FORMAL DESIGN PROCEDURES FOR DIFFERENTIAL CASCODE VOLTAGE SWITCH WITH PASS GATE (DCVSPG) LOGIC

Damu Radhakrishnan
Division of Computer Engineering, School of Applied Science
Nanyang Technological University, Nanyang Ave., Singapore 639798
email: asdrkrishnan@ntu.edu.sg, Fax: (65)792-6559

ABSTRACT

In this paper, a formal design approach for the differential cascode voltage switch with pass gate (DCVSPG) logic is presented. This permits the design of any combinational logic function by the use of a k-map or by a modified Quine-McCluskey algorithm. The basic design approach used is based on pass logic and shows that a DCVSPG logic gate designed with minimum number of transistors may not always provide the fastest possible circuits.

1. INTRODUCTION

The demand for high performance digital systems requires continuously faster CMOS circuit speed. Dynamic logic clearly provides a performance advantage over the traditional static logic, and thus is being used more and more frequently [2]. But they suffer from charge sharing, low noise margin, design complexity, and difficulty in testing. Differential cascode voltage switch (DCVS) is claimed to have advantages over the traditional static CMOS design in terms of circuit delay, layout area, logic flexibility, and power dissipation [4,7]. They also offer several testability advantages, such as extended random pattern testability, efficient on-line testing and self checking capabilities [3,8]. They have been used to implement high-performance arithmetic circuits such as fast multipliers as well as pipelined DSP circuits [1]. They are also very suitable for asynchronous designs which have the advantage that the logic works only when needed, ie., the clocks are not running all the time. This reduces power consumption, especially for large and complex circuits. Design procedures for implementing random logic functions in DCVS logic using a minimum number of transistors is given in [5,15]. However, DCVS can cause floating-node in both legs of its nMOS logic tree thereby making it a ratioed logic, which in turn produces current spikes and additional delay [12]. Ratioed logic problem can be solved in dynamic DCVS by using the pre-charging scheme. Unfortunately the floating node problem still exists and may trigger another problem such as the charge redistribution. This might develop into false logic evaluation, thus making the dynamic DCVS very unreliable. Complementary pass transistor logic (CPL) was developed to solve this floating node problem [17]. The loading in CPL was chosen using static inverters instead of the cross-coupled pMOS latch as in conventional DCVS to restore the signal. This results in a mismatch problem between the input signal level and the logic threshold voltage of the static CMOS inverter. It also causes CPL to have poorer noise margin and speed degradation. The double pass transistor logic (DPL) was developed to solve the CPL problem at the expense of double transistor counts and silicon area [10,16]. It also improves circuit performance at reduced supply voltage. However, DPL has not yet been fully adopted because of its high transistor count. The DCVSPG logic family overcomes the drawbacks of DCVS and CPL [9]. The regeneration problem in DCVS caused by the floating-node is solved by a pass gate network. It has symmetrical logic topology, and shorter stack height. This DCVSPG logic is shown to be a ratioless circuit thus making it highly suitable for the design of high performance digital systems. The inputs need not activate transistors, but merely pass signals. The stated performance advantage over normal DCVS is shown to be 16%.

The design procedure given in [9] uses the k-map in a recursive manner. In this paper we show that instead of using this recursive procedure optimal pass logic expressions can be very easily derived by the procedures given in [15], which directly translates to a DCVSPG gate. The rest of the paper is organised as follows. First an overview of pass networks is given in Section 2 to provide enough background for the reader to follow the rest of the material. In Section 3 a brief introduction and the design procedures for DCVS logic is given. In Section 4 the DCVSPG logic is introduced, followed by a formal synthesis procedure for its minimal transistor design. Finally Section 5 concludes the paper.

2. PASS NETWORKS

A pass transistor is an nMOS (pMOS) transistor with the signal input fed to the source and the signal output taken from drain. A pass network is an interconnection of a number of pass transistors to achieve a particular switching function. The propagation of the signal through the transistor is controlled by a signal applied to its gate. In the case of an nMOS transistor, a logic '1' at the gate passes the input from source to drain and a logic '0' opens the source to drain path. A pMOS transistor exhibits similar behavior with a control signal of logic level 0. Only nMOS pass networks are considered further in this paper. If signals X and Y are connected to the gate and source of an nMOS transistor, then it is denoted as X(Y) and read as 'X passing Y'. The logic symbol and the pass logic expression for an nMOS pass gate is shown in Figure 1.

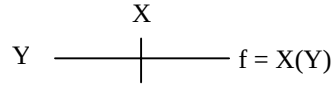

Figure 1. nMOS Pass Transistor Gate

A series connection of a number of nMOS transistors passes the input to the output when all control signals are high. This is represented by the expression $X_1X_2\ldots X_n(V)$, where $X_1, X_2,\ldots,X_n$ are the control variables applied to the gates of the nMOS transistors and V is the pass variable. A product term $P = X_1X_2\ldots X_n$, passing an input signal V, is defined as the pass implicant and is denoted by P(V). A pass function is formed by the logical sum of a number of pass implicants. In a minimal pass function, all pass implicants belong to the set of pass prime implicants. A pass prime implicant is a pass implicant that cannot subsume any other pass implicant with a smaller number of literals in it that implies the same function.

The major difference in the design of pass network, when compared to the gate network, is that both 0's and 1's of the function must be included in the minimization algorithm. Detailed analysis and synthesis procedures for pass networks are presented in [13-15]. The pass network design procedures given in [13] first selects all essential pass prime implicants from the function and then selects a minimal set of pass prime implicants to cover the whole function. The following example illustrates the k-map minimization procedure described in [13,14].

Example 1: A 3 variable k-map is shown in Figure 2(a). The pass implicants are grouped in the map. The implicant corresponding to the first row of cells in the k-map is $\bar{C}(A)$. By forming similar expressions for the other two pass implicants, we get the complete nMOS pass function for the k-map as:

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

The corresponding nMOS pass network implementation is shown in Figure 2(b).

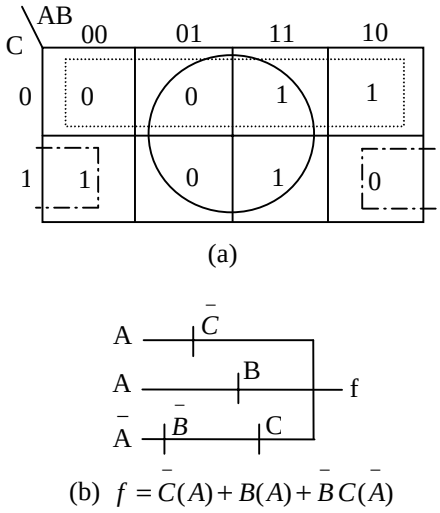

Figure 2. A 3 Variable k-map and its nMOS Pass Network

3. DIFFERENTIAL CASCODE VOLTAGE SWITCH (DCVS) LOGIC

The DCVS logic was first proposed in [7]. It consists of two complementary nMOS networks N_0 and N_1 connected to a pair of cross coupled pMOS pull up transistors as shown in Figure 3. For any input state one of the networks will be ON and the other will be OFF. (N_0 will be ON for f = 0 and OFF for f = 1.) Design procedures for these gates were given in [5]. The nonoptimality of the design procedures given in [5] were pointed out in [15], which presented a formal design approach based on pass network design. In [15] first a minimal Binary Tree Structured (BTS) pass function [11] is derived either from a k-map or by the modified Q-M technique presented in [13,14]. For Exclusive-OR functions, it is seen that factorization of the function by taking the largest complementary sum gives minimum literals in its expression [6]. The following procedure is then used to implement the DCVS gate.

(a) Form the BTS pass network for the function.
(b) Connect the output node to V_{SS}.
(c) For each pass variable connect the corresponding input node of the pass network to both f and f' according to the following criteria:
  (i) If the pass variable is 0, then connect the input node to f only.
  (ii) If the pass variable is 1, then connect the input node to f' only.
  (iii) If the pass variable is X, then connect the input node to f and f' through transistors controlled by X' and X, respectively.
  (iv) If the pass variable is X', then connect the input node to f and f' through transistors controlled by X and X', respectively.

The above procedure does not always guarantee a minimal transistor design. This especially happens when there are more than one solution with differing pass variables, since for every nonconstant pass variable two extra nMOS transistors are needed to connect the pass network to the outputs, whereas if the pass variable is either a 0 or a 1 no extra transistors are needed.

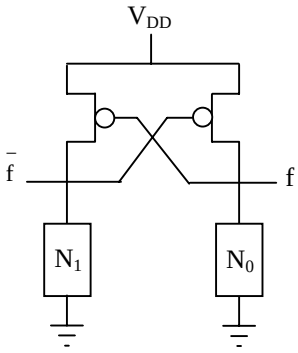

Figure 3. A DCVS gate

4. DIFFERENTIAL CASCODE VOLTAGE SWITCH WITH PASS-GATE (DCVSPG) LOGIC

In a DCVS gate the two nMOS networks behave in a complementary manner (like n and p nets in CMOS gate networks). As mentioned earlier, when one nMOS network is

ON, the other is OFF. Thus only one of the output leads is pulled down by the nMOS network. The other output is pulled up due to the regenerative feedback through the latch connected pMOS pull up transistors. This may even cause a momentary floating output, and if the ratios are not proper, the gate could fail momentarily. This is illustrated by considering a traditional DCVS AND circuit in [9] where it has been shown that during the transition period when both A and B switch from low to high, the output node Q floats. This could also make the logic function fail. In [9] a new DCVS gate called pass gate DCVS (DCVSPG) is introduced which eliminates the floating node problem and also improves the performance of the circuit while decreasing power consumption. The DCVSPG gate has the same logic structure as the DCVS gate shown in Figure 3 except that it uses pass logic structures for N_0 and N_1 instead of restricting the pass variables strictly to 0's. Furthermore the nMOS networks N_0 and N_1 are turned ON and OFF in a complementary manner in a DCVS gate. But in DCVSPG, the two networks are identical and are always ON, and are passing complementary logic values through them, thereby improving the performance. This makes it a ratioless circuit. Once the pass network function for f is obtained, the same network is replicated and the pass variables are complemented to generate $\bar{f}$.

The k-map design procedure for the DCVSPG gate given in [9] first assumes the control and/or pass variables and then uses the k-map in a recursive manner to obtain the final expression. This procedure may not always guarantee a minimal transistor design. On the otherhand, the procedure given in [13,14] uses the k-map once and provides the minimal pass logic expression without making any assumptions on control and/or pass variables. Hence in the following example we show a synthesis procedure for a DCVSPG gate using pass logic design techniques presented in [13,14].

Example 2: Consider the k-map shown in Figure 4(a). To implement the DCVSPG logic first the nMOS pass function for f is derived. One of the minimal expressions is given by:

$f = C(B) + \bar{C}\bar{A}(B) + AB(1) + A\bar{B}\bar{C}(D)$. By factoring,

$f = C(B) + \bar{C}\bar{A}(B) + A(\ B(1) + \bar{B}\bar{C}(D)\ )$

This needs 7 transistors to implement f and is connected to the f output of the cross coupled pMOS transistors as shown in Figure 4(b). $\bar{f}$ uses the same network function, but with all pass variables complemented, and needs another set of 7 transistors. The complete implementation is shown in Figure 4(b). The implementation due to [9] will also result in the same circuit.

By using $B\bar{C}(A)$ instead of AB(1) in f, the number of transistors can be reduced to 6 for f. Hence the optimality cannot always be guaranteed in the above design. Many times the pass logic expressions obtained from the k-map can be simplified further to minimize the number of literals in the expression by using pass logic theorems given in [14]. But this may not always be obvious from the network function. Binary Tree Structured (BTS) pass networks on the otherhand are shown to use the minimum number of transistors to realize a switching function [11]. The BTS pass function for the k-map in Example 2 is given by: $f = C(B) + \bar{C}(\ \bar{A}(B) + A(B(1) + \bar{B}(D))\ )$. This needs only 6 transistors for implementing f and is optimal in its transistor count.

Even though BTS pass networks use the minimum number of transistors for their implementation, the stack height of the tree may be higher for certain input patterns. This may in turn increase the delay during the corresponding transitions. This is illustrated by the following example.

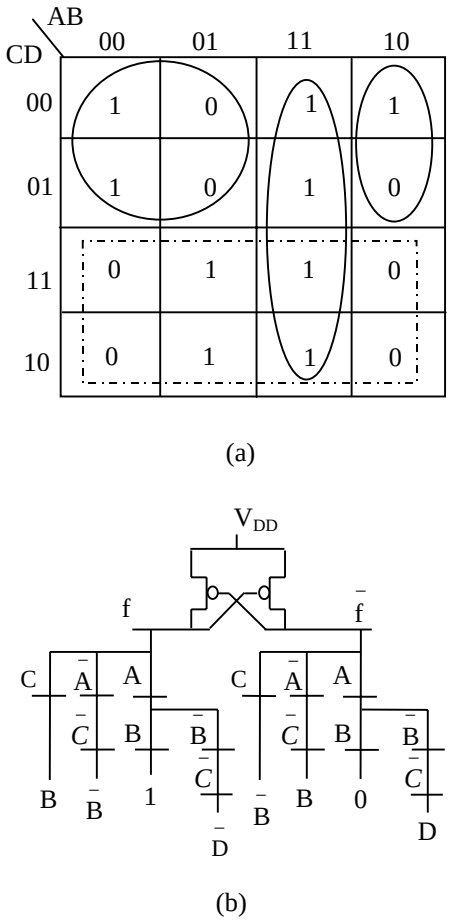

Figure 4. DCVSPG gate for Example 2

Example 3: Consider a four variable switching function given by: f(A,B,C,D) = $\sum$m(0,2,3,5,7,8,9,10,11,13,15)

A minimal pass logic expression for f is given by:

$f = \bar{D}(B) + B(D) + AD(1) + \bar{A}\bar{B}D(C)$

A BTS pass logic expression for the same function is given by:

$f = \bar{D}(B) + D(\ A(1) + \bar{A}\ (\bar{B}(C) + B(1))\ )$

Comparing the two expressions it is seen that the term B(D) in the normal pass function gets modified to $D\bar{A}B(1)$ in the BTS pass function, thus introducing extra delays in generating the output during their corresponding transitions.

When the number of variables is more than 5, k-map minimization becomes difficult. Then the modified Quine-McCluskey algorithm given in [13,14] can be used to find a minimal pass function f. However, for a circuit with more than five variables the stack height increases, thereby degrading the performance of the circuit. This will be a critical factor in the design of these circuits especially when the power supply voltage is scaled down.

5. CONCLUSIONS

A formal design approach for DCVSPG gate is presented in this paper. This design is shown to use minimum number of transistors when the pass networks are designed as BTS pass networks. But this may in turn increase the switching delay in certain transitions compared to other pass network implementations.

6. REFERENCES

[1] Adams R. D., et. al., "A370-MHz memory built-in self-test state machine". *1995 European Design and Test Conference*, 1995, pages 139-141.

[2] Annaratone M., *Digital CMOS Circuit Design*. Kluwer, New York, 1986.

[3] Barzilai Z., et al., "Accurate fault modelling and efficient simulation of differential CVS circuits". *in Proceedings of IEEE International Test Conference*, 1985, pages 722-729.

[4] Chu K.M. and Pulfrey D.I., "A comparison of CMOS circuit techniques: Differential cascode voltage switch logic versus conventional logic". *IEEE Journal of Solid-State Circuits*, vol. SC-22, 1987, pages 528-532.

[5] Chu K.M. and Pulfrey D.I., "Design procedures for differential cascode voltage switch circuits". *IEEE Journal of Solid-State Circuits*, vol. SC-21, 1986, pages 1082-1087.

[6] Feizi A. and Radhakrishnan D. , "Multiple Output Pass Networks: Design and Testing". Proceedings of *IEEE International Test Con*ference, Nov. 1985, pages 907-911.

[7] Heller L.G., Griffin W.R., Davis J.W., and Thoma N.G., "Cascode voltage switch logic: A differential CMOS logic family". in *Digest of Technical Papers, ISSCC*, 1984, pages 16-17.

[8] Kanopoulos N. and Vasanthavada N., "Testing of differential cascode voltage switch (DCVS) circuits". *IEEE Journal of Solid-state Circuits*, vol. 25, June 1990, pages 806-813.

[9] Lai F.S. and Hwang W., " Design and implementation of Differential cascode voltage switch with pass-gate (DCVSPG) logic for high-performance digital systems". *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, April 1997, pages 563-573.

[10] Ohkubo N., et. al., "A 4.4nS CMOS 54 X54-b Multiplier Using pass Transistor Multiplexer". *Proceedings of IEEE 1994 Custom Integrated Circuits Conference*, San Diego, California, May 1994.

[11] Peterson G.E. and Maki G.K., "Binary Tree Structured Logic Circuits: Design and Fault Detection". *Proceedings of International Conference on Computer Design*, NY, USA, Oct. 1984, pages 671-676.

[12] Pfennings L.C., Mol W.J., Bastiaens J.J., and VanDijk J.M., "Differential split-level CMOS logic for subnanosecond speeds". *IEEE Journal of Solid-State Circuits*, vol. SC-20, 1985, pages 1050-1055.

[13] Radhakrishnan D., Whitaker S.R. and Maki G.K., "Formal Design Procedures for Pass Transistor Switching Circuits". *IEEE Journal of Solid-State Circuits*, SC-20, April 1985, pages 531-536.

[14] Radhakrishnan D., "The design and analysis of pass transistor switching circuits". *Ph.D. Dissertation*, University of Idaho, Moscow, Idaho, USA, May 1983.

[15] Radhakrishnan D., "Design of CMOS circuits". *IEE Proceedings - Circuits, Devices and Systems*, vol. 138, no. 1, 1991, pages 83-90.

[16] Suzuki M., Ohkubo N., Shinbo T., Yamanaka T., Shimizu A., Sasaki K., and Nakagome Y., "A 1.5-ns 32-b CMOS ALU in Double Pass-Transistor Logic". *IEEE Journal of Solid-State Circuits*, vol. 28, no. 11, Nov. 1993, pages 1145-1151.

[17] Yano K., et. al., "A 3.8-ns CMOS 16X16 -b multiplier using complementary pass transistor logic". *IEEE Journal of Solid-State Circuits*, vol. 25, April 1990, pp. 388-395.