Read each question carefully before you answer. You have 1 hour and 50 minutes for the exam, which is on average 22 minutes per problem. If you find yourself spending more than 22 minutes on any problem, move on to the next problem and come back to it later. Don't spend the majority of your time on any one problem if there are still other problems to solve, or you may risk running out of time before finishing the exam.

Make sure to show all your work in the space provided on the exam. Points will be taken off for answers without any work. Partial credit will be given where substantial work has been done.

Problem 1 (20 points)

Suppose we want to design a digital clock that displays the current time in hours and minutes as $H_1H_2 : M_1M_2$, such as, for example 12:30, which would set H_1=1, H_2=2, M_1=3, and M_2=0.

We will use the standard 12-hour format, so 04:00 can either represent 4am or 4pm (as opposed to the 24-hour military format such as 1600 for 4:00pm).

a) Determine for each digit (H_1, H_2, M_1, M_2) of the display:
   i. The range of each digit (i.e. what is the smallest value and the largest value, in decimal)
   ii. The minimum number of bits necessary to represent each digit in a standard binary format (not BCD)

i ✓ H_1 0 - 1

H_2 0 - 9.

M_1 0 - 5

M_2 0-9

ii ✓ H_1 1 bit

H_2 4 bit.

M_1 3 bit.

M_2 4 bit

12-bit in all.

b) ✓ Is the total number of bits for representation the minimum, or is there a more efficient (less number of bits) way to represent the time as a single component (which may need some digit conversion to display the time properly as $H_1H_2 : M_1M_2$)

If there is **not** a more efficient way to represent the time, explain why not.

If there **is** a more efficient way, explain why, and what is the minimum **total** number of bits to represent the time in hours and minutes.

12 hr × 60 min : 720. Can be represented in

$\log_2 720 < 10$. 10-bits.

We can convert hrs to minutes and count the time in a single component of minutes from 12:00, in binary format.

Problem 2 (20 points)

Prove or disprove the following equalities, by either constructing the corresponding tables, or using the laws of Boolean Algebra (you do not have to state the name of the law in each step). Note: ⊕ represents the XOR function, and ⊙ represents the XNOR function, + represents the OR function, and • represent the AND function.

✓ and

= not a or not b

not xor

a b

a a y

b b z

a ⊕ x

1 1 x

0 0 ✓

0 1 x

a) $((w \oplus x) \bullet (x \oplus y) \bullet (y \oplus z))' = (w \odot x) + (x \odot y) + (y \odot z)$

left = $(w \oplus x)' + (x \oplus y)' + (y \oplus z)'$

$= (w \odot x) + (x \odot y) + (y \odot z)$

= right side

Proved

not a ⊕ b

not a out and b

✓ b) $(((w'+x+y)' + (w+x'+y)' + (w+x+y')' + (w'+x'+y')')')' = w' \oplus x \oplus y'$

left = $((w'+x+y) \cdot (w+x'+y) \cdot (w+x+y') \cdot (w'+x'+$

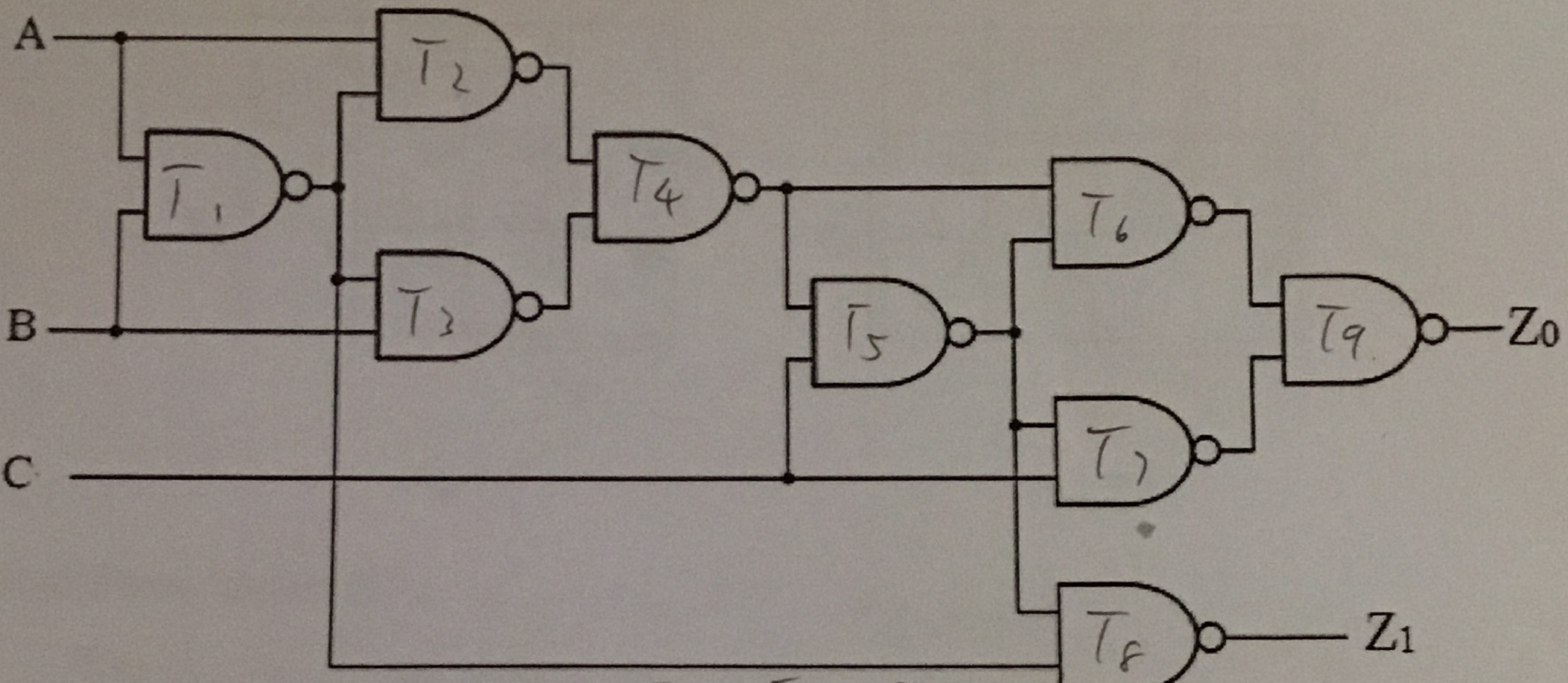
W	x	y.	(left)'	Right:	left
0	0	0	1	0.	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0.	1	1
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0.	1.	1

Right = left

Proved

Problem 3 (20 points)

a) Analyze the following network by determining the expressions for the outputs (Z_1, Z_0) based on the inputs (A, B, C).
{Hint: label each gate as a sub-function, T_i, and using a table, determine the result of each sub-function based on its inputs, until you reach the values for Z_1 and Z_0}
DO NOT CALCULATE THE CRITICAL DELAY OF THE NETWORK.



A	B	C		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8 (Z_1)	T_9 (Z_0)
0	0	0	0	1	1	1	0	1	1	1	0	0
0	0	1	1	1	1	1	0	1	1	0	0	1
0	1	0		1	1	0	1	1	0	1	0	1
0	1	1	2	1	1	0	1	0	1	0	1	1
1	0	0		1	0	1	1	1	0	1	0	1
1	0	1		1	0	1	1	0	1	~~0~~	~~0~~	0
1	1	0	2	0	1	1	0	1	1	1	1	0
1	1	1	3	0	1	1	0	1	1	0	1	1

$Z_1 = A'BC' + AB'C + ABC' + ABC. = \Sigma m(3, 5, 6, 7)$

$Z_2 = A'B'C + A'BC' + A'BC + AB'C' + AB'C + ABC = \Sigma m(1, 2, \cancel{3}, 4, 7)$

-2

b) From a high-level description, what mathematical function is the network performing? Do not state your answer simply as when the output equals 1 such as: "$z_1 = 1$ *when A=... and B=... and C=... or when A=... and B=... and C=...* ", but give a clear descriptive answer.

$Z_1 Z_0 = A + B + C.$

where $Z_1 Z_0$ is a two-bit binary number

Problem 4 (20 points)

a) Using the 5-input K-map below, find all the prime implicants of the function:
$f(x_4, x_3, x_2, x_1, x_0) = \Sigma m(7,11,13,14,15,19,21,22,23,25,26,27,28,29,30,31)$

x_0

0 0	1 0	3 0	2 0
4 0	5 0	7 1	6 0
12 0	13 1	15 1	14 1
8 0	9 0	11 1	10 0

x_2 x_3 x_1

$x_4 = 0$

x_0

0 0	1 0	3 1	2 0
4 0	5 1	7 1	6 1
12 1	13 1	15 1	14 1
8 0	9 1	11 1	10 1

x_2 x_3 x_1

$x_4 = 1$

List of Prime implicants: $X_3X_0X_1$, $X_3X_0X_2$, $X_2X_1X_0$, $X_1X_2X_3$

✓ $X_4X_0X_1$, $X_4X_3X_2$, $X_4X_0X_2$, $X_4X_2X_1$, $X_4X_1X_3$, X_4X_3X

✓ b) Which of the prime implicants from part a) are essential?

All of them are essential

✓ c) Obtain a minimal sum of products for *f*.

$f = X_3X_0X_1 + X_3X_0X_2 + X_2X_1X_0 + X_1X_2X_3 + X_4X_0X_1 + X_4X_3X_2 + X_4X_0$
$+ X_4X_2X_1 + X_4X_1X_3 + X_4X_3X_0$

d) Describe at a high-level what the function *f* is determining, i.e. when is *f*=1? Do not merely restate the problem by saying *f*=1 when *x*=7 or *x*=11 or *x*=13,… but give a clear explanation. {Hint: It is a not a complex function, but a common function used when comparing bits}

-2

$f = 1$ when at least any of three bits of th

five-bit $X_4X_3X_2X_1X_0$ is 1.

or $f = 1$ if $X_0 + X_1 + X_2 + X_3 + X_4 \geq 3$

Problem 5 (20 points)

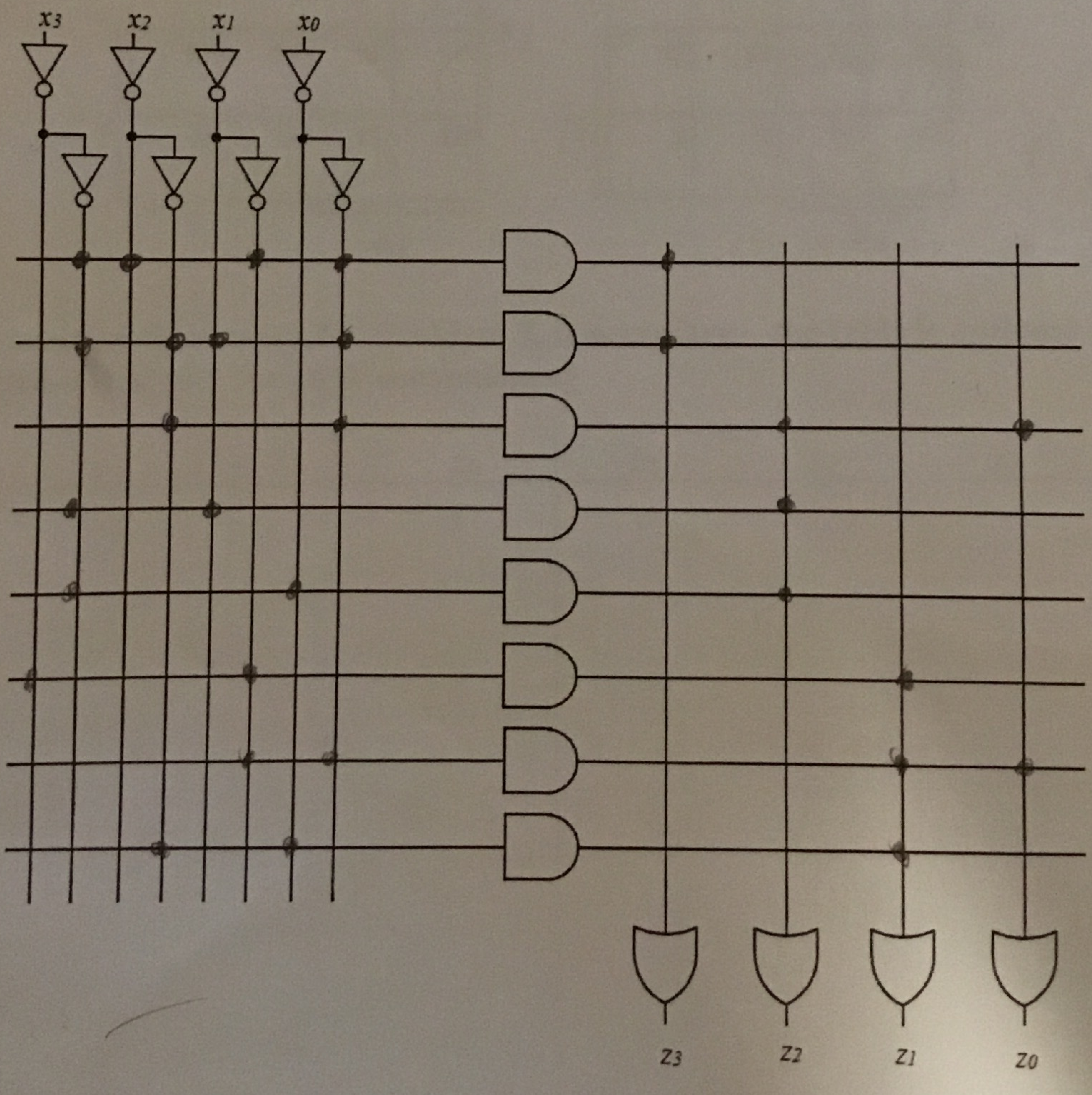
Using only the PLA below, without adding any rows of extra AND gates, implement a system that represents the following function $z = (z_3, z_2, z_1, z_0)$ given the input $x = (x_3, x_2, x_1, x_0)$, where $0 \leq x \leq 15$, and both x and z are represented in standard binary representation.

$$z = \begin{cases} x & \text{if } x \text{ is a prime number} \\ \lfloor x/2 \rfloor & \text{otherwise} \end{cases}$$

Be sure to label every connection made in the PLA with a dot: • at the appropriate intersection of a horizontal and a vertical line. Do not label unused connections with ×

{Note: x=0 and x=1 are not prime numbers, and a prime number is only divisible by itself and 1. The notation $\lfloor x/2 \rfloor$ represents the "floor" function of taking the integer part of diving by 2).

{Hint: Create a minimal sum-of-products for each output bit of z using either K-maps or laws of Boolean Algebra, and reuse prime implicants for multiple output bits of z}


x3
x2
x1
x0
z3
z2
z1
z0