APPLICATION OF MULTIPLE-VALUED SWITCH-LEVEL ALGEBRA TO THE DESIGN AND ANALYSIS OF PASS-TRANSISTOR SWITCH NETWORKS

Mou Hu
Dept. of Computer Engineering
Shanghai Institute of Railway Technology
Shanghai, CHINA 200333

Kenneth C. Smith
Dept. of Electrical Engineering
University of Toronto
Toronto, Ont., CANADA M5S 1A4

ABSTRACT

Pass-transistor switch networks have many advantages over gate-level networks, including high density, high speed and low power dissipation. However, their design and analysis lack rigorous mathematical tools. This paper studies the application of the multiple-valued switch-level algebra introduced in [3] *to the design and analysis of pass-transistor networks. After briefly introducing this multiple-valued switch-level algebra, this paper discusses the design and analysis procedures in detail. As well, some design and analysis examples are given.*

I. INTRODUCTION

Pass-transistor networks use a group of interconnected pass transistors to perform specified switch functions. In comparison with conventional gate-level switch networks, pass-transistor switch networks have many advantages, including high density, high speed and low power dissipation. By way of example, reference [1] describes a digital-magnitude comparator. The NMOS gate-level implementation for this comparator requires 79x130= 10,270 sq μm of silicon area. However, the pass-transistor counterpart requires only 52x43=2,236 sq μm of silicon area. Thus a 78-percent reduction of silicon area is achieved through the use of pass-transistor networks. As for speed, the NMOS gate-level network has a typical delay of 8 ns, while the NMOS pass-transistor network provides a typical delay of only 1 ns.

Although pass-transistor networks have these attractive features, their analysis and design still lack a rigorous and mathematical formulation. One difficulty is that Boolean algebra cannot be used directly to design and analyze pass-transistor networks, simply because the basic element of such networks is the MOS transistor rather than the gate. As a result, many designers of pass-transistor networks still depend on intuition, experiece and perseverance. The only formal design procedures [1,2] are the Karnaugh map method and tabular methods extended from the design procedures for gate-level networks.

This paper proposes a new algebraic method for the design and analysis of pass-transistor networks. It is based on the multiple-valued switch-level algebra first proposed in [3] . It is shown that this algebra is suited for the design and analysis of pass-transistor networks as a consequence of its ability to model MOS circuits at the transistor-switch level.

Section II briefly introduces multiple-valued switch-level algebra. Section III discusses an algebraic method for analysis of pass-transistor networks. Section IV presents an algebraic method for design of pass-transistor networks. Section V gives a discussion and a comparison. Section VI concludes the paper.

II. MULTIPLE-VALUED SWITCH-LEVEL ALGEBRA

In this section, the multiple-valued switch-level algebra first proposed in [3] is briefly introduced from the point of view of its application to the design and analysis of pass-transistor networks.

1. Basic logic-value set V_4

The basic logic-value set of the multiple-valued switch-level algebra is composed of four elements and, thus, is called V_4.

$$V_4 = \{U, L, H, Z\}$$

where U represents an undetermined, unknown or intermediate value; L represents logic 0; H represents logic 1; Z represents the high-impedance state.

These four values are useful signals for the processes of design and analysis of pass-transistor networks. For example, when a pass transistor conducts, it can pass an L or H signal; yet, when it is cut off, its output signal is Z; moreover, if there is a conflict in the circuit design, a U signal will appear. The latter situation can occur when two pass transistors conduct simultaneously, with one passing an L signal while the other passes an H signal. In this way the presence of a U signal can indicate a design error.

2. Connect operation

The Connect operation models the effect of connecting two signals together as shown in Fig.1. When two signals A and B are wired together, a new signal C is produced. The signals A, B, and C take values from V_4. The truth table for # operation on V_4 is shown in Table 1.

3. Switch operation *

The Switch operation provides a model for an NMOS pass transistor. As shown in Fig.2, if D_{in}, G and D_{out} are used to represent the pass (input) signal, control signal and output signal respectively, the operation of an NMOS pass transistor can be represented as

$$D_{out} = D_{in} * G .$$

The truth table of the * operation on V_4 is shown in Table 2.

III. APPLICATION OF MULTIPLE-VALUED SWITCH-LEVEL ALGEBRA TO THE ANALYSIS OF PASS-TRANSISTOR NETWORKS

The object of logic analysis of a pass-transistor network is to derive a Boolean expression (or a truth table) for the given pass-transistor network. A procedure for the logic analysis of a pass-transistor network is as follows:

1. Derive a switch-level algebraic expression according to the logic diagram.

This can be done directly by using the switch-level algebra introduced in Section II. In a switch-level expression, the convention is that the priority of the * operation is higher than that of the # operation. Because at the input and output of the network there may be NAND or NOR gates, or inverters, the switch-level expression may contain AND, OR or NOT operators. The corresponding circuit model is called a mixed-level circuit model [4] .

Now a term "path" will be defined: A path in a pass-transistor network is a chain of pass transistors connected in series from an input signal to an output of the network. One important structural feature of a pass-transistor network is that whenever a control signal appears in a path which terminates in an output of the network, its complement will appear in another path terminated in the same output of the network. This feature can be organized to avoid the high impedance output state during the normal operation of a pass-transistor network.

Thus it is important to maintain this feature when a switch-level expression is derived. The best way is to keep the expression in a nested symmetric form.

Example 1.

Derive a switch-level expression for the circuit shown in Fig.3.

Solution:

According to the circuit structure, a switch-level expression can be derived as follows:

$$F_s = L * \bar{C} \# [H * B \# (\bar{D} * \bar{A} \# D * A) * \bar{B}] * C .$$

2. Derive a Boolean expression from the switch-level expression; then simplify the Boolean expression.

As mentioned earlier, uncomplemented and complemented signals should always appear in pairs in the switch-level expression. Thus expressions will generally be of the form $F_1 * G \# F_2 * \bar{G}$. Fig.4 shows the corresponding logic diagram. In general, F_1 and F_2 may be logic constants H or L, logic variables, or outputs of subnetworks. Such an expression will be called a symmetrical expression. The following rule can be used to derive a Boolean expression from a symmetrical switch-level expression.

Rule 1: If the switch-level expression of a pass-transistor network can be expressed as

$$F_s = F_1 * G \# F_2 * \bar{G} ,$$

where F_1, F_2, and F_s take values from $\{L,H\}$, the corresponding Boolean expression is

$$F_b = F_1 \cdot G + F_2 \cdot \bar{G} ,$$

where F_1, F_2, G and F_b take values from $\{0,1\}$.

The correctness of Rule 1 can be proven as follows. By considering every combination of values of F_1, F_2 and G from $\{L,H\}$, and evaluating the F_s expression according to the rules for the * and # operations (Tables 1 and 2), the truth table for F_s can be obtained as shown in Table 3. Also by evaluating the F_b expression for every combination of values for F_1, F_2 and G from $\{0,1\}$, the truth table for F_b can be obtained as shown in Table 4.

Comparing Table 3 with Table 4, it can be concluded that the logic functions F_s and F_b are equivalent.

Example 2.

Derive a Boolean expression for the circuit of Fig.3.

Solution:

From Example 1, the switch-level expression for the circuit of Fig.3 is as follows:

$$F_s = L * \bar{C} \# [H * B \# (\bar{D} * \bar{A} \# D * A) * \bar{B}] * C.$$

Since this expression takes the nested symmetrical form, Rule 1 can be used repeatedly until a Boolean expression is obtained as follows:

$$F_b = 0 \cdot \bar{C} + [1 \cdot B + (\bar{D} \cdot \bar{A} + D \cdot A) \cdot \bar{B}] \cdot C.$$

Simplifying F_b, we can get

$$\begin{aligned} F_b &= B \cdot C + \bar{D} \cdot \bar{A} \cdot \bar{B} \cdot C + D \cdot A \cdot \bar{B} \cdot C \\ &= B \cdot C + \bar{D} \cdot \bar{A} \cdot C + D \cdot A \cdot C \\ &= C \cdot (B + \bar{A} \cdot \bar{D} + A \cdot D). \end{aligned}$$

IV. APPLICATION OF MULTIPLE-VALUED SWITCH-LEVEL ALGEBRA TO THE DESIGN OF PASS-TRANSISTOR NETWORKS

The logic design of a pass-transistor network is a process whose goal is to obtain a logic diagram of a network from its truth table or Boolean expression. Two methods for the design of pass-transistor networks using switch-level algebra will be proposed. They will be referred to as the basic method and the buffer method.

A. BASIC METHOD

The basic method is composed of the following steps:

1. Derive a switch-level expression from the Boolean expression of the function.

If the given specification is a truth table, a Boolean expression should be derived in advance.

If the Boolean expression takes the symmetrical form, the following rule can be used.

Rule 2: If the Boolean expression of a logic network can be expressed as

$$F_b = F_1 \cdot G + F_2 \cdot \bar{G}$$

where F_1, F_2, G and F_b take values from $\{0,1\}$, the corresponding switch-level expression is

$$F_s = F_1 * G \# F_2 * \bar{G}$$

where F_1, F_2, G and F_s take values from $\{L,H\}$.

The correctness of Rule 2 can be proven exactly the same way as for Rule 1.

However, if the Boolean expression is not in a symmetrical form, some preprocessing of the Boolean expression should be undertaken. The following two rules canbe used:

Rule3: If the Boolean expression takes the form $F_1 \cdot G$, change the expression to the form

$$F_1 \cdot G = F_1 \cdot G + 0 \cdot \bar{G}.$$

This equation is always true, since the second term at the right side is always equal to logic 0.

Rule 4: If the Boolean expression takes the form $G+F_1$, change the expression to the form

$$G + F_1 = 1 \cdot G + F_1 \cdot \bar{G}$$

The correctness of the above equation can be proven as follows:

$$\begin{aligned} G + F_1 &= G + (G + \bar{G}) \cdot F_1 \\ &= G + G \cdot F_1 + \bar{G} \cdot F_1 \\ &= G + \bar{G} \cdot F_1 \\ &= 1 \cdot G + \bar{G} \cdot F_1 . \end{aligned}$$

By repeatedly using Rules 3 and 4, any Boolean expression can be changed to the symmetrical form and, finally, Rule 2 can be used to derive a switch-level expression.

Example 3.

Derive a switch-level expression for the circuit of Fig.3 from the Boolean expression derived in Example 2.

Solution:

$$\begin{aligned} F_b &= C \cdot (B + \bar{A} \cdot \bar{D} + A \cdot D) \\ &= (B + \bar{A} \cdot \bar{D} + A \cdot D) \cdot C + 0 \cdot \bar{C} \\ &= [1 \cdot B + (\bar{A} \cdot \bar{D} + A \cdot D) \cdot \bar{B}] \cdot C + 0 \cdot \bar{C} \end{aligned}$$

The corresponding switch-level expression is

$$F_s = [H * B \# (\bar{A} * \bar{D} \# A * D) * \bar{B}] * C \# L * \bar{C}.$$

2. Derive a logic diagram from the switch-level expression.

Example 4.

Design a full adder implemented as a pass-transistor network.

Solution:

The Boolean expressions for a full adder are

$$SUM_b = \bar{A} \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot \bar{C} + A \cdot \bar{B} \cdot \bar{C} + A \cdot B \cdot C$$

$$CARRY_b = A \cdot B + A \cdot C + B \cdot C .$$

Transform these two expressions to a symmetrical form:

$$SUM_b = (\bar{A} \cdot \bar{B} + A \cdot B) \cdot C + (\bar{A} \cdot B + A \cdot \bar{B}) \cdot \bar{C}$$

$$\begin{aligned} CARRY_b &= (A \cdot B) \cdot \bar{C} + (A + B) \cdot C \\ &= (0 \cdot \bar{B} + A \cdot B) \cdot \bar{C} + (1 \cdot B + A \cdot \bar{B}) \cdot C . \end{aligned}$$

The corresponding switch-level expressions are

$$SUM_s = (\bar{A} * \bar{B} \# A * B) * C \# (\bar{A} * B \# A * \bar{B}) * \bar{C}$$

$$CARRY_s = (A * B \# L * \bar{B}) * \bar{C} \# (H * B \# A * \bar{B}) * C .$$

The circuits for the full adder are shown in Fig.5.

B. BUFFER METHOD

Usually pass-transistor networks will emply intermediate and output buffers. There are two reasons for buffers. The first is that the delay of the pass-transistor network is propotional to the number of transistors in a path. The second is that for an NMOS transistor, a 0 signal is passed very well, but a 1 signal is attenuated. Thus, for a combination of these reasons, long paths in the network should be partitioned into small chains, with an output buffer added at the output of each chain stage.

The total number of transistors can be reduced if one considers the output buffer early in the design. This is the basic idea of the Buffer Method. The Buffer Method is essentially the same as the Basic Method except that some preprocessing of the Boolean expressions is done. This prepro-

cessing is composed of the following steps:

1. Complement the original Boolean expression twice.

2. Use DeMorgan's law to obtain an expression with an output inversion.

After preprocessing, the Basic Method can be used.

Example 5.

Design a full adder implemented as a pass-transistor network with an output buffer.

Solution:

Preprocess the Boolean expression for the full adder.

$$SUM_b = \overline{\bar{A}\cdot\bar{B}\cdot C + \bar{A}\cdot B\cdot\bar{C} + A\cdot\bar{B}\cdot\bar{C} + A\cdot B\cdot C}$$

$$= \overline{(\bar{A}\cdot\bar{B}\cdot C)\cdot(\bar{A}\cdot B\cdot\bar{C})\cdot(A\cdot\bar{B}\cdot\bar{C})\cdot(A\cdot B\cdot C)}$$

$$= \overline{(A+B+\bar{C})\cdot(A+\bar{B}+C)\cdot(\bar{A}+B+C)\cdot(\bar{A}+\bar{B}+\bar{C})}$$

$$= \overline{(A+B\cdot C+\bar{C}\cdot\bar{B})\cdot(\bar{A}+B\cdot\bar{C}+C\cdot\bar{B})}$$

$$= \overline{A\cdot B\cdot\bar{C} + A\cdot\bar{B}\cdot C + \bar{A}\cdot\bar{B}\cdot\bar{C} + \bar{A}\cdot B\cdot C}$$

$$CARRY_b = \overline{\overline{A\cdot B + A\cdot C + B\cdot C}}$$

$$= \overline{(\bar{A}+\bar{B})\cdot(\bar{A}+\bar{C})\cdot(\bar{B}+\bar{C})}$$

$$= \overline{(\bar{A}+\bar{B}\cdot\bar{C})\cdot(\bar{B}+\bar{C})}$$

$$= \overline{\bar{A}\cdot\bar{B} + \bar{A}\cdot\bar{C} + \bar{B}\cdot\bar{C}}$$

Obtain the switch-level expression:

$$SUM_s = \overline{(A * \bar{B} \# \bar{A} * B) * C \# (\bar{A} * \bar{B} \# A * B) * \bar{C}}$$

$$CARRY_s = \overline{(\bar{A} * \bar{B} \# L * B) * C \# (H * \bar{B} \# \bar{A} * B) * \bar{C}}.$$

Finally, the circuits can be drawn as shown in Fig.6.

V. DISCUSSION AND COMPARISON

In this section, the design and analysis of CMOS pass-transistor networks using switch-level algebra will be discussed. Comparison of design methods based on switch-level algebra with conventional design methods will be made.

1. Analysis of a CMOS pass-transistor network

The procedure for the logic analysis of a CMOS pass-transistor network is essentially the same as that for an NMOS pass-transistor network presented in Section III. Only one additional concept is needed, that is, Negative Switch Operation $\overline{*}$. As defined in [3], the Negative Switch Operation $\overline{*}$ provides a model for a PMOS pass transistor. If D_{in}, G and D_{out} are used to represent the pass (input) signal, control signal and output signal respectively, the operation of a PMOS pass transistor can be represented as

$$D_{out} = D_{in} \overline{*} \bar{G}.$$

Now that both NMOS transistor and PMOS transistor have their own models, a swith-level algebraic expression can be derived for a CMOS pass-transistor network.

The Negative Switch Operation $\overline{*}$ can be transformed into Switch Operation * according to the following equation:

$$D_{in} \overline{*} G = D_{in} * \bar{G}.$$

Thus, the switch-level algebraic expression of a CMOS pass-transistor network can be transformed into a switch-level algebraic expression of an equivalent NMOS pass-transistor network.

2. Design of a CMOS pass-transistor network

The design methods of NMOS pass-transistor networks presented in Section IV can be extended to design methods for CMOS pass-transistor networks as follows.

(1) Design an NMOS transistor network according to the truth table or Boolean expression given.

(2) For each path in the resultant NMOS network, add a PMOS path in parallel to the NMOS path. This PMOS path should be composed of the same number of transistors as the corresponding NMOS path with each control signal being the complement of corresponding control signal in NMOS path.

(3) If a path in the resultant NMOS network passes a 0, only NMOS path is needed.

(4) If a path in the resultant NMOS network passes a 1, only PMOS path is needed.

3. Comparison of design methods based on switch-level algebra with conventional design methods

The full adder design will be used as an example for the comparison.

The design based on switch-level algebra given in Fig.6 uses 12 transistors and 5 inverters (3 for inputs and 2 for outputs). The conventional design as given in [4] uses 12 transistors, 4 inverters and 1 three-input NOR gate.

Obviously the design methods proposed in this paper gives simpler circuits than conventional design methods.

VI. CONCLUSIONS

The application of switch-level algebra [3] to the design and analysis of pass-transistor networks is studied in this paper. The special features of switch-level algebraic expressions for pass-transistor networks are discussed. Rules for transforming a Boolean expression to a switch-level expression, and a switch-level expression to a Boolean expression, are given. Procedures for the design and analysis of pass-transistor networks are developed.

It can be seen that switch-level algebra as described in [3] is a powerful tool for the design and analysis of pass-transistor networks.

ACKNOWLEDGEMENT

The work reported in this paper is partly supported by the National Natural Science Foundation of China.

REFERENCES

[1] D. Radhakrishman, S.R. Whitaker, and G. Maki, "Formal Design Procedures for Pass Transistor Switching Circuits", IEEE Journal of Solid-State Circuits, Vol.SC-20, No.2, pp.531-536, April 1985.

[2] J.C. Shovic, "Full Custom Pass Transistor Design Methodology", Presented at IEEE 1989 Workshop on VLSI, Feb. 1989.

[3] Mou Hu, "A Multiple-Valued Algebra for Modeling MOS VLSI Circuits at Switch-Level", Proc. of 1989 IEEE International Symposium on Multiple-Valued Logic, pp.329-336, May 1989.

[4] J.P. Hayes, "An Introduction to Switch-Level Modeling", IEEE design and Test of Computers, Vol.4, No.4, pp.18-25, Aug. 1987.

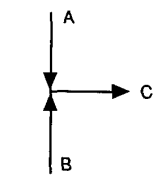

Fig.1 Connect Operation
C=A # B

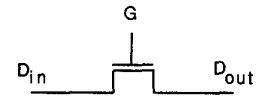

Fig.2 Switch Operation
$D_{out} = D_{in} * G$

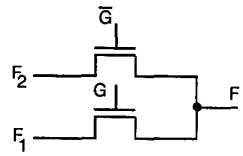

Fig.4 Circuit for
$F = F_1 * G \# F_2 * \bar{G}$

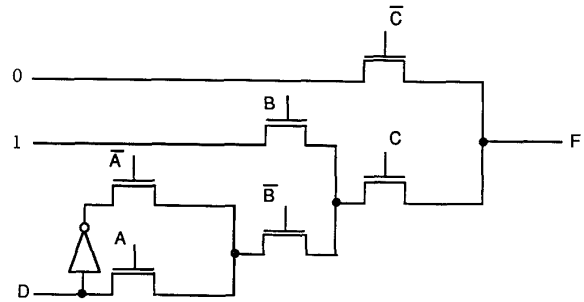

Fig.3 Circuit for Example 1

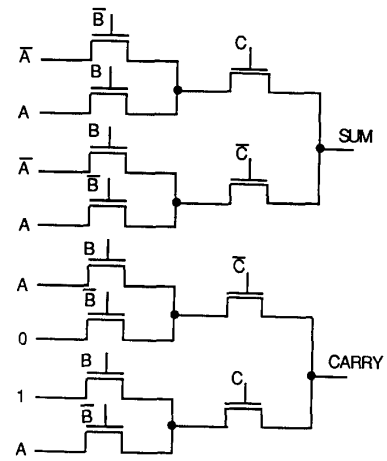

Fig.5 Full Adder

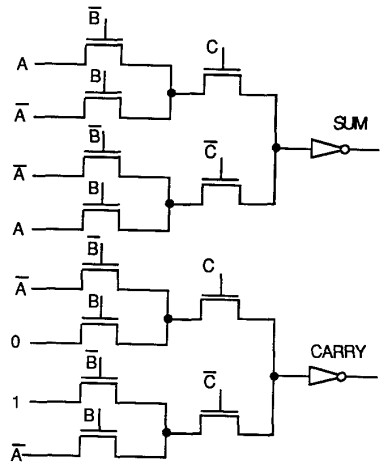

Fig.6 Full Adder with
Output Buffer

Table1. Truth Table of # Operation on V_4
$C = A \# B$

A \ B	U	L	H	Z
U	U	U	U	U
L	U	L	U	L
H	U	U	H	H
Z	U	L	H	Z

Table 2. Truth Table of * Operation on V_4
$D_{out} = D_{in} * G$

D_{in} \ G	U	L	H	Z
U	U	Z	U	U
L	U	Z	L	U
H	U	Z	H	U
Z	U	Z	Z	U

Table 3. Truth Table for

$F_s = F_1 * G \# F_2 * \bar{G}$

F_1	F_2	G	F_s
L	L	L	L
L	L	H	L
L	H	L	H
L	H	H	L
H	L	L	L
H	L	H	H
H	H	L	H
H	H	H	H

Table 4. Truth table for

$F_b = F_1 \cdot G + F_2 \cdot \bar{G}$

F_1	F_2	G	F_b
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1