\mathbf{Q} = E(\mathbf{Q})$	then $\mathbf{Q} \rho \mathbf{Q}$;
if	$\mathbf{Q} \neq E(\mathbf{Q})$	then $\mathbf{Q} \rho \tilde{\mathbf{Q}}$ for all $\tilde{\mathbf{Q}} \neq \mathbf{Q}$

such that $\tilde{Q}^i = Q^i$ or $\tilde{Q}^i = E(Q^i)$ for $i = 1, \ldots, k$.

We interpret this relation as that of a "General Multiple Winner" model [BY 79] as follows: If $\mathbf{Q} = E(\mathbf{Q})$ then $\mathbf{Q}$ is stable and $\mathbf{Q}$ is the only possible successor key state. If $\mathbf{Q} \neq E(\mathbf{Q})$, then some of the key nodes are unstable. In this case, let $\tilde{\mathbf{Q}}$ be any key state derived from $\mathbf{Q}$ by changing any non-empty subset of unstable key nodes to their corresponding values in $E(\mathbf{Q})$. Then $\tilde{\mathbf{Q}}$ is a possible successor of $\mathbf{Q}$. We repeat this process for $\tilde{\mathbf{Q}}$, etc. Altogether we find all the possible successors of initial key state $\mathbf{Q}$. This information is conveniently summarized in the graph of the relation ρ. Usually, one is only interested in the "non-transient" successors of $\mathbf{Q}$. Following [BY 79], we define a cycle in the graph of ρ to be *transient* iff there exists $i, 1 \leq i \leq k$, such that

node Q^i is unstable and has the same value for all the states in the cycle. A C-graph cannot be in such a cycle for a long time, for eventually Q^i must change. The final behavior of a C-graph is then reflected in the non-transient cycles of ρ.

A ternary input vector is *forcing* if the graph of the relation ρ has exactly one non-transient cycle, and this cycle is of length 1. The key state corresponding to this cycle is said to be the *response* of the input vector. A ternary input vector is *key-combinational* if it is forcing and its response is binary.

Given a key-combinational input vector we define the output value Z by the table:

g_0t_0 \ g_1t_1	00	01	11
00	2	3	1
01	3	3	3
11	0	3	3

Z

A C-graph Γ is *combinational (T-combinational)* if every binary input vector is key-combinational, and the corresponding output is boolean (tri-state). One verifies that this definition is an extension of the concepts from Section 4. In particular, a cascade is boolean (tri-state) iff it is combinational (T-combinational) considered as a C-graph.

Let us now return to the two implementations of the AND function in Figure 5.1. If the ternary input vector $A = X,\ B = 0$ is applied to each cascade started with $Z^1 = Z = X$, our analysis shows that this input is forcing and the response is $Z^1 = 1, Z = 0$ for Figures 5.1(a) but it is $Z^1 = X, Z = X$ for Figure 5.1(b). Therefore the input vector $A = X, B = 0$ is key-combinational for the network of Figure 5.1(a) but not for the network of Figure 5.1(b). In general, starting with key node $Z^1 = X$ and applying our analysis, one obtains the tables of Figure 5.2.

Let us now also reconsider the XOR gate of Figure 5.3. When $B = 0$, $Q^1 = A'$, since the transistor labeled B_N is OFF. If $B = 1$ and $A = 0$, there is a P-path from Q^1 to 1 and no path of ON transistors from Q^1 to 0. Finally if $A = B = 1$, there is an N-path from Q^1 to 0 and a self-dependent path from Q^1 to $B = 1$. Hence $E(Q^1) = 0$. Altogether, our model verifies that this C-graph realizes the XOR function reliably.

6. Composition of C-Graphs

In this section we consider composite CMOS networks obtained by connecting the output of one network to an input of another network. Whereas such an interconnnection is well understood as far as gate networks are concerned, the composition of MOS networks poses a new problem whenever an interconnection

is bidirectional. We first consider some logical aspects of such compositions, and then relate them to circuit concepts.

Let Γ^1 and Γ^2 be C-graphs, let A be an input variable of Γ^2 and let Z^1 be the output node of Γ^1. Without loss of generality we assume that Γ^2 has at most one input node labeled A, since any two such nodes can be identified without changing the behavior of Γ^2. The *composition* of Γ^1 and Γ^2 *with respect to* A is defined to be the C-graph $\Gamma = \Gamma^1[A]\Gamma^2$, where

(a) If there is no input node labeled A in Γ^2 then Γ is the union of Γ^1 and Γ^2 with all A-labels in Γ^2 replaced by Z^1. (In this case Γ may be viewed as a "cascade" of the C-graphs Γ^1 and Γ^2.

(b) If there is an input node labeled A in Γ^2, the graph Γ is now obtained from the union of Γ^1 and Γ^2 by identifying the output node of Γ^1 with the input node labeled A of Γ^2, and relabeling the new node Z^1 and the edges of Γ^2 as in (a).

The output node of Γ is Z^2 — the output node of Γ^2.

A cascade of two cells may be viewed as the composition of the component cells. For example, the composition of Γ^1 and Γ^2 of Figure 6.1 with respect to C yields the C-graph of Figure 5.1(a). This example illustrates case (a) in the definition of C-graph composition. Case (b) is illustrated in Figure 6.2. The C-graph of Figure 6.2(c) is the composition $\Gamma^1[C]\Gamma^2$, where Γ^1 and Γ^2 are shown in Figures 6.2(a) and (b), respectively. Figure 6.2(c) essentially coincides with Figure 5.3; thus Figure 6.2(c) also implements $A \oplus B$. Note that Γ^2 of Figure 6.2(b) is not combinational; nevertheless $\Gamma^1[C]\Gamma^2$ *is* combinational.

In the sequel we establish conditions for the composition of C-graphs to be combinational. These conditions will then be applied to formulate design rules for the implementation of boolean functions. Consider again Γ^2 of Figure 6.2(b). Although Γ^2 is not combinational, its output Z^2 is boolean for every binary input vector satisfying the condition $C=A'$. Indeed, under the condition $C=A'$, we have $Z^2=A \oplus B$. This observation motivates the following definitions.

Let Γ be a C-graph with $\mathbf{A} = A^1, \ldots, A^n$ as the vector of input letters and let f be a boolean function over $\mathbf{A}$. A binary n-tuple $(a^1, \ldots, a^n)$ *satisfies* f iff $a^1 = f(a^1, \ldots, a^n)$.

The C-graph Γ is *combinational* (T-*combinational*) *with respect to* f *and* A^1 iff each binary input n-tuple satisfying f is key-combinational and the corresponding output is boolean (tri-state).

Let Γ^1 and Γ^2 be two C-graphs over $\mathbf{A}$ and let $\Gamma = \Gamma^1[A^1]\Gamma^2$. Let Γ^1 be boolean with output function f^1, and let Γ^2 be combinational with respect to f^1 and A^1. Then Γ^1 and Γ^2 are said to be *composable with respect to* A^1. If Γ is combinational, one verifies that the output of Γ is

$$Z = f^2(f^1(\mathbf{A}), A^2, \ldots, A^n),$$

where f^2 is any boolean function satisfying the condition $Z^2 = f^2(a^1, \ldots, a^n)$

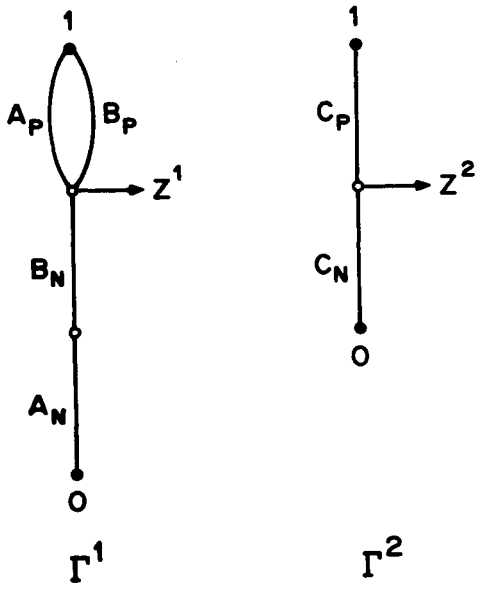

Figure 6.1 Illustrating composition of cells.

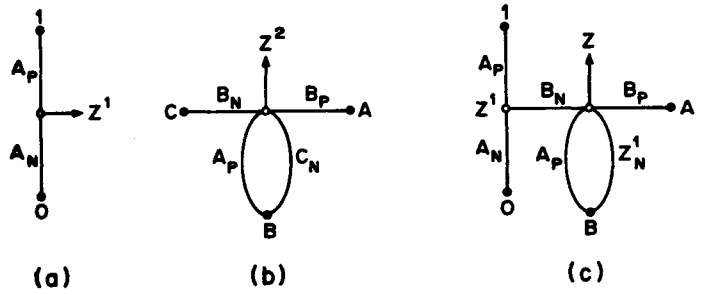

Figure 6.2 (a) Γ^1 ; (b) Γ^2 ; (c) $\Gamma^1[C]\Gamma^2$.

for every n-tuple $(a^1, \ldots, a^n)$ satisfying f^1.

In the following proposition, which is easily verified, we state a sufficient condition for the composition of two C-graphs to be combinational.

Proposition 6. Let Γ^1 and Γ^2 be two C-graphs over **A** which are composable with respect to A^1. Assume that no input node of Γ^2 is labeled A^1. Then $\Gamma^1[A^1]\Gamma^2$ is combinational.

Evidently a cascade $\Gamma = \Gamma^1, \ldots, \Gamma^k$ of cells may be viewed as the

composition $\Gamma^1[Z^1]\hat{\Gamma}$, where $\hat{\Gamma}$ is the C-graph formed by the subcascade $\Gamma^2, \ldots, \Gamma^k$. If Γ is boolean, one verifies that Γ^1 and $\hat{\Gamma}$ are composable with respect to Z^1. Conversely, if Γ^1 and $\hat{\Gamma}$ are composable with respect to Z^1, then their composition is combinational in view of Proposition 6. It follows that the cascade Γ is boolean.

A method for implementing arbitrary boolean functions by means of transmission gates is based on the following proposition, which is also easily verified.

Proposition 7. Let $f(A^1, \ldots, A^n)$ be a boolean function and let

$$f(A^1, \ldots, A^n) = (A^1)' f^1 + A^1 f^2$$

be its decomposition with respect to A^1, i.e. f^1 and f^2 are boolean functions on $A^2, \ldots, A^n$. Let Γ^1 and Γ^2 be arbitrary implementations of f^1 and f^2, respectively. Let Γ^3 be the C-graph shown in Figure 6.3. Then the C-graph $\Gamma = \Gamma^2[H](\Gamma^1[G]\Gamma^3)$ is combinational and realizes f.

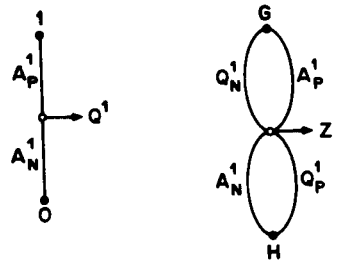

Figure 6.3 C-graph Γ^3.

Our final example of this section illustrates a circuit that cannot be explained in our model, since its design requires circuit considerations beyond the switch level approach. In Figure 6.4 we show the XOR gate of Figure 6.2(c), in which the input nodes A and B are driven by the outputs of inverters. Consider the input-key state $A = 1$, $B = 0$, $\overline{A} = \overline{B} = Q = X$. Node $\overline{A}$ has an N-path to 0 (A_N), but also a potential path $(\overline{B}_P, \overline{B}_N, \overline{A}_P)$ to 1. According to our model $E(\overline{A}) = X$. Similarly, one verifies that $\overline{B}$ has a P-path to 1 (B_P), and a potential path $(\overline{A}_P, \overline{B}_N, \overline{A}_N)$ to 0, giving $E(\overline{B}) = X$. Also $E(Q) = X$. Therefore this input vector is not key-combinational, and the circuit is not accepted by our model. Nevertheless the circuit can be properly designed by choosing appropriate circuit parameters. For instance, the resistance of the path through transistor A_N from $\overline{A}$ to 0 would be much smaller than the resistance of the potential path $(\overline{B}_P, \overline{B}_N, \overline{A}_P)$ to 1, causing $\overline{A}$ to become 0, etc.

In general, whenever our model accepts a circuit, additional restrictions may have to be imposed, e.g. no more than n transistors in series or no more than k cells in a cascade, etc. A circuit that satisfies our model and such restrictions should be relatively easy to implement. On the other hand, a circuit which

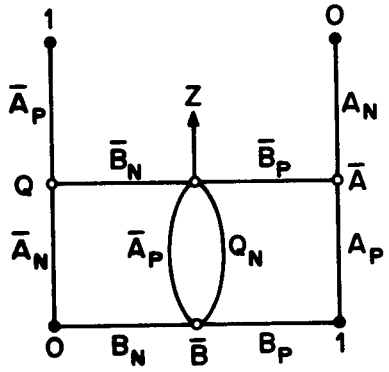

Figure 6.4 Another XOR gate.

is rejected by our model may require more effort on the part of the circuit designer to make it work.

7. Design Examples

In this section we illustrate the applicability to design of the theoretical aspects discussed above.

As already pointed out, any boolean function can be implemented by a cascade of separated cells, each cell implementing a negative function. Examples of such implementations have been given in Section 4. As a further example, consider the function $Z = AB \oplus C$. The minimal sum expression for Z, namely

$$Z = ABC' + A'C + B'C, \qquad (*)$$

yields the 2-level NAND expression

$$Z = NAND\ [NAND(A,B,C'),\ NAND(A'C),\ NAND(B'C)].$$

The corresponding cascade of separated cells is highly inefficient: it consists of 7 cells (3 inverters and 4 NAND cells) and its transistor count is 26.

Any boolean function which can be represented by a single product of literals, can be decomposed into two negative functions. For example $U = A'B'CDE$ becomes $U = A'B'V'$, where $V = (CDE)'$. Applying this approach to each product of the above minimal sum expression (*) we obtain:

$$\begin{aligned} Z &= (Z^1)'C' + A'(Z^2)' + B'(Z^2)' \\ &= (Z^1)'C' + (A'+B')(Z^2)', \end{aligned}$$

where $Z^1 = (AB)'$ and $Z^2 = C'$. This decomposition yields a cascade of 3 separated cells, with a transistor count of 16.

The implementation of the function $Z = AB \oplus C$ can be further improved. Consider the decomposition:

$$Z^1 = (AB+C)',$$
$$Z = Z^2 = (Z^1+ABC)'.$$

The corresponding cascade has a transistor count of 14, and is minimal in the number of separated cells. An algorithm for finding such decompositions is described in [M].

Another universal design method is based on the concepts of transmission gates and composition of C-graphs. Consider the decomposition of a given boolean function $f(A^1, \ldots, A^n)$ with respect to one of its variables, say A^1 :

$$f(A^1, \ldots, A^n) = (A^1)' f^1 + A^1 f^2,$$

where f^1 and f^2 are boolean functions on $A^2, \ldots, A^n$. Let Γ^1 and Γ^2 be arbitrary implementations of f^1 and f^2, respectively. Proposition 7 indicates how f can be realized from Γ^1 and Γ^2 using the above decomposition. Note that this design step may now be applied to Γ^1 and Γ^2. For example, the function $Z = AB \oplus C$ may be implemented by applying the above decomposition step twice, yielding the C-graph of Figure 7.1. This design is based on the decompositions

$$AB \oplus C = A'C + A(B \oplus C),$$
$$B \oplus C = B'C + BC'$$

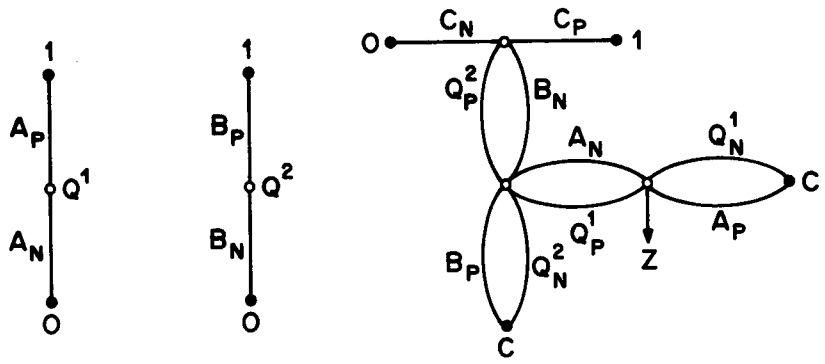

Figure 7.1 A C-graph for $AB \oplus C$.

It contains 4 transmission gates, controlled respectively by A, $A' = Q^1$, B, and $B' = Q^2$. The transistor count of this implementation is again 14.

The universal design methods described above can be improved by combining them with various heuristic design techniques. For example, the design shown in Figure 7.1 uses an 8-transistor implementation for the auxiliary function $B \oplus C$. However, this implementation of $B \oplus C$ can be replaced by the 6-transistor implementation for XOR, shown in Figure 5.3. Thus the overall transistor count is reduced from 14 to 12.

8. Concluding Remarks

We have considered the class of static combinational CMOS networks. For this class we have developed a mathematically precise switch-level model, namely the C-graph and its associated analysis techniques. Given any circuit diagram of a static CMOS network, one easily obtains the corresponding C-graph. The behavior of such a C-graph can be analyzed using the methods developed here; one can determine whether the C-graph is combinational and, if so, find the boolean function realized by the network. If this is done, one does not need switch-level simulation to verify that the circuit is logically correct. If the given C-graph has the structure of a cascade of cells, considerably simpler methods can be applied, as discussed in Section 4.

A novel feature of our analysis of C-graphs is the fact that intermediate voltage values at input and internal nodes can be properly handled. This permits us to distinguish between unknown binary values and intermediate values (X), as illustrated by the two implementations of the AND function in Section 5. The use of the third value X also permits us to handle unknown intermediate conditions.

With regard to design, our models form a basis for several synthesis approaches: (a) using negative-function decompositions, (b) using the decomposition technique (Figure 6.3) based on transmission gates, and (c) using a mixture of (a) and (b) as well as "tricky" building blocks like the 6-transistor XOR gate. These approaches are based on logic as opposed to circuit design, and may need to be modified by restrictions imposed by the circuit designer. The reader interested in design techniques is referred to [WE , M 86] for futher discussion.

The methods presented here can be extended to static asynchronous sequential CMOS networks, along the lines of [YB 84 , YB 85]. Another promising research direction is the generalization of our model to dynamic CMOS. Finally, further work is required to improve the efficiency of the transient analysis of C-graphs (Section 5).

Acknowledgement

The authors wish to thank Professors R. E. Bryant of Carnegie-Mellon University and T. R. Viswanathan of the University of Waterloo for many helpful comments.

REFERENCES

[B] Bryant, R. E., "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Trans. Computers C-33*, 2, February 1984, 160-177.

[BY 76] Brzozowski, J. A., and Yoeli, M., *Digital Networks,* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976.

[BY 79] Brzozowski, J. A., and Yoeli, M., "On a Ternary Model of Gate Networks," *IEEE Trans. Computers C-28,* 3, March 1979, 178-184.

[C] Caldwell, S. H., *Switching Circuits and Logical Design,* John Wiley, 1958.

[H] Hayes, J. P., "A Unified Switching Theory with Applications to VLSI Design," *Proc. IEEE 70,* 10, October 1982, 1140-1151.

[M 86] Mukherjee, A., "*Introduction to nMOS and CMOS VLSI Systems Design*", Prentice-Hall, Inc., 1986.

[M] Muroga, S., *VLSI System Design,* John Wiley, 1982.

[WE] Weste, N., and Eshraghian, K., *Principles of CMOS VLSI Design,* Addison-Wesley, 1985.

[YB 84] Yoeli, M., and Brzozowski, J. A., "A Mathematical Model of Digital CMOS Networks", Research Report No. CS-84-22, Department of Computer Science, University of Waterloo; August 1984.

[YB 85] Yoeli, M., and Brzozowski, J. A., "A Mathematical Model of Digital CMOS Networks," *1985 Canadian Conference on VLSI,* Toronto, Ontario; November 1985, 117-120.