

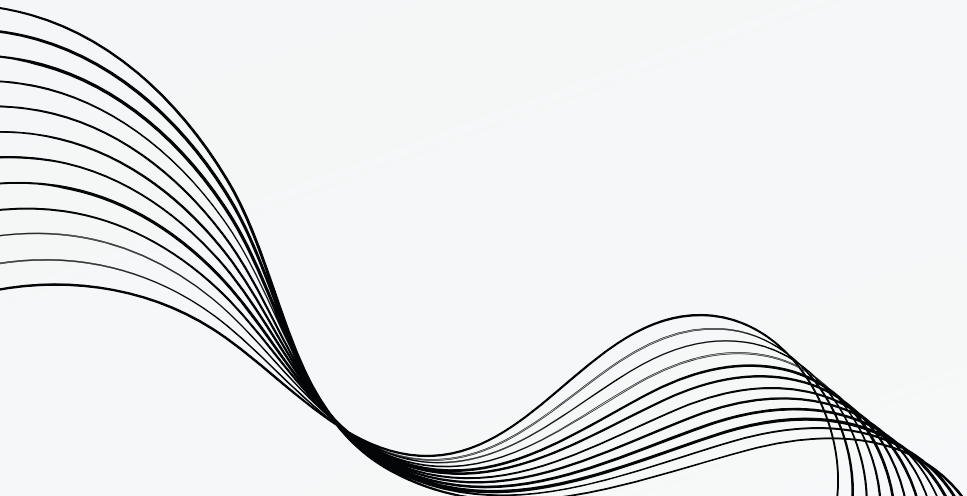


A SYSTOLIC ARRAY-BASED SCHEDULING STRATEGY FOR SPARSE CNN ACCELERATORS



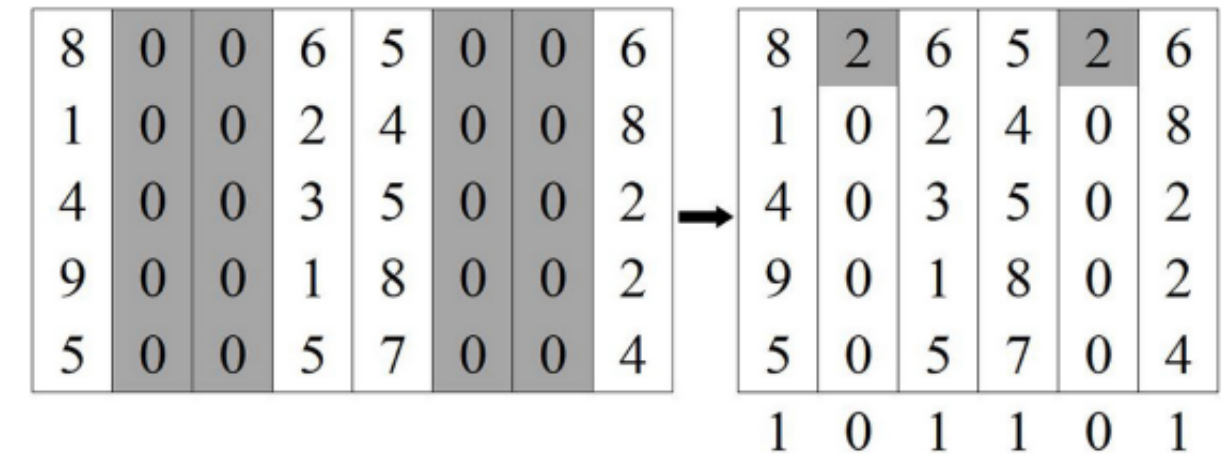
KEY CONTRIBUTIONS TO THE DESIGN

1. A **vector transmission coding design** which has the characteristics of the interval distribution of the effective vector and the invalid zero-valued vector.
2. A **scheduling module** that enables **register array**-based effective data scheduling to achieve data reuse in the process of convolution window sliding computation. Additionally, the PE array does not need to store feature map data and convolution kernel data, which is provided by the previous column computing unit or external input.
3. A general **vector scheduling method** for different sparsely distributed convolution kernel weights and a weight data pre-ordering algorithm designed to improve the problem of computing arrays waiting for each other



TRANSMISSION CODING DESIGN :

- This method encodes and transmits data as vectors, effectively compressing the data while maintaining the distribution characteristics of valid and zero-valued vectors.
- The encoding uses run-length encoding to compress consecutive zero vectors, adding a 1-bit wide flag to indicate whether a vector is valid or not.
- Example: If two consecutive columns of vectors are zeros, they are merged, and the flag bit for the vector is set to 0, with the value (invalid_num) representing the count of consecutive zeros.
- Upon receiving the input vector, the array calculates the column address of the vector based on the row address of the transmitted column vector, the number of valid vectors, and the invalid_num value of the invalid vector



8	0	0	6	5	0	0	6
1	0	0	2	4	0	0	8
4	0	0	3	5	0	0	2
9	0	0	1	8	0	0	2
5	0	0	5	7	0	0	4

8	2	6	5	2	6
1	0	2	4	0	8
4	0	3	5	0	2
9	0	1	8	0	2
5	0	5	7	0	4

1	0	1	1	0	1
---	---	---	---	---	---

EFFECTIVE SCHEDULING METHOD :

- Utilizes a register array outside the computation array input to facilitate data reuse and reduce memory access costs.
- The scheduling control module provides index addresses to the register array, allowing the processing element (PE) array to read effective feature map vectors for computation.
- A state machine helps manage the scheduling process, ensuring that only valid data is processed by the PE array.
- Example: In a sliding convolution window, if a feature map vector has valid data at specific intervals, the control module updates the register array only at these intervals, reducing unnecessary computations.
- The design is based on a 3×3 convolution kernel. But when the kernel size increases, the states corresponding to the effective vector scheduling method becomes more complex. Therefore, to make the control scheduling method more universal, we need to split the vectors of 5×5 or larger. For example, a 5×5 convolution kernel is split into 5×3 and 5×2 convolution kernels, thus enabling scheduling for different kernel sizes

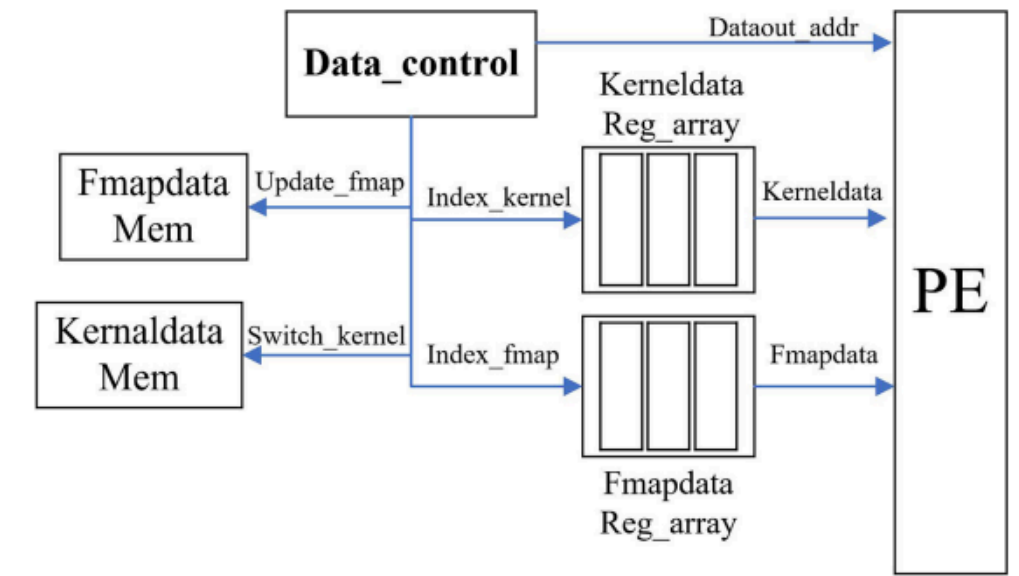
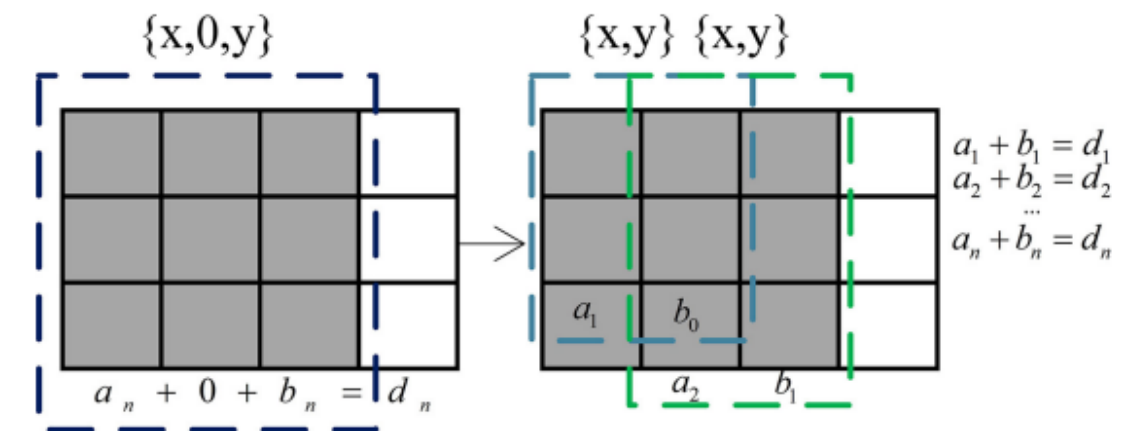


Fig. 2. Interaction between scheduling control module, PE array and storage module.



SPARSE CONVOLUTION KERNEL SCHEDULING DESIGN:

- Matches convolution kernel weight data scheduling to the feature map data unit.
- Uses a general vector scheduling method to handle different sparse distributions of convolution kernel weights.
- Pre-orders convolution kernel weight data to ensure that PE arrays share a unified scheduling mechanism, avoiding idle waiting times.
- Example: For a 3x3 convolution kernel with different sparse distributions, the method ensures that only the valid parts of the kernel are processed, skipping zero-valued vectors to optimize computation.

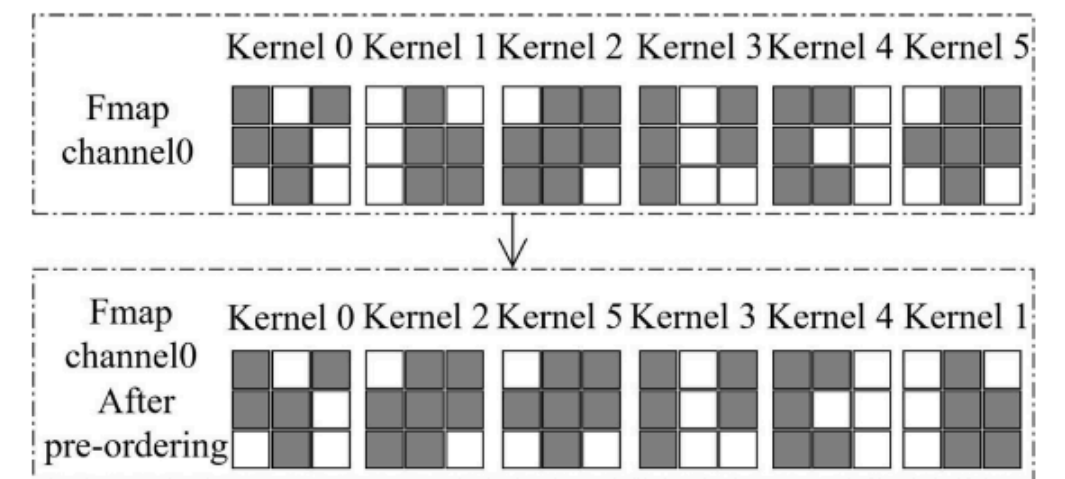


Fig. 4. Convolution kernel weights with different sparse vector distribution.

RESULTS

- PE Utilization: Employing kernel optimization on top of feature map optimization significantly improved PE utilization, with a maximum rate of 83.58% and an average rate of 65.34%, compared to an unoptimized rate of 13.96%.
- Speedup Ratio: For convolutional layers conv1_1 to conv3_3, the scheduling optimization scheme showed significant improvement. However, from conv4_1 to conv5_3, the optimization effect was relatively low. This is because the scheduling method is based on index scheduling for input efficient vectors, leading to increased invalid calculations due to higher zero-valued elements in smaller feature maps.

Impact of Feature Map Size and Sparsity

- Observations:
- Larger input feature map sizes and higher sparsity in feature maps and convolutional kernel weights lead to better optimization in terms of reducing computation cycles and improving computation unit utilization.

RESULTS

TABLE II
SPEEDUP AND PE UTILIZATION OF VGG-16 CONVOLUTIONAL LAYER UNOPTIMIZED VS. OPTIMIZED GROUPS

Layer	Sparsity		Speedup		PE utilization		
	Weight sparsity	Feature map sparsity	Feature map optimization	All optimization	Unoptimized	Feature map optimization	All optimization
conv1_1	41%	73%	3.27	3.59	18.90%	62.46%	83.58%
conv1_2	78%	72%	3.22	3.95	10.20%	26.90%	73.56%
conv2_1	89%	69%	2.86	3.65	6.40%	14.46%	62.38%
conv2_2	81%	68%	2.82	3.50	10.10%	22.15%	66.46%
conv3_1	74%	64%	2.27	2.86	12.40%	28.80%	65.32%
conv3_2	78%	63%	2.24	2.87	13.10%	25.26%	64.10%
conv3_3	75%	64%	2.25	2.86	11.00%	29.80%	64.24%
conv4_1	69%	59%	1.65	1.97	13.70%	31.57%	58.71%
conv4_2	71%	56%	1.61	1.93	14.80%	28.61%	63.34%
conv4_3	69%	52%	1.51	1.80	15.90%	29.87%	63.75%
conv5_1	66%	56%	1.33	1.58	17.00%	33.14%	62.26%
conv5_2	62%	53%	1.30	1.51	17.90%	35.94%	61.32%
conv5_3	64%	52%	1.26	1.48	20.30%	31.85%	60.42%
Average	71%	62%	2.12	2.58	13.98%	30.83%	65.34%

KEY TAKEAWAYS

- The proposed scheduling strategy significantly improves PE utilization and convolution calculation speed when both feature map sparse vectors and convolution kernel weights are optimized.
- The optimization is most effective for larger feature maps with high sparsity.
- The design offers a power-efficient solution for sparse convolution calculations, outperforming previous implementations like Cambricon-X in terms of power consumption.

