Tuan Nguen

**<u>Question 1:</u>**

a) Truth tables can be used to prove associativity and commutativity of XOR. To prove that XOR is associative, we need to prove that (a ⊕ b) ⊕ c = a ⊕ (b ⊕ c). The truth table for the LHS is:

| a | b | c | z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

The truth table for the RHS is:

| a | b | c | z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Therefore, XOR is associative. From the truth table given, we can also see that XOR is commutative. In the cases when the inputs a and b are both 0 or 1, it does not matter if it is a ⊕ b or $b \oplus a$. In the cases when one of the inputs is 0 and the other is 1, we can see that the result is not affected from the ordering.

b) From the given truth table for XOR, we can see that there are two cases when the output is 1. Let the function p be such that for input a and b, the output is 1 iff $a = 0$ and $b = 1$. Similarly, let q be the function such that for input a and b, the output is 1 iff $a = 1$ and $b = 0$. Therefore, $p = \neg a \wedge b$, $q = a \wedge \neg b$. So XOR = $(\neg a \wedge b) \vee (a \wedge \neg b)$.

c) I can't prove that we can't.

d) I am going to prove that the circuit computes $a \oplus b$ with a truth table.

| a | b | u | v | w | z |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |

**Question 2:**

a) *See pdf file for circuit*

b) *See pdf file for circuit*

c) The principle that connects a) and b) to the CMOS gates from the lectures is that if two p-types are parallel, than the corresponding n-types are in series and vice versa.

**Question 3:**

a) Suppose that if $a = b = 1$, the output is going to be 0. This means that if b is 1, regardless of the input a and the state of z at the previous clock edge, the output is going to be 0. This means that the function might be $z = \neg b \wedge (\ldots)$. The remaining cases are when $b = 0$. If $a = 0$, then the state of z will remain from the previous clock edge. Otherwise, the state changes to 1. Therefore, the function will be:

$$z_t = (z_{t-1} \vee a) \wedge \neg b$$

*See pdf for circuit*

b) The diagram of state is:

| a | b | $z_{t-1}$ | $z_t$ | w |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

We can see that w is 1 only when both b and $z_{t-1}$ are 1. So the circuit should be: *See pdf for circuit*

**<u>Question 4:</u>**
a) *See pdf file for circuit*

b) I am going to build a 3-bit binary counter for simplicity. Suppose we have 3 T-type flip flops for the three bits, which share a clock input. The cycle, which the counter should go through is: 000, 001, 010, 011, 100, 101, 110, 111, and back to 000. This means that the three t inputs for the three T-type flip flops should be: 001, 011, 001, 111, 001, 011, 001, 111. We can notice that the input for the third flip flop(say C) is always 1. We can also notice that the input for B(second T flip flop) is 1, only when the output of C is 1. There is a pattern – the input for the next flip flop should be 1, only when the output of all T flip flops before it are 1. Therefore, the circuit should be: *See the pdf file for circuit*

c) From part a), we know how to convert D flip flop to T flip flop with XOR gate. The half-adder has one XOR gate and one AND gate. So we can use only the sum output of the half-adder to simulate T flip flop and use the carry output, to simulate AND gates. Therefore, the circuit should be: *See pdf for circuit*

d) The connection between part b) and c), is that we can always simulate T flip flop with D flip flop and XOR gate, and vice versa.

**<u>Question 5:</u>**
Suppose the wanted behaviour can be achieved by using the circuit from the lectures. Let's look at the diagram of states:

| a | o | d |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

where a is the input, o is the output of the circuit from lectures and d is the desired output. The only thing that has to be changed about the behaviour of the circuit from the lectures is when the cord is pulled, the light should not light up. This function is just ¬a ∧ c. *See pdf for circuit*

**Question 6:**

a) All the gates can be simulated using NOR gates. *See pdf for diagram*