Homework 8

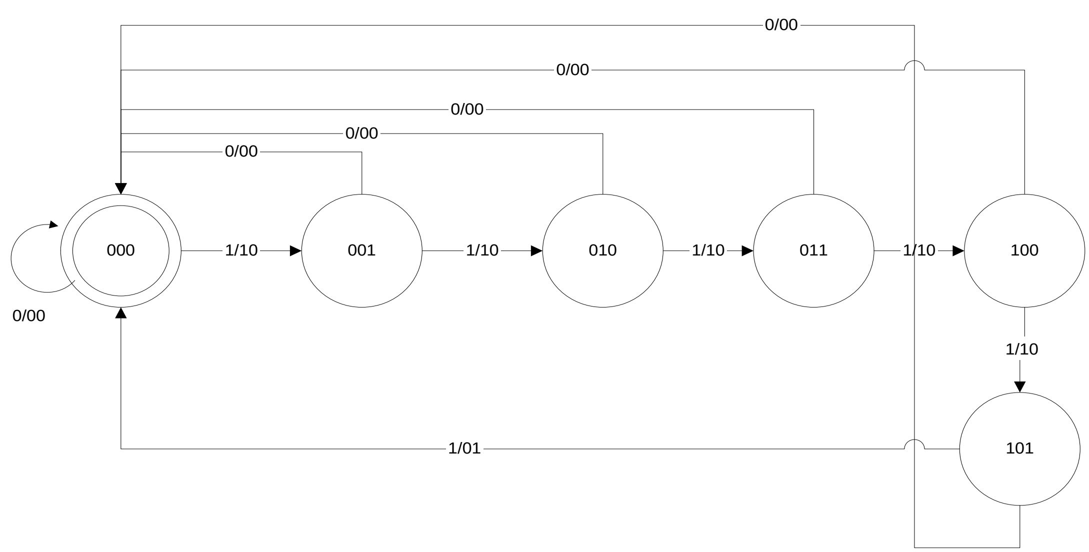
The sequence 01111110 is a flag used in a message communications network that represents the beginning of a message. This flag must be unique. As a consequence, at most five 1's in sequence may appear anywhere else in the message. Since this is unrealistic for normal message content, a trick called zero-insertion is used. The normal message, which can contain strings of 1's longer than 5, enters input X of a sequential zero-insertion circuit given below:

The circuit has two outputs Z and S. When a sixth 1 in sequence appears on X, a 0 is inserted into the stream of outputs appearing on Z and the output S = 1 indicating that a zero-insertion has happened. Zero-insertion is illustrated by the following example sequences:

Message sequence on X: 01111110011111111100001011110101
Output sequence on Z: 01111100011111011100001011110101
Zero-insertion indicator on S: 00000010000000100000000000000000

Task: Design the zero-insertion circuit above following the 9-step design procedure given in Lecture 10 using SR Flip-Flops and logic gates.

We have to insert 1 zero every time we recognize 5 ones in the sequence and another 1 is waiting at input. We can make a state machine of this situation but it is clear that we need 6 states to remember the zero to five ones, which means we need 3 bits to encode this. This also means we need 3 flip flops.



Here we have the table for the state machine (S is only 1 when Z is forced to be zero)

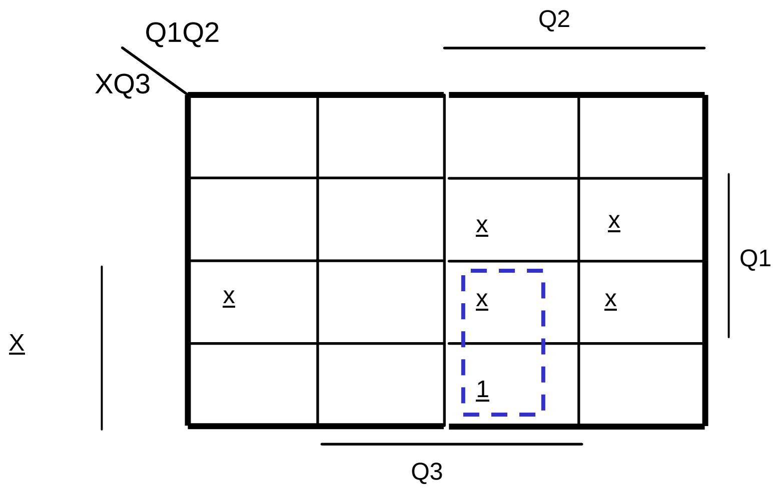
X	Q1(t)	Q2(t)	Q3(t)	Q1(t+1)	Q2(t+1)	Q3(t+1)	Z	S
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
0	1	1	0	x	x	x	x	x
0	1	1	1	x	x	x	x	x
1	0	0	0	0	0	1	1	0
1	0	0	1	0	1	0	1	0
1	0	1	0	0	1	1	1	0
1	0	1	1	1	0	0	1	0
1	1	0	0	1	0	1	1	0
1	1	0	1	0	0	0	0	1
1	1	1	0	x	x	x	x	x
1	1	1	1	x	x	x	x	x

Because Q1, Q2 and Q3 are SR flip-flops (which means 2 inputs) we need to model their behaviour with another table showing their input values.

X	Q1(t)	Q2(t)	Q3(t)	Q1(t+1)	Q2(t+1)	Q3(t+1)	S1	R1	S2	R2	S3	R3	Z	S
0	0	0	0	0	0	0	0	x	0	x	0	x	0	0
0	0	0	1	0	0	0	0	x	0	x	0	1	0	0
0	0	1	0	0	0	0	0	x	0	1	0	x	0	0
0	0	1	1	0	0	0	0	x	0	1	0	1	0	0
0	1	0	0	0	0	0	0	1	0	x	0	x	0	0
0	1	0	1	0	0	0	0	1	0	x	0	1	0	0
0	1	1	0	x	x	x	x	x	x	x	x	x	x	x
0	1	1	1	x	x	x	x	x	x	x	x	x	x	x
1	0	0	0	0	0	1	0	x	0	x	1	0	1	0
1	0	0	1	0	1	0	0	x	1	0	0	1	1	0
1	0	1	0	0	1	1	0	x	x	0	1	0	1	0
1	0	1	1	1	0	0	1	0	0	1	0	1	1	0
1	1	0	0	1	0	1	x	0	0	x	1	0	1	0
1	1	0	1	0	0	0	0	1	0	x	0	1	0	1
1	1	1	0	x	x	x	x	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x	x	x	x	x

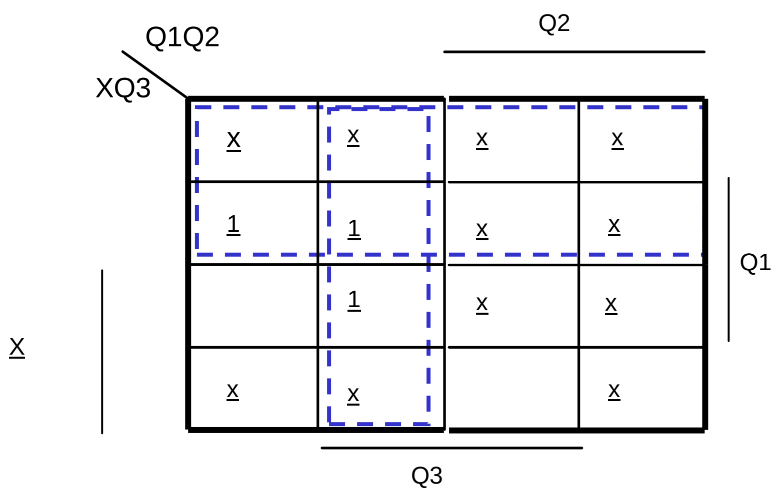
Now that we have all the needed tables, we can construct our Boolean equations and simplify them with k-maps.

S1)



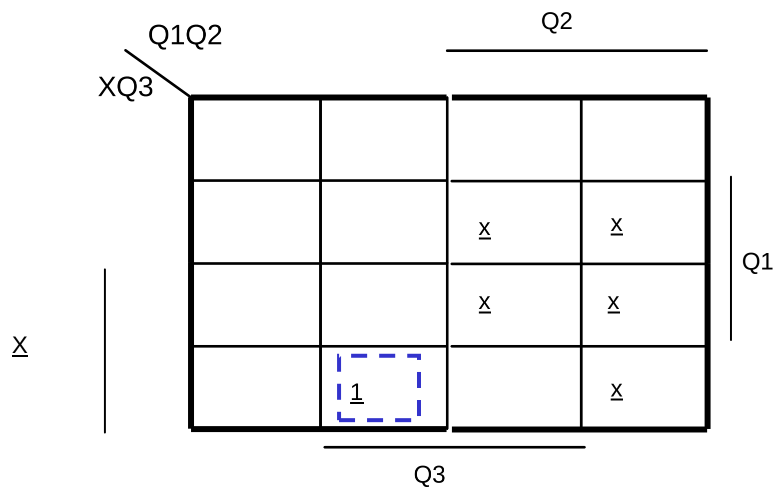
$S1 = XQ_2Q_3$

R1)



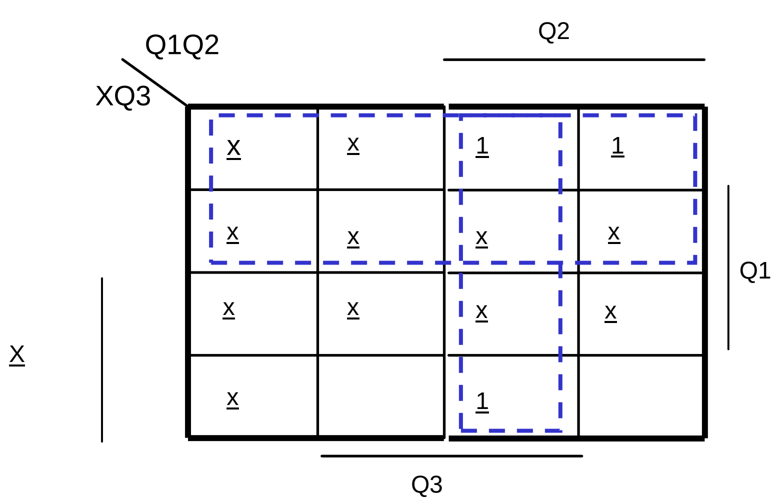
$R1 = X' + Q_2'Q_3$

S2)



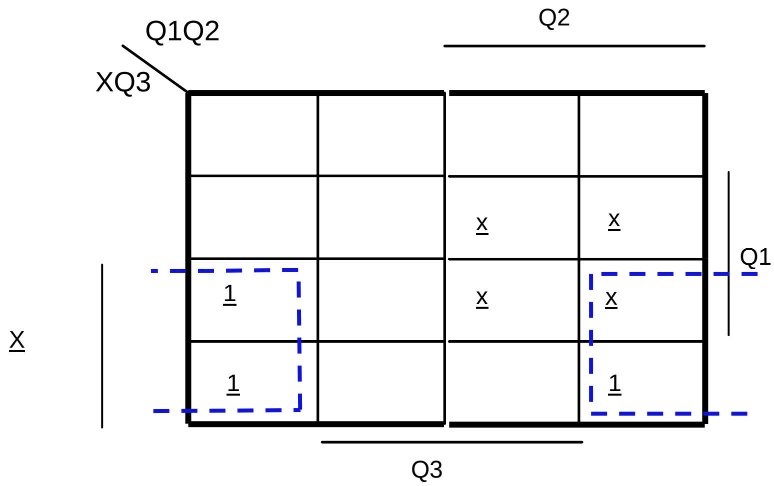
S2 = $XQ_1'Q_2'Q3$

R2)



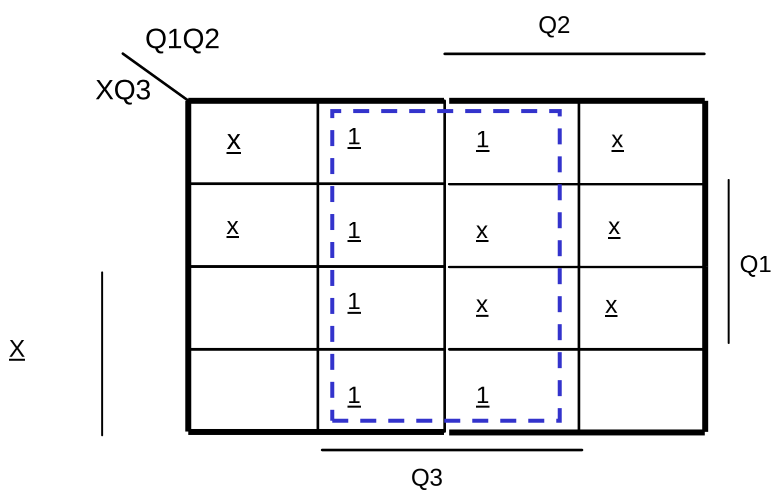
R2 = $X' + Q_2Q_3$

S3)



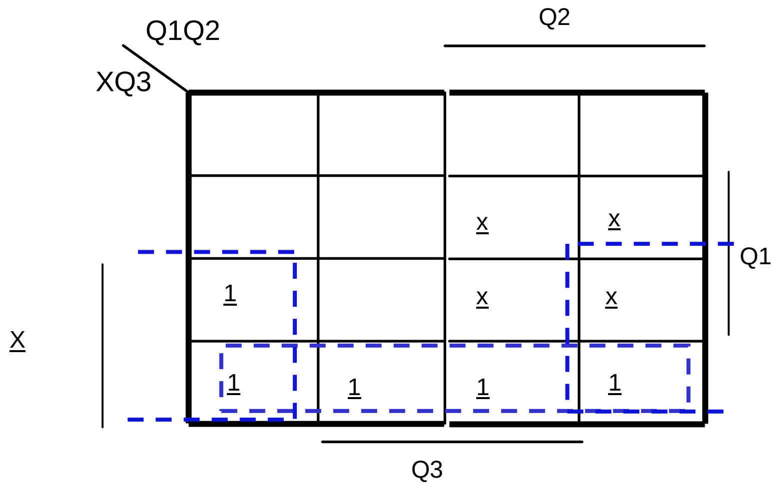
S3 = XQ_3'

R3)



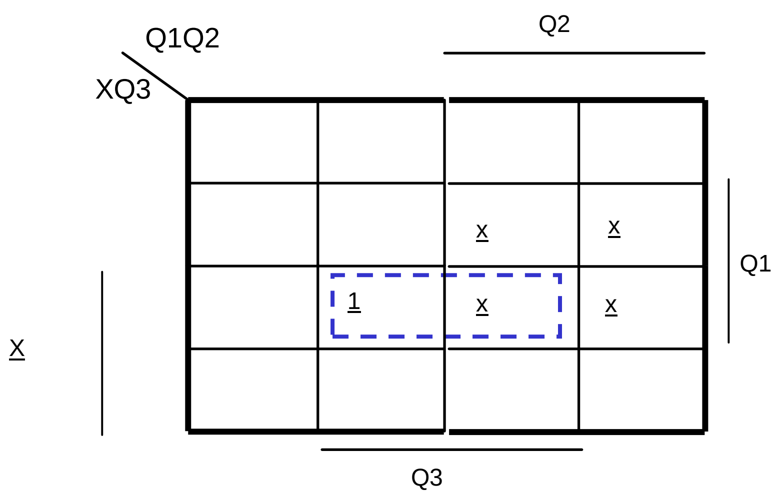
R3 = Q_3

Z)



$Z = XQ_1' + XQ_3'$

S)



$S = XQ_1Q_3$

The schematic of our functions