Supplement to

Logic and Computer Design Fundamentals 4th Edition[1]

More on CMOS Circuit-Level Design

Selected topics not covered in the fourth edition of *Logic and Computer Design Fundamentals* are provided here for optional coverage and for self-study, if desired. This material fits well with the desired coverage in some programs but not may not fit within others due to time constraints or local preferences. This supplement, which is referenced in Chapters 2 and 6 assumes that the reader has studied the subsection CMOS Circuit Technology in Section 6-1, Chapter 6, 4th Edition. This supplement contains sections on Complex Gates, Three-state Buffer Implementation, Transmission Gates, and Transmission Gate-Based Flip-Flops. These sections are largely independent except that the Flip-Flop material depends on the Transmission Gates section. The material in this section is particularly appropriate for coverage by electrical engineering and computer engineering students not expected to study similar material in other courses.

So far we have dealt largely with implementing logic circuits in terms of gates. In this supplement, we continue the exploration of implementing gates and logic using CMOS technology that we began with CMOS Circuit Technology in Chapter 6. CMOS implementation is important because we sometimes design CMOS logic from Boolean equations directly to the transistor level, skipping the logic gate level.

Complex Gates

Note that all of the circuits studied in Section 6-1 operate under DeMorgan's laws, i. e., an output inversion property is characteristic of CMOS gates. In fact, to devise a general design procedure for fully complementary CMOS circuits, we assume that all functions are implemented as $F = \overline{\overline{F}}$. This avoids working directly with p-

channel switches, which involve complementing all literals. Thus, we design the n-channel network for $\overline{F}$ and take the dual to get the p-channel network for F. For functions more complex than NAND, NOR, and NOT, the resulting circuits are called *complex gates* and are designed in accordance with the following procedure for function F:

1. Find and optimize the complement of F, $\overline{F}$.
2. Implement $\overline{F}$ as a switch network using n-channel switch models.
3. Connect the n-channel switch network between ground and output F, and convert the switches to n-channel transistors.
4. Take the dual of the n-channel switch network for $\overline{F}$, and replace the n-channel switch models with p-channel switch models, keeping switch inputs unchanged.
5. Connect the p-channel switch network between +V and the output F, and convert the switches to p-channel transistors.

Step 4 is different than expected: we take the dual of the n-channel switch network to get the p-channel switch network. Recall that the difference between the dual and the complement is that, in taking the complement, all literals in the expression are complemented. We need not do this complementing, however, since the complement of the variables is automatically taken by replacing the n-channel switch models with p-channel switch models. Taking the dual of a function means replacing AND with OR and OR with AND. For a switch network, this corresponds to taking switches or subnetworks that are in parallel and placing them in series and taking switches or subnetworks that are in series and placing them in parallel.

We illustrate this design procedure with an example.

• EXAMPLE 1 Design of a Complex Gate for $F = A\overline{B} + AC + B\overline{C}$

In step 1 of the foregoing procedure, we place F on a map and, from the map, use the 0's to obtain a sum-of-products expression for $\overline{F}$ and the 1's to obtain a product-of-sums expression for $\overline{F}$:

$$\overline{F} = \overline{A}\,\overline{B} + \overline{A}C \qquad \text{SOP}$$

$$\overline{F} = \overline{A}(\overline{B} + C) \qquad \text{POS}$$

Since the product-of-sums expression has fewer literals (three instead of four), we select it for use in the next step.

In step 2, we find an n-channel switch model network for $\overline{F}$. From the product-of-sums expression, $\overline{A}$ is ANDed with the term $\overline{B} + C$. Thus, we place a switch with input $\overline{A}$ in series with a network implementing $\overline{B} + C$, as shown in Figure 1(a). We implement $\overline{B} + C$ with a switch having input $\overline{B}$ in parallel with a switch having input C. Checking step 2, the final network in Figure 1(a) has a path through it for $A = 0$ and either $B = 0$ or $C = 1$ or both.

The network from Figure 1(a) is converted into the corresponding n-channel transistor circuit between ground and the output F, as given in the lower part of Figure 1(c), completing step 3.

Next, in step 4, we are to take the dual of the n-channel switch network from step 2. First, we take the switches B and C that are in parallel in Figure 1(a) and place them in series. Then we take A, which is in series with the parallel combination of B and C and place it in parallel with the series combination of B and C. Finally, we replace the n-channel switch models with p-channel switch models, keeping the complementation on the input values to the switches unchanged. The circuit in Figure 1(b) results.

The network in Figure 1(b) is converted to the p-channel transistor circuit between $+V$ and F, as given in the upper part of Figure 1(c), completing the circuit in step 5.

We can use any Boolean expression for $\overline{F}$, although by minimizing the number of literals as much as possible, as is done in this example, we minimize the number of transistors in the circuit. In the actual design of these transistor circuits, it is also necessary to take electronic considerations into account. For example, in most cases, no path through one of the networks can contain more than four or five transistors in series. This clearly limits the functions that can be implemented in a single complex gate.

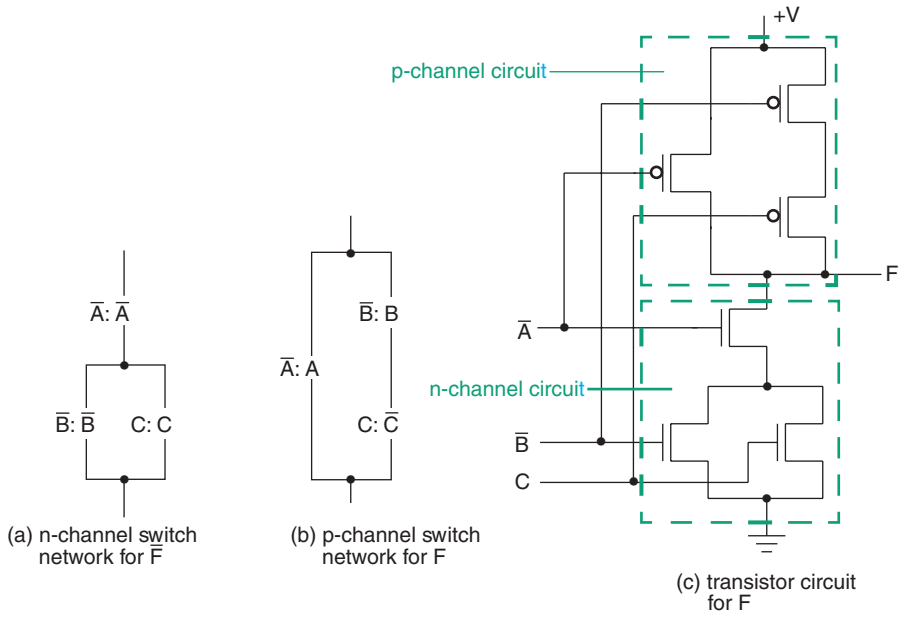


FIGURE 1
Networks and Circuit for Example 1: $F = A + B\overline{C}$

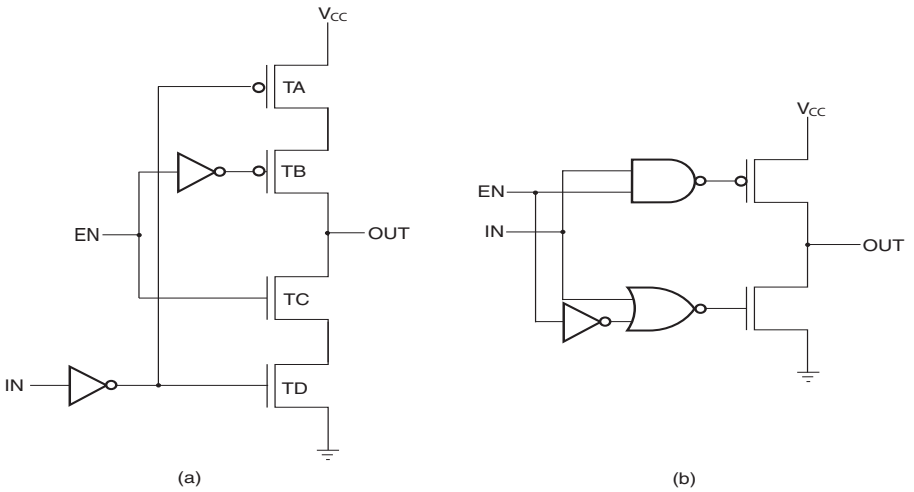


FIGURE 2
Three-State Buffer

Three-State Buffer Implementation

A simple CMOS three-state non-inverting buffer implementation to be used for driving small capacitive loads such as short interconnects is given in Figure 2(a). The input inverter is used to provide the non-inverting property. It drives transistors TA and TD that act as the second inverter. Transistor TB and TC provide the high-impedance state. For $EN = 0$, transistor C is off and $\overline{EN}$ equals 1. For $\overline{EN} = 1$, transistor TB is also off. With TB and TC both off, OUT is no longer connected by any path to either V_{CC} or Ground and, thus, in the high-impedance state. For $EN = 1$, transistor TC is on and $\overline{EN}$ equals 0 turning on transistor TB. For IN equal to 1, transistor TD is on and transistor TA is off. With TC and TD on, Ground is applied to OUT giving OUT = 1. For IN equal to 0, the inverter produces $\overline{IN} = 1$, turning on transistor TD. With TC and TB on, V_{CC} is applied to OUT giving OUT = 1. For IN equal to 0, the inverter produces $IN = 0$, turning on transitory D. With A and B on, V_{CC} is applied to OUT giving OUT = 1. For $EN = 0$, transistor C is on. For IN equal to 0, the inverter produces $\overline{IN} = 1$, turning on transitory D. With C and D on, Ground is applied to OUT giving OUT = 0. Thus the respective values 0 and 1 on IN appear at OUT.

For driving large capacitive loads, the serial pairs of transistors between V_{CC} and OUT and Ground and OUT would need to be very large and due to their size would provide large capacitive loads on their inputs. A different design which uses just one transistor between VCC and OUT and one transistor between Ground and OUT reduces this capacitive loading. Such a design is shown in Figure 2(b) which uses logic to provide the functionality provided by the pair of series transistors. Each gate must drive the input capacitance of the transistor on its output which, in turn, depends on the capacitance driven by OUT and the expected performance. In order to provide small enough input capacitances on IN and EN, it

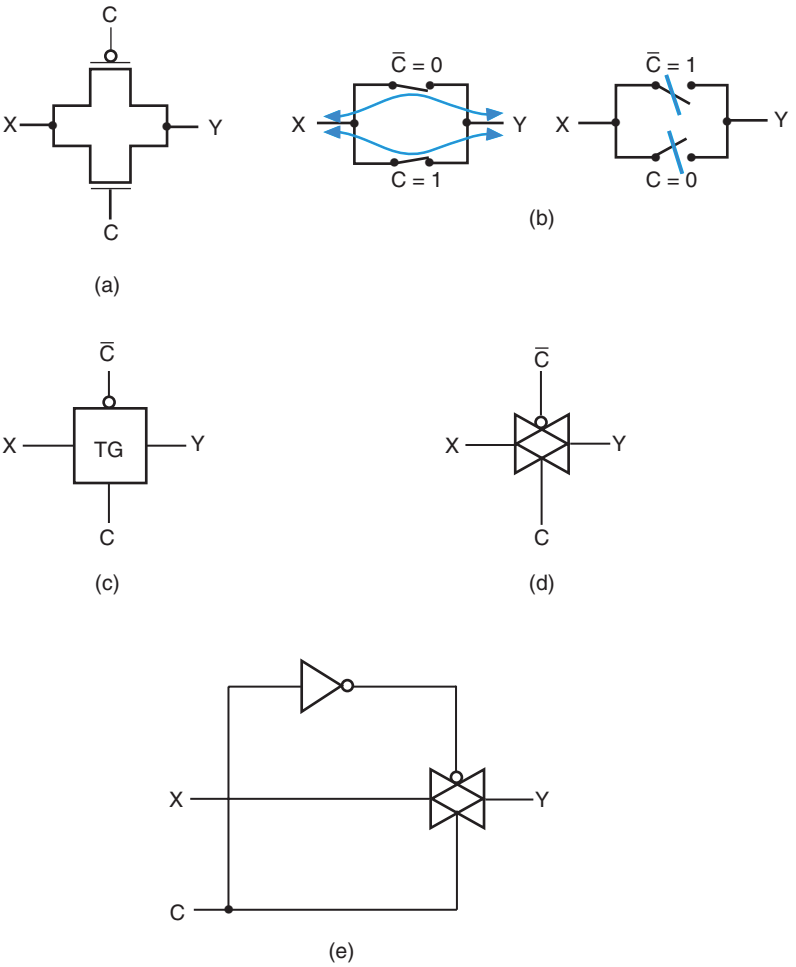


FIGURE 3
Transmission Gate (TG)

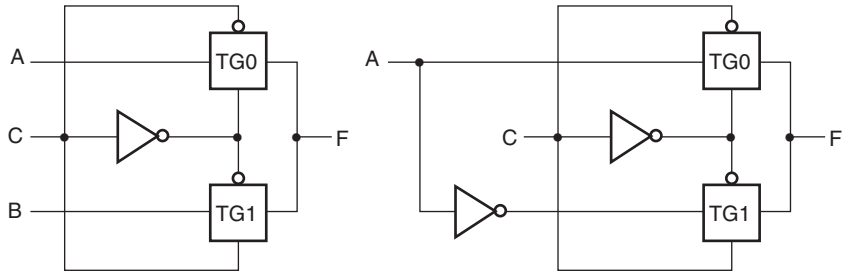
may be necessary to provide additional inverters or buffers on the inputs or between the gates and the inputs.

Transmission Gates

Besides primitive and complex gates and three-state buffers, there is an additional transistor circuit frequently used in CMOS logic. This circuit is the transmission gate (TG). It has its own symbol and is often included in gate-level logic circuit diagrams. A transmission gate is used as an electronic switch for making a connection between two points in a circuit. It consists of an n-channel transistor and a p-channel transistor in parallel, as shown in Figure 3(a). The two types of transistor are used because the p-channel transistor passes 1 (H) well and the n-channel transistor passes 0 (L) well. Figure 3(b) is the switch model for the transmission gate.

Here X is the input, Y is the output, and the two terminals C and $\overline{C}$ are control inputs. If $C = 1$ (H) and $\overline{C} = 0$ (L), there is a path between X and Y for the signal to pass through. If $C = 0$ and $\overline{C} = 1$, there is no path, and the circuit behaves like an open switch. The IEEE symbol for the transmission gate is given in Figure 3(c). The symbol in Figure 3(d), however, is more popular in general use. This symbol consisting of two overlapped triangles serving as reminder that, unlike other gates in which signals flow from inputs to output, signals flow through a transmission gate in both directions. We say that the transmission gate is bidirectional, a property that it shares with a mechanical switch. In most applications, we do not expect signals to flow from output to input. Thus, circuits containing transmission gates must be carefully designed to prevent unintentional flow of signals from outputs back to inputs and to prevent paths from multiple inputs to an output node to be present simultaneously. For the transmission gate to be properly controlled, the control inputs must turn the n-channel and p-channel devices on or off simultaneously, so C and $\overline{C}$ must be provided as shown in Figure 3(e).

Transmission gates are particularly useful for performing selection functions. A TG-based circuit that selects one of two values A and B to apply to an output F is shown in Figure 4(a). If $C = 0$, then a path exists through TG0 connecting F to A, and no path exists through TG1. If $C = 1$, then a path exists through TG1 connecting F to B, and no path exists through TG0. In Chapter 3, such a selection circuit is a 2-to-1-line multiplexer. So we call this circuit a transmission gate–based multiplexer.



(a) (b)

A	C	TG1	TG0	F
0	0	No path	Path	0
0	1	Path	No path	1
1	0	No path	Path	1
1	1	Path	No path	0

(c)

FIGURE 4
Selector and Exclusive-OR Constructed with Transmission Gates

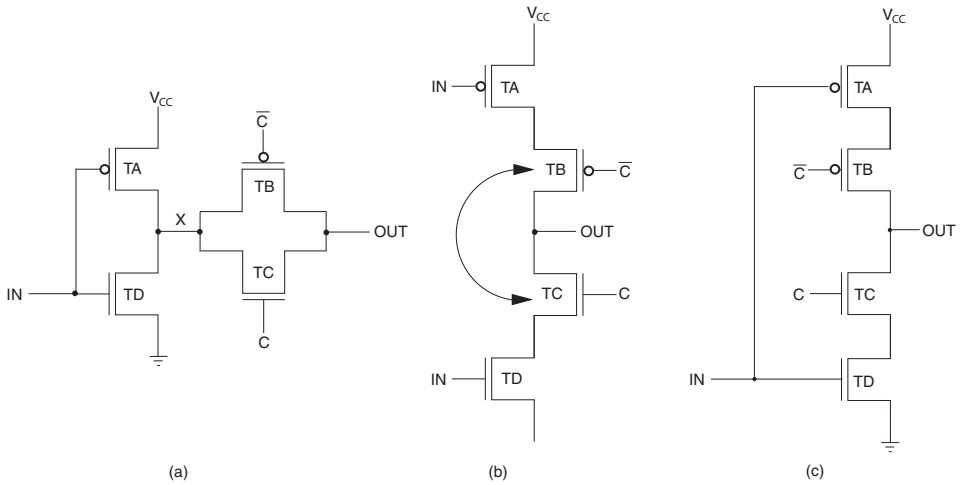


FIGURE 5
Invert Plus Transmission Gate to Inverting Three-State Buffer

By making B = $\overline{A}$ for the selector, an exclusive-OR gate can be constructed with two transmission gates and two inverters, as shown in Figure 4(b). Input *C* controls the paths in the transmission gates, and input *A* and its inverse provide the output for *F*. If input *C* is equal to 1, a path exists through transmission gate TG1 connecting *F* to $\overline{A}$, and no path exists through TG0. If input *C* is equal to 0, a path exists through TG0 connecting *F* to *A*, and no path exists through TG1. Thus, the output *F* is connected to *A*. This results in the exclusive-OR truth table, as indicated in Figure 4(c).

Inverter -Transmission Gate Combination Versus Inverting Three-State Buffer

In addition to being bidirectional, the transmission gate is a redundant circuit in the sense that both of the transistors are not necessary for the transmission gate to function. If one of the transistors fail to turn on (an open circuit failure), the other transistor alone turning on and off can provide the necessary path switching. While this tolerance for failure may seem like a good thing, there is a disadvantage. This failure produces an increase in the propagation delay through the transmission gate or a degrading of one of the two voltage levels on the output of the transmission gate. In contrast to the case in which the function of the transmission gate becomes faulty, these failure symptoms are much more difficult to detect, requiring, for example, a test for increased propagation delay instead of simply a test for correct changes in Boolean output values.

We now consider the situation in a circuit in which each transmission gate is preceded by an inverter as shown in Figure 5(a). Note the circuit node X in this circuit. This node can be split so that TA is no longer connected to TD as shown in Figure 5(b) without changing the function of the circuit with respect to its 0, 1, and high-impedance outputs. The circuit with the node split can be redrawn as Figure

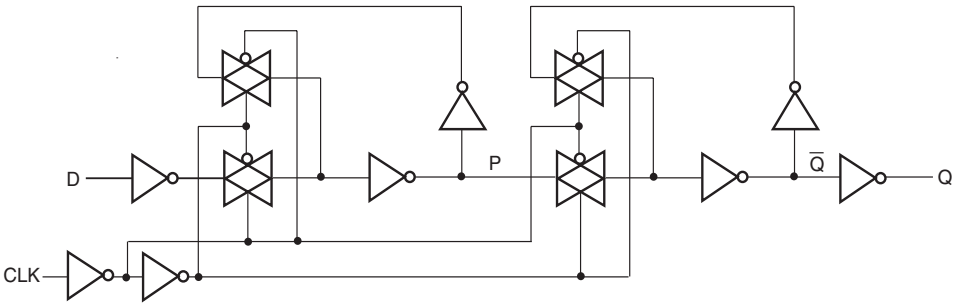


FIGURE 6
Transmission Gate-Based Edge-Triggered D Flip-flop

5(c) which is identical to the inverting 3-state output portion of circuit shown in Figure 2(a). This transformation shows that an a inverter/TG combination and an inverting3-state buffer circuit have identical functions. The key change is that the 3-state buffer is no longer a redundant circuit and can be tested just by testing its output changes rather than requiring a propagation delay test or some other electrical test to determine if a transistor has an open circuit failure. Thus by replacing inverter/TG pairs with inverting 3-state buffers, we can eliminate this testing problem. In those cases where there is no inverter, one can be added. A similar transformation can also be performed with pairs containing gates more complex than inverters as the first gate, permitting other inverting functions to be combined with 3-state output capability.

Transmission Gate-Based Flip-Flop

Transmission gates have also played a prominent role in efficient implementation of CMOS latches and flip-flops. A transmission gate-based edge-triggered D flip-flop is shown in Figure 6. This flip-flop is made up of two latches, one with output P and one with output Q. With CLK = 0, there is a path through the lower transmission gate from $\overline{D}$ into the first latch so $\overline{D}$ appears at P. Further, the upper transmission gate is open, so no feedback path is provided from P back to P. Also, with CLK = 0, a feedback path exists through the upper transmission gate and the pair of inverters in the second latch storing the value $\overline{Q}$ which drives output Q. Also, there is no path from P into the second latch. When CLK changes to 1, in the first latch, the path from $\overline{D}$ is removed and the upper transmission gate provides a feedback path from $\overline{P}$ to $\overline{P}$ storing the value of $\overline{D}$. Further, the path from $\overline{D}$ into the latch is broken. At the same time, CLK = 1 closes the path from P into the second latch connecting $P = D$ to output Q. Further, the feedback path storing the previous value of $\overline{Q}$ is broken. Thus, the value $\overline{D}$ is stored in the first latch and value D is applied to the output Q. So the state of the flip-flop has been changed to D. When CLK changes to 0, the output remains unchanged, but is now stored in the loop from $\overline{Q}$ to $\overline{Q}$ in the second latch.

References

1. WESTE, N. H. E., AND ESHRAGHIAN, K. *Principles Of CMOS VLSI Design: A Systems Perspective*, 2nd ed. Reading, MA: Addison-Wesley, 1993.
2. RABAEY, J. M., CHANDRAKASAN, A., AND NIKOLIC, B. *Digital Integrated Circuits*: *A Design Perspective.* 2nd ed. Upper Saddle River, NJ: Pearson Education, Inc., 2003.

Problems

1. Find the CMOS complex gate circuit for each of the following functions. In each case, use a minimum number of transistors.

(a) $F(X,Y,Z) = YZ + \overline{X}Z + \overline{X}Y\overline{Z}$

(b) $F(A,B,C,D) = A\overline{D} + A\overline{B} + BD + BC$

2. Find the CMOS complex gate circuit for each of the following functions:

(a) $F(A,B,C,D) = (A + \overline{C})(\overline{A} + C)(B + \overline{D})(\overline{B} + D)$

(b) $F(W,X,Y,Z) = \Sigma m(4,7,9,11,12,13,14,15)$, $d(W,X,Y,Z) = \Sigma m(3,10)$

3. Find a multiple-level NAND circuit for F in Problem 2(a), and compare the number of transistors used with the number used in that problem. A NAND gate with n inputs uses $2n$ transistors.

4. Verify that the two circuits given in Figure 2 implement 3-state buffers with the same function.

5. Construct an exclusive-NOR circuit with two NOT gates and two transmission gates.

6. Construct a 4-to-1-line multiplexer using transmission gates and inverters.

7. On the copy of Figure 6 given below, (a) mark the closed paths through the circuit for CLK = 0 in one color and (b) mark the closed paths through the circuit for CLK = 1 in a different color. (c) Using the results in a and b explain how the circuit implements the positive edge-triggered D flip-flop function.

8. Redesign the D flip-flop in Figure 6 by replacing inverter/transmission gate pairs with three-state buffers. Add or rearrange inverters as necessary to produce inverter/transmission gate pairs for replacement.