

A LOGIC DESIGN THEORY FOR VLSI*

by

John P. Hayes

Digital Integrated Systems Center
andDepartments of Electrical Engineering and Computer Science
University of Southern California
Los Angeles, California 90007ABSTRACT

Classical switching theory fails to account for some key structural and logical properties of the transistor circuits used in VLSI design. This paper proposes a new logic design methodology called CSA theory which is suitable for VLSI. Three kinds of primitive logic devices are defined: connectors (C), switches (S), and attenuators (A); the latter have the characteristics of pullup/pulldown components. It is shown that four new logic values are required, in addition to the usual Boolean 0 and 1 values. These values introduce a concept of gain or drive capability into logic design; they also account for the high-impedance state of tri-state devices. The elements of CSA theory and its application to some basic VLSI design problems are described. It is demonstrated that CSA theory provides a more powerful and more rigorous replacement for the mixed logic/electronic methods currently used in VLSI design.

* This research was supported by the National Science Foundation under Grant No. MCS78-26153, and by the Naval Electronic Systems Command under Contract No. N00039-80-C-0641.

1. INTRODUCTION

The development of very large-scale integrated (VLSI) circuits using the philosophy espoused by Mead and Conway [1] and others involves a complex interplay of various design techniques at the electronic, logical and systems levels. These techniques are ad hoc for the most part, with the result that the VLSI designer is mainly guided by experience rather than theory. It might be expected that the large body of results in switching theory and logic design that has accumulated over the past 40 years can readily be applied to VLSI design, at least for analysis purposes if not for synthesis. This does not appear to be the case, however. Several reasons may be cited for this.

(1) The basic component of VLSI circuits is the MOS transistor whose logical behavior is that of a three-terminal digital switch. Neither of the classical models from switching theory, branch-type networks (also called contact networks) or gate-type networks [2], adequately capture the structure or logical behavior of MOS transistor circuits. The primitive components of gate-type circuits are logic gates which allow signal transmission in one direction only. An MOS transistor, on the other hand, is inherently bidirectional. The components of branch-type networks are (relay) contacts. A contact is bidirectional, but unlike a transistor, it is basically a two-terminal device.

(2) Classical switching theory hides some types of logic devices that have a significant impact on integrated circuit design and layout. For example, it does not recognize the important role played by connectors in logical behavior. Connections to power and ground are omitted from standard logic diagrams, yet they are the sources of the logical 0 and 1 values on which the logical operation of all circuits depends. The selection and layout of connectors is a central issue in VLSI design. Components like amplifiers and pullup/pulldown loads, which are crucial to proper logical or digital operation, are also invisible at the standard logic level. To see these devices we must move to the more detailed electronic or analogue level.

(3) Only the two logical values 0 and 1 are recognized in standard switching theory. However, in modern design practice extensive use is made of

at least one additional logic value, the high-impedance state Z . Indeed it has been suggested that MOS technology is inherently a three-state technology [3].

At the present time, the usual remedy for the foregoing difficulties is to combine design methods from switching theory and electronic circuit theory heuristically. This results in "mixed" circuit diagrams which couple logic gates, transistors, etc. in a manner that, strictly speaking, is meaningless. It also causes important logic design techniques such as wired logic and tri-state logic to be treated as anomalous special cases.

In this paper a new logic design theory is introduced that attempts to overcome the difficulties cited above. The key components of this theory are connectors (C), switches (S) and attenuators (A); we therefore refer to it as *CSA theory*. A switch here is a three-terminal device that can accurately model the digital operation of a PMOS or NMOS transistor. An attenuator models a pullup or pulldown load; it has no counterpart in classical switching theory. Central to our approach is a six-valued logic which, in addition to the usual "strong" Boolean values 0 and 1, has "weak" versions of these values denoted by $\tilde{0}$ and $\tilde{1}$. This weak/strong signal dichotomy allows the electronic concepts of signal amplification and attenuation to be transferred to the logic level. The high-impedance state Z is also treated as an explicit logic value. CSA theory provides a uniform and consistent alternative to the mixed design approach mentioned earlier. It can be used to analyze both branch- and gate-type networks, as well as such nonclassical structures as wired logic and tri-state logic.

Section 2 presents an informal development of the basic concepts of CSA theory. In Sec. 3 these ideas are presented in more rigorous and complete fashion. Finally in Sec. 4, CSA theory is applied to the design of a simple but important class of circuits, namely inverters.

2. INFORMAL DEVELOPMENT

In this section the basic concepts of connector and switch are examined in detail. We show that six logical values are needed for an adequate description of their behavior, as well as a new logic device which we call an attenuator.

Connectors

In classical switching theory the only logical operation associated with a connector or wire is the trivial identify function $v \rightarrow v$. At the CSA level of complexity, connectors are seen as the fundamental devices for performing nontrivial operations of the AND and OR type. To demonstrate this, we first need a precise definition of a connector and its behavior. A *terminal* T is a designated connection point in a network; it is denoted by a black dot in logic diagrams as shown in Fig. 1a. A *simple connector* is a continuous conducting path between two terminals. It may represent a metal, diffusion or polysilicon conductor in an integrated circuit, and is represented by a line as in Fig. 1b. A (complex) *connector* is a linked set of simple connectors; Fig. 1c shows an example. Any point in a connector may be designated a terminal, therefore a connector can be viewed as simply a sequence of contiguous terminals.

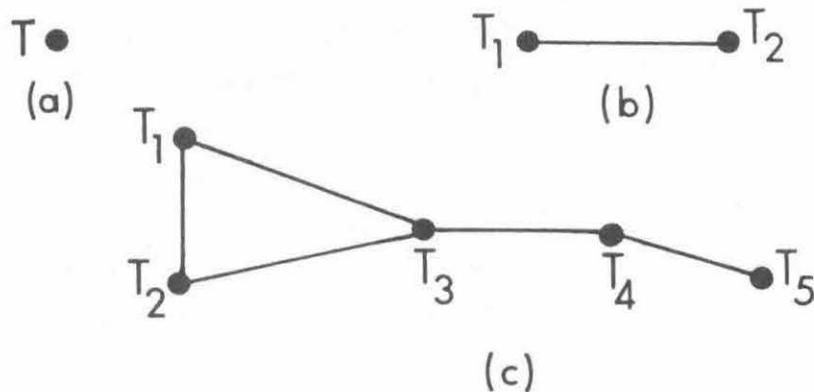


Fig. 1 (a) A terminal T . (b) A simple connector. (c) A general connector C .

Let V be the set of logic values or signals of interest. V contains the usual Boolean constants 0 and 1; additional values will be added later. With every connector C we associate a set of *input values* $v_{in}(C)$ taken from V . The $v_{in}(C)$ values are typically derived from external signal sources that are connected to C . Thus in the connector of Fig. 2a the external signal sources are indicated by arrows, and the input signal set is $v_{in}(C) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$.

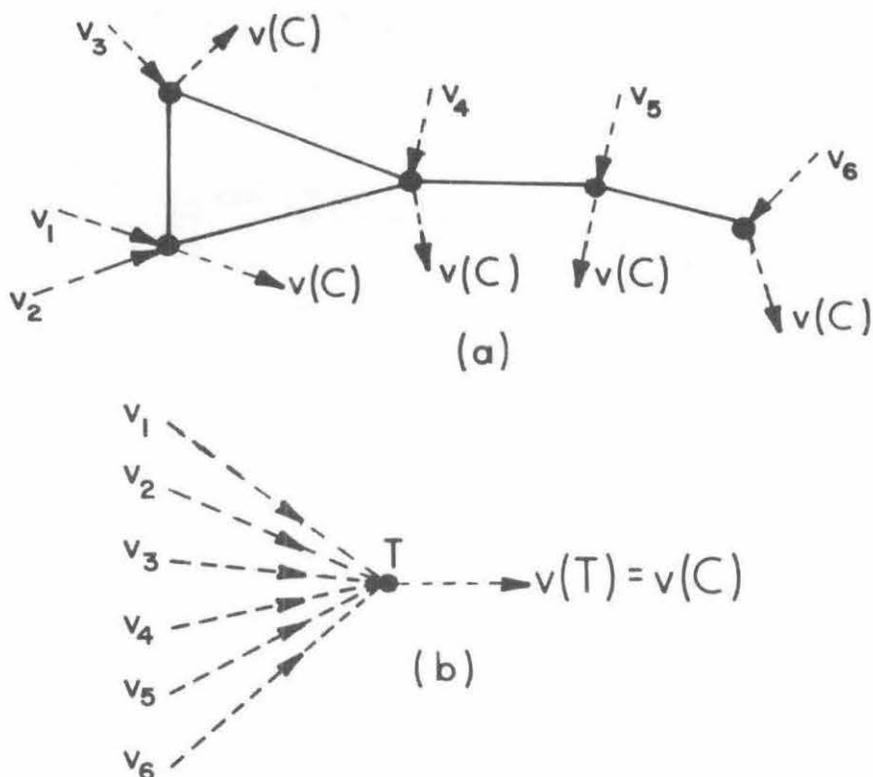


Fig. 2 (a) Input-output signals of the connector C . (b) An equivalent terminal T .

While several different input values may be applied to a connector simultaneously, we assume that the connector produces a unique *output value* $v(C)$ at all its terminals, where $v(C) \in V$. Thus if the physical signals associated with C are voltages, then C has the equipotential property of a perfect electrical conductor. It follows that a complex connector can always be replaced by a single terminal as illustrated in Fig. 2b.

Suppose that the input values $v_1, v_2 \in V$ are applied to connector C . For logical consistency and completeness, we require $v(C)$ to be defined uniquely for all possible combinations of v_1 and v_2 . We can write

$$v(C) = \#(v_1, v_2)$$

where $\#$ denotes the *connection function* implemented by C . Let v_1 and v_2 assume the values 0 and 1. If $v_1 = v_2$, then we expect the following equations to hold:

$$\begin{aligned} \#(0,0) &= 0 \\ \#(1,1) &= 1 \end{aligned} \quad (1)$$

If $v_1 \neq v_2$ is allowed (this is normally considered to be improper behavior in binary switching circuits), then we need a third logic value which we denote U. U (for unknown) has frequently been used in logic simulation programs to model signal values during transitions between 0 and 1, and the values associated with uninitialized states [4]. We use U in the sense of a conflict value that results in the connector behavior defined by the following set of equations:

$$\#(0,1) = \#(0,U) = \#(1,U) = \#(U,U) = U \quad (2)$$

Next we consider the notion of a switch as a controlled connector, and show that it requires the introduction of three additional logic values.

Switches

A *switch* S is defined here as a three-terminal device with a "control" terminal K and two symmetric "data" terminals D_1 and D_2 . It is represented by the circuit symbol of Fig. 3a. The set of values $V(D)$ is assigned to D_1 and D_2 . The set $V(K)$ containing the two values ON and OFF is assigned to K. Later we will equate $V(K)$ and $V(D)$. When $v(K) = \text{ON}$, D_1 and D_2 are joined by a connector as in Fig. 3b. When $v(K) = \text{OFF}$, there is no connection between D_1 and D_2 via the switch. Examples of switches that can be easily made to conform to this model are a manual on-off switch, a single-contact relay, and an MOS transistor.

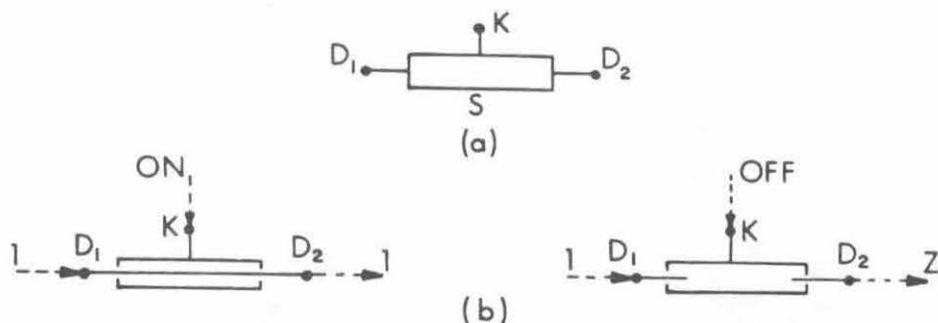


Fig. 3 (a) An isolated switch S, and (b) its behavior.

Suppose that an isolated switch S is to be used to control the signal value appearing at one of its data terminals, say D_2 . Intuitively the following type of behavior is expected:

$$v(K) = \text{ON} \quad \text{implies} \quad v(D_2) = 1 \quad (3)$$

$$v(K) = \text{OFF} \quad \text{implies} \quad v(D_2) = 0 \quad (4)$$

If $v(K) = \text{ON}$, then $v(D_1) = v(D_2)$, so we can satisfy (3) by applying the constant 1 to D_1 as shown in Fig. 3b. When $v(K) = \text{OFF}$, however, D_2 becomes an isolated terminal, and it "floats" to a value that is distinct from 0, 1 and U. We therefore introduce a new logical value Z to denote $v(C)$ when C is an isolated connector. Z corresponds to the usual high-impedance state of tri-state logic. It is a weak value in the sense that it can be overridden by each of the logic values 0, 1 and U. This suggests that Z should satisfy the following set of equations:

$$\begin{aligned} \#(0, Z) &= 0 \\ \#(1, Z) &= 1 \\ \#(U, Z) &= U \\ \#(Z, Z) &= Z \end{aligned} \quad (5)$$

To satisfy (4) above, we can attempt to apply to D_1 an external signal v that forces $v(D_2)$ to 0. Thus when $v(K) = \text{OFF}$, we require

$$\#(v, Z) = 0 \quad (6)$$

To satisfy (3) at the same time requires

$$\#(v, 1) = 1 \quad (7)$$

assuming that 1 has been applied to the input data terminal D_1 . It is easily seen that none of the values 0, 1, U, Z can satisfy equations (1), (2), (5), (6) and (7) simultaneously. Thus we introduce a fifth logical value denoted

$\tilde{0}$ which, like 0, is an acceptable "0-like" value in the realization of Boolean functions. If we replace (6) and (7) by

$$\#(\tilde{0}, Z) = \tilde{0}$$

and

$$\#(\tilde{0}, 1) = 1 \quad (8)$$

respectively, no contradiction results. Now (8) implies that 1 overrides $\tilde{0}$ when both are applied to the same connector, hence $\tilde{0}$ is a weak 0-like value, that is, a value with low (logical) drive capability. In a similar manner, we define a weak 1-like value denoted $\tilde{1}$. The foregoing analysis suggests that $\tilde{0}$ and $\tilde{1}$ should satisfy the following set of equations involving the connection operator #:

$$\begin{aligned} \#(\tilde{0}, Z) &= \#(\tilde{0}, \tilde{0}) = \tilde{0} \\ \#(\tilde{1}, Z) &= \#(\tilde{1}, \tilde{1}) = \tilde{1} \\ \#(\tilde{0}, 0) &= \#(\tilde{1}, 0) = 0 \\ \#(\tilde{0}, 1) &= \#(\tilde{1}, 1) = 1 \\ \#(\tilde{0}, \tilde{1}) &= \#(\tilde{0}, U) = \#(\tilde{1}, U) = U \end{aligned} \quad (9)$$

Attenuators

We have just seen that if a switch is used to transmit 0-like and 1-like signals, we need two new values $\tilde{0}$ and $\tilde{1}$ that can be applied externally to its data terminals. We now define a new logic element called an attenuator whose function is to generate $\tilde{0}$ and $\tilde{1}$. An *attenuator* is a unidirectional two-terminal device whose output is $\tilde{0}$ or $\tilde{1}$ when 0 or 1 respectively are applied to its input terminal. Figure 4 shows the symbol used for an attenuator, as well as its typical use to force the output of a switch to have values from the set $\{0, 1, \tilde{0}, \tilde{1}\}$. The circuit of Fig. 4 is thus a complete switching circuit that meets the original behavior specifications suggested by (3) and (4).

It is apparent from Fig. 4 that an attenuator is a device that can pull an isolated connector up (from Z to $\tilde{1}$) or down (from Z to $\tilde{0}$). It thus models

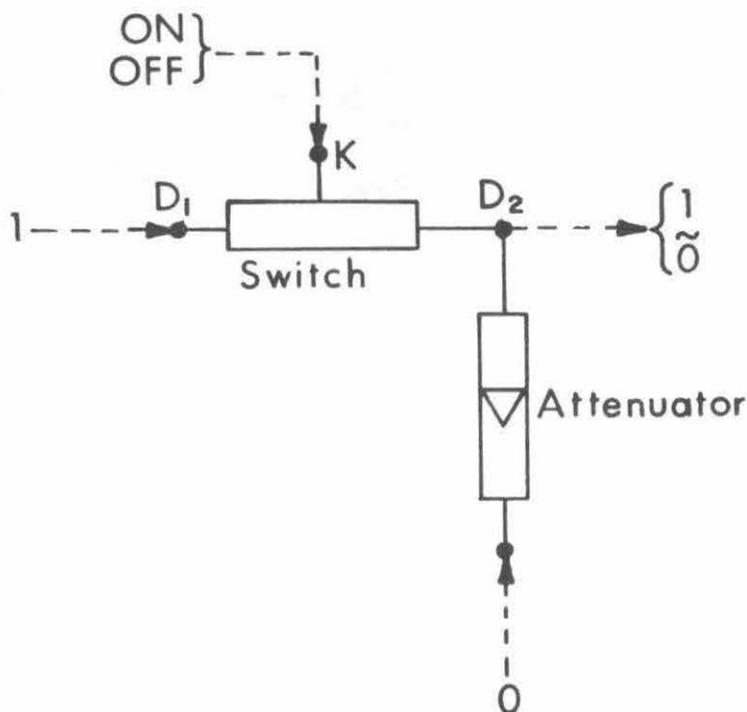


Fig. 4 Typical application of an attenuator.

the behavior of a pullup or pulldown device in an electronic circuit. This may be a resistor or, in the case of VLSI circuits, a load transistor. Since it also converts "strong" to "weak" signals, an attenuator can be regarded as a digital impedance.

The final primitive component we need is an amplifier that converts $\tilde{0}$ and $\tilde{1}$ to 0 and 1 respectively. Standard amplifying devices perform this function satisfactorily, hence we denote our amplifiers by the standard triangle symbol of Fig. 5a. Note that an attenuator is the inverse of an amplifier, a fact that guided our choice of symbol for an attenuator. The attenuator symbol contains a reversed amplifier, and also suggests an impedance or load element. In accordance with normal logic design practice we insert a small circle in a line to denote the following nonamplifying inversion operation:

$$1 \rightarrow 0, 0 \rightarrow 1, \tilde{1} \rightarrow \tilde{0}, \tilde{0} \rightarrow \tilde{1}, U \rightarrow U, Z \rightarrow Z.$$

Thus an inverting amplifier can be represented as shown in Fig. 5b.

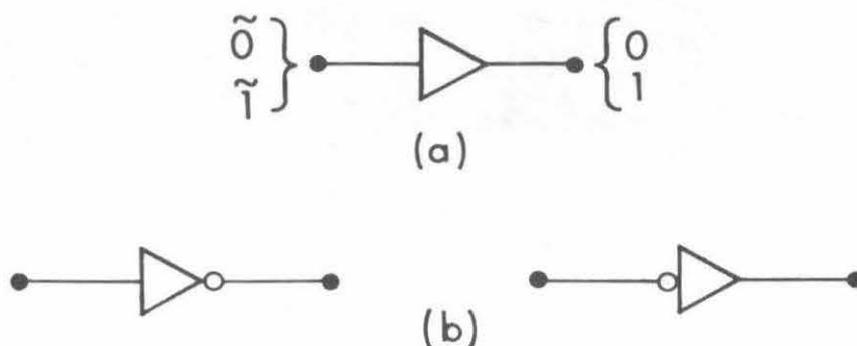


Fig. 5 (a) A non-inverting amplifier. (b) Inverting amplifiers.

3. THEORY

We now present a formal description of CSA theory. A CSA network is composed of four basic component types: n -terminal connectors, three-terminal switches, and two-terminal amplifiers and attenuators. Components are connected via their terminals, where a terminal is the simplest connector. The behavior of a network is determined by the output signal values of its terminals. A set of six logical signal values is recognized: $V_6 = \{0, 1, \tilde{0}, \tilde{1}, U, Z\}$. The behavior of all CSA component types is completely defined in terms of V_6 .

Connection Function

It is useful to introduce a concept of relative strength among the members of V_6 .

Definition 1: Let $v_1, v_2 \in V_6$. v_1 is (logically) stronger than v_2 , denoted $v_1 \cong v_2$, if $\#(v_1, v_2) = v_1$ where $\#$ is defined by Eqns. (1), (2), (5) and (9).

The relation \cong imposes a partial ordering on V_6 which is depicted graphically in Fig. 6. Clearly U is stronger than all other member of V_6 , while Z is weaker than all other values. The values 0 and 1 are not related by \cong

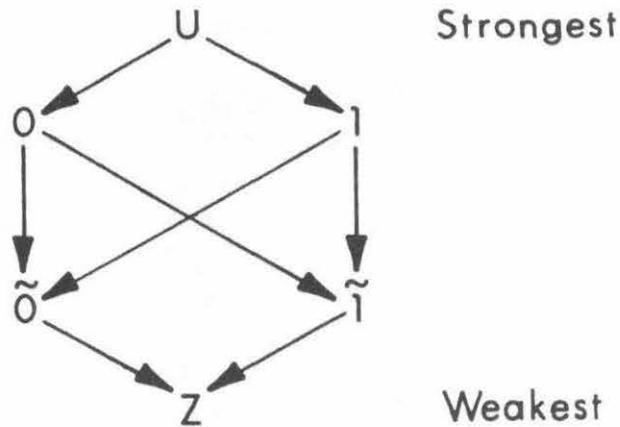


Fig. 6 Relative strength of the logic values in V_6 .

because $\#(0,1) = U$; these values are said to be *contradictory*. Similarly $\tilde{0}$ and $\tilde{1}$ are contradictory. Using the foregoing notions, we can now generalize Eqns. (1), (2), (5) and (7) to obtain the following concise definition of the behavior of a connector.

Definition 2: (The k -place *connection function* $\#$) Let C be a connector to which the input signals $v_1, v_2, \dots, v_k \in V_6$ are applied; cf. Fig.2. C generates a unique output signal $v(C) = \#(v_1, v_2, \dots, v_k) \in V_6$ defined as follows. If v_1, v_2, \dots, v_k contain no contradictory values, then

$$\#(v_1, v_2, \dots, v_k) = v_i$$

where $v_i \cong v_j$ for all $j = 1, 2, \dots, k$. If v_1, v_2, \dots, v_k contain contradictory values, then

$$\#(v_1, v_2, \dots, v_k) = U.$$

The action of $\#$ on the members of V_6 determines the interpretation of these logical quantities in practical digital circuits. Z is the logical value of an isolated connector, and also corresponds precisely to the high-impedance state used in tri-state circuits. 0 and 1 correspond to the usual Boolean variables 0 and 1 . Here, however, they are seen as strong signals that can override their weaker counterparts $\tilde{0}$ and $\tilde{1}$. Thus 0 and 1 denote sig-

nals with high drive capability, such as power, ground, and amplifier output signals. $\tilde{0}$ and $\tilde{1}$ represent weak signals that have relatively low drive capability; such signals are typically produced by passive load devices. Note that the $\tilde{0}$ and $\tilde{1}$ signals are easily mapped onto 0 and 1 respectively by passing them through an amplifier. Thus from the viewpoint of implementing Boolean functions, we may choose either 0 or $\tilde{0}$ to represent Boolean zero, and either 1 or $\tilde{1}$ to represent Boolean one. U represents a conflict resulting from the simultaneous application of contradictory Boolean values of equal strength to a connector. U is not normally encountered in properly designed or "well-behaved" circuits.

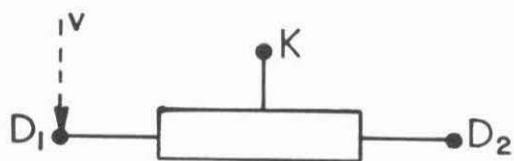
The standard Boolean operations AND and OR which do not require inversions can be implemented by means of a connector alone. Suppose, for example, that k devices have their output terminals z_1, z_2, \dots, z_k joined by a connector C . Let the values v_1, v_2, \dots, v_k applied to C via the z_i terminals be restricted to the subset $\{\tilde{0}, 1\}$ of V_6 . Then $v(C)$, which is defined by Def. 2, implements the OR function. This is because a (strong) 1 applied to any terminal of C overrides a (weak) $\tilde{0}$ applied to any other terminal of C . Similarly, when the v_i 's are confined to $\{0, \tilde{1}\}$, C implements the AND function. It is believed that this type of "wired logic" underlies the behavior of all switching circuits, including both branch and gate-type circuits.

CSA Networks

A switch, as indicated by Fig. 3, contains a control terminal K and two symmetric data terminals D_1 and D_2 . The switch is *homogeneous* if the values that all three terminals can assume are identical, that is, $v(K) = v(D)$. A manual on-off switch is not homogeneous because $v(K)$ and $v(D)$ are defined in incompatible mechanical and electrical domains, respectively. An MOS transistor is homogeneous if the K input (the gate) and the D inputs (the source and drain) all employ the same digital voltage levels. An inhomogeneous switch is useful as a transducer between physically incompatible signal domains. Here we will restrict our attention to homogeneous switches where all three terminals may assume values from V_6 .

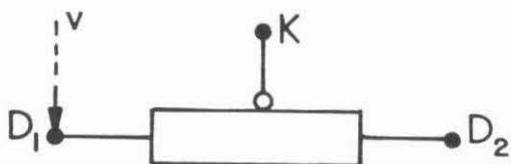
Figure 7 defines the behavior of the most basic switch in terms of V_6 . The switch is turned on and off by the K values 1 and 0, respectively; there is

no amplification or gain associated with this switch. It is also convenient to treat the other three switches of Fig. 7 as basic components of CSA networks. In the switch of Fig. 7b the polarity of the K terminal is reversed. The switches of Figs. 7c and 7d can be switched on or off by the weak control signal values $\tilde{1}$ and $\tilde{0}$, thus they have built-in gain. These switches directly model the digital behavior of NMOS and PMOS transistors, respectively. In all cases we make the somewhat arbitrary assumption that $v(D_1) = v(D_2) = U$ when either Z or U is applied to K. It should be noted that other definitions of switch behavior may be more appropriate for some technologies or circuit configurations. New switch types can be added to those of Fig. 7 as required.



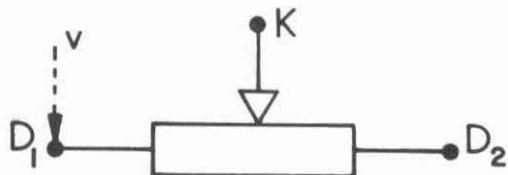
$v(K)$	$v(D_1)$	$v(D_2)$	Switch State
0	v	Z	Off
1	v	v	On
$\tilde{0}, \tilde{1}, U, Z$	U	U	Undefined

(a)



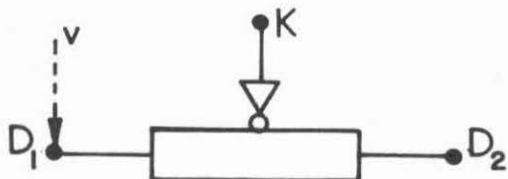
$v(K)$	$v(D_1)$	$v(D_2)$	Switch State
1	v	Z	Off
0	v	v	On
$\tilde{0}, \tilde{1}, U, Z$	U	U	Undefined

(b)



$v(K)$	$v(D_1)$	$v(D_2)$	Switch State
$\tilde{0}, 0$	v	Z	Off
$\tilde{1}, 1$	v	v	On
U, Z	U	U	Undefined

(c)



$v(K)$	$v(D_1)$	$v(D_2)$	Switch State
$\tilde{1}, 1$	v	Z	Off
$\tilde{0}, 0$	v	v	On
U, Z	U	U	Undefined

(d)

Fig. 7 Four basic switch types and their behavior.

Figure 8 formally defines attenuators and amplifiers. It is easily shown that we can dispense with explicit amplifiers if amplifying switches of the kind appearing in Figs. 7c and 7d are available.

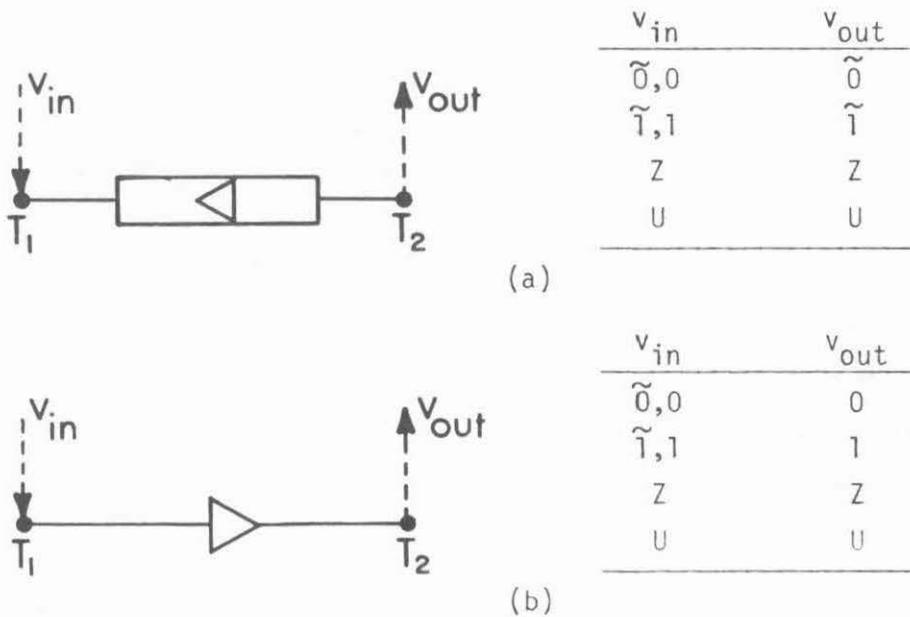


Fig. 8 Definition of (a) an attenuator, (b) an amplifier.

A *CSA network* may be defined as a set of switches, attenuators and amplifiers linked by connectors. The interconnection rules that yield well-behaved networks can be defined in several ways depending on the application at hand. In most cases we require network output signals to be confined to the subset $\{0, 1, \tilde{0}, \tilde{1}\}$ of V_6 . Z is a tolerable output value also, since an attenuator can be used to convert it to $\tilde{0}$ or $\tilde{1}$. Only U represents an intolerable conflict state that cannot be overridden. We can generalize the concept of a CSA network to include as primitive components or cells, any devices whose behavior can be described as a function on V_6 . This includes all standard combinational and sequential circuits that are defined on the Boolean subset $\{0, 1\}$ of V_6 . Thus we can easily add high-level primitives like PLAs, decoders, registers, etc. to CSA-type circuits.

Figure 9 shows two simple CSA networks that implement the logical OR operation defined on $\{0, 1\}$. That of Fig. 9a uses a parallel configuration of switches which is the standard branch (contact network) implementation of OR;

the attenuator makes the circuit act as a unidirectional OR gate. As this example suggests, both branch-type and gate-type networks can be modeled efficiently by means of CSA networks. However, CSA networks also include many useful structures that are not covered by classical switching theory; the OR network of Fig.9b is an example.

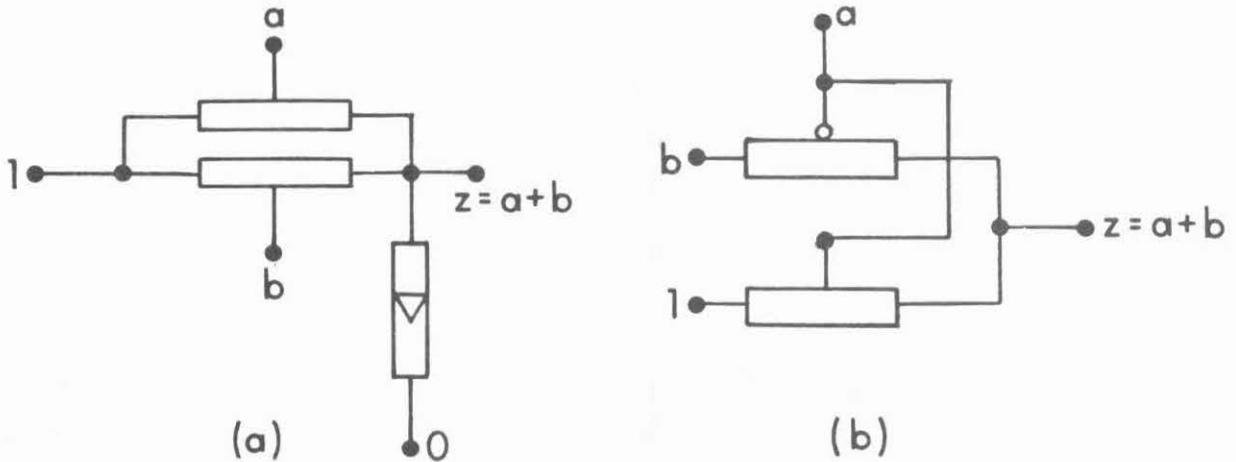


Fig. 9 Two CSA implementations of the OR operation.

Characteristic Equations

As in standard logical design it is useful to be able to analyze CSA networks by means of algebraic functions and equations. While it is possible to represent CSA network behavior directly in terms of 6-valued functions such as $\#$, it seems to be more useful to employ two-valued Boolean functions. To this end we associate a Boolean function $z^v(C)$ with each connector C such that $z^v(C) = 1$ whenever $v(C) = v \in V_6$, and $z^v(C) = 0$ otherwise. The behavior of C can therefore be fully defined by means of the six functions $z^0(C)$, $z^1(C)$, $z^{\bar{0}}(C)$, $z^{\bar{1}}(C)$, $z^Z(C)$, $z^U(C)$, which we term the *characteristic (Boolean) functions* of C . Characteristic functions for the basic CSA elements are easily derived. They can be described by truth tables as in Figs. 7 and 8, or by means of Boolean equations. Boolean equations that define the characteristic functions of the output terminals of a CSA element are called its *characteristic equations*. By combining the characteristic equations of the individual components, characteristic equations describing the logical behavior of any CSA network can be constructed.

If a and b can assume any values from V_6 , the characteristic equations have the following slightly more complex forms:

$$z^1 = a^1 b^1 + a^1 b^0 + a^0 b^1$$

$$z^{\tilde{0}} = a^0 b^0$$

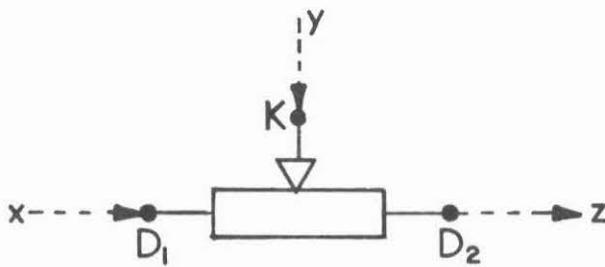
$$z^U = a^{\tilde{0}} + a^{\tilde{1}} + a^U + a^Z + b^{\tilde{0}} + b^{\tilde{1}} + b^U + b^Z$$

$$z^0 = z^{\tilde{1}} = z^Z = 0.$$

4. APPLICATIONS

A major objective of introducing CSA networks is to obtain logically consistent replacements of the mixed-type diagrams widely used in VLSI design specification. Figure 11a shows a representative mixed circuit N_1 (from plate 4 in [1]) which contains three distinct and basically incompatible types of component symbols: logic symbols (the inverter I), standard electronic symbols (the pullup transistor T_0), and "stick" IC layout symbols (various connectors and the transistors T_1 , T_2 and T_3). Figure 11b shows the CSA equivalent circuit N_2 whose behavior can be rigorously defined in terms of V_6 without appealing to electronic or IC layout concepts. Note that there is a one-to-one correspondence between the components and connectors of N_1 and those of N_2 . Thus the CSA network N_2 can be used in the same manner as N_1 to provide an approximate indication of the geometry of an IC implementing these circuits. Hence CSA circuits appear to be useful in the preliminary description of IC layouts. If desired, the connector (color-) coding scheme of Fig. 11a can also be used in CSA networks.

We next demonstrate the utility of CSA theory in a small but important aspect of VLSI design, the synthesis of inverters from MOS transistors. Classical logic design reveals none of the internal structure of inverters, which significantly affects their layout cost. Inverters are thoroughly analyzed from a VLSI viewpoint in [1,5] using electronic circuit models. We give here a parallel analysis for certain types of inverters using CSA logical models.



$$\begin{aligned}
 z^0 &= x^0 y^T \\
 z^1 &= x^1 y^T \\
 \tilde{z}^0 &= x^{\tilde{0}} y^T \\
 \tilde{z}^1 &= x^{\tilde{1}} y^T \\
 z^Z &= y^F + x^Z y^T \\
 z^U &= y^Z + y^U + x^U y^T
 \end{aligned}$$

Fig. 10 Definition of switch behavior by means of characteristic Boolean equations.

Figure 10 shows a set of characteristic equations for the switch of Fig. 7c. It is convenient in such equations to let T (true) denote either 1 or $\tilde{1}$, and F (false) denote either 0 or $\tilde{0}$. Similar equations defining connectors, attenuators, amplifiers, etc. can readily be obtained. Surprisingly, the characteristic equations for a connector (which must be equivalent to Def. 2 above) are relatively complex. For instance, the conditions under which the connector output value is $\tilde{1}$ can be specified as follows.

$$z^{\tilde{1}} = (v^{\tilde{1}}_1 + v^{\tilde{1}}_2 + \dots + v^{\tilde{1}}_k)(v^{\tilde{1}}_1 + v^Z_1)(v^{\tilde{1}}_2 + v^Z_2) \dots (v^{\tilde{1}}_k + v^Z_k) .$$

Despite this apparent complexity, the characteristic equations for well-behaved logic networks can often be reduced to forms that closely resemble standard Boolean equations. Consider, for instance, the OR network of Fig. 9a. Assume that the values of the inputs a and b are confined to the subset $\{0,1\}$ of V_6 , which will be the case in any well-behaved CSA network employing this particular OR gate. Then the output function z always assumes values from $\{\tilde{0},1\}$ according to the following set of characteristic equations:

$$\begin{aligned}
 z^1 &= a^1 + b^1 \\
 \tilde{z}^0 &= a^0 \cdot b^0 \\
 z^0 &= z^{\tilde{1}} = z^U = z^Z = 0 .
 \end{aligned}$$

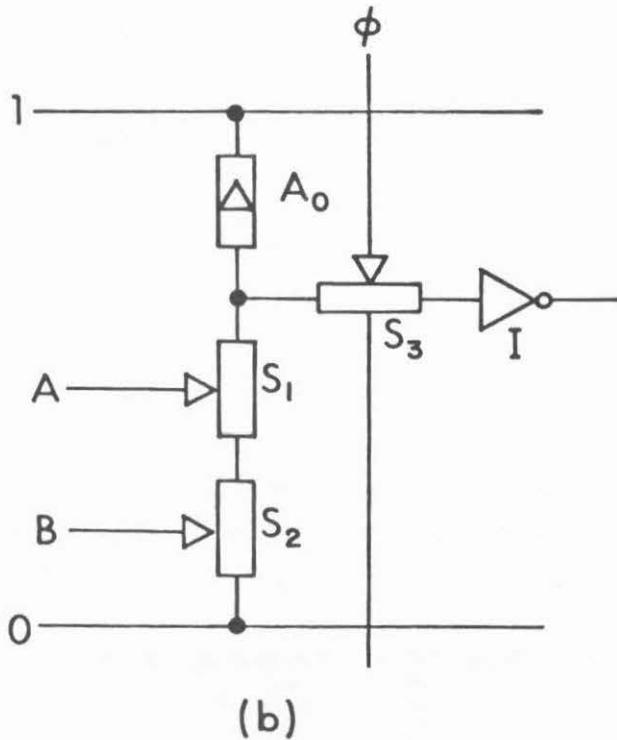
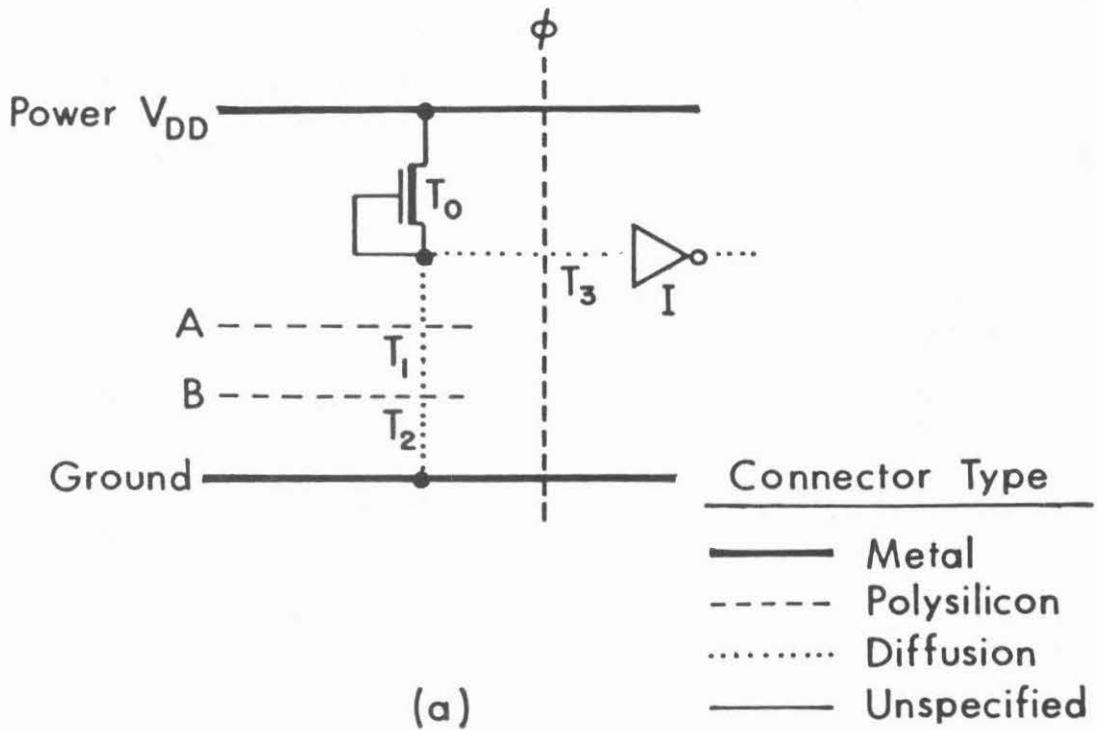


Fig. 11 (a) A typical mixed circuit. (b) The equivalent CSA network.

The building blocks of most VLSI circuits are MOS transistors that are configured either as transistor switches or a pullup/pulldown loads; of Fig. 11. We therefore need only two CSA component types, switches and attenuators. The switches may be amplifying or nonamplifying, inverting or non-inverting, depending on the particular MOS technology being used. It is easily proven that at least two CSA components are needed to build an inverter, and that the simplest inverter must contain either two switches, or a switch and an attenuator. Figure 12a shows an inverter composed of two switches of opposite K polarity. The fact that it inverts $\{0,1\}$ can be quickly proven as follows. Suppose that the primary input x is 0. This causes switch S_1 to close making $a=1$, while simultaneously S_2 opens making $b=Z$. The primary output z therefore assumes the value $v(T_2) = \#(a,b) = \#(1,Z) = 1 = \bar{x}$. Conversely if $x=1$, then $a=Z$, $b=0$, and $z = \#(Z,0) = 0$. The behavior of this network can also be defined by the following characteristic equations:

$$z^0 = x^1$$

$$z^1 = x^0$$

$$z^{\tilde{0}} = z^{\tilde{1}} = z^U = z^Z = 0.$$

Hence for any $x \in \{0,1\}$, $z = \bar{x}$, so the network represents a Boolean NOT gate or inverter. If the switches are replaced by amplifying versions (see Fig. 7), then the network of Fig. 12a also performs the inversions $\tilde{0} \rightarrow 1$ and $\tilde{1} \rightarrow 0$. It models very closely both the structure and logical behavior of the CMOS inverter [5] appearing in Fig. 12b.

Figure 12c shows another CSA inverter composed of two elements, this time a switch and an attenuator. If the input signal x is $\tilde{0}$ or 0, then S_1 opens making $a=Z$. The attenuator's output signal b is $\tilde{1}$, therefore the primary output value z is determined by $\#(a,b) = \#(Z,\tilde{1}) = \tilde{1}$. Similarly if $x = \tilde{1}$ or 1, then $a=0$, and $z = \#(0,\tilde{1}) = 0$. Notice that in one case z assumes the weak 1-like value $\tilde{1}$. This asymmetric lack of drive is indeed a basic characteristic of the NMOS ratio-type inverter of Fig. 12d [1] which is represented at the CSA level by the network of Fig. 12c. Thus the CSA model describes the

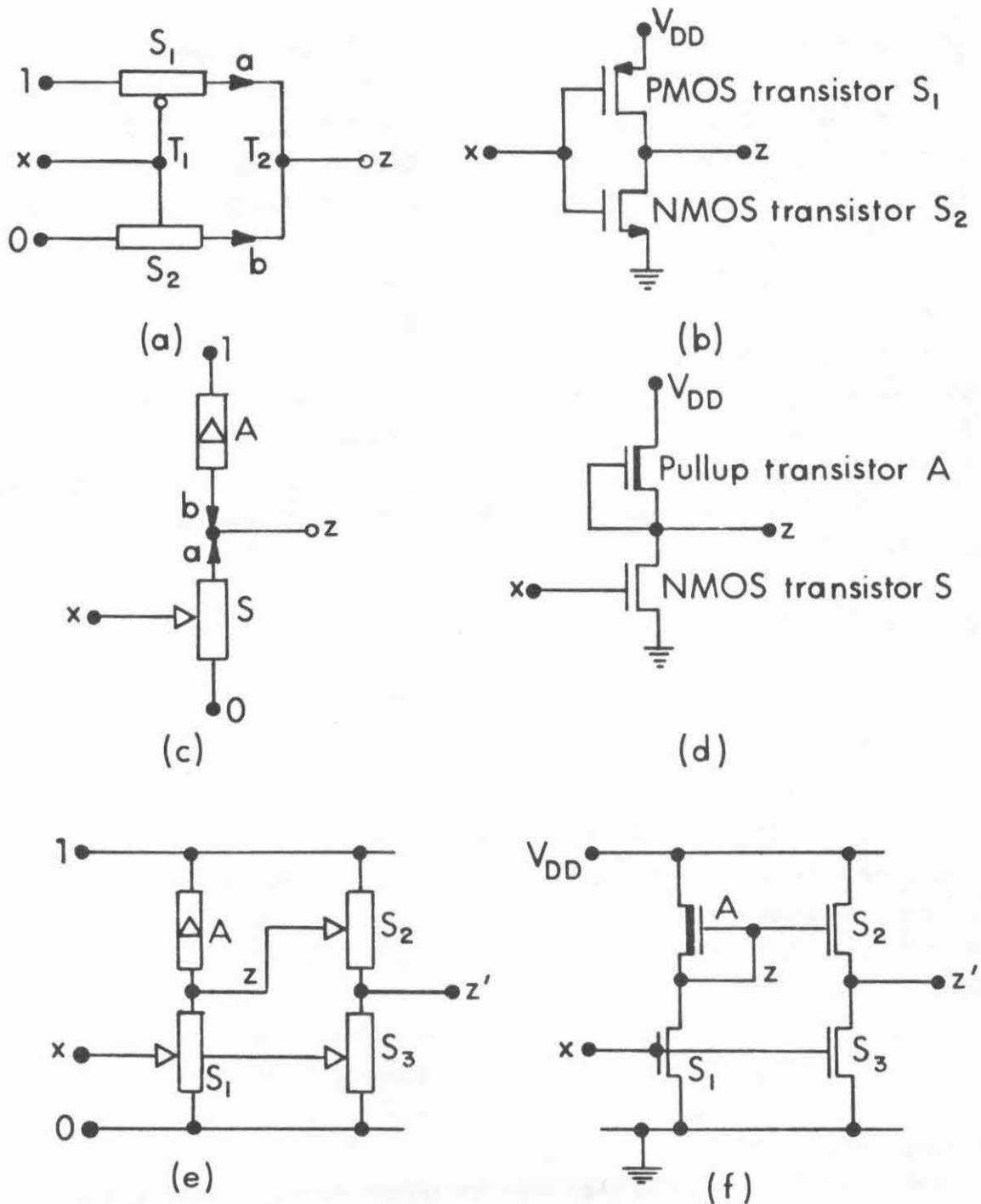


Fig. 12. Various CSA inverter designs and the MOS circuits they model.

structure, logical behavior, and drive capability of this NMOS inverter.

We can attempt to amplify the weak $\tilde{1}$ output signal of the CSA network in Fig. 12c by adding more components. Figure 12e shows the most straightforward way of doing this. A second switch S_2 is inserted in the line z to convert the offending $\tilde{1}$ to 1; it does so simply by connecting the "power line" to the new primary output z' . To ensure that z' is also driven to 0 when x is $\tilde{1}$ or 1, we need the third switch S_3 . S_3 is controlled by x and connects the "ground line" to z' . The result is an inverter that always generates the strong output values 0 and 1. As Fig. 12f indicates, we have reinvented the inverting super buffer [1], a standard circuit in VLSI design. It is used to replace the ratio-type inverter of Fig. 12d in situations where the asymmetric output values 0 and $\tilde{1}$ of the latter are unacceptable.

5. DISCUSSION

The logical design theory presented here can be regarded as a refinement of standard two-valued switching theory which reveals some hidden variables and components that are of central importance in VLSI design. CSA components have the same number of terminals as the MOS devices they model, so that a CSA network displays all the connectors of a circuit. Explicit representation of these connectors is essential in IC chip layout. Our approach also emphasizes the important role played by connectors in logical operations. CSA networks use attenuators to model pullup/pulldown devices which are hidden in classical logical design. These devices are at least as important as switches for example, they often occupy more chip area than simple switches. The new variables $\tilde{0}$ and $\tilde{1}$ allow a simplified concept of drive capability to be applied to logic circuits, while the variable Z represents a rigorous logical definition of the well-known high-impedance state.

CSA networks can also be viewed as simplified electronic circuits in which only a few limiting values of the analogue signals being processed are retained. These are the logic values of interest in digital design, and are represented by V_G . A CSA switch thus models only the digital behavior of a transistor switch, while an attenuator is a kind of digital resistor. The voltage-current signal values of a resistor R can vary over a continuous range; those of an attenuator A are confined to V_G . Unlike R , A has only two

modes of operation. In the first mode A is "non-conducting" where the output values of its terminals are both F ($\tilde{0}$ or 0), or both T ($\tilde{1}$ or 1). Thus there is no "logical potential" across A. In the other "conducting" mode of operation, one terminal of A has the value T, while the other has the value F.

CSA theory provides a uniform and rigorous methodology for VLSI design. It employs logical component types that accurately model the digital behavior and external interconnection requirements of MOS transistors. (Bipolar transistors can also be modeled.) It thereby eliminates much of the need for mixed logic/electronic models, and reduces VLSI design at this level to a logical rather than an electronics design process.

Much work remains to be done to determine the extent to which the CSA model can be used by VLSI designers. Of particular interest is the development of CSA-based design algorithms. These algorithms can be expected to be quite rigorous, and should be easy to incorporate into computer-aided design systems. CSA theory also appears to be promising for the analysis of the failure modes and testing requirements of VLSI systems. Unlike standard logic diagrams, a CSA network displays essentially all the connectors of a circuit, hence the standard connector stuck-at-0/1 fault model [4] can be used more accurately. Furthermore this fault model can be augmented by stuck-at- $\tilde{0}/\tilde{1}/Z$ fault types. Stuck-at-Z faults are known to occur frequently in practice [3]. Stuck-at- $\tilde{0}/\tilde{1}$ faults can model certain types of load-dependent failures.

REFERENCES

- [1] C. Mead and L. Conway: *Introduction to VLSI Systems*, Addison-Wesley, Reading, Massachusetts, 1980.
- [2] M. A. Harrison: *Introduction to Switching and Automata Theory*, McGraw-Hill, New York, 1965.
- [3] R. L. Wadsack: "Fault modeling and logic simulation of CMOS and MOS integrated circuits," *Bell System Technical Journal*, vol. 57, pp. 1449-1474, May-June 1978.
- [4] M. A. Breuer and A. D. Friedman: *Diagnosis and Reliable Design of Digital Systems*, Computer Science Press, Woodland Hills, California, 1976.
- [5] W. N. Carr and J. P. Mize: *MOS/LSI Design and Application*, McGraw-Hill, New York, 1972.