CS 203.4860 **Secure Multi-Party Computation** Spring 2023

Homework 1: Report

Lecturer: Dr. Adi Akavia *Student(s): Nir Segal, Hallel Weinberg, Michael Rodel*

1 exercise 1

a	x	$f_{a,4}(x)$
0	0	0
0	1	0
0	2	0
0	3	0
1	0	0
1	1	0
1	2	0
1	3	0
2	0	0
2	1	0
2	2	1
2	3	1
3	0	0
3	1	0
3	2	1
3	3	1

2 exercise 2

In order to implement a Boolean circuit as in the requirements of the question, we would like to build a black box that will perform multiplication between two binary numbers of length 2 ($a_1 \cdot x_1, a_2 \cdot x_2$) and a black box that will perform an addition between two binary numbers of length 4 ($a_1 \cdot x_1 + a_2 \cdot x_2$), as shown in figure 1, and in figure 3.
Now, we will describe each circuit verbally.
First, we will describe the circuit of multiplication (the circuit in Figure 1). As input, we get 2 numbers: x, and y, which correspond to $a_i \cdot x_i$. Each number consists of 2 bits (a and b for x and c and d for y). Generally, this acts the same as "long multiplication". Since AND between 2 bits returns the multiplication of them, the rightest AND gate takes the bits b and d, and outputs their AND value. This returned bit is the LSB of our output. Then we also take d and multiply it with a (AND). To this value, we add the multiplication of b and c (AND). The addition is performed with XOR gate - and then becomes the second bit (from right) of the output. To store the carry bit of this multiplication we use AND gate. Now, to compute $a \cdot c$, we multiply using AND gate these bits, and then add them with the carry bit of the previous bits (with XOR gate). In case both $a \cdot c = 1$ and the carry bit from the previous bits are 1, we check the carry bit with AND gate. This carry bit will be

our MSB. The output 4 bits represent the multiplication of $x = ab$ and $y = cd$.
Second, we will describe the circuit shown in Figure 2. This circuit shows how to add 2 bits, with consideration to the carry bit. Given 2 bits, x_i and x_j, their sum (without carry) will be their XOR. To this sum we add the carry bit using XOR, and this sum will be the LSB of our output. To check the carry bit of the addition of x_i, x_j and the carry bit, we pass the sum of $x_i + x_j$ (x_i XOR x_j) and the carry bit to an AND gate (carry check). The output of this AND gate generates the carry, and then we pass it through a XOR gate, with the carry bit of x_i and x_j - which generated by applying AND gate on x_i and x_j. The returned value of the XOR is the sum of its inputs, and this will be our MSB output bit.
Third, we will describe the circuit shown in Figure 3. This circuit generalizes and summarizes the addition procedure. Here, x represents $a_1 \cdot x_1$, and y represents $a_2 \cdot x_2$. This also acts like regular long multiplication. First, we add the LSBs (d and h) using XOR gate, and its output generates us the LSB of the general output for add. Then, we calculate the carry of these bits (d and h), using and gate, and "send" the output to the carry bit input of AddByBits circuit (which shown in Figure 2), and the other inputs will be the next bits (c and g). AddByBits generates output of size 2 bits, so just as in long multiplication, the LSB will "stay" for the general output, and the MSB will be send to be the carry input of the next to bits of x and y (b and f). In the same way, the the LSB will "stay" for the general output, and the MSB will be send to be the carry input of the next to bits of x and y (a and e). Then, we will apply AddByBits circuit on the carry bit input, and a and e inputs. The LSB, as in previous stages will "stay" for the general output, and the MSB will also "stay" for the general output - since the largest number the addition of 4 bits' numbers can produce is 30 (in decimal), which requires $\lceil log_2 30 \rceil = 5$ bits to represent.
Now, we combine the previous circuits to one circuit (i.e. the circuit shown in Figure 4), which will bring us to check finally whether $a_1 \cdot x_1 + a_2 \cdot x_2 \geq 4$ or not. First, we take the bits of a_1 and x_1, and send them to mul black-box (i.e. the circuit shown in Figure 1), and the same procedure we do with a_2 and x_2. Then, the result of each multiplication is sent to the add black-box (i.e. the circuit shown in Figure 3). Now, we have to check if the result is equal or greater than 4, or not. Here, we have to note that the relevant bits for the checking are only the 3 MSB bits of the sum: because 4 in binary, using 5 bits is 00100, so only if one (or more) of the 3 MSB bits is 1, the answer is 'yes' (1), and otherwise - not (0). So, because we want to check if 1 or more bits of the 3 MSB bits are 1, we should do OR between these 3 bits: (bit_5 OR bit_4 OR bit_3). Equally, we can check: [(bit_5 OR bit_4) OR (bit_4 OR bit_3)]. Now, we will present the OR gate using only XOR and AND gates. Assume we have the following expression: x OR y. Then: (x OR y) $\equiv$ [(x XOR y) XOR (x AND y)]. The result of this check will be our output bit (1 or 0), since if at least 1 bit of the sum is one, the sum is certainly equals or greater than 4, as required.

3 exercise 3

Description of the circuit: First, we will multiply a_1, x_1 and multiply a_2, x_2 by two $\times mod 11$ gates. We will note that the maximum result that will be obtained as a result of the multiplication is 9 so that the modulo operation will not change the result.
Now, in order to add $a_1 \cdot x_1$ and $a_2 \cdot x_2$, we cannot use one $+mod 11$ gate because after the addition we will get a number in $\{0, 18\}$ and for numbers in $\{11, 18\}$ the modulo operation

will change the expected result.
Therefore, we will separate into 2 situations:
1. If the result of the addition will be between 0 and 10: we will use the $+mod11$ gate.
2. If the addition result is between 11 and 18:
According to the distinction[1], $a_1 \cdot x_1 >= 4$ or $a_2 \cdot x_2 >= 4$. Therefore, instead of using the $+mod11$ gate we will check if $a_1 \cdot x_1 >= 4$ or $a_2 \cdot x_2 >= 4$.
In short, as you can see in figure 6, after the two $\times mod11$ gates we will perform 3 steps:
1. We will check if $a_1 \cdot x_1 >= 4$.
2. We will check if $a_2 \cdot x_2 >= 4$.
3. We will add $a_1 \cdot x_1$ and $a_2 \cdot x_2$ by a $+mod11$ gate.
Note: Later on, we will define in detail a black box that, given a number, will determine whether it is greater than or equal to 4.
Now, we would like to check whether for one of the three cases above, we returned 1, i.e. we got a number greater than or equal to 4.
To do this, first we will use a box that implements a not gate - if we returned 1 (i.e. the number is greater than or equal to 4) we will get 0. If we returned 0 (i.e. the number is less than 4) we will get 1.
After we execute not for each of the three results of the black box (which checks if a number is greater than or equal to 4) and mark the 3 results as n_1, n_2, n_3 (WLOG), we can simply execute $a = \times mod11(n_1, n_2)$ and $b = \times mod11(n_2, n_3)$ and finally $\times mod11(a, b)$.
Note that because we used the not gate, the output will be not of $\times mod11(a, b)$.

A black box: Greater than or equal to 4 (as you can see in figure 5).
Explanation: First, a number is greater than or equal to 4 if it is equal to 0/1/2/3. Therefore, we will use 4 $+mod11$ gates that will add the input and $0/-1/-2/-3$.
Now, we will use Fermat's little theorem to get 0 if the result of the addition is 0 and 1 otherwise.
We would like to return 1 if we received one Fermat's little theorem that returned 1, so we will use 3 $\times mod11$ gates that will be used as an or operation.

4 exercise 4

Complexity analysis of the Boolean circuit:

(i) circuit depth: 12.
The longest path I chose starts in the input x_{10}, the path go through the $mul(x, y)$ with one and gate. After that we $Add(x, y)$ through the first entry of the first multiplication, and go out through the fifth entry. And after that we go in the last section in the circuit through 2 and gates and 2 xor gates. Overall we go through 7 and gates and 5 xor gates, therefor 12 gates.

[1] **Distinction:** The addition result will be between 11 and 18 if $a_1 \cdot x_1 >= 4$ or $a_2 \cdot x_2 >= 4$.
Proof: Suppose the addition result is between 11 and 18. Negatively assume that $a_1 \cdot x_1 < 4$ or $a_2 \cdot x_2 < 4$. Therefore, the maximum value of $a_1 * x_1 + a_2 * x_2$ is 9, contrary to the fact that the addition result is greater than 11.

(ii) circuit size: 42.
In the circuit there is 22 and gates and 20 xor gates.

(iii) $\times$-depth: 7.
In the path I showed in (i), there are the most and gates.

(iv) number of MULT: 22.

Complexity analysis of the arithmetic circuit:

(i) circuit depth: 20.
The longest path I chose starts in the input x_1, the path go through the multiplication gate and afterward go through addition gate. The path then enter to $GreaterThanFour(x, y)$. In $GreaterThanFour(x, y)$ we go through 12 gates and after we enter out from $GreaterThanFour(x, y)$ we going through another 6 additional gates. Therefor we go through 15 multiplication gates and 5 addition gates.

(ii) circuit size: 142.
There is 125 multiplication gate and 17 addition gates.

(iii) $\times$-depth: 16.
In the path I showed in (i), there are the most multiplication gates.

(iv) number of MULT: 125.

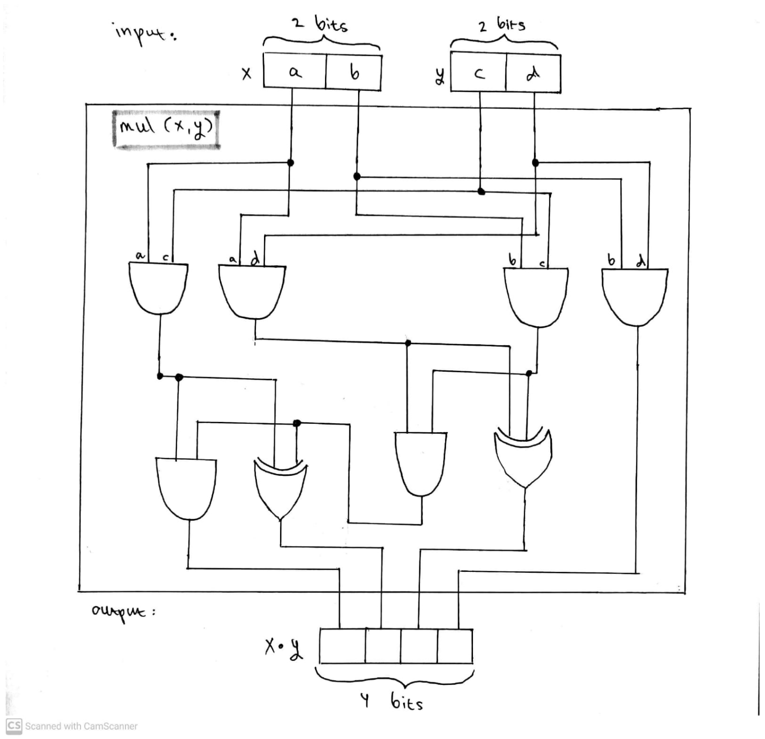

Figure 1: A black box of multiplication

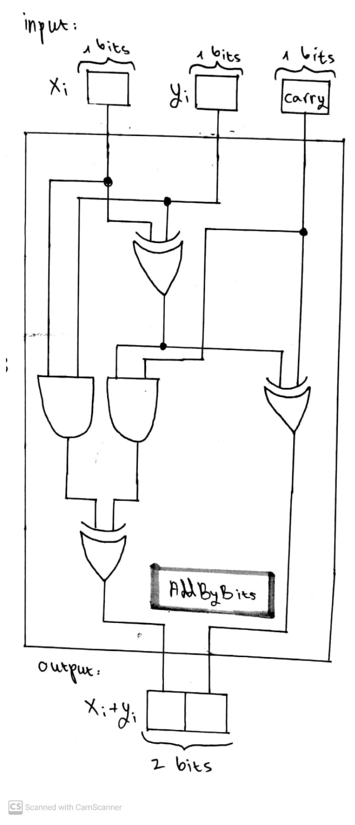

Figure 2: A black box of addition of 2 bits with consideration for carry

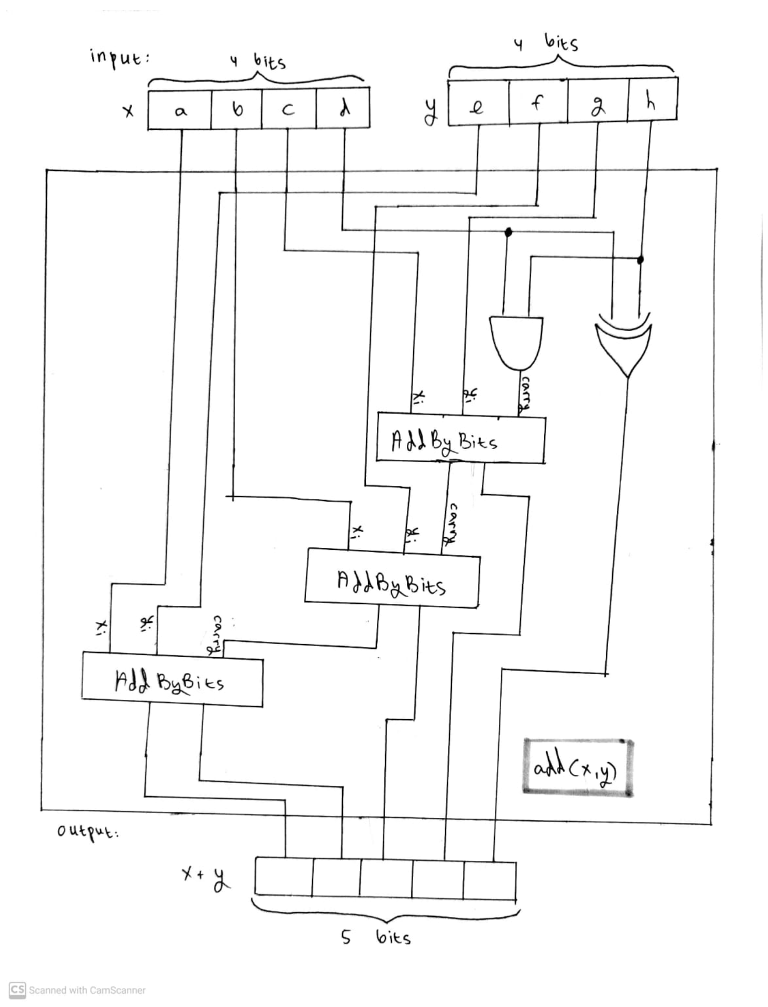

Figure 3: A black box of addition of 2 numbers

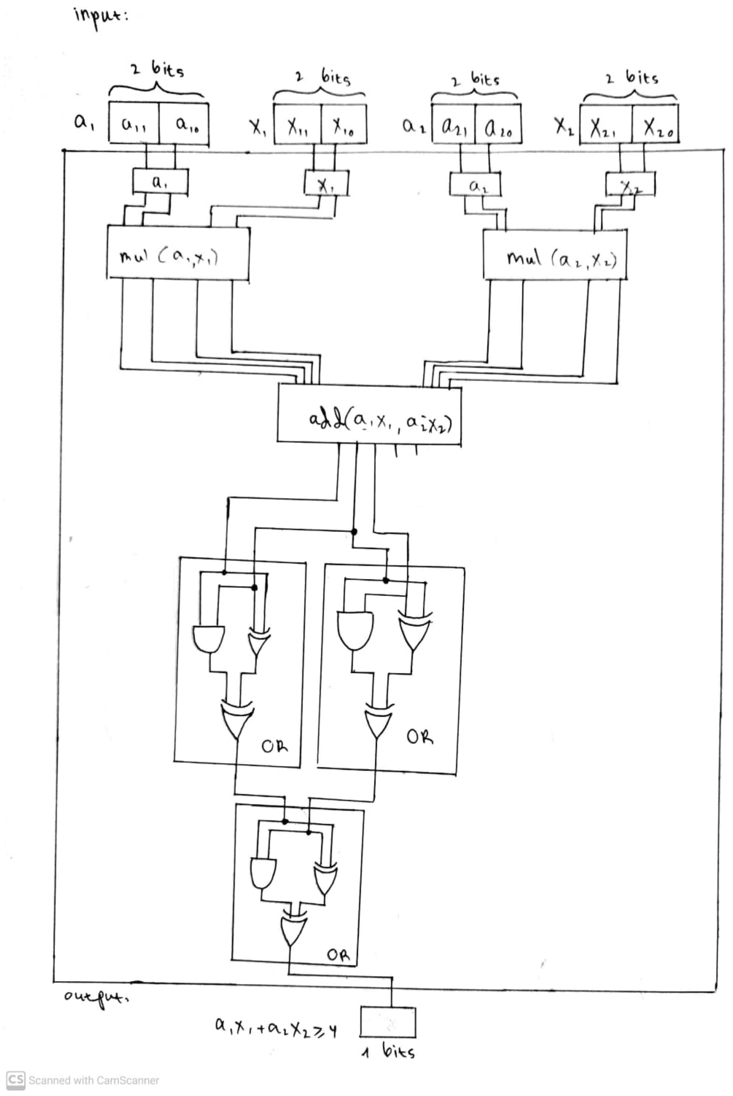

Figure 4: The entire Boolean circuit

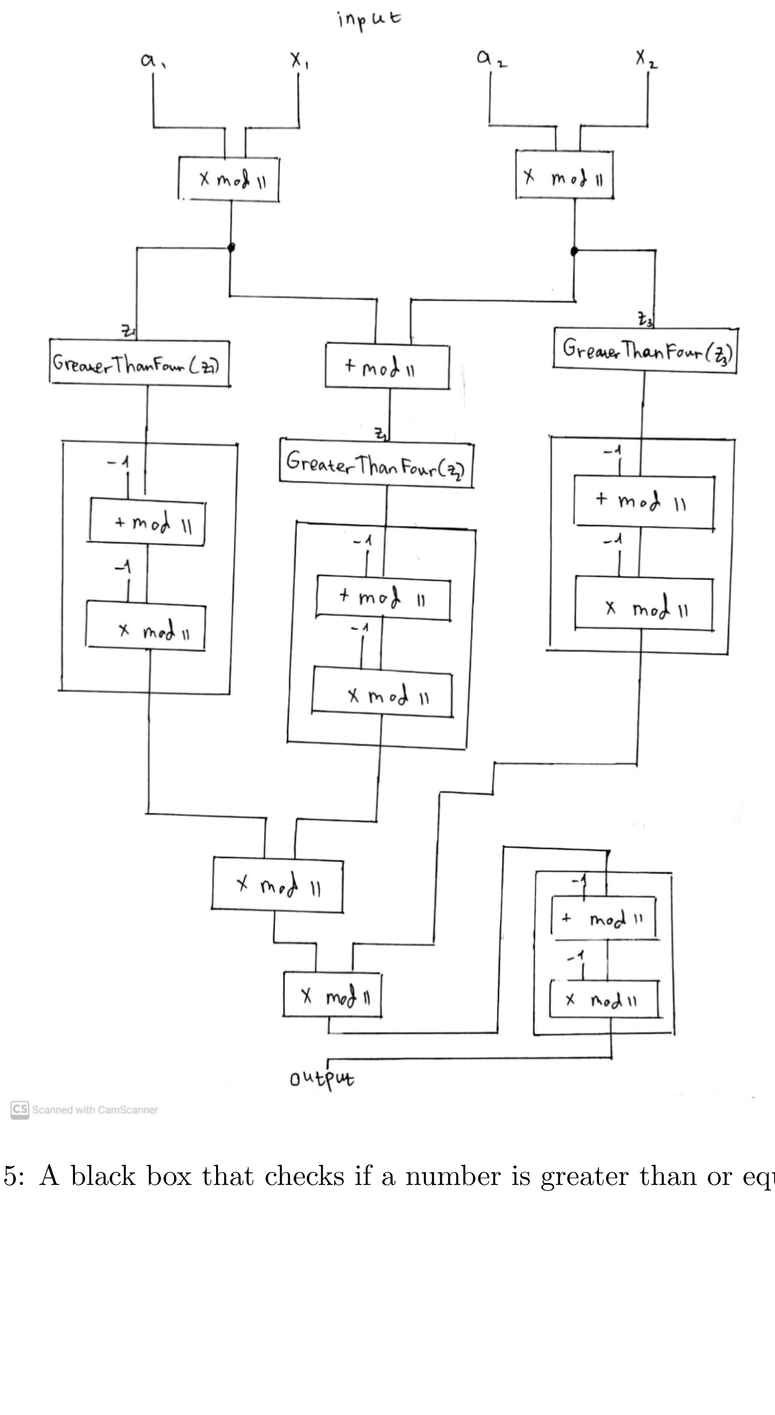

Figure 5: A black box that checks if a number is greater than or equal to 4

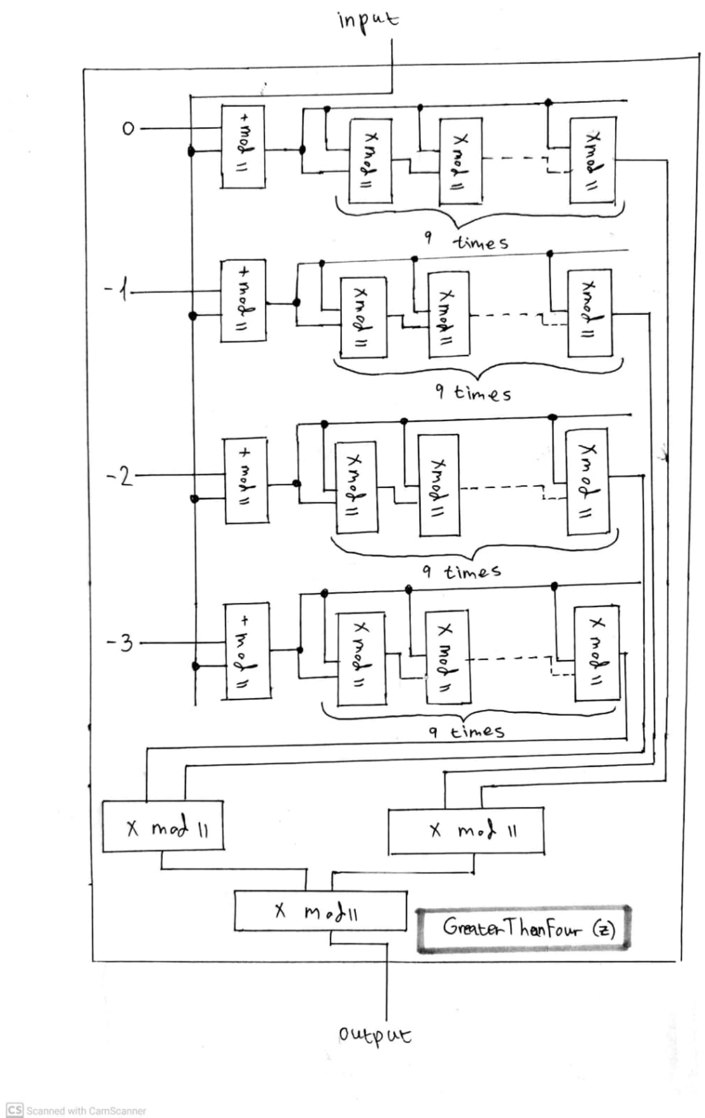

Figure 6: The arithmetic circuit