Closed-form Expressions for Systolic Array Latency and Utilization

Zhanhao Liu, Tram Nguyen, Penny Zhang, Tati Zhang

Fall 2021, Advisor: Bradley McDanel

Introduction

A systolic array is a special-purpose hardware architecture for high-level computations such as matrix multiplication. Data flows through the computational units of a systolic array in a regular fashion, which reduces computation time and increases efficiency. While this type of hardware architecture is commonly used in many commercial systems (e.g., Google TPU [1]), there is little work on closed-form expressions for its performance. Our research explores how to find closed-form expressions of latency and utilization of systolic arrays.

How do Systolic Arrays Work?

- Systolic arrays [2] are a type of hardware architecture that use many instances of a basic building block called a systolic cell to efficiently perform high-level computations like matrix multiplication.
- Figure 1 shows a 4x4 systolic array (16 systolic cells). Each systolic cell connects directly to its neighboring cells in order to perform the specific computation. Only the first row and last column of the systolic array are connected directly to memory, while the remaining cells take input from previous cells.
- Figure 2 shows an example of a 2×2 matrix entering a 2×2 systolic array. The weight matrix must be loaded into the systolic array ahead of time before the matrix multiplication is performed.
- Systolic array's strength lies in its extremely parallel nature. For large matrices, each cell performs useful work each time unit.

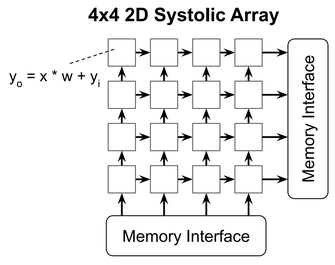

Figure 1:A 4×4 systolic array.

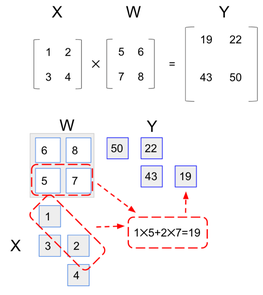

Figure 2:(A high-level view of systolic array computation.

Computing Matrix Multiplication using Systolic Array

- Each systolic cell in the systolic array is called a MAC (multiply accumulate) unit as shown in Figure 3. Each MAC takes in x and y as input and computes $y = w \times x + y$.
- The newly calculated y is served as the input y for the MAC on the right and x is unit for the MAC above. Note that w is stored in the MAC unit before the calculation begins.
- Figure 4 shows how the two matrices interact with each other in the systolic array. B represents the weight matrix, which is preloaded into the systolic array and each element stays in their respective cell. Meanwhile, A represents the data matrix that flows into the array from below, interacting with the B elements and produce the result matrix C.

- Figure 5 illustrates how the data matrix enter the systolic array in a skewed fashion. This action is necessary to maintain data synchronization within the systolic array, as the partial results from the previous cycle are needed in the next cycle in neighboring cells.

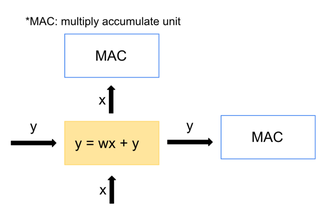

Figure 3:Multiplication and accumulation in a single systolic cell.

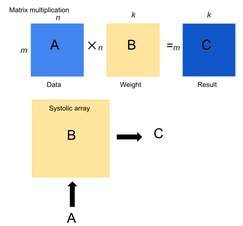

Figure 4:Multiplication multiplication performed in a systolic array.

Computing the Latency

- Latency refers to the number of time units, called cycles, that it takes for the systolic array to compute the complete result matrix.
- We analyze the latency for an $M \times K$ data matrix flowing through a $K \times K$ systolic array and producing a $K \times M$ result matrix.

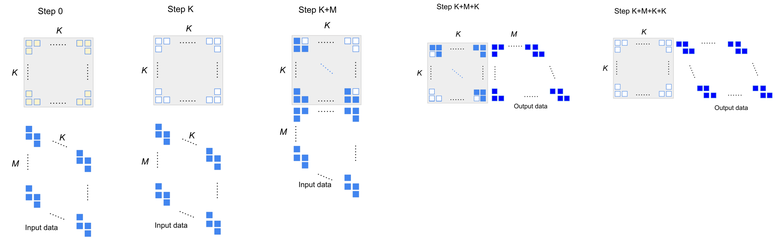

Figure 5:Matrix multiplication in systolic array.

Determining the latency can be decomposed into several steps that are depicted in Figure 5:

1. **(Step K) Preload the systolic array**: It takes K cycles to store the weight matrix in the systolic array (one of the K rows per cycle).
2. **(Step K+M) Top-left corner reached**: Since the data matrix is input in a skewed fashion, there is a point when data reaches the top-left corner of the systolic array. This step takes K cycles as the data matrix shifts K units upwards.
3. **(Step K+M+K) Last data enters systolic array**: The data continues to be shifted up to a stage where the last data enters the bottom-right systolic cell. This step takes M cycles.
4. **(Step K+M+K+K) Result matrix leaves the systolic array**: The partial result matrix is moved from the left to right until it is finally output. Since this movement proceeds diagonally as depicted in Introduction, this step also takes K step.

Thus, the latency is $M + 3K$ cycles.

Computing the Utilization

- The instantaneous utilization of systolic array is the number cells doing useful work (e.g., a MAC operations) over the total number of cells (e.g., $K \times K$ cycles).
- The average utilization of a systolic array is the average of instantaneous utilization across all steps in the computation.

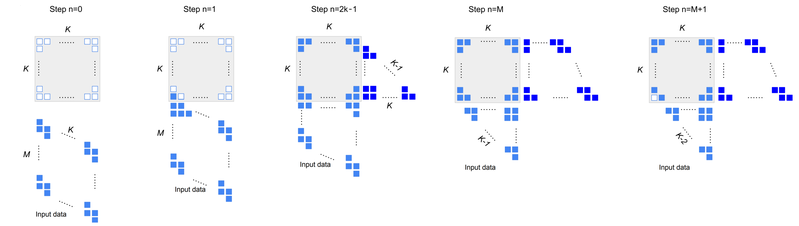

Figure 6:Systolic array utilization.

Since an $M \times K$ data matrix enters the systolic array in a skewed fashion, there are wasted MAC operations before the element the (K, K) data element enters the systolic array as shown in Figure 6. Additionally, there is waste after data element $(0, M)$ exits the systolic array. Thus, we can use the number of wasted MACs in each cycle to compute the utilization as follows:

$$
\begin{aligned}
W(1) &= \frac{((K-1)\cdot K) + (1\cdot(K-1))}{K^2} = \frac{K^2-1}{K^2} = W(0) - \frac{1}{K^2} \\
W(2) &= \frac{((K-2)\cdot K) + (1\cdot(K-1)) + (1\cdot(K-2))}{K^2} = \frac{K^2-3}{K^2} = W(1) - \frac{2}{K^2} \\
&\vdots \\
W(K) &= \frac{((K-K)\cdot K) + (1\cdot(K-1)) + (1\cdot(K-2)) + \cdots + (1\cdot(1))}{K^2} = W(K-1) - \frac{K}{K^2} \\
&\vdots \\
W(2K-2) &= \frac{(0\cdot K) + (0\cdot(K-1)) + (0\cdot(K-2)) + \cdots + (1\cdot(1))}{K^2} = \frac{1}{K^2}
\end{aligned}
$$

giving us the instantaneous waste function $W(n) = W(n-1) - \frac{n}{K^2}$. Since the waste area for the first $2K-1$ step and the final $2K-1$ step are the same, the total waste is $2\Sigma_{i=0}^{2K-2} W(i) = 2\frac{2K-2}{2} = 2K-2$. Therefore, the the total utilization U is: $\frac{MK}{2K-2+MK}$

Conclusion

Since we derived closed-form expressions for the latency and the utilization, we can directly calculate the efficiency of a systolic array. These calculations are typically computed in a systolic array simulator. Thus, we hope this is useful for more efficiently scheduling many matrix multiplications in sequence, as the latency determines how long each multiplication will take.

References

[1] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, ``A domain-specific supercomputer for training deep neural networks,'' *Communications of the ACM*, vol. 63, no. 7, pp. 67--78, 2020.

[2] H.-T. Kung, ``Why systolic architectures?'' *IEEE Computer*, vol. 1, p. 5, 1998.