ECE 120: Introduction to Computing

Fall 2024 – Midterm 2

Tuesday, October 22, 2024

Write your answers only in the space provided. We will not grade the back side of the pages

- Ensure that your exam booklet has 14 pages.
- **Please, write your NAME, NetID, and UIN clearly.**
- Do not tear the exam booklet apart. You can only detach the last two pages, (ASCII Table and scratch paper) if needed.
- This is a closed book/notes exam. You may use a calculator.
- You are allowed **one handwritten sheet** of notes (both sides). Write your name on the cheat sheet. The cheat sheet will be collected at the end of your exam.
- Absolutely no interaction between students is allowed.
- Clearly indicate any assumptions that you make.
- **The questions are not weighted equally**. Budget your time accordingly.
- Show your work and write legibly. Solutions in i**llegible handwriting will be graded as incorrect.**
- Write your **UIN (9-digit #) on each page in the provided space.**

NAME **ANSWER KEY**

NetID	UIN

Problem 1	18 points	_____
Problem 2	27 points	_____
Problem 3	21 points	_____
Problem 4	13 points	_____
Problem 5	21 points	_____
Total	100 points	_____

Problem 1 (18 points): K-maps and CMOS

Consider two 4-variable functions `F1(A,B,C,D) and F2(A,B,C,D)`, with the following K-maps**:**

Minimal SOP

F1(A,B,C,D)

AB \ CD	00	01	11	10
00	0	X	X	1
01	1	X	0	0
11	1	X	1	1
10	0	1	X	1

Minimal POS

F2(A,B,C,D)

AB \ CD	00	01	11	10
00	0	X	0	0
01	0	0	x	0
11	X	1	1	X
10	0	1	X	0

1. **(3 points)** Give a **minimal SOP** expression for `F1(A,B,C,D)` and show the corresponding loops on the **left map**.

   Minimal SOP: **F1(A,B,C,D)=**_____

2. **(3 points)** Is the solution to part 1 unique?

   Circle one: **UNIQUE** **NOT UNIQUE**

   If not unique, write a different minimal SOP solution (but do not mark the loops):

   Minimal SOP: **F1(A,B,C,D)=**_____

3. **(3 points)** Give a **minimal POS** expression for `F2(A,B,C,D)` and show the corresponding loops on the **right map**.

   Minimal POS: **F2(A,B,C,D)=**_____

4. **(3 points)** Is the solution to part 3 unique?

   Circle one: **UNIQUE** **NOT UNIQUE**

   If not unique, write a different minimal POS solution (but do not mark the loops):

   Minimal POS: **F2(A,B,C,D)=**_____

UIN----------

Minimal SOP

F1 (A,B,C,D)

AB \ CD	00	01	11	10
00	0	X	X	1
01	1	X	0	0
11	1	X	1	1
10	0	1	X	1

Minimal POS

F2 (A,B,C,D)

AB \ CD	00	01	11	10
00	0	X	0	0
01	0	0	x	0
11	X	1	1	X
10	0	1	X	0

1. **(3 points)** Give a **minimal SOP** expression for `F1(A,B,C,D)` and show the corresponding loops on the **left map**.

   Minimal SOP: **F1 (A,B,C,D) =** $\bar{B}C + AB + \bar{C}D + B\bar{C}$

2. **(3 points)** Is the solution to part 1 unique?

   Circle one: **UNIQUE** **NOT UNIQUE**

   If not unique, write a different minimal SOP solution (but do not mark the loops):

   Minimal SOP: **F1 (A,B,C,D) =** $(AD + \bar{B}C + AB + B\bar{C})$ OR $(\bar{B}C + \bar{C}D + AC + B\bar{C})$ $(\bar{B}C + AD + AC + B\bar{C})$

3. **(3 points)** Give a **minimal POS** expression for `F2(A,B,C,D)` and show the corresponding loops on the **right map**.

   Minimal POS: **F2 (A,B,C,D) =** AD

4. **(3 points)** Is the solution to part 3 unique?

   Circle one: **UNIQUE** **NOT UNIQUE**

   If not unique, write a different minimal POS solution (but do not mark the loops):

   Minimal POS: **F2 (A,B,C,D) =** NA

5. (6 points) Logic function with CMOS devices

Given the truth table (as shown below) implement the function Q using a single CMOS gate (i.e. the CMOS circuit will have only one gate delay).

hints: write the output Q = $\overline{(expression)}$. Then, apply duality principles.

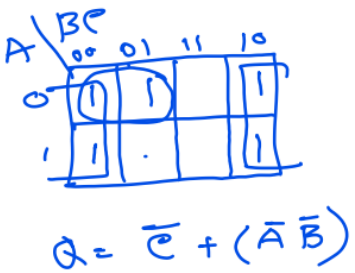
A	B	C	Output (Q)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

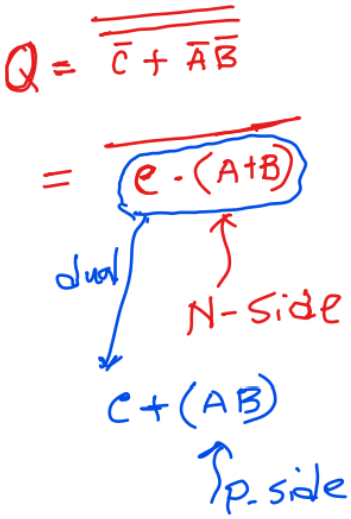
5. (6 points) Logic function with CMOS devices

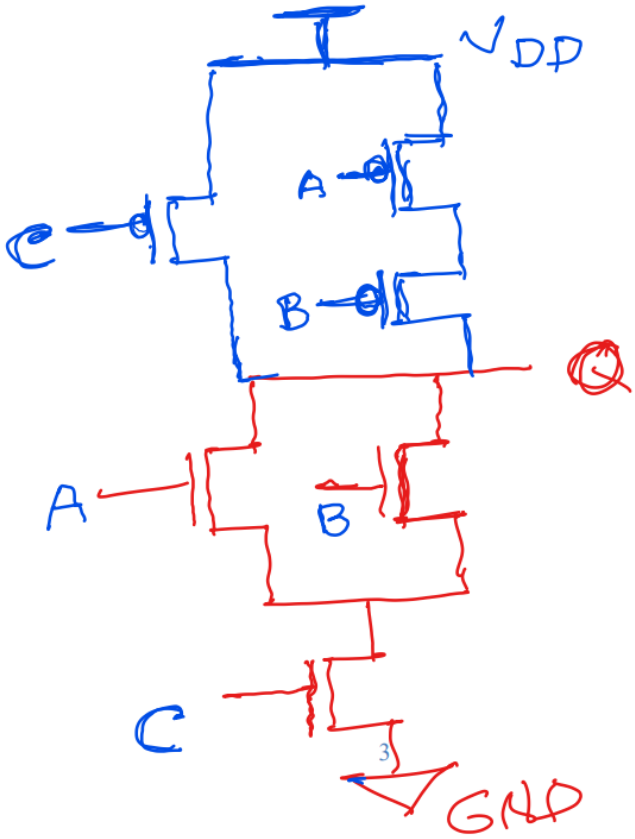
Given the truth table (as shown below) implement the function Q using a single CMOS gate (i.e. the CMOS circuit will have only one gate delay).

hints: write the output Q = $\overline{(expression)}$. Then, apply duality principles.

A	B	C	Output (Q)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

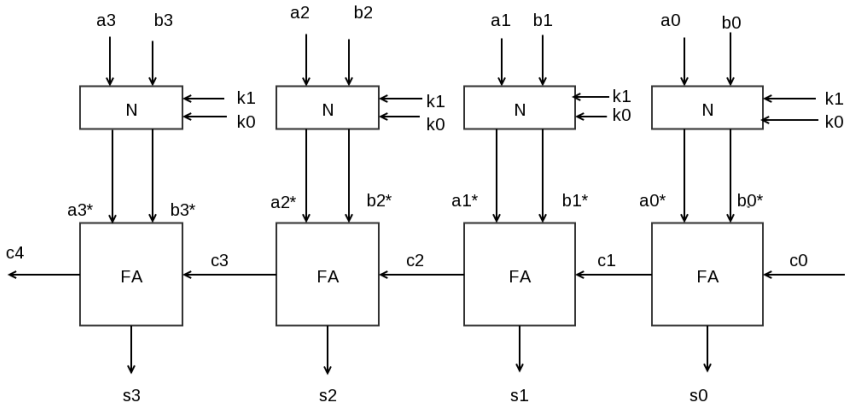






Problem 2 (27 points): Design of an Arithmetic unit (bit slice design)

Shown below is the structure of an arithmetic unit, consisting of four 1-bit Full Adders (FAs) and four copies of a circuit N:



k_1	k_0	Function
0	0	-A PLUS B
0	1	A PLUS B
1	0	A MINUS 1
1	1	-2A MINUS 1

Design a 4-bit arithmetic unit that accepts inputs a_3, ..., a_0, and b_3, ..., b_0, interpreted as the 4-bit two's-complement representations of integers A and B, and control inputs k_1, k_0, and performs the arithmetic operations defined in the above table.

1. **(9 points)** Fill in the following table by expressing values of a_i*, b_i*, and c_0 in terms of a_i, b_i, their complements, and constant binary values. The second row is already filled in for you as an example. *Hints:* $-A = \mathrm{NOT}(A)+1$

k_1	k_0	Operation	a_i*	b_i*	c_0
0	0	-A PLUS B			
0	1	A PLUS B	a_i	b_i	0
1	0	A MINUS 1			
1	1	-2A MINUS 1			

2. **(6 points)** Express a_i* and b_i* as functions of k_1, k_0, a_i, and b_i. Write their minimal SOPs: (you may use the provided scratch paper to draw the corresponding K-maps)

   a_i* = _____

   b_i* = _____

3. **(2 points)** Define c_0 as a function of k_1, k_0 using a single gate.

   c_0 = _____

1. **(9 points)** Fill in the following table by expressing values of a_i^*, b_i^*, and c_0 in terms of a_i, b_i, their complements, and constant binary values. The second row is already filled in for you as an example. *Hints:* –A = NOT(A)+1

$-2A-1 = \text{NOT}(A) + \text{NOT}(A) + 2 - 1 = (\text{NOT}(A) + \text{NOT}(A)) + 1$

k_1	k_0	Operation	a_i^*	b_i^*	c_0
0	0	-A PLUS B	$\bar{a}_i$	b_i	1
0	1	A PLUS B	a_i	b_i	0
1	0	A MINUS 1	a_i	1	0
1	1	-2A MINUS 1	$\bar{a}_i$	$\bar{a}_i$	1

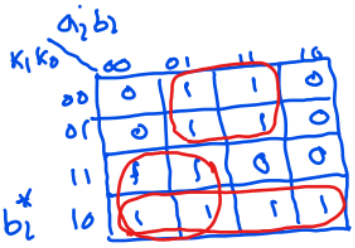
2. **(6 points)** Express a_i^* and b_i^* as functions of k_1, k_0, a_i, and b_i. Write their minimal SOPs: (you may use the provided scratch paper to draw the corresponding K-maps)

$a_i^* = \bar{k}_1 \bar{k}_0 \bar{a}_i + \bar{k}_1 k_0 a_i + k_1 \bar{k}_0 a_i + k_1 k_0 \bar{a}_i$

$b_i^* = k_1 \bar{a}_i + \bar{k}_1 b_i + k_1 \bar{k}_0$

3. **(2 points)** Define c_0 as a function of k_1, k_0 using a single gate

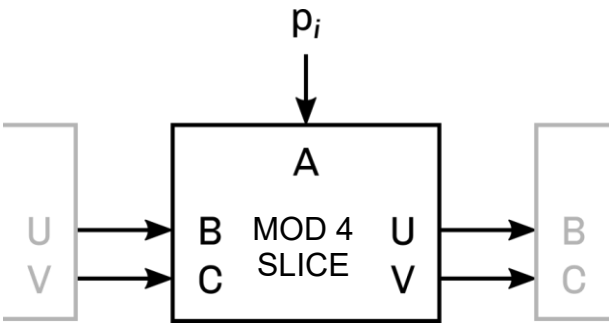
$c_0 = \bar{k}_1 \bar{k}_0 + k_1 k_0 \Rightarrow k_1 \text{ XNOR } k_0$



(2+3+3+1+1 = 10 points) Bit Slice Design - Divisible by 4

4. For this problem, you will design a bit-sliced unit to test whether an integer P given in *n*-bit unsigned representation is divisible by 4. Your divisible-by-4 unit should output a two-bit value, denoted UV, with the meaning given by the table below on the right.

Each bit slice takes a bit p_i of P = $p_{n-1}\ldots p_1 p_0$, provided as input A to the bit slice, and the two-bit output of the bit slice to the left as inputs B and C. The most significant bit p_{n-1} is the input to the leftmost bit slice and least significant bit p_0 is the input to the rightmost bit slice.



UV	Meaning
00	P is divisible by 4
01	P mod 4 = 1
10	P mod 4 = 2
11	P mod 4 = 3

Give a **minimal SOP** expression for the bit slice output variable U, a **minimal POS** expression for the bit slice output variable V, and the initial values for B and C to be provided as the first bit slice's inputs B and C. Note that U and V must be functions of A, B, and C only.

Hint: Think about the how the long division process works in the decimal system (and the base-2 system) and how we can use the P mod 4 values to capture the process. Check your bit-sliced circuit with several test values to make sure it produces the expected values of U and V at the least-significant bit position.

a. First bit slice's inputs B and C:

B_0 = _____ C_0 = _____

b. A **minimal SOP** expression for the bit slice output variable U

U = _____

c. A **minimal POS** expression for the bit slice output variable V

V = _____

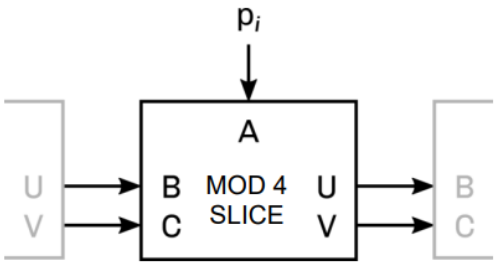
d. Use the area heuristics to compute the area of each bit-slice: _____

e. The total gate delay of the n-bit divisible by 4 circuit is: _____
(Ignore the NOT gate delay at the input)

(2+3+3+1+1 = 10 points) Bit Slice Design - Divisible by 4

4. For this problem, you will design a bit-sliced unit to test whether an integer P given in *n*-bit unsigned representation is divisible by 4. Your divisible-by-4 unit should output a two-bit value, denoted UV, with the meaning given by the table below on the right.

Each bit slice takes a bit p_i of P = $p_{n-1}\ldots p_1 p_0$, provided as input A to the bit slice, and the two-bit output of the bit slice to the left as inputs B and C. The most significant bit p_{n-1} is the input to the leftmost bit slice and least significant bit p_0 is the input to the rightmost bit slice.



UV	Meaning
00	P is divisible by 4
01	P mod 4 = 1
10	P mod 4 = 2
11	P mod 4 = 3

Give a **minimal SOP** expression for the bit slice output variable U, a **minimal POS** expression for the bit slice output variable V, and the initial values for B and C to be provided as the first bit slice's inputs B and C. Note that U and V must be functions of A, B, and C only.

Hint: Think about the how the long division process works in the decimal system (and the base-2 system) and how we can use the P mod 4 values to capture the process. Check your bit-sliced circuit with several test values to make sure it produces the expected values of U and V at the least-significant bit position.

BCA	UV
000	00
001	01
010	10
011	11
100	00
101	01
110	10
111	11

a. First bit slice's inputs B and C:

B_0 = 0 C_0 = 0

b. A **minimal SOP** expression for the bit slice output variable U

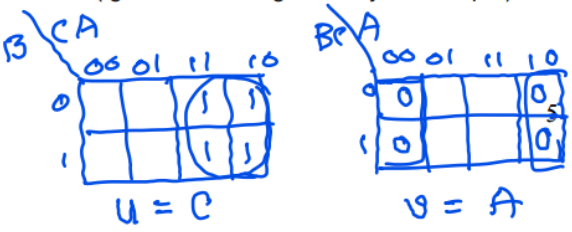
U = C

c. A **minimal POS** expression for the bit slice output variable V

V = A

d. Use the area heuristics to compute the area of each bit-slice: 0 (No gate is needed)

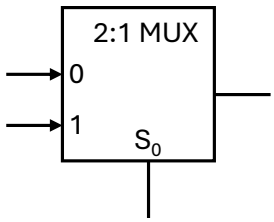
e. The total gate delay of the n-bit divisible by 4 circuit is: 0
(Ignore the NOT gate delay at the input)



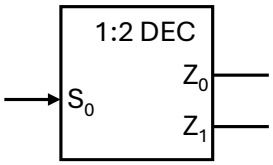
Problem 3 (21 points): Multiplexers (MUX) and Decoders (DEC)

You will implement the circuit for the Boolean function F shown in Part 3. Parts 1 and 2 will help you build the circuit.

1) (2 points) Implement an inverter using the 2:1 MUX shown below. Label all inputs of the MUX. You are free to use 0's and 1's as inputs. Note: Complemented input is not available.



2) (2 points) Now implement an inverter for input **y** using the 1:2 DEC shown below. Label the input and only one output as **OUT**. You do not need to label both outputs. Complemented input (y') is not available.
Hint: You may fill out the truth table (shown on the right) for the 1:2 DEC with 0's and 1's. The table is only there to help you, and it will **not** be graded.



Input	Output	
S_0	Z_0	Z_1
0		
1		

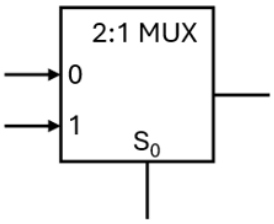
3) (5 points) Based on your answers to Parts 1 & 2, implement F (whose K-map is shown below) using only one 2:1 MUX, one 1:2 DEC, and one 4:1 MUX. The select bits for the 4:1 MUX are fixed as shown below. You **may not** use any additional gates. Complemented inputs are **not** available. You do not need to draw any loops on the K-map.

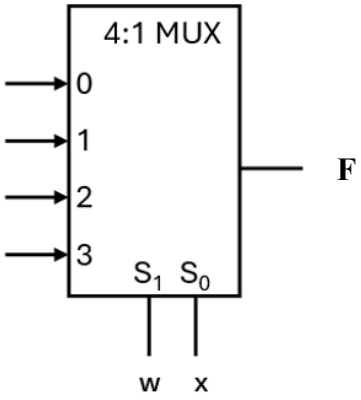
Hints: Look at the K-map, the function values at each row position and their relationship with yz. You are free to assign 0 or 1 to X (don't-care) and you are allowed to use 0's and 1's as inputs to the MUX. Necessary components (Decoder and MUXs) are provided on the next page. Complete the circuit. Note: You **may not** use any additional gates. Complemented inputs are **not** available.

yz

wx	00	01	11	10
00	X	1	1	1
01	1	X	0	0
11	1	0	0	1
10	X	0	1	X

F



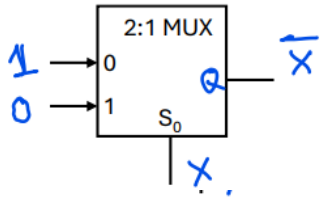


4) (8 points) Implement the function F of part 3 using a 3x8 Decoder and a four-input NOR gate. Consider that all the don't cares are 1. Draw the decoder and label its input, outputs, Enable (E), and NOR gates inputs/output clearly (use the space below).

Problem 3 (21 points): Multiplexers (MUX) and Decoders (DEC)

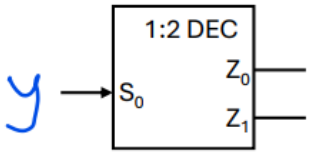
You will implement the circuit for the Boolean function F shown in Part 3. Parts 1 and 2 will help you build the circuit.

1) **(2 points)** Implement an inverter using the 2:1 MUX shown below. Label all inputs of the MUX. You are free to use 0's and 1's as inputs. Note: Complemented input is not available.



x	Q
0	1
1	0

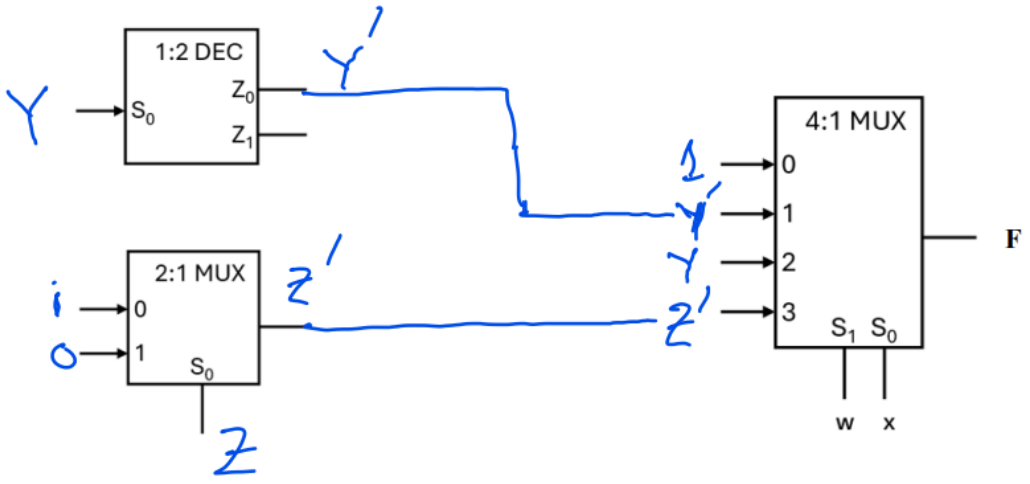
2) **(2 points)** Now implement an inverter for input **y** using the 1:2 DEC shown below. Label the input and only one output as **OUT**. You do not need to label both outputs. Complemented input (y') is not available.
Hint: You may fill out the truth table (shown on the right) for the 1:2 DEC with 0's and 1's. The table is only there to help you, and it will **not** be graded.



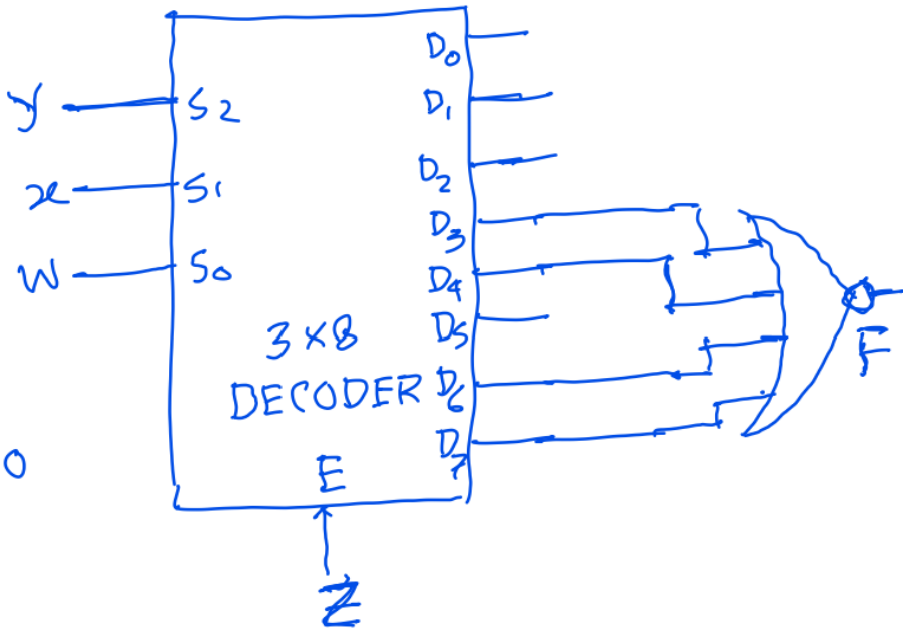
OUT

Input S_0	Output Z_0	Z_1
0	1	0
1	0	1

UIN----------



4) (8 points) Implement the function F of part 3 using a 3x8 Decoder and a four-input NOR gate. Consider that all the don't cares are 1. Draw the decoder and label its input, outputs, Enable (E), and NOR gates inputs/output clearly (use the space below).



when $z=0$

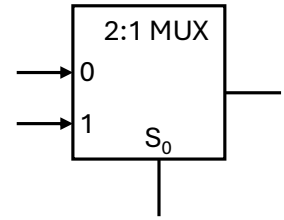
$D_0 \cdots D_7 = 0$

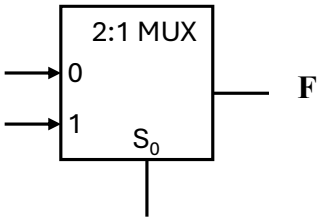
when $z=1$

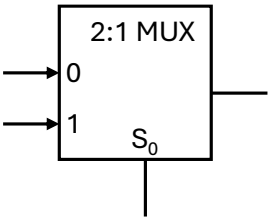
	w x y	F
3	0 1 1	0
4	1 0 0	0
6	1 1 0	0
7	1 1 1	0

otherwise $F=1$

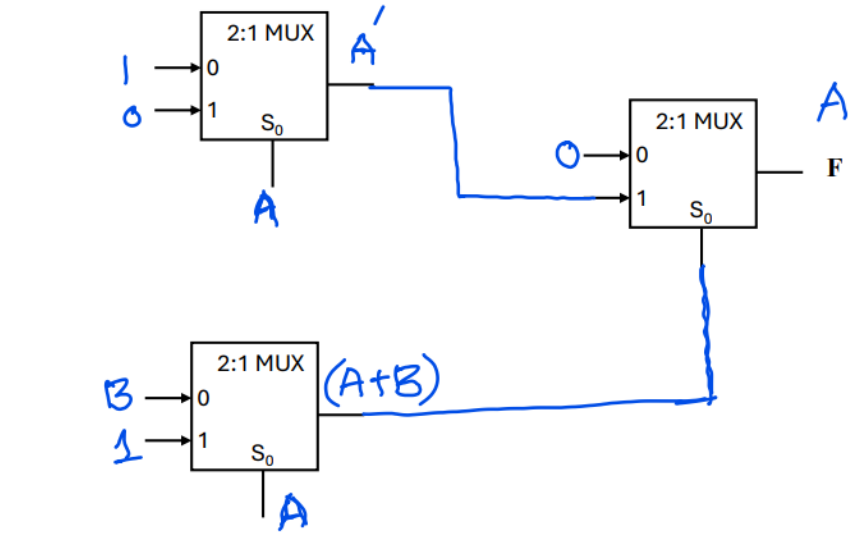
5) (4 points) Implement the following Boolean function **F = A'(A+B)** using three **2-to-1 MUX (given below). Note:** You are free to use 0's and 1's as inputs. **DO NOT simplify the expression. You may not use any other logic gates.** Assume the Complemented input A' is **not** available**. Hints:** how do you implement OR and AND gates using MUX?







5) **(4 points)** Implement the following Boolean function **F = A'(A+B)** using three **2-to-1 MUX (given below). Note:** You are free to use 0's and 1's as inputs. **DO NOT simplify the expression. You may not use any other logic gates.** Assume the Complemented input A' is **not** available**. Hints:** how do you implement OR and AND gates using MUX?



Alternative Output Mux Label

$S_0 \leftarrow A'$

$0 \leftarrow 0$

$1 \leftarrow (A+B)$

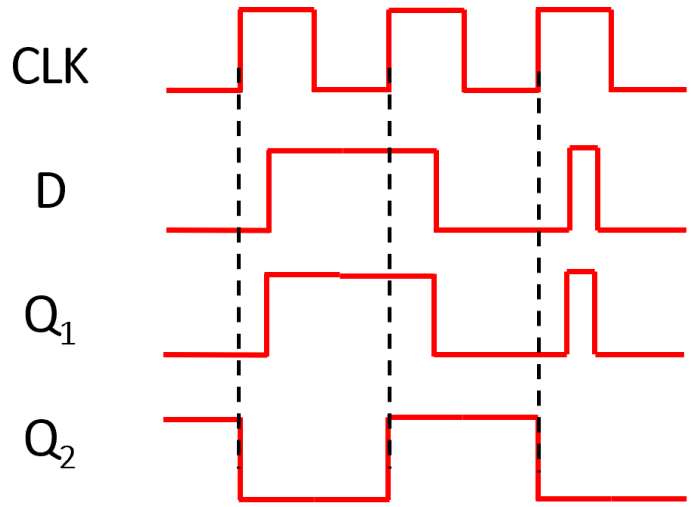
[redacted]

Problem 4 (13 points): Latch, Flip-flops, and Registers

1. **(2 points)** One of your colleagues designed two sequential circuit elements but forgot which one is a Latch and which one is a flip flop. Based on the timing diagram, please tell him which is a flip-flop and which is a latch?

Q1 _____

Q2 _____

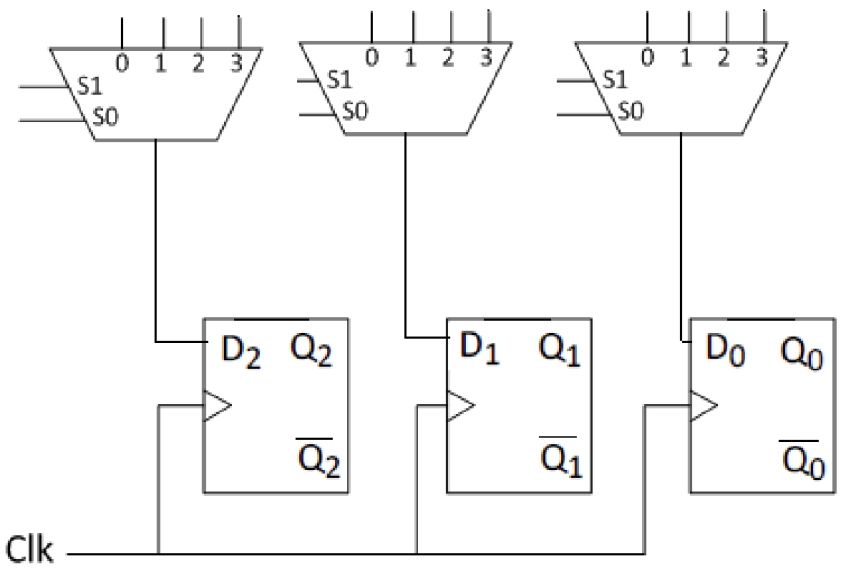


2. **(11 points) Registers**

Consider a 3-bit shift register that has the functionality specified in the table to the right.

1. (6 points) In the figure below, label the inputs to the muxes to complete the design of this 3-bit register that performs the operations listed in the table. **Use labels and do not draw any additional gates or wires.**

A	B	Operation
0	0	Hold value
0	1	Bitwise complement
1	0	Logical shift right
1	1	Arithmetic shift right



2. (2 points) If the shift register initially stores $Q_2Q_1Q_0$=101, what is stored in the register after one clock cycle when AB=11?

   Answer: $Q_2Q_1Q_0$ = _____

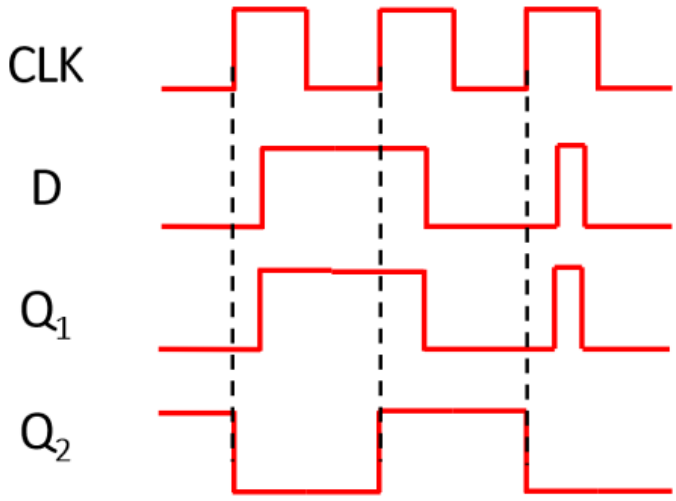
3. (3 points) If the shift register initially stores $Q_2Q_1Q_0$=100, and in the next clock cycle, if we want the register to hold the values $Q_2Q_1Q_0$=010, what should be A and B and the operation to yield this result?

   A = _____, B = _____, Operation = _____

1. **(2 points)** One of your colleagues designed two sequential circuit elements but forgot which one is a Latch and which one is a flip flop. Based on the timing diagram, please tell him which is a flip-flop and which is a latch?

Q1 Latch

Q2 D-FF

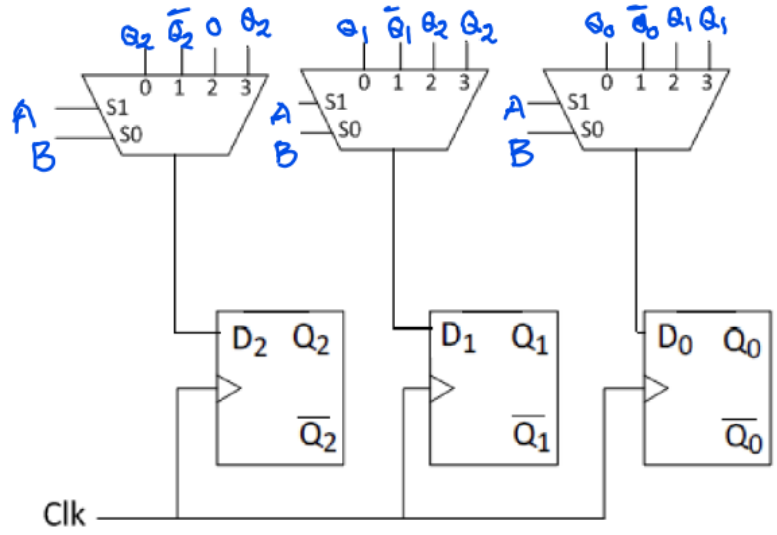


2. (11 points) Registers

Consider a 3-bit shift register that has the functionality specified in the table to the right.

A	B	Operation
0	0	Hold value
0	1	Bitwise complement
1	0	Logical shift right
1	1	Arithmetic shift right

1. (6 points) In the figure below, label the inputs to the muxes to complete the design of this 3-bit register that performs the operations listed in the table. **Use labels and do not draw any additional gates or wires.**



2. (2 points) If the shift register initially stores $Q_2Q_1Q_0$=101, what is stored in the register after one clock cycle when AB=11?

   Answer: $Q_2Q_1Q_0$ = 110

3. (3 points) If the shift register initially stores $Q_2Q_1Q_0$=100, and in the next clock cycle, if we want the register to hold the values $Q_2Q_1Q_0$=010, what should be A and B and the operation to yield this result?

   A = 1, B = 0, Operation = logical shift Right

Problem 5 (21 points): Design with Components and Abstraction

In this problem, you will design binary multipliers that multiply unsigned binary numbers. You will start by examining the multiplication of 2-bit numbers, extend the concepts to 4-bit numbers, and finally generalize the design for multiplying any M-bit and N-bit binary numbers.

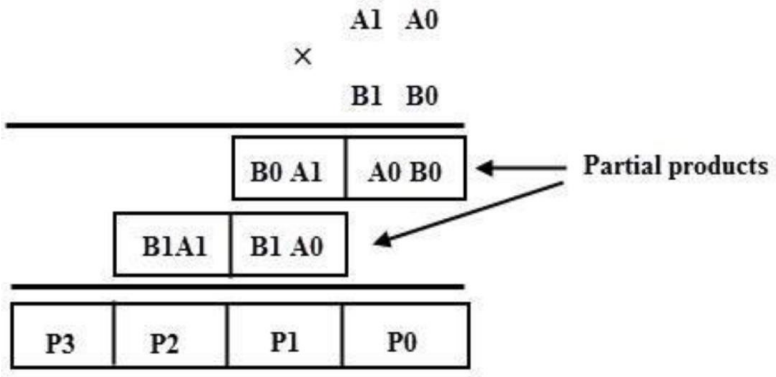
Background: Binary Multiplication

Binary multiplication operates under the same principles as decimal multiplication but uses only two digits: 0 and 1. The fundamental rules for binary multiplication are:

- 0×0=0
- 0×1=0
- 1×0=0
- 1×1=1

When multiplying two binary numbers, each bit of the multiplier is multiplied by each bit of the multiplicand, generating partial products. These partial products are then appropriately shifted (based on their positional value) and summed to produce the final product.

Part 1: (2+10 = 12 points) Understanding 2-bit Binary Multiplication



1.1 Let $A=A_1A_0$ and $B=B_1B_0$ be two 2-bit unsigned binary numbers, where A_1 and B_1 are the most significant bits (MSBs), and A_0 and B_0 are the least significant bits (LSBs). Using AND, and ADD operators, write the expressions for each bit P_0, P_1, P_2, P_3 of the product $P=P_3P_2P_1P_0$ resulting from the multiplication of A and B. Note: Carry from P_1 will be denoted as $C_{in, P1}$, Carry from P_2 will be denoted as $C_{in, P2}$

P_0 =_____

P_1 =_____

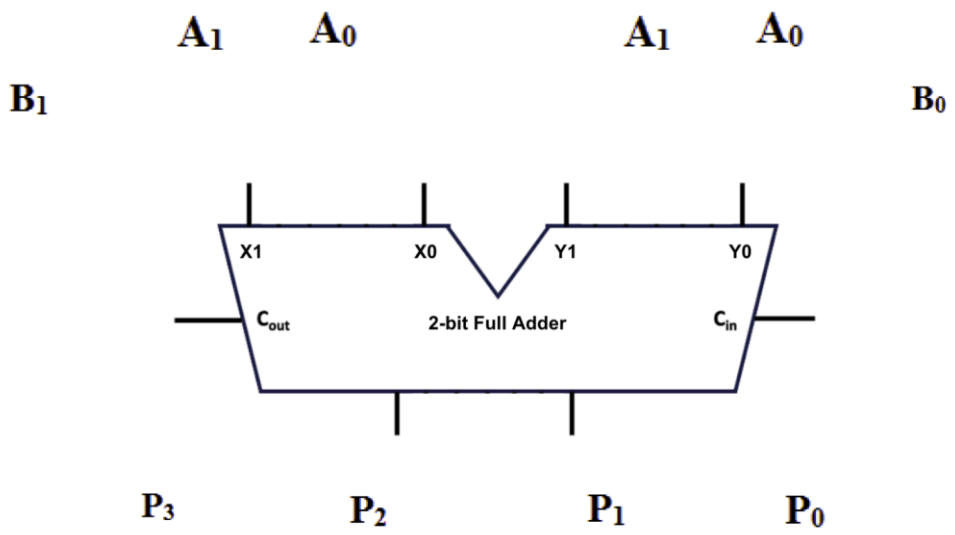
P_2 =_____

P_3 =_____

1.2 Complete the following logic schematic diagram of a 2-bit by 2-bit binary multiplier using:

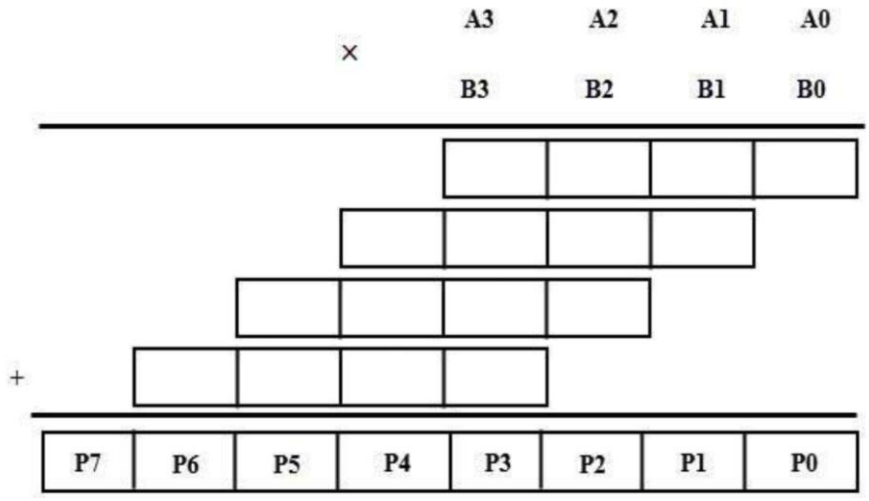
- AND gates to generate the partial products.
- 2-bit Full Adder to sum the partial products.

Ensure that all inputs, outputs, and intermediate signals are clearly labeled.



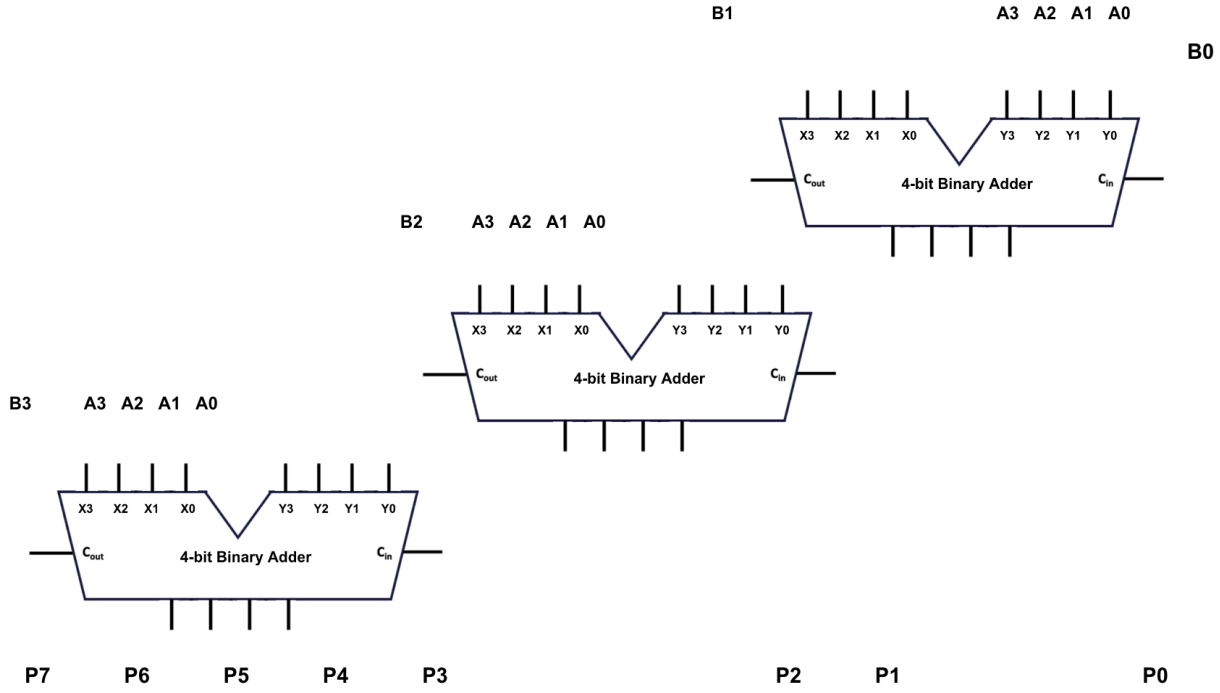
Part 2 (1+6=7 points): Extending to 4-bit Binary Multiplication

(1 point) Now consider multiplying two 4-bit unsigned binary numbers: begin by analyzing the multiplication process and completing the following diagram for 4-bit by 4-bit binary multiplication (fill in all the partial products generated 4-bit by 4-bit binary multiplication (fill in all the partial products generated when multiplying (4-bit unsigned) A by B)



(6 points) Designing the Logic Schematic for a 4-bit Binary Multiplication

2.1. Using the concepts from Parts 1 and 2, design a logic schematic diagram for a 4-bit by 4-bit binary multiplier. Your design should use AND gates to generate the partial products and 4-bit binary adders to sum them. (Inputs including: $A_0\,A_1\,A_2\,A_3\,B_0\,B_1\,B_2\,B_3$; outputs including: $P_0\,P_1\,P_2\,P_3\,P_4\,P_5\,P_6\,P_7$)



Part 3: (2 pts) Generalizing to M-bit by N-bit Binary Multiplication

Consider multiplying an M-bit unsigned binary number (multiplicand) by an N-bit unsigned binary number (multiplier).

3.1 Determine the number of AND gates required to generate all the partial products for an M-bit by N-bit binary multiplication.

Answer:_____

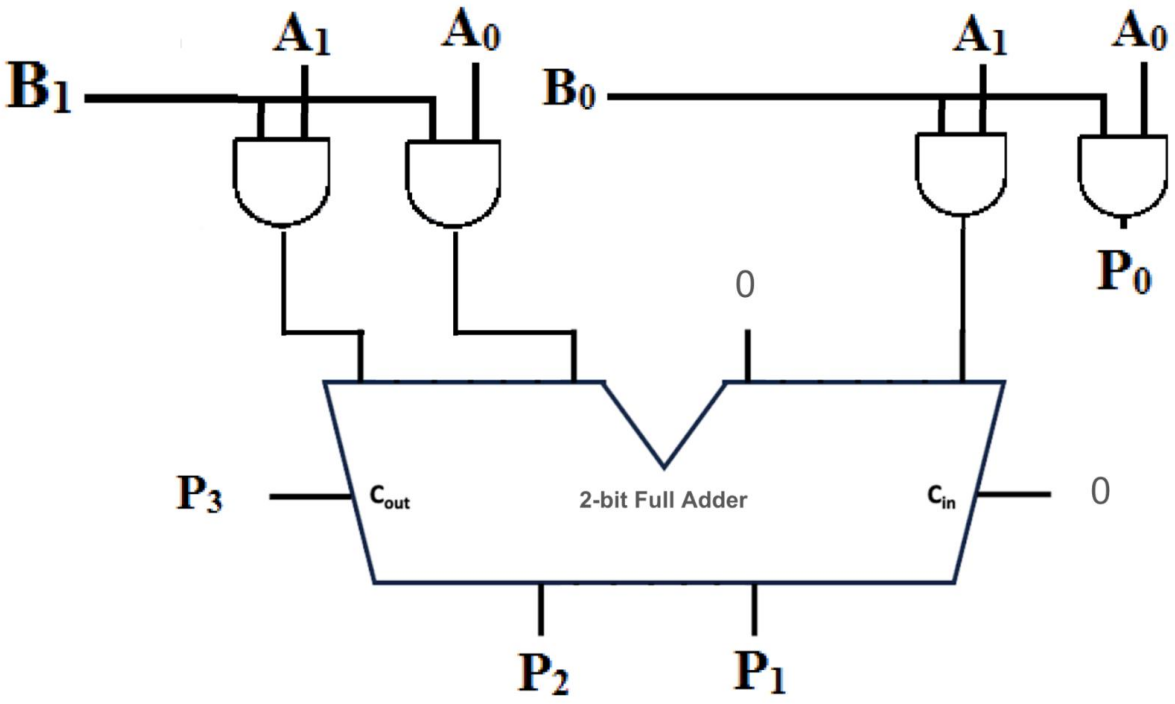
3.2 After generating the partial products, calculate the number of N-bit binary adders needed to obtain the final output?

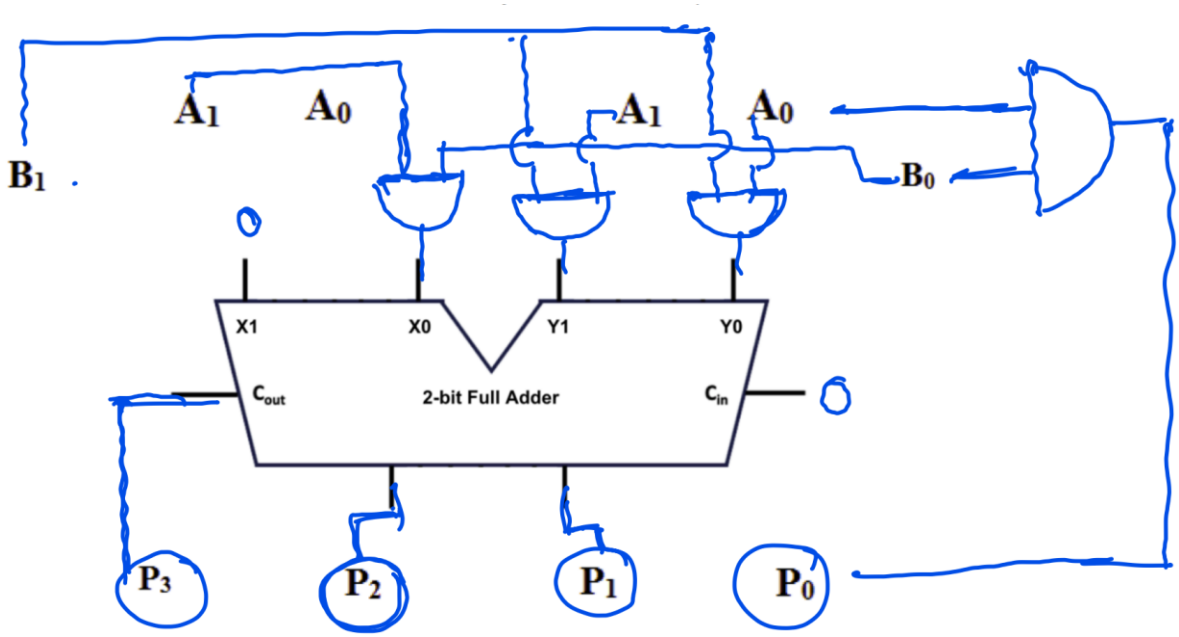
Answer:_____

5.1

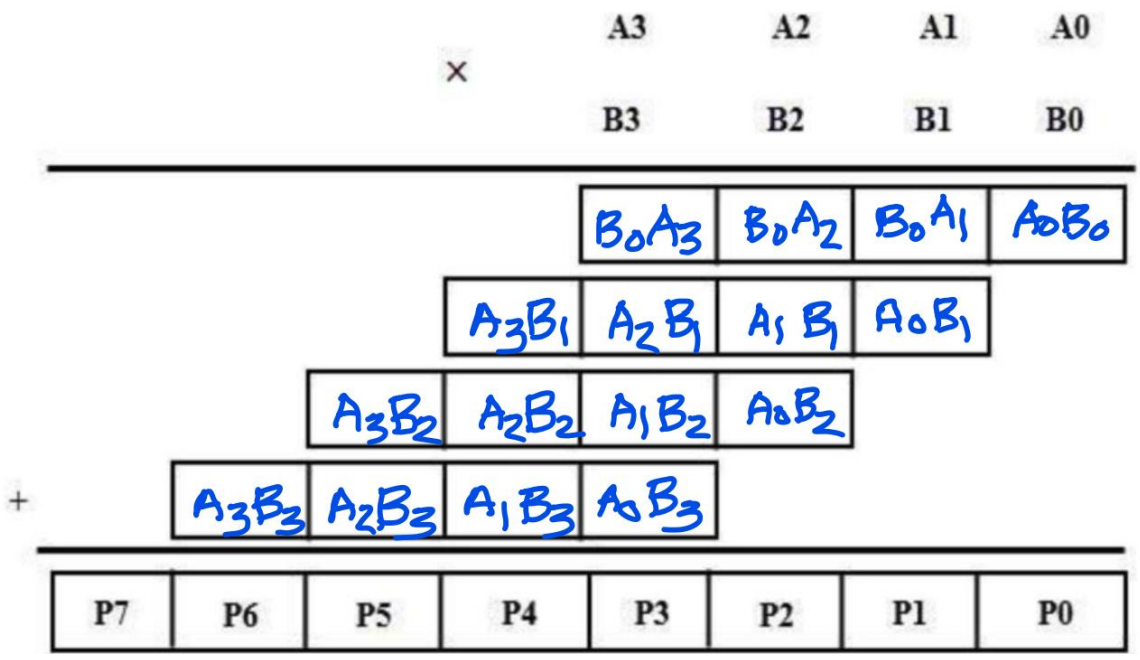
- $P_0 = A_0 B_0$
- $P_1 = (A_1 B_0) + (A_0 B_1)$
- $P_2 = (A_1 B_1) + C_{\text{in},P_1}$
- $P_3 = C_{\text{in},P_2}$

5.2

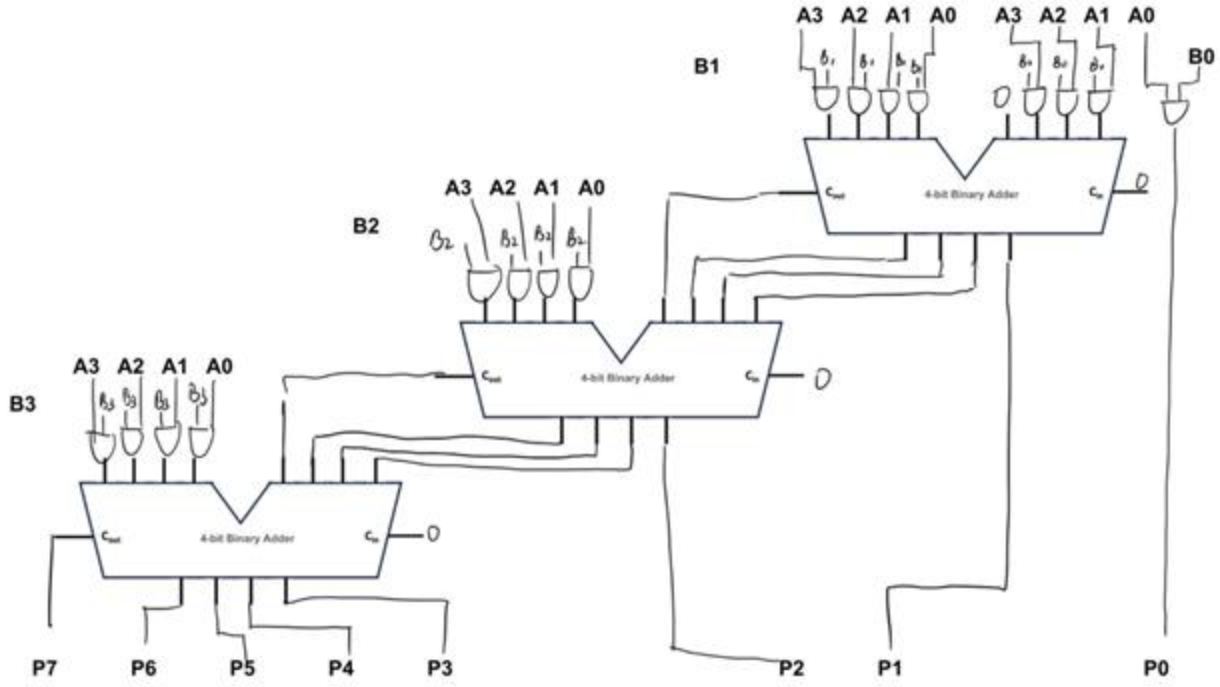



A1
A0
A1
A0
B1
B0
0
X1
X0
Y1
Y0
Cout
2-bit Full Adder
Cin
0
P3
P2
P1
P0

5.2.1


A3
A2
A1
A0
×
B3
B2
B1
B0
B0A3
B0A2
B0A1
A0B0
A3B1
A2B1
A1B1
A0B1
A3B2
A2B2
A1B2
A0B2
+
A3B3
A2B3
A1B3
A0B3
P7
P6
P5
P4
P3
P2
P1
P0

Part -2: 5.2 (Circuit)



5.3.1

Need N*M AND Gates

5.3.2

Expression dependent on M, not N

Boolean algebra properties

Commutativity	x·y = y·x	x + y = y + x
Associativity	(x·y)·z = x·(y·z)	(x + y) + z = x + (y + z)
Distributivity	x·(y + z) = x·y + x·z	x + y·z = (x + y)·(x + z)
Idempotence	x·x = x	x + x = x
Identity	x·1 = x	x + 0 = x
Null	x·0 = 0	x + 1 = 1
Complementarity	x·x' = 0	x + x' = 1
Involution	(x')' = x	
DeMorgan's	(x·y)' = x' + y'	(x + y)' = x'·y'
Absorption	x·(x + y) = x	x + x·y = x
No-Name	x·(x' + y) = x·y	x + x'·y = x + y
Consensus	(x+y)·(y+z)·(x'+z) = (x+y)·(x'+z)	x·y + y·z + x'·z = x·y + x'·z

Table of ASCII Characters

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
(nul)	0	00	(sp)	32	20	@	64	40	`	96	60
(soh)	1	01	!	33	21	A	65	41	a	97	61
(stx)	2	02	"	34	22	B	66	42	b	98	62
(etx)	3	03	#	35	23	C	67	43	c	99	63
(eot)	4	04	$	36	24	D	68	44	d	100	64
(enq)	5	05	%	37	25	E	69	45	e	101	65
(ack)	6	06	&	38	26	F	70	46	f	102	66
(bel)	7	07	'	39	27	G	71	47	g	103	67
(bs)	8	08	(	40	28	H	72	48	h	104	68
(ht)	9	09	)	41	29	I	73	49	i	105	69
(lf)	10	0a	*	42	2a	J	74	4a	j	106	6a
(vt)	11	0b	+	43	2b	K	75	4b	k	107	6b
(ff)	12	0c	,	44	2c	L	76	4c	l	108	6c
(cr)	13	0d	-	45	2d	M	77	4d	m	109	6d
(so)	14	0e	.	46	2e	N	78	4e	n	110	6e
(si)	15	0f	/	47	2f	O	79	4f	o	111	6f
(dle)	16	10	0	48	30	P	80	50	p	112	70
(dc1)	17	11	1	49	31	Q	81	51	q	113	71
(dc2)	18	12	2	50	32	R	82	52	r	114	72
(dc3)	19	13	3	51	33	S	83	53	s	115	73
(dc4)	20	14	4	52	34	T	84	54	t	116	74
(nak)	21	15	5	53	35	U	85	55	u	117	75
(syn)	22	16	6	54	36	V	86	56	v	118	76
(etb)	23	17	7	55	37	W	87	57	w	119	77
(can)	24	18	8	56	38	X	88	58	x	120	78
(em)	25	19	9	57	39	Y	89	59	y	121	79
(sub)	26	1a	:	58	3a	Z	90	5a	z	122	7a
(esc)	27	1b	;	59	3b	[	91	5b	{	123	7b
(fs)	28	1c	<	60	3c	\	92	5c	|	124	7c
(gs)	29	1d	=	61	3d	]	93	5d	}	125	7d
(rs)	30	1e	>	62	3e	^	94	5e	~	126	7e
(us)	31	1f	?	63	3f	_	95	5f	(del)	127	7f

Scratch Paper