A proof rule for restoring logic circuits

J.A. Bergstra and J.W. Klop

Department of Computer Science, Mathematical Centre, 1098 SJ Amsterdam, The Netherlands

Received 21 May 1983

Abstract. An axiomatic semantics is given for restoring logic circuits, both statically and dynamically. As an example the Muller C-element is discussed in detail. It is shown that a consistent circuit reacts in an unambiguous way on new inputs.

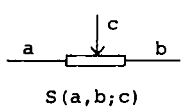
Keywords. Switching theory, Muller C-element, restoring logic, axiomatic semantics.

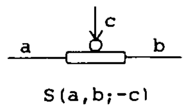
J.A. Bergstra received his Ph.D. in mathematics in 1976 at the University of Utrecht, The Netherlands. From 1976 to 1982 he was with the University of Leiden; since 1983 he is employed at the Mathematical Centre in Amsterdam. His main topics of interest are recursion theory, lambda calculus, abstract datatypes, program verification, concurrent process theory.

J.W. Klop obtained his Ph.D. in 1980 at the University of Utrecht, The Netherlands, on a thesis about lambda calculus and term rewriting systems. Since 1980 he is employed at the Department of Computer Science of the Mathematical Centre in Amsterdam. His main topics of interest are lambda calculus, term rewriting systems, program verification, concurrency, process algebra.

Introduction

A perfect switch (in the notation of Rem [6] and Rem and Mead [7]) is a logic component connecting three wires:





The switch $S(a, b; c)$ connects a and b if the voltage of c is high (1) and disconnects a and b if c's voltage is low (0); the switch $S(a, b; -c)$ connects a and b if c has voltage 0 and disconnects them if c has voltage 1.

A circuit is a configuration of wires and switches, with several wires connected to constants 0 or 1. An interesting example is the Muller C-circuit (see Fig. 1) (see [6] for a discussion). This circuit has a memory capacity of one bit.

In stable states the wires of a circuit will have voltages 0 and 1. Some wires then, preferably including the ones that are used as outputs, will be restoring in the sense that the circuit keeps their voltages firmly at 0 or 1.

To decide which wires are indeed restoring is very much a matter of the physical implementation of the circuit. Further, the voltages of certain wires may be changed thus invoking a process of change throughout the circuit. Describing this is principally a matter of physics as well. Yet, in the words of Mead and Conway [4, p. 68]: "It is important to simplify our mental model of integrated circuitry, so as to more quickly and easily analyze or explain the function of a given circuit, and more easily visualize and invent new circuit structures without drifting too far away from physically realizable and workable solutions."

The present aim is to find an *axiomatic semantics* of circuits by providing proof rules about restoring logic and circuit dynamics. Viewed mathematically, the rules are plausible, and a semantical theory about much more complex circuits can

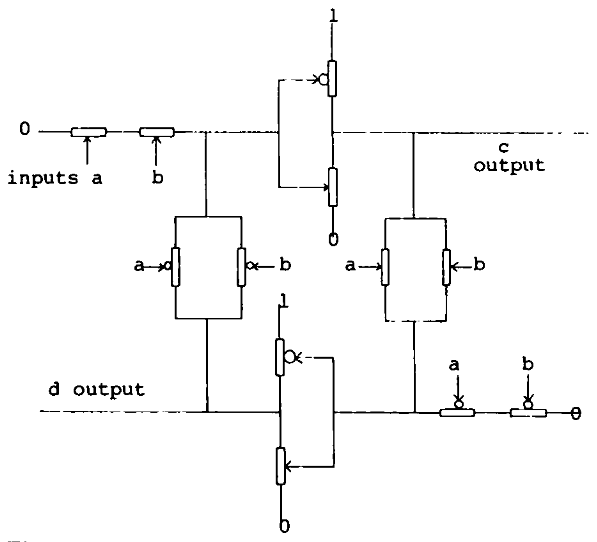


Fig. 1.

probably be based on them (that theory must describe clocks, timing and communication).

In [1, p. 46] an exercise (namely (2.1)) about analyzing a circuit occurs. The analysis made there [1, p. 52] is inconsistent with ours; the proof rules we offer reject that circuit as being inconsistent.

Of course, a typical question might now be posed. For which operational *semantics* are these axioms and rules sound and complete? We have not the slightest idea on how to find a plausible semantics underlying these proof rules, apart from systems of partial differential equations that defeat any analysis.

It must be mentioned that classical switching theory (see, e.g., [5] or [3, Chapter 11]) contains much information about similar types of circuits (viz. asynchronous sequential circuits). Our axiomatic semantics, however, might be new.

This work has been inspired by reading Rem's [6] paper. There Rem explores formal semantics for circuits, useful as a basis for a theory of silicon compilation.

The structure of the remainder of this paper is as follows: Section 1 gives preliminary definitions, in Section 2 a proof system called *Restoring Circuit Logic* is presented, Section 3 describes circuits subject to changing inputs, in Section 4 we apply the reduction system of Section 3 to derive the behavior of the Muller C-circuit, and Appendix A deals with input and output wires of a circuit.

1. Preliminary definitions

In this section we will give the basic definitions of *circuits, subcircuits* and *labeled* circuits. Our definition of 'circuit' is (adapted) from [6].

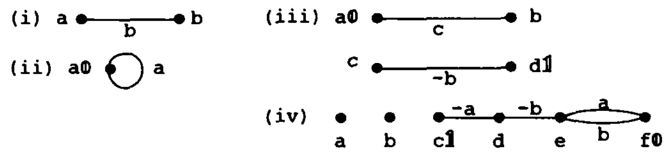
1.1. Definition. A *circuit* C is

(i) a graph (consisting of a set of *nodes* {a, b, c,...} and some *arcs* between the nodes) such that

(ii) each arc is labeled by 'a' or '−a' for some node named a,

(iii) together with a specification of two disjoint subjects *C*𝟘, *C*𝟙 of the set of nodes.

1.2. Example



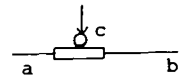
In Example 1.2(ii), (iii), (iv) the nodes in *C*𝟘 are designated by writing 𝟘 at that place, likewise for *C*𝟙. Note that the circuit graphs may be disconnected and may have multiple arcs between a pair of nodes.

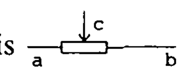
1.3. Notation. (1) Henceforth the nodes of circuit C will be called *wires*; they form the set *W*(*C*). The arcs will be called *switches*; *S*(*C*) is the set of switches. A switch between $a, b \in W(C)$, labeled by $c \in W(C)$, is named $S(a, b;\ c)$. Likewise $S(a, b;\ -c)$ denotes the switch between a, b labeled by $-c$.

(It will be clear from the sequel that circuits containing more than one switch $S(a, b;\ c)$ for fixed a, b, c are redundant. That is, we may assume that for each $a, b, c \in W(C)$ there is at most one $S(a, b;\ c)$. Likewise for $S(a, b;\ -c)$.)

Note that $S(a, b;\ c)$ and $S(b, a;\ c)$ denote the same switch.

(2) To conform more to our intuitions, we will (in diagrams) not represent circuits as graphs but as networks where the wires are indeed wires and where, as in [6], switches are denoted as follows:

$S(a, b;\ c)$ is 

and $S(a, b;\ -c)$ is 

1.4. Example. The circuits of Example 1.2 are represented by the diagrams of Fig. 2.

1.5. Definition. C' is called a *subcircuit* of C, notation: $C' \subseteq C$, iff

(i) $W(C') \subseteq W(C)$ and $S(C') \subseteq S(C)$,

(ii) $S(a, b;\ (-)c) \in S(C') \Rightarrow a, b, c \in W(C')$ (every switch in the subcircuit C' is supported by wires in C'),

(iii) $C'\mathbb{0} \subseteq C\mathbb{0} \cap W(C')$ and $C'\mathbb{1} \subseteq C\mathbb{1} \cap W(C')$.

1.6. Example. In the NOR-circuit of Fig. 2(iv) the heavily drawn part (see Fig. 3), consisting of $\{a, c, e, f\}$, $\{S(e, f;\ a)\}$ is a subcircuit. (Note that it contains two separate parts.) Furthermore, $C'\mathbb{0} = \emptyset$ and $C'\mathbb{1} = \{c\}$, if C' is the subcircuit.

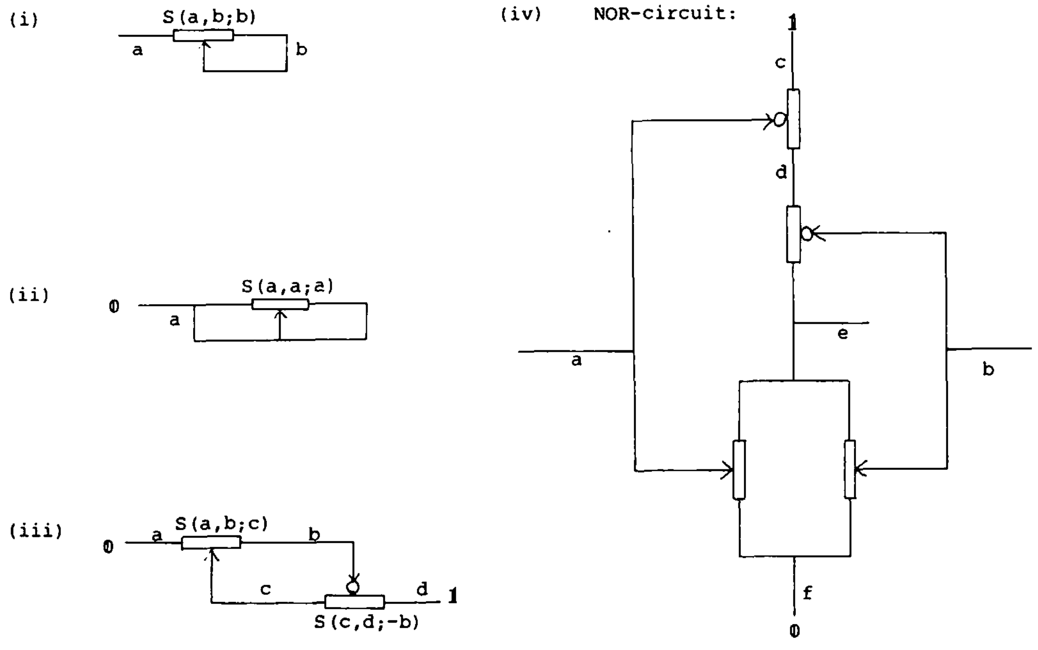


Fig. 2.

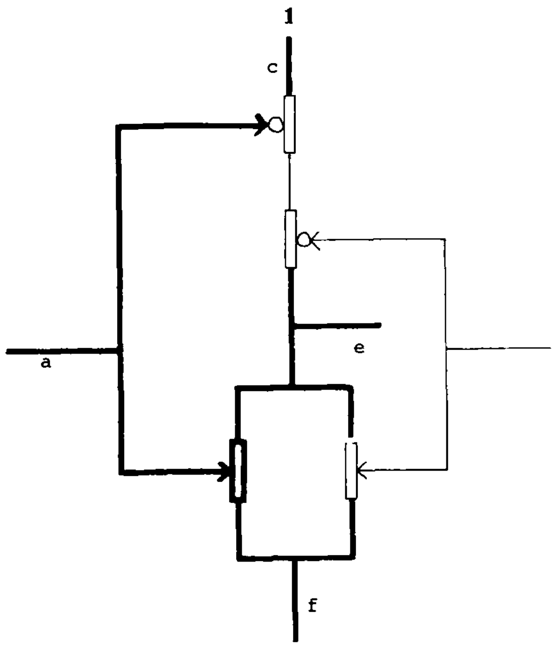


Fig. 3.

1.7. Definition. *Labeled circuits.*

(i) The set of *labels* is $\{\mathbb{0}, \mathbb{1}, 0, 1\}$. Here $\mathbb{0}$ and $\mathbb{1}$ are called 'restoring 0' and 'restoring 1'.

(ii) A *labeling* of a circuit C is a map $L: W(C) \to \{\mathbb{0}, \mathbb{1}, 0, 1\}$. A *labeled circuit* C is a pair (C, L) where L is a labeling of C.

(iii) Let (C, L) be a labeled circuit. The *weakening* of L, written L^-, is the labeling defined by

$$L^-(a) = \begin{cases} 0 & \text{if } L(a) = \mathbb{0} \text{ or } L(a) = 0, \\ 1 & \text{if } L(a) = \mathbb{1} \text{ or } L(a) = 1. \end{cases}$$

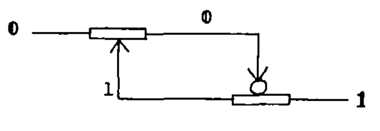
(iv) (C', L') is a *labeled subcircuit* of the labeled circuit (C, L) iff $C' \subseteq C$ and $L' = L \upharpoonright S(C')$ (L restricted to $S(C')$).

(v) A labeling L of the circuit C is *correct* iff

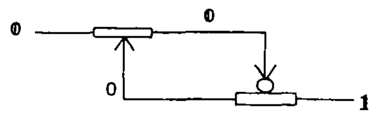
$$(1) \quad L^-(a) = \begin{cases} 0 & \text{for all } a \in C\mathbb{0}, \\ 1 & \text{for all } a \in C\mathbb{1}; \end{cases}$$

$$(2) \quad \begin{array}{l} \text{for every } S(a, b; c) \in S(C): \ L^-(c) = 1 \Rightarrow L^-(a) = L^-(b), \\ \text{for every } S(a, b; -c) \in S(C): \ L^-(c) = 0 \Rightarrow L^-(a) = L^-(b). \end{array}$$

1.8. Example



(a)



(b)

Example 1.8(a) exhibits a correct labeling, while Example 1.8(b) does not.

2. Restoring logic circuits

We shall now present a proof system called RCL (for *Restoring Circuit Logic*) which is designed to prove statements of the form

$$L \vdash_{\mathrm{RCL}} a\mathbf{0}$$

(likewise for $a\mathbf{1}$). Often the subscript RCL will be omitted. Here it is important that L is required to be a *correct* labeling of the circuit C under consideration.

The proof system RCL will be used to derive, from a given correct labeling L of C, a *stable* labeling (as given in Definition 3.3).

The system RCL is built as a 'natural deduction' proof system and has the following axioms and rules:

(1) *Axioms*

$$a\mathbf{0} \quad \text{if } a \in C\mathbf{0}, \qquad a\mathbf{1} \quad \text{if } a \in C\mathbf{1}$$

(2) *Switch rules*

$$\frac{a\mathbf{1} \quad c\mathbf{1}}{b\mathbf{1}} S(a, b; c) \qquad \frac{a\mathbf{0} \quad c\mathbf{1}}{b\mathbf{0}} S(a, b; c)$$

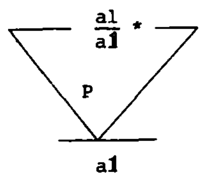
$$\frac{a\mathbf{1} \quad c\mathbf{0}}{b\mathbf{1}} S(a, b; -c) \qquad \frac{a\mathbf{0} \quad c\mathbf{0}}{b\mathbf{0}} S(a, b; -c)$$

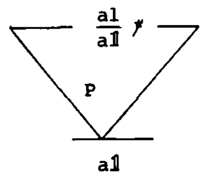
(3) *Assumption rules*

$$\frac{a1}{a\mathbf{1}} * \qquad \frac{a0}{a\mathbf{0}} *$$

This rule enables us to assume that $a1$ will be strengthened to $a\mathbf{1}$. This assumption is marked by '$*$', and in the next rule we have the means of discharging the assumption.

(4) *Restoring logic rule*

if 

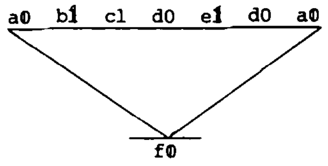
is a proof, then 

is a proof

(and likewise with 1, $\mathbb{1}$ replaced by 0, $\mathbb{0}$).

Here P must be nonempty; that is, the two displayed occurrences of '$a\mathbb{1}$' may not coincide. Further, it is allowed to discharge (i.e., crossing out '$*$') all occurrences of $\frac{a1}{a\mathbb{1}}*$ simultaneously.

A proof P is *assumption rule free* if it contains no undischarged '$*$'. If P has the form, say,



where $a\mathbb{0}$, $b\mathbb{1}$ are axioms and P is assumption rule free, we may write

$$c1,\ d0,\ e\mathbb{1} \vdash f\mathbb{0}.$$

Finally, if L is a *correct* labeling such that $L(c) = 1$, $L(d) = 0$, $L(e) = \mathbb{1}$, we may write

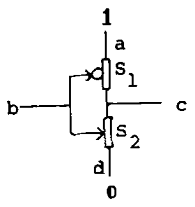
$$L \vdash f\mathbb{0}.$$

To ease some formulations, we will also write $L \vdash pi$ if $L(p) = i$, $i \in \{\mathbb{0}, \mathbb{1}, 0, 1\}$.

2.1. Remark. Note that it is not possible to give 'redundant' proofs; e.g. $\frac{a\mathbb{1}}{a\mathbb{1}}$ is not allowed.

2.2. Examples

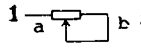
(i) 'NOT' is the circuit



Now

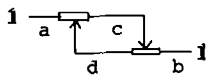
$$\frac{a\mathbb{1} \qquad b\mathbb{0}}{c\mathbb{1}}\,s_1 \quad \text{hence} \quad b\mathbb{0} \vdash c\mathbb{1}.$$

Likewise $b\mathbb{1} \vdash c\mathbb{0}$.

(ii) Let C be  and let $L(b) = 1$; so L is correct. Now

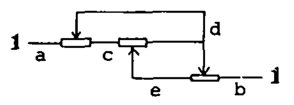
$$\frac{\dfrac{b1}{b\mathbb{1}}* \qquad a\mathbb{1}}{b\mathbb{1}} \quad \text{hence} \quad \frac{\dfrac{b1}{b\mathbb{1}}\cancel{*} \qquad a\mathbb{1}}{b\mathbb{1}}.$$

Therefore $b1 \vdash b\mathbb{1}$.

(iii) Let C be  and let $L(c) = L(d) = 1$. Then

$$\frac{\dfrac{a\mathbb{1} \qquad \dfrac{d1}{d\mathbb{1}}\cancel{*}}{c\mathbb{1}} \qquad b\mathbb{1}}{d\mathbb{1}} \quad \text{hence} \quad d1 \vdash d\mathbb{1}.$$

(iv) Let C be



Then $c1, e1 \vdash e\mathbb{1}, c\mathbb{1}, d\mathbb{1}$. E.g.,

$$\cfrac{\cfrac{\cfrac{\cfrac{\frac{c1}{c\mathbb{1}}\not\uparrow \quad \frac{e1}{e\mathbb{1}}\not\uparrow}{d\mathbb{1}} \qquad a\mathbb{1}}{c\mathbb{1}} \qquad \frac{e1}{e\mathbb{1}}\not\uparrow}{d\mathbb{1}} \qquad b\mathbb{1}}{e\mathbb{1}}$$

2.3. Remark. Note that the proof system RCL only *restores* values 0,1. It is not possible to *change* values 0 to $\mathbb{1}$ or 1 to $\mathbb{0}$. (In the next section, however, this will be possible.)

2.4. Definition. Let (C, L) be a correctly labeled circuit. Then the RCL-*closure* of L, denoted $\bar{L}$, is defined by

$$L \underset{\text{RCL}}{\vdash} ai \Leftrightarrow \bar{L}(a) = i \quad \text{for all } a \in W(C) \text{ and } i \in \{\mathbb{0}, \mathbb{1}, 0, 1\}.$$

Since L is correct, $\bar{L}$ is uniquely determined. (For, in $(C, \bar{L})$ some of the $a0$, $b1$ in (C, L) are 'strengthened' to $a\mathbb{0}$, $b\mathbb{1}$; values do not change sign.)

2.5. Definition. (C, L) is an *everywhere restoring labeled circuit* if L is correct and $L: W(C) \to \{\mathbb{0}, \mathbb{1}\}$.

Note that if (C, L) is everywhere restoring, then $L = \bar{L}$. (The reverse is not true: consider, e.g., the circuit C of Example 2.2(ii) with $L(b) = 0$ and $L(a) = \mathbb{1}$.)

3. Circuits subject to changing inputs

We will now describe the *dynamic behaviour* of a circuit C, i.e., what happens after a change of the values at some 'input ports'. This means that $C\mathbb{0}$, $C\mathbb{1}$ are modified. Since the circuit may have memory capacity (internal states), as is the case, e.g., in the Muller C-circuit (see Section 4), the 'old' labeling has to be taken into account during such a modification. However, combining the modification of the inputs with the old labeling will, in general, result in an *incorrect* labeling. To describe these transformations of incorrectly labeled circuits we use a *reduction system* which closely resembles and is based on the proof system RCL in Section 2. The objective is that this reduction system enables us to 'reduce' the circuit, starting from a possibly incorrect labeling and, via possibly incorrect intermediate labelings, to a 'stable' final labeling, which is then the result of the input modification:

$$(C, L) \to (C, L') \to (C, L'') \to \cdots \to (C, L_{\text{final}}).$$

We will assume that after an input modification the old labeling L lingers on in weakened form, that is, as L^-. First we define the reduction system.

3.1. Definition. Consider the class of labeled circuits (C, L) where C is fixed and L is arbitrary (and possibly incorrect). On this class of labelings of C a relation '$\to$', called *reduction*, will be defined as follows.

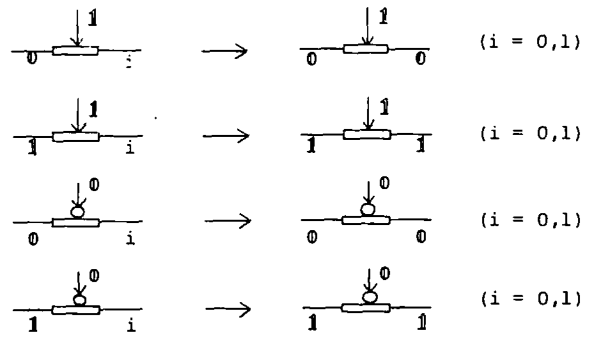
First we defined in (1), (2), (3) below the reduction relation on labeled subcircuits (C', L') of (C, L). Such (C', L') as in the left-hand sides of (1), (2), (3) will be called *redexes*, and may be conceived as 'elementary' parts which are candidates for reduction.

(1) *Input reduction rules*

$$\mathbb{0}\underline{\;\;i\;\;} \to \mathbb{0}\underline{\;\;\mathbb{0}\;\;} \quad (i = 0{,}1),$$

$$\mathbb{1}\underline{\;\;i\;\;} \to \mathbb{1}\underline{\;\;\mathbb{1}\;\;} \quad (i = 0{,}1).$$

(2) *Switch reduction rules*



(3) *Restoring logic reduction rule.* Let (C', L') be a *correctly* labeled subcircuit. (So $\overline{L'}$ is given as in Definition 2.3.) Suppose $L' \neq \overline{L'}$. Then

$$(C', L') \to (C', \overline{L'}).$$

(4) *Subcircuit rule.* If $(C', L') \subseteq (C, L)$ and $(C', L') \to (C', L'')$, then

$$(C, L) \to (C, L[L''/L']).$$

Here $L[L''/L']$ is L where L' is replaced by L'', i.e.,

$$L[L''/L'](a) = \begin{cases} L''(a) & \text{if } a \in W(C'), \\ L(a) & \text{otherwise.} \end{cases}$$

3.2. Notation. (1) The transitive reflexive closure of '$\to$' will be denoted by '$\twoheadrightarrow$'.

(2) If $(C, L) \to (C, L')$, we will say: (C, L) reduces *in one step* to (C, L'). For brevity we will sometimes write $L \to L'$.

3.3. Definition. (i) A labeled circuit (C, L) is *in normal form* iff no reduction rule applies to it. (C, L) *has* a normal form iff $(C, L) \twoheadrightarrow (C, L')$ for some L' such that (C, L') is in normal form.

(ii) (C, L) is *unambiguous* if it has precisely one normal form.

(iii) (C, L) is *stable* if it is correct and in normal form.

(iv) (C, L) is *inconsistent* if it has an incorrect normal form.

We will now consider whether it is possible to reduce a labeled circuit to a normal form, and whether such a normal form is unique. We start with a simple observation.

3.4. Proposition. *Every labeled circuit* (C, L) *has a normal form. Moreover, every reduction of* (C, L) *must end in a normal form.*

Proof. In every reduction step the number of occurrences of $\mathbb{0}$,$\mathbb{1}$ increases. (In the restoring logic reduction rule, this is so because we required there $L' \neq \overline{L'}$.) Furthermore, occurrences of $\mathbb{0}$ and $\mathbb{1}$ are permanent. Since $W(C)$ is finite, the proposition follows. □

3.5. Lemma. *Let* (C, L_0) *be a* ***consistent*** *labeled circuit (not necessarily correct), and suppose that* $(C, L_0) \to (C, L_1)$ *and* $(C, L_0) \to (C, L_2)$.

Then there is a labeling L_3 *such that* $(C, L_i) \twoheadrightarrow (C, L_3)$ *for* $i = 1, 2$.

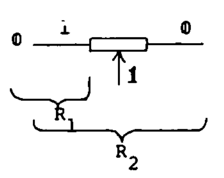
$$\begin{tikzcd} & (C, L_0) \arrow[dl] \arrow[dr] & \\ (C, L_1) \arrow[dr, two heads] & & (C, L_2) \arrow[dl, two heads] \\ & (C, L_3) & \end{tikzcd}$$

Proof. We will distinguish redexes of type (1), (2) or (3), according to the rules in Definition 3.1.

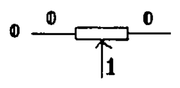
If the two redexes $(R_i, L_i) \subseteq (C, L_0)$, which are reduced in the steps $(C, L_0) \to (C, L_i)$ $(i = 1,2)$, are *disjoint* (in an obvious sense), then the statement of the lemma is evidently true. Likewise if (R_1, L_1) and (R_2, L_2) are identical (as subcircuit occurrences). Otherwise, the following cases arise.

Case 1. The two redexes are both (1)-redexes. This can only be the case if they are disjoint or identical.

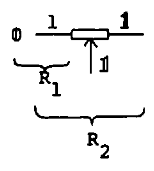
Case 2. (R_1, L_1) is a (1)-redex and (R_2, L_2) is a (2)-redex. E.g.,



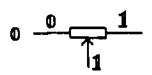
Then the common reduct is

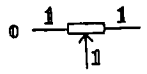


A case as, e.g.,



cannot arise, since (C, L_0) is consistent. For, this subcircuit reduces either to



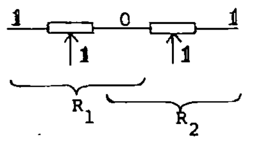
or to


and during every further reduction this configuration stays the same. In particular, any normal form of (C, L_0) is incorrect, whence (C, L_0) is inconsistent.

The other cases of the type (1)-redex vs. (2)-redex are analogous.

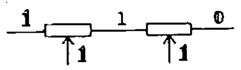
Case 3. (R_1, L_1) and (R_2, L_2) are both (2)-redexes.

A typical example is given below:



which leads to the common reduct consisting of the same circuit with 0 replaced by $\mathbb{1}$.

Cases like



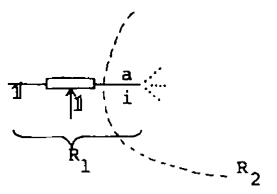
cannot occur since such a configuration reduces to a permanent (i.e., in every further reduction) incorrect labeled circuit (viz. the same subcircuit where 1 is replaced by either $\mathbb{0}$ or $\mathbb{1}$).

The other cases of this type, (2) vs. (2), are similar.

Case 4. (2) vs. (3).

(a) If the (2)-redex (R_1, L_1) is part of the (3)-redex (R_2, L_2), there is no problem, due to the correctness requirement in a (3)-redex.

(b) Otherwise, we have, e.g., the situation given by the following diagram:



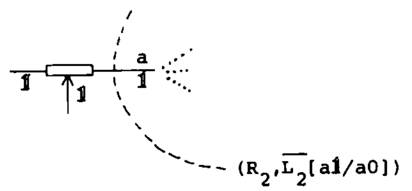
Several subcases arise according to the values of a and $\overline{L_2}(a)$. We will treat some typical subcases. The most interesting case of this proof is subcase (iii).

(i) $i = 0$ and $\overline{L_2}(a) = \mathbb{0}$: the circuit is inconsistent.

(ii) $i = 0$ and $\overline{L_2}(a) = \mathbb{1}$ or 1: this cannot happen, since in a correct labeling (in casu L_2) values can only be restored.

(iii) $i = 0$ and $\overline{L_2}(a) = 0$.

Claim. *The common reduct is*



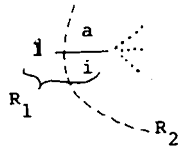
Proof of the claim. Let (R_2^*, L_2^*) be the subcircuit of (R_2, L_2) obtained by removing a and the switches connected to a; L_2^* is L_2 restricted to R_2^*. Clearly, (R_2^*, L_2^*) is also a (3)-redex, that is, L_2^* is correct. Reducing this (3)-redex yields $(R_2^*, \overline{L_2^*})$. Adding a, the switches connected to a and the labeling $a0$ again, results in $(R_2, \overline{L_2})$, because in the proof $L_2 \vdash \overline{L_2}$ the value $a0$ cannot be used, and also switches $S(b, c; (-)a)$ cannot appear in the proof $L_2 \vdash \overline{L_2}$.

To reach the common reduct we only have to replace in $(R_2, \overline{L_2})$ the value $a0$ by $a\mathbb{1}$.

The remaining cases for this type (2) vs. (3), are similar.

Case 5. (1) vs. (3).

If the (1)-redex (R_1, L_1) is a part of the (3)-redex (R_2, L_2), there is no problem, since L_2 is correct. Otherwise we have a situation like the one given in the diagram below:



and this is analogous to subcase (iii) above.

Case 6. (3) vs. (3).

This (last) case is easy: let (R_1, L_1) and (R_2, L_2) be both (3)-redexes. Let $(R_1 \cup R_2, L_1 \cup L_2)$ denote the result of combining the two subcircuits into one subcircuit. Clearly, $L_1 \cup L_2$ is again correct. Then $(R_1 \cup R_2, \overline{L_1 \cup L_2})$ is a common reduct of $(R_1 \cup R_2, \overline{L_1} \cup L_2)$ and $(R_1 \cup R_2, L_1 \cup \overline{L_2})$. □

3.6. Remark. Proposition 3.4 can be rephrased, in a well-known terminology, as stating that the reduction '$\rightarrow$' has the Strong Normalization (SN) (or Strong Termination) property. Lemma 3.5 says that '$\rightarrow$' is weakly confluent, or has the weak Church–Rosser property (WCR).

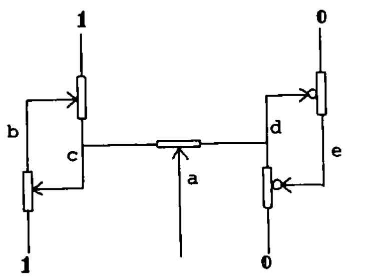
Combining these two properties we have the following theorem.

3.7. Unique Evaluation Theorem. *Let (C, L) be a consistent labeled circuit. Then (C, L) is unambiguous. Moreover, every reduction of (C, L) terminates eventually in the unique normal form.*

Proof. A direct consequence of SN and WCR for '$\rightarrow$', via 'Newman's Lemma' (see, e.g., [2]). □

3.8. Example. This example shows that an inconsistent (C, L) may have several correct normal forms. Let C be the circuit given by the diagram at the top of page 173. Now let (C, L) be

$(a, b, c, d, e) = (\mathbb{0}, \mathbb{1}, \mathbb{1}, \mathbb{0}, \mathbb{0})$, a correct initial labeling.



Then (C, L^-) is

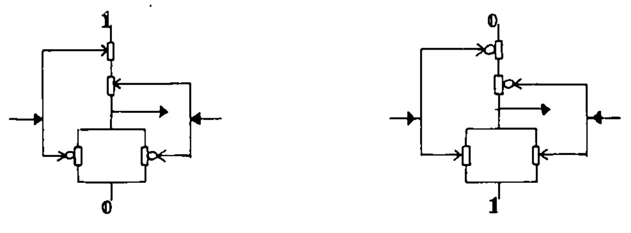
$$(a, b, c, d, e) = (0, 1, 1, 0, 0), \quad \text{the weakened labeling.}$$

Let (C', L^-) be

$$(a, b, c, d, e) = (\mathbb{1}, 1, 1, 0, 0), \quad \text{change of input } a.$$

(C', L^-) reduces to $(a, b, c, d, e) = (\mathbb{1}, \mathbb{1}, \mathbb{1}, \mathbb{1}, 0)$, a correct normal form. (Apply the restoring logic reduction rule on the left 'cycle', followed by an application of a switch reduction rule.) Likewise, (C', L^-) reduces to $(a, b, c, d, e) = (\mathbb{1}, 1, \mathbb{0}, \mathbb{0}, \mathbb{0})$ by applying the restoring logic rule first on the right 'cycle'. Finally, by a simultaneous application of this rule on both cycles, we obtain the incorrect normal form $(a, b, c, d, e) = (\mathbb{1}, \mathbb{1}, \mathbb{1}, \mathbb{0}, \mathbb{0})$.

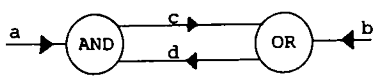
3.9. Example. Let circuits 'AND' and 'OR' be defined as follows:



AND OR

(the ' $\rightarrow$ ' arrows denote input and output ports).

Let C be the circuit given by the following diagram:



Now $(a, b, c, d) = (\mathbb{0}, \mathbb{1}, \mathbb{0}, \mathbb{1})$ is a stable labeling. After weakening to $(0, 1, 0, 1)$ and simultaneously changing the inputs (a, b) to $(\mathbb{1}, \mathbb{0})$, we obtain $(\mathbb{1}, \mathbb{0}, 0, 1)$, an incorrect normal form.

(If the inputs (a, b) are changed sequentially to $(\mathbb{1}, \mathbb{0})$, the result is either $(\mathbb{1}, \mathbb{0}, \mathbb{1}, \mathbb{1})$ in case a is first changed, or $(\mathbb{1}, \mathbb{0}, \mathbb{0}, \mathbb{0})$ in case b is first changed. Both labelings are stable.)

3.10. Example. This example occurs in [1, Exercise 2.1, p. 46, solution on p. 52], where it is asked to analyze the dynamic behavior. However, the circuit with the

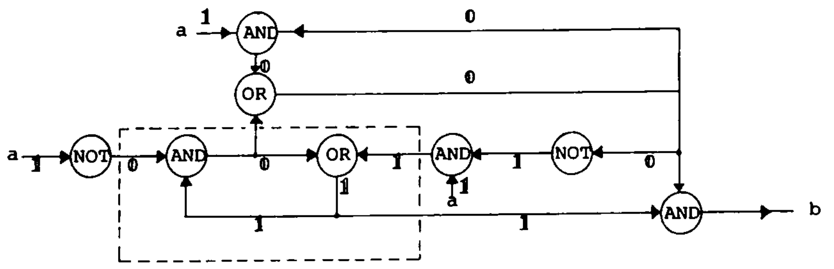


Fig. 4.

correct labeling as displayed in Fig. 4 is inconsistent after weakening and changing the value of the input a to **0**. We then have the labeling as in the diagram shown in Fig. 5. This labeling is an incorrect normal form. In [1, p. 52] an intuitive procedure is sketched to reach from this labeling a correct normal form: take the value for y_1, y_2 (i.e., 0, 1) and 'compute further' till stabilization occurs. Then (y, y_2, z_1, z_2) will be (**1**, **1**, **0**, **1**). However, if z_1, z_2 is taken as starting point for this intuitive stabilization, then the result is (**0**, **0**, **0**, **0**). Our reduction procedure does not suffer from this ambiguity, since the labeled circuit in Fig. 5 is rejected as being inconsistent.

The essential difficulty of this circuit is already present in the subcircuit indicated by the rectangle above, which was considered in Example 3.9.

We are now in a position to describe the effects of an input modification:

(1) Let circuit C be given with a correct labeling L.

(2) Weaken L to L^-.

(3) Modify C**0**, C**1**. Results: a circuit C' with C'**0**, C'**1**. (*Note*: the underlying circuit, i.e., C without specification of C**0**, C**1** has of course remained the same in C'.)

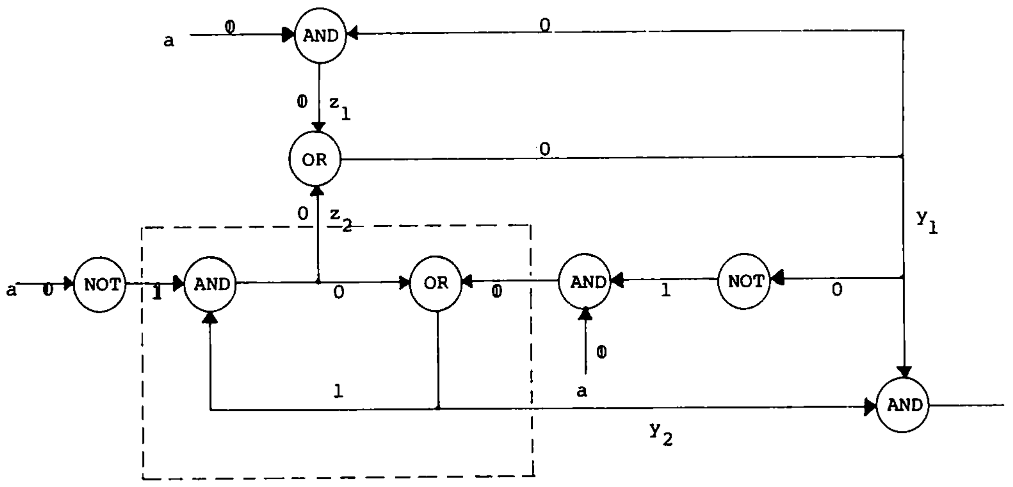


Fig. 5.

(4) (C', L^-) is now an incorrectly labeled circuit, which is not in normal form. If (C', L^-) is inconsistent, i.e., has an incorrect normal form, the circuit is disqualified. Otherwise:

(5) Reduce (C', L^-) to the unique correct normal form (C', L').

In the next section we will apply the reduction system and the procedure defined above to analyze the behaviour of the Muller C-circuit. We will close this section with two definitions, for which we need the concept of an I/O-circuit. Up to here, it was (deliberately) not specified for a circuit which channels where the input and output channels. (In Appendix A we will deal with this question.) Let us assume in advance that an I/O-circuit is defined. Then we have the following definition.

3.11. Definition. (i) Let C be an I/O-circuit. C is called *fully consistent* if for every possible assignment of values 𝟘, 𝟙 to the input channels and every correct labeling L extending this input assignment, the resulting labeled circuit (C, L) is consistent.

(ii) Let (C, L) be a correctly labeled I/O-circuit. Then (C, L) is *fully restoring* if it is consistent and the values of the output channels in the unique normal form (C, L') are 𝟘 or 𝟙.

4. An example in detail: The Muller C-element

We will now apply the reduction system in Section 3 to derive the behavior of the Muller C-circuit, as shown in the Introduction. The circuit has inputs a, b and outputs c, d.

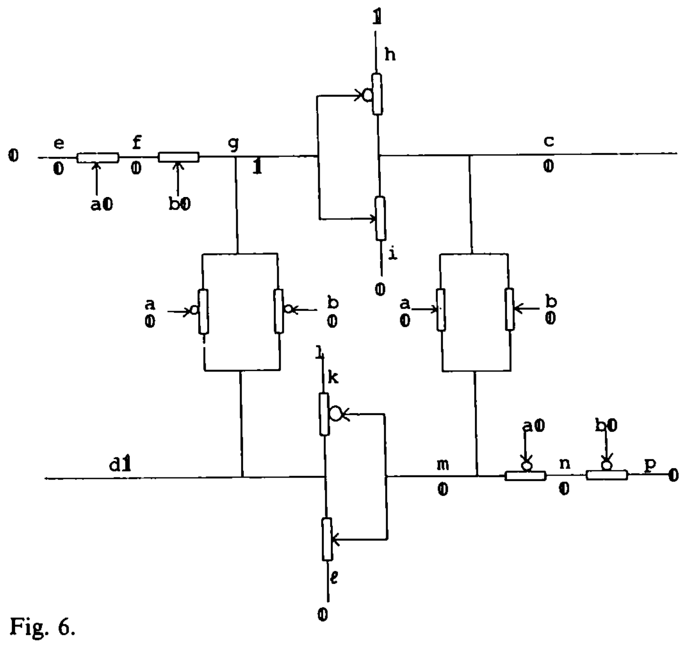


Fig. 6.

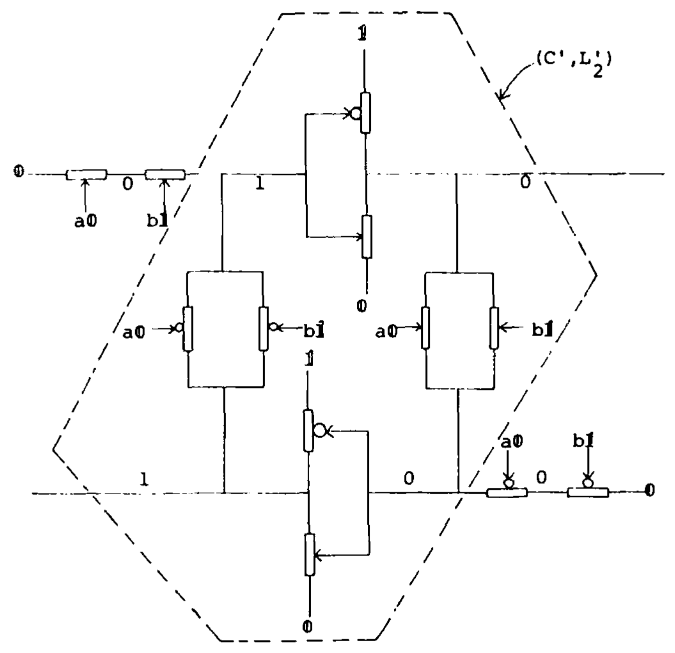


Fig. 7.

(1) Start with $(a, b) = (\mathbb{0}, \mathbb{0})$ and the correct labeling L_1 shown in the diagram given by Fig. 6 (the value of f is arbitrary). Replace (a, b) by $(\mathbb{0}, \mathbb{1})$ after weakening L_1 to L_1^-, as in the diagram given by Fig. 7.

(2) The enclosed subcircuit (C', L_2') in Fig. 7 is correctly labeled. Using the restoring logic rule (4) of Section 2 we prove:

$$\cfrac{\cfrac{\cfrac{\cfrac{\frac{g1}{g\mathbb{1}}\ddagger \qquad i\mathbb{0}}{c\mathbb{0}} \qquad b\mathbb{1}}{m\mathbb{0}} \qquad k\mathbb{1}}{d\mathbb{1}} \qquad a\mathbb{0}}{g\mathbb{1}}$$

Likewise we prove that *all* values in the subcircuit C' are made restoring in L_2'.

Now the restoring logic reduction rule enables us to substitute these restored values instead. This yields $(C', \overline{L_2'})$. From this labeling we prove

$$\frac{g\mathbb{1} \quad b\mathbb{1}}{f\mathbb{1}} \quad \text{and} \quad \frac{m\mathbb{0} \quad a\mathbb{0}}{n\mathbb{0}}$$

thus arriving at a stable position L_2.

(3) Now change (a, b) to, say, $(\mathbb{1}, \mathbb{1})$, after weakening L_2 to L_2^-. Then we arrive at the stable position L_3 in the diagram given by Fig. 8, without using this time the restoring logic (reduction) rule. Note the non-restoring $n0$.

Continuing this analysis we find the (expected) behavior of the Muller C-circuit

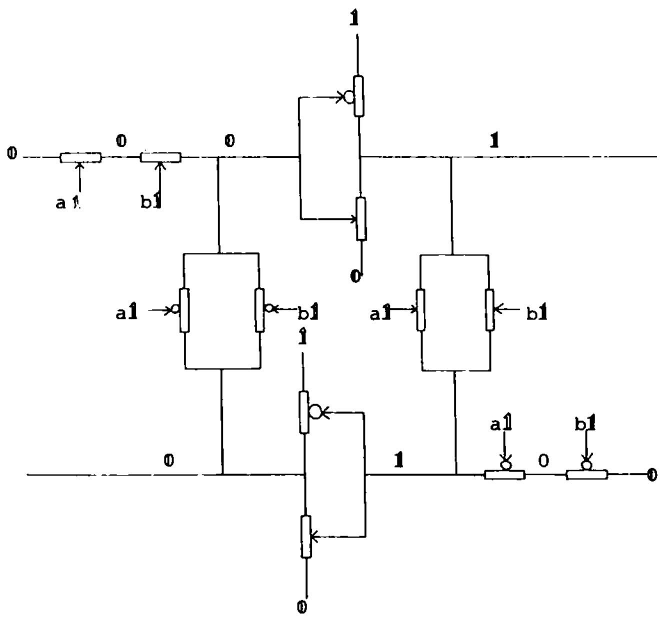


Fig. 8.

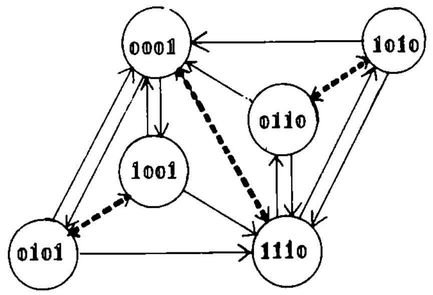
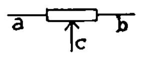


Fig. 9.

as in the transition diagram given by Fig. 9. Here the 4-tuples indicate the value of (a, b, c, d). (The dashed lines denote a *simultaneous* change of the input values a, b.) Note that the circuit is fully restoring, i.e., the output values are always restoring. Also the circuit is fully consistent. The circuit has six stable positions, not taking into account the values of f and n which are irrelevant in some labelings.

Appendix A. Inputs and outputs

For a circuit C we are interested in deciding which wires can serve as inputs and which ones can serve as outputs. Already in the simple case of one switch



it is clear that the situation is not straightforward:

(i) if a is an input, then b is not,

(ii) if b is an input, then a is not,

(iii) c is not an output.

Let $(C, \boldsymbol{a}, \boldsymbol{b})$ be a circuit C together with a specification of disjoint sets $\boldsymbol{a}, \boldsymbol{b} \in W(C)$. Here $\boldsymbol{a} = a_1, \dots, a_n$ and $\boldsymbol{b} = b_1, \dots, b_m$. Furthermore, we employ the following notations.

A.1. Notations. (i) $\mathscr{L}$ is the set of *correct* $\{0, 1\}$-labelings of C.

(ii) $\mathscr{J} = \{\mathbb{0}, \mathbb{1}\}^n$ is the set of n-tuples of restoring values $\mathbb{0}$, $\mathbb{1}$ (which will serve as assignments to $\boldsymbol{a} = a_1, \dots, a_n$).

(iii) For $L \in \mathscr{L}$ and $\alpha = (\alpha(1), \dots, \alpha(n)) \in \mathscr{J}$ we write $L[\alpha]$ to denote L relabeled by α on $\boldsymbol{a}$, that is

$$(L[\alpha])(a_i) = \alpha(i) \quad (i = 1, \dots, n), \qquad (L[\alpha])(c) = L(c) \quad \text{if } c \notin \boldsymbol{a}.$$

(iv) If $L[\alpha]$ is *consistent*, it has a unique normal form which is denoted by $\mathrm{NF}(L, \alpha)$. Further, $F(L, \alpha) = \mathrm{NF}(L, \alpha)^-$. Note that $F: \mathscr{L} \times \mathscr{J} \to \mathscr{L}$ (i.e., $F(L, \alpha)$ is again correct).

Now we can state the definition of input and output ports.

A.2. Definition. Let $(C, \boldsymbol{a}, \boldsymbol{b}), \mathscr{L}, \mathscr{J}$ be as described above. Suppose, for all $L \in \mathscr{L}$, $\alpha \in \mathscr{J}$:

(i) $L[\alpha]$ is consistent,

(ii) $\mathrm{NF}(L, \alpha)(b_j) \in \{\mathbb{0}, \mathbb{1}\}$ $(j = 1, \dots, m)$.

Then $(C, \boldsymbol{a}, \boldsymbol{b})$ is called a restoring I/O-circuit with input ports $\boldsymbol{a}$ and output ports $\boldsymbol{b}$.

References

[1] Hopcroft, J.E. and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[2] Klop, J.W., Combinatory reduction systems, *Mathematical Centre Tracts* **127**, Mathematisch Centrum, Amsterdam, 1980.

[3] Kohavi, Z., *Switching and Finite Automata Theory* (McGraw-Hill, New York, 1970).

[4] Mead, C. and L. Conway, *Introduction to VLSI Systems* (Addison-Wesley, Reading, MA, 1980).

[5] Miller, R.E., *Switching Theory, Vol. 2* (Wiley, New York, 1965).

[6] Rem. M., Partially ordered computations, with applications to VLSI design, in: *Proc. 4th Advanced Course on Foundations of Computer Science, Part 2*, Mathematical Centre Tracts **159** (Mathematisch Centrum, Amsterdam, 1983).

[7] Rem. M. and C. Mead, A notation for designing restoring logic circuitry in CMOS, in: C.L. Seitz, ed., *Proc. 2nd Caltech Conf. on VLSI*, California (Institute of Technology, Pasadena, CA, 1981).