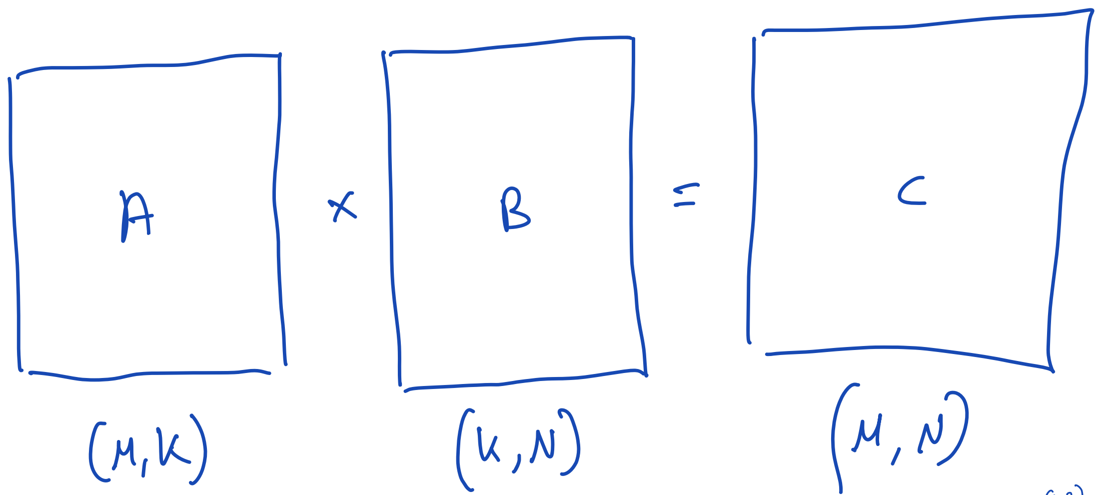
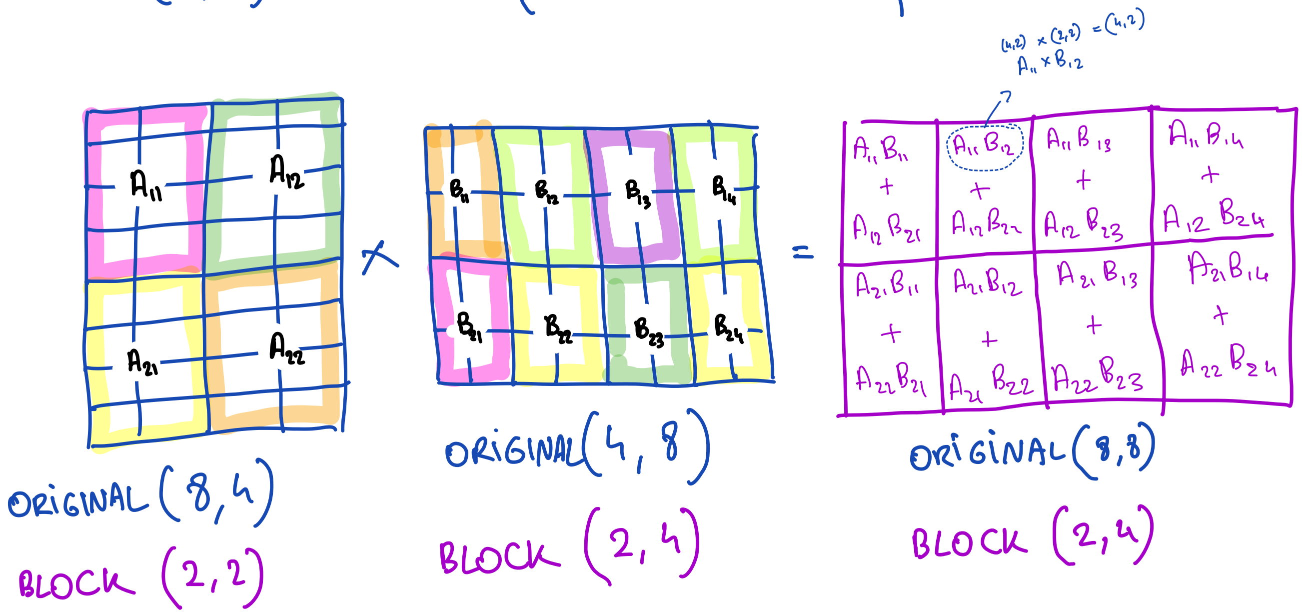
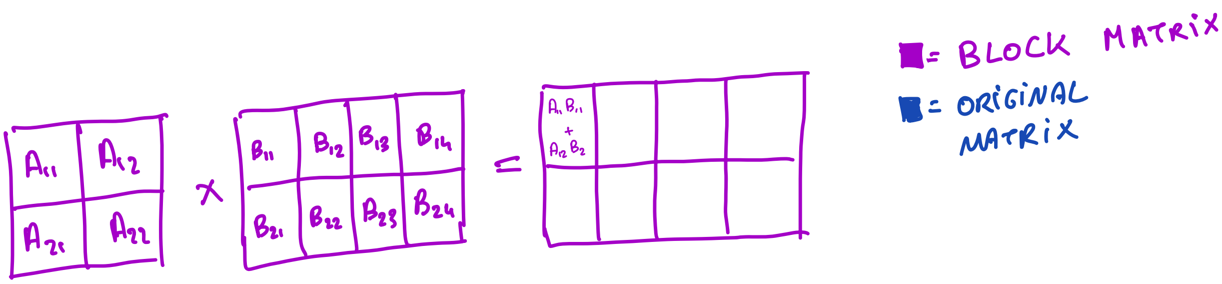
BLOCK MATRIX MULTIPLICATION







WHY SHOULD WE CARE ?

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N\times N}, \quad \cancel{\mathbf{P} = \mathrm{softmax}(\mathbf{S}) \in \mathbb{R}^{N\times N}}, \quad \mathbf{O} = \overset{\mathbf{S}}{\cancel{\mathbf{P}}}\mathbf{V} \in \mathbb{R}^{N\times d},$$

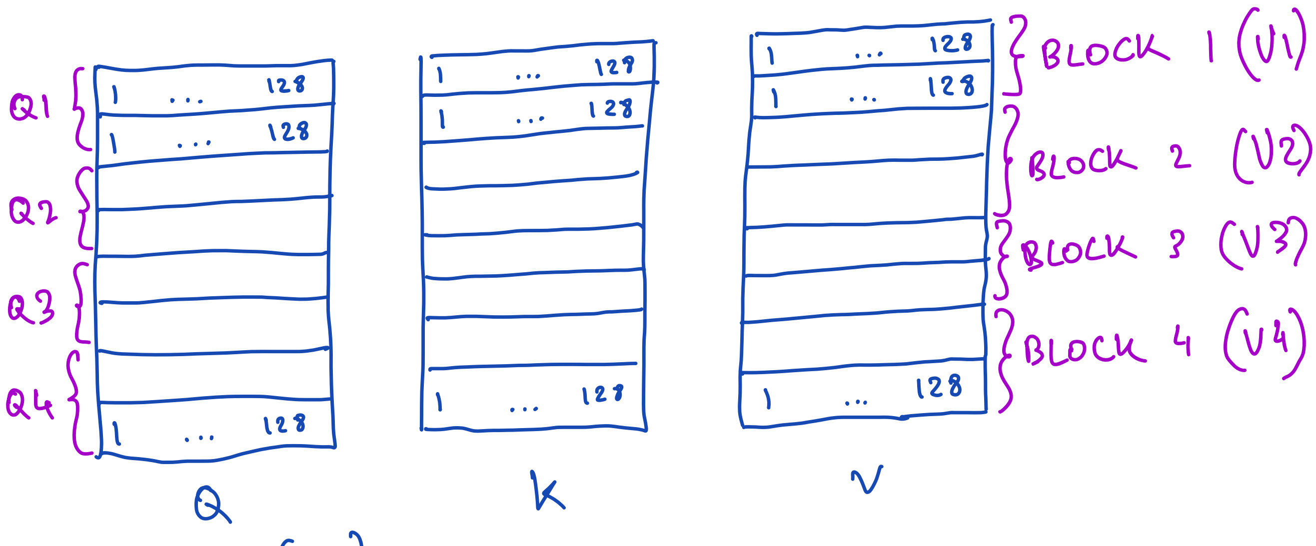
Let's ignore the softmax for a moment...

Suppose for a moment that we want to do the following operation

$$O = (QK^T)V \in R^{N\times d}$$

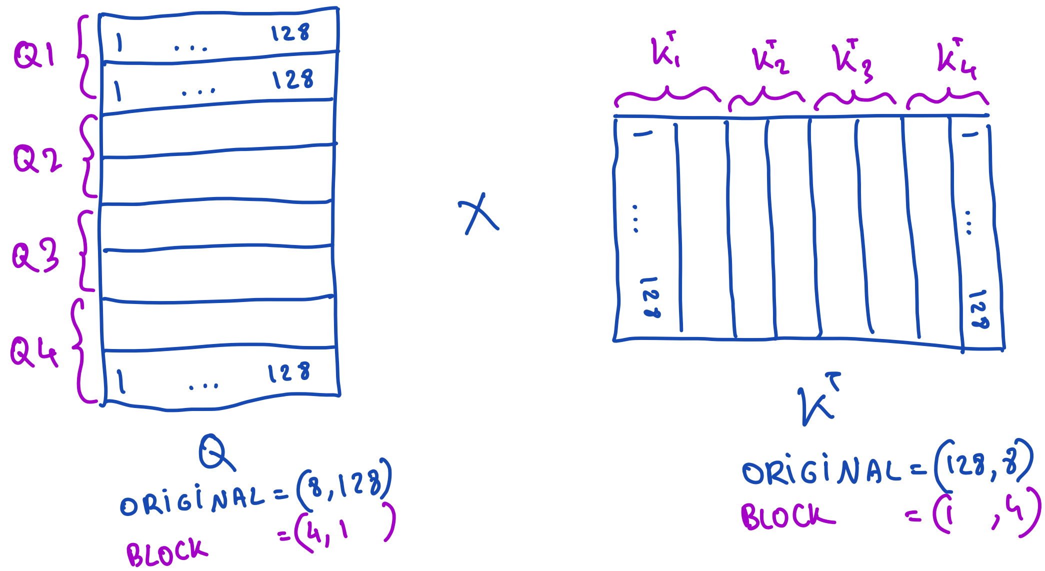
■ = BLOCK
■ = ORIGINAL MATRIX

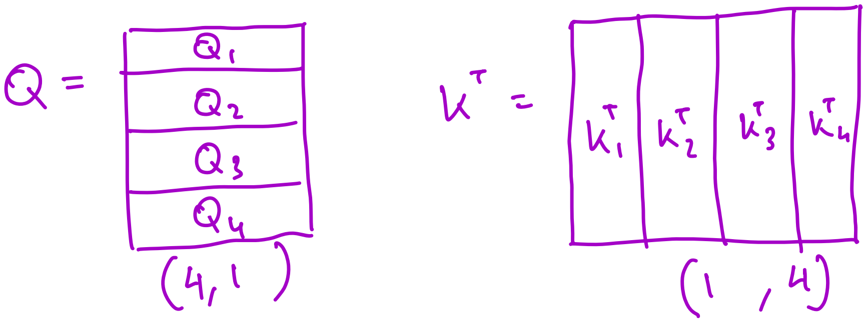
We know that $Q, K, V \in R^{N\times d}$

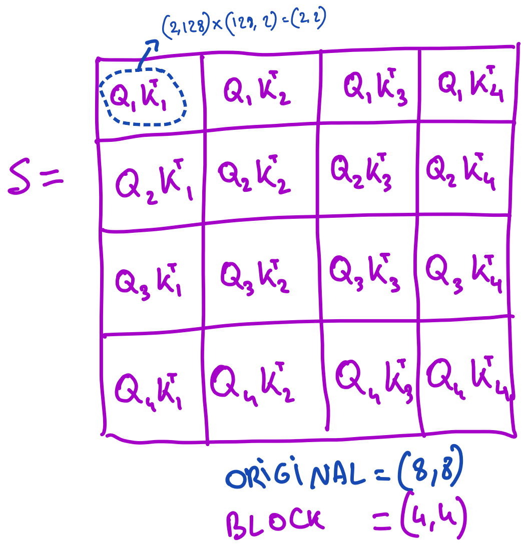


ORIGINAL = (8, 128)

BLOCK = (4, 128)


Q1
Q2
Q3
Q4
1 ... 128
1 ... 128
1 ... 128
×
K1ᵀ K2ᵀ K3ᵀ K4ᵀ
128
128
Q
ORIGINAL = (8,128)
BLOCK = (4,1)
Kᵀ
ORIGINAL = (128,8)
BLOCK = (1,4)


Q =
Q1
Q2
Q3
Q4
(4,1)
Kᵀ =
K1ᵀ K2ᵀ K3ᵀ K4ᵀ
(1,4)


(2,128)×(128,2)=(2,2)
S =
Q1K1ᵀ Q1K2ᵀ Q1K3ᵀ Q1K4ᵀ
Q2K1ᵀ Q2K2ᵀ Q2K3ᵀ Q2K4ᵀ
Q3K1ᵀ Q3K2ᵀ Q3K3ᵀ Q3K4ᵀ
Q4K1ᵀ Q4K2ᵀ Q4K3ᵀ Q4K4ᵀ
ORIGINAL = (8,8)
BLOCK = (4,4)

Let's multiply by V

$$S = \begin{bmatrix} Q_1K_1^T & Q_1K_2^T & Q_1K_3^T & Q_1K_4^T \\ Q_2K_1^T & Q_2K_2^T & Q_2K_3^T & Q_2K_4^T \\ Q_3K_1^T & Q_3K_2^T & Q_3K_3^T & Q_3K_4^T \\ Q_4K_1^T & Q_4K_2^T & Q_4K_3^T & Q_4K_4^T \end{bmatrix}$$

ORIGINAL = (8,8)
BLOCK = (4,4)

$$V = \begin{array}{c} \begin{bmatrix} 1 & \dots & 128 \\ 1 & \dots & 128 \\ & & \\ & & \\ & & \\ & & \\ & & \\ 1 & \dots & 128 \end{bmatrix} \end{array} \begin{array}{l} \left.\right\} V_1 \\ \left.\right\} V_2 \\ \left.\right\} V_3 \\ \left.\right\} V_4 \end{array}$$

ORIGINAL = (8,128)
BLOCK = (4,1)

$$\left[(2\times128)\times(128\times2)\right]\times(2\times128) = (2\times2)\times(2\times128) = (2\times128)$$

$$O = \begin{bmatrix} (Q_1K_1^T)V_1 + (Q_1K_2^T)V_2 + (Q_1K_3^T)V_3 + (Q_1K_4^T)V_4 \\ (Q_2K_1^T)V_1 + (Q_2K_2^T)V_2 + (Q_2K_3^T)V_3 + (Q_2K_4^T)V_4 \\ (Q_3K_1^T)V_1 + (Q_3K_2^T)V_2 + (Q_3K_3^T)V_3 + (Q_3K_4^T)V_4 \\ (Q_4K_1^T)V_1 + (Q_4K_2^T)V_2 + (Q_4K_3^T)V_3 + (Q_4K_4^T)V_4 \end{bmatrix}$$

EACH BLOCK OF THE OUTPUT MATRIX O IS ACTUALLY MADE UP OF TWO ROWS!

PSEUDOCODE

```
FOR EACH BLOCK Qi
    Oi = ZEROES(2,128)    // OUTPUT IS INITIALLY ZEROES
    FOR EACH BLOCK KJ
        Oi ← Oi + (Qi KJ^T) VJ
    END FOR
END FOR
```

WAIT... What happened to the SOFTMAX?

Let's restore it... with a twist!

S_{11}

$(2,128)\times(128,2)=(2,2)$

$$S = \begin{bmatrix} Q_1K_1^T & Q_1K_2^T & Q_1K_3^T & Q_1K_4^T \\ Q_2K_1^T & Q_2K_2^T & Q_2K_3^T & Q_2K_4^T \\ Q_3K_1^T & Q_3K_2^T & Q_3K_3^T & Q_3K_4^T \\ Q_4K_1^T & Q_4K_2^T & Q_4K_3^T & Q_4K_4^T \end{bmatrix} \xRightarrow{\text{SOFTMAX}^*} P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{bmatrix}$$

$(2,128)\times(128,2)=(2,2)$

ORIGINAL = (8,8)
BLOCK = (4,4)

ORIGINAL = (8,8)
BLOCK = (4,4)

$$\text{SOFTMAX}^*(S_{ij}) = \exp\left[S_{ij} - \text{rowmax}(S_{ij})\right]$$

* Note: $P_{11} = \text{softmax}^*(Q_1K_1^T)$
$P_{12} = \text{softmax}^*(Q_1K_2^T)$
etc...

This is a 2x2 matrix

Let's multiply by V

$m_{11} = (2 \times 1)$

$P_{11} = (2 \times 2)$

$$P = \begin{array}{|c|c|c|c|} \hline P_{11} & P_{12} & P_{13} & P_{14} \\ \hline P_{21} & P_{22} & P_{23} & P_{24} \\ \hline P_{31} & P_{32} & P_{33} & P_{34} \\ \hline P_{41} & P_{42} & P_{43} & P_{44} \\ \hline \end{array} \times \begin{array}{c} V \\ \begin{array}{|c|} \hline 1 \quad \dots \quad 128 \\ \hline 1 \quad \dots \quad 128 \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline 1 \quad \dots \quad 128 \\ \hline \end{array} \end{array} \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} V_1 \\ \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} V_2 \\ \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} V_3 \\ \left.\vphantom{\begin{array}{c}1\\1\end{array}}\right\} V_4 \end{array}$$

ORIGINAL = (8,8)
BLOCK = (4,4)

ORIGINAL = (8,128)
BLOCK = (4,1)

$$O = \begin{array}{|c|} \hline P_{11}V_1 + P_{12}V_2 + P_{13}V_3 + P_{14}V_4 \\ \hline \dots \\ \hline \dots \\ \hline \dots \\ \hline \end{array}$$

WRONG!

EACH OF THE P_{ij} BLOCK HAS BEEN INDIPENDENTLY CALCULATED, SO THE MAX ELEMENT FOR EACH ROW IN EACH BLOCK IS NOT THE GLOBAL FOR EACH ROW, BUT THE ONE LOCAL TO EACH BLOCK

IF ONLY WE HAD A WAY TO FIX THE SOFTMAX...

PSEUDOCODE

```
FOR EACH BLOCK Qi
    Oi = ZEROES(2,128)    // OUTPUT IS INITIALLY ZEROES
    FOR EACH BLOCK KJ
        PIJ = softmax*(Qi KJ^T)
        Oi <- Oi + PIJ VJ
    END FOR
END FOR
```

HOW CAN WE FIX THE PREVIOUS ITERATION'S OUTPUT ?

THE ONLINE SOFTMAX

$$
\begin{aligned}
&m_0 = -\infty \\
&l_0 = 0 \\
&\text{for } i = 1 \text{ to } N \\
&\qquad m_i = \max(m_{i-1}, x_i) \\
&\qquad l_i = l_{i-1} \cdot e^{m_{i-1} - m_i} + e^{x_i - m_i} \\
&\text{for } k = 1 \text{ to } N \\
&\qquad x_k \leftarrow \frac{e^{x_k - m_N}}{l_N}
\end{aligned}
$$

THE IDEA

IF WE CAN "FIX" THE SOFTMAX WHILE ITERATING ON A ROW, WE CAN ALSO FIX BLOCKS OF ROWS, SINCE THE SOFTMAX IS APPLIED INDEPENDENTLY TO EACH ROW

Let's see how to apply the online softmax here...

$$O_1 = P_{11}V_1 + P_{12}V_2 + P_{13}V_3 + P_{14}V_4 \Leftarrow$$ WE NEED TO FIX THESE

$\Downarrow$

PSEUDO CODE

```
FOR EACH BLOCK Qi
    Oi = ZEROES(2, 128)        // OUTPUT IS INITIALLY ZEROES
    FOR EACH BLOCK KJ
        PiJ = softmax*(Qi KJ^T)
        Oi <- Oi + PiJ VJ
    END FOR
END FOR
```

INITIALIZATION

$$m_0 = \begin{bmatrix} -\infty \\ -\infty \end{bmatrix}$$

$$\ell_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$O_0 = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$ 2x128 matrix

STEP 1

$$m_1 = \max\left(\text{rowmax}(Q_1 K_1^T), m_0\right)$$
$$S_1 = Q_1 K_1^T$$
$$\ell_1 = \text{rowsum}\left[\exp(S_1 - m_1)\right] + \ell_0 \cdot \exp(m_0 - m_1)$$
$$P_{11} = \exp(S_1 - m_1)$$
$$O_1 = \text{diag}\left(\exp(m_0 - m_1)\right) O_0 + P_{11} V_1$$

As you can see, we never normalize the "softmax" values. We will do it at the end.

STEP 2

$$m_2 = \max\left(\text{rowmax}(Q_1 K_2^T), m_1\right)$$
$$S_2 = Q_1 K_2^T$$
$$\ell_2 = \text{rowsum}\left[\exp(S_2 - m_2)\right] + \ell_1 \cdot \exp(m_1 - m_2)$$
$$P_{12} = \exp(S_2 - m_2)$$
$$O_1 = \text{diag}\left(\exp(m_1 - m_2)\right) O_1 + P_{12} V_2$$

AND SO ON UNTIL THE LAST STEP. THEN, WE APPLY THE "ℓ" NORMALIZATION FACTOR.

STEP 5

$$O_5 = \left[\text{diag}(\ell_4)\right]^{-1} O_4$$

Algorithm 1 FlashAttention-2 forward pass

Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM, block sizes B_c, B_r.

Divide $\mathbf{Q}$ into $T_r = \left\lceil \frac{N}{B_r} \right\rceil$ blocks $\mathbf{Q}_1, \dots, \mathbf{Q}_{T_r}$ of size $B_r \times d$ each, and divide $\mathbf{K}, \mathbf{V}$ in to $T_c = \left\lceil \frac{N}{B_c} \right\rceil$ blocks $\mathbf{K}_1, \dots, \mathbf{K}_{T_c}$ and $\mathbf{V}_1, \dots, \mathbf{V}_{T_c}$, of size $B_c \times d$ each.

Divide the output $\mathbf{O} \in \mathbb{R}^{N \times d}$ into T_r blocks $\mathbf{O}_i, \dots, \mathbf{O}_{T_r}$ of size $B_r \times d$ each, and divide the logsumexp L into T_r blocks $L_i, \dots, L_{T_r}$ of size B_r each.

for $1 \le i \le T_r$ **do** ← FOR EACH Qi BLOCK

  Load $\mathbf{Q}_i$ from HBM to on-chip SRAM.

  On chip, initialize $\mathbf{O}_i^{(0)} = (0)_{B_r \times d} \in \mathbb{R}^{B_r \times d}, \ell_i^{(0)} = (0)_{B_r} \in \mathbb{R}^{B_r}, m_i^{(0)} = (-\infty)_{B_r} \in \mathbb{R}^{B_r}$.

  for $1 \le j \le T_c$ **do** ← FOR EACH Kj BLOCK

    Load $\mathbf{K}_j, \mathbf{V}_j$ from HBM to on-chip SRAM.

    On chip, compute $\mathbf{S}_i^{(j)} = \mathbf{Q}_i \mathbf{K}_j^T \in \mathbb{R}^{B_r \times B_c}$.

    On chip, compute $m_i^{(j)} = \max(m_i^{(j-1)}, \mathrm{rowmax}(\mathbf{S}_i^{(j)})) \in \mathbb{R}^{B_r}$, $\tilde{\mathbf{P}}_i^{(j)} = \exp(\mathbf{S}_i^{(j)} - m_i^{(j)}) \in \mathbb{R}^{B_r \times B_c}$ (pointwise), $\ell_i^{(j)} = e^{m_i^{j-1} - m_i^{(j)}} \ell_i^{(j-1)} + \mathrm{rowsum}(\tilde{\mathbf{P}}_i^{(j)}) \in \mathbb{R}^{B_r}$.

    On chip, compute $\mathbf{O}_i^{(j)} = \mathrm{diag}(e^{m_i^{(j-1)} - m_i^{(j)}})^{-1} \mathbf{O}_i^{(j-1)} + \tilde{\mathbf{P}}_i^{(j)} \mathbf{V}_j$.

  end for

  On chip, compute $\mathbf{O}_i = \mathrm{diag}(\ell_i^{(T_c)})^{-1} \mathbf{O}_i^{(T_c)}$.

  On chip, compute $L_i = m_i^{(T_c)} + \log(\ell_i^{(T_c)})$.

  Write $\mathbf{O}_i$ to HBM as the i-th block of $\mathbf{O}$.

  Write L_i to HBM as the i-th block of L.

end for

Return the output $\mathbf{O}$ and the logsumexp L.