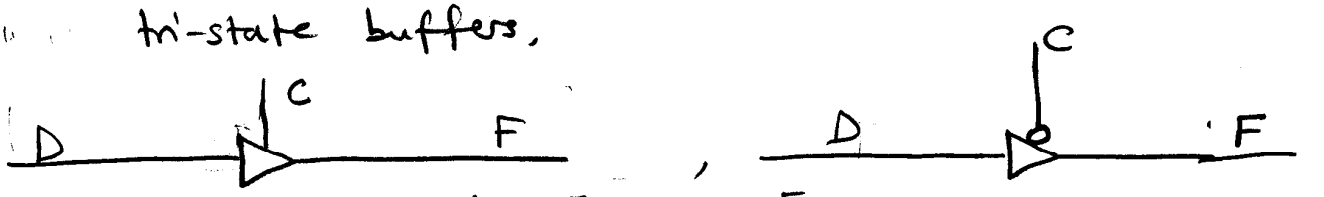
Problem 18

(40 points) In this problem, you will get practice with tri-state buffers.



[enabled when C is high] , [enabled when C is low.]

(a) Why is it called a "tri-state buffer"? What are the 3 values that F can take?

(b) Implement a 2:1 mux using only tristate buffers.

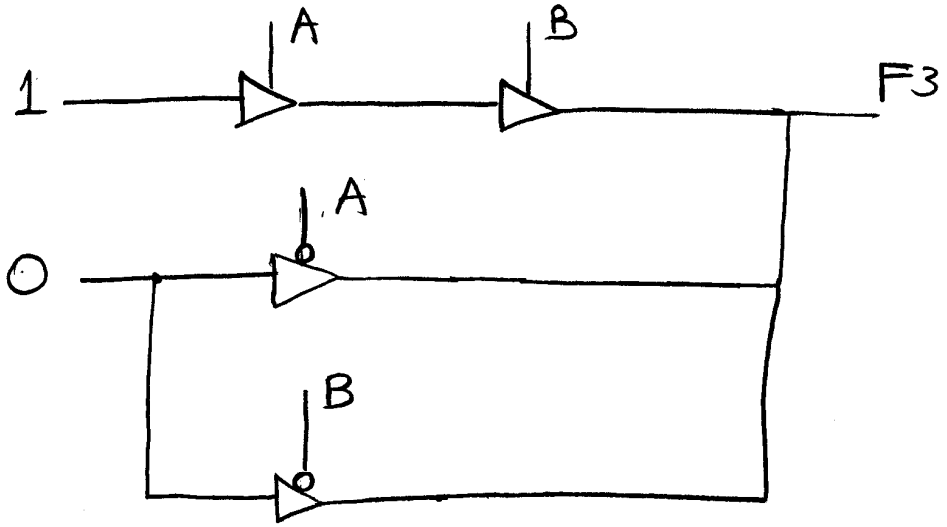
(c) Implement a 4:1 mux using only tristate buffers.

(d) Now, we would like to implement a 2:4 decoder using only tristate buffers.

A	B	F3	F2	F1	F0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Fig: Truth table for a 2:4 decoder.

A possible implementation is as follows: Let us examine F3: We want to route the logic value 1 to F3 when A and B are both 1; and route the logic value of 0 to F3 otherwise.



this part implements A·B (ON-terms)

this part implements $(A \cdot B)' = A' + B'$ (OFF-terms)

Note how the tristate buffers are used above to build a "switching network". The AND of two inputs is implemented as two tristate buffers in series, and the OR of two inputs is implemented as two tristate buffers in parallel.

- Using a similar approach, implement F2, F1, and F0.
- How many tristate buffers have you used in total to implement a 2:4 decoder?

(e) What is the advantage of using tri-state buffers rather than AND, OR, NOT gates? What might be the disadvantage?

(f) Can every Boolean function be implemented using only tri-state buffers? If so, give the sketch of a proof. If not, give a counterexample.