Computation Structures

Combinational Logic Worksheet

Concept Inventory:

- Truth tables ↔ sum-of-products equations
- implementation using NOT/AND/OR
- Demorgan's Law, implementation using NAND/NOR
- Simplification, truth tables w/ don't cares
- Karnaugh maps
- Implementation using MUXes and ROMs

Here's a Design Approach

Truth Table

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

1. Write out our functional spec as a truth table
2. Write down a Boolean expression with terms covering each '1' in the output:

$$Y = \bar{C}B\bar{A} + \bar{C}BA + C\bar{B}\bar{A} + CBA$$

3. We'll show how to build a circuit using this equation in the next two slides.

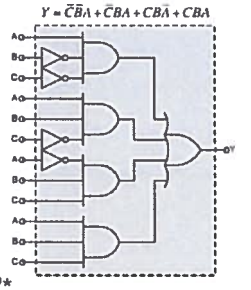
This approach will always give us Boolean expressions in a particular form: SUM-OF-PRODUCTS

Straightforward Synthesis

We can implement SUM-OF-PRODUCTS with just three levels of logic:

1. Inverters
2. ANDs
3. OR

Propagation delay -- No more than 3 gate delays?*



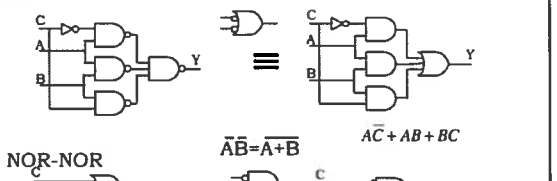
*assuming gates with an arbitrary number of inputs, which, as we'll see, isn't a good assumption!

CMOS Sum-of-products Implementation

NAND-NAND

$\overline{AB} = \bar{A} + \bar{B}$

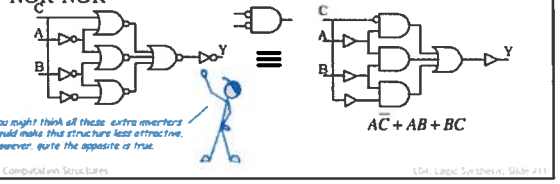
"Pushing Bubbles"



$A\bar{C} + AB + BC$

NOR-NOR

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



$A\bar{C} + AB + BC$

You might think all these extra inverters would make this structure less attractive. However, quite the opposite is true.

Finding Implicants

An implicant

- is a rectangular region of the K-map where the function has the value 1 (i.e., a region that will need to be described by one or more product terms in the sum-of-products)
- has a width and length that must be a power of 2: 1, 2, 4
- can overlap other implicants
- is a prime implicant if it is not completely contained in any other implicant.

C\AB	00	01	11	10
0	0	0	1	1
1	1	0	0	0

$A\bar{C}$

$\bar{A}BC$

C\AB	00	01	11	10
0	1	0	0	1
1	1	1	0	1

$\bar{A}\bar{C}$

$\bar{B}$

- can be uniquely identified by a single product term. The larger the implicant, the smaller the product term.

Write Down Equations

Picking just enough prime implicants to cover all the 1's in the KMap, combine equations to form minimal sum-of-products.

C\AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$Y = A\bar{C} + BC$

We're done!

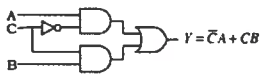
\AB C|	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	0	0	1

$Y = D + B\bar{C} + A\bar{C} + \bar{B}C$

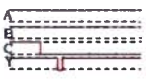
Minimal SOP is not necessarily unique!

Prime Implicants, Glitches & Leniency

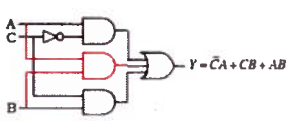
This circuit produces a glitch on Y when A=1, B=1, C: 1→0



C\AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0



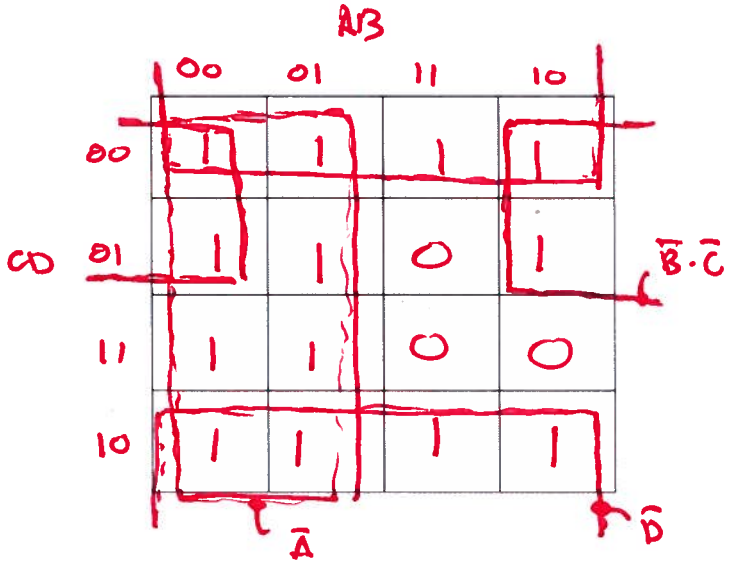
To make the circuit lenient, include product terms for ALL prime implicants.



Problem 1.

Given a function F defined by the truth table to the right, provide a minimal sum-of-products expression for F. Hint: Use a Karnaugh Map.

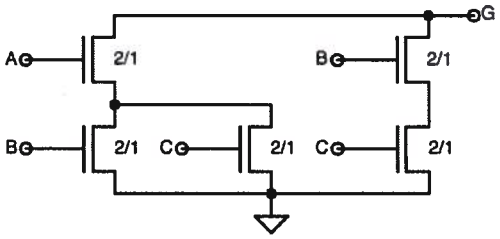
A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0



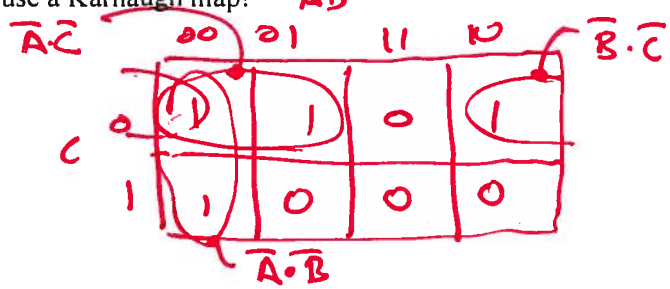
Minimal Sum-of-products Expression for F: $\bar{A} + \bar{D} + \bar{B}\cdot\bar{C}$

Problem 2.

(A)A correctly-formed CMOS gate implementing G(A,B,C) uses the pulldown circuit shown on the right. Please give a *minimal sum-of-products expression* for G(A,B,C). Hint: use a Karnaugh map!



FILL IN 0's USING PULLDOWNS; REST ARE 1's



Minimal sum-of-products expression for G(A,B,C): $\bar{A}\cdot\bar{B} + \bar{A}\cdot\bar{C} + \bar{B}\cdot\bar{C}$

(B) Is the function G(A,B,C) from part (B) universal? In other words, can we implement any Boolean function using combinational circuits built only from G gates and the Boolean constants 0 and 1?

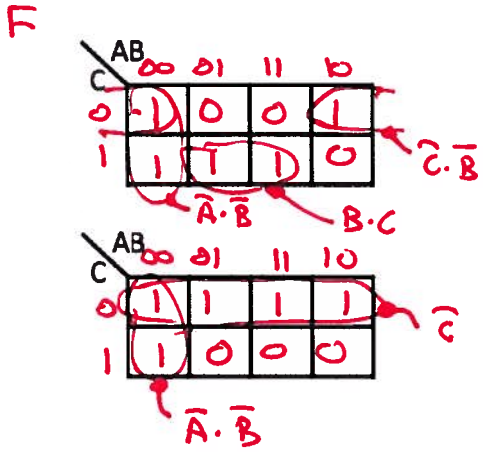
YES

G(0,B,C) == NAND(B,C)

Problem 3.

Consider the following truth table which defines two functions F and G of three input variables (A, B, and C).

A	B	C	F	G
0	0	0	1	1
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	0



Give the **minimal sum of products** (minimal SOP) logic equation for each of the two functions. Then determine if the minimum sum of products expression would result in a **lenient** implementation of the function. If it does, then enter "**SAME**" for the lenient SOP expression. **If not**, specify what sum of products expression would result in a lenient implementation. Hint: Use Karnaugh maps above to determine the minimal sum of products.

Minimal sum of products F(A,B,C) = $\bar{A}\cdot\bar{B} + B\cdot C + \bar{B}\cdot\bar{C}$

Does minimal SOP for F result in a lenient circuit (circle one)? Yes (No)

consider A=0, C=0, B: 0→1.

If "No", give lenient SOP expression for F(A,B,C) = $\bar{A}\cdot\bar{B} + B\cdot C + \bar{B}\cdot\bar{C} + \bar{A}C$

extra implicant!

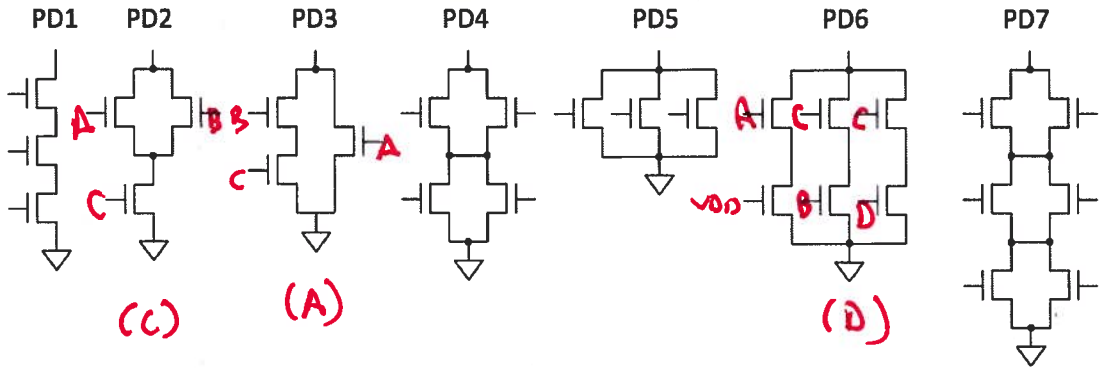
Minimal sum of products G(A,B,C) = $\bar{A}\cdot\bar{B} + \bar{C}$

Does minimal SOP for G result in a lenient circuit (circle one)? (Yes) No

If "No", give lenient SOP expression for G(A,B,C) = —

Problem 4.

You are trying to select pulldowns for several 3- and 4-input CMOS gate designs. The Pulldowns-R-Us website offers seven different pulldowns, given names PD1 through PD7, diagrammed below:



The web site explains that the customer can choose which inputs or constants (GND, VDD) are connected to each NFET, allowing their pulldowns to be used in various ways to build gates with various numbers of inputs. Since Pulldowns-R-Us charges by transistor, you are interested in selecting pulldowns using the minimum number of transistors for each of the 3-input gates you are designing.

For each of the following 3- and 4-input Boolean functions, choose the appropriate pulldown design, i.e., the one which, properly connected, implements that gate's pulldown using the *minimum number* of transistors. This may require applying Demorgan's Laws or minimizing the logic equation first. If none of the above pulldowns meets this goal, write "NONE".

(A) $F(A,B,C) = \overline{A + (B \cdot C)}$ **Choice or NONE:** PD3

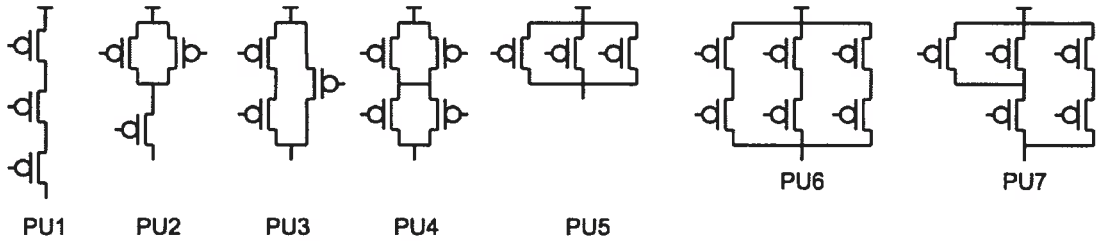
(B) $F(A,B,C) = A + \overline{B \cdot C}$ non inverting! **Choice or NONE:** NONE

(C) $F(A,B,C) = (\overline{A} \cdot \overline{B}) + \overline{C} = \overline{(A+B)} + \overline{C}$
$= \overline{(A+B) \cdot C}$ **Choice or NONE:** PD2

(D) $F(A,B,C,D) = \overline{A + C \cdot (B + D)}$ **Choice or NONE:** PD6

Problem 5.

You are trying to select pullups for several 3-input CMOS gate designs. The Pullups Galore web site offers seven different pullups, given names PU1 through PU7, diagrammed below:



The web site explains that the customer can choose which inputs are connected to each PFET, allowing their pullups to be used in various ways to build gates with various numbers of inputs. Since Pullups Galore charges by transistor, you are interested in selecting pullups using the minimum number of transistors for each of the 3-input gates you are designing.

For each of the following 3-input Boolean functions, choose the appropriate pullup design, i.e., the one which, properly connected, implements that gate's pullup using the *minimum number* of transistors. This may require minimizing the logic equation first. If none of the above pullups meets this goal, write "NONE".

(A) $F(A,B,C) = \overline{A} + \overline{B} + \overline{C}$ **Choice or NONE:** PU5

(B) $F(A,B,C) = \overline{A} + \overline{B \cdot C}$ same as (A)! **Choice or NONE:** PU5

(C) $F(A,B,C) = \overline{A + B \cdot C}$ $= \overline{A} \cdot (\overline{B} + \overline{C})$ **Choice or NONE:** PU2

(D) $F(A,B,C) = A + \overline{B \cdot C}$ not inverting! **Choice or NONE:** NONE

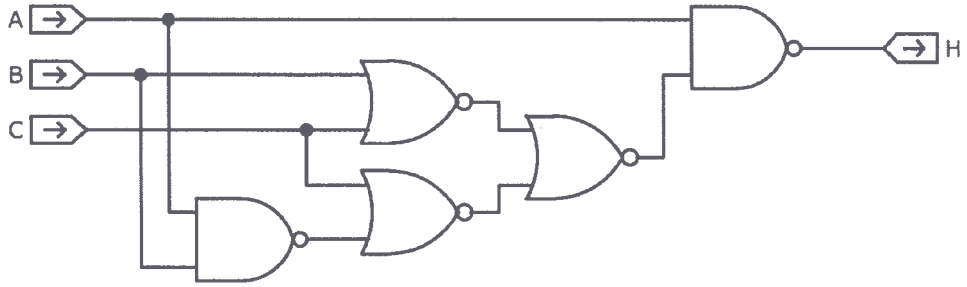
(E) $F(A,B,C) = \overline{(A+B)} + \overline{(B+C)} + \overline{(A+C)}$
$= \overline{A} \cdot \overline{B} + \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{C} = \overline{A} \cdot (\overline{B} + \overline{C}) + \overline{B} \cdot \overline{C}$ **Choice or NONE:** PU7 *

(F) $F(A,B,C) = \overline{(A+C) \cdot B}$
$= \overline{A} \cdot \overline{C} + \overline{B}$ **Choice or NONE:** PU3

* PU6 would also work, but uses more PFETs

Problem 6.

Consider the Boolean function $H(A,B,C) = \overline{A} + \overline{B}\cdot\overline{C} + A\cdot B\cdot\overline{C}$. Its truth table is shown to the right and a possible implementation is shown in the schematic below.



A	B	C	H
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(A) Give a minimal sum-of-products expression for H. A couple of scratch 3-input Karnaugh map templates are provided for your convenience.

minimal sum-of-products expression for H: $\overline{A} + \overline{C}$

AB

H	00	01	11	10
0	1	1	1	1
1	1	1	0	0

C

$\overline{C}$

$\overline{A}$

	00	01	11	10
0				
1				

(B) What is the largest number of product terms possible in a minimal sum-of-products expression for a 3-input, 1-output Boolean function?

Largest number of product terms possible: 4

	00	01	11	10
0				
1				

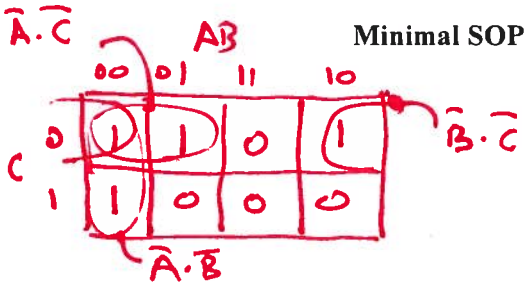
Think of a checker board pattern of 1's. ⇒ 4 product terms.

if you add another product term, k-map can be used to simplify

Problem 7.

A minority gate has three inputs (call them A, B, C) and one output (call it Y). The output will be 0 if two or more of the inputs are 1, and 1 if two or more of the inputs are 0.

(A) Give a *minimal* **sum-of-products** Boolean expression for the minority gates output Y, in terms of its three inputs A, B, and C.



Minimal SOP expression: Y = $\bar{A}\cdot\bar{B} + \bar{A}\cdot\bar{C} + \bar{B}\cdot\bar{C}$

(B) Is a minority gate *universal*, in the sense that using only minority gates (along with constants 0 and 1) its possible to implement arbitrary combinational logic functions?

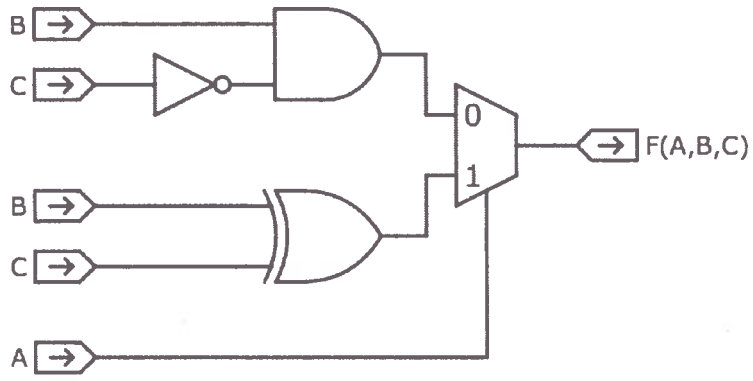
Universal? Circle one: **YES** **can't tell** **NO**

minority(A,B,1) == NOR(A,B)

↑ known to be universal

Problem 8.

A 6.004 intern at Intel has designed the combinational circuit shown below. His boss can't figure out what it does and has asked for your help.



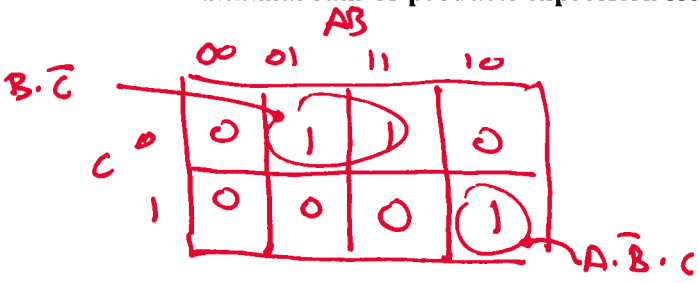
A	*B*	*C*	*F(A,B,C)*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	0	0

(A) Please fill in the truth table for F(A,B,C) above.

Fill in truth table above

(B) Express F(A,B,C) in minimal sum-of-products form. Hint: use a Karnaugh map!

minimal sum-of-products expression for F(A,B,C) = $A \cdot \bar{B} \cdot C + B \cdot \bar{C}$

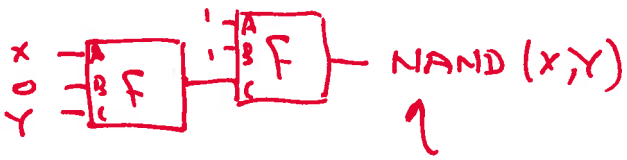


(C) The boss isn't quite sure what it means but he knows his engineers are always impressed if he asks "is the circuit universal?" Is it? Circle YES or NO.

F(A,B,C) universal? YES .. NO

$F(1,1,C) = \bar{C}$

$F(A,0,C) = A \cdot C$

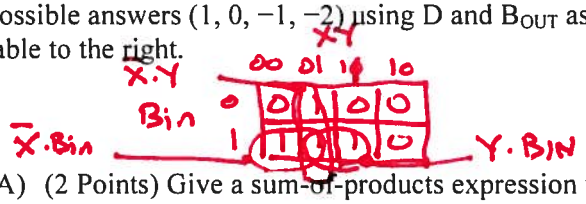


known to be universal

Problem 9.

The full subtractor (FS) implements a one-column binary subtraction of two bits (X and Y) producing their difference (D), accepting a borrow-in (B_{IN}) from the previous column and producing a borrow-out (B_{OUT}) for the next column. Numerically FS computes $X-Y-B_{IN}$ and encodes the possible answers (1, 0, −1, −2) using D and B_{OUT} as shown in the truth table to the right.

X	Y	B_{IN}	D	B_{OUT}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



(A) (2 Points) Give a sum-of-products expression for B_{OUT}.

Sum of products expression: B_{OUT} = $\bar{X}\cdot Bin + Y\cdot Bin + \bar{X}\cdot Y$

(B) (1 Point) Is the FS(X,Y,B_{IN}) circuit universal in the sense that 2-input NOR and 2-input NAND are universal? In other words, using only acyclic networks of FS circuits (perhaps with one or more of their inputs tied to "0" or "1"), can one implement any combinational logic function?

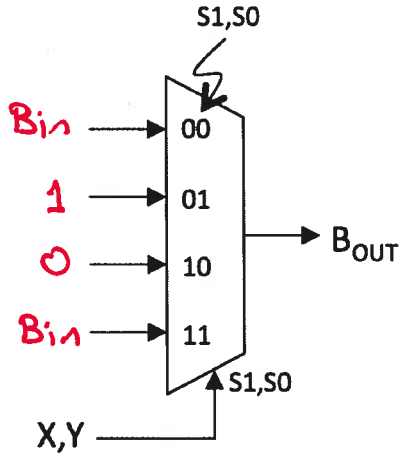
F(0,Y,Bin) = OR(Y,Bin) using B_{OUT}
F(1,Y,Bin) = AND(Y,Bin) using B_{OUT}
F(0,1,Bin) = NOT(Bin) using D

FS Universal: ... YES ... NO ...

← everything you need to build sum-of-products

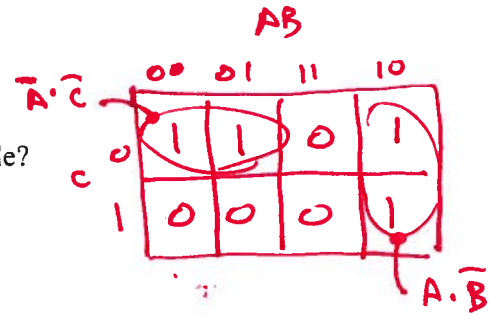
(C) (2 Points) You're trying to build an implementation for the B_{OUT} part of the FS circuit (see truth table above) but discover that the NITGFOC supply room only has 4-to-1 multiplexors in stock. In desperation, you call up your 6.004 TA who says "No problem! In fact, you can produce B_{OUT} with just a single 4-to-1 mux: connect X to S1 and Y to S0, then hook each of the data inputs to the appropriate choice of '0', '1' or B_{IN}." Using this hint, finish off the implementation shown below.

Show connections for data inputs using only '0', '1' or B_{IN}



Problem 10.

The 3-input Boolean function G(A,B,C) computes $\overline{A}\cdot\overline{C}+A\cdot\overline{B}+\overline{B}\cdot\overline{C}$.



(A) How many 1's are there in the output column of G's 8-row truth table?

4

(B) Give a minimal sum-of-products expression for G.

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

(C) There's good news and bad news: the bad news is that the stockroom only has G gates. The good news is that it has as many as you need. Using only combinational circuits built from G gates, one can implement (choose the best response)

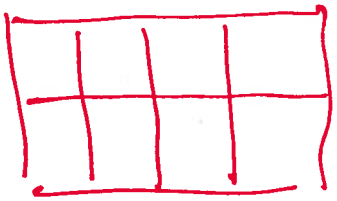
(A) only inverting functions
~~(B)~~ only non-inverting functions
(C) any function (G is universal)
(D) only functions with 3 inputs or less
(E) only functions with the same truth table as G

G(A,1,C) == NAND(A,C)

(D) Can a sum-of-products expression involving 3 input variables with greater than 4 product terms *always* be simplified to a sum-of-products expression using fewer product terms?

YES (see 6B)

Think of 3-input K-map



↑ if more than 4 1's, two must be adjacent ⇒ can use patch to cover both 1's.

6.004 Computation Structures
Spring 2017