Midterm Test

(Time allowed: 90 minutes)
November 7, 2021

NOTE: Answer ALL **9** questions. Show all intermediate steps, except Question 1.

1. (10 points) Are the following statements necessarily true (Y) or not (N)?

(1) The dual symbol for a NAND gate is a negative-AND symbol.

(2) Any Boolean function can be implemented using only NOR gates.

(3) A demultiplexer is a logic circuit that allows digital information from a single source to be routed onto several lines.

(4) Addition in Boolean algebra is equivalent to the NOR function.

(5) A truth table illustrates how the input level of a gate responds to all the possible output level combinations.

(6) A NOR gate output is LOW if any of its inputs is LOW.

(7) Good troubleshooting is done by looking at the input signal and how it interacts with the circuits.

(8) A NAND gate output is LOW only if all the inputs are HIGH.

(9) An exclusive-NOR gate output is HIGH when the inputs are unequal.

(10) A full-adder has three inputs and two outputs.

Solution:

(1) N
(2) Y
(3) Y
(4) N
(5) N
(6) N
(7) N
(8) Y
(9) N
(10) Y

2. (15 points)

(1) Convert the binary number 110011.11 into **Octal**;

(2) Perform the subtraction of the **signed** numbers :

$$00110011 - 01101010, \tag{1}$$

i.e. subtract 01101010 from 00110011. Express your result in the 2's complement form.

(3) Divide 01111010 by 00101000 using the 2's complement addition. Find the quotient and remainder in the binary form. Please show each calculation step.

Solution:

(1) $(110011.11)_2 = (63.6)_8$;

(2) 11001001

$$\begin{array}{r} 00110011 \\ +\ 10010110 \\ \hline 11001001 \end{array}$$

(3) The quotient is $(0000011)_2 = 3$ whereas remainder is $(00000010)_2 = 2$.

Step 1:

$$\begin{array}{r} 01111010 \\ +\ 11011000 \\ \hline 01010010 \end{array}$$

Quotient: 00000001

Step:2:

$$\begin{array}{r} 01010010 \\ +\ 11011000 \\ \hline 00101010 \end{array}$$

Quotient: 00000010

Step:3:

$$\begin{array}{r} 00101010 \\ +\ 11011000 \\ \hline 00000010 \end{array}$$

Quotient: 00000011

Remainder: 00000010

3. (10 points)

(1) Multiply 01101010 by 11110001 in the 2's complement form. Note the resulting product may have more than eight bits.

(2) Apply CRC to the data bits 10110010 using the generator code 1010 to produce the transmitted CRC code.

Solution:

(1) The 2's complement form of 11110001 is 00001111. Thus, we have

$$
\begin{array}{r}
01101010 \\
\times\ 00001111 \\
\hline
01101010 \\
01101010 \\
\hline
100111110 \\
01101010 \\
\hline
1011100110 \\
01101010 \\
\hline
1100011010
\end{array}
$$

Finally, changing to 2's complement with sign, we have 100111001010.

(2) The remainder can be found as follows:

$$
\begin{array}{r}
101100100000 \\
1010 \\
\hline
1001 \\
1010 \\
\hline
1100 \\
1010 \\
\hline
1100 \\
1010 \\
\hline
1100 \\
1010 \\
\hline
1100 \\
1010 \\
\hline
\text{Remainder} = 0110
\end{array}
$$

CRC is 101100100110.

4. (10 points)

(1) Design a one-bit basic comparator that implements the following truth table

Inputs		Output
A	B	$A = B$
0	0	1
0	1	0
1	0	0
1	1	1

Table 1: one-bit comparator

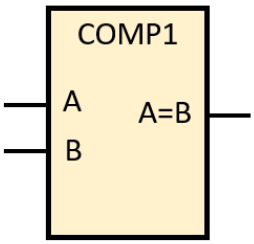


Figure 1: One-bit basic comparator

(2) Design a one-bit *inequality* comparator with the following truth table by using the one-bit basic comparator constructed in (1) and any necessary basic logic gates:

Inputs		Output		
A	B	$A < B$	$A = B$	$A > B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Table 2: one-bit inequality comparator

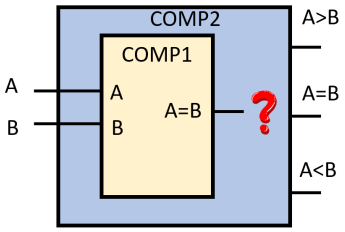


Figure 2: One-bit inequality comparator

(3) Implement a two-bit inequality comparator using two one-bit inequality comparators designed in (2) and any necessary basic logic gates, assuming that A_1 and B_1 are the most significant bits.

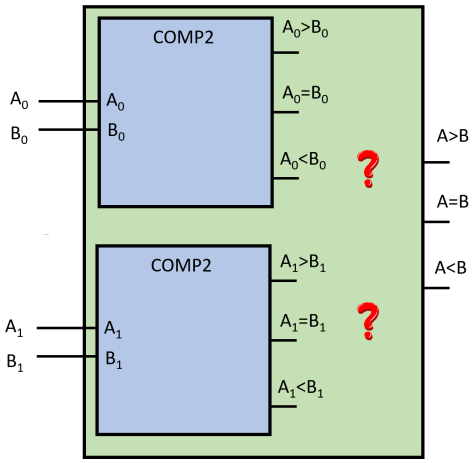


Figure 3: The two-bit inequality comparator

Solution:

(1) One-bit basic comparator

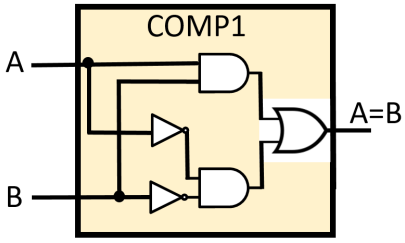


Figure 4: One-bit basic comparator

(2) One-bit inequality comparator

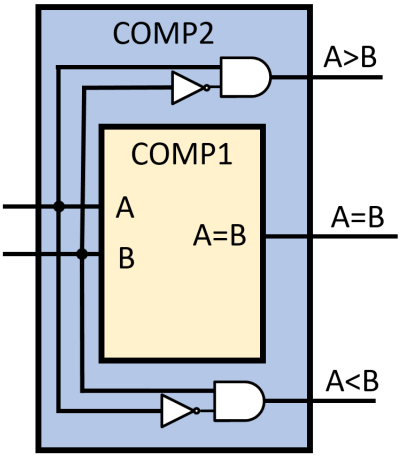


Figure 5: One-bit inequality comparator

(3) Two-bit inequality comparator

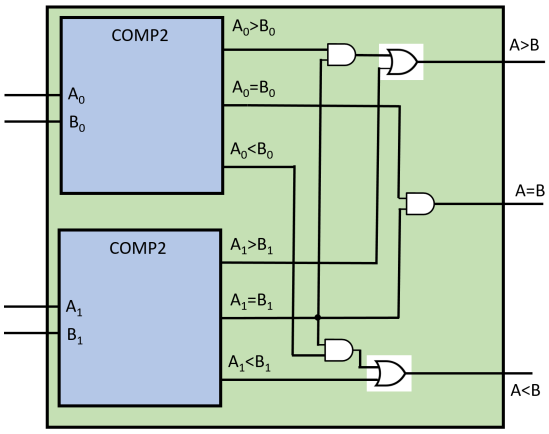


Figure 6: Two-bit inequality comparator

5. (10 points) The following figure shows a BCD-to-binary converter consisting of 4-bit adders.

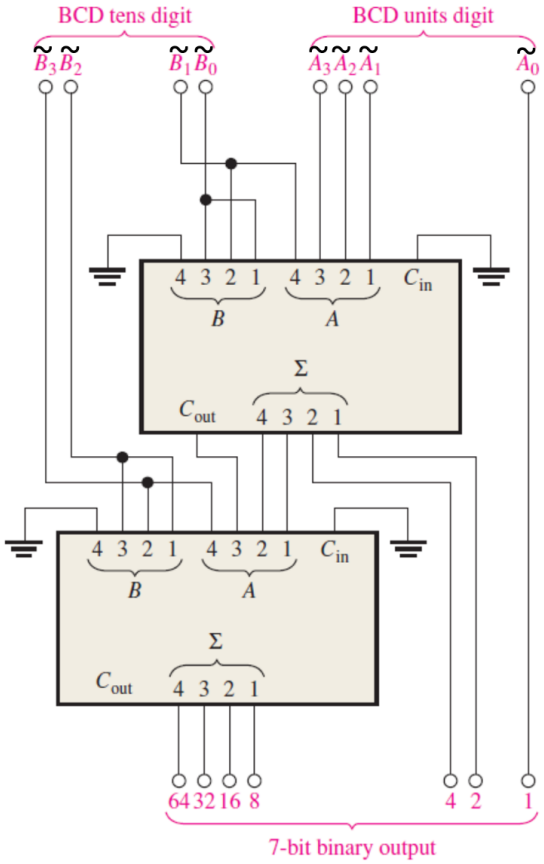


Figure 7: BCD-to-binary converter

(1) Express the final binary output into *decimal* in terms of input $\tilde{B}_3$, $\tilde{B}_2$, $\tilde{B}_1$, $\tilde{B}_0$, $\tilde{A}_3$, $\tilde{A}_2$, $\tilde{A}_1$, $\tilde{A}_0$;

(2) Given input $\tilde{B}_3\tilde{B}_2\tilde{B}_1\tilde{B}_0 = 0011$ and $\tilde{A}_3\tilde{A}_2\tilde{A}_1\tilde{A}_0 = 0101$, calculate the output C_{out} and $\Sigma_4\Sigma_3\Sigma_2\Sigma_1$ for both the top and bottom adders;

(3) Explain why such a configuration of $B_3B_2B_1B_0$ can correctly convert BCD to binary.

Solution:

(1) $Z = \tilde{B}_3 * 80 + \tilde{B}_2 * 40 + \tilde{B}_1 * 20 + \tilde{B}_0 * 10 + \tilde{A}_3 * 8 + \tilde{A}_2 * 4 + \tilde{A}_1 * 2 + \tilde{A}_0$

(2) The top adder: $C^t_{\text{out}} = 1$ and $\Sigma^t_4\Sigma^t_3\Sigma^t_2\Sigma^t_1 = 0001$.
The bottom adder: $C^b_{\text{out}} = 0$ and $\Sigma^b_4\Sigma^b_3\Sigma^b_2\Sigma^b_1 = 0100$.

(3) For input $\tilde{B}_0 = 1$, 10 can be represented as $8+2$. Similarly, input $\tilde{B}_1 = 1$, i.e. 20 can be represented by $16+4$. Since the least significant bit input $\tilde{A}_0$ is directly connected to the output, A_4, A_3, A_2 and A_1 of the top adder are actually weighted by 16, 8, 4 and 2, respectively. The same weighting configuration is also true for B_4, B_3, B_2 and B_1. Therefore, input $\tilde{B}_0$ is connected to the B_1 and B_3 of the top adder whereas $\tilde{B}_1$ to A_4 and B_2.

For the bottom adder, A_4, A_3, A_2 and A_1 are weighted by 64, 32, 16 and 8. By representing $80 = 64+16$ and $40 = 32+8$, we can connect input $\tilde{B}_2$ to B_1 and B_3 of the top adder whereas $\tilde{B}_3$ to A_4 and B_2.

6. (10 points) Design a circuit with four input, namely A, B, C and D and one output Y. The output is 1 when there are at least three consecutive 1's or 0's in the inputs, e.g. 1110.

(1) Develop the truth table for Y with four inputs A, B, C and D;

(2) Use a Karnaugh map to find the minimum **product of sums** (POS) form of Y;

Solution:

(1) The truth table for Y with four inputs A, B, C and D is given as follows:

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

(2) Use a Karnaugh map to find the minimum **product of sums** (POS) form of Y;

$$Y = \left(\overline{B} + C\right)\left(\overline{A} + C + \overline{D}\right)\left(A + B + \overline{C}\right)\left(A + \overline{C} + D\right)\left(\overline{A} + B + \overline{C}\right) \tag{2}$$

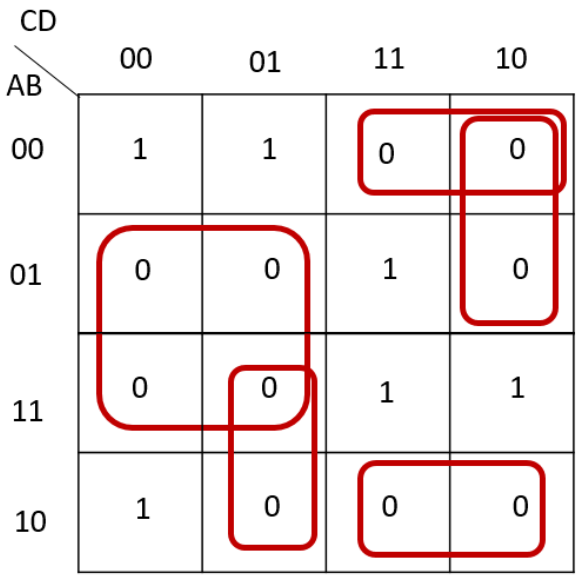


Figure 8: Karnaugh Map for Q.6.

7. (10 points) Given the circuit and input waveforms as shown below

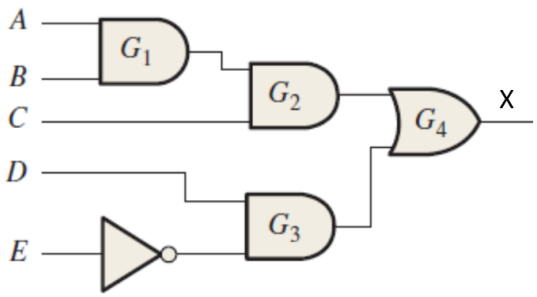
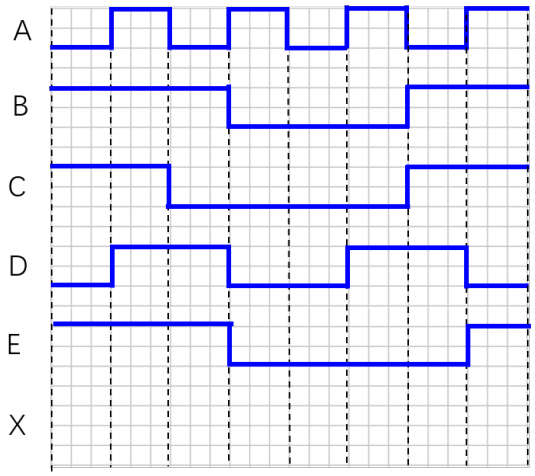


Figure 9: Logic circuit and input waveforms for Q.4.

(1) Write the Boolean expression of X;

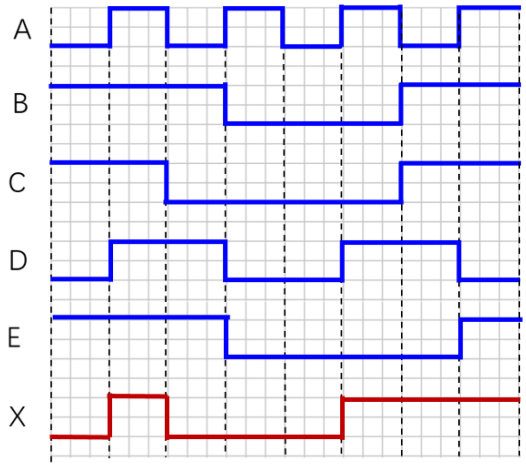
(2) Sketch the output waveform X in proper relation to the inputs.



Solution:

(1) $X = ABC + D\overline{E}$;

(2) The output timing diagram is as follows:



8. (10 points) Implement the logic function specified in the following truth table by using a 4-input data selector and all necessary basic logic gates.

Inputs				Output
A_3	A_2	A_1	A_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Table 3: The desired logic function table

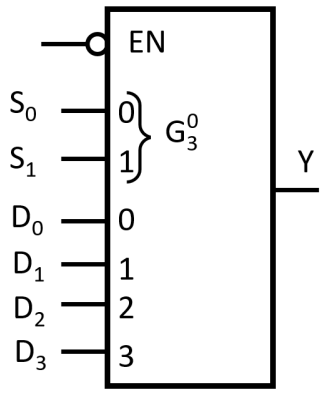
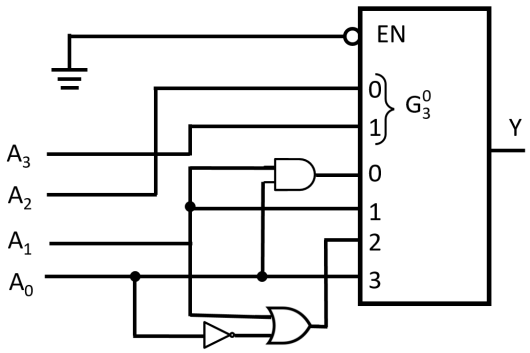


Figure 10: The 4-input Logic Symbol

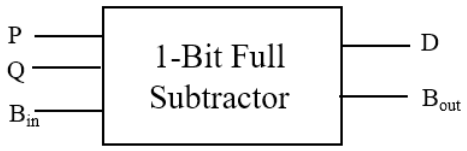
Solution:

Inputs		Output
A_3	A_2	Y
0	0	A_1A_0
0	1	A_1
1	0	$A_1 + \overline{A_0}$
1	1	A_0

Table 4: The equivalent logic function table



9. (15 points) We implement a 1-bit full subtractor that performs the binary subtraction as shown below.



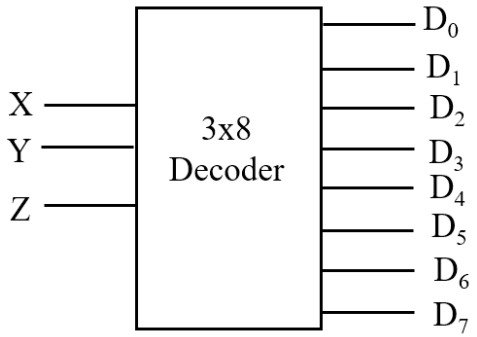
P and Q are 1-bit variables and B_{in} is the borrow input from the previous stage. It produces two outputs: the difference D and the borrow output B_{out}. The truth table describing the function of this 1-bit full subtractor is shown in the following table.

B_{in}	P	Q	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

(1) Derive the Boolean expressions for D and B_{out} using K-maps;

(2) Implement D and B_{out} using all necessary basic logic gates;

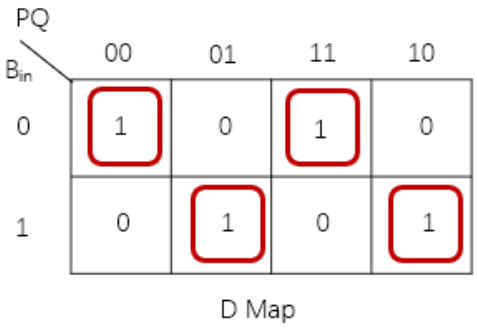
(3) Alternatively, we can design the 1-bit full subtractor above using a 3×8 decoder whose truth table is as shown below. Show your implementation with all necessary basic logic gates.

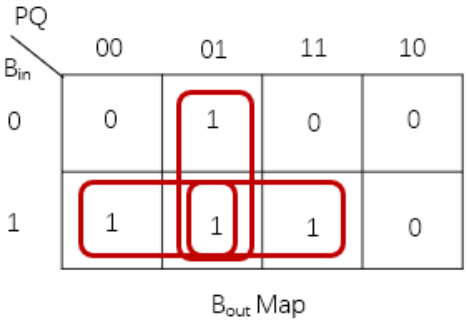


X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Solution:

(1) The logic expressions for D and B_{out} can be found as

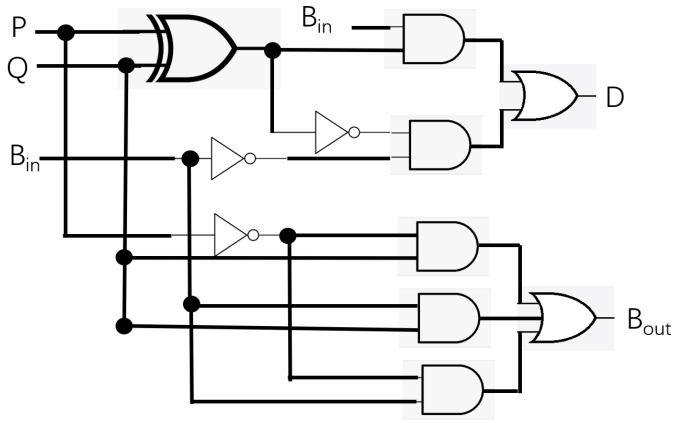




$$D = \overline{B_{\text{in}}}\ \overline{P \oplus Q} + B_{\text{in}}\,(P \oplus Q) \tag{3}$$

$$B_{\text{out}} = \overline{P}Q + B_{\text{in}}Q + B_{\text{in}}\overline{P} \tag{4}$$

(2) The subtractor can be implemented with basic logic gates as follows:



(3) Set X, Y and Z as B_{in}, P and Q respectively, we can use two four-input OR gates and the decoder to implement the full subtractor.

$$D = D_1 + D_2 + D_4 + D_7 \tag{5}$$

$$B_{\text{out}} = D_1 + D_4 + D_5 + D_7 \tag{6}$$