Midsem

(Monsoon 2021-22)

Q1. In ancient times, people used parrots to make predictions. Let us consider a parrot that we are using predict the result of a badminton match between player 1 and player 2. Once we know the oucome of the match, we make a logic function (F) that outputs “1” when the parrot was right and “0” when the parrot was wrong. Make a truth-table for this logic function. Assuming all the possible outcomes in the truth-table are equally likely, what is the probability that our parrot predicted correctly ? Now, let us say we are unhappy with the probability of correct prediction, and want to increase our chances of success. For that, we now employ three parrots and create a function (G) that outputs “1” when one or more of them was right, and “0” only when all of them were wrong simultaneously. Make a truth-table, a K-map and the simplest logic circuit for this function.

A1.

- We go from logic problem -> function I/O -> K-map -> expression -> circuit
- Output of the function is given as F/G. What are the inputs ?
- Match the result and parrot’s prediction
- The truth-table will be:

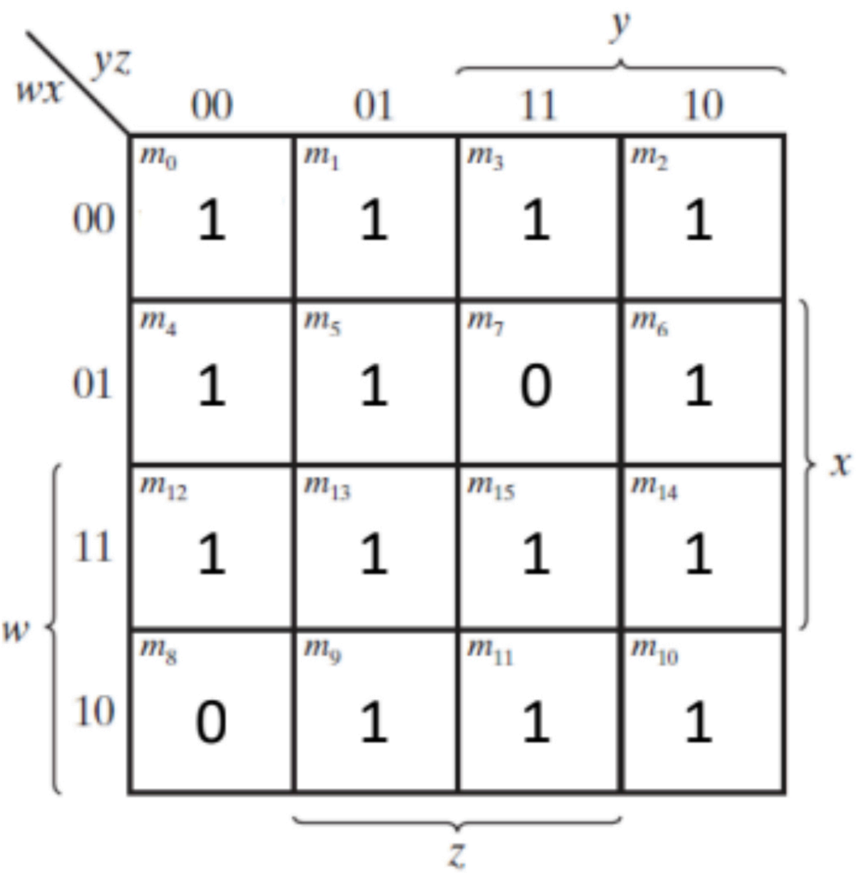
R	P	F
P1	P1	1
P1	P2	0
P2	P1	0
P2	P2	1

- We can take P1 as 0 and P2 as 1 to make it resemble a normal truth-table
- For three parrots:

R	P1	P2	P3	G
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	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1

R	P1	P2	P3	G
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

- "Correctness" of the parrot is a derived variable from the result of the match and the prediction of the parrot
- The K-map for the function G



- No clusters of 16 or 8
- Clusters of 4 ?
- Total 12: 4 vertical/horizontal, 8 squares
- No essential prime implicants
- The function can be written as G = wz + xy + w'x' + yz'
- The function can also be written as G = (w + x' + y' + z')(w' + x + y + z)

Q2. Let us define a set B = {1,p,q,pq}, where p and q are distinct prime natural numbers. Let us define two binary operators on the elements of this set as Least Common Multiple denoted by (a # b), and Greatest Common Factor denoted by (a * b). Are the operations # and * closed on the set B ? Do p and q need to be prime for the closure to work ? Do the operations # and * have an identity element, if so, what are they ?

A2.

- We can make a table with all the inputs and see what the outputs are
- P and q should be co-prime for the closure to work
- Identity in this case is 1 and pq for LCM and GCD
- In general case, it is important to define which set you are working on - because operators are defined on sets
- Identity for LCM = 1. Identity for GCD on Integer set is 0; on natural numbers is infinity/not defined

Q3. We can design the 2's complement circuit using adder circuits. But is that the most optimum way ? Design an optimal logic circuit for obtaining the 2's complement of a 4-bit binary number (ABCD). Count the number of transistors used and compare the number with the adder implementation.

A3.

- Yes a more optimum circuit exists
- Use K-maps to make 2-level circuit
- The number of transistors can be calculated using the following:
    1. For an AND Gate no of transistors = 2n where n is the number of inputs
    2. For a NAND Gate no of transistors = 2n+2 where n is the number of inputs

Q4. Do the following conversions

1. $(93.25)_{10} = ( )_8$
2. $(1011)_{10} = ( )_{16}$
3. $(CAD.004)_{16} = ( )_8$
4. $(123)_4 = ( )_{16}$
4. $(3.3)_{10} = ( )_2$

A4.

1.

8	93	5
8	11	3
8	1	1
	0	

Taking the integral part $(93)_{10} = (135)_8$

Taking the decimal part

0.25	2.0	2
0	-	-

$(0.25)_{10} = (0.2)_8$

$\therefore (93.25)_{10} = (135.2)_8$

2.

16	1011	3
16	63	15 (F)
16	3	3
	0	

$\therefore (1011)_{10} = (3F3)_{16}$

3. Path => 16 -> 2 -> 8

	C	A	D	.	0	0	4	
	1100	1010	1101	.	0000	0000	0100	
110	010	101	101	.	000	000	000	100
6	2	5	5	.	0	0	0	4

∴ $(CAD.004)_{16} = (6255.0004)_8$

4. Path => 4 -> 2 -> 16

1	2	3
01	10	11

0001	1011
1	B

∴ $(123)_4 = (1B)_{16}$

5. Taking integral part

2	3	1
2	1	1
2	0	

Taking integral part $(3)_{10} = (11)_2$

Taking decimal part

0.3	0.6	0
0,6	1.2	1
0.2	0.4	0
0.4	0.8	0
0.8	1.6	1
0.6	1.2	1
0.2		

Repeatition

$(0.3)_{10} = (0.\overline{1001})_2$

∴ $(3.3)_{10} = (11.0\overline{1001})_2$