Digital multiplexers reduce chip count in logic design

by James E. Siebert
Michigan State University, East Lansing, Mich.

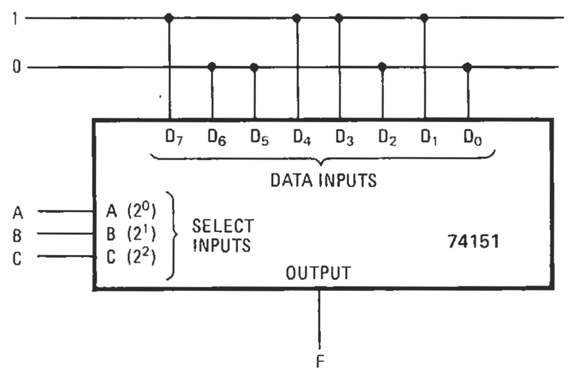
When attempting circuit optimization, logic designers traditionally use algorithms directed at minimizing gate count, but not chip count. However, if digital multiplexers are used to implement the logic functions in a circuit containing less than six input variables, chip count may often be minimized. This can usually be done without tradeoffs in cost, speed, or propagation delay.

Multiplexers in the transistor-transistor-logic family are available in quad two-input, dual four-input, or single eight- and 16-input devices. A two-variable logic block of up to four functions can be generated by a quad two-input device, with the constraint that one variable is common to each function. A three-variable function can normally be realized by using a single eight-input multiplexer, and a four-variable function can be implemented with a single 16-input multiplexer.

The primary design tool is the truth table of the particular function desired. For example, a function of three variables such as $F = AB+A\overline{C}+\overline{A}\,\overline{B}C$ may be implemented with a 74151 eight-input multiplexer. Using its truth table, the desired function is implemented as shown in Fig. 1. The multiplexer-select lines serve as the inputs for variables A, B and C. Data inputs on the multiplexer corresponding to the select addresses are tied high or low to provide the proper output. Implementing the same function using gates requires two three-input NANDs, two two-input NANDs, and inverters to derive the complement of the input variables for a total of 1⅔ packages.

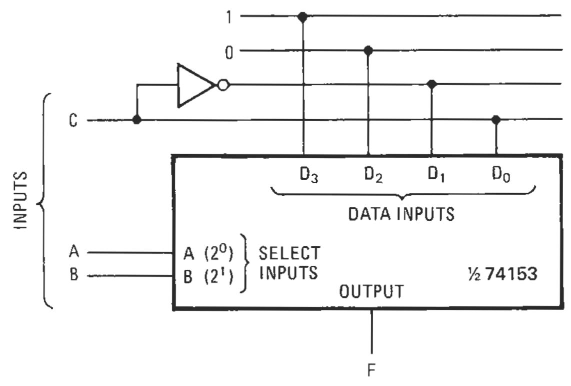
By providing one variable at the data input, a lower-order multiplexer may be used to realize the function. That is, a three-variable function can be implemented with only a four-input multiplexer, where an eight-input device was previously required. As shown in Fig. 2, partitioning the function table can aid in minimizing

SELECT ADDRESS			OUTPUT
C	B	A	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

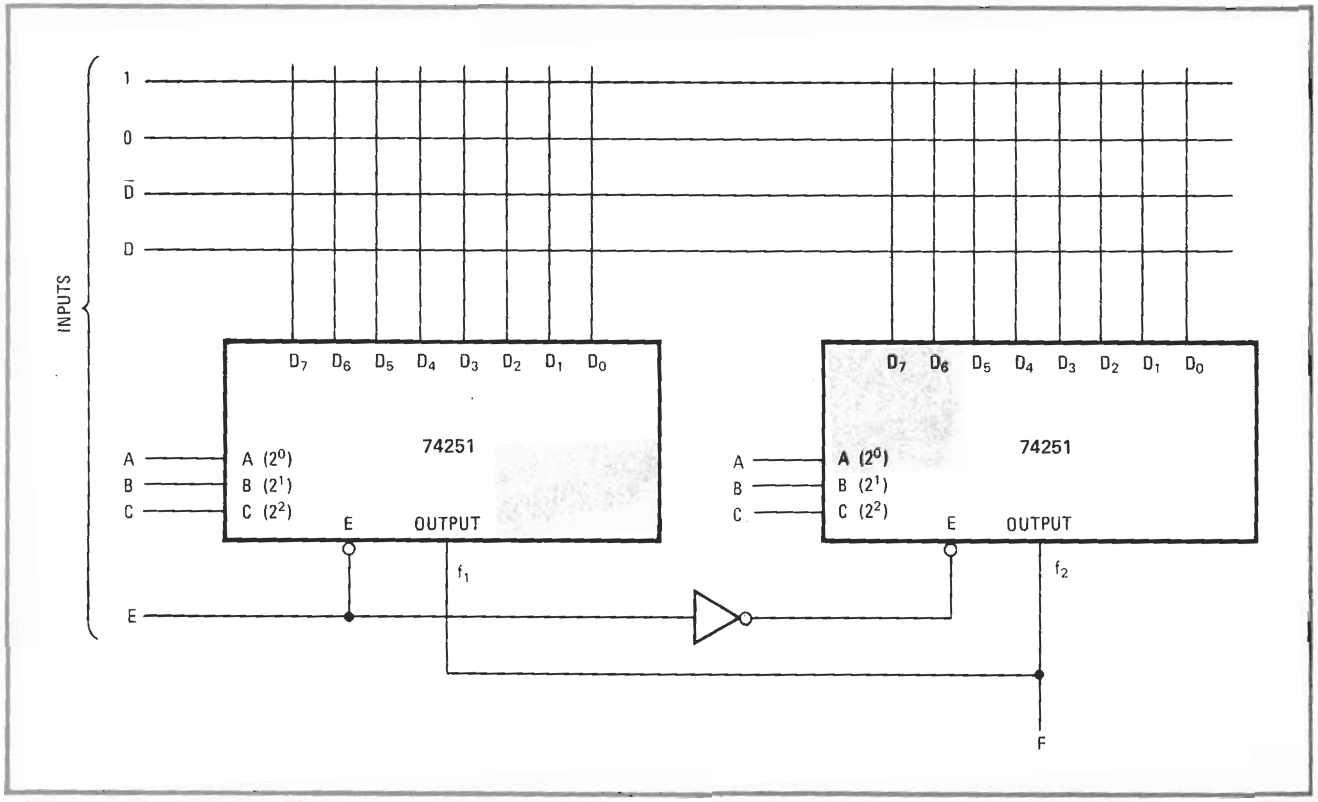


1. One-chip solution. This circuit implements the logic function $F = AB+A\overline{C}+\overline{A}\,\overline{B}C$. Variables A, B, C steer select inputs, and data inputs are connected HIGH or LOW depending on the multiplexer function. Discrete-gate implementation requires 1⅔ packages.

INPUTS			OUTPUT
SELECT ADDRESS			
B	A	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



2. Increased-efficiency circuit. Reduction of an 8-input multiplexer to a 4-input multiplexer is possible if one variable is introduced at data inputs (other variables form the select addresses). Truth table is partitioned for variable C, and design proceeds as explained in text.



3. Five-variable realization. Two three-state eight-input multiplexers can generate any five-input function. Variable E is partitioned, and two four-input problems are solved when implementing function. This can also be accomplished by any 16-input multiplexer.

design time. The single partitioned variable is made available to the data inputs; the other variables form the select addresses. By comparing the function output and the partitioned variable for each select address, multiplexer data-input connections can be readily determined. This method is also useful when implementing a four-variable function with an eight-input multiplexer. However, an eight-input multiplexer usually cannot be used for five-variable functions.

With one 24-pin package, a five-input function may be realized either by the above method or by an alternative one. The first method is needed to generate a five-variable function with a 16-input multiplexer; four variables form the select address, and the partitioned variable serves the inputs. An alternative approach is to use the three-state output available on some multiplexers. In this case, two 16-pin packages are needed, as shown in Fig. 3. The variable D is made available at the data inputs, but E is partitioned for ease in design in the truth table. There are two four-input problems to be solved, and each may be resolved by the method previously described. Variable E actually selects one of the two multiplexer functions.

Minimizing more than a five-variable combinational function is an unwieldy problem. This design approach does not result in a low package count or propagation delay. A six-variable function requires five packages. Generation of a seven-variable function requires nine multiplexers, and an eight-variable function requires 17 multiplexers. □

Calculator notes