11.1 Arithmetic Circuits

Let $GF(q)$ be a finite fireld of order $q > 2$. Then TRUE is associated with $1 \in GF(q)$ and FALSE with $0 \in GF(q)$. A binary circuit C_b can then be transformed to an equivalent arithmetic circuit by exchanging the four gates with the following operations:

11.1.1 Logical AND

$\text{AND}(x, y) = x \cdot y \ (mod\ 2)$
$\Rightarrow \ \text{AND}(x, y) = 1 \ \Leftrightarrow \ x = y = 1$

11.1.2 Logical NOT

$\text{NOT}(x) = (x + 1) \ (mod\ 2)$
$\Rightarrow \ \text{NOT}(0) = 1$ and $\text{NOT}(1) = 0$

11.1.3 Logical OR

We know that $x \vee y = \neg(\neg x \wedge \neg y)$, so in our notation:
$\text{OR}(x, y) = \text{NOT}(\text{AND}(\text{NOT}(x), \text{NOT}(y))) = (((x + 1) \ (mod\ 2)) \cdot ((y + 1) \ (mod\ 2)) + 1) \ (mod\ 2)$
$\Rightarrow \ \text{OR}(0, 0) = 0$ and $\text{OR}(0, 1) = \text{OR}(1, 0) = \text{OR}(1, 1) = 1$

11.1.4 Logical XOR

$\text{XOR}(x, y) = (x + y) \ (mod\ 2)$
$\Rightarrow \ \text{XOR}(0, 0) = \text{XOR}(1, 1) = 0$ and $\text{XOR}(0, 1) = \text{XOR}(1, 0) = 1$

11.2 Multiplication Gate with Preprocessing

We have a finite field $GF(q)$ of prime order with $x, y, z, w_j, w_k, m_j, m_k, w_t$ as defined in the exercise. We can then create the following equation:

$$\begin{aligned} m_j m_k + m_j y + m_k x + z &= (w_j - x) \cdot (w_k - y) + (w_j - x) \cdot y + (w_k - y) \cdot x + xy \\ &= w_j w_k - w_j y - w_k x + xy + w_j y - xy + w_k x - xy + xy \\ &= w_j w_k \end{aligned}$$

Therefore the requirement $m_j m_k + m_j y + m_k x + z = w_j w_k$ is fulfilled and hence the product $w_t = w_j w_k$ can be calculated from m_j, m_k, y, x, z by using the following steps for each of the summands:

$[w_j - x, w_k - y]$ can be calculated locally as parties possess sharings of w_j, x, w_k and y, and subtraction is a local operation.

$m_j = w_j - x$ and $m_k = w_k - y$ can the be reconstructed as described in the exercise, using the protocol for output wires.

$m_j \cdot m_k$ can the be calculated as a local multiplication of two reconstructed values.

$[m_j \cdot y]$ and $[m_k \cdot x]$ can then similarly be calculated locally using the multiplication with a constant as discussed in the lecture, since the parties posess sharings $[x]$ of x and $[y]$ of y.

$[z]$ is known to the parties as they agreed on this during the preprocessing step.

As such, each party can calculated $m_j \cdot m_k + [m_j \cdot y] + [m_k \cdot x] + [z]$ locally, with values that are either known by all parties or have a sharing of it. So given that each party possesses the pre-shared sharings of $[x]$, $[y]$ $[z] = x \cdot y$, the multiplication can be optimized to only consist of local operations and a reconstruction operation at the end.